# Grafotopia: introdução ao pensamento algorítmico e à depuração nos primeiros anos do Ensino Fundamental

**Rafael de Mattia**[1], **Igor Basílio Valerão**[1],
**Luciana Foss**[1], **Simone André da Costa Cavalheiro**[1]

[1]Centro de Desenvolvimento Tecnológico - Universidade Federal de Pelotas (UFPEL)
CEP 96010-610 - Pelotas - RS - Brasil

{rdmattia,ibvalerao,lfoss,simone.costa}@inf.ufpel.edu.br

***Abstract.*** *With the approval of regulations regarding the teaching of Computer Science in Basic Education, numerous challenges emerge, one of which is the need for instructional support to effectively meet the established learning objectives. This research paper introduces an activity designed to facilitate the introduction of algorithmic thinking and debugging skills in the early years of elementary education. Named Grafotopia, this proposal was developed using a game engine based on Graph Grammars.*

***Resumo.*** *Com a aprovação das normas sobre o ensino de Computação na Educação Básica, diversos desafios se apresentam, entre os quais a necessidade de suporte instrucional para a promoção dos objetivos de aprendizagem estabelecidos. O presente trabalho propõe uma atividade que objetiva auxiliar a introdução do pensamento algorítmico e da depuração nos primeiros anos do ensino fundamental. Denominada Grafotopia, a proposta foi desenvolvida em um motor de jogos baseado em Gramáticas de Grafos.*

## 1. Introdução

O Pensamento Computacional (PC) engloba um leque de ferramentas mentais que capacitam indivíduos a resolver problemas complexos e, embora seja fundamentado no vasto campo da Ciência da Computação, suas aplicações vão além deste domínio específico. Essas ferramentas fornecem uma abordagem abrangente para enfrentar desafios de diversas áreas do conhecimento [Wing 2006]. Wing (2006) descreve o PC como uma abordagem para solucionar problemas, projetar sistemas e compreender o comportamento humano. Ela considera o PC uma habilidade fundamental para qualquer pessoa.

Em 2022, foram aprovadas as normas para o ensino de Computação na Educação Básica no Brasil [BRASIL 2022], nas quais o PC consta como um eixo para o Ensino Fundamental. Desde então, diversos desafios se apresentam, entre os quais a necessidade de suporte instrucional para a promoção dos objetivos de aprendizagem estabelecidos. O pensamento algorítmico (PA), objeto de estudo deste trabalho, engloba o uso da lógica para desenvolver soluções. Na BNCC, o PA deve ser trabalhado ao longo de toda educação básica. Nos anos iniciais do ensino fundamental, a ênfase é dada à introdução de algoritmos e das estruturas de controle, enquanto nos anos finais foca-se em habilidades relacionadas à programação.

Diversos trabalhos que focam no desenvolvimento do PA com depuração para a Ensino Fundamental utilizam a plataforma Scracth [Wong et al. 2024, Lee et al. 2014],

recursos do Code.org [Holenko Dlab et al. 2019, Ali and ÇAKIR 2021] e/ou jogos específicos [Rose 2019]. Esses trabalhos relatam resultados positivos no aprimoramento das habilidades relacionadas. Em contraste, a abordagem proposta neste trabalho, trata-se de uma abordagem não baseada em blocos. Grafotopia propõe o uso de grafos (diagramas) para a descrição de algoritmos. Diagramas são abstrações bastante utilizadas para descrever processos, deixando explícito o fluxo de execução deles, tornando o seu entendimento mais fácil, quando comparados com uma descrição estruturada (com delimitadores de blocos de instruções). Levando em conta este cenário, este artigo apresenta uma atividade desenvolvida na plataforma GrameStation [Junior et al. 2021], um motor de jogos que permite criar e executar jogos baseados em Gramática de Grafos, para promover a aprendizagem de conceitos relacionados ao PA nos anos iniciais do Ensino Fundamental.

