http://periodicos.letras.ufmg.br/index.php/texolivre
Ano: 2014 – Volume: 7 – Número: 2

PROGRAMAR É BOM PARA AS CRIANÇAS? UMA VISÃO CRÍTICA SOBRE O ENSINO DE PROGRAMAÇÃO NAS ESCOLAS

PROGRAMMING IS GOOD FOR CHILDREN? A CRITICAL VIEW ABOUT TEACHING PROGRAMMING IN SCHOOLS

Wendell Bento Geraldes/Instituto Federal de Goiás

RESUMO: O artigo apresenta reflexões sobre o ensino de programação nas escolas e os impactos positivo e negativo dessa prática nos dias atuais. O texto trata ainda das iniciativas relativas ao ensino de programação nas escolas, considerando também a opinião de alguns especialistas sobre o assunto. Afinal, é bom para as crianças aprender a programar computadores nas escolas? Todas as pessoas podem aprender a programar computadores? Qual a importância da aprendizagem de programação para a sociedade atual? Em busca das respostas a essas questões, serão discutidas aqui as vantagens e desvantagens a respeito deste tema.
PALAVRAS-CHAVE: educação; escolas; *software*; computador; inovação.

ABSTRACT: This article presents reflections on teaching programming in schools and the positive and negative impact of this new methodology today. The study also discusses the initiatives relating to teaching programming in schools, considering also the opinion of experts on the subject. The following questions are addressed: Is it good for children to learn to program computers in schools? Can all people learn to program computers? What is the importance of learning for today's society? The pros and cons regarding teaching programming in schools will be discussed in search of answers to these questions.
KEYWORDS: education; schools; *software*; computer; innovation.

1 Introdução

O computador é uma máquina composta por circuitos eletrônicos, cabos, fontes de alimentação; sob esse ponto de vista, ele parece não ter muita utilidade. Na verdade, o computador só consegue armazenar dados, efetuar cálculos e realizar outras diversas funções por meio de programas que realizam o processamento de dados. Sendo assim, pode-se considerar que o computador possui duas partes: o hardware, que é a parte física, e o *software*, que são os programas.

O desenvolvimento de *software* para computadores é realizado utilizando-se uma linguagem de programação que tem como objetivo escrever um código que, tanto o computador como o criador do *software*, entendam. As etapas para o desenvolvimento de um programa de computador são:

- **Análise:** Durante esta etapa, o programador estuda os requisitos necessários para criar o programa, quais os problemas a serem resolvidos e como resolver esses problemas através de um *software*.

- **Algoritmo:** Nesta etapa, o programador cria um esboço do programa por meio de uma linguagem natural e compreensível por seres humanos. Diagramas e fluxogramas também podem ser utilizados aqui para descrever o passo a passo das tarefas a serem executadas pelo computador.
- **Codificação:** Aqui o programador transforma o algoritmo em códigos da linguagem de programação que ele escolheu.

A programação de computadores é uma tarefa padronizada e bem organizada que exige do programador raciocínio lógico e capacidade de solucionar problemas.

Desde as primeiras eras da computação, a programação vem sendo vista como uma atividade reservada apenas a alguns especialistas, capazes de reunir as competências necessárias para tal proeza. No entanto, existem alguns estudiosos que defendem que a programação de computadores é uma atividade acessível a todas as pessoas e deve ser ensinada desde cedo a crianças e adolescentes.

Em 1967, o matemático e educador Seymour Papert criou uma linguagem de programação para crianças chamada LOGO. Ele também cunhou o termo construcionismo. Segundo Papert, o aprendiz constrói o seu conhecimento por intermédio de alguma ferramenta, nesse caso, o computador.

Atualmente existem diversas iniciativas que defendem o ensino da programação de computadores nas escolas regulares, sem restrições. Várias personalidades da tecnologia, como Bill Gates, Mark Zuckerberg, Jack Dorsey, o cantor Will.i.am e políticos como Al Gore e Michael Bloomberg, têm defendido publicamente o ensino de programação nas escolas como forma de inclusão digital. Para eles, interpretar e escrever código é tão importante quanto ler e escrever.

Recentemente o próprio presidente dos Estados Unidos, Barack Obama, disse: “Não compre, apenas, um jogo, crie um. Não se limite a fazer download de uma nova aplicação, ajude a desenvolvê-la. Não jogue no seu celular, programe-o.” (OBAMA, 2013).

Mas é importante salientar que também existem vários especialistas e educadores contra essa tendência. Em entrevista ao site Businnes Insider, Linus Torvalds, responsável pelo kernel do famoso sistema operacional Linux, disse: “Acho que é algo especializado, e ninguém espera que a maioria das pessoas façam isso. Não é como aprender a ler e escrever ou fazer contas básicas de matemática” (LOVE, 2014).

