Oficinas de Aprendizagem de Programação com Scratch e Python em um Curso de Engenharia de Computação

Camille L. Jesus[1], **Bianca L. Santana**[1], **Roberto A. Bittencourt**[1]

[1] UEFS – Universidade Estadual de Feira de Santana
Av. Transnordestina, s/n, Novo Horizonte
Feira de Santana – BA, Brasil – 44036-900

{camillejesus27,biancasantana.ls}@gmail.com, roberto@uefs.br

Abstract. *It is known that novice students have difficulties to learn programming. In the first programming course in higher education, such difficulties are more evident, and lead to high levels of dropout and failure. To deal with this issue in our institution, we held introductory programming workshops with Computer Engineering freshmen. We used a playful approach based on challenges and game building in Scratch and Python. Results point to the advantages of Python as the first language, students' high participation in the learning process, as well as their integration with classmates and senior students.*

Resumo. *É um fato conhecido da comunidade acadêmica que estudantes novatos apresentam dificuldades na aprendizagem de programação. Na primeira disciplina de programação do ensino superior, tais dificuldades são mais evidentes e levam a altos níveis de evasão e reprovação. Para amenizar este problema em nossa instituição, realizamos oficinas de programação introdutória com calouros do curso de Engenharia de Computação. Utilizamos uma abordagem lúdica baseada em desafios e construção de jogos em Scratch e Python. Os resultados apontam para as vantagens do uso de Python como primeira linguagem, a alta participação dos estudantes no processo de aprendizagem, além da integração destes com seus colegas e estudantes de períodos anteriores.*

1. Introdução

O ensino introdutório de programação tem sido um desafio para os cursos de graduação em computação, pois a programação usualmente é tida como algo difícil de se aprender. Esta é uma percepção apresentada por professores de modo geral e uma das justificativas que sustentam esta ideia são os índices de desistência e reprovação registrados em cursos de introdução à programação ao redor do mundo, incluindo universidades brasileiras [Watson and Li 2014, Bosse and Gerosa 2015]. Em nossa instituição, as dificuldades apresentadas pelos calouros durante a disciplina introdutória de programação do curso de Engenharia de Computação e os consequentes índices de reprovação também são causa de preocupação entre os professores que a ministram [Bittencourt et al. 2015].

A fim de diminuir as dificuldades iniciais enfrentadas por nossos calouros, desde 2013, passamos a oferecer uma oficina introdutória de programação antes do início de cada semestre. A oficina oferecida possui duração de uma semana e tem uma abordagem lúdica, baseada principalmente na criação de jogos com a ferramenta Scratch. Nas primeiras edições, o último dia de oficina era reservado para promover

a transição entre conceitos aprendidos no Scratch e a linguagem de programação C. Nosso grupo de pesquisa já apresentou à comunidade estudos a respeito destas oficinas [Bittencourt et al. 2015, Sena et al. 2015]. Cada edição da oficina é avaliada pelos participantes através de entrevistas e questionários, cujos resultados culminam em melhorias para as edições posteriores.

Recentemente, reformulamos a oficina, iniciando com dinâmicas de computação desplugada, trabalhando a programação em Scratch com jogos e, na etapa final, a programação em Python com a biblioteca gráfica Turtle. Neste contexto, relatamos nossas experiência com a nova versão da oficina, realizadas no segundo semestre de 2018 e no primeiro semestre de 2019. Dentre as lições aprendidas, destacamos: atividades desplugadas promovem uma introdução construtiva ao universo da programação; o uso do Scratch simplifica o ensino; Python é mais simples de aprender do que C; e não se trata apenas de aprender a programar, mas de entender a dinâmica do ambiente universitário.

2. Fundamentação Teórica

Existe todo um corpo do conhecimento sobre como ensinar programação efetivamente no nível superior [Jenkins 2002, Robins et al. 2003]. As soluções propostas normalmente alteram características do cursos de introdução à programação como currículo, pedagogia, escolha da linguagem ou ferramentas para apoiar a prática e a aprendizagem [Pears et al. 2007]. As abordagens propostas costumam interferir em um ou mais desses fatores, promovendo a colaboração e apoio entre pares, melhoria e contextualização do conteúdo, reconfiguração de recursos e da avaliação do curso, dentre outras [Vihavainen et al. 2014]. Em nossa oficina, nos concentramos em oferecer um contexto lúdico para motivar os estudantes e oferecer suporte para a introdução à programação. Assim, o apoio da literatura vem justificar nossas escolhas em relação ao Scratch, Python e da biblioteca gráfica Turtle.

