INFORMÁTICA NA EDUCAÇÃO: teoria & prática Porto Alegre, v.20, n.2, mai./ago. 2017. ISSN digital 1982-1654
ISSN impresso 1516-084X

Desenvolvimento do Pensamento Computacional com Valores da Ética Hacker

Computational Thinking with Hacker's Ethics

FLÁVIA LINHALIS ARANTES
Universidade Estadual de Campinas (UNICAMP)

PAULA EDUARDA JUSTINO RIBEIRO
Universidade Estadual de Campinas (UNICAMP)

Resumo: A introdução de conceitos de computação aos jovens é importante pelo seu caráter transversal às demais áreas do conhecimento. Com o objetivo de estimular o desenvolvimento do pensamento computacional, este artigo apresenta o design e a avaliação de uma oficina com a utilização do software livre Scratch realizada no Projeto Jovem Hacker. O Projeto busca maneiras de auxiliar na formação de jovens tendo como base a ética hacker, importante para orientar os alunos a ter um senso crítico e uma ética mais voltada ao compartilhamento de ideias e soluções de software. A oficina descrita poderá ser replicada em outros contextos em que o Scratch for utilizado e o material produzido poderá ser reutilizado e remixado em projetos similares.

Palavras-chave: Pensamento Computacional. Ética Hacker. Scratch.

Abstract: The introduction of computing concepts to young people is important because of its interdisciplinary character to other areas of knowledge. With the aim of stimulating the development of computational thinking, this article presents the design and evaluation of a Scratch course in the Jovem Hacker Project. The project studies ways to assist in the training of young people on the basis of the hacker ethic, important to guide the students to have a critical sense and a more openness to sharing ideas and software solutions. The workshop can be replicated to other contexts where the Scratch is used and the material produced can be reused and remixed in similar projects.

Keywords: Computational Thinking. Hacker Ethics. Scratch.

ARANTES, Flávia Linhalis; RIBEIRO, Paula Eduarda Justino. Desenvolvimento do Pensamento Computacional com Valores da Ética Hacker. *Informática na Educação*: teoria & prática, Porto Alegre, v. 20, n. 2, p. 188-206, mai./ago. 2017.

1 Introdução

Atualmente, muitos referem-se aos jovens como "nativos digitais" devido a uma aparente fluidez com a tecnologia. De fato, a maioria deles sente-se confortável ao usar dispositivos diversos – mandam mensagens, usam jogos online, navegam na Internet. Mas isso os torna realmente fluentes com as novas tecnologias? Apesar de interagirem muito com mídias digitais, poucos são capazes de criar seus próprios jogos, animações ou simulações. É como se pudessem "ler", mas não "escrever". De acordo com Resnick e colegas (2009), fluência digital não se trata apenas de trocar mensagens, navegar e interagir usando o computador, mas também de ter a habilidade de imaginar, projetar e criar com novas mídias (RESNICK et al., 2009).

Mas para fazer isso é preciso aprender a programar. Com o intuito de contribuir com a formação de uma geração que seja mais capacitada tecnologicamente, surgiu o Projeto Jovem Hacker[1], que investiga maneiras de auxiliar na formação de uma geração mais autônoma tecnologicamente e se empodere dos rumos da nossa sociedade, tendo como base a ética *hacker* e o software livre (ARANTES et al., 2014; AMIEL et al., 2015).

O objetivo não é necessariamente preparar os jovens para carreiras profissionais da programação e da computação, mas nortear uma geração que seja mais criativa, com um pensamento mais sistemático, que se sinta confortável para expressar suas ideias (RESNICK et al., 2009).

O desenvolvimento do pensamento sistemático que a programação proporciona é chamado de "pensamento computacional". Cuny, Snyder e Wing (2010) definem pensamento computacional como sendo "*os processos de pensamento envolvidos na formulação de problemas e suas soluções, sendo que as soluções são representadas de modo a serem efetivamente realizadas por um agente de processamento de informações*". Uma ferramenta que tem se destacado nessa direção é o software livre Scratch[2]. De acordo com Brennan e Resnick (2012), ao programar e compartilhar projetos interativos em Scratch, crianças e adolescentes aprendem conceitos computacionais, além de aprenderem a pensar de maneira criativa e a trabalhar colaborativamente – habilidades importantes atualmente. Programar com Scratch, portanto, pode oferecer um contexto e um conjunto de oportunidades para contribuir com o desenvolvimento do pensamento computacional.

Na literatura, existem vários relatos do uso do Scratch com alunos do ensino fundamental e médio iniciantes em programação (AURELIANO e TEDESCO, 2012; SCAICO et al., 2013; FRANÇA e AMARAL, 2013; RODRIGUES et al., 2015). No entanto, além de ensinar programação, é preciso orientar os alunos a ter um senso crítico e uma ética que nortearão a maneira como irão lidar com o software e a tecnologia. Neste artigo, apresentamos o *design* de uma oficina de Scratch, no contexto do Projeto Jovem Hacker, o qual busca maneiras de auxiliar na formação de jovens considerando a ética *hacker*. Além de mostrar o *design* da oficina, consideramos importante também ter a avaliação da aprendizagem como uma prática

[1] Sítio oficial: http://jovemhacker.org.
[2] Sítio oficial: http://scratch.mit.edu.

contínua que busca nortear a orientação aos aprendizes durante o processo de formação, bem como servir de elemento balizador para futuras edições do Projeto Jovem Hacker.

Dois assuntos são abordados neste artigo: (1) o *design* de uma oficina de Scratch com conceitos da ética *hacker* e (2) uma abordagem avaliativa para a oficina, que pode ser adaptada a outros contextos de formação com Scratch.

2 O Projeto Jovem Hacker

De acordo com Amiel e colegas (2015), o projeto Jovem Hacker, fazendo jus ao seu nome, é fortemente alicerçado em uma cultura de partilha, busca do conhecimento e valorização da liberdade, pilares essenciais da cultura *hacker*.

Os *hackers* não programam necessariamente por ganhos financeiros, mas porque sentem prazer em fazê-lo. Linus Torvalds costuma dizer que "*para um hacker, o computador por si só já é uma diversão"* (HIMMANEN, 2001, p. 19). Muitos bons projetos podem ser resultado desse tipo de atividade, como ocorreu com o sistema operacional Linux, a World Wide Web e tantas outras criações relevantes (HIMMANEN, 2001).

No projeto Jovem Hacker, criamos uma estrutura de curso focada na apropriação tecnológica que perpassa o hardware e o software. Com base em uma noção emancipatória da tecnologia, uma ética *hacker* fundamentada no software livre e no pensamento computacional, o Projeto propõe a estrutura de um curso através das seguintes oficinas modulares (ARANTES et al., 2014; AMIEL et al., 2015):

i) Arquitetura básica de computadores e de software.
ii) Lógica de programação ou introdução ao pensamento computacional com Scratch.
iii) Desenvolvimento web com HTML, CSS e JavaScript.
iv) Introdução de uma linguagem de programação para web (e além) utilizando Python.
v) Como momento final da formação é definido, em conjunto com os alunos, um projeto de interesse coletivo ou individual, desenvolvido com base nas atividades apresentadas nas oficinas.