Vera Rita da Costa (2014), em seu artigo intitulado “*Computação para os pequenos*”, publicado no portal UOL, diz que os críticos dessa questão afirmam que o ensino de programação nas escolas pode atrapalhar o desenvolvimento das crianças e adolescentes, privando estes do convívio social e das experiências comuns dessas fases da vida, como brincar e divertir-se ao ar livre.

Segundo Setzer (1988), uma das maneiras mais seguras de fazer uma criança perder sua deliciosa – e necessária – infantilidade é dar-lhe um computador. Ele faz o adolescente perder sua juventude e o torna senil dos pontos de vista mental, emocional e volitivo. Uma das características fundamentais da infância e da juventude é a generalidade; o computador, em virtude de suas peculiaridades, obriga a uma especialização mental precoce.

Este artigo está organizado da seguinte maneira: A seção 2 apresenta um histórico de como tudo começou, o desenvolvimento da linguagem Logo e a teoria do

construcionismo. A seção 3 traz uma relação de projetos atuais que incentivam o aprendizado da programação nas escolas. A seção 4 conclui este artigo.

2 Como tudo começou: a linguagem Logo e o construcionismo de Papert

Entre os anos de 1967 e 1968, Seymor Papert, então diretor do grupo de Epistemologia e Aprendizado do Massachusetts Institute of Technology (MIT), e sua equipe desenvolveram uma linguagem de programação totalmente voltada para a educação, o Logo. Do ponto de vista educacional, é uma linguagem simples, porque possui características que tornam acessível o seu uso por sujeitos de diversas áreas e de diferentes níveis de escolaridade. Computacionalmente, Logo é considerada uma linguagem bastante sofisticada (PRADO, 2000).

A linguagem Logo tem caráter procedural, e sua principal característica é a forma imperativa com a qual se comunica com o computador. As linguagens procedurais são baseadas na lógica formal. Nelas, o sujeito é quem comanda a máquina, indicando o que o computador deve fazer, sem duplicidade de sentidos e de forma ordenada. O sujeito precisa analisar as ações necessárias para realizar cada tarefa, já que existem várias soluções possíveis.

Papert denominou de construcionista a abordagem pela qual o aprendiz constrói, por intermédio do computador, o seu próprio conhecimento (PAPERT, 1996). O construcionismo é uma reconstrução teórica a partir do construtivismo piagetiano feita por Seymour Papert, que estudou durante quatro anos com Piaget no Centro de Epistemologia Genética, em Genebra (CORREIA, SILVA, 2005).

Ao utilizar esse termo, ele se refere a um outro nível de construção do conhecimento, onde o aluno, por intermédio do computador, constrói um objeto de seu interesse, como uma obra de arte, um relato de experiência ou um programa de computador. Nesse caso, o computador requer certas ações que são bastante efetivas no processo de construção do conhecimento (VALENTE, 1993).

O aluno, ao usar as linguagens de programação, transforma seu conhecimento em procedimentos, ou seja, descreve todos os passos necessários para atingir um certo objetivo, para atingir a resolução de um certo problema; em suma, está “ensinando” o computador a atingir um objetivo através de um programa. O computador não fornece os conhecimentos para que o aluno dê respostas (ALMEIDA, 1999).

Nesse contexto também é importante avaliar o papel do erro construtivo. O erro normalmente é tratado como um sintoma da ignorância ou do mau aprendizado. É preciso que se observe o erro sob a perspectiva de um acontecimento rico, que oferece ao aluno e também ao professor a possibilidade de compartilhar ideias e juntos coordenar suas ações, visando identificar o erro e agir para modificá-lo.

Ao desenvolver um programa usando a linguagem Logo, o aluno tem a oportunidade de pensar sobre o que ele está fazendo e criar possibilidades possibilidades para solução de problemas. E é nesse contexto que o erro deve ser encarado como um importante aliado no processo de ensino-aprendizagem.

Há também que se destacar o importante papel do professor no ambiente Logo. O papel de facilitador do aprendizado assumido por ele traz numerosos benefícios, pois, ao assumir que não é o detentor do conhecimento, ele se torna um participante, junto com o

aluno, do processo de criação, e não mais um repassador de informações já prontas.

É preciso também que as atividades propostas promovam a interação, não apenas com o computador, mas também com os colegas da turma, visando criar um ambiente colaborativo, onde a troca de informações e a busca de novas hipóteses se façam presentes (CORREIA, SILVA, 2005).