Diversos trabalhos demonstram o potencial de Scratch em cursos introdutórios de programação em nível de graduação. Malan and Leitner (2007) utilizam Scratch em uma versão de verão de um curso de introduçao à programação na Universidade de Harvard e constatam que a ferramenta motiva os estudantes e familiariza os inexperientes com fundamentos de programação sem a distração da sintaxe . Mishra et al. (2014) descrevem uma intervenção de duas semanas com Scratch em um curso de introdução à programação com C++, ministrado para estudantes de engenharia . Os resultados obtidos demonstram que os estudantes novatos são capazes de acompanhar os estudantes mais avançados e a maioria deles reconhece que Scratch ajudou-os a aprender conceitos de programação, facilitando a aprendizagem de C++. As vantagens de Scratch também já foram estudadas a partir de versões anteriores de nossa oficina. O resultados da pesquisa realizada com base na edição de 2014 demonstraram que, embora a maioria dos estudantes não tenham utilizado Scratch após a oficina, eles acreditam que o uso da ferramenta os ajudou com a disciplina de introdução à programação com C, principalmente na aprendizagem de estruturas de controle e operações lógicas [Bittencourt et al. 2015, Sena et al. 2015].

Diversos trabalhos também demonstram o potencial do uso de Python como primeira linguagem. Enbody et al. (2009) analisam a viabilidade do uso do Python em um curso introdutório de programação no nível superior, e comprovam que Python preparou os estudantes para o segundo curso da sequência, ministrado em C++, promovendo a fácil

transição entre linguagens . Shannon (2003) descreve o uso de Python como alternativa as linguagens C++ e Java em um curso introdutório de programação . Os resultados demonstram que Python facilita o ensino da programação devido à sua sintaxe mais simples e, nas disciplinas subsequentes, muitos alunos optariam por usar Python na escrita de seus programas se eles fossem autorizados a usar a linguagem de sua escolha.

3. Organização da Oficina

Nesta seção, descrevemos as ferramentas, os participantes e o planejamento da oficina.

3.1. Ferramentas

Scratch é uma ferramenta de programação visual, criada pelo *MIT Media Lab*. O processo de programação em Scratch é facilitado através de sua coleção de "blocos gráficos de programação" que são moldados para se encaixarem apenas em formas que fazem sentido sintático [Resnick et al. 2009]. Scratch é uma ferramenta amplamente utilizada para fins educacionais, pois além de facilitar a aprendizagem de programação, permite a construção de jogos, simulações, e animações, dentre outros contextos.

Python é uma linguagem de programação de propósito geral, que combina os paradigmas procedural, funcional e orientado a objetos. É uma linguagem muito utilizada por programadores profissionais e seu uso como primeira linguagem em cursos introdutórios de programação vem se mostrando uma alternativa viável em comparação as linguagens mais tradicionais como C, C++ e Java. Dentre as vantagens oferecidas por Python como primeira linguagem estão: a sintaxe mais simples, o reforço à indentação, o fato de ser interpretada, tipagem dinâmica de variáveis, os *loops for* podem variar sobre coleções, dentre outras vantagens [Agarwal and Agarwal 2005].

A biblioteca gráfica Turtle é uma implementação semelhante à tartaruga da linguagem de programação Logo [Papert 1980]. A essência do Turtle está nos comandos e no modo como as construções exigem conceitos básicos de lógica de programação, a medida em que a complexidade da figura desenhada aumenta. Vidal Duarte (2016) identifica melhoria nas notas dos estudantes que participaram de uma abordagem de ensino de programação no nível superior utilizando Python com Turtle .

3.2. Participantes

Os participantes foram convidados durante a matrícula na universidade, de modo que a participação não foi obrigatória. No início da oficina, os estudantes concordaram em participar da pesquisa, assinando um termo de consentimento livre e esclarecido. Os dados foram colhidos através da aplicação de dois questionários, pré- e pós-intervenção, com o intuito de obter uma avaliação dos estudantes em relação a oficina.