Em 2015, o Projeto Jovem Hacker aconteceu em paralelo em duas cidades e dois contextos diferentes, enriquecendo nossas bases para pesquisa e melhoria da proposta – Edição Cultura Campinas e Edição IFSP Capivari. Mais informações sobre as edições do Projeto e sobre o currículo podem ser obtidas em Amiel e colegas (2015).

Neste artigo, apresentamos e discutimos a oficina de Scratch que aconteceu na Edição Cultura Campinas, entre maio e dezembro de 2015. O material produzido nessa edição está disponível no sítio do Projeto[3].

3 Trabalhos Relacionados

Quando os computadores foram introduzidos nas escolas, na década de 80, houve um entusiasmo para ensinar crianças e adolescentes a programar. Papert apresentou a linguagem Logo como uma nova maneira de pensar a educação e o aprendizado (PAPERT, 1980). O

[3] Material disponível no sítio do Projeto em http://wiki.jovemhacker.org/index.php/Material

construtivismo de Papert colocava o computador como uma ferramenta que a criança podia controlar e programar, a linguagem Logo tornou-se uma poderosa aliada na construção do conhecimento.

Entretanto, aquele entusiasmo inicial durou pouco. A maioria das escolas passou a utilizar o computador para outras tarefas, tais como criar textos e navegar na Internet. Desde então, a tecnologia evoluiu muito e os computadores diminuíram ainda mais de tamanho – dispositivos móveis tornaram-se parte da rotina de crianças e adolescentes.

Nos últimos anos, as tentativas de introduzir programação para crianças e adolescentes têm voltado à tona. Algumas iniciativas surgiram nos Estados Unidos, com projetos como code.org[4], pois estima-se que em poucos anos faltará mão de obra qualificada para atuar no mercado de trabalho americano na área de programação.

No Brasil surgiram algumas iniciativas que são basicamente extensões de projetos dos Estados Unidos. Um exemplo é o Projeto Code Club[5], cujo objetivo é oferecer a oportunidade para crianças aprenderem a programar. Para isso, o projeto oferece material de ensino para uma rede de voluntários apoiar a realização de atividades extracurriculares ligadas à programação de computadores. A primeira parte do projeto utiliza Scratch para estimular o desenvolvimento do pensamento computacional.

Alguns projetos introduzem a programação por meio de oficinas como forma complementar ao ensino. No trabalho de França e Amaral (2013), os autores descrevem a realização de uma oficina em Scratch para estimular o pensamento computacional em estudantes do ensino básico de uma escola pública de Pernambuco. Os autores aplicam e avaliam conceitos computacionais propostos por Brennan e Resnick (2012) e apresentam resultados positivos que podem ser explorados com o uso do Scratch para disseminar o pensamento computacional, em conjunto com avaliação contínua.

O trabalho desenvolvido por Scaico e colegas (2013) utilizou Scratch com o intuito de ensinar programação a alunos do ensino médio do interior da Paraíba. O artigo relata uma olimpíada de programação, onde os alunos foram estimulados a enfrentar diversas situações que envolviam conceitos computacionais e exigiam esse conhecimento para a solução de problemas.

No trabalho de Rodriguez e colegas (2015), os autores descrevem um projeto com o objetivo de desenvolver noções básicas do pensamento computacional junto a sete alunos do primeiro ano do ensino médio de uma escola pública, por meio dos recursos do Scratch, no âmbito de um programa de Pré-Iniciação Científica. Como resultado, os autores destacam três jogos desenvolvidos pelos alunos que permitiram verificar como se apropriaram dos recursos cognitivos inerentes ao pensamento computacional.

O Projeto Jovem Hacker não tem como objetivo necessariamente ensinar a programar. Seu diferencial é justamente entender as diferentes metodologias e objetivos do “aprender a programar”. O projeto procura despertar no jovem a habilidade de partir de algo que existe, para então fuçar, remixar e alterar para outros propósitos (AMIEL et al., 2015). Aliado a isso, o

[4] Sítio oficial: https://code.org.
[5] Sítio oficial: http://codeclubbrasil.org.

projeto também está fortemente alicerçado a uma ética *hacker*, que acreditamos contribuir com uma geração tecnologicamente mais independente e que possa fazer parte de um futuro com mais compartilhamento e abertura tecnológica.

Na próxima seção, abordaremos o conceito de ética *hacker*, sua relação com o Projeto Jovem Hacker e, consequentemente, com a oficina de Scratch apresentada neste artigo.

4 A Ética Hacker

No mundo da tecnologia, muito se ouve falar sobre *hackers*. Uma definição muito utilizada para o termo diz que *hackers* são "*indivíduos que se dedicam com entusiasmo à programação, que acreditam que o compartilhamento de informações é um bem poderoso e positivo*" (HIMMANEN, 2001). Muitos confundem o termo *hacker* com *cracker*. Esse segundo é o termo usado para designar quem pratica a quebra (ou *cracking*) de um sistema de segurança, usam seu conhecimento de forma ilegal, portanto, são vistos como criminosos[6].

Ética pode ser definida como um conjunto de valores morais e princípios que norteiam a conduta humana na sociedade. Pekka Himanem (2001) descreveu a ética dos *hackers* através de sete valores que considerou importantes para a maioria dos *hackers*:

(i) Paixão: o fator gerador de alegria e motivação para o trabalho dos *hackers* é a paixão, o que justifica o trabalho empregado no alcance de seus objetivos.

(ii) Liberdade: a ética de trabalho dos *hackers* consiste em combinar paixão com liberdade.

(iii) Valor Social: os *hackers* não dão prioridade aos lucros, mas sim em criar soluções que sejam úteis para a sociedade como um todo. Ao mesmo tempo, também buscam destaque dentro das comunidades em que atuam e esperam reconhecimento pelas suas criações.

(iv) Abertura: muitos *hackers* disponibilizam suas criações gratuitamente para que outras pessoas possam utilizá-las e melhorá-las. Esse compartilhamento contribui para que as soluções fiquem mais completas e robustas, além de aumentar o alcance de suas criações.

(v) Atividade: os *hackers* defendem a ideia de que a comunidade deve ser participativa e auxiliar na construção dos ambientes de informação.

(vi) Consideração: os *hackers* defendem a ideia de que os recursos tecnológicos são como fontes públicas, motivando as pessoas a serem mais ativas na defesa da liberdade na Internet, por exemplo.

(vii) Criatividade: há um desejo dos *hackers* por melhorar constantemente suas criações através da criatividade.

De acordo com Araldi e colegas (2015), a definição de ética *hacker* usada por Pekka Himanen (2001), deixa evidente a busca pela evolução da sociedade através do compartilhamento de habilidades técnicas, o que pode contribuir para um mundo melhor e mais justo para todos.

[6] Consulte o sítio: http://olhardigital.uol.com.br/noticia/qual-a-diferenca-entre-hacker-e-cracker/38024.

