Proposta Metodológica de Ensino e Avaliação para o Desenvolvimento do Pensamento Computacional com o Uso do Scratch

Rozelma Soares de França[1], Haroldo José Costa do Amaral[2]

[1]Centro de Informática – Universidade Federal de Pernambuco (UFPE)
CEP 50.740-560 – Recife, PE – Brasil

[2]Universidade de Pernambuco, Campus Garanhuns (UPE)
CEP 55.294-902 – Garanhuns, PE – Brasil

{rozelma.soares, haroldo.amaral}@gmail.com

Abstract. *Teaching basic concepts of Computer Science in basic education is fundamental to the development of skills necessary for problem solving, can support and relate to other sciences. In order to stimulate the development of Computational Thinking, this paper presents the design and evaluation of a workshop held Scratch with students from a public school in state of Pernambuco. Although the sample size is not statistically significant, the results indicate the effectiveness of the workshop in relation to the learning content Computing. The method described can be replicated in other contexts where Scratch can be used.*

Resumo. *O ensino de conceitos básicos da Ciência da Computação na Educação Básica é fundamental para o desenvolvimento de habilidades, necessárias à resolução de problemas, podendo apoiar e relacionar-se com outras ciências. No intuito de estimular o desenvolvimento do Pensamento Computacional, este trabalho apresenta o design e avaliação de uma oficina de Scratch realizada com estudantes de uma escola pública no Estado de Pernambuco. Apesar do tamanho da amostra não ser estatisticamente significativo, os resultados apontam a eficácia da oficina em relação à aprendizagem de conteúdos de Computação. O método descrito poderá ser replicado a outros contextos em que o Scratch for utilizado.*

1. Introdução

A computação permeia todas as atividades humanas, das artes às tecnologias, e hoje não se pode imaginar uma sociedade sem computador (NUNES, 2008). Assim, não se pode conceber o cidadão ignorante em Computação, do ponto de vista da Ciência da Computação. A introdução de conceitos de Computação na Educação Básica é fundamental pelo seu caráter transversal às demais áreas do conhecimento (CSTA, 2011). Percebendo a importância em desenvolver desde cedo habilidades como o Pensamento Computacional, os Estados Unidos e países da Europa têm implantado um currículo mínimo de Computação em suas escolas (CSTA, 2005). No Brasil, o debate ainda é incipiente, destacando-se iniciativas como o projeto Reinventando o Ensino Médio que, dentre os seus objetivos, busca inserir conceitos de Tecnologia da Informação e Pensamento Computacional nos currículos das escolas estaduais de Minas Gerais (CARVALHO *et al.*, 2013).

Como enfatiza Paulo Blinkstein, a lista de habilidades e conhecimentos necessários para o pleno exercício da cidadania no século XXI é extensa, incluindo o Pensamento Computacional, sendo este, talvez, o mais importante e menos compreendido (BLIKSTEIN,

DOI: 10.5753/CBIE.WIE.2013.179

2008). O Pensamento Computacional baseia-se em fundamentos da Computação, envolvendo a resolução de problemas, a capacidade de projetar sistemas e a compreensão do comportamento humano (WING, 2006). É um pensamento analítico e compartilha com a Matemática a resolução de problemas, com a Engenharia a concepção e avaliação de um sistema grande e complexo que opera dentro dos limites do mundo real e com a Ciência a compreensão sobre computabilidade, inteligência, a mente e o comportamento humano (WING, 2008). Ele constitui uma habilidade fundamental para todos e não apenas para os cientistas da computação, influenciando a pesquisa em várias áreas do conhecimento (BUNDY, 2007). Evidências da influência do Pensamento Computacional em outros campos incluem a Biologia com o algorimo que acelera o sequenciamento do genoma humano; as Ciências e Engenharia com a simulação de modelos matemáticos de processos físicos encontrados na natureza; as pesquisas aeroespaciais com a simulação de missões espaciais, dentre outros (WING, 2008).

