Texto Livre: Linguagem e Tecnologia
E-ISSN: 1983-3652
revista@textolivre.org
Universidade Federal de Minas Gerais
Brasil

Linhalis Arantes, Flávia; da Silva Ferreira, José Michael Leandro; Ribeiro, Paula Eduarda Justino
SCRATCH – UM PRIMEIRO OLHAR
Texto Livre: Linguagem e Tecnologia, vol. 8, núm. 2, julio-diciembre, 2015, pp. 137-152
Universidade Federal de Minas Gerais

http://periodicos.letras.ufmg.br/index.php/textolivre
Ano: 2015 – Volume: 8 – Número: 2

SCRATCH – UM PRIMEIRO OLHAR

SCRATCH – A FIRST GLANCE

Flávia Linhalis Arantes
Universidade Estadual de Campinas
farantes@unicamp.br

José Michael Leandro da Silva Ferreira
Universidade Estadual de Campinas
josefh.leandro@hotmail.com

Paula Eduarda Justino Ribeiro
Universidade Estadual de Campinas
rpaulaeduarda@gmail.com

RESUMO: Neste artigo, relatamos a atividade “Scratch, um primeiro olhar”, realizada com o objetivo de registrar as impressões e as respostas dos alunos ao olhar pela primeira vez para programas em Scratch. Apresentamos uma revisão sobre a caracterização da linguagem e do ambiente do Scratch, procurando identificar o que um aluno que nunca teve contato com programação pode usar para extrair informações dos programas. Diferente de outros trabalhos na literatura, destacamos o fato de os comandos serem escritos na língua materna dos alunos como um elemento grandemente facilitador para o entendimento dos mesmos por parte dos iniciantes. Os resultados dos experimentos mostram que a capacidade de leitura dos programas é uma grande aliada no entendimento daqueles que nunca programaram.
PALAVRAS-CHAVE: Scratch; iniciantes em programação; linguagem de programação.

ABSTRACT: In this paper, we report the “Scratch, a first glance” activity, conducted with the aim of registering the impressions and the responses of students when they looked for the first time to Scratch programs. We present a review on the characterization of Scratch language and environment, trying to identify what a student who never had contact with programming can use to extract information from the programs. Different from other works in literature, we highlight the commands written in the mother tongue of the students as a great facilitator to the understanding of the programs by the beginners. The results of the experiments show that the ability to read the programs is of great importance in the understanding from those who never had previous contact with programming.
KEYWORDS: Scratch; beginners in programming; programming language.

1 Introdução

Da década de 80, quando os computadores foram introduzidos nas escolas, houve um entusiasmo inicial para ensinar crianças a programar. Papert apresentou a linguagem Logo como uma nova maneira de pensar a educação e o aprendizado (PAPERT, 1980). O Logo pregava uma modificação estrutural quanto ao papel da tecnologia, colocando o

computador sob o controle das crianças, transformando-o em um aliado na construção e materialização do conhecimento.

Apesar de algumas crianças e professores terem trabalhado com essas novas possibilidades, essa fase 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 – *laptops*, *tablets* e *smarthphones* tornaram-se itens integrantes das bolsas e mochilas das crianças e adolescentes.

Tornou-se comum referir-se aos jovens como “nativos digitais” devido à aparente fluidez que eles têm com a tecnologia. De fato, a maioria deles sente-se confortável com a tecnologia – mandam mensagens, usam jogos *online*, navegam na Internet. Mas isso os torna realmente fluentes? Apesar de eles interagirem com mídias digitais o tempo inteiro, poucos são capazes de criar seus próprios jogos, animações ou simulações. É como se eles 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.

Mas para fazer isso é preciso aprender a programar. A habilidade de programar pode trazer benefícios importantes: ela expande a capacidade de criar e se expressar com o computador. Entretanto, a programação de computadores frequentemente é vista como uma atividade muito técnica e difícil, acessível apenas a algumas pessoas (BARRELLA, 1993; RESNICK *et. al.*, 2009).

Nos últimos anos, novas tentativas de introduzir programação para crianças e adolescentes têm voltado à tona. Algumas iniciativas surgiram nos Estados Unidos, com projetos como `code.org`[1], pois se estima que em poucos anos falte mão de obra qualificada para atuar no mercado de trabalho americano na área de programação. Com o intuito de contribuir com a formação de uma geração que seja mais capacitada tecnologicamente, surgiu também o Projeto Jovem Hacker[2], que investiga maneiras de auxiliar na formação de uma geração que seja autônoma tecnologicamente, tendo como base a ética *hacker* e o *software* livre (ARANTES *et. al.*, 2014; AMIEL *et. al.*, 2015).

O objetivo de tudo isso não é necessariamente preparar os jovens para carreiras de 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 desse 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 Scratch[3]. 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 – todas

1 Disponível em: <https://code.org/>. Acesso em: 14 dez. 2015.

2 Disponível em: <http://jovemhacker.org/>. Acesso em: 14 dez. 2015.

3 Disponível em: <http://scratch.mit.edu>. Acesso em: 14 dez. 2015.

habilidades importantes atualmente. Programar com Scratch, portanto, pode oferecer um contexto e um conjunto de oportunidades para contribuir com o desenvolvimento do pensamento computacional.

Neste artigo, relatamos uma atividade realizada no início do Projeto Jovem Hacker. O intuito da atividade foi investigar o conhecimento dos alunos ao olhar pela primeira vez para programas em Scratch. Apresentamos uma revisão sobre a caracterização da linguagem e do ambiente do Scratch, procurando identificar pistas sobre o que um aluno que nunca teve contato com programação pode usar para extrair informação do programa. Nesse contexto, os artigos de Resnick e colegas, disponíveis na literatura (RESNICK *et. al.,* 2009; MALONEY *et. al.*, 2010; BRENNAN e RESNICK, 2012), apontam características do ambiente e da sintaxe do Scratch como fatores importantes para entendimento dos programas por parte dos iniciantes em programação. Entretanto, os autores quase não falam sobre a importância da língua usada no Scratch – o fato de os comandos poderem ser utilizados na língua materna ou em uma língua familiar ao usuário facilita grandemente o entendimento dos programas por parte dos iniciantes.