Para o Projeto Jovem Hacker, programação é uma habilidade como correr, ler, escrever e várias outras que as pessoas podem aprender. Da mesma forma, *hacker* não é um talento mas uma habilidade muitas vezes associada com procurar e explorar falhas em um computador ou uma rede de computadores, modificar software ou hardware de um computador pessoal e contornar limitações de forma criativa[7]. Nesse contexto, se a ética *hacker* for abordada desde o início da atividade de programação dos novatos, então há uma chance maior de enxergar a tecnologia com um olhar mais voltado à liberdade e à partilha de informação.

5 Desenvolvimento do pensamento computacional com valores da ética *hacker*

Nesta seção, relataremos a oficina de Scratch realizada no Projeto Jovem Hacker. A oficina aconteceu durante o mês de junho de 2015, foram 4 encontros, com duração de 4 horas cada, totalizando 16 horas. Quinze jovens, com idades entre 13 e 16 anos participaram da oficina, que aconteceu no espaço Minha Campinas[8].

A metodologia adotada foi adaptada de Brennan e Resnick (2012), que abordam os seguintes conceitos computacionais: sequência, evento, paralelismo, loop, condicionais, operadores e dados. Além disso, essa metodologia introduz práticas comuns em programação, tais como iteratividade, teste e depuração, reuso e remix, abstração e modularização.

Acrescentamos à metodologia de Brennan e Resnick (2012) os conceitos de passagem de mensagem (sincronização), procedimentos e passagem de parâmetros, pois esses conceitos são importantes para aqueles que irão experimentar outras linguagens além do Scratch, como é o caso dos alunos do Projeto. Além disso, enfatizamos as práticas de remix e compartilhamento, importantes para estimular a ética *hacker* nos jovens.

Todas as oficinas do Projeto Jovem Hacker utilizaram apenas software livre. É importante diferenciar aqui software livre de software gratuito. O software gratuito (*freeware*) é um programa de computador que as pessoas podem utilizar sem pagar. Já o software livre (*free software* ou *open source software*) é qualquer software cuja licença garanta ao seu usuário liberdades relacionadas ao uso, alteração e redistribuição. Exemplos de licenças de software livre são GNU-GPL, Apache, BSD, MIT, dentre outras. Uma lista completa das licenças pode ser consultada em FSF (2016a).

Mais especificamente, software livre se refere à existência simultânea de quatro tipos de liberdade, definidas pela *Free Software Foundation* (FSF, 2016b):

1. A liberdade de executar o programa, para qualquer propósito (liberdade nº 0)
2. A liberdade de estudar como o programa funciona, e adaptá-lo para as suas necessidades (liberdade nº 1). Acesso ao código-fonte é um pré-requisito para esta liberdade.
3. A liberdade de redistribuir cópias de modo que você possa ajudar ao seu próximo (liberdade nº 2).
4. A liberdade de aperfeiçoar o programa, e liberar os seus aperfeiçoamentos, de modo que toda a comunidade se beneficie (liberdade nº 3). Acesso ao código-fonte é um pré-requisito para esta liberdade.

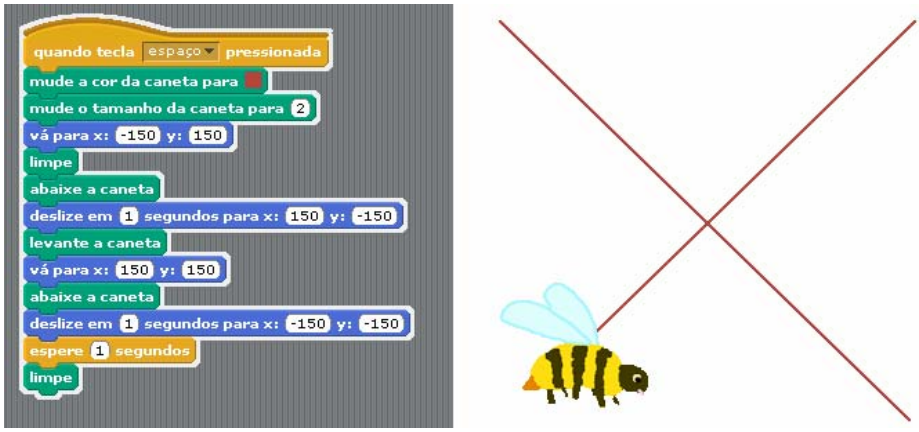
[7] Consulte a página: http://material.jovemhacker.org/apresentacao/01-bem-vindo.html.
[8] Sítio oficial: http://www.minhacampinas.org.br.

Para a oficina, usamos o Scratch versão 1.4, uma versão mais antiga do Scratch, mas que é totalmente livre (independente de softwares proprietários para sua execução). O Scratch 2.0 é mais recente, mas apesar de também ser livre, utiliza o software Flash ou o Adobe Air para executar, ambas tecnologias proprietárias/fechadas, apesar de gratuitas. Nas próximas subseções, apresentaremos os conceitos desenvolvidos na oficina.

5.1 Sequência

A sequência é a organização lógica das instruções necessárias para que um programa seja executado com êxito. A Figura 1 mostra um exemplo de programa que deixa claro que a sequência dos comandos é muito importante para o resultado final. Nesse exemplo a abelha precisa desenhar um "X" na tela. Os comandos de abaixar, levantar a caneta e deslizar devem respeitar uma determinada sequência para alcançar o resultado esperado.

Figura 1 – Exemplo de programa para explorar o conceito de sequência.



Fonte: Elaborada pelos autores.

5.2 Eventos e Passagem de Mensagem

O evento é um acontecimento que produz uma ação. O início de um programa normalmente está associado a um evento – um clique no mouse ou uma tecla pressionada, por exemplo.

Os eventos possuem grande utilidade em relação à sincronização das ações de mais de um objeto. Os objetos em Scratch, referidos como *sprites*, não podem chamar os *scripts* uns dos outros diretamente. Ao invés disso, o Scratch usa um mecanismo de *broadcast* para suportar comunicação e sincronização entre *sprites* (MALONEY et al., 2010). Qualquer *sprite* pode fazer o *broadcast* de uma mensagem (uma *string* qualquer) utilizando os comandos apresentados na Figura 2(b). Sincronização e passagem de mensagem são considerados conceitos avançados de programação, mas no Scratch eles são tratados de uma maneira muito simples, o que facilita o entendimento por parte de iniciantes em programação.

Para introduzir o conceito de eventos com passagem de mensagem, mostramos um exemplo onde havia um diálogo entre um gato e um cachorro. Primeiramente, mostramos a sincronização do diálogo usando tempo e falamos sobre as limitações dessa abordagem. Em

seguida, o mesmo diálogo foi reconstruído usando eventos para sincronizar as falas. Por fim, foi proposto aos estudantes um exercício no qual teriam de criar uma animação com diálogo empregando os eventos para sincronizar os personagens.