Visando facilitar a aprendizagem de crianças e jovens, ferramentas têm sido propostas, por pesquisas, para mediar o processo de ensino de Computação e a promoção do Pensamento Computacional. Neste cenário, podemos destacar o Scratch (MALONEY *et al*., 2004), um ambiente de programação visual que, segundo Brennan (2011), possibilita a exploração de diversos conceitos, práticas e perpectivas computacionais de maneira criativa com o uso de uma abordagem de aprendizagem baseada no conceito de *design*. Na literatura, há diversos relatos de uso de Scratch com estudantes do ensino fundamental e/ou médio e iniciantes em programação (MALONEY *et al*., 2008; AURELIANO *et al.*, 2012). No entanto, além de ensinar, faz-se necessário avaliar os efeitos do uso desse ambiente na aprendizagem dos alunos, assumindo a avaliação da aprendizagem como uma prática contínua que busque melhorar as aprendizagens em curso, contribuindo para o acompanhamento e orientação dos aprendizes durante o seu processo de formação.

Desse modo, neste trabalho, são abordadas duas questões: (1) *Que conceitos e práticas computacionais podem ser explorados com o uso do Scratch na disseminação do Pensamento Computacional na Educação Básica?*; (2) *Como conceitos computacionais podem ser avaliados continuamente com o uso do Scratch?*. Estas duas questões são tratadas a partir da apresentação do *design* de uma oficina realizada com estudantes da Educação Básica e dos resultados da avaliação contínua da aprendizagem obtidos com as atividades desenvolvidas. Apesar do tamanho da amostra não ser estatisticamente significativo, os resultados apontam a eficácia da oficina em relação à aprendizagem de conteúdos fundamentais da Computação. A abordagem avaliativa, aqui descrita, poderá também ser adaptada a outros contextos em que o Scratch seja o ambiente utilizado para realização de atividades que visem à disseminação do Pensamento Computacional.

O restante do artigo está organizado como segue: a Seção 2 apresenta uma visão geral do ambiente de programação Scratch, bem como suas possibilidades de aprendizagem; a Seção 3 relata uma experiência com estudantes da Educação Básica, uma oficina realizada na aplicação de conceitos e práticas computacionais, para a disseminação do Pensamento Computacional; a Seção 4 discute a aprendizagem e sua avaliação, decorrentes das atividades desenvolvidas, bem como ilustra alguns resultados; por fim, a Seção 5 apresenta considerações finais acerca de todo trabalho.

2. Scratch

Scratch é um ambiente de programação visual desenvolvido pelo Lifelong Kindergarten Group (LLK), grupo de pesquisa do MIT Media Lab. Ele baseia-se nas ideias construcionistas do Logo (PAPERT, 1980) e do Etoys (KAY 2010; STEINMETZ, 2002 *apud* MALONEY *et al*., 2010) e permite a criação de artefatos como histórias interativas,

jogos e animações (MALONEY *et al*., 2008). Para auxiliar os usuários a construírem seus artefatos de maneira envolvente, motivadora e pessoalmente significativa, Scratch facilita a importação e criação de vários tipos de mídia (imagens, sons, música). Ainda, há um site (http://scratch.mit.edu/) que permite aos usuários compartilharem seus projetos, receber *feedback* e encorajamento de seus pares e aprender com os projetos já publicados por outras pessoas (RESNICK *et al.*, 2009).

A meta principal do Scratch é introduzir a programação para quem não tem experiência no assunto. A programação é feita arrastando-se blocos de comandos que devem ser encaixados uns aos outros. Os comandos assemelham-se a peças de quebra-cabeça e quando combinados formam programas sintaticamente corretos, tendo o usuário que focar apenas na lógica de funcionamento do seu projeto (MALAN *et al.*, 2007). Para exemplificar uma das diferenças existentes entre Scratch e outras linguagens de programação, a Figura 1 apresenta a sintaxe de um simples programa feito no ambiente e em Java.