2.1 O Logo

O Logo é uma linguagem de programação que tem como principal objetivo auxiliar o processo de aprendizado e o raciocínio.

O aluno é levado a pensar na solução de problemas de forma direta ou através da correção de seus próprios erros. Na linguagem Logo, o erro não é tratado como algo ruim, mas como uma tentativa de acerto, o que torna possível a construção do conhecimento de uma forma prática e intuitiva, proporcionando o autoaprendizado do aluno.

No ambiente de programação Logo, originalmente se utilizava uma tartaruga robótica, que andava por uma sala e era direcionada por meio de comandos. Atualmente as versões da linguagem utilizam uma interface gráfica contendo uma tartaruga que anda pela tela do computador (Figura 1). A ideia de se utilizar esse símbolo é despertar a curiosidade e o interesse da criança, fazendo com que a tarefa de programar o computador seja substituída pela tarefa de “ensinar a tartaruga”.

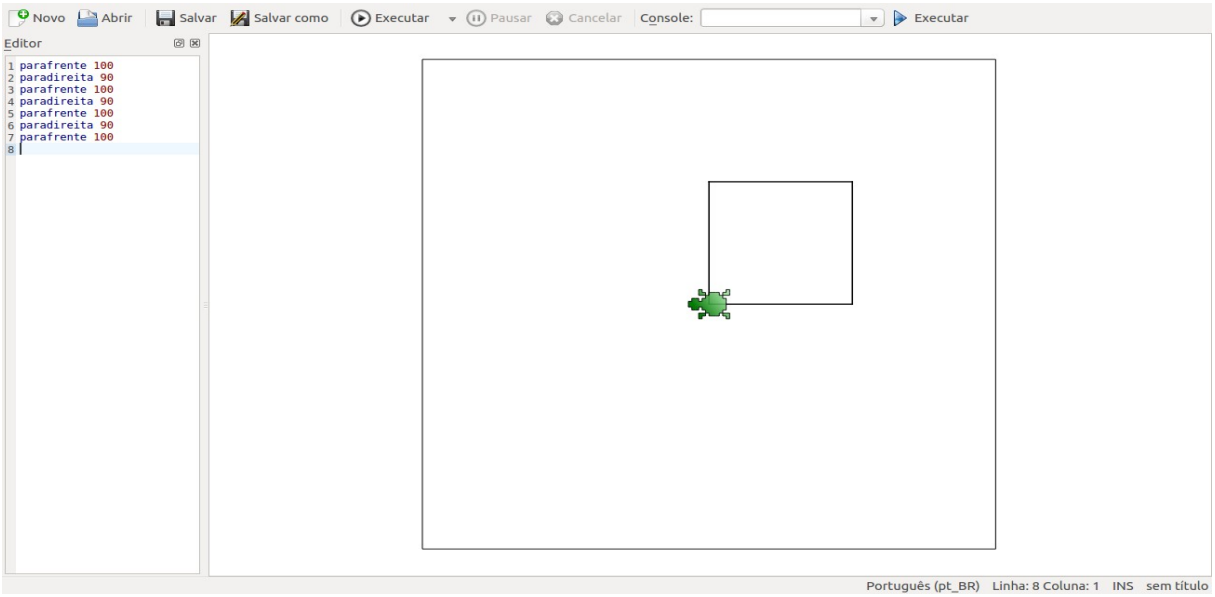

Figura 1: O ambiente de programação Logo: Kturtle.

Os comandos básicos para movimentar a tartaruga são os mesmos utilizados para realizar uma caminhada, ou seja, andar para frente, andar para trás, virar para a esquerda, virar para a direita. No caso da linguagem Logo, são utilizadas as palavras-chave parafrente e paratrás acompanhado do tamanho do passo, e também paradireita e paraesquerda acompanhado do ângulo do giro. Por exemplo, para fazer a tartaruga andar 50 passos para frente, basta digitar no editor: parafrente 50. Desse modo, ela irá se

deslocar 50 passos para frente. Da mesma forma, também é possível andar para trás digitando no editor o comando: paratrás 50. Para realizar um giro de 90º para a direita, digita-se o comando paradireita 90. E, para girar a tartaruga para a esquerda os mesmos 90º, digita-se o comando: paraesquerda 90.

Ao se movimentar pela tela, a tartaruga desenha uma linha reta do tamanho exato dos passos que foram expressos no comando parafrente ou paratrás. Dessa forma, é possível desenhar várias figuras geométricas, como um quadrado, por exemplo. Para desenhar o quadrado, o aprendiz pode digitar os comandos passo a passo, como pode ser visto na Figura 1, ou utilizar um comando chamado repita, que repete n vezes os comandos inseridos (Figura 2).