A oficina do segundo semestre de 2018 foi mediada por sete graduandos em Engenharia de Computação e contou com 21 participantes, dos quais 19 (90,5%) eram do sexo masculino e dois (9,5%), do sexo feminino. A idade média desses estudantes era de $18{,}19 \pm 1{,}89$ anos. Dos participantes, apenas seis relataram ter algum conhecimento prévio de programação e apenas um já conhecia Scratch.

A oficina do primeiro semestre de 2019 foi mediada por oito graduandos em Engenharia de Computação e contou com 26 participantes, dos quais 20 (76,9%) eram do

sexo masculino e seis (23,1%), do sexo feminino. A idade média desses estudantes era de 18,23±1,53 anos. Dos participantes, apenas oito relataram ter algum conhecimento prévio de programação e apenas dois já conheciam o Scratch.

3.3. Planejamento

A oficina foi planejada com carga horária de 20 horas, distribuídas em cinco sessões de quatro horas cada. Em cada edição, são escalados um tutor e dois ou mais monitores para auxiliarem os estudantes nas atividades. O tutor e os monitores são estudantes veteranos do curso de Engenharia de Computação. Esta escolha tem como objetivo facilitar a integração dos calouros à nova realidade.

O conteúdo é dividido em três etapas: primeiro, trabalhamos lógica de programação e criação de algoritmos, em seguida, trabalhamos a programação com Scratch e, na ultima etapa, trabalhamos programação com a linguagem Python. A Tabela 1 apresenta em detalhes o planejamento da oficina para cada dia. A primeira aula tem um tom introdutório e busca fazer com que os estudantes aprendam algoritmos. Os dias seguintes são dedicados a construções de projetos de games, enquanto os dois últimos dias são reservados a uma introdução à linguagem Python.

Tabela 1. Planejamento da oficina.

Dia	Objetivos	Conteúdos	Atividades
1	- Entender o funcionamento dos computadores; - Conhecer os principais fundamentos da programação; - Desenvolver o pensamento computacional; - Trabalhar com representação de algoritmos.	- Algoritmos; - Lógica de programação; - Fluxogramas.	- 2 dinâmicas de fluxograma humano; - 2 dinâmicas de ordenação de fluxogramas; - 2 dinâmicas de construção de fluxogramas.
2	- Conhecer a ferramenta Scratch; - Criar uma animação; - Implementar um jogo básico.	- Estruturas de controle; - Movimentação de atores; - Troca de trajes; - Sensores.	- Animação livre; - Jogo Pong.
3	- Implementar jogos no Scratch; - Usar condições e repetições de maneira consistente; - Usar variáveis de maneira consistente.	- Controle de movimentação; - Sensores; - Condicionais; - Comunicação entre objetos; - Operadores; - Variáveis.	- Jogos Arqueiro e Interlagos.
4	- Conhecer a sintaxe básica da linguagem Python; - Criar desenhos e figuras com a biblioteca Turtle.	- Movimentação da Turtle; - Funções da Turtle; - Funções em Python; - Condições; - Repetições.	- Desenho de formas geométricas; - Desenho de tabuleiro de xadrez.
5	- Aprender outros comandos do Python; - Implementar programas com interação com o usuário; - Organizar adequadamente a estrutura de um programa.	- Variáveis; - Operadores aritméticos; - Condições; - Repetições; - Funções (parâmetros e retorno); - Entrada/Saída de dados.	- Calculadora com as quatro operações básicas; - Jogo de adivinhação.

Nossa abordagem, além de lúdica, também trata os conteúdos de maneira repetida em cada uma das etapas, o que remete à abordagem de aprendizagem em espiral. Nos dias em que trabalhamos com a linguagem Python, repetimos parte dos conceitos já trabalhados com a programação em Scratch, mas em um nível de complexidade maior. Acreditamos que este tipo de abordagem permite uma revisão do conteúdo, oferecendo maior suporte para que as habilidades sejam aprofundadas nos dias subsequentes.

Durante as explicações, não utilizamos formalismos e oferecemos descrições breves e objetivas. A participação dos estudantes é incentivada o tempo todo pelo instrutor e pelos monitores. Durante as sessões, as explanações teóricas são curtas e a maior parte do tempo é dedicado às atividades práticas em laboratório. Cada aluno constrói a sua própria solução para os problemas dados com o auxílio de monitores, caso necessário.