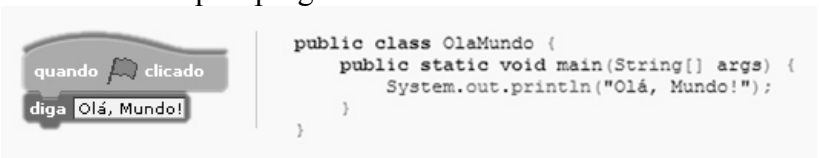

Figura 1. Sintaxe de um programa feito no Scratch e em Java

Scratch possibilita a aprendizagem baseada no conceito de *design*, abordagem que, segundo Brennan (2011), enfatiza a *concepção* (criar e não apenas utilizar ou interagir), a *personalização* (criando algo que é pessoalmente significativo e relevante), a *colaboração* (trabalhando com outras pessoas nas criações) e a *reflexão* (revendo e repensando as práticas criativas de cada um). De acordo com a Partnership for the 21st Century (2003), o ambiente apoia o desenvolvimento de habilidades de aprendizagem do século XXI. O relatório identifica nove tipos de habilidades de aprendizagem, divididos em três áreas chaves, onde Natalie Rusk, Mitchel Resnick e John Maloney destacam as formas como Scratch suporta o desenvolvimento de tais habilidades:

- *Habilidades de informação e comunicação*: ao trabalhar em projetos Scratch, os alunos aprendem a selecionar, criar e gerenciar múltiplas formas de mídia, incluindo texto, imagens, animações e gravações de áudio. Como os estudantes ganham experiência com a criação de meios de comunicação, tornam-se mais perspicazes e críticos ao analisar os meios de comunicação que veem ao redor deles. Além disso, considerando que uma comunicação eficaz no mundo de hoje exige mais do que a capacidade de saber ler e escrever um texto, Scratch envolve os alunos na escolha, manipulação e integração de uma variedade de meios, permitindo que expressem-se de maneira criativa e persuasiva.
- *Habilidades de pensar e resolver problemas*: à medida que aprendem a programar, os estudantes aprimoram seu raciocínio crítico e pensamento sistemático, no desenvolvimento de suas soluções, em contextos significativos. Criar um projeto Scratch requer pensar em uma ideia, em seguida, descobrir a forma de dividir o problema em etapas e implementá-las usando a programação em blocos. O ambiente ainda é projetado para que os alunos possam alterar dinamicamente partes do código e ver imediatamente os resultados; dessa forma, durante todo o processo de desenvolvimento de um projeto, os estudantes se engajam na experimentação e resolução de problemas de maneira iterativa. Também, Scratch encoraja o pensamento criativo ao envolver os aprendizes na busca de soluções inovadoras para problemas, não apenas aprender a resolver um problema predefinido, mas estar preparado para chegar a novas soluções para os desafios que surgirem.

- *Habilidades interpessoais e autodirecionáveis:* como os programas Scratch são construídos de blocos gráficos, o código de programação é mais legível que em outras linguagens de programação. Os objetos visuais e código modular suportam a colaboração, habilitando os alunos a trabalharem em conjunto. Salienta-se também que ter uma ideia e descobrir como programá-la em Scratch requer persistência e prática. No entanto, quando os estudantes trabalham em projetos que sejam pessoalmente significativos, as suas ideias fornecem motivação interna para a superação de desafios e frustrações encontradas no processo de concepção e resolução de problemas. Além disso, quando os alunos criam projetos Scratch, eles têm um público alvo em mente e precisam pensar sobre como outras pessoas irão reagir e responder a eles. Considerando que projetos Scratch são fáceis de mudar e rever, os estudantes podem modificá-los com base no *feedback* dos outros. Por fim, como os programas são compartilháveis, estudantes podem utilizar Scratch para provocar importantes discussões com outros membros da comunidade.

3. Ensinando Computação e promovendo o Pensamento Computacional com o uso do Scratch

A oficina *Aprendendo Conceitos de Ciência da Computação com Scratch* foi projetada com dois objetivos específicos: (1) engajar estudantes da Educação Básica em atividades pedagógicas que lhes permitissem aprender sobre fundamentos da Ciência da Computação e desenvolver habilidades relacionadas ao Pensamento Computacional e (2) avaliar a aprendizagem dos participantes em relação aos conteúdos desenvolvidos durante o curso. O estudo foi realizado no segundo semestre de 2012 e contou com a participação de 24 estudantes com idade variando entre 13 e 14 anos, pertencentes a uma escola pública do estado de Pernambuco, que aceitaram voluntariamente participar da pesquisa, mediante uma breve explicação sobre a mesma. A oficina durou 3h e ocorreu no laboratório de informática da instituição na qual os participantes estavam vinculados.