5.3 Paralelismo

O paralelismo possibilita que sequências de instruções sejam executadas ao mesmo tempo. Esse conceito, também conhecido como multi-threading, usualmente é considerado uma técnica de programação avançada. Mas a natureza do Scratch torna o conceito fácil de ser absorvido pelos iniciantes em programação. Nosso dia a dia é altamente paralelo, então é natural para os usuários do Scratch terem seus personagens fazendo várias coisas ao mesmo tempo ou ter vários personagens na mesma cena.

Durante a oficina, os conceitos de concorrência e paralelismo foram abordados no diálogo do gato e do cachorro descrito na seção anterior, pois é um exemplo que mostra dois personagens realizando ações simultaneamente. Os exemplos a partir daqui são praticamente todos com ações simultâneas, o que contribui para o conceito ser absorvido pelos alunos.

5.4 Laços

O laço ou *loop* é um comando que tem como função repetir um trecho de sequências quantas vezes forem necessárias. No Scratch há dois tipos de laços, o laço sempre, que repetirá os comandos indefinidamente, e o laço repita, no qual é possível definir o número de vezes da repetição. Dois exercícios sobre este conceito foram apresentados, tendo como objetivo saber se os alunos escolheriam o melhor laço para cada situação e se usariam o laço corretamente. O primeiro exercício era para “tocar uma música e fazer um personagem dançar” o segundo era para “animar pelo menos quatro personagens no fundo do mar”, onde cada personagem teria um laço.

5.5 Condicionais e Operadores

Os condicionais são blocos de comandos que permitem que decisões sejam tomadas tendo em vista condições pré-definidas. No bloco abaixo, SE a <condição> for verdadeira, ENTÃO a <ação1> é executada, SENÃO a <ação2> é executada.

```
SE <condição>
ENTÃO <ação1>
SENÃO <ação2>
```

Os blocos de condicionais devem ser utilizados em conjunto com as condições. Por sua vez, as condições podem ser criadas utilizando operadores relacionais e/ou operadores lógicos.

A instrutora explicou o conceito e, em seguida, fez um exercício junto com os alunos, onde um cachorro perseguia um gato usando as teclas de navegação. O gato, por sua vez, fugia do cachorro também usando teclas de navegação. Em seguida, cinco exercícios envolvendo condicionais foram propostos. A descrição completa dos exercícios pode ser encontrada no sítio

do Projeto Jovem Hacker na parte de Condicionais e Operadores do material de Scratch[9].

5.6 Variáveis

As variáveis são utilizadas para guardar valores nos programas. Por exemplo, ao fazer um jogo em que haverá contagem de pontos, é preciso que exista uma variável para guardar essa pontuação.

Na maioria das linguagens de programação textuais, as variáveis são abstratas, o que leva muitos iniciantes a terem dificuldades para compreender seu conceito. Em Scratch, as variáveis são mais concretas, isto é, os usuários podem vê-las, arrastá-las e manipulá-las como fazem com os demais comandos. Além disso, a tipagem de dados é mais simples, as variáveis podem armazenar três tipos de dados – booleanos, numéricos e *strings*.

O conceito de variáveis foi introduzido por meio de uma dinâmica, que trabalhava também o conceito de condicional. A dinâmica foi feita tendo como exemplo o jogo do Pong com pontos. Na primeira parte os alunos se dividiram em quatro grupos. Os grupos receberam tiras de tecido EVA coloridas, onde estavam escritos os comandos correspondentes ao *script* da bolinha do jogo. A instrutora perguntava qual comando eles achavam que deveria ser utilizado para realizar os movimentos da bolinha. O grupo que tinha o comando consigo se manifestava e juntos – alunos e instrutora – montaram o *script* da bolinha.

Depois dos *scripts* prontos, iniciamos a segunda parte da dinâmica – a simulação da execução do programa. Durante a simulação, a variável foi incrementada com pequenos pedaços de chocolate. Com isso, foi possível mostrar o que acontecia com a bolinha e com a variável a cada iteração do laço de cada *thread* concorrente. Foi uma maneira diferente e descontraída de mostrar conceitos importantes como variáveis, condicionais e concorrência. Mais detalhes sobre a dinâmica podem ser encontrados em Arantes e Ferreira (2015).

Em seguida, a instrutora mostrou o código do Rail Rush (trem e carro) e pediu para usar variáveis para fazer as seguintes alterações: (i) cada vez que o carro atravessar sem bater no trem, o jogador ganha 1 ponto; (ii) o jogo acaba em 30 segundos ou se o trem bater no carro.

O segundo exercício era da bruxa e do fantasma e nele havia três tarefas: (i) adicionar um ponto para cada segundo que a bruxa não tocar no fantasma; (ii) acabar o jogo em trinta segundos e (iii) perder cinco pontos toda vez que a bruxa tocar no fantasma.

5.7 Procedimentos e Passagem de Parâmetros

Os procedimentos e as funções são maneiras de modularizar os programas para que fiquem mais fáceis de entender e modificar. São recursos muito importantes, pois estruturam o programa e evitam repetição de código.

O Scratch 1.4 não trabalha com procedimentos nem com funções. O Scratch 2.0 suporta apenas procedimentos de uma maneira bem limitada:

- Os comandos e procedimentos operam apenas no *sprite* onde aparecem – um *sprite* não pode invocar um comando ou procedimento em outro *sprite*. Isto significa que o mesmo

[9] Material sobre operadores e condicionais em Scratch: http://wiki.jovemhacker.org/index.php/Condicionais_e_Operadores

procedimento não pode ser usado em objetos diferentes daquele onde o procedimento foi declarado. Essa limitação torna os procedimentos pouco úteis no Scratch, pois uma das vantagens da sua utilização é poder reutilizar código.

- Não é possível chamar um procedimento dentro dele mesmo (não possibilita recursão).

Apesar dessas limitações no Scratch, o conceito de procedimento com passagem de parâmetro foi incluído em nossa oficina pelos seguintes motivos:

- A abstração procedural é uma das grandes ideias da ciência da computação, com grande valor para estruturar programas. Como o objetivo do projeto Jovem Hacker é abordar outra linguagem de programação depois do Scratch, achamos melhor mostrar o conceito de procedimentos.
- Ao explorar outros programas em Scratch, os alunos certamente iriam se deparar com procedimentos. Então é importante que conheçam esse conceito.

Entretanto, para praticar e ver exemplos com procedimentos foi necessário usar o Scratch 2.0. Ao justificar o uso do Scratch 2.0 para os alunos, aproveitamos para falar sobre software livre e software proprietário, pois apesar de ser um software livre, o Scratch 2.0 usa o Flash para executar, que é um software proprietário. Explicamos aos alunos que o software livre está de acordo com as quatro liberdades mencionadas no início da seção 5. Por outro lado, o software proprietário não dá acesso ao código-fonte e, apesar de muitos serem gratuitos, não significa que são livres.