No primeiro dia de oficina, trabalhamos com Algoritmos e Lógica de Programação. São propostas seis atividades em formato de dinâmicas desplugadas realizadas em grupos. As atividades promovem a introdução à lógica de programação através do desenvolvimento de fluxogramas. No segundo e terceiro dias de oficina, trabalhamos com a ferramenta Scratch no contexto de criação de animações e jogos. No segundo dia, propomos a criação de uma animação de tema livre e o desenvolvimento do jogo Pong. No terceiro dia, propomos a implementação dos jogos clássicos Interlagos e Arqueiro. Nesta fase, são introduzidas as estruturas de controle, variáveis e as diversas operações de interações entre objetos que Scratch proporciona (movimento, aparência, sensores). No terceiro e quarto dias de oficina, o conhecimento adquirido é reforçado e é feita a transição entre linguagem visual (Scratch) e linguagem textual (Python). No quarto dia, trabalhamos com a biblioteca gráfica Turtle em atividades de desenhos de figuras geométricas. No quinto dia, os alunos desenvolvem os projetos de uma calculadora e de um jogo da adivinhação, sem o uso da biblioteca gráfica.

4. Nossa Experiência

As oficinas foram mediadas por uma tutora, uma das autoras deste trabalho. As sessões ocorreram em um laboratório de informática, com 19 computadores disponíveis. A prioridade era que cada estudante ficasse em um computador, mas, devido a quantidade de participantes ser superior à de máquinas disponíveis para uso em ambas as edições, alguns estudantes trabalharam em dupla.

Nas subseções a seguir, relatamos os resultados diários extraídos pela tutora em um diário de bordo e os resultados para alguns construtos dos questionários aplicados. Os resultados da Subseção 4.1 são apenas do primeiro semestre de 2019, enquanto os resultados apresentados na Subseção 4.2 são referentes aos dois semestres em que a nova versão da oficina foi ministrada.

4.1. Dinâmica da Oficina

Descrevemos, a seguir, a dinâmica de uma das oficinas em uma narrativa temporal.

4.1.1. Primeiro Dia – Algoritmos e Lógica de Programação

No início da oficina, fizemos uma breve explanação sobre o funcionamento geral de computadores, linguagens de programação e apresentamos o conceito de algoritmos e

exemplos. Realizamos duas dinâmicas desplugadas intituladas fluxograma humano. A dinâmica consistiu em separar os participantes em grupos, em que cada grupo um desempenha uma função: interpretador, memória e instruções. O interpretador era o responsável pela execução do algoritmo, instrução por instrução, e a memória armazenava os valores das variáveis. Separamos os estudantes em grupos que simulavam a execução dos algoritmos para encontrar o módulo e o fatorial de um número.

Na segunda parte da manhã, tratamos da formalização do conceito de fluxogramas e sua simbologia. Também trabalhamos, brevemente os conceitos de variáveis, valores lógicos, estruturas de decisão, repetições e entrada/saída de dados. Realizamos duas dinâmicas em que os participantes tinham que ordenar as partes de um fluxograma, passar um algoritmo do fatorial de um número da forma de descrição narrativa para um fluxograma, e criar o fluxograma de um algoritmo para calcular a média e a situação final (aprovado/reprovado) de um aluno. Nesta dinâmica, os estudantes apresentaram maiores dificuldades pois um dos requisitos do algoritmo era que utilizassem apenas duas variáveis para armazenar as três notas, de maneira que se fez necessário o uso de repetições e acumuladores. Muitos estudantes relataram que as dinâmicas desplugadas proporcionaram momentos em que tiveram mais oportunidade de conhecer e se relacionar com os colegas.

4.1.2. Segundo e Terceiro Dias – Scratch

No segundo dia, iniciamos o período de utilização de Scratch. Na primeira parte da aula, fizemos uma curta exposição do funcionamento geral da ferramenta e seus comandos, com ênfase nas estruturas de controle, movimentação de atores e troca de trajes. Em seguida, os estudantes foram desafiados a criarem sua primeira animação em Scratch.

As atividades subsequentes foram todas voltadas para a implementação de jogos, onde introduzimos também os conceitos de condições, *loops*, operadores e variáveis. A tutora apresentou as regras de funcionamento do jogo Pong e desafiou os estudantes a o implementarem. Durante a implementação, alguns estudantes apresentaram dificuldades em relação à utilização correta dos comandos de repetição. Alguns utilizaram o comando *Repita* ao invés do *Sempre* para movimentos que deveriam ocorrer indefinidamente, outros utilizaram o *Sempre* em ações que ocorriam uma única vez. Devido a estas dificuldades, apenas 65,4% dos participantes relataram que conseguiram finalizar a implementação do projeto no tempo reservado da aula.