O *design* da oficina foi fortemente influenciado por três fatores: (1) possibilidade do uso de conceitos computacionais no desenvolvimento de projetos de diferentes gêneros (histórias, jogos, animações, entre outros); (2) arranjo de estudantes trabalhando em pares por computador, motivando, assim, a colaboração no ato de aprender e na proposição de soluções para os problemas levantados; e (3) avaliação, na medida em que os estudantes deveriam realizar exercícios (não avaliados) que lhe permitissem compreender o conteúdo apresentado e projetos (avaliados), onde eles deveriam aplicar o conhecimento que já possuíam com o que haviam acabado de adquirir, na projeção de soluções que culminassem em projetos de diversos gêneros. A partir dessa definição, diversos conceitos e práticas computacionais (BRENNAN, 2011; BRENNAN & RESNICK, 2012) puderam ser explorados no decorrer da oficina, conforme descrito nas Subseções 3.1 e 3.2 deste artigo.

3.1. Conceitos Computacionais

À medida que os participantes projetavam soluções interativas com Scratch, eles se envolviam com um conjunto de conceitos computacionais, comuns a outras linguagens de programação. A oficina permitiu a exploração de vários conceitos: *sequência*, *evento, paralelismo*, *loop, condicionais*, *operadores* e *dados*. Para cada conceito, foi apresentada a sua definição, exemplos de uso e, quando possível, a aplicação desses conceitos em outros contextos que não a Computação. A seguir, são descritos cada conceito computacional estudado, atrelando-se um exemplo a cada definição para facilitar o entendimento:

- *Sequência:* consiste numa série de etapas individuais ou instruções que podem ser executadas pelo computador, especificando a ação que deve ser produzida. Por exemplo,

com a sequência de instruções exposta na Figura 2 o objeto Gato foi programado para mover-se 10 passos e após 2 segundos declarar a frase "Estou programando!".

Figura 2. Exemplo de sequência de instruções

- *Evento:* faz com que um acontecimento produza uma ação. A Figura 3 ilustra situações em que: (1) quando a bandeira verde é pressionada, o objeto diz "Olá"; (2) quando a tecla espaço é pressionada, o objeto gira 90º no sentido horário, espera 2 segundos e gira 90º no sentido anti-horário.

Figura 3. Exemplos de eventos produzindo ações

- *Paralelismo:* possibilita que sequências de instruções sejam executadas ao mesmo tempo. A Figura 4, por exemplo, mostra um programa em que três atividades são executadas paralelamente quando a bandeira verde é pressionada: (1) uma trilha sonora é continuamente reproduzida, (2) o objeto Gato move-se continuamente para frente e para trás e (3) apresenta-se para os telespectadores citando seu nome e gosto pela dança.

Figura 4. Exemplo de paralelismo com um objeto

- *Loop:* mecanismo que permite executar a mesma sequência várias vezes. A Figura 5 ilustra como um *loop* pode ser utilizado para expressar uma sequência de instruções de maneira mais sucinta, ao invés do uso dos mesmos comandos repetidas vezes. No exemplo, em vez de tocar o som, mover e esperar com 10 blocos de comandos consecutivos, usou-se apenas 5: tocar o som, mover-se 10 passos, esperar 2 segundos, mover-se 10 passos e esperar 2 segundos, estando estes blocos cercados por um comando de repetição com o número desejado de iterações.
- *Condicionais:* permitem que decisões sejam tomadas tendo em vista condições pré-definidas. A Figura 6 ilustra o uso de uma estrutura condicional a qual determina que *se* o ponteiro do mouse tocar o objeto, este terá seu traje alterado para "Abelha", caso contrário ficará com o traje "Letra A".