Gramática de Grafos (GG) [Rozenberg 1997] é uma linguagem de especificação formal que usa grafos para descrever os estados de um sistema e regras de transformação de grafos para representar as mudanças de estados. Por ser uma linguagem de especificação, ela permite abstrair detalhes de implementação que são necessários em linguagens de programação. A representação dos jogos e suas regras através de grafos é bastante intuitiva até mesmo para crianças mais jovens [Junior et al. 2017]. Além disso, da Silva Junior et al. (2019) e (2020) destaca uma série de relações entre o desenvolvimento de habilidades do PC e a manipulação de GG. Assim, ao usar o Gramestation para realizar a atividade proposta, os estudantes, além de trabalhar conceitos do PA, têm o potencial para desenvolver diferentes habilidades do PC.

A atividade é constituída de uma série de desafios que são resolvidos com a criação e execução de algoritmos descritos por grafos dirigidos. A execução de um algoritmo pode ser realizada passo a passo, facilitando a depuração do mesmo. Além disso, por se tratar de uma atividade criada e executada na mesma plataforma, pode ser facilmente adaptada e estendida de acordo com os objetivos de ensino do momento. O restante do texto está organizado como segue. A Seção 2 introduz a fundamentação teórica deste trabalho, o PC e a plataforma GrameStation. A Seção 3 descreve a proposta da atividade. Por fim, na Seção 4 apresentam-se as considerações finais.

## 2. Fundamentação Teórica

Esta seção apresenta as habilidades do PC consideradas neste trabalho, na subseção 2.1, bem como introduz a plataforma GrameStation e suas funcionalidades, na subseção 2.2.

### 2.1. Pensamento Computacional

O PC inclui habilidades como [Csizmadia et al. 2015]:

- Abstração: envolve a capacidade de identificar os aspectos importantes de um problema e ignorar detalhes irrelevantes, permitindo uma compreensão mais clara e simplificada;
- Decomposição: consiste em dividir um problema complexo em partes menores e mais gerenciáveis, facilitando a análise e a resolução eficiente;
- Pensamento algorítmico: refere-se à habilidade de resolver problemas e realizar tarefas através da definição bem definida de passos;
- Generalização: implica na capacidade de identificar padrões e tendências em diferentes situações, permitindo a aplicação de soluções semelhantes a problemas semelhantes;

- Avaliação: envolve a capacidade de testar, analisar e corrigir erros em algoritmos e soluções computacionais, promovendo a resolução de problemas de forma eficaz.

O PC influencia diretamente a compreensão dos processos para a criação de uma solução e, com este entendimento, é possível realizar uma análise detalhada das etapas do desenvolvimento [Wing 2006]. A depuração é uma das formas de realizar essa análise permitindo a identificação de erros e a compreensão dos comportamentos do sistema. Segundo Soloway and Ehrlich (1984), a depuração promove um amadurecimento na curva de aprendizagem de habilidades cognitivas, proporcionando aos desenvolvedores uma compreensão mais profunda dos algoritmos e estruturas de dados. Esse processo enriquece a experiência de aprendizado, desenvolvendo habilidades de persistência e aquisição de conhecimentos a partir dos erros. Além disso, aprimora o reconhecimento de padrões e o raciocínio algorítmico, essenciais para abordar desafios de forma sistemática e organizada [Wong and Jiang 2018].

## 2.2. GrameStation

O GrameStation (GS) [da Silva Junior et al. 2021] é um motor de jogos baseado em gramáticas de grafos tipadas (GG) usado para criar e executar jogos. Os jogos são descritos através de três componentes que definem uma GG: um grafo tipo, um grafo inicial e um conjunto de regras. Da Silva Junior et al. (2020) detalha como os conceitos de representação de dados, decomposição de problemas, abstração, pensamento algorítmico e paralelismo são desenvolvidos através do design e execução de um jogo. Desta forma, a representação de jogos como GG também promove o desenvolvimento de competências relacionadas ao PC. Essas habilidades são desenvolvidas tanto pela pessoa que cria um jogo (especifica uma GG) quanto pela pessoa que executa um jogo (simula uma GG).