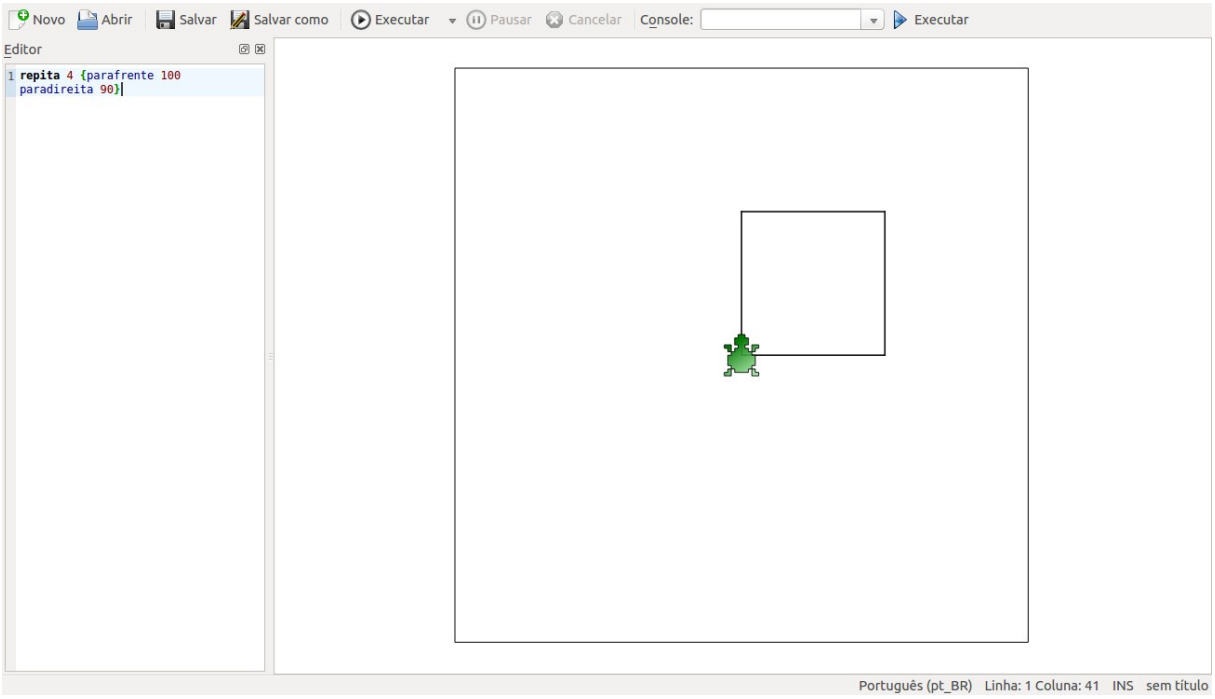

Figura 2: Comando repita.

Existem diversos outros comandos que podem ser utilizados para criar os mais diferentes tipos de desenhos no Logo. Há também a possibilidade de criar procedimentos, que são um conjunto de instruções inseridas pelo usuário, que recebem um nome e podem ser executadas dentro da linguagem para realizar tarefas. Procedimentos são funções como aquelas encontradas em outras linguagens de programação. Por exemplo, ao inserir todos os comandos necessários para criar a figura geométrica de um quadrado, é possível criar um procedimento chamado quadrado e salvar na memória do computador. Toda vez que o aluno precisar desenhar um quadrado de tamanho fixo, ele escreverá o nome do procedimento e a linguagem executará todos os passos para desenhar a figura.

Ainda hoje, a linguagem Logo é bastante utilizada em escolas espalhadas pelo mundo, não só como ferramenta de aprendizagem para iniciantes em programação, mas principalmente para auxiliar o professor no processo de aprendizado e raciocínio de outros conceitos incluídos em diversas áreas: matemática, linguagem, música e robótica.

3 O ensino de programação hoje

3.1 O Scratch

O Scratch é uma nova linguagem gráfica de programação criada no Media Lab do Instituto de Tecnologia de Massachusetts (MIT), inspirada nas linguagens Logo e Squeak, mas que pretende ser mais simples, fácil de utilizar e mais intuitiva (SCRATCH, 2012).

A linguagem foi feita para que as crianças criem programas sem a necessidade de digitar códigos complicados. Em vez disso, elas programam através de blocos de comandos que são encaixados uns nos outros, formando um conjunto de instruções.