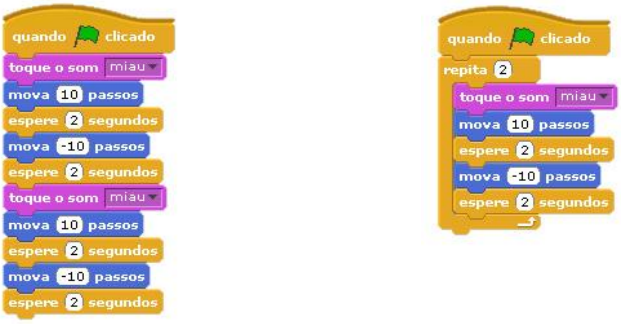

Figura 5. Exemplo de sequência de instruções repetidas expressa em um *loop*

Figura 6. Exemplo de condicional

- *Operadores*: os operadores fornecem suporte a expressões matemática, lógica e *string*. No Scratch, há suporte para operações matemáticas como adição, subtração, multiplicação e divisão, bem como para manipulação de *strings* (concatenação, por exemplo). Na Figura 7, é possível visualizar alguns dos operadores disponíveis no ambiente.

Figura 7. Exemplos de operadores em Scrach

- *Dados:* envolve armazenar, recuperar e atualizar valores. Scratch oferece dois *containers* para dados: variáveis (podem manter um número ou *string*) e listas (podem manter uma coleção de números ou *strings*). No exemplo ilustrado na Figura 8, uma história é criada a partir dos dados fornecidos pelo usuário. Perguntas são feitas ao utilizador e suas respostas são armazenadas em diferentes variáveis. Ao final, a história é composta a partir dos dados fornecidos.

Figura 8. Exemplo de uso de variáveis para armazenar dados fornecidos pelo usuário

3.2. Práticas Computacionais

Práticas computacionais concentram-se no processo de pensar e aprender, focando não apenas naquilo que se está aprendendo, mas em como se está aprendendo (BRENNAN & RESNICK, 2012). Dessa forma, na oficina, buscou-se explorar uma variedade de estratégias

que poderiam ser utilizadas pelos participantes no desenvolvimento de projetos Scratch: *iterativo e incremental; abstração e modularização*; *teste e depuração; reutilização e reformulação.*

Desenvolver um projeto não é um processo sequencial, em que primeiramente identificam-se os conceitos, depois se desenvolve um plano e codifica-se. O processo é adaptativo, no qual o plano pode mudar e a solução ao problema pode ser codificada em pequenas tarefas que interagem na execução de uma tarefa maior. Assim, o desenvolvimento *iterativo e incremental* é uma atividade necessária, ao passo que possibilita desenvolver uma parte, depois verificar se ela funciona e, em seguida, desenvolver um pouco mais. Durante a oficina, os estudantes foram motivados a fazer um esboço das atividades ou tarefas necessárias para desenvolver seus projetos; escrever uma lista preliminar dos recursos necessários para construí-los; rever os elementos do planejamento e alterar, caso necessário; considerar alternativas para a solução e codificá-la continuamente.

Considerando que as habilidades desenvolvidas na oficina poderiam culminar no desenvolvimento de grandes soluções, *abstração e modularização* foram abordadas, já que são práticas importantes no entendimento e resolução de grandes problemas. Tais práticas possibilitam que uma solução seja quebrada em partes, seguindo o princípio "Dividir para Conquistar", facilitando assim o seu desenvolvimento.

Além de codificar uma tarefa, é necessário desenvolver estratégias para lidar com erros. Assim, faz-se necessário o *teste e depuração* para certificar que tudo está funcionando como planejado, como também encontrar e corrigir erros. Dessa forma, a cada parte codificada, os participantes eram encorajados a testá-las e, caso estivesse funcionando corretamente, uma nova parte da solução poderia ser codificada. Caso contrário, os estudantes eram incentivados a realizar sucessivas ações que possibilitassem a identificação e correção de erros. Em todos os casos, os participantes eram sujeitos ativos no processo de aprendizagem, conduzindo ações mediante o ciclo descrição-execução-reflexão-depuração (VALENTE, 1993).

Os projetos desenvolvidos permitiram aos estudantes seguirem os seus interesses e explorarem de forma autônoma as capacidades desenvolvidas. Além de criarem os seus, os participantes foram estimulados a *reutilizar e reformular* projetos produzidos por usuários de Scratch, disponibilizados na comunidade on-line. O desenvolvimento dessa habilidade é importante à medida que pode potencializar ideias mais complexas das que os aprendizes seriam capazes de ter por conta própria. Além disso, *reutilizar e reformular* apóiam o desenvolvimento da capacidade de leitura e interpretação de códigos e podem provocar discussões sobre propriedade e autoria.