Foi importante também diferenciar código-fonte e arquivo executável. O código-fonte é o conjunto de palavras e símbolos contendo as instruções do programa, escritas em uma linguagem de programação como C, Java, Python, dentre outras. Para ilustrar, mostramos o exemplo de uma instrução condicional na linguagem C e a comparamos com uma em Scratch. A instrutora explicou que para modificar o programa é preciso ter o código-fonte e, em linhas gerais, é isso que diferencia o software livre dos softwares proprietários. Depois que o programa foi escrito usando uma linguagem de programação, é comum gerar um arquivo executável do programa para que as pessoas possam executá-lo sem precisar do código-fonte. Explicamos que, quando o programa é proprietário, o usuário tem acesso apenas ao arquivo executável, por isso não é possível alterar o programa.

Os procedimentos no Scratch 2.0 são definidos na aba Mais Blocos. O trecho da Figura 2 mostra a definição de um procedimento chamado velocidade.

Figura 2 – Exemplo de procedimento em Scratch.

Fonte: Elaborada pelos autores.

O procedimento velocidade recebe o parâmetro `number1`. O parâmetro é uma variável local, que pode ser utilizada apenas dentro desse procedimento. Nesse exemplo, o procedimento velocidade irá modificar os valores das variáveis `max` e `min` de acordo com o valor passado para o parâmetro `number1`.

Um exemplo completo que utiliza o procedimento acima pode ser acessado na comunidade do Scratch[10].

5.8 Remix e Compartilhamento

Fazer remix e compartilhamento de código é muito comum e desejado quando se trabalha com software livre. O remix consiste em utilizar partes dos programas de outras pessoas para fazer o seu programa. O compartilhamento acontece quando você mostra o código fonte do seu programa para os outros de maneira livre (usando licenças livres). Assim, outras pessoas poderão utilizar na íntegra ou remixar o seu programa.

Para fazer remix e compartilhamento de programas em Scratch, é preciso entrar na comunidade do Scratch 2.0[11] e fazer um rápido cadastro. É possível examinar os projetos disponíveis na comunidade sem ter um *login* e uma senha, mas para fazer remix e compartilhamento é preciso tê-los.

Na comunidade, é possível remixar, compartilhar, discutir e receber *feedback* de seus projetos. Além disso, explorar projetos de outros autores pode servir como fonte de inspiração para novos projetos e aprendizado de novas técnicas de programação (Resnick et al., 2009).

Faz parte da ética *hacker* criar uma cultura onde o programador sente-se orgulhoso, e não aborrecido, quando seu projeto é adaptado e remixado por outros. De acordo com Resnick e colegas (2009), na comunidade do Scratch 2.0, quando alguém remixa um projeto, o sítio automaticamente adiciona um link de volta ao projeto original como uma maneira do autor ter os créditos. Além disso, cada projeto tem um link para seus “derivados” (projetos que foram remixados a partir dele) e os projetos mais remixados são destacados no sítio.

No último encontro da oficina, todos entraram na comunidade Scratch e exploraram projetos já existentes. Em seguida, a instrutora pediu que os alunos criassem um projeto para ser compartilhado – que podia ser remixado ou não. Dos 14 alunos que estavam presentes nesse encontro, 7 (50%) compartilharam o projeto na comunidade.

Aqueles que não compartilharam, não o fizeram devido aos seguintes motivos:

- um aluno (7,1%) achou que o projeto não ficou bom;
- três (21,4%) gostariam de compartilhar, mas não souberam como fazê-lo;
- um (7,1%) achou que não valia a pena compartilhar porque muitos já haviam compartilhado o mesmo projeto;
- dois (14,2%) não concluíram o projeto no tempo disponível.

Os dois alunos que não terminaram o projeto afirmaram que concluiriam em casa e compartilhariam em seguida.

Um fato que merece destaque é que os alunos escolheram projetos pequenos e remixaram

[10] Exemplo completo com o procedimento velocidade: https://scratch.mit.edu/projects/67670514
[11] Sítio oficial da comunidade do Scratch: http://scratch.mit.edu

pouco. Por exemplo, um aluno escolheu um projeto que gerava fractais indefinidamente. Ele remixou partes do projeto que mudavam os fractais de cor e de tamanho. Outro aluno remixou o projeto do labirinto, um dos mais populares da comunidade do Scratch, alterando os tamanhos de alguns obstáculos.

Boa parte do trabalho intelectual do remix está em ler e entender o código (ou partes dele). Os alunos encontraram programas interessantes que gostariam de remixar, mas ao abrir o código, ficaram assustados com o grande tamanho e muitos desistiram de tentar entender por falta de tempo e experiência em ler código. Acreditamos que isso se deve ao fato de não termos solicitado aos alunos para "ler código". Então, em uma próxima oficina, é bom reservar um tempo para praticar leitura de código antes de falar sobre remix e compartilhamento.

Todos gostaram muito dessa atividade de exploração e deram depoimentos como "*gostei muito da comunidade e de ver como há tantos projetos e animações, gostei muito de criar o remix e mexer na comunidade*", "*bem empolgante*", "*interessante, onde posso compartilhar descobertas e ideias*".

6 Valores da ética *hacker* aplicados à oficina de Scratch

O Projeto Jovem Hacker foi pensado de maneira a valorizar práticas e conceitos que são comuns na ética *hacker*. Durante a oficina descrita na seção anterior, conceitos e práticas da cultura *hacker* foram trabalhados da seguinte maneira:

- Procuramos despertar nos jovens o gosto (a paixão) pela computação e pela programação, procurando desmistificar que programar é algo difícil e acessível a uma minoria. Uma das maneiras de atingir esse objetivo foi trabalhar a programação de uma maneira divertida/lúdica e criar condições para que os jovens se interessarem pelo que estavam aprendendo e realizando. Uma atividade lúdica está relacionada ao entretenimento, que dá prazer e diverte as pessoas envolvidas. Atividades desse tipo, usualmente, estão relacionadas a jogos e ao ato de brincar. Os conteúdos lúdicos são importantes para incutir nas pessoas a noção de que aprender pode ser divertido. O uso do Scratch ajudou muito nesse sentido, pois seu ambiente oferece recursos para criar jogos e outros tipos de animações, o que favorece o aprendizado de conceitos computacionais considerados avançados de uma maneira mais lúdica e atrativa.
- Mostramos que o software livre é útil para a sociedade como um todo, pois empresas e órgãos públicos encontram nas soluções livres uma maneira para racionalizar seus custos e despesas, já que não há a necessidade de pagar um valor para licenças de uso.
- Mostramos que é importante dar reconhecimento e créditos para aquele que criou ou contribuiu com determinada solução. Conforme mencionado na seção 5.8, faz parte da ética *hacker* criar uma cultura onde o programador sinta-se orgulhoso, e não aborrecido, quando seu projeto é adaptado e remixado por outros. Trabalhamos essa questão durante a aula de remix e compartilhamento, onde cerca da metade dos alunos compartilhou suas soluções.
- Mostramos aos jovens que existem as comunidades de software livre, onde várias pessoas se envolvem em atividades para auxiliar uns aos outros na construção de

soluções de software. Essa ideia de comunidades também vai ao encontro da concepção de que o software deve ser um recurso público, que todos podem utilizá-lo e melhorá-lo, para que as soluções fiquem mais robustas e aumentem seu alcance. Em nosso último encontro, os jovens participaram da comunidade do Scratch, explorando e remixando as soluções de outras pessoas. Procuramos mostrar que é construtivo usar sua criatividade para melhorar sua própria solução e para contribuir com a solução de outros. Também citamos as comunidades GNU/Linux, que é o sistema operacional instalado nas máquinas usadas no Projeto Jovem Hacker, mas os jovens não chegaram a entrar/participar nessas comunidades.