A maior contribuição deste artigo é destacar algo que outros trabalhos na literatura tocam superficialmente – a capacidade de ler os programas em Scratch. Os experimentos que realizamos mostram que essa capacidade é uma grande aliada no entendimento dos programas por parte de pessoas que nunca programaram. Em nossa visão, a capacidade de leitura dos programas é mais importante do que a sintaxe e do que o ambiente do Scratch.

Este artigo está organizado da seguinte maneira: na seção 2, revisitamos o ambiente e a sintaxe do Scratch, procurando extrair características usualmente utilizadas pelos iniciantes para ler, criar e compreender os programas. Ainda nessa seção, acrescentamos o fato de o Scratch poder usar uma língua conhecida pelo usuário como fator determinante na hora de os iniciantes compreenderem os programas. Na seção 3, descrevemos a atividade “Scratch – um primeiro olhar”, realizada com alunos que nunca haviam programado, quando procuramos observar o que foi realmente útil para o entendimento dos programas por parte de alunos que nunca tiveram contato com programação. Na seção 4, apresentamos uma discussão e limitações desta pesquisa. Finalmente, a seção 5 conclui este artigo e aponta direcionamentos para trabalhos futuros.

2 Scratch – ambiente, sintaxe e linguagem

Scratch é um ambiente de programação visual que permite aos usuários aprender programação enquanto trabalham em seus projetos de interesse, tais como histórias animadas e jogos. Os projetos em Scratch podem conter personagens, mídias e *scripts*. Som e imagens podem ser criados ou importados usando uma ferramenta para desenho e um gravador de som. A programação é feita juntando-se blocos de comandos coloridos para controlar personagens gráficos – chamados de *sprites*, que se movem em um fundo, chamado palco. Os projetos podem ser salvos no sistema de arquivos local ou no site da comunidade do Scratch (MALONEY *et. al.*, 2010).

O objetivo do Scratch é introduzir programação para quem nunca teve contato com programação. Mas, quais são as características do Scratch que o tornam diferente das demais linguagens de programação para iniciantes? De acordo com Maloney e colegas

(2010), o ambiente e a sintaxe do Scratch foram projetados para facilitar o desenvolvimento do pensamento computacional em iniciantes. Em nosso trabalho, acrescentamos a esses aspectos a questão da língua utilizada nos comandos. No restante desta seção, aspectos do ambiente de programação do Scratch e da sintaxe utilizada são revisados. Aspectos linguísticos são discutidos e apontados como fatores de grande importância para o iniciante em programação.

2.1 Ambiente de programação

O ambiente do Scratch conta com uma tela principal em que o usuário pode arrastar comandos para construir seus programas – veja a Figura 1. À esquerda, é possível observar o conjunto destes comandos: movimento, aparência, som, caneta, controle, sensores, operadores e variáveis. Os comandos são destacados por cores diferentes, justamente para sinalizar que cada conjunto deles é responsável por determinada função. À direita, há um espaço para que o programador visualize os resultados de seu programa. O ambiente também permite adicionar imagens, planos de fundo, músicas e personagens personalizados.

Os itens a seguir são características de projeto do ambiente de programação do Scratch apontadas por Maloney e colegas (2010) como aspectos relevantes para iniciantes em programação:

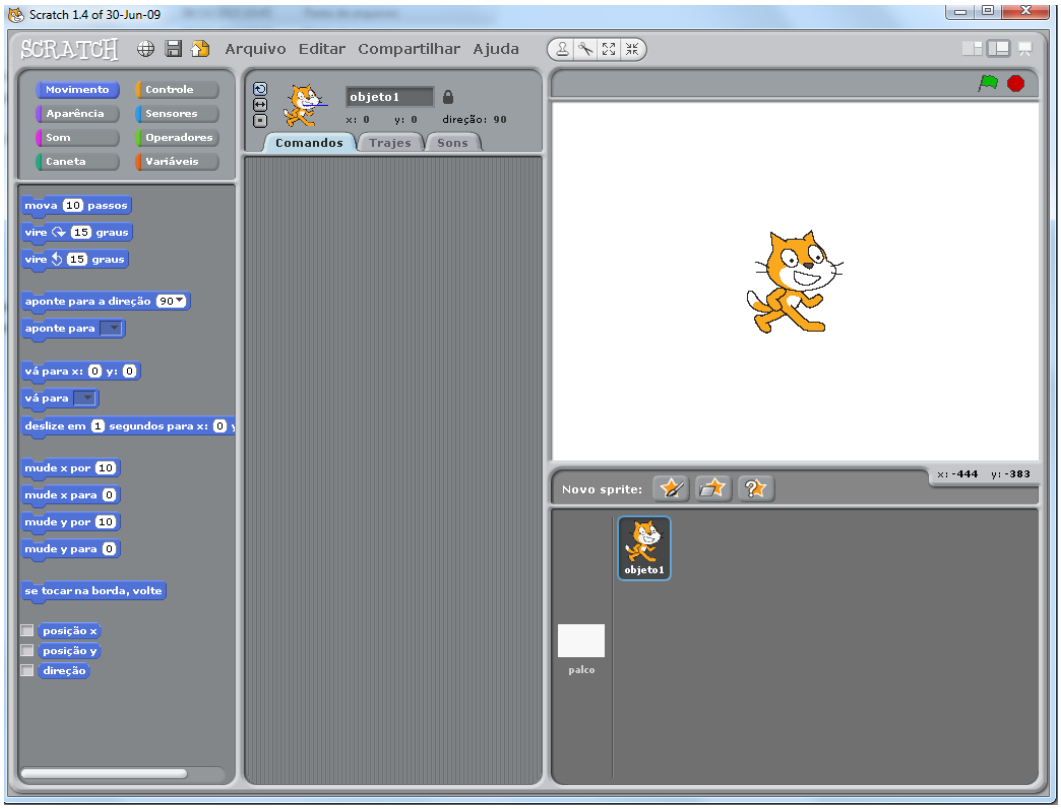

Figura 1: Screenshot do ambiente do Scratch 1.4.
Fonte: própria.