É um *software* gratuito e possui uma IDE[1] onde não é preciso digitar funções, endereços, etc. Seu objetivo primário é facilitar a introdução de conceitos de matemática e de computação, enquanto também induz o pensamento criativo, o raciocínio sistemático e o trabalho colaborativo (SCRATCH, 2012).

O Scratch (Figura 3) permite que sejam criados programas que controlam e misturam imagens, animação, texto, música e som.

Segundo Mitchel Resnick, criador da linguagem, “Quando estudam programação, as pessoas não só aprendem a programar, como também programam para aprender”. Ele defende que noções de programação devem ser ensinadas desde cedo às crianças (SERRANO, 2014).

Para Resnick, “Essas habilidades serão úteis não apenas para cientistas da computação mas para qualquer pessoa, independentemente da idade, da experiência, do interesse ou da profissão que optar por seguir.” (SERRANO, 2014).

O ambiente de programação Scratch é dividido em 3 blocos. No primeiro bloco, existem os comandos que podem ser adicionados ao programa. Há comandos para controle, movimentos, operações, aparência, sons e outros. O segundo bloco exibe o programa que está sendo criado, ficando nesta parte todos os blocos de comandos adicionados. No terceiro bloco, há uma tela para exibição da animação, que é onde o programa será executado.

1 IDE, do inglês *Integrated Development Environment* ou Ambiente Integrado de Desenvolvimento de *software*.

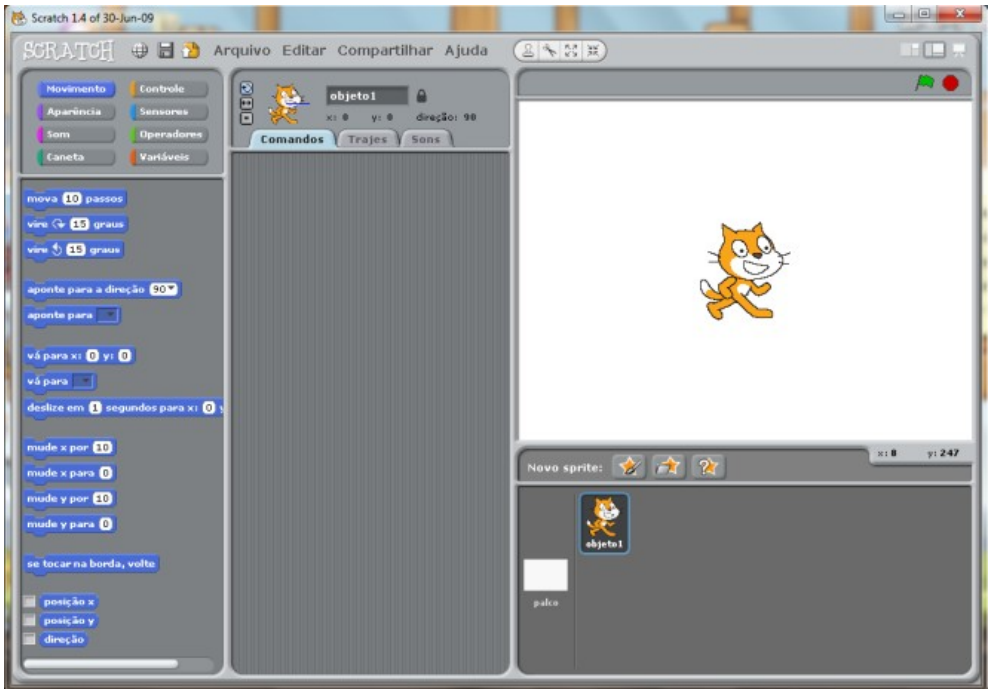

Figura 3: Tela Inicial do Scratch.

Na linguagem Scratch, os programas são criados através de blocos de comandos, que são encaixados uns nos outros, formando a sequência de comandos que se deseja. Esses blocos se encaixam apenas de uma única forma, característica esta que evita erros de sintaxe no programa, já que não há outra maneira de os blocos serem encaixados.

Na página web[2] da linguagem, é possível inserir e também visualizar programas criados por outras pessoas, experimentar, reutilizar e ainda adaptar imagens e scripts.

O Scratch ajuda jovens a aprender de maneira criativa, refletir de maneira sistemática e trabalhar de forma colaborativa – habilidades essenciais para a vida no século 21 (SCRATCH, 2014).