7 Avaliação Formativa

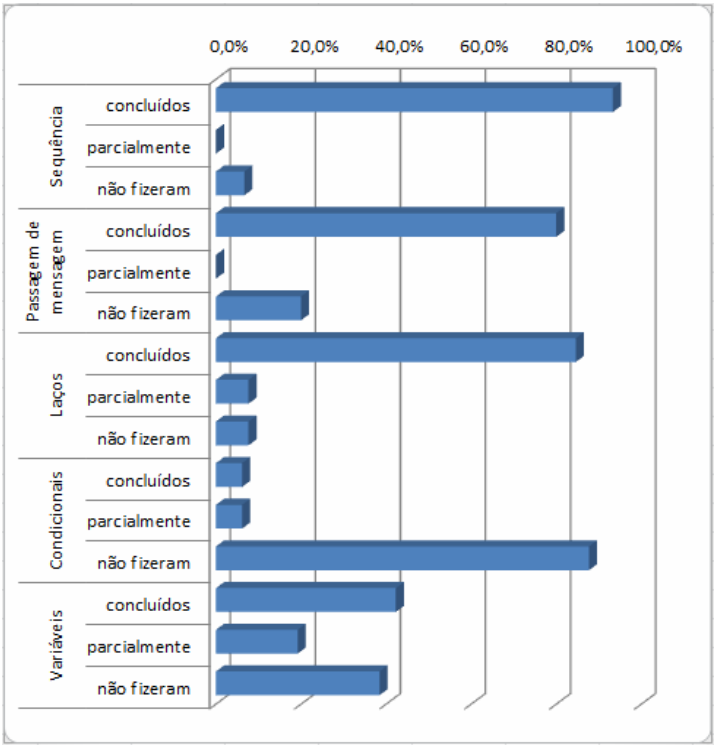
Com relação à avaliação, evitamos instrumentos como provas, pois nosso objetivo era que a oficina fosse divertida e agradável e as provas poderiam desmotivar os alunos devido ao seu sentido de cobrança. A avaliação foi feita com base nos exercícios desenvolvidos durante a oficina (conforme seção 5), o que caracteriza uma natureza formativa. Assim, foi analisada a coleção de trabalhos construídos ao longo da oficina, enfatizando a natureza evolutiva, ao invés da análise de um único projeto final, o que seria um exame somativo. Cada exercício foi avaliado considerando a correta utilização dos conceitos computacionais. Além disso, para cada exercício os alunos diziam o que acharam do nível de dificuldade. Podemos dizer, portanto, que a avaliação consistiu em 2 partes:

(i) Análise dos exercícios feitos pelos alunos: para tanto, no final de cada encontro, os instrutores gravavam em pendrives os programas que os alunos desenvolveram e analisaram seus conteúdos posteriormente. O gráfico da Figura 3 mostra os resultados da análise dos exercícios.

(ii) Impressões dos alunos com relação aos exercícios: no final de cada encontro, solicitamos resposta para a seguinte pergunta com relação ao nível de dificuldade dos exercícios: "*Dê uma nota de 1 a 5 para o exercício, sendo 1-Muito difícil e 5-Muito fácil. Deixe em branco se não fez o exercício*". Com essa pergunta, buscamos saber se os exercícios estão adequados para a oficina e se os alunos estão compreendendo os conceitos, pois assumimos que se estão achando os exercícios fáceis é porque estão assimilando os conceitos. O gráfico da Figura 4 mostra o resultado das impressões dos alunos com relação ao nível de dificuldade dos exercícios.

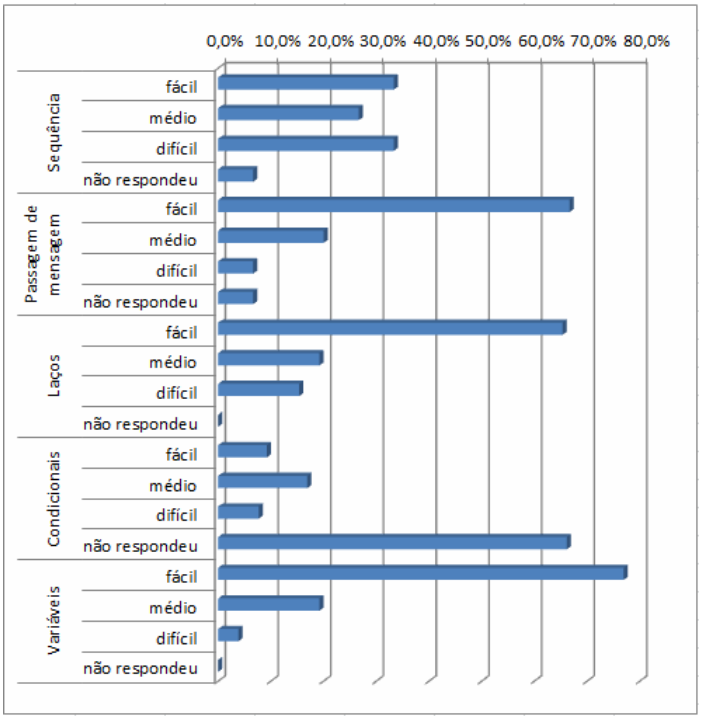
A análise dos exercícios feitos sobre o conceito de sequência (seção 5.1) mostrou que 14 dos 15 presentes (93,3%) concluíram os três exercícios de maneira satisfatória. Cinco consideraram os exercícios fáceis (33,3%), cinco difícil (33,3%), quatro tiveram um pouco de dificuldade (26,7%) e um não respondeu (6,7%). Apesar das dificuldades relatadas por alguns, eles conseguiram realizar as atividades. Acreditamos que a dificuldade inicial deve-se ao fato do primeiro contato com Scratch ter sido algo novo, pois alguns só compreenderam conceitos de palco, mudança de trajes e animação de personagens depois da ajuda dos instrutores. Os alunos requisitaram ajuda dos instrutores para fazer o primeiro exemplo. Os outros dois exemplos, apesar de um pouco mais elaborados, foram feitos com pouca ajuda.

Figura 3 – Gráfico mostrando o resultado da tabulação de dados dos exercícios feitos pelos alunos na oficina de Scratch.



Fonte: Elaborada pelos autores.

Figura 4 – Gráfico mostrando os dados sobre as impressões dos alunos sobre o nível de dificuldade dos exercícios.



Fonte: Elaborada pelos autores.

Com relação aos eventos e passagem de mensagem (seção 5.2), verificamos que doze dos quinze estudantes (80%) finalizaram o exercício e a maioria (66,7%) o classificou como fácil.