- *Feedback* imediato: para encorajar o auto aprendizado, o ambiente de programação do Scratch foi desenvolvido para fornecer um *feedback* imediato com o resultado da execução do *script* e tornar os dados visíveis. Não há nenhum passo de compilação antes da execução. O usuário pode clicar em um bloco de comandos e ver instantaneamente o seu resultado.
- Janelas simples para navegação: a interface do Scratch torna a navegação mais fácil. Ela usa uma janela simples com painéis para os comandos de maneira a garantir que os componentes principais estejam sempre visíveis.
- Categorização dos comandos: os comandos são divididos em oito categorias, tais como: Movimento, Aparência, Som e Controle. Essa divisão evita listas muito longas de comandos. Em cada categoria, os comandos mais úteis aparecem primeiro. Os blocos de comandos são coloridos por categoria, o que ajuda os usuários a encontrar blocos relacionados.
- Execução visível: quando um *script* está executando, ele fica destacado por uma borda branca brilhante. Esse *feedback* ajuda o usuário a entender quando os *scripts* executam e por quanto tempo eles executam. Se um *script* encontrar um erro (divisão por zero, por exemplo), o bloco que causou o erro fica destacado em vermelho.

2.2 Sintaxe

O Scratch oferece linhas de código prontas, cabendo ao programador apenas desenvolver a estrutura do projeto para sua animação, jogo ou história. Os comandos têm características específicas; cada qual é responsável por executar determinada função. Existem comandos que iniciam a animação, comandos que fazem com que o personagem se movimente, além de comandos que permitem ao programador inserir valores em variáveis.

Os *scripts* ou programas do Scratch são construídos juntando-se comandos, também chamados de blocos. Os formatos dos blocos sugerem como eles se encaixam. Os blocos são arrastados e encaixados uns nos outros, mas esse encaixe não acontece se os blocos não puderem ser usados em determinadas situações. No Scratch, a gramática visual dos formatos dos blocos e suas combinações fazem o papel da sintaxe em uma linguagem de programação puramente textual (MALONEY *et. al.*, 2010).

Há quatro tipos de blocos no Scratch, como mostra a Figura 2. De acordo com Maloney e colegas (2010), o desenho (ou sintaxe) dos blocos foi projetado para possuir certas características que tornam a construção dos programas em Scratch mais intuitiva do ponto de vista dos iniciantes:

- Blocos de comandos (Figura 2(a)): os blocos de comandos têm um “encaixe” na parte superior e um correspondente na parte inferior, como peças de Lego. Esses blocos podem ser unidos para criar uma sequência de comandos e são como as chamadas aos procedimentos em linguagens de programação convencionais.
- Blocos de funções (Figura 2(b)): são como operadores. Esses blocos não são unidos em sequências lineares como os blocos de comandos. Eles são usados como argumentos para comandos e aninhados juntos para construir

expressões. São blocos moldados de acordo com os tipos de valores que retornam: ovais para números e hexágonos para booleanos. Blocos condicionais têm espaços vazios em forma de hexágono, indicando que um valor booleano é necessário.

– Blocos de gatilho (Figura 2(c)): são blocos que conectam eventos, tais como início de programas, cliques do mouse e teclas pressionadas.
– Blocos de estruturas de controle (Figura 2(d)): são tipos de blocos de comandos com uma ou mais sequências embutidas. O formato desses blocos os torna fáceis de usar. Na maioria das linguagens de programação textuais, os delimitadores do bloco podem ser esquecidos, o que leva a erros. Em Scratch, um bloco de controle é uma unidade indivisível, o que é mais intuitivo e evita erros. Esses blocos são em forma de "C" para sugerir que outros blocos devem ser colocados dentro deles.

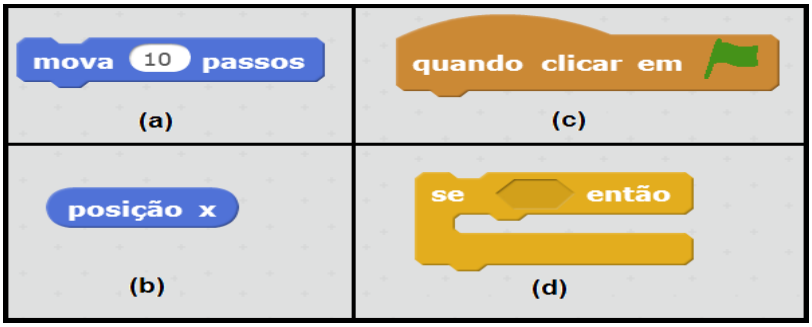

Figura 2: Tipos de blocos em Scratch – (a) comandos, (b) funções, (c) gatilho e (d) estrutura.
Fonte: Figura adaptada de Maloney *et. al.* (2010).

2.3 Aspectos Linguísticos

Geralmente as linguagens de programação possuem uma sintaxe difícil e são implementadas em inglês, obrigando os alunos a dedicarem muito esforço no aprendizado da linguagem de programação enquanto código linguístico artificial. A aprendizagem do código muitas vezes ocorre de maneira descontextualizada, isto é, aprender a linguagem em si não tem relação com a atividade de programação. A programação passa a ser uma atividade de resolução de problemas bastante complexa e a linguagem serve somente para codificar a resolução final a fim de ser passada para o computador (BARRELLA, 1993).

Existe um período de aquisição da linguagem artificial, durante o qual o aluno reflete sobre a sua própria língua natural e sobre essa linguagem artificial, com o objetivo de compreender o funcionamento e especificidades dessa nova linguagem. Essa aprendizagem é possibilitada pelo conhecimento linguístico que o aluno possui sobre a língua natural da qual ele é falante (BARRELLA, 1993). No caso do Scratch, o fato de o aluno ser falante de uma língua natural que é a mesma utilizada para criar os programas implica que essa aquisição é um processo mais direto e simples.

O Scratch possui tradução para diversas línguas do mundo, pois, segundo Resnick e colegas (2009), para promover o intercâmbio e colaboração internacional, houve uma alta prioridade em traduzir o Scratch para vários idiomas.