No terceiro dia, os estudantes implementaram os jogos Arqueiro e Interlagos. Durante a implementação do jogo Arqueiro, muitos estudantes adicionaram atores diferentes para cada traje do arqueiro em vez de adicionar vários trajes a um mesmo ator. A maioria dos participantes também tiveram dificuldades em alterar a direção do movimento do ator balão, o que indica dificuldades com fluxo de execução e eventos. 76,9% dos estudantes conseguiram implementar o projeto do arqueiro ainda em sala. Durante a implementação do jogo Interlagos, muitos estudantes apresentaram dificuldades em implementar a colisão entre atores. Os estudantes também apresentaram dificuldades relacionadas à lógica de pontuação do jogo, que exigia a manipulação de sensores e variáveis. Mesmo após a intervenção da tutora, ao perceber que havia uma dúvida generalizada, parte dos estudantes continuou em dificuldades, de modo que apenas 50% dos participantes relataram que conseguiram finalizar o projeto do Interlagos ainda em sala.

4.1.3. Quarto e Quinto Dias – Python

No quarto dia, iniciamos com a linguagem Python e os comandos da biblioteca Turtle, para movimentação da tartaruga e realização de desenhos e figuras geométricas. A tutora iniciou a sessão criando o código para desenhar um quadrado. Neste momento, a participação dos estudantes foi incentivada. Após este primeiro exemplo, a tutora solicitou que que eles implementassem sozinhos os desenhos de retângulo, triângulo, hexágono e octógono pintados. Inicialmente, os estudantes apresentaram dificuldade em determinar o ângulo que a caneta deveria girar para fazer determinados desenhos de modo que a tutora precisou intervir, explicando para a turma uma maneira de encontrar o ângulo da figura com base na quantidade de lados. Após esta atividade, introduzimos os conceitos de *loops* e funções, e solicitamos aos estudantes que aplicassem estes conceitos ao código elaborado. Os estudantes não apresentaram dificuldades aparentes para aplicar estes conceitos. Na segunda parte da aula, os estudantes foram desafiados a criarem um tabuleiro de xadrez. Mesmo utilizando repetições e funções, eles tiveram dificuldades em deixar o código do tabuleiro mais "enxuto" e muitos não conseguiram incluir o uso de condições. Entretanto, todos conseguiram finalizar a atividade.

No último dia de oficina, O conteúdo de funções foi novamente abordado, desta vez, com ênfase nos parâmetros e retorno de funções. Após uma breve explicação, dois desafios foram propostos: o da calculadora e o jogo da adivinhação. No desafio da calculadora, os estudantes tiveram que implementar um programa que realiza as operações de adição, subtração, multiplicação e divisão, a partir de uma entrada do usuário. A turma não apresentou dificuldades com esta atividade e utilizaram corretamente os conceitos de funções, sendo que 96,2% do participantes conseguiram concluir essa atividade. No desafio do jogo de adivinhação, os estudantes implementaram um programa em que o usuário tenta adivinhar um número sorteado pelo computador. Este desafio apresentou maior nível de complexidade, demandando o uso de estruturas condicionais e *loops*. As dúvidas apresentadas durante esta atividade envolviam o fluxo de execução do programa. Durante a utilização de Python, a sintaxe da linguagem não foi percebida como empecilho à aprendizagem dos estudantes.

4.2. Avaliação

Nesta seção, apresentamos os resultados quantitativos obtidos a partir da aplicação de questionários com os estudantes. Através dos questionários aplicados, percebemos quais as dificuldades conceituais enfrentadas durante o uso de Scratch e Python.

A Figura 1 exibe a percepção dos estudantes sobre a facilidade que tiveram para usar as categorias de comandos de Scratch. Comandos dos grupos de Sensores e Operadores foram mais difíceis que os das demais categorias. No entanto, em todos eles, ao menos 65% dos estudantes afirmaram ter muita facilidade em usar os comandos trabalhados.

A Figura 2 exibe a percepção dos estudantes sobre a facilidade que tiveram para usar os comandos de Python. Os comandos de funções, parâmetros de funções e uso do *while* foram mais difíceis que os demais.

