O Uso de Jogos para Apoiar o Ensino e Aprendizagem de Programação

João Stephan Maurício[1], **Alessandreia Oliveira**[1], **Marcelo Caniato Renhe**[1]

[1]Departamento de Ciência da Computação - Universidade Federal de Juiz de Fora

{joaostephan, alessandreia.oliveira, marcelo.caniato}@ice.ufjf.br

Abstract. *This article presents GameProgJF, an approach to support the teaching and learning of programming in a playful and interactive way using game development as a resource. Its implementation was based on the use of the C programming language in CodeBlocks and supported by the SFML library. GameProgJF's objective is to embrace innovative strategies in order to increase student's interest and motivation, as well as to reduce evasion in courses of Computer Science and related areas. A preliminary version was presented to the students and its evaluation was conducted through a questionnaire. Analysis of the answers indicated that the approach motivated the students with regard to programming studies.*

Resumo. *Esse artigo apresenta o GameProgJF, uma abordagem para apoiar o ensino e aprendizagem de programação de maneira lúdica e interativa usando como recurso o desenvolvimento de jogos. A sua implementação foi pautada no uso da linguagem C, a partir do CodeBlocks e apoiado pela biblioteca SFML. O objetivo do GameProgJF é adotar estratégias inovadoras para aumentar o interesse e a motivação do aluno, bem como reduzir a evasão dos cursos de Ciência da Computação e áreas afins. Foi apresentada aos alunos uma primeira versão do GameProgJF e a avaliação desta versão foi feita através de um questionário. A análise das respostas indicou que a abordagem motivou os alunos com relação ao estudo de programação.*

1. Introdução

As disciplinas relacionadas à programação são um dos alicerces dos cursos de Ciência da Computação e áreas afins. Saber programar, entender estruturas computacionais e pensar logicamente são características importantes para os alunos destes cursos [Zorzo et al. 2017]. Essas habilidades vêm sendo trabalhadas em projetos para alunos da educação infantil, ensino fundamental e médio como apresentado em [Oliveira et al. 2014, Dantas et al. 2019, Mattos et al. 2019, von Wangenheim et al. 2020], mas ainda não são uma prática comum. Consequentemente, alguns alunos chegam aos cursos de Ciência da Computação e áreas afins com poucos conhecimentos prévios relacionados à programação e ao próprio pensamento lógico, o que pode acarretar em problemas no processo inicial de aprendizagem. Além disso, alguns alunos entram no curso desconhecendo as possibilidades no âmbito profissional de um cientista da computação [Hoed 2016]. Aspectos como esses são considerados desmotivadores, bem como também podem aumentar os índices de reprovação e de evasão do curso [Gomes and Mendes 2014, Hoed 2016]. Adotar estratégias inovadoras para o

DOI: 10.5753/cbie.sbie.2020.381

ensino de programação pode incrementar o interesse e a motivação do aluno, permitindo a construção de uma base computacional que o ajudará a avançar e permanecer no curso.

Esses problemas associados ao ensino de programação já receberam diversos tratamentos, e muitas das soluções propostas se relacionam ao uso ou ao desenvolvimento de jogos [Leal et al. 2016, Rissetti et al. 2017, Coutinho et al. 2018, Figueiredo 2015, da Silva et al. 2018, Silva et al. 2018, Netto et al. 2017, Battistella et al. 2016]. Os jogos, além de serem cada vez mais acessíveis, tanto para jogar quanto para desenvolver, muitas vezes são também utilizados para atrair alunos para os cursos de Ciência da Computação ou áreas afins [Marques et al. 2011].

Diante disso, este artigo apresenta o GameProgJF, uma estratégia para apoiar o ensino e a aprendizagem de programação na disciplina de Laboratório de Programação do primeiro período dos cursos do Departamento de Ciência da Computação da Universidade Federal de Juiz de Fora (UFJF). Para tanto, o GameProgJF propõe o desenvolvimento de jogos através da linguagem C e da biblioteca SFML (Simple and Fast Multimedia Library)[1]. Uma primeira avaliação do GameProgJF foi realizada a partir de um estudo preliminar e a recepção pelos alunos foi majoritariamente positiva. Com base neste estudo, foi desenvolvida a versão do projeto aqui apresentada que possui quatro partes para serem aplicadas em diferentes aulas. Já está em desenvolvimento uma nova versão com mais partes, que será aplicada no próximo semestre letivo e adaptada ao ensino remoto, devido à pandemia do COVID-19 e às medidas de isolamento social.

O restante do artigo está organizado da seguinte maneira. A Seção 2 apresenta alguns trabalhos relacionados à proposta deste artigo. A Seção 3 apresenta uma visão geral do GameProgJF. Na Seção 4, um estudo preliminar realizado no segundo semestre de 2019 é apresentado. A Seção 5 descreve a avaliação feita com os alunos e os resultados iniciais são discutidos. A Seção 6 analisa as ameaças à validade do projeto. Para finalizar, a Seção 7 conclui o artigo e aponta algumas sugestões de trabalhos futuros.

2. Trabalhos Relacionados

Nesta seção são apresentadas algumas propostas da literatura voltadas para o auxílio no processo de ensino e aprendizagem de programação. Os trabalhos a seguir podem ser categorizados de acordo com o tipo de abordagem adotada. Alguns utilizam o desenvolvimento de jogos como forma de motivação. Nessa abordagem, objetiva-se levar o aluno a desenvolver pequenos jogos completos ou determinadas mecânicas comumente encontradas em jogos. Esses trabalhos são os que mais se aproximam da proposta deste artigo. Há ainda aqueles que utilizam gamificação ou jogos sérios como estratégia engajadora. A gamificação se trata da transposição de elementos de jogos para outros contextos não relacionados a jogos [Deterding et al. 2011]. Jogos sérios, por sua vez, são jogos que não têm o entretenimento como foco primário [Michael and Chen 2005]. Diferentemente da abordagem de desenvolvimento de jogos, essa categoria utiliza os jogos como um meio, e não como um produto final das atividades dos alunos. Embora as abordagens sejam distintas, todos os trabalhos a seguir buscam estimular a motivação dos alunos, contribuindo para a absorção de conceitos de programação e o desenvolvimento do pensamento lógico.

Na abordagem baseada no desenvolvimento de jogos, [Leal et al. 2016] apresenta uma proposta de minicurso focado na construção de três jogos com níveis crescentes de

[1] https://www.sfml-dev.org/

dificuldade utilizando o ambiente Scratch[2]. O minicurso foi aplicado em dois eventos separados, sendo que somente o segundo, adaptado para alunos dos ensinos fundamental e médio, foi avaliado. Os alunos se manifestaram de forma bastante positiva, com uma parcela expressiva destes afirmando, após o evento, cogitar o ingresso em cursos da área. Já em [Coutinho et al. 2018], a ferramenta de criação de jogos digitais GameMaker[3] foi aplicada a alunos de uma disciplina de programação, de forma integrada com outras disciplinas ligadas à área de *design*. Os participantes, via questionário, consideraram que o uso da ferramenta, além de motivador, auxiliou no entendimento de lógica de programação, embora isso não tenha se materializado em um melhor desempenho na disciplina. Especificidades do GameMaker e o pouco tempo dedicado à apresentação da plataforma foram apontados como possíveis razões para este resultado. O trabalho de [Rissetti et al. 2017], por outro lado, tem como foco o aprendizado de conceitos de computação gráfica. Uma API (Application Programming Interface) para programação de jogos 2D é proposta como alternativa ao OpenGL (Open Graphics Library)[4], muitas vezes considerado de difícil compreensão por iniciantes no tema. Os alunos entrevistados destacaram a facilidade de uso da API e o interesse de utilizá-la em outras disciplinas.

Na categoria relacionada à gamificação, uma estratégia de adoção da metodologia Gamification Design Framework é apresentada em [Figueiredo 2015]. Medalhas e pontuações foram elaboradas, além de um sistema de ranqueamento. Conclusões preliminares apontaram para uma maior cooperação entre os alunos quando adotado um ranqueamento por grupos. Como o ambiente utilizado não oferecia um controle automatizado das pontuações, houve dificuldade no monitoramento manual das mesmas. Por outro lado, [da Silva et al. 2018] propõe um modelo conceitual que facilita o acompanhamento do professor, além de oferecer uma personalização em função do perfil de aprendizado do aluno. O modelo contempla o domínio cognitivo da Taxonomia de Bloom [Bloom et al. 1956] e apresenta exercícios na forma de *quiz*, usando as interações do aluno para classificar seu perfil e ativar elementos de gamificação relacionados. Utilizando um grupo de controle, verificou-se que os índices de engajamento aumentaram em todas as turmas em que o modelo foi aplicado. Entrevistas com os professores também apontaram para uma maior dedicação dos estudantes. Já em [Silva et al. 2018], a gamificação é ainda associada a uma experiência de *storytelling*, com a apresentação de um enredo de contextualização associado ao conteúdo. Por meio de uma entrevista coletiva, os alunos destacaram a motivação gerada pelo sentimento de curiosidade atrelado ao enredo.

Pode-se ainda citar duas propostas de utilização de jogos sérios no ensino de programação. A primeira, apresentada em [Netto et al. 2017], utiliza um jogo digital para estimular o raciocínio lógico. O jogo é composto por uma série de desafios e um sistema de conquistas, além de recompensas que estimulam o aluno à busca por soluções mais simples para os problemas apresentados. A experiência foi classificada como positiva pela maioria dos discentes, que disseram ter sido possível identificar assuntos já ministrados em sala de aula, além de terem compreendido facilmente a proposta do jogo. A segunda proposta [Battistella et al. 2016] envolve um jogo ligado ao algoritmo *heapsort* [Drozdek 2002]. O jogo, chamado SORTIA, foi avaliado seguindo o modelo de avaliação de jogos educacionais MEEGA [Savi et al. 2012]. Os resultados do experimento rea-

[2] https://scratch.mit.edu/
[3] https://www.yoyogames.com/gamemaker
[4] https://www.opengl.org/

lizado foram em geral positivos, tanto nos âmbitos da motivação e da experiência de usuário, quanto no aspecto de aprendizagem. Além da avaliação do jogo em si, foram realizadas avaliações com os discentes antes e depois do experimento, evidenciando uma melhora significativa em todos os aspectos analisados.

O presente trabalho adota a abordagem de desenvolvimento de jogos, mas se diferencia destes discutidos anteriormente por não requerer dos alunos o uso de ferramentas específicas. Embora muito úteis no ensino de programação, essas ferramentas se baseiam no conceito de programação em blocos, estando desvinculadas da sintaxe da linguagem usada na disciplina-alvo. Assim, aqui é proposta uma metodologia de trabalho em que o aluno resolve problemas de desenvolvimento de jogos utilizando somente a linguagem de programação estudada no âmbito da disciplina. A seção a seguir apresenta esta proposta.

3. GameProgJF

O projeto do jogo foi desenvolvido e configurado para uso no CodeBlocks[5]. Além de gratuito, o CodeBlocks é a primeira IDE apresentada aos discentes na disciplina-alvo deste projeto, que trabalha com a linguagem C. Foi utilizada também a biblioteca SFML, que é um *software* livre desenvolvido com o intuito de fornecer suporte para o uso de componentes multimídia, contendo módulos para a criação de janelas, a renderização de elementos gráficos, a reprodução de áudio e a transmissão de dados por UDP e TCP. O projeto pode ser dividido em duas etapas: a preparação dos materiais e a aplicação destes com os alunos. A etapa de preparação envolve a elaboração do jogo e das atividades a serem desenvolvidas pelos alunos. O jogo e as atividades são divididos em quatro partes, cada uma abordando um subconjunto do conteúdo programático da disciplina.

- 1ª parte: declaração de variáveis e uso de funções;
- 2ª parte: estruturas condicionais;
- 3ª parte: estruturas de repetição, vetores e *strings*;
- 4ª parte: matrizes e estruturas de dados heterogêneas.

São geradas quatro versões do jogo, uma para cada parte, conforme ilustrado na Figura 1. Cada versão acrescenta novas mecânicas ao jogo que podem ser implementadas com o conteúdo visto na disciplina até então. Foi utilizado o Git[6], para que fosse mantido controle sobre o progresso e as alterações no desenvolvimento do jogo, com cada versão associada a um *branch* específico. Além disso, o código foi desenvolvido de forma a minimizar o contato do aluno com funções da SFML e as estruturas de dados usadas na implementação do jogo, evitando que isso pudesse confundi-los.

Na primeira parte, os alunos trabalham com a movimentação do personagem (Figura 1a), utilizando o conteúdo aprendido de funções e declaração de variáveis. O jogo possui algumas funções auxiliares que encapsulam as funções de entrada da SFML. Assim, o objetivo do aluno é utilizar essas funções para detectar as teclas pressionadas pelo usuário e calcular o deslocamento do personagem. Há ainda um outro exercício que consiste em alterar a movimentação do personagem de movimento uniforme para acelerado.

Já na segunda parte, utilizando estruturas condicionais e conceitos de geometria, os alunos devem implementar a colisão do personagem com outros elementos do jogo.

[5] http://www.codeblocks.org/
[6] https://git-scm.com/

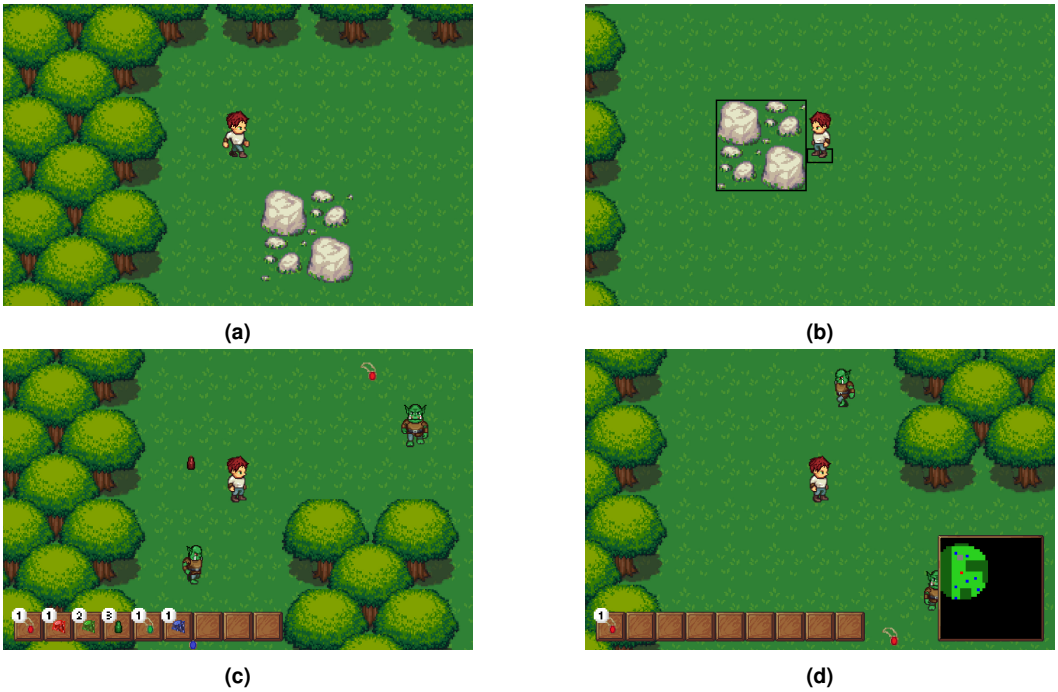

Figura 1. Ilustração das 4 partes do jogo. Em (a), o personagem se desloca pelo cenário. Em (b), os retângulos envolventes usados na detecção de colisão do personagem com obstáculos. Em (c), o inventário usado na terceira parte do jogo. Por fim, em (d), o minimapa construído na última parte.

São elaborados exercícios para a colisão do personagem com as bordas do mapa e com outros objetos espalhados pelo cenário (Figura 1b). Para isso, o aluno deve testar um conjunto de condições que verificam se o retângulo envolvente do personagem intercepta o de um outro objeto e, em caso positivo, com qual lado do objeto o personagem está colidindo. Os alunos também implementam uma função para incluir dois tipos de movimentação para um inimigo: uma em que este se move livremente de um lado ao outro do mapa, e outra em que o inimigo persegue o jogador.

A terceira parte abrange o conteúdo de vetores e *strings*. O objetivo é implementar um inventário de itens para o jogador (Figura 1c). Nessa parte, os dois primeiros exercícios tratam da escolha dos itens que devem ser coletados pelo personagem. Cada item é representado por um caractere, de modo que uma lista de itens é definida por uma *string*. Assim, há duas *strings*: uma que representa os itens disponíveis e outra que indica os itens que devem ser coletados. O aluno deve gerar a *string* de itens coletáveis aleatoriamente, a partir da *string* que contém todos os itens. Dessa forma, o aluno trabalha com algoritmos de concatenação e remoção de caracteres em uma *string*. Já no terceiro exercício, o aluno deve implementar o inventário, que é representado por dois vetores: um de caracteres, contendo os itens coletados, e um de inteiros que possui as quantidades de cada item. O aluno deve implementar uma busca linear para encontrar a posição do item no inventário e atualizar o contador. Por fim, há ainda uma função de comparação de *strings*, que verifica a condição de vitória do jogo, que ocorre quando os itens coletados correspondem aos da *string* gerada pelas funções dos dois primeiros exercícios.

Na quarta e última parte, é incluída a mecânica de um minimapa no jogo (Fi-

gura 1d), utilizando matrizes e estruturas de dados heterogêneas. Inicialmente, os alunos devem criar duas estruturas para armazenar os dados do jogador (sua posição e raio do campo de visão) e os elementos do minimapa (como a matriz que o representa e a variável que define o jogador), bem como implementar as funções para atualização desses dados. Em seguida, é implementada uma função para alterar a matriz do minimapa, de forma que a célula da matriz referente à posição do jogador fique em destaque. O último exercício consiste em estabelecer um campo de visão para o personagem no minimapa, de modo que toda célula a um raio de distância de onde o jogador está fique oculta.

Após o desenvolvimento do jogo, cada parte foi adaptada para a aplicação em laboratório, preparando o código-fonte para que os alunos resolvessem os exercícios. O código foi estruturado de forma que as funções que precisam ser implementadas estejam disponíveis no arquivo principal do projeto, facilitando o acesso dos alunos, que ainda não trabalham com projetos compostos por vários arquivos. As funções possuem o mínimo necessário de código para garantir a compilação e a execução do jogo mesmo sem a mecânica implementada. Um exemplo de como o código fica pode ser visto na Listagem 1. O objetivo é o aluno implementar a função para que a mecânica de jogo solicitada funcione adequadamente. Uma das soluções deste exercício pode ser vista na Listagem 2.

```
1  ///EXERCICIO 3
2  int movimentaPersonagem(Jogador* p, float deltaTempo)
3  {
4      return 0;
5  }
```

Listagem 1. Exercício com o objetivo de movimentar o personagem

```
1  ///EXERCICIO 3
2  int movimentaPersonagem(Jogador* p, float deltaTempo)
3  {
4      float x, y;
5      x = obtemPosicaoX(p);
6      y = obtemPosicaoY(p);
7      x = x + deslocamentoLateral(p, deltaTempo);
8      y = y + deslocamentoVertical(p, deltaTempo);
9      atualizaPosicao(p, x, y);
10     return 0;
11 }
```

Listagem 2. Resolução do exercício de movimentação do personagem

4. Estudo Preliminar

Como mencionado, a versão descrita na Seção 3 foi desenvolvida a partir dos resultados de um estudo realizado com uma versão inicial do GameProgJF, que foi apresentada aos alunos do primeiro período do curso de Ciência da Computação da UFJF, na disciplina de Laboratório de Programação, no segundo semestre de 2019. Foram agendados quatro encontros, correspondentes a cada uma das partes mencionadas na Seção 3. As três primeiras partes foram implementadas no formato de um jogo de plataforma, com exercícios bem similares aos da versão atual do jogo. Para a quarta parte, foi utilizada uma versão simplificada do jogo Tetris[7], visando facilitar a visualização do conceito de matriz através

[7] https://tetris.com/play-tetris

da grade bidimensional onde as peças do jogo são dispostas.

Em cada encontro, os bolsistas vinculados ao projeto apresentavam para a turma o objetivo daquela aula, apresentando através de *slides* os exercícios. A mecânica de jogo presente em cada exercício era ressaltada, com uma demonstração de qual seria o resultado esperado após a correta implementação da função. Em alguns casos, o aluno somente conseguiria visualizar o resultado de sua implementação após concluir outros exercícios. Explicitar essa dependência durante a apresentação se mostrou importante para não gerar frustração entre os alunos, que muitas vezes implementavam a função mas não tinham como averiguar visualmente a corretude da mesma.

Durante a resolução individual dos exercícios, os bolsistas, juntamente com os monitores e o professor da disciplina, ficaram à disposição para esclarecer dúvidas e auxiliar os alunos. Isso foi importante para que os alunos se mantivessem motivados, conseguindo esclarecer suas dúvidas e avançar nos problemas propostos. Foi possível observar que no primeiro encontro, os alunos apresentaram maiores dificuldades, pois, apesar da simplicidade dos exercícios, a maioria ainda não estava habituada com a sintaxe da linguagem e o pensamento algorítmico. Nos encontros seguintes, os alunos já apresentaram uma maior desenvoltura com a linguagem de programação. Os exercícios envolvendo vetores e estruturas de repetição na terceira parte também apresentaram desafios aos discentes, que acabavam por se confundir em algumas manipulações com *strings*, como o percurso e a remoção de caracteres. Já na última parte, as principais dificuldades dos alunos foram relacionadas à sintaxe de uso de estruturas de dados heterogêneas. Essas dificuldades são similares às encontradas pelos alunos em aulas tradicionais de laboratório de programação. Contudo, a motivação proporcionada pelo jogo fez com que a grande maioria dos alunos se mantivessem engajados nos problemas propostos durante as aulas.

5. Avaliação da abordagem

Para avaliar o GameProgJF, os alunos envolvidos foram convidados a preencher um formulário de encerramento disponibilizado via Google Forms. Como o preenchimento era opcional, apenas 9 dos 27 alunos preencheram o formulário que possuía duas partes, uma de avaliação do projeto e outra para sugestões e observações. Parte das questões utilizou a escala de Likert [Likert 1932], dada sua adequabilidade para medir a opinião dos participantes e facilidade de análise dos resultados.

Na seção de avaliação, inicialmente os alunos definiram a experiência de participação do projeto. A maioria afirmou que foi interessante, motivadora e que aumentou o interesse deles na disciplina. Porém alguns citaram que sentiram dificuldades e ficaram intrigados com a proposta. Quando questionados sobre o quanto as atividades desenvolvidas despertaram seu interesse, em comparação às atividades tradicionais, dos 9 alunos, 6 responderam que o interesse foi aumentado e 3 que não foi afetado. Na terceira questão os alunos deveriam indicar se, após as aulas, eles se motivaram a pesquisar por conta própria algo relacionado ao desenvolvimento de jogos, às tecnologias utilizadas ou à programação em geral. Dos 9 alunos, 8 responderam sim. Já na questão "Você julga que aplicar princípios de programação ao desenvolvimento de um jogo aumentou seu entendimento acerca desses mesmos princípios?", 7 alunos confirmaram que aprenderam mais e de maneira mais fácil. Quando questionados sobre quais dos fatores agregaram mais dificuldade à resolução dos problemas, a maioria indicou a ausência de conhecimentos

prévios de programação. As próximas questões são sintetizadas na Tabela 1.

Tabela 1. Avaliação dos alunos relacionada aos seus conhecimentos e ao projeto

Fator analisado	MB	B	N	A	MA
Motivação gerada pelo desenvolvimento de jogos	0	2	1	4	2
Familiaridade com a programação de jogos	4	3	0	1	1
Entendimento dos exercícios propostos	0	2	3	4	0
Dificuldade de completar exercícios	0	1	4	4	0

MB - Muito baixo; B - Baixo; N - Neutro; A - Alto; MA - Muito alto

A primeira seção do estudo foi finalizada com a avaliação dos bolsistas (Tabela 2). Já na seção de sugestões e observações, os alunos envolvidos sugeriram, entre outras coisas, maior detalhamento dos exercícios propostos e de suas soluções bem como mais bolsistas para esclarecimentos necessários. As sugestões apresentadas foram levadas em conta no desenvolvimento do novo material e na versão do projeto descrita neste artigo.

Tabela 2. Avaliação dada com relação aos bolsistas

Fator analisado	DC	DP	N/I	CP	CC
Os bolsistas foram eficientes	0	0	0	2	7
As apresentações foram claras e organizadas	0	0	1	2	6
Os bolsistas estimularam o interesse dos alunos	0	0	0	3	6
Os bolsistas utilizaram bem o tempo durante as aulas	0	0	0	2	7
Os bolsistas foram acessíveis e prestativos	0	0	0	2	7

DC - Discordo completamente; DP - Discordo parcialmente; N/I - Neutro ou indiferente; CP - Concordo parcialmente; CC - Concordo completamente

6. Ameaças à Validade

Durante o planejamento deste estudo, buscou-se minimizar ameaças que pudessem impactar ou limitar a validade dos resultados obtidos [Wohlin et al. 2012]. No entanto, não é possível garantir que tais ameaças não tenham afetado os resultados.

A avaliação não foi executada em um único dia por todos os participantes. Contudo, o mesmo roteiro foi utilizado com a intenção de minimizar esta ameaça.

O entendimento dos participantes sobre as questões do formulário é diretamente influenciado pela forma como elas foram elaboradas. A análise dos instrumentos utilizados (inclusive o formulário) a partir de um estudo piloto visou reduzir esta interferência.

O tamanho da amostra é limitado, o que não é ideal do ponto de vista estatístico. Desta forma, os resultados do estudo não são conclusivos, somente fornecem indícios. Além disso, a participação dos estudantes era opcional. No próximo estudo, o planejamento inclui a participação de todos. Assim, o número será superior a 30, o que é considerado alto [Juristo and Moreno 2013] e aumenta a validade estatística das conclusões.

7. Considerações Finais

Este artigo apresentou o GameProgJF, que tem como objetivo apoiar o processo de ensino e aprendizagem de programação. Uma primeira versão da proposta foi aplicada no

segundo semestre de 2019. A avaliação realizada indicou que o projeto motivou os alunos envolvidos e aumentou o interesse deles por programação. Por outro lado, os exercícios propostos foram considerados difíceis e desafiadores por parte deles. Em consequência disso, algumas sugestões foram feitas pelos próprios alunos e a maioria delas se relacionava ao nível de dificuldade enfrentado. Estas sugestões foram levadas em consideração no planejamento e desenvolvimento da segunda versão do GameProgJF e do material atualizado descrito na Seção 3.

Como trabalho futuro, a nova versão da abordagem e do material desenvolvido serão adaptados ao ensino remoto, em função da pandemia e apresentados à turma de Laboratório de Programação do próximo semestre letivo. Outro trabalho já em andamento consiste no desenvolvimento de novas partes para o projeto, de modo a trabalhar separadamente alguns assuntos que hoje estão sendo tratados em conjunto, como é o caso de estruturas de repetição e vetores (3ª parte) bem como matrizes e *structs* (4ª parte). Além disso, um estudo experimental para que seja verificado o impacto no ensino e aprendizagem de programação está sendo elaborado e será aplicado ao final das atividades.

Referências

Battistella, P. E., Petri, G., von Wangenheim, C. G., von Wangenheim, A., and Martina, J. E. (2016). Sortia 2.0: Um jogo de ordenação para o ensino de estrutura de dados. In *Anais do XII Simpósio Brasileiro de Sistemas de Informação*, pages 558–565. SBC.

Bloom, B., Englehart, M., Furst, E., Hill, W., and Krathwohl, D. (1956). Taxonomy of educational objectives: Handbook i. *Cognitive domain. New York: David McKay*.

Coutinho, E., Bonates, M., and Moreira, L. O. (2018). Relato sobre o uso de uma ferramenta de desenvolvimento de jogos para o ensino introdutório de lógica de programação. In *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, volume 7, page 689.

da Silva, T. S. C., de Melo, J. C. B., and Tedesco, P. C. d. A. R. (2018). Um modelo para promover o engajamento estudantil no aprendizado de programação utilizando gamification. *Revista Brasileira de Informática na Educação*, 26(03):120.

Dantas, I., Neto, J., Silva, L., Neto, L., Lima, D., Scaico, P., and Costa, T. (2019). Ensino de lógica de programação no ensino fundamental utilizando o jogo robotizen: um relato de experiência. In *Anais do XXVII Workshop sobre Educação em Computação*, pages 51–60, Porto Alegre, RS, Brasil. SBC.

Deterding, S., Dixon, D., Khaled, R., and Nacke, L. (2011). From game design elements to gamefulness: defining gamification. In *Proceedings of the 15th Intern. Academic MindTrek Conference: Envisioning future media environments*, pages 9–15. ACM.

Drozdek, A. (2002). *Estrutura de Dados e Algoritmos em C++*. Pioneira Thomson Learning.

Figueiredo, K. (2015). Proposta de gamificação de disciplinas em um curso de sistemas de informação. In *XI Simpósio Bras. de Sistemas de Informação*, pages 611–614. SBC.

Gomes, A. and Mendes, A. (2014). A teacher's view about introductory programming teaching and learning: Difficulties, strategies and motivations. In *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, pages 1–8. IEEE.

Hoed, R. M. (2016). Análise da evasão em cursos superiores: o caso da evasão em cursos superiores da área de computação.

Juristo, N. and Moreno, A. M. (2013). *Basics of software engineering experimentation*. Springer Science & Business Media.

Leal, V., Oliveira, A., and Borges, M. (2016). Despertando para a programação com a criação de jogos. In *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, volume 5, page 358.

Likert, R. (1932). A technique for the measurement of attitudes. *Archives of psychology*.

Marques, D. L., Costa, L. F. S., de Azevedo Silva, M. A., and Rebouças, A. D. D. S. (2011). Atraindo alunos do ensino médio para a computação: Uma experiência prática de introdução à programação utilizando jogos e python. In *Anais do Workshop de Informática na Escola*, pages 1138–1147.

Mattos, M., Kohler, L., Zucco, F., Fronza, L., Bizon, A., Silveira, H., Santos, B., and Carlo, G. (2019). Ambiente de programação para a introdução da lógica de programação. *Anais do Workshop de Informática na Escola*, 25(1):1259.

Michael, D. R. and Chen, S. L. (2005). *Serious games: Games that educate, train, and inform*. Muska & Lipman/Premier-Trade.

Netto, D., Medeiros, L. M., de Pontes, D., and de Morais, E. (2017). Game logic: Um jogo para auxiliar na aprendizagem de lógica de programação. In *Anais do XXV Workshop sobre Educação em Computação*. SBC.

Oliveira, M. d., Souza, A. d., Ferreira, A., and Barbosa, E. (2014). Ensino de lógica de programação no ensino fundamental utilizando o scratch: um relato de experiência. In *XXXIV Congresso da SBC-XXII Workshop de Ensino de Computação, Brasília*.

Rissetti, G., Machado, F., and Miranda, P. (2017). Fx canvas2d: uma api de jogos bidimensionais para auxiliar na aprendizagem de programação. In *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, volume 6, page 912.

Savi, R., von Wangenheim, C. G., Borgatto, A., Buglione, L., and Ulbricht, V. (2012). Meega–a model for the evaluation of games for teaching software engineering. Technical report, Technical Report INCoD.

Silva, J. A. L., Oliveira, F. C. S., and Martins, D. J. S. (2018). Gamificação e storytelling como estratégia motivacional no ensino de programação. *Proceedings of SBGames*.

von Wangenheim, C., Medeiros, G., Filho, R. M., Petri, G., Pinheiro, F., Ferreira, M. N., and Hauck, J. (2020). Desenvolvimento e avaliação de um jogo de tabuleiro para ensinar o conceito de algoritmos na educação básica. *RBIE*, 27(03):310.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media.

Zorzo, A. F., Nunes, D. J., Matos, E. D. S., Steinmacher, I. F., Leite, J. C., De Araujo, R. M., Correia, R. C. M., and Martins, S. D. L. (2017). Referenciais de formação para os cursos de graduação em computação.