O fato de o Scratch ser traduzido para 68 idiomas certamente facilita o processo de aprendizagem de programação[4]. Por exemplo, no uso social que o aluno mantém com sua língua natural, possivelmente, ele já atribuiu significados às expressões que no contexto do Scratch são os nomes dos comandos. De acordo com Barrella (1993), esse conhecimento de língua natural ajuda o aluno no contexto de programação porque remete a um universo de interpretação que o ajuda a re-interpretar esses termos na situação de programação. Por exemplo, quando o aluno usa um comando como “mova 10 passos”, ele sabe que vai acontecer alguma coisa relacionada ao movimento do personagem na tela.

A experiência relatada neste artigo está relacionada ao “primeiro olhar” do aluno aos programas em Scratch. Nosso objetivo foi observar o que os alunos, que nunca tiveram contato com programação, podem compreender dos programas em Scratch, apenas olhando para eles – isto é, apenas lendo esses programas em sua língua natural.

3 Scratch – um primeiro olhar

Nesta seção, relatamos uma atividade realizada no início do Projeto Jovem Hacker (ARANTES *et. al.*, 2014; AMIEL *et. al.*, 2015). O intuito da atividade foi investigar o conhecimento e as conclusões dos alunos ao olhar pela primeira vez para programas em Scratch. No contexto do Projeto Jovem Hacker, um dos objetivos dessa atividade foi desmistificar a ideia de que programação é difícil e despertar os alunos sobre a necessidade de ler com atenção os programas – prestar atenção nos detalhes e no significado dos termos é parte importante da rotina de quem lida com programação.

A atividade “Scratch, um primeiro olhar” aconteceu no dia 13 de maio de 2015, no espaço Minha Campinas[5]. Quinze jovens, com idades entre 13 e 16 anos, participaram da atividade. A turma se apresentava com o perfil que esperávamos: jovens usuários de Internet, redes sociais e jogos, com acesso a computadores e dispositivos móveis, mas com praticamente nenhum conhecimento sobre como os *softwares* são desenvolvidos. Um dos participantes disse que seu pai, um programador na linguagem C, mostrava os programas para ele de vez em quando. Outro participante disse ter tido uma oficina de Scratch na escola.

Nas próximas subseções, apresentamos cada etapa da atividade com os resultados dos alunos e algumas observações sobre esses resultados.

3.1 Programa 1 – gato que muda de cor

O programa 1 foi apresentado aos alunos da seguinte maneira: “A seguir, há alguns exemplos de programas em Scratch que afetam ou provocam ações nos personagens ao lado. Escreva o que você acha que acontecerá com os personagens”. O *script* e o personagem do programa 1 estão na Figura 3(a). O gráfico da Figura 3(b) mostra o percentual de respostas dos alunos consideradas corretas, parcialmente corretas e incorretas.

4 Até a data atual, 19/10/2015, a interface web do Scratch 2.0 conta com 68 idiomas no endereço <http://scratch.mit.edu>.

5 Disponível em: <http://www.minhacampinas.org.br/>. Acesso em: 14 dez. 2015.

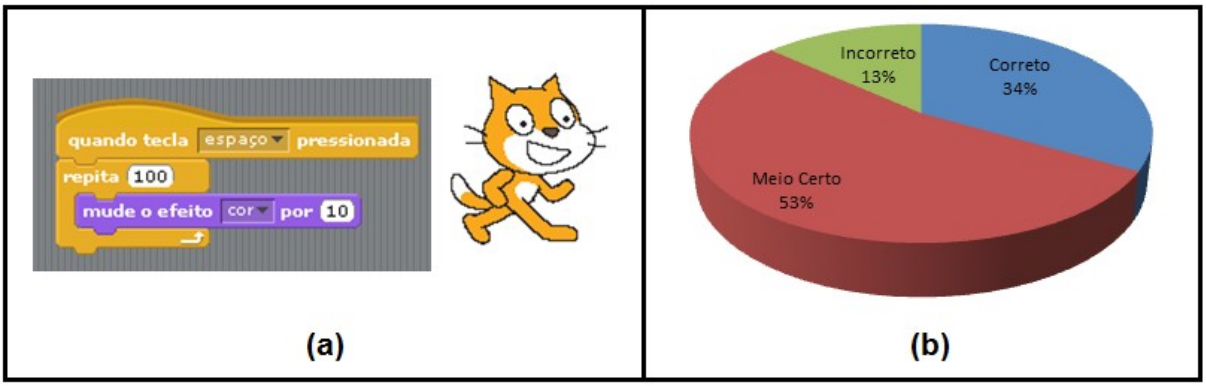

Figura 3: (a) *Script* e personagem do programa 1, (b) percentual de acertos e erros.
Fonte: própria.

Nesse exemplo, ao pressionar a tecla espaço, o gato muda de cor 100 vezes. Cada cor tem um código, que é incrementado em 10 unidades por 100 vezes. A mudança do código de 10 em 10 implica em cores semelhantes, que mudam muito rapidamente. Por isso, é difícil prever o efeito visual desse programa sem ver a execução. Apesar de o gato mudar de cor 100 vezes, isso acontece tão rapidamente que não é possível perceber as 100 cores, pois a visão humana não acompanha as mudanças – só conseguimos ver o gato mudando de cor cerca de 12 vezes. Mas, para chegar a essa conclusão, o aluno precisa ver o programa executando. Ainda assim, mesmo depois de terem visto a execução, os alunos iniciantes perguntaram “Onde estão as 100 cores? Só vi 12!”.

Para esse exemplo, consideramos correto quem respondeu que, ao pressionar a tecla espaço, o gato muda de cor 100 vezes – é exatamente isso que o programa faz e essa leitura é possível de ser feita sem ver a execução, apenas prestando atenção no que está escrito no programa. A resposta “Quando pressionar a tecla espaço, irá repetir 100 vezes a mudança de cor variando entre 10 cores” foi considerada correta, pois o aluno concluiu que haverá uma mudança de cor 100 vezes. Ele concluiu também que a cor irá variar de 10 em 10.