Conforme mencionado na seção 5.4, foram dois exercícios sobre laços. Doze dos treze estudantes que estavam presentes (92,3%) concluíram o primeiro exercício com êxito. Com relação ao segundo exercício, dez estudantes concluíram com êxito (76,9%), um concluiu de forma parcial (7,7%) e um não fez a atividade (7,7%) por não ter conseguido realizá-la no tempo disponível. A média dos alunos que concluíram os exercícios com êxito é 84,6%. No nível de dificuldade atribuído aos exercícios sobre laços, a maioria dos alunos os classificou como fáceis (média de 65,4%).

A instrutora solicitou cinco exercícios sobre condicionais com laços (seção 5.5 acima). Diferente dos outros exercícios, esses foram solicitados faltando pouco tempo para encerrar o encontro. Por esse motivo, a instrutora indicou para os alunos terminarem em casa. Os gráficos mostram a média dos resultados. Observe que os resultados do conceito de condicionais está discrepante, quando comparados aos outros exercícios. A maioria dos alunos fez no máximo um dos exercícios, outros fizeram apenas um exercício incompleto. Apenas um aluno fez dois exercícios em casa.

Um fato que chamou a atenção foi que alguns responderam sobre o nível de dificuldade dos exercícios sobre condicionais, mas não os fizeram. Acreditamos que isso se deve ao fato dos alunos terem explorado os exercícios e respondido às questões apenas com a primeira impressão que tiveram, pois o tempo da oficina não foi suficiente para terminarem. Isso ficou claro nas respostas sobre o exercício do fundo do mar, no qual 30,8% classificaram o exercício como fácil, mas ninguém o concluiu com êxito. Uma possível explicação é que os alunos podem ter achado o exercício fácil, mas pouco atrativo. Podem não ter sentido motivação em fazê-lo, já que outros exercícios pareciam mais interessantes.

Na aula sobre variáveis (seção 5.6), alguns exercícios sobre condicionais que os alunos não fizeram em casa foram retomados e explorados. Dois deles foram solicitados como exercícios para serem feitos durante a aula, onde os alunos deviam usar as condicionais e acrescentar variáveis para contar pontos nos jogos. Para resolver o exercício do Rail Rush era preciso uma variável para o tempo e outra para a pontuação. Em relação à pontuação, nove dos treze estudantes (69,2%) fizeram corretamente, e em relação ao tempo e finalização do jogo, sete dos treze estudantes (53,9%) concluíram com êxito. Dez dos treze (77%) estudantes classificaram este exercício como fácil.

Sobre o exercício da bruxa e do fantasma com variáveis, sete dos treze estudantes (53,9%) adicionaram pontos com êxito. Sobre o tempo e finalização do jogo, sete dos treze estudantes (53,9%) também conseguiram concluir. E em relação à perda de pontos, apenas quatro dos treze estudantes (30,8%) conseguiram realizar corretamente. Na pergunta sobre as dificuldades encontradas no exercício, dez estudantes (77%) classificaram como fácil, o que foi discrepante, já que apenas 4 alunos fizeram o exercício completo.

Ao analisar os gráficos das Figuras 3 e 4, percebemos que o Scratch apresenta-se como um ambiente que pode ser utilizado satisfatoriamente pelos jovens para o desenvolvimento de habilidades e conhecimentos relativos ao pensamento computacional.

8 Lições e Desafios

Nesta seção, apresentamos as principais lições que aprendemos com a oficina de Scratch no contexto do Projeto Jovem Hacker, lições que servirão para refinarmos as próximas edições do Projeto. Apresentamos também alguns desafios que precisamos levar em consideração em edições futuras.

Uma das lições que aprendemos foi sobre deixar as aulas mais dinâmicas e participativas. Devido à aparente dificuldade que os alunos tiveram na aula sobre condicionais, o conceito de variáveis foi introduzido por meio de uma dinâmica, que trabalhava também com condicionais. A dinâmica do chocolate (ARANTES e FERREIRA, 2015) proporcionou uma maneira mais participativa e exploratória de apresentar a teoria, o que deixou a aula mais motivadora e descontraída.

Outra lição que aprendemos foi a de que solicitar muitos exercícios pode desmotivar os alunos, o ideal é solicitar um ou dois exercícios de cada vez. Na aula sobre condicionais, a instrutora solicitou muitos exercícios de uma só vez, o que não foi bom, pois os alunos não conseguiram fazer quase nenhum, porque preferiram explorar todos os exercícios ao invés de manter o foco na solução de um deles. Na ocasião, a instrutora aproveitou para incentivá-los a terminar os exercícios em casa, mas a maioria sequer tentou fazer. Para aprender uma linguagem de programação é preciso estudar, praticar, dedicar-se em casa. A oficina contou com 4 encontros com cerca de 4 horas cada – foram apenas 16 horas, pouco tempo para aprender lógica de programação, efetivamente. Então, um dos desafios deste projeto, que precisa ser levado em consideração nas próximas edições, é motivá-los a explorar e estudar fora das aulas presenciais, em casa e outros ambientes.

Na aula sobre laços e condicionais, notamos que alguns alunos se distraíram com joguinhos e celulares. Apesar dos instrutores terem chamado a atenção, essa distração certamente atrapalhou a realização dos exercícios e consequentemente a absorção do conceito de condicionais. Imaginamos que a distração se deve ao fato dos alunos terem percebido que não teriam tempo hábil para fazer os exercícios e/ou terem se sentido desmotivados a fazer os exercícios, por acharem difíceis ou pouco interessantes.

A aula que eles mais se interessaram e mantiveram o foco foi sobre exploração da comunidade do Scratch. Nas próximas edições, será reservado mais tempo para os alunos explorarem a comunidade, remixarem e compartilharem.

Ainda sobre a aula de remix e compartilhamento, antes de abordar propriamente o remix, consideramos que será mais adequado solicitar aos alunos que leiam o código. Boa parte do trabalho do remix está em ler e entender o código (ou partes dele). Muitos alunos encontraram programas interessantes que gostariam de remixar, mas não estavam habituados a ler o código e desistiram de tentar entender os programas que consideraram longos. Nas próximas edições, reservaremos um tempo para praticar leitura de código antes de abordar remix e compartilhamento.

A questão da avaliação e os conceitos de fácil e difícil são muito relativos, o que pode ser apontado como uma limitação desta pesquisa. Na avaliação realizada, procuramos obter

informações dos alunos a respeito do nível de dificuldade dos exercícios. Entretanto, essa questão é muito particular. Outros alunos, de futuras edições do Jovem Hacker, poderão atribuir níveis de dificuldade diferentes, o que provavelmente levará a outros resultados.

9 Considerações Finais

O estudo apresentado neste artigo teve por objetivo demostrar as potencialidades do ambiente de programação Scratch na disseminação do pensamento computacional, permeado por conceitos da ética hacker.

Os resultados apontam que os jovens aprenderam diversos conceitos de Ciência da Computação através dos exercícios desenvolvidos durante a oficina – sequência, evento, passagem de mensagem, paralelismo, laços, condicionais, variáveis, procedimentos, passagem de parâmetros, variáveis, remix e compartilhamento. Mais que conteúdos, os jovens puderam conhecer e exercitar conceitos e práticas relacionados à ética *hacker*, tais como uso de software livre, despertar do gosto pela programação, exploração de uma comunidade, remix e compartilhamento de soluções de programação.