Em outras questões, os estudantes escolheram quais as atividades mais motivadoras realizadas na oficina. As atividades com Turtle e o Jogo da Adivinhação, ambas em Python, foram apontadas como as atividades mais motivadoras.

Figura 1. Facilidade em usar comandos em Scratch.

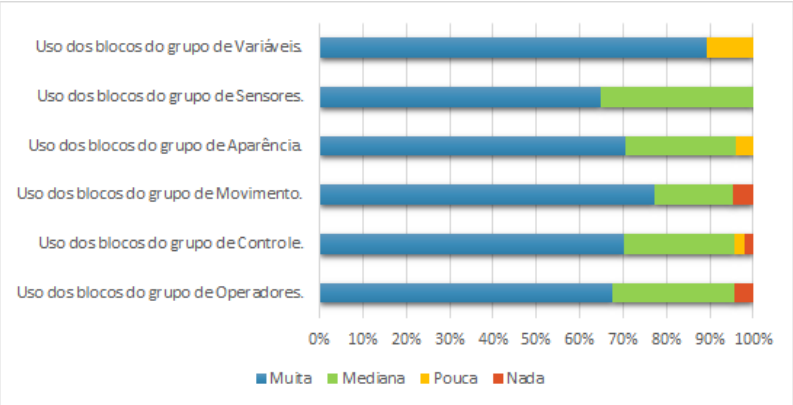
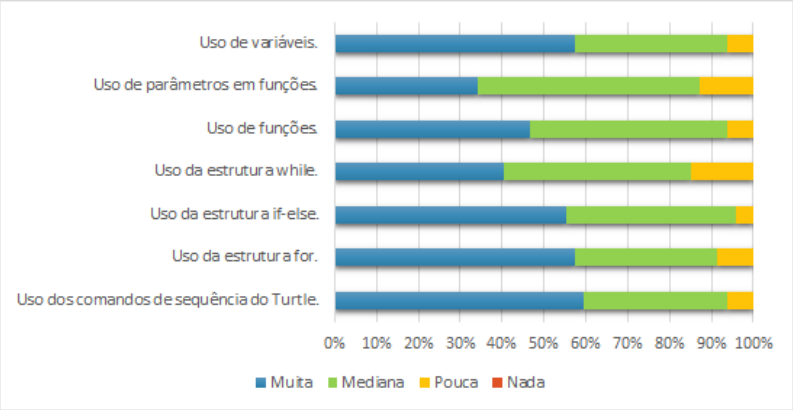


Figura 2. Facilidade em usar comandos em Python.



5. Lições Aprendidas

Não é objetivo deste trabalho avaliar as vantagens e/ou desvantagens da nova versão da oficina em relação à versão antiga, mas entender como esta nova abordagem pode ser útil para os estudantes calouros de cursos de computação. Neste sentido, algumas lições podem ser destacadas:

Atividades desplugadas promovem uma introdução construtiva ao universo da programação. A utilização de atividades desplugadas proporciona um primeiro contato com problemas computacionais que exigem o uso da criatividade e da lógica, sem a necessidade de utilizar programas específicos, mas vivenciando as ideias de algoritmos e programas.

O uso de Scratch simplifica a atividade de ensino. O ambiente Scratch foi planejado para aprender através da exploração. Assim, nas atividades com Scratch, usamos uma abordagem de desafios de implementação, onde os estudantes tentam resolvê-los através da exploração da paleta de comandos e por tentativa e erro, apoiados no feedback imediato oferecido pela ferramenta. Com isso, os tutores não precisam realizar longas explanações sobre os conceitos de programação.

Python é mais fácil de aprender do que C. Comparado com oficinas anteriores, que utilizavam a linguagem C, o uso de Python como primeira linguagem de programação textual levou os estudantes a apresentarem menos dificuldades na construção dos progra-

mas [Bittencourt et al. 2015, Sena et al. 2015]. Além disso, acharam as atividades com Python mais motivadoras do que as atividades com Scratch, o que contrasta com as oficinas anteriores com C.

Não se trata apenas de aprender a programar, mas de entender a dinâmica do ambiente universitário. Além do objetivo de aprendizagem de programação, a oficina é também um momento de integração entre calouros com os estudantes veteranos e com o próprio ambiente universitário. A interação entre estudantes de diferentes estágios do curso é bastante benéfica no sentido de tornar a oficina um ambiente mais divertido, acolhedor e repleto de experiências diversas.