A resposta “Quando a tecla espaço é pressionada, o objeto muda de cor por 10 segundos e isso irá se repetir 100 vezes” foi considerada parcialmente correta. Observe que o aluno percebeu que o gato mudará de cor 100 vezes, mas achou que o tempo (em segundos) tem alguma influência na mudança de cor. Três alunos (20%) deram respostas similares, considerando o tempo. Isso se deve ao fato de a frase “Mude o efeito cor por 10” parecer inacabada do ponto de vista linguístico. Alguns alunos tiveram a suposição de que a frase completa seria “Mude o efeito cor por 10 segundos”.

A resposta “Após a tecla espaço ser pressionada o gato irá ficar na cor azul por 10 segundos” foi considerada incorreta, pois não considera a repetição e não há indícios no programa de que o gato ficará azul.

3.2 Programa 2 – animação da arara

O programa 2 também foi apresentado aos alunos como o anterior. O *script* e o personagem estão na Figura 4(a). Na Figura 4(b), apresentamos um gráfico com o percentual de respostas dos alunos.

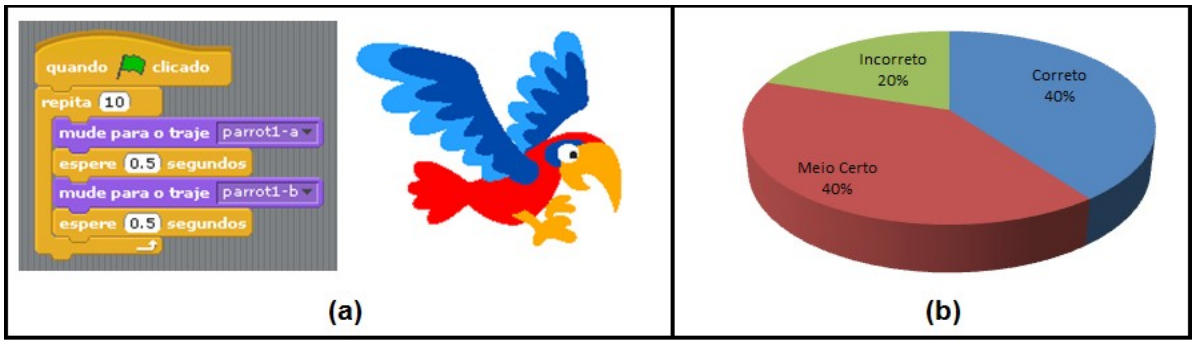

Figura 4: (a) *Script* e personagem do programa 2, (b) percentual de acertos e erros.
Fonte: própria.

Nesse programa, quando a bandeira for clicada, a arara irá bater as asas. Os alunos que perceberam que a arara irá mudar de traje repetidas vezes, tiveram a resposta considerada correta. A resposta “quando a bandeira for clicada, ele mudará de traje em 0,5 segundos por 10 vezes” foi considerada correta. Observe que o aluno transcreveu exatamente o que está escrito no programa. Entretanto, não é possível saber se ele percebeu que a arara irá bater as asas. A resposta “o papagaio irá voltar a voar”, apesar de não ser tão fiel ao que está escrito no programa, também foi considerada correta, pois o aluno conseguiu abstrair que a arara irá bater as asas.

A resposta “clicando na bandeira o pássaro mudará de posição a cada 0,5 segundos umas 10 vezes” foi considerada parcialmente correta. O aluno percebeu a repetição e as pausas da ação do personagem, mas não há indícios nos comandos de que a arara irá se movimentar ou mudar de posição. Apesar de não haver comandos de movimentos nesse programa, uma boa parcela dos alunos (40%) achou que o personagem iria se movimentar, embora, na verdade, a arara apenas bate as asas, sem mudar de posição. Acreditamos que o fato de a imagem da arara estar com as asas abertas, fez os alunos abstraírem que ela irá voar e consequente movimentar-se.

Outra resposta considerada parcialmente correta foi “quando se clica no ícone (bandeira), o objeto muda o trajeto para um lado e depois de 5 segundos muda para o outro, isso se repete 10 vezes”. Nota-se claramente que o aluno associou, erroneamente, a palavra “traje” ao trajeto da arara. Essa palavra indica que haverá uma mudança na imagem do objeto, não em seu trajeto. A resposta do aluno mostra que a palavra “traje” não é muito significativa para o comando e pode levar a outras interpretações.

A resposta “o papagaio irá trocar de cor (azul e vermelho) para outra cor, e logo depois irá voltar para as mesmas cores” foi considerada incorreta, pois não há indícios de que a arara mudará de cor.

3.3 Objetivo do repita

Depois de mostrar os dois primeiros programas, perguntamos aos alunos para que eles acham que serve o comando “repita”. Observe que os dois programas anteriores usam esse comando. Acreditamos que todos compreenderam que o comando serve para repetir algo. A maioria dos alunos acertou (66,67%), com respostas como “para repetir as ações que estão dentro do repita” ou “repetir a ação”. As respostas consideradas parcialmente corretas (33,33%) associaram a repetição a outros comandos, tais como “para repetir a sequência de números (por segundo)” ou “para repetir a tecla que pede”.

Observe que os alunos perceberam que o comando fará uma repetição, mas associaram a repetição a ações específicas. Não houve respostas consideradas incorretas.

É interessante observar que as respostas “para repetir as ações que estão dentro do repita” e “o repita serve para repetir as ações encaixadas nele um número de vezes determinado” indicam claramente a influência da sintaxe do Scratch na resposta dos alunos. Conforme mencionado na seção 2.2, os blocos de controle, como é o caso do repita, são em forma de “C” para sugerir que outros blocos devem ser colocados dentro deles.

3.4 Programa 3 – gato que desenha um quadrado

O programa 3 foi apresentado aos alunos da mesma maneira dos dois anteriores, isto é, o aluno deveria observar o programa e dizer o que aconteceria com o personagem. A Figura 5(a) mostra o *script* e o personagem, nesse caso um gato visto de cima. A Figura 5(b) mostra o resultado da execução do *script* e a Figura 5(c) mostra um gráfico com o percentual dos resultados dos alunos.