3.2 O Code Club

O Code Club, ou Clube do Código, em português, foi criado pela professora e designer britânica Clare Sutcliffe e pela desenvolvedora Linda Sandvik, em abril de 2012. Elas lançaram o projeto com o plano de chegar a um quarto das 21.000 escolas da Grã-Bretanha até 2015. Para isso, elas contam com o apoio do Google, e hoje já são 1.668 clubes no Reino Unido e 142 em outros países. O Code Club é uma rede mundial de atividades extracurriculares gratuitas, gerenciada por voluntários, e tem como principal objetivo ensinar programação de computadores a crianças.

A filosofia do Code Club é diversão, criatividade e aprendizagem pela descoberta. É importante que as crianças curtam o tempo que elas passam no Code Club e que não se sintam como se estivessem em mais uma aula da escola. Essas crianças devem entender que são responsáveis pelo que o computador faz, e podem (e devem) fazer com

2 <http://scratch.mit.edu/>.

que o computador faça o que elas querem, não o contrário (CODE CLUB BRASIL, 2013).

Para ensinar programação como atividade extracurricular nas escolas, voluntários utilizam projetos criados pelo Code Club: são jogos, animações e páginas da internet. Qualquer pessoa pode se tornar um voluntário, criar um clube próximo de sua casa e ministrar as aulas de programação, uma hora por semana. Os cursos 1 e 2, disponibilizados pelo Code Club, utilizam o Scratch como ferramenta para ensinar as bases da programação de computadores. O curso 3 é uma introdução ao desenvolvimento web e utiliza HTML e CSS. O curso 4 ensina a linguagem de programação Python.

No Brasil, o Code Club nasceu da iniciativa de Heverton Herman, que em 2013 viu um vídeo chamado “O que as escolas não ensinam”, produzido pelo projeto Code.org, e resolveu traduzir alguns vídeos. Ele então entrou em contato com as criadoras do projeto na Inglaterra e montou o site brasileiro (Figura 4).

Figura 4: Logo do website Code Club Brasil.

3.3 O Codecademy

Criado pelos americanos Zach Sims e Ryan Rubinski, o Codecademy (Figura 5) é um site que oferece aulas gratuitas de programação de computadores através de uma plataforma interativa.

Segundo Zach Sims, “Programação é parte das linguagens necessárias no século 21. Nós achamos que programar ajuda a pessoa a ter diferentes pontos de vista e alcançar seus sonhos” (GOMES, 2012).

Figura 5: Logomarca do website Codecademy.

As aulas funcionam de maneira que os alunos progridam nas lições passo a passo.

Dessa forma, cada nível ensina a trabalhar com uma quantidade maior de códigos à medida que se aprende.

O site premia os alunos com medalhas para cada exercício concluído. Para isso, há um sistema de pontuação que mostra o total de pontos obtidos e cria um hanking dos melhores resultados.

Outro recurso muito interessante é a possibilidade de qualquer pessoa criar um novo curso através de uma ferramenta chamada “Criador de Curso”. Atualmente o site, originalmente em inglês, já conta com várias traduções para outros idiomas, inclusive o português.

3.4 O Code.org

O Code.org foi criado pelo iraniano Hadi Partovi, formado em Ciências da Computação pela Universidade de Harvard e grande investidor de empresas como Facebook, Dropbox e Airbnb. Em 2011 ele teve a idéia de criar um vídeo para estimular jovens a aprender programação, usando depoimentos de personalidades do mundo da tecnologia. O primeiro a gravar um depoimento seria Steve Jobs, mas ele morreu antes de o projeto começar. Então, Partovi chegou à conclusão de que deveria fazer algo mais efetivo do que apenas gravar os vídeos. Com a ajuda de seu irmão Ali, lançou em fevereiro de 2013 o projeto para promover o ensino de programação nas escolas. No ínicio o Code.org recebeu o apoio de 60 pessoas, entre elas Bill Gates, Mark Zuckenberg, Jack Dorsey e ainda o cantor Will.i.am e alguns políticos como Al Gore e Michel Bloomberg. Conseguiram então o apoio financeiro do Google, da Microsoft, do Amazon e do Linkedln. Segundo Partovi, a decisão de dedicar-se integralmente ao projeto veio depois de saber que o vídeo sobre o projeto havia recebido mais de 11 milhões de visualizações na internet. Em 9 de dezembro de 2013, o Code.org lançou a iniciativa “Hour of Code 2013”, em português, “A hora do código 2013”, em que qualquer pessoa poderia dedicar 1 (uma) hora para aprender a programação de computadores.