4. Avaliação da aprendizagem de Computação e do desenvolvimento do Pensamento Computacional

Diversos desafios na elaboração de uma metodologia de avaliação para a oficina foram enfrentados. Um deles diz respeito ao curto período de duração do curso, composto por vários conceitos e práticas computacionais. Ainda, a perspectiva dos participantes para que a oficina fosse divertida e agradável, evitando instrumentos avaliativos como provas, pois poderia desmotivar a participação dos estudantes. Em particular, não se verificou o conhecimento prévio dos alunos sobre os conteúdos que seriam ministrados, usando pré-testes. No entanto, uma avaliação diagnóstica já está sendo desenvolvida para ser aplicada nas próximas edições da oficina. No início do curso, os estudantes foram questionados sobre sua experiência prévia em Computação e sobre o uso do Scratch. Por meio das discussões informais, percebeu-se que nenhum dos participantes tinha experiência com o ambiente de

programação que seria utilizado, nem possuía conhecimento formal sobre os conteúdos que seriam tratados na oficina.

Para a avaliação, os principais artefatos foram os projetos desenvolvidos, o que caracteriza a sua natureza formativa. Assim, foi analisada a coleção de trabalhos construídos ao longo do tempo, enfatizando a natureza evolutiva e do desenvolvimento de um portfólio dos pares de aprendizes, ao invés, por exemplo, da análise de um único projeto desenvolvido ao final do curso, com características de um exame somativo. Cada projeto foi avaliado tendo em vista a correta utilização de conceitos computacionais na sua implementação. Assim, o uso inadequado de determinado conceito não foi contabilizado. Considera-se que mais importante que utilizar é entender e saber aplicar um conceito específico na construção de uma solução. Além disso, cada trabalho foi classificado em um gênero (animação, história, jogo e música), permitindo analisar a capacidade dos estudantes no desenvolvimento de diferentes gêneros de projetos.

A Figura 9 mostra os resultados da avaliação da aprendizagem em relação aos conceitos computacionais estudados. Quatro projetos foram desenvolvidos por cada uma das doze duplas. Até o desenvolvimento do Projeto 1, foram ministrados os conceitos *sequência*, *evento* e *paralelismo*. Como ilustra o gráfico da Figura 9, todas as duplas demonstraram domínio sobre *sequência* e *evento* aplicando o aprendizado na construção de todos os projetos interativos realizados. Em relação ao conceito *paralelismo*, no Projeto 1, apenas três duplas conseguiram utilizá-lo corretamente. Ainda, uma dupla estendeu o conhecimento sobre esse conteúdo para a construção do Projeto 4, ocorrido ao final do curso.

Após exposição do conceito *loop*, os estudantes estavam aptos a construírem o Projeto 2 e, como pode ser observado na Figura 9, sete duplas compreenderam tal conceito e conseguiram aplicá-lo corretamente em seus trabalhos. Também, observa-se que *loop* foi utilizado corretamente na construção dos Projetos 3 e 4, por duas e cinco duplas, respectivamente. Logo após, o conceito *condicionais* foi exposto, o qual pôde ser avaliado a partir do Projeto 3. Percebe-se que mais da metade da turma entendeu e soube usá-lo satisfatoriamente e que três duplas também utilizaram *condicionais* na implementação do Projeto 4. Finalmente, *operadores* e *dados* foram abordados e deram subsídio à construção do último projeto. Percebe-se na Figura 9 um resultado de aprendizagem satisfatório em relação a esses dois conceitos, uma vez que mais de 60% da turma utilizaram-os corretamente na proposição de soluções interativas.