GrameStation é separado em três módulos: *Builder*, *Player* e *Explorer*, que permitem criar, executar e encontrar os jogos, respectivamente. Conforme mencionado por [da Silva et al. 2021], o primeiro passo para especificar uma gramática no GS é importar os recursos externos (como imagens) a serem incluídos no jogo, e então criar o grafo tipo, que desempenha na plataforma a função de uma área de declaração. A próxima etapa corresponde à criação do grafo (estado) inicial que mostre a organização inicial do jogo e, finalmente, à criação de regras que definam as ações do jogo. Uma regra é composta por um grafo esquerdo (LHS, *left-hand side*), indicando os itens que devem estar presentes no estado para que ela seja aplicada, um grafo direito (RHS, *right-hand side*), detalhando o efeito da regra e um mapeamento (homomorfismo de grafos). Os elementos que estão no LHS e não são mapeados para o RHS devem ser deletados, os elementos mapeados são preservados e os que estão somente no RHS são criados. Uma regra está habilitada a ser aplicada, se houver um *match* do LHS da regra com o grafo estado atual, isto é, o LHS aparece como um subgrafo do grafo estado. No modo de execução das atividades do GS, os jogadores se deparam com o grafo estado e com um botão para abrir a seleção de regras. A única forma de alterar esse grafo é fazendo uso das regras disponíveis na atividade.

O GS possui um grafo transparente aos jogadores, chamado de Grafo Piloto, que permite definir a quantidade de jogadores, habilitar e desabilitar o acesso dos jogadores a fases e regras, além de possuir ferramentas específicas, como os *Watchers*, que são capazes de aplicar regras de forma automática, testando o *match* a cada tempo preestabelecido.

Com isso, é possível que regras sejam aplicadas durante a atividade sem que o jogador tenha acesso diretamente a elas. Esta funcionalidade pode ser utilizada de diferentes formas, e no Grafotopia foi utilizada para completar de forma automática as ações selecionadas pelo jogador, minimizando a complexidade da atividade. Além disso, essa funcionalidade permitiu a generalização das regras do jogo, diminuindo consideravelmente a quantidade de regras.

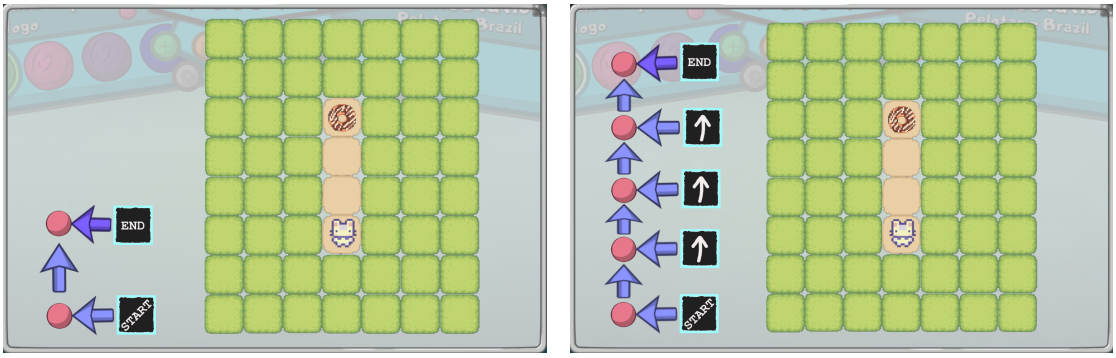
Essas características possibilitam tanto a criação quanto a jogabilidade dos jogos, mesmo para aqueles com conhecimento limitado sobre GG [Junior et al. 2017]. Isso é especialmente importante ao considerar o público-alvo dessa atividade, ou seja, alunos do ensino fundamental nos anos iniciais. Como os jogos têm o propósito de trabalhar conceitos de PC, a capacidade de abstrair a complexidade das gramáticas, mesmo que de forma implícita, torna o GS ainda mais interessante.

## 3. Atividade Grafotopia

A atividade Grafotopia foi criada com o objetivo de desenvolver habilidades relacionadas ao pensamento algorítmico e a avaliação (depuração), voltada para estudantes dos anos iniciais do Ensino Fundamental. Inicialmente, o jogador deve guiar o protagonista até seu destino, utilizando uma sequência de movimentos previamente determinada pelo próprio jogador. Os movimentos disponíveis incluem avançar para cima, avançar para a esquerda, avançar para a direita e avançar para baixo. A sequência de movimentos é descrita por um grafo, no qual os movimentos são representados por vértices conectados por arestas direcionadas que indicam a ordem destes movimentos.