Nesse programa, quando a bandeira for clicada, o gato desenha um quadrado vermelho na tela. Os alunos que perceberam que o quadrado será desenhado tiveram a resposta considerada correta (apenas 20%). Na maior parte das respostas os alunos descreveram exatamente o que está escrito no programa, por exemplo, “quando a bandeira verde for clicada o cursor ficará vermelho e quando o cursor for abaixado o gato andará 100 passos e virará num ângulo de 90 graus e repetirá 3 vezes e na última esperará 3 segundos e depois o gato será deletado”. Essa e outras repostas similares foram consideradas parcialmente corretas (53%) – são respostas que descrevem o programa, mas não consideram o todo. Isto é, a maioria dos alunos leu o programa linha por linha, sem procurar entender o significado ou o resultado da execução.

A resposta “ele irá seguir o cursor do mouse” foi considerada incorreta, pois não há indícios no programa que sugerem que o gato seguirá o cursor do mouse.

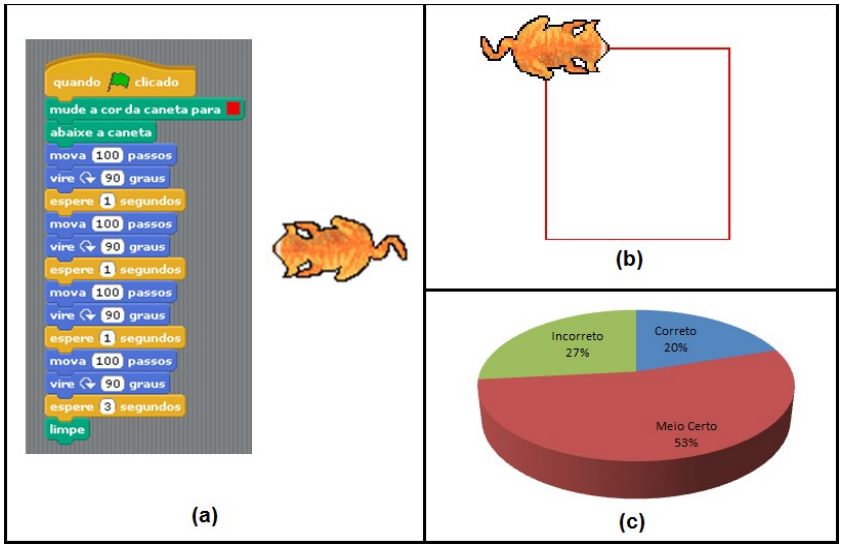

Figura 5: (a) *Script* com personagem, (b) resultado da execução, (c) gráfico mostrando o percentual de acertos e erros dos alunos.
Fonte: própria.

3.5 Programa do gato usando repita

Nessa atividade, foi pedido que os alunos reescrevessem o programa do gato que desenha o quadrado (seção anterior) utilizando o laço repita. No exercício anterior, havia um mesmo trecho do programa que se repetia claramente três vezes, indicando uma repetição. Além disso, nos dois primeiros exercícios, eles viram exemplos do repita. Veja os resultados no gráfico da Figura 6.

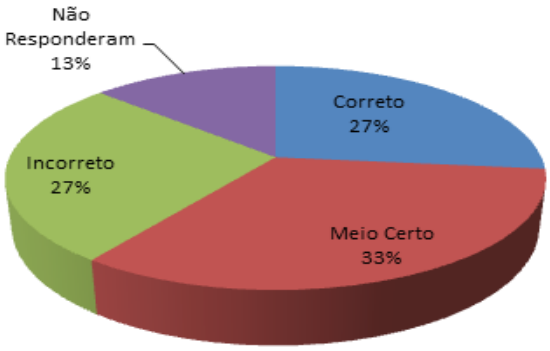

Figura 6: Resultado dos alunos sobre reescrever o programa da seção anterior utilizando o laço repita.
Fonte: própria.

Quatro alunos (27%) conseguiram refazer o programa conforme especificado, o que consideramos um número satisfatório, pois os alunos não conhecem programação e sequer viram a execução dos exemplos anteriores. Cinco alunos (33%) fizeram o programa incompleto ou usaram respostas coerentes, mas não fizeram o algoritmo. Quatro alunos (27%) tiveram respostas consideradas incorretas, tais como “o repita seria usado em animações, mostrando o gato fazendo o preenchimento da tela” ou “para equilíbrio dos animais”. Dois alunos (13%) não responderam.

3.6 Programa 4 – polvo e caranguejo

O programa do polvo e do caranguejo foi apresentado para os alunos da mesma maneira que os anteriores. Nesse exemplo específico, foi falado que o *script* refere-se apenas ao polvo e que o caranguejo fica parado. A Figura 7(a) mostra o *script* e os personagens e a Figura 7(b) mostra um gráfico com o percentual de respostas dos alunos.

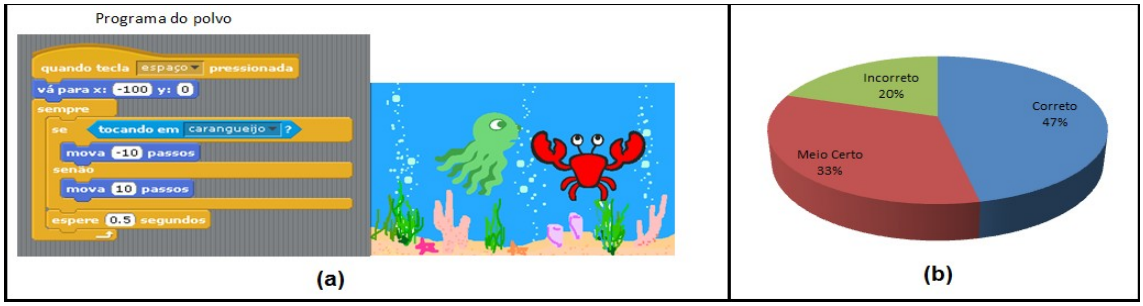

Figura 7: (a) *Script* e personagens, (b) percentual de respostas dos alunos.
Fonte: própria.

Nesse exemplo, o polvo se move em direção ao caranguejo e se afasta se tocar nele. O objetivo aqui foi perceber se os alunos compreendem os comandos se e senão. Sete alunos (47%) responderam corretamente, com explicações do tipo "quando a tecla espaço for pressionada o polvo irá para localização x=-100 y=0, se tocar no caranguejo volta 10 passos e se não tocar anda 10 passos".