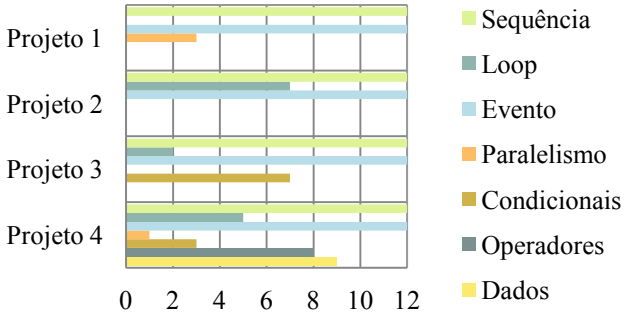

Figura 9. Avaliação formativa de conceitos computacionais

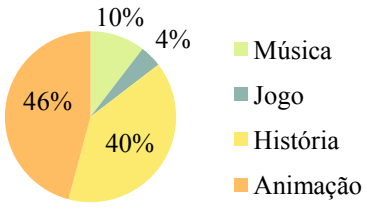

Figura 10. Gênero dos projetos

Em relação aos gênenos dos projetos Scratch, a Figura 10 expõe os dados dos 48 projetos implementados pelos participantes da oficina. Nesse sentido, 46% das propostas foram classificadas na categoria Animação e 40% em História, o que demostra a capacidade dos estudantes em construir projetos de diferentes gêneros utilizando diversos conceitos

computacionais. Em menor proporção, projetos do tipo Jogo foram implentados, como também Música, os quais carregam fortes características de diversão e de preferência musical de seus criadores.

Ao analisar tais resultados, percebe-se que Scratch apresenta-se como um ambiente que pode ser utilizado satisfatoriamente por estudantes da Educação Básica no desenvolvimento de habilidades e conhecimentos tão necessários na atualidade, como o Pensamento Computacional. Ainda, através do acompanhamento da construção de conhecimentos pelos estudantes, é possível perceber que o conceito *paralelismo* não é trivial de ser aprendido num curso com curto período de duração, considerando que os aprendizes não possuem conhecimento prévio sobre o assunto. Dado que, naturalmente, o mundo funciona em paralelo, a compreensão sobre o conceito *paralelismo* torna-se útil para modelar o mundo real e, dessa forma, sugere-se que em cursos que abordem esse tema com estudantes da Educação Básica, um tempo maior seja dedicado a tais atividades. Por outro lado, os demais conceitos tratados na oficina obtiveram bons resultados, os quais podem ter sido alcançados a partir do *design* proposto da oficina, o qual possibilitou que os aprendizes trabalhassem em pares e criassem projetos que fossem pessoalmente significativos, permitindo, assim, que suas ideias gerassem motivação interna para superar os desafios e frustações decorrentes do processo de resolução de problemas.

5. Considerações Finais

O estudo apresentado neste trabalho teve por objetivo demonstrar as potencialidades do ambiente de programação visual Scratch, na disseminação do Pensamento Computacional na Educação Básica, bem como avaliar a intervenção de seu uso na aprendizagem de conceitos computacionais. Para tal, uma oficina foi ministrada a estudantes e contou com a exposição de diversos conceitos e práticas computacionais. Apesar das limitações, em função do tempo de realização e por ser uma atividade extracurricular, os resultados apontam que os estudantes aprenderam diversos conceitos de Ciência da Computação. Através dos projetos desenvolvidos, eles puderam demonstrar competência em *sequência, evento, paralelismo, loop, condicionais, operadores* e *dados,* sendo *paralelismo* o conceito menos compreendido e *sequência* e *evento* os que apresentaram melhores resultados de aprendizagem. Mais que conteúdos, eles puderam conhecer e exercitar práticas computacionais na realização dos desafios que lhes eram apresentados e expor suas soluções em projetos de variados gêneros, de acordo com suas preferências e habilidades.

Tendo em vista os resultados obtidos com o estudo realizado, considera-se que os objetivos inicialmente traçados foram alcançados. Com o uso do Scratch, conseguiu-se explorar os conceitos computacionais apontados por Brennan & Resnick (2012), na disseminação do Pensamento Computacional na Educação Básica. Além disso, tais conteúdos puderam ser avaliados continuamente, possibilitando, assim, acompanhar a aprendizagem dos estudantes durante o seu processo de formação, não se limitando à verificação da aprendizagem apenas ao final curso. O modelo para ensino e avaliação da aprendizagem em projetos Scratch apresentado poderá ser adaptado a outros contextos para obterem-se *insights* sobre as competências dos estudantes nos assuntos tratados. Os resultados apresentados não são estatisticamente significativos e, por isso, em trabalhos futuros pretende-se aplicar o curso e utilizar a técnica de avaliação descrita numa amostra maior. Também, almeja-se criar um instrumento que possa mensurar o conhecimento prévio dos participantes sobre os conteúdos que serão abordados para que seja possível comparar os resultados alcançados e identificar os efeitos reais do curso na aprendizagem de conceitos da Ciência da Computação. Além disso, planeja-se investigar as práticas computacionais utilizadas pelos aprendizes no planejamento e construção de suas soluções interativas.