À esquerda da Figura 1(a) é ilustrado o tabuleiro que descreve o desafio da primeira fase: mover o personagem até o "donut", avançando três vezes para cima. Ainda nessa figura, à esquerda, está ilustrado o grafo de movimentos inicial. No grafo de movimento cada vértice de movimento está associado a um vértice de posição (círculo vermelho), que representa uma posição de uma lista. Os vértices de posição são utilizados para permitir a generalização das regras que manipulam o grafo de movimentos, reduzindo a quantidade de regras a serem utilizadas pelo jogador. Inicialmente, o grafo de movimentos possui apenas os vértices START e END, delimitando o início e o final da sequência, respectivamente, conectados aos seus correspondentes vértices de posição. Observa-se que ainda não há nenhum movimento definido. Já a Figura 1(b) ilustra o grafo de movimentos com a sequência dos movimentos a serem realizados para completar o primeiro desafio. Esse grafo possui cinco vértices de posição, três vértices de movimento PARA_CIMA, um vértice START e um vértice END.

Os movimentos para cumprir o desafio devem ser adicionados através da aplicação de regras. Dentre essas regras, existem quatro regras que o jogador usa para adicionar os vértices de movimento na sequência de movimentos, porém elas não reorganizam a lista. Para reorganizar a lista existe uma regra transparente ao usuário que é disparada de forma automática (sempre que existir um *match* no grafo estado para ela). A definição destas regras automáticas também auxilia na redução da quantidade de regras para o jogador. Na Figura 2 são apresentadas as regras que podem ser utilizadas pelo jogador para adicionar movimentos no grafo de movimentos. A primeira regra, denominada `vaParaCima`, por exemplo, permite adicionar o movimento de avançar para cima criando uma nova posição na lista (vértice com *loop* lilás), anterior ao de ponto de inserção (vértice com *loop*




(a) Grafo inicial. (b) Grafo de movimentos com a solução.

**Figura 1. Grafos da primeira fase da atividade.**

amarelo). O ponto de inserção deve ser diferente do vértice de posição do START, pois não se pode inserir uma instrução antes do START. Além desta, o usuário possui a opção de utilizar as demais regras inserção de movimento. Todas definem as mesmas alterações do grafo de movimentos, variando apenas no movimento a ser inserido. Para aplicar uma regra o usuário apenas seleciona o ponto de inserção (posição onde o movimento deve ser adicionado). Após a aplicação de uma regra de inserção de movimento no grafo, a regra `reorganiza` é disparada automaticamente para refazer as conexões da lista, acomodando o vértice de movimento criado. Na Figura 3, é ilustrada essa regra, na qual a aresta que chega no ponto de inserção (com *loop* amarelo) deve ser redirecionada para o vértice de posição criado (com *loop* lilás). Existe ainda um conjunto de regras que permitem o jogador excluir vértices de movimentos, que são seguidas pela aplicação de regras automáticas que reorganizam a lista de posições.