Cinco alunos (33,4%) acharam que se tratava de um jogo e que os números -10 e 10 seriam pontos ganhos pelo polvo. Supomos que os alunos acharam isso porque há uma condição em que um personagem toca no outro, além de um número positivo e outro negativo, o que remete a incremento e decremento de pontos. Os alunos nessa situação tiveram suas respostas consideradas parcialmente corretas. Por exemplo, a resposta "clicando no espaço o jogo iniciará e o polvo se moverá e se você clicar no caranguejo perderá 10 pontos acertando o polvo ganhará 10 pontos" foi considerada parcialmente correta.

A resposta "se o polvo chegar perto do caranguejo o jogo irá acabar" foi considerada incorreta porque não há indícios que o programa irá acabar caso o polvo chegue perto do caranguejo.

3.7 Objetivo do se...senão

Depois de mostrar o exemplo do polvo e do caranguejo, perguntamos aos alunos para que eles achavam que servia o comando "se...senão". Veja o resultado no gráfico da Figura 8.

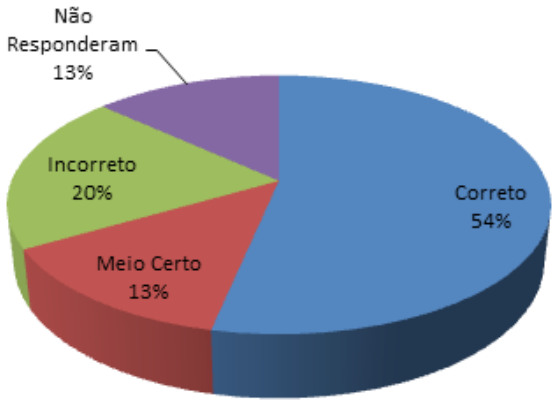

Figura 8: Respostas dos alunos sobre o objetivo do se...senão.
Fonte: própria.

A maioria dos alunos acertou (53,4%), com respostas do tipo "se e senão seria como poder e não poder" ou "se uma ação for feita acontece algo / senão aconteceu outra ação". A resposta "para se o polvo fazer tal coisa, acontecerá os comandos do se e do senão" foi considerada parcialmente correta, pois o aluno não escreveu explicitamente que os comandos se e senão desencadeiam ações diferentes. A resposta "acho que é para modificar, ou alterar" foi considerada incorreta porque indica que o aluno não compreendeu os comandos.

4 Discussão

A Tabela 1 mostra um resumo do percentual de respostas consideradas corretas, parcialmente corretas e incorretas, bem como a média de respostas. A Figura 9 mostra o gráfico com a média das respostas dos alunos.

Tabela 1: Resumo e média dos resultados dos alunos.

Atividade	Corretas	Parcialmente Corretas	Incorretas	Não responderam
Gato que muda de cor	34%	53%	13%	0%
Animação da arara	40%	40%	20%	0%
Objetivo do repita	67%	33,33%	0%	0%
Gato que desenha quadrado	20%	53%	27%	0%
Gato usando repita	27%	33%	27%	13%
Polvo e caranguejo	47%	33%	20%	0%
Objetivo do se...senão	54%	13%	20%	13%
Média	**41,24%**	**36,90%**	**18,14%**	**3,71%**

Fonte: própria.

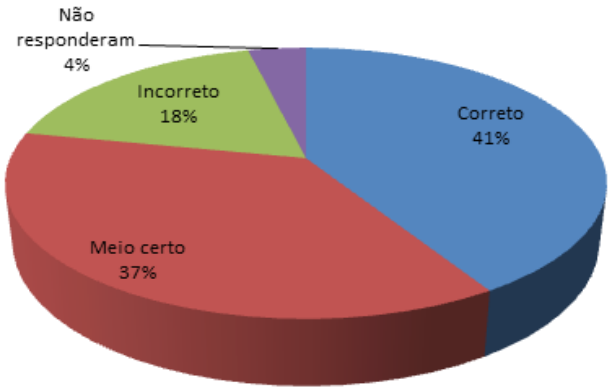

Figura 9: Média dos resultados.
Fonte: própria.

Consideramos os resultados apresentados na Tabela 1 e no gráfico da Figura 9 bastante satisfatórios – 41,24% das respostas foram consideradas corretas e 36,9% foram consideradas parcialmente corretas.

As atividades relatadas na seção 3 mostram que os alunos chegaram a esses resultados praticamente pela leitura que fizeram dos programas. Com isso, podemos concluir que o fato de os programas em Scratch poderem ser escritos na língua materna dos iniciantes em programação é de grande importância para seu entendimento. Conforme relatado na seção 2, o ambiente de programação e a sintaxe dos programas também são fatores relevantes para os iniciantes. Mas, no experimento realizado, os alunos não tiveram contato com o ambiente nem com a montagem dos programas, que já estavam prontos. Então, podemos dizer que as respostas dos alunos foram feitas

praticamente com a leitura dos programas.

Apesar dos bons resultados, é importante considerarmos e discutirmos algumas questões linguísticas envolvidas no aprendizado de uma linguagem de programação. Sabemos que o Scratch não é uma linguagem de programação tradicional. Pelo fato de o Scratch conservar, intencionalmente, algumas das características da nossa língua natural, como o nome dos comandos e usar a forma escrita como meio de comunicação com a máquina, pode-se pensar que o sujeito que aprende a programar com Scratch não precisa aprendê-lo do ponto de vista linguístico, o que não é verdade (BARRELLA, 1993).

As atividades abordadas demonstram que os cuidados tomados na implementação do Scratch, apesar de facilitarem a interação entre o sujeito e o computador, não são uma garantia de que o sujeito não precisa "pensar" sobre o significado dos comandos que mediam essa interação. Um exemplo disso foi a atividade do gato que desenha um quadrado (seção 3.4). Boa parte dos alunos descreveram as ações do programa corretamente, mas não perceberam que o gato desenharia um quadrado, pois não pensaram no significado do conjunto dos comandos.