Considerando os resultados obtidos com a oficina, avaliamos que os objetivos inicialmente traçados foram alcançados. Com o uso do Scratch conseguimos explorar os conceitos computacionais apontados por Brennan e Resnick (2012) na disseminação do Pensamento computacional para jovens. Além disso, conseguimos explorar na prática conceitos da ética *hacker*, importantes para a formação de uma geração mais aberta ao compartilhamento de ideias e soluções de software. Por fim, os conteúdos puderam ser avaliados continuamente por meio de exercícios, possibilitando acompanhar a aprendizagem dos estudantes durante seu processo de formação.

A oficina para ensino e avaliação do pensamento computacional com valores da ética *hacker* poderá ser adaptada a outros contextos, bem como ser utilizada nas edições futuras do Projeto Jovem Hacker. O material que foi produzido na oficina, bem como os exercícios utilizados nas aulas e na avaliação, estão disponíveis sob licença pública *Creative Commons* no sítio do Projeto[12].

Agradecimentos

Agradecemos à Secretaria de Cultura do Estado de São Paulo e ao FAEPEX da Unicamp pelo apoio financeiro ao Projeto Jovem Hacker.

Ao Raniere Silva, por ajudar a preparar o material que foi utilizado na oficina. Ao Felipe Rodrigues e ao Gabriel de Souza Fedel, pelo apoio durante as oficinas.

Por fim, agradecemos ao Minha Campinas pelo espaço físico cedido para realização das oficinas.

[12] Endereço com o material produzido no Projeto Jovem Hacker: http://wiki.jovemhacker.org/index.php/Material

Referências

AMIEL, T.; FEDEL, G. S.; ARANTES, F. L.; AGUADO, A. G. Dominando para não ser dominado: Autonomia tecnológica com o projeto jovem hacker. In: WORKSHOP INTERNACIONAL DE SOFTWARE LIVRE, 16., 2015, Porto Alegre. *Anais...*, p.1-13.

ARALDI, M. L. C.; MARTINS, A. R. Q.; TEIXEIRA, A. C. Pesquisa de uso da ética hacker na formação de estudantes do ensino fundamental. In: WORKSHOP INTERNACIONAL DE SOFTWARE LIVRE, 16., 2015, Porto Alegre. *Anais...*, p. 1–13.

ARANTES, F. L.; AMIEL, T.; FEDEL, G. Nos rumos da autonomia tecnológica – desafios e lições aprendidas para a formação de jovens. In: WORKSHOP DE INFORMÁTICA NA ESCOLA, 20., 2014, Dourados, MS. *Anais...*, p.1-10.

ARANTES, F. L.; FERREIRA, J. M. L. S. Uma dinâmica para ensino de conceitos fundamentais de programação. In: WORKSHOP DE ENSINO EM PENSAMENTO COMPUTACIONAL, ALGORITMOS E PROGRAMAÇÃO (WAlgProg), 1., 2015, Maceió. *Anais...*, p. 1-10.

AURELIANO, V. C. O.; TEDESCO, P. C. A. R. Ensino-aprendizagem de programação para iniciantes: uma revisão sistemática da literatura focada no SBIE e WIE. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 23., 2012, Rio de Janeiro. *Anais...* , p. 1–10.

BRENNAN, K.; RESNICK, M. New frameworks for studying and assessing the development of computational thinking. In: ANNUAL MEETING OF THE AMERICAN EDUCATIONAL RESEARCH ASSOCIATION, 2012, Vancouver, Canada. *Proceedings…*, p. 1–25.

CUNY, J.; SNYDER, L.; WING, J. M. *Demystifying computational thinking for non-computer scientists*. Unpublished manuscript in progress, 2010. Disponível em: http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf. Acesso em: 23 jan. 2017.

FRANÇA, R. S.; AMARAL, H. J. C. Proposta metodológica de ensino e avaliação para o desenvolvimento do pensamento computacional com o uso do Scratch. In: WORKSHOP DE INFORMÁTICA NA ESCOLA (WIE), 19., 2013, Campinas. *Anais...*, p. 179–188.

FREE SOFTWARE FOUNDATION (FSF 2016a). *Various Licenses and Comments about Them*. 2016. Disponível em: https://www.gnu.org/licenses/license-list.en.html. Acesso em: 23 de jan. 2017

FREE SOFTWARE FOUNDATION (FSF 2016b). *O que é o software livre?* 2016. Disponível em: https://www.gnu.org/philosophy/free-sw.html. Acesso em: 23 de jan. 2017

HIMANEN, P. *A ética dos Hackers e o espírito da era da informação*: a diferença entre o bom e o mau hacker. Rio de Janeiro: Editora Campus, 2001.

MALONEY, J.; RESNICK, M.; RUSK, N.; SILVERMAN, B.; EASTMOND, E. The Scratch programming language and environment. *Transactions on Computers and Education*, v. 10, n. 4, p.16:1-16:15, Nov. 2010.

PAPERT, S. *Mindstorms*: Children, Computers, and Powerful Ideas. New York: Basic Books, 1980.

RESNICK, M.; MALONEY, J.; MONROY-HERNÁNDEZ, A.; RUSK, N.; EASTMOND, E.; BRENNAN, K.; MILLNER, A.; ROSENBAUM, E.; SILVER, J.; SILVERMAN, B.; KAFAI, Y. Scratch: Programming for all. *Communications of the ACM*, v. 52, n. 11, p.60–67, Nov. 2009.

RODRIGUEZ, C. L.; ZEM-LOPES, A. M.; MARQUES, L.; ISOTANI. S. Pensamento computacional: transformando ideias em jogos digitais usando o Scratch. In: WORKSHOP DE INFORMÁTICA NA ESCOLA (WIE), 21., 2015, Maceió, Brasil. *Anais...*, p. 1–10.

SCAICO, P.; DE LIMA, A.; AZEVEDO, S.; DA SILVA, J. B.; RAPOSO, E. H.; PAIVA, L. F.; ALENCAR, Y.; MENDES, J. P.; SCAICO, A. Ensino de programação no ensino médio: Uma abordagem orientada ao design com a linguagem Scratch. *Revista Brasileira de Informática na Educação*, v. 21, n. 02, p. 92–103, 2013.

Recebido em outubro de 2016
Aprovado para publicação em agosto de 2017

Flávia Linhalis Arantes
Núcleo de Informática Aplicada à Educação (NIED) – Universidade Estadual de Campinas (UNICAMP) – Campinas, SP, Brasil, farantes@unicamp.br

Paula Eduarda Justino Ribeiro
Programa de Formação Interdisciplinar Superior (ProFIS) e Núcleo de Informática Aplicada à Educação (NIED) – Universidade Estadual de Campinas (UNICAMP) – Campinas, SP, Brasil, rpaulaeduarda@gmail.com