A criação de jogos para o ensino de computação: uma análise comparativa*†

Júlia Veiga da Silva[1], **Braz Araujo da Silva Junior**[1], **Luciana Foss**[1], **Simone André da Costa Cavalheiro**[1]

[1]Laboratório de Fundamentos da Computação – Universidade Federal de Pelotas (UFPel)
CEP 96.010-610 – Pelotas – RS – Brasil

{jvsilva, badsjunior, lfoss, simone.costa}@inf.ufpel.edu.br

Abstract. *Considering the challenges of teaching computing and the use of digital games as one of the ways to help this theme, this paper presents a comparative analysis of three game creation platforms used in computer education: Greenfoot, Scratch and GrameStation. For the analysis, we define topics in common and discuss specific features of each tool. We conclude that the GrameStation platform, although based on a mathematical formalism, it has characteristics equivalent to the others and can be also used to introduce formal specification in computer education.*

Resumo. *Considerando os desafios do ensino de computação e a utilização de jogos digitais como maneira de auxiliar esse cenário, este artigo apresenta uma análise comparativa de três plataformas destinadas à criação de jogos, utilizadas no ensino de computação: Greenfoot, Scratch e GrameStation. Foram definidos, para a análise, tópicos comuns às ferramentas, bem como suas características individuais. A plataforma GrameStation, embora baseada em um formalismo matemático, possui características equivalentes às demais, sendo possível a discussão quanto a sua utilidade para a introdução da especificação formal no ensino de computação.*

1. Introdução

Sendo introduzida ao currículo escolar em diversos países, a computação provoca desafios aos educadores. Assim, o aprendizado de pedagogias apropriadas para o ensino de conteúdos como algoritmos, programação, Pensamento Computacional (PC), entre outros, torna-se fundamental [Sentance e Csizmadia 2017]. Neste cenário, os jogos digitais são comumente utilizados como recurso educacional, pois, interativos e visualmente atraentes, proporcionam diversão e estimulam a criatividade dos estudantes. Além disso, métodos de ensino centrados na criação de jogos digitais possuem um elevado potencial educacional, proporcionando, inclusive, o desenvolvimento de habilidades em STEM[1] [Wernbacher et al. 2020]. Egenfeldt-Nielsen (2010) evidencia três maneiras diferentes de aplicação de jogos na educação formal, sendo elas:

*Trabalho concluído.

†O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001, do MCTIC/CNPq (Rede Sacci), da FAPERGS, do CNPq, da SMED/Pelotas e da PREC e PRPPG/UFPel.

[1]Ciência, Tecnologia, Engenharia e Matemática (do inglês *Science, Technology, Engineering and Mathematics*).

1. Aprendizagem com o auxílio de jogos: refere-se à utilização de jogos com o objetivo de ensinar um determinado conteúdo. São jogos desenvolvidos, especificamente, para fins educacionais (jogos sérios).
2. Aprendizagem com jogos: refere-se à adaptação de jogos que não possuem fins didáticos, para serem utilizados durante o estudo de temas, conceitos, métodos etc.
3. Aprendizagem por meio da criação de jogos: refere-se ao desenvolvimento de um projeto, permitindo a sistematização do conhecimento quanto a um determinado tópico. Não é necessário que o jogo possua uma finalidade educativa. As competências, habilidades e conhecimento adquiridos durante seu desenvolvimento são mais importantes do que a própria finalidade.

Trabalhos relacionados apresentam levantamentos em relação aos recursos utilizados para o ensino de computação na educação. Doss et al. (2011) exibem uma pesquisa referente às principais plataformas de desenvolvimento de jogos utilizadas como ferramentas de ensino. Os aspectos técnicos – como tipo de interface, mecanismo de depuração, compartilhamento de projetos etc. – também são analisados e comparados. Já Brezolin e Silveira (2021), expõem um mapeamento sistemático – realizado a partir dos principais fóruns brasileiros promovidos pela Sociedade Brasileira de Computação – relativo às principais ferramentas tecnológicas utilizadas para o ensino de programação e desenvolvimento do PC. Destacando o terceiro tópico apontado por Egenfeldt-Nielsen (2010), este artigo apresenta uma análise comparativa de três plataformas utilizadas para o ensino de computação por meio da criação de jogos. Além disso, evidencia a importância da plataforma GrameStation – baseada em um formalismo matemático – neste contexto.

O artigo está organizado como segue. A Seção 2 conta com a metodologia abordada para a seleção das plataformas. A Seção 3 apresenta e descreve as plataformas selecionadas. A Seção 4 é destinada à análise dessas ferramentas. A Seção 5 discorre acerca das considerações finais e trabalhos futuros desta proposta.

2. Metodologia

A pesquisa foi realizada por meio do mecanismo Google Acadêmico, com foco em plataformas destinadas à criação de jogos para o ensino de computação. No processo de seleção, foram excluídos trabalhos que propunham a utilização de jogos para o ensino de computação, trabalhos cujas plataformas propostas não eram destinadas à criação de jogos e, por fim, trabalhos em que as plataformas propostas não permitiam a criação de projetos próprios. Por fim, a busca resultou em três plataformas: Greenfoot, Scratch e GrameStation.

Para a fase da análise, os critérios definidos foram divididos em duas categorias: características que normalmente são observadas em plataformas de desenvolvimento utilizadas na educação e características específicas, definidas a partir das plataformas selecionadas. Da primeira categoria, foram analisados quatro critérios: **(1)** Disponibilidade: as plataformas estão facilmente disponíveis (incluindo custo)? **(2)** Personalização: as plataformas permitem a importação de recursos externos (como imagens e sons), possibilitando a criação de jogos personalizados? **(3)** Compartilhamento: há a possibilidade de comunicação e compartilhamento de atividades entre os usuários? **(4)** Feedback imediato: as plataformas oferecem feedback visual imediato sobre as ações dos usuários?

Neste caso, feedback imediato refere-se à perspectiva WYSIWYG – acrônimo em inglês para "*What You See Is What You Get*" – ou seja, verifica se as ferramentas permitem a visualização em tempo real dos elementos a serem publicados ou impressos na tela, ainda durante a edição do documento.

Da segunda categoria, referente às características específicas, foram analisados seis critérios: **(1)** Conceitos introduzidos: quais conceitos da computação são introduzidos em cada uma dessas ferramentas? **(2)** Público-alvo: qual a idade indicada para a utilização das ferramentas? **(3)** Linguagem: que tipo de linguagem é utilizada para a criação dos jogos? **(4)** Amigabilidade: os usuários são capazes de descobrir as funcionalidades necessárias para atingir um determinado objetivo facilmente? **(5)** Conhecimento prévio: é necessário algum nível de conhecimento prévio a respeito do tema trabalhado pela plataforma antes de sua utilização? **(6)** Paradigma: qual o paradigma de programação utilizado pela plataforma?

3. Plataformas

Nesta seção são apresentadas as ferramentas selecionadas para a análise.

3.1. Greenfoot

Proposto por Henriksen e Kölling (2004), o Greenfoot[2] é um ambiente de desenvolvimento que visa apoiar o ensino e a aprendizagem do paradigma de Programação Orientada a Objetos (POO) a partir da criação de jogos. O ambiente possui um sistema integrado – ou seja, além da interface principal, dispõe de um editor, de um compilador e de um depurador – e utiliza a linguagem de programação Java para o desenvolvimento dos jogos. Além disso, conta com 9 classes nativas que facilitam o processo de criação: 1) Actor, classe que contém os métodos disponíveis para os atores do jogo; 2) Color, classe utilizada para o preenchimento de cores na tela; 3) Font, classe correspondente à fonte utilizada para os textos apresentados na tela; 4) Greenfoot, classe utilizada para a comunicação com o ambiente; 5) GreenfootImage, classe que contém os métodos de apresentação e manipulação de imagens; 6) GreenfootSound, classe que contém os métodos de apresentação e manipulação de sons; 7) MouseInfo, classe que contém os métodos de captura de eventos do mouse; 8) UserInfo, classe utilizada para o armazenamento permanente de dados em um servidor e para o compartilhamento desses dados entre diferentes usuários quando o cenário é executado no site da plataforma; e 9) World, classe responsável pela implementação dos métodos relacionados ao cenário do jogo.

O design do ambiente possibilita a introdução de conceitos fundamentais da POO, tradicionalmente difíceis de ensinar devido ao alto grau de abstração, de uma maneira facilmente compreensível, por meio de interações [Kölling 2010]. Entre os principais conceitos estão: 1) Programa, o qual consiste em um conjunto de classes; 2) Classes, das quais se podem criar objetos; 3) Criação de vários objetos a partir de uma classe; 4) Identificação de métodos e campos para objetos de uma mesma classe; 5) Estado individual de objetos dado pelos valores de seus campos; 6) Comunicação entre objetos por meio de seus métodos; 7) Métodos, que podem possuir parâmetros e retornar valores; e 8) Tipos, de parâmetros e de valores de retorno. Assim, as atividades dos alunos incluem: instanciar objetos, executar uma simulação, chamar interativamente métodos únicos, ler e

[2]Disponível em: https://www.greenfoot.org/.

modificar o código, compilar e criar subclasses de objetos de simulação etc. O Greenfoot tem o objetivo de apresentar conceitos antes da sintaxe. Para Kölling (2010), a sintaxe não deve ser o primeiro obstáculo a ser enfrentado pelos alunos antes de atingir o sucesso em seus projetos. No entanto, pela experiência de manipulação do código-fonte – ainda que posterior –, o Greenfoot é recomendado para um público-alvo a partir de 14 anos, também podendo ser utilizado no Ensino Superior.

De acordo com Kölling (2010), uma série de objetivos sustentam o design do ambiente. Tais objetivos podem ser resumidos em dois pontos, a partir de duas perspectivas diferentes: da perspectiva do aluno, o objetivo é tornar a programação envolvente, criativa e satisfatória; do ponto de vista do professor, o objetivo é que o ambiente auxilie ativamente no ensino de conceitos de programação importantes e universais.

3.2. Scratch

O Scratch[3] [Resnick et al. 2009] é uma linguagem de programação em blocos, disponível on-line, que possibilita a criação não só de jogos, mas também de animações e histórias por iniciantes, sem a necessidade de uma sintaxe tradicional. Um dos principais objetivos do Scratch é apresentar a programação àqueles sem experiências prévias, motivando o aprendizado de conceitos de uma maneira lúdica. Esse objetivo impulsionou diversos aspectos do design do Scratch, como a escolha de uma linguagem de blocos, do layout de interface e de um conjunto mínimo de comandos [Maloney et al. 2010]. A interface, por exemplo, é dividida em quatro painéis, de modo a garantir que os componentes principais estejam sempre visíveis: o primeiro painel refere-se aos comandos disponíveis, divididos em 8 categorias – entre elas, “Movimento”, “Controle”, “Som” e “Operadores”; o segundo painel apresenta os *scripts* relacionados ao *sprite*[4] selecionado no momento, bem como sons e imagens referentes a esse *sprite*; o terceiro painel é onde a ação acontece; e o quarto painel exibe as miniaturas de todos os *sprites* no projeto.

Os *scripts* são construídos a partir da conexão de blocos que representam instruções, expressões e estruturas de controle. As formas dos blocos sugerem seu encaixe e o sistema de arrastar e soltar se recusa a conectar blocos de formas incoerentes. Com isso, a gramática visual das formas dos blocos e suas regras de combinação desempenham o papel de sintaxe em uma linguagem baseada em texto. Conforme Maloney (2010), a linguagem de blocos elimina a possibilidade de erros de sintaxe, permitindo que os usuários se concentrem em problemas de interesse imediato ao invés de se empenhar em apenas fazer seu programa compilar. Além disso, o sistema está sempre ativo: não há etapa de compilação ou alternância entre a execução e a edição do programa. Desse modo, para o autor, o feedback visual enquanto os blocos são arrastados auxilia na compreensão da organização de programas e utilização de diferentes tipos de dados.

3.3. GrameStation

O GrameStation[5] é um motor de jogos baseado em Gramática de Grafos (GG). A GG é um formalismo matemático para especificar e verificar sistemas. Formalmente é definida por um grafo tipo, que diferencia e restringe os elementos (vértices e arestas) permitidos

[3]Disponível em: https://scratch.mit.edu/.

[4]No Scratch, *sprites* são objetos que executam funções controladas por *scripts* (são os personagens e objetos dos projetos).

[5]Disponível em: https://wp.ufpel.edu.br/pensamentocomputacional/gramestation-pt/.

em um sistema; um grafo inicial (composto por elementos de tipos definidos pelo grafo tipo), que especifica o estado inicial do sistema; e um conjunto de regras de transformação de grafos, que representam os possíveis comportamentos do sistema. Dessa forma, é possível realizar a simulação de um sistema a partir da especificação de uma GG, verificando estados alcançáveis e detectando e evitando estados indesejados. Silva Junior (2020) relaciona características das GGs a diversas habilidades do PC, como análise e representação de dados, decomposição de problemas, abstração, algoritmos e processos, simulação e paralelismo.

No GrameStation, a especificação de um jogo corresponde à especificação de uma GG, com a definição de grafos e regras. Sendo assim, o motor é estruturado em módulos: Game Tutorial, onde o usuário pode aprender a utilizar a plataforma; Game Builder, onde o usuário pode construir seus jogos especificando as GGs; e Game Player, onde o usuário pode executar seus jogos simulando as GGs, isto é, selecionando regras e aplicando-as a partir do grafo inicial.

4. Análise das Plataformas

Ainda que possuam focos distintos, as três plataformas utilizam a criação de jogos como maneira de estimular o desenvolvimento de habilidades relacionadas à computação. Greenfoot e Scratch, por exemplo, são indicados para atividades relacionadas ao ensino de programação. Scratch, no entanto, conta com uma abordagem de introdução a lógica, enquanto Greenfoot faz uso da POO, ou seja, é indicado a quem já possui conhecimentos acerca do desenvolvimento de programas. Dessa forma, por visar usuários mais jovens e focar na aprendizagem autodirigida, enfatizando a facilidade da manipulação, o Scratch pode ser utilizado como uma introdução ao Greenfoot [Maloney et al. 2010].

Com relação a características comumente observadas em tecnologias voltadas à educação, Greenfoot, Scratch e GrameStation são consideravelmente equivalentes. A Tabela 1 ilustra os quatro aspectos considerados. Relativo ao primeiro item, as três plataformas encontram-se disponíveis on-line e de forma gratuita. A importação de recursos externos também é permitida, proporcionando a criação de jogos personalizados. Referente ao compartilhamento de projetos, apenas Greenfoot e Scratch contam com esse recurso, permitindo o compartilhamento em seus sites. Por último, o feedback imediato está presente em todas as plataformas.

Tabela 1. Características comuns das plataformas

	Greenfoot	**Scratch**	**GrameStation**
Disponibilidade	✓	✓	✓
Personalização	✓	✓	✓
Compartilhamento	✓	✓	×
Feedback imediato	✓	✓	✓

Conforme ilustrado na Tabela 2, os conceitos introduzidos pelas plataformas são: Programação Orientada a Objetos (Greenfoot), Lógica de Programação (Scratch) e Gramática de Grafos (GrameStation). Informações relativas aos conceitos trabalhados em Greenfoot e Scratch foram identificadas em [Henriksen e Kölling 2004] e [Resnick et al. 2009], respectivamente, também podendo ser encontradas em seus sites

oficiais. Sobre o público-alvo, não foram encontradas informações referentes ao GrameStation. Greenfoot e Scratch, de acordo com informações encontradas em seus sites, são recomendados para usuários a partir de 14 e 8 anos, respectivamente. Scratch ainda ressalta, em seu site, a linguagem ScratchJr[6], projetada para crianças entre 5 e 7 anos. No que se refere às linguagens, Greenfoot utiliza Java, o que, para os autores, produz efeitos em diversos aspectos da plataforma – enquanto alguns são benéficos, outros impõem limitações e complexidades. O domínio sobre uma linguagem textual é, em geral, consideravelmente mais complexo para usuários jovens, o que estabelece um nível mínimo de maturidade para sua manipulação. Scratch faz uso de blocos que podem ser arrastados pelo usuário. Esses blocos possuem conectores que sugerem sua organização. De acordo com os autores, tal mecanismo facilita o processo de criação de projetos por crianças. O GrameStation, por sua vez, utiliza grafos. Apesar de visual, a linguagem pode oferecer certo grau de dificuldade na especificação de jogos àqueles que não estão habituados a esse formalismo.

Tabela 2. Características específicas das plataformas

	Greenfoot	**Scratch**	**GrameStation**
Conceitos introduzidos	POO	Lógica de Programação	GG
Público-alvo	14 anos+	8 anos+	Não determinado
Linguagem	Java	Blocos	Grafos
Amigabilidade	Sim	Sim	Parcial
Conhecimento prévio	Orientação a Objetos	Programação Procedural	GG
Paradigma	Imperativo (Orientado a Objetos)	Imperativo (Orientado a Eventos)	Declarativo

Com relação ao quarto tópico, Greenfoot é considerado amigável pois, para os autores, mesmo fazendo uso de uma linguagem de programação padrão, a plataforma exclui diversos mecanismos encontrados em IDEs genéricas, como controle de versão, teste de unidade e refatoração [Henriksen e Kölling 2004]. Henriksen e Kölling (2004) ressaltam que o objetivo é promover aos alunos a adaptação completa com a interface em poucos dias. A fim de facilitar a experiência dos usuários com o Scratch, três princípios básicos de design foram estabelecidos: torná-lo mais manipulável, mais significativo e mais social em relação a outros ambientes de programação [Resnick et al. 2009]. A manipulação é obtida por meio da programação em blocos: mesmo que os alunos iniciem, simplesmente, conectando as estruturas em diferentes sequências e combinações e verificando o resultado, os blocos são moldados apenas para encaixarem-se de formas que façam sentido sintático. Para torná-lo mais significativo, dois critérios de design foram evidenciados: a diversidade, com a possibilidade de elaborar diferentes tipos de projetos (histórias, jogos e animações); e a personalização, visto que os usuários podem customizar seus projetos por meio da importação de recursos. Por último, a opção de compartilhamento das atividades foi estabelecida a fim de tornar a ferramenta mais social. Segundo os autores, para muitos usuários, a oportunidade de apresentar seus projetos a um grande público, bem como receber feedback e conselhos de outros usuários é um forte fator de motivação. Por fim, o GrameStation possui amigabilidade parcial pois, ainda que intuitivo, a especificação de uma GG pode não ser trivial a usuários que não possuem conhecimentos prévios.

O quinto tópico refere-se aos conhecimentos prévios necessários para a criação e

[6]Disponível em: https://www.scratchjr.org/.

o desenvolvimento de jogos nas plataformas. Greenfoot trabalha o paradigma de POO e, sendo assim, um conhecimento prévio a respeito desse assunto torna-se necessário. Para o Scratch, são necessários conhecimentos prévios sobre programação procedural, devido a manipulação de estruturas de sequência, decisão e iteração durante o desenvolvimento dos projetos. Para o GrameStation, é necessário conhecimento prévio em GG, uma vez que a especificação dos jogos na plataforma é baseada nesse formalismo.

Quanto ao último tópico, Greenfoot possui paradigma imperativo (orientado a objetos), trabalhando conceitos como objeto, herança, polimorfismo, entre outros; Scratch possui paradigma imperativo (orientado a eventos), trabalhando conceitos como ciclos, loops, entre outros; e GrameStation possui paradigma declarativo, trabalhando conceitos como grafos, regras de transformação de grafos, *match*, entre outros.

5. Considerações Finais

Este artigo apresenta a análise comparativa de três plataformas destinadas à criação de jogos utilizadas no ensino de computação: Greenfoot, Scratch e GrameStation. Embora possuam objetivos de aprendizagem distintos, as plataformas contam com diversas características semelhantes. Com base na análise, foi observada uma tendência em lidar/facilitar a questão da sintaxe em ambientes de desenvolvimento – reconhecida como um empecilho no aprendizado. A plataforma Greenfoot, por exemplo, é indicada para usuários acima de 14 anos pois, em determinada etapa da criação, é preciso lidar com a sintaxe da linguagem.

A plataforma GrameStation, embora baseada em um formalismo matemático – uma linguagem que, apesar de intuitiva, pode não ser trivial àqueles que não estão habituados –, possui diversas características comuns às demais plataformas. Tal fato pode ser interpretado como um esforço para a introdução da especificação formal na educação em computação. Segundo Silva Junior (2019), o processo para o desenvolvimento de conhecimentos associados à GG pode ser facilitado por meio da interatividade e de recursos visuais. Neste sentido, a plataforma apoia-se em recursos frequentemente defendidos na literatura, como o uso de linguagens visuais para facilitar a sintaxe e ambientes relacionados a jogos para manter a motivação. Como trabalho futuro, pretende-se expandir a presente análise, considerando plataformas que objetivam o aprendizado de demais conceitos da computação – seja por meio do desenvolvimento de jogos ou não – e, também, plataformas especificamente voltadas para o ensino/introdução de especificações formais.

Referências

Brezolin, C. V. S. e Silveira, M. S. (2021). Panorama Brasileiro de Uso de Ferramentas para Desenvolvimento do Pensamento Computacional e Ensino de Programação. In *Workshop sobre Educação em Computação (WEI)*, pages 398–407.

Doss, K., Juarez, V., Vincent, D., Doerschuk, P., e Liu, J. (2011). Work in Progress — A Survey of Popular Game Creation Platforms Used for Computing Education. In *Frontiers in Education Conference (FIE)*, pages F1H–1–F1H–2.

Egenfeldt-Nielsen, S. (2010). The Challenges to Diffusion of Educational Computer Games. *Leading Issues in Games Based Learning*.

Henriksen, P. e Kölling, M. (2004). Greenfoot: Combining Object Visualisation with Interaction. page 73–82. Association for Computing Machinery (ACM).

Kölling, M. (2010). The Greenfoot Programming Environment. Association for Computing Machinery (ACM).

Maloney, J., Resnick, M., Rusk, N., Silverman, B., e Eastmond, E. (2010). The Scratch Programming Language and Environment. *Transactions on Computing Education*, 10(4).

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., e Kafai, Y. (2009). Scratch: Programming for All. *Communications of the ACM*, 52(11):60–67.

Sentance, S. e Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Education and Information Technologies*, 22(2):469–495.

Silva Junior, B. A. (2020). GGasCT: Bringing Formal Methods to the Computational Thinking. Master's thesis, Universidade Federal de Pelotas, Pelotas, RS, Brasil.

Silva Junior, B. A., Cavalheiro, S. A. C., e Foss, L. (2019). Métodos Formais na Educação Básica: Operando Gramáticas de Grafos em um Jogo Educacional. *Workshop-Escola de Informática Teórica*, pages 178–186.

Wernbacher, T., Reuter, R., Denk, N., Pfeiffer, A., König, N., Fellnhofer, K., Grixti, A., Bezzina, S., e Jannot, E. (2020). Create Digital Games for Education: Game Design as a Teaching Methodology. In *Proceedings of ICERI2020 Conference*, volume 9.