De acordo com Barrella (1993), os processos de re-significação dos nomes dos comandos e de construção de regras de uso dos mesmos no contexto de programação indicam que existe um período inicial de aprendizagem que é dedicado à aquisição do Scratch como sistema linguístico artificial. Essa aprendizagem é possibilitada pelo conhecimento linguístico que o sujeito possui sobre a língua natural, da qual ele é falante. Entretanto, o fato de o sujeito ser falante de uma língua natural não implica que essa aquisição seja um processo direto. O sujeito hipotetiza sobre o funcionamento da linguagem de programação e esse processo é marcado por hesitações, comprovações, depurações, ajustes, uma série de atividades linguísticas permitidas pela sua língua natural.

Quando o aluno usa um comando como "vire 90 graus", ele sabe que o personagem irá virar. Entretanto, a manipulação dos comandos em situações de programação é que pode levar o sujeito a construir as regras de uso dos mesmos. Em muitos casos, o sujeito precisa ver como isso acontece no computador para reconstruir o significado do nome do comando. Um exemplo claro disso foi uma resposta de um aluno descrita na seção 3.2, em que ele associou a palavra "traje" ao trajeto da arara e não à mudança de imagem da arara. A interpretação do aluno mostrou que a palavra não é significativa para o comando. Mas, se esse aluno pudesse ter visto a execução do programa, sua resposta provavelmente seria diferente, pois ele poderia ressignificar o termo "traje" no contexto do comando.

Outro exemplo da importância da depuração e da ressignificação seguida de comprovação, foi dada na atividade do gato que muda de cor (seção 3.1), em que 20% dos alunos deram respostas considerando o tempo apesar de não haver indícios de contagem de tempo no programa. As respostas se devem ao fato de a frase "mude o efeito cor por 10" parecer inacabada. Os alunos imaginaram que a frase completa seria "mude o efeito cor por 10 segundos". Se os alunos tivessem visto a execução do programa, poderiam ressignificar o comando e concluir que ele não estava relacionado ao tempo.

É provável que certos fenômenos linguísticos tenham passado desapercebidos porque não conseguimos observar todos os detalhes envolvidos nesse período inicial de aquisição do código da linguagem de programação. Além disso, podemos apontar as

seguintes limitações desta pesquisa:

- A questão da avaliação é muito relativa. No experimento realizado, procuramos estabelecer alguns critérios para definir o que consideramos como respostas corretas, parcialmente corretas e incorretas. Outras pessoas poderiam considerar critérios diferentes, o que provavelmente levaria a outros resultados.
- Não é possível saber exatamente o que os alunos estavam pensando quando escreveram suas respostas. Pode ter acontecido de o aluno pensar uma coisa e escrever outra ou não saber expressar o que pensou, visto que o pensamento computacional é algo ainda imaturo naqueles que nunca tiveram contato com programação.

5 Conclusões e trabalhos futuros

O Scratch é constituído de uma coleção de blocos de programação, que os usuários encaixam para criar programas. De acordo com Resnick e colegas (2009), tal como acontece com peças de Lego, a sintaxe do Scratch é composta por esses blocos que sugerem como eles devem ser colocados juntos. No ambiente do Scratch, os iniciantes em programação podem começar por simplesmente mexer com os “tijolos”, colocando-os juntos em diferentes sequências e combinações para ver o que acontece.

Além das facilidades apontadas por Resnick e colegas (2009) com relação à sintaxe e ao ambiente de programação do Scratch, neste artigo destacamos a questão da língua como sendo de fundamental importância para iniciantes em programação. Para observar e tecer conclusões a respeito dessa importância, realizamos a atividade “Scratch – um primeiro olhar”, descrita na seção 3. O gráfico da Figura 9 mostrou resultados expressivos – 41,24% de respostas consideradas corretas e 36,9% parcialmente corretas. Esse resultado ratifica o que outros trabalhos na literatura já constataram sobre o Scratch ser uma ferramenta que contribui com o desenvolvimento do pensamento computacional em iniciantes. Além disso, os resultados evidenciam que os comandos do Scratch escritos em uma língua familiar ao iniciante facilita grandemente o entendimento dos programas, o que quase não é destacado em outros trabalhos na literatura.

Neste artigo, analisamos *scripts* em Scratch escritos em português. Como trabalho futuro, é interessante ter experiências com códigos escritos em outra língua (como inglês), com o objetivo de realizar comparações com os resultados já obtidos. Outro trabalho futuro interessante é avaliar a capacidade de leitura em um segundo olhar, depois que os alunos tiverem uma interação com o ambiente do Scratch.

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: *16º Workshop Internacional de Software Livre (WSL)*, evento realizado em conjunto com o Fórum Internacional de Software Livre (FISL 16), p. 1-13, Porto Alegre, BRASIL, 2015.

ARANTES, F. L.; AMIEL, T.; FEDEL, G. S. Nos rumos da autonomia tecnológica –

desafios e lições aprendidas para a formação de jovens. In: *XX Workshop de Informática na Escola - WIE 2014*. III Congresso Brasileiro de Informática na Educação (CBIE), Vol. 1, p. 308-317, Dourados, BRASIL, 2014.

BARRELLA, F. M. F. O Trabalho Linguístico do Sujeito ao Adquirir a Linguagem Logo, In: VALENTE, J. A. (Org.). *Computadores e Conhecimento – repensando a educação*. Campinas: gráfica Central da UNICAMP, 1993. p. 257–273.

BRENNAN, K.; RESNICK, M. New frameworks for studying and assessing the development of computational thinking. In: *Annual Meeting of the American Educational Research Association (AERA)*, 2012, Vancouver, Canada. *Proceedings of AERA* 2012. p. 1-25.

CUNY, J.; SNYDER, L.; WING, J. M. *Demystifying computational thinking for noncomputer scientists*. Online: http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf. 2010.

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

PAPERT, S. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, Inc., New York, NY, USA, 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, 2009.

Agradecimentos

À Secretaria de Cultura do Estado de São Paulo e ao FAEPEX da UNICAMP pelo apoio financeiro ao Projeto Jovem Hacker.

Recebido em 20 de outubro de 2015.
Aprovado em 19 de novembro de 2015.