O Code.org mantém um website (Figura 6) onde o aluno em questão pode aprender os primeiros passos na programação de computadores através de uma plataforma de tutoriais, inclusive em língua portuguesa. No tutorial para iniciantes, o aluno tem o primeiro contato com a programação de computadores através da linguagem Blockly, que possibilita arrastar e soltar blocos para escrever os códigos. Segundo o website, grande parte do código é digitado, mas o Blockly é visual, e cada bloco corresponde a uma linha de código real. O objetivo do primeiro programa é conseguir que o personagem Angry Bird atravesse um labirinto para chegar ao Porco Verde. O lado esquerdo do labirinto apresenta a área onde o programa é executado. Abaixo desse labirinto pode-se encontrar as instruções para cada quebra-cabeça. No meio há uma caixa de ferramentas que possui os comandos necessários para movimentar o personagem. À direita fica o local chamado de área de trabalho, para onde o aluno arrastará os blocos da caixa de ferramentas para construir o programa. Para apagar os blocos, é possível arrastá-los para uma lixeira que fica no canto da tela.

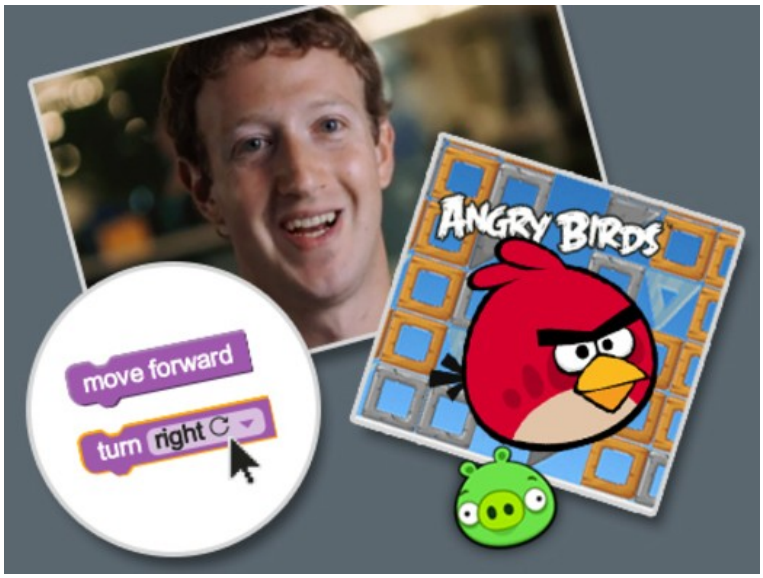

Figura 6: Ambiente de aprendizagem do Code.org.

Cada bloco é uma instrução: se o aluno arrastar o bloco onde está escrito “Avançar” para a área de trabalho e depois pressionar o botão “Executar programa”, o personagem irá se mover em um determinado espaço no labirinto.

Para que o personagem faça outras ações, é só adicionar outros blocos à área de trabalho e juntá-los: cada ação será executada de cima para baixo.

Caso o programa não funcione corretamente, é possível corrigi-lo através do botão “Recomeçar” (CODE.ORG, 2014).

Além do tutorial para iniciantes, é possível encontrar outros tutoriais para aprendizagem de linguagens, como JavaScript, aplicativos para dispositivos móveis e até uma introdução à robótica.

3.5 Programar não é para todos

Os projetos que visam ensinar programação para crianças e jovens pelo mundo não são uma unanimidade entre especialistas e nem entre pessoas ligadas à educação.

Recentemente, Linus Torvalds, o criador do sistema operacional Linux, disse em uma entrevista (LOVE, 2014): “Na verdade, eu não acredito que todos devem necessariamente tentar aprender a programar […]. Não é como saber ler e escrever e fazer contas básicas”.

“Eu acho que as pessoas devem ter alguma forma de obter mais contato, mas apenas para que possam descobrir por si mesmas se têm aptidão para isso”. Para Linus, essa seria a justificativa para ensinar programação de computadores nas escolas (LOVE, 2014).

O argumento de alguns especialistas da área de Tecnologia da Informação (TI) é o mesmo apresentado por Torvalds: para eles, a programação de computadores é algo que requer raciocínio lógico apurado e capacidade de resolver problemas com alto grau de complexidade, características que nem todas as pessoas possuem.

Nesse sentido, o ensino de programação nas escolas não produziria bons resultados, pois os desenvolvedores formados no ambiente escolar não seriam bem qualificados.

O mercado de TI seria então inundado de pessoas mal preparadas, e a qualidade dos serviços nessa área poderia ser prejudicada.

O americanizado Jeff Atwood, programador e blogueiro que mantém um site de perguntas e respostas chamado *Stack Overflow*, é uma das principais vozes nos EUA contra o ensino de programação nas escolas. Em uma entrevista ao jornal Zero Hora, ele disse: “Os geeks (fãs de tecnologia) estão conquistando o mundo! Mas o objetivo final de qualquer programador é fazer o seu trabalho tão bem a ponto de que outras pessoas não precisem escrever códigos para resolver os seus problemas, ou produzir trabalhos artísticos, escrever textos ou gerenciar os seus negócios” (ATWOOD, 2013).

Em outra frente, alguns educadores e especialistas na área apresentam seus argumentos contra o ensino de programação nas escolas.

Segundo Setzer, a introdução precoce de computadores prejudica a infância e a juventude e pode causar desastres mentais nas pessoas que, por serem mentais, não podem ser vistos (SETZER, 2002).

Ainda segundo Setzer, “Ao usar o computador, a criança é obrigada a exercer um tipo de pensamento que deveria empregar somente em idade mais avançada. Com isso podemos dizer que os computadores roubam das crianças sua necessária infantilidade. Elas são obrigadas a pensar e usar uma linguagem que deveria ser dominada apenas por adultos” (SETZER, 2002).

Muitos educadores acreditam nessa teoria e afirmam categoricamente que o ensino de programação prejudica a infância, por retirar delas a oportunidade de socialização com outras crianças através das brincadeiras e jogos próprios dessa fase.

4 Considerações finais

Ao analisar o contexto histórico do ensino de programação de computadores, é possível observar que, desde as experiências com a linguagem Logo de Papert até os dias de hoje, existe um forte argumento a favor da inclusão dessa disciplina nas escolas de ensino regular.

Como afirma José Armando Valente, as características dos computadores, que incluem a expressão do que o aprendiz está pensando em termos de uma linguagem formal e precisa e a execução do que ele está pensando em termos de resultados fiéis e imediatos, estão presentes nas atividades de programação e auxiliam o aprendiz a alcançar a fase de compreensão de conceitos. Ele pode refletir sobre os resultados de suas ações e ideias, e essa reflexão é o mecanismo pelo qual o aprendiz se torna consciente de seu conhecimento e, assim, pode transformar seus esquemas mentais em operações e noções mais complexas (VALENTE, 1999).

As crianças, hoje, já nascem imersas em um mundo digital, mas, ao contrário do que se possa imaginar, elas não conhecem o funcionamento desse mundo, apenas utilizam suas ferramentas passivamente.

Segundo Douglas Rushkoff, “aprender uma linguagem de programação nos dias atuais é tão importante quanto aprender a ler e escrever, pois, na chamada sociedade da

informação, a distância entre usar um programa e criar um tornou-se cada vez mais ampla, a ponto de as pessoas não saberem mais o que está acontecendo atrás da tela do computador; portanto qualquer pessoa que saiba criar um programa de computador será capaz de criar a realidade em que o restante estará inserido" (RUSHKOFF, 2012).

Para Rushkoff, "Quando nós humanos adquirimos linguagem, aprendemos não somente a ouvir, mas a falar. Quando ganhamos a escrita, nós aprendemos não apenas a ler, mas a escrever. E na medida em que nos movemos em direção a uma realidade crescente digital, nós precisamos aprender não apenas a usar programas, mas a fazê-los também. No panorama emergente, altamente programado, ou você criará o *software* ou será o *software*. Simples assim: programe ou será programado" (RUSHKOFF, 2012).

O criador da linguagem Scratch, Mitchel Resnick, afirma que "Quando estudam programação, as pessoas não só aprendem a programar como também programam para aprender" (SERRANO, 2014).

Essas afirmações reforçam a importância da aprendizagem das metodologias e técnicas utilizadas para a programação de computadores não só para quem deseja se profissionalizar na área, mas por todas as pessoas.

Mas é importante ressaltar que a inclusão do ensino de programação nas escolas deve respeitar o desenvolvimento cognitivo e social das crianças. A escola, além de local de aprendizagem, é um espaço onde relações pessoais são construídas através da interação entre os colegas de turma e também com os professores. As brincadeiras e jogos próprios da infância, apesar de fortemente influenciados pelo uso das novas tecnologias, não devem ser esquecidos. As crianças não devem ser privadas do convívio social com outras crianças e também com adultos, pois esse contato é essencial para a formação do indivíduo.

Respeitando-se os estágios de desenvolvimento intelectual das crianças, é possível incluir o ensino de programação de computadores nas escolas sem causar prejuízo nenhum ao aluno e trazer vários benefícios ao processo de ensino e aprendizagem de outras disciplinas.

Referências

ALMEIDA, M. E. B. de. *Informática e Formação de Professores*. Coleção Informática para a mudança na Educação. MEC/SEED/