Referências

Aureliano, V. C. O. ; Tedesco, P. C. A. R. . (2012) "Avaliando o uso do Scratch como abordagem alternativa para o processo de ensino-aprendizagem de programação". In: XX Workshop sobre Educação em Computação, 2012, Curitiba. XXXII CSBC.

Blikstein, P. (2008). O pensamento computacional e a reinvenção do computador na educação. Disponível em < http://www.blikstein.com/paulo/documents/online/ol_pensamento_computacional.html> Acesso em: 05 de set. de 2011.

Brennan, K. (2011). "Creative computing: A design-based introduction to computational thinking". ScratchEd.

Brennan, K.; Resnick, M. (2012) "New frameworks for studying and assessing the development of computational thinking". In: AERA 2012.

Bundy, A. (2007). "Computational thinking is pervasive". J. Scient. Pract. Comput. 1, 67–69.

Carvalho, M. L. B; Chaimowicz, L.; Moro, M. M (2013). "Pensamento Computacional no Ensino Médio Mineiro". In: XXI Workshop sobre Educação em Computação, 2013, Maceió. Anais do XXXIII CSBC.

CSTA - Computer Science Teacher Association. (2005). "The New Educational Imperative: Improving High School Computer Science Education". Final Report of the CSTA. Curriculum Improvement Task Force. ACM - Association for Computing Machinery.

CSTA - Computer Science Teacher Association. (2011). "CSTA K-12 Computer Science Standards". CSTA Standards Task Force. ACM - Association for Computing Machinery.

Malan, D. J.; Leitner, H. H. (2007) "Scratch for budding computer scientists". Proceedings do 38th SIGCSE'07, Kentucky, USA, p. 223–227.

Maloney, J.; Peppler, K.; Kafai, Y. B.; Resnick, M.; Rusk, N. (2008). "Programming by Choice: Urban Youth Learning Programming with Scratch". In: SIGCSE'08, March, pp. 367-371.

Maloney, J.; Burd, L.; Kafai, Y.; Rusk, N.; Silverman, B.; Resnick, M. (2004). "Scratch: A Sneak Preview". Second International Conference on Creating, Connecting, and Collaborating through Computing. Kyoto, Japan, pp. 104-109.

Maloney, J.; Resnick, M.; Rusk, N.; Silverman, B.; Eastmond, E. (2010). "The scratch programming language and environment". ACM Transactions on Computing Education, vol. 10, n. 4, article 16, 15p.

Nunes, D. J. (2008). "Licenciatura em Computação". Jornal da Ciência, 30 de Maio.

Papert, S. (1980). "Mindstorms: Children, Computers, and Powerful Ideas". BasicBooks, New York.

Partnership for 21st Century Skills (2003). Learning for the 21st Century. Disponível em <http://www.21stcenturyskills.org/>. Acesso em: 05 de ago. de 2013.

Resnick, M.; Maloney, J.; Monroy-Hern ´Andez, A.; Rusk, N.; Eastmond, E.; Brennan, K.; Millner, A.; Rosenbaum, E.; Silver, J.; Silverman, B.; Kafai, Y. 2009. "Scratch: Programming for all". Comm. ACM 52, 11, 60–67.

Rusk, N.; Resnick, M.; Maloney, J. "Learning with Scratch: 21st Century Learning Skills". Lifelong Kindergarten Group. MIT Media Laboratory.

Valente, J. A. (1993). "Diferentes usos do Computador na Educação". In: Valente, J. A. (org.), Computadores e Conhecimento: Repensando a Educação. Campinas, SP, Gráfica Central da Unicamp.

Wing, J. M. (2006). "Computational thinking". Commun. ACM, 49(3):33–35.

Wing, J. M. (2008). "Computational thinking and thinking about computing". Phil. Trans. R. Soc. A, 366(1881):3717–3725.