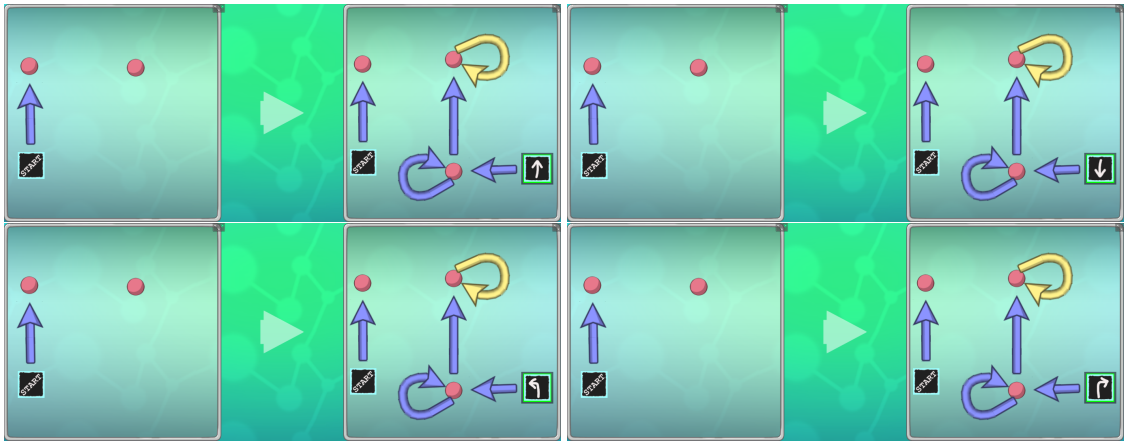


**Figura 2. Regras do jogador para inserção de movimentos.**

A configuração das regras automáticas é feita no grafo piloto. A Figura 4 ilustra o grafo piloto da Grafotopia. Neste grafo, as regras do jogo são representadas pelos vértices lilás em formato de planeta. As regras automáticas habilitadas no início são aquelas conectadas com *Watchers* (vértices em forma de despertador amarelo indicando o intervalo de tempo para tentativa de aplicação) por uma aresta ciano. Ainda neste grafo temos outras configurações definidas: o vértice Core (vértice em forma de engrenagem cinza), que aponta para os jogadores e para a fase corrente; as fases do jogo (vértices

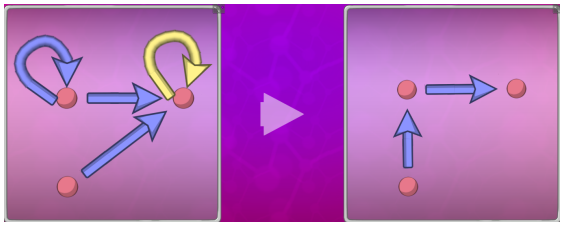


**Figura 3. Regra automática para reorganização do grafo de movimentos.**

em forma de planeta cor de rosa), sendo que a fase inicial está conectada ao Core; os jogadores (vértices em forma de pessoas brancas), sendo que neste caso temos apenas um jogador; as regras disponíveis para o jogador, que estão conectadas a ele por arestas na cor lilás; as regras que serão habilitadas em algum momento do jogo (vértices de regras desconectados).

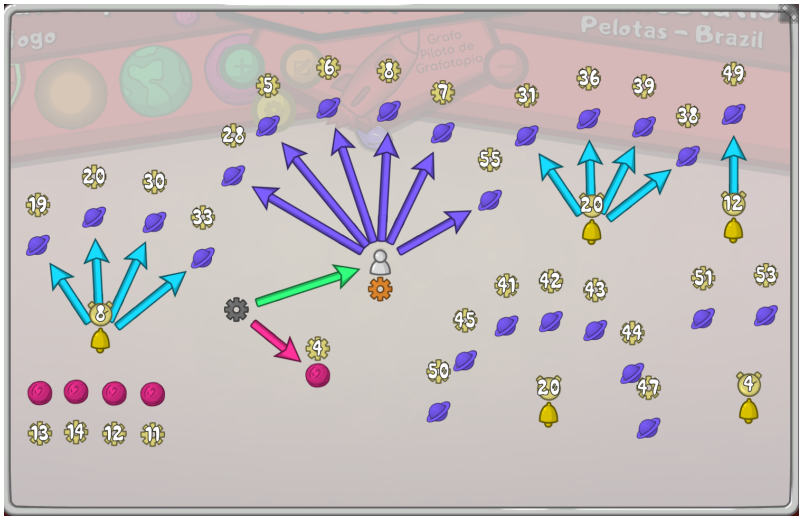



**Figura 4. Grafo piloto.**

Depois de criado o grafo de movimentos que soluciona o desafio, ele pode ser executado a partir da aplicação da regra `Run`, ilustrada na Figura 5. À esquerda dessa figura, é mostrado o efeito da regra no grafo estado: cria um cursor (vértice verde com seta branca) conectado ao vértice de posição do START, indicando o movimento corrente (que deve ser executado). Além disso, a regra `Run` altera o grafo piloto (efeito ilustrado no lado direito da Figura 5), apagando sua conexão com o jogador (impedindo que o jogador consiga aplicá-la mais de uma vez) e conectando ao *Watcher* as regras que disparam a movimentação automática do personagem pelo tabuleiro, simulando a execução de cada instrução no grafo de movimentos.

As casas do tabuleiro são logicamente interligadas por arestas que as conectam a vértices intermediários, os vértices de ligação. Na Figura 6 são representadas essas conexões. Os vértices de ligação e as arestas que os conectam ficam ocultos sob as casas e têm a função de permitir a generalização das regras de movimentação. As adjacências das casas são definidas por arestas de diferentes cores entre as casas e os vértices de ligação. Uma aresta amarela indica o vizinho à esquerda, uma aresta rosa define o vizinho à direta, uma aresta azul aponta para o vizinho acima e uma aresta laranja indica o vizinho de

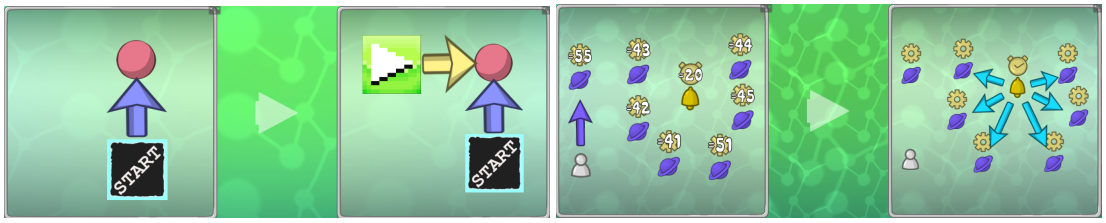



**Figura 5. Regra `Run`: efeitos no grafo estado (à esq.) e no grafo piloto (à dir.).**

baixo. Já as arestas verdes e vermelhas, definem que o vizinho é um caminho (quadrado bege) ou uma parede (quadrado verde), respectivamente.

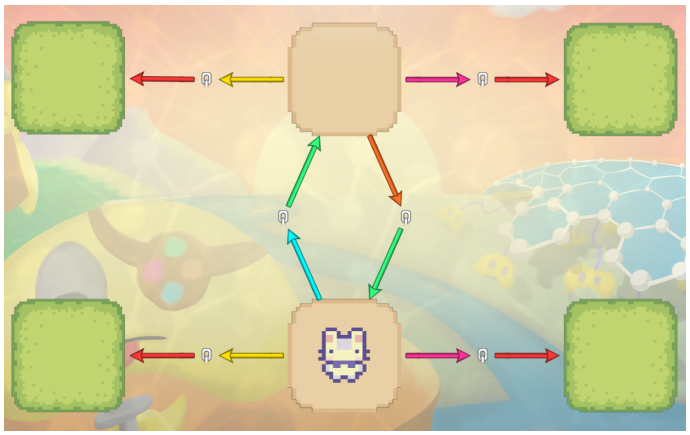


**Figura 6. Representação da lógica das conexões do tabuleiro.**

A movimentação do personagem é realizada pela aplicação de 3 regras automáticas: (1) Regra `proxMovimento`, Figura 7 – o cursor verde é substituído por um cursor cinza, conectado à próxima posição do grafo de movimentos, o que habilita as regras que movimentam o personagem; (2) Regra `moveLigamentoX`, exemplo na Figura 8 (à esq.) – o personagem é movido para o vértice de ligação na direção definida pelo movimento corrente (X) e o cursor cinza é substituído pelo cursor verde, habilitando a regra `proxMovimento` novamente; (3) Regra `moveCaminho`, Figura 8 (à dir.) – o personagem é movido de um vértice de ligação para o caminho a ele conectado. Existe uma regra análoga a essa última tratando o caso em que há uma parede conectada ao vértice de ligação. Neste caso, uma regra que indica a derrota é habilitada no grafo piloto e ela reinicia a atividade.

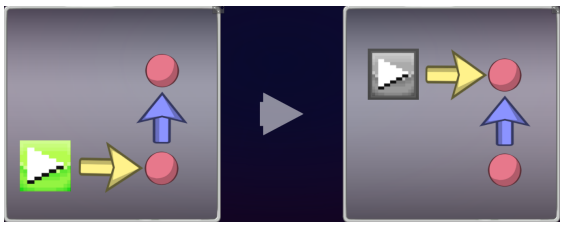


**Figura 7. Regra `proxMovimento`.**

Caso o personagem se conecte a um vértice de caminho, a sequência de movimentos continua a ser executada até que chegue ao fim da sequência ou alcance o seu objetivo,

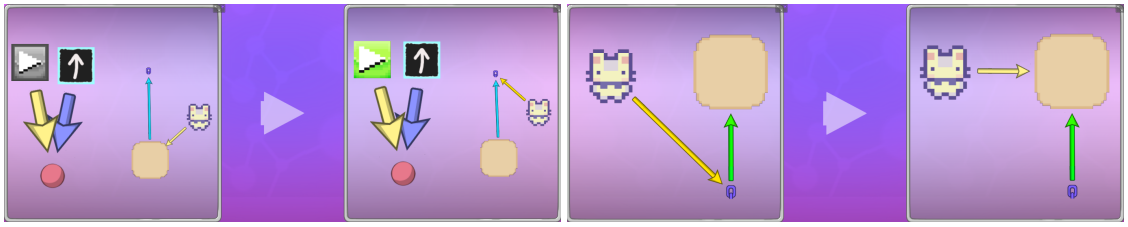


**Figura 8. Regras `moveLigamentoCima` (à esq.) e `moveCaminho` (à dir.).**

que é estar conectado na mesma casa que o donut. Neste último caso o jogador completa seu desafio e passa para a próxima fase. A cada fase que o jogador avança, o caminho a ser percorrido para atingir o objetivo é mais complexo. Na Figura 9 são ilustrados os desafios da segunda e da terceira fases.

Apesar de existirem diversas regras, o jogador não tem acesso às regras automáticas. Dessa forma, o jogador não precisa se preocupar como os movimentos são selecionados para serem executados e nem como ocorre toda a movimentação no tabuleiro, ele pode focar apenas em alcançar o objetivo, dando uma maior ênfase ao pensamento algorítmico.

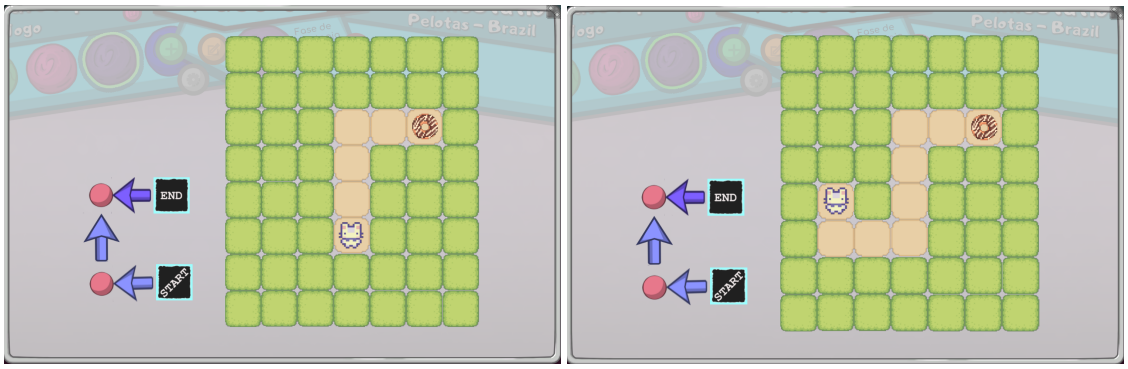



**Figura 9. Grafos iniciais da segunda (à esq.) e terceira (à dir.) fases.**

A atividade Grafotopia admite também um modo de execução das regras passo a passo, para que uma análise, por meio da depuração, torne-se factível. Neste caso, a regra `Passo` deve ser aplicada. Ela tem o mesmo efeito da regra `Run` além de tornar a regra `proxMovimento` acessível ao jogador (não sendo mais automática). Assim como `Run` ela também só pode ser aplicada uma única vez. Nas primeiras fases, a atividade envolve a criação de algoritmos, enquanto nas fases subsequentes a ênfase é dada à depuração. Nestes casos, o jogador recebe o algoritmo e deve descobrir se a solução está correta. Caso não esteja, deve corrigir o algoritmo dado.

## 4. Conclusão

Este trabalho propõe a atividade Grafotopia que tem como objetivo auxiliar a introdução do pensamento algorítmico e da depuração nos primeiros anos do Ensino Fundamental. Ela permite a criação de algoritmos representados por grafos e a execução deles através da aplicação de regras de transformação de grafos. Nessa atividade a simulação dos algoritmos pode ser feita de forma automática ou ainda passo a passo, possibilitando a depuração

desses algoritmos. As regras que realizam a execução das instruções (movimentos) e as movimentações do personagem são transparentes ao jogador, que deve focar apenas na criação da sequência correta de movimentos que devem ser realizados para se atingir o objetivo de cada fase.

Essa atividade permite ainda sua adaptação e extensão visto que foi criada em uma plataforma aberta que possibilita, além da sua execução, a edição de suas fases e regras.

Como trabalhos futuros, pretende-se estender a atividade para incluir laços de repetição e condicionais para estruturar os fluxos de execução. Além disso, planeja-se aplicar a atividade Grafotopia em turmas dos anos iniciais do Ensino Fundamental para avaliar sua adequação.

## Referências


Ali, O. and ÇAKIR, R. (2021). The effect of code. org activities on computational thinking and algorithm development skills. *Journal of Teacher Education and Lifelong Learning*, 3(2):32–40.

BRASIL (2022). Base nacional comum curricular: Computação - complemento à bncc. Disponível em `http://portal.mec.gov.br/index.php?option=com_docman&view=download&alias=236791-anexo-ao-parecer-cneceb-n-2-2022-bncc-computacao&category_slug=fevereiro-2022-pdf&Itemid=30192`. Acessado em julho de 2022.

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., and Woollard, J. (2015). Computational thinking - A guide for teachers. Guide, Computing at School. `https://eprints.soton.ac.uk/424545/1/150818_Computational_Thinking_1_.pdf`.

da Silva, J. V., da Silva Junior, B. A., Foss, L., and da Costa Cavalheiro, S. A. (2021). Adaptaçao do processo engaged para o desenvolvimento de conteudos curriculares em uma plataforma de jogos baseada em gramática de grafos. In *Anais do XXXII Simpósio Brasileiro de Informática na Educação*, pages 316–327. SBC.

da Silva Junior, B. A., da Costa Cavalheiro, S. A., and Foss, L. (2020). GGasCT: Bringing formal methods to the computational thinking. In *Anais dos Workshops do IX Congresso Brasileiro de Informática na Educação*, pages 83–83. SBC.

da Silva Junior, B. A., da Costa Cavalheiro, S. A., and Foss, L. (2021). Gramestation: Specifying games with graphs. In *Anais do XXXII Simpósio Brasileiro de Informática na Educação*, pages 499–511. SBC.

Holenko Dlab, M., Hoic-Bozic, N., Andelic, M., and Boticki, I. (2019). Digital games and tools for development of computational thinking in primary school. In *Proceedings of the international conference on management, economics & social science-ICMESS*, pages 1–7.

Junior, B., Cavalheiro, S., and Foss, L. (2017). A última árvore: exercitando o pensamento computacional por meio de um jogo educacional baseado em gramática de grafos. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 28, page 735.

Junior, B., Cavalheiro, S., and Foss, L. (2019). Revisitando um jogo educacional para desenvolver o pensamento computacional com gramática de grafos. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)*, 30(1):863.

Junior, B. S., Cavalheiro, S., and Foss, L. (2021). Gramestation: Specifying games with graphs. In *Anais do XXXII Simpósio Brasileiro de Informática na Educação*, pages 499–511, Porto Alegre, RS, Brasil. SBC.

Lee, M. J., Bahmani, F., Kwan, I., LaFerte, J., Charters, P., Horvath, A., Luor, F., Cao, J., Law, C., Beswetherick, M., et al. (2014). Principles of a debugging-first puzzle game for computing education. In *2014 IEEE symposium on visual languages and human-centric computing (VL/HCC)*, pages 57–64. IEEE.

Rose, S. (2019). *Developing children's computational thinking using programming games*. Sheffield Hallam University (United Kingdom).

Rozenberg, G., editor (1997). *Handbook of graph grammars and computing by graph transformation: volume I. foundations*. World Scientific Publishing Co., Inc., USA.

Soloway, E. and Ehrlich, K. (1984). Empirical studies of programming knowledge. *IEEE Transactions on Software Engineering*, SE-10(5):595–609.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3):33–35.

Wong, G. K., Jian, S., and Cheung, H.-Y. (2024). Engaging children in developing algorithmic thinking and debugging skills in primary schools: A mixed-methods multiple case study. *Education and Information Technologies*, pages 1–50.

Wong, G. K. and Jiang, S. (2018). Computational thinking education for children: Algorithmic thinking and debugging. In *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, pages 328–334.