6. Conclusões

Neste artigo, relatamos nossa experiência com a versão reformulada de nossas oficinas de introdução à programação para calouros do curso de Engenharia de Computação em nossa instituição. As oficinas oferecidas possuem duração de uma semana e utilizam uma abordagem lúdica, que combina a realização de dinâmicas desplugadas, o uso da ferramenta Scratch em um contexto de criação de animações e jogos, e da linguagem de programação Python, no contexto de desenhos geométricos com a biblioteca gráfica Turtle e a implementações de pequenos projetos.

Nossas percepções iniciais revelam que a oficina facilita o primeiro contato dos estudantes calouros com a programação. Além disso, o trabalho inicial com dinâmicas desplugadas potencializou o aprendizado, ajudando na transição entre as ferramentas e linguagens utilizadas. Dentre as lições aprendidas, elencamos: atividades desplugadas promovem uma introdução construtiva ao universo da programação; o uso de Scratch simplifica a atividade de ensino; Python é mais fácil de aprender do que C; e oficina oferece não trata apenas de aprender a programar, mas de apresentar a dinâmica do ambiente universitário aos estudantes recém-chegados.

Como trabalhos futuros, pretendemos fazer uma análise aprofundada a respeito dos impactos da nossa oficina ao longo da disciplina de introdução à programação. Acreditamos também que a replicação deste tipo de oficina em outras instituições pode enriquecer o conhecimento sobre dificuldades de aprendizagem de programação e reduzir potencialmente os índices de evasão e reprovação em disciplinas de introdução à programação na educação superior.

7. Agradecimentos

Este projeto foi apoiado pelo programa PROBIC/UEFS, através de bolsa de iniciação científica, e pela FAPESP, através do auxílio de pesquisa 2015/24331-1 e da bolsa de treinamento técnico 2018/12799-7.

Referências

Agarwal, K. K. and Agarwal, A. (2005). Python for CS1, CS2 and beyond. *Journal of Computing Sciences in Colleges*, 20(4):262–270.

Bittencourt, R. A., Santos, D. M. B., Rodrigues, C. A., Batista, W. P., and Chalegre, H. S. (2015). Learning programming with peer support, games, challenges and Scratch. In *Frontiers in Education Conference (FIE), 2015*. IEEE.

Bosse, Y. and Gerosa, M. A. (2015). As disciplinas de introdução à programação na USP: um estudo preliminar. In *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, volume 4, page 1389.

Enbody, R. J., Punch, W. F., and McCullen, M. (2009). Python CS1 as preparation for C++ CS2. *ACM SIGCSE Bulletin*, 41(1):116–120.

Jenkins, T. (2002). On the difficulty of learning to program. In *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, volume 4, pages 53–58.

Malan, D. J. and Leitner, H. H. (2007). Scratch for budding computer scientists. In *ACM SIGCSE Bulletin*, volume 39, pages 223–227. ACM.

Mishra, S., Balan, S., Iyer, S., and Murthy, S. (2014). Effect of a 2-week Scratch intervention in CS1 on learners with varying prior knowledge. In *Proc. 2014 Conference on Innovation & technology in Computer Science Education*, pages 45–50. ACM.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc.

Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., Devlin, M., and Paterson, J. (2007). A survey of literature on the teaching of introductory programming. *ACM SIGCSE Bulletin*, 39(4):204–223.

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

Robins, A., Rountree, J., and Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, 13(2):137–172.

Sena, J. P. S., Rios, M. S., Batista, W. P., Santos, D. M. B., and Bittencourt, R. A. (2015). Reduzindo Dificuldades em Introdução à Programação em um Curso de Engenharia de Computação através de Ambiente Lúdico Scratch. In *COBENGE 2015 - XLIII Congresso Brasileiro de Educação em Engenharia*.

Shannon, C. (2003). Another breadth-first approach to CS I using Python. In *ACM SIGCSE Bulletin*, volume 35, pages 248–251. ACM.

Vidal Duarte, E. (2016). Teaching the first programming course with Python's Turtle graphic library. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '16, New York, NY, USA. ACM.

Vihavainen, A., Airaksinen, J., and Watson, C. (2014). A systematic review of approaches for teaching introductory programming and their influence on success. In *Proceedings of the Tenth Annual Conference on International Computing Education Research*.

Watson, C. and Li, F. W. (2014). Failure rates in introductory programming revisited. In *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education*, pages 39–44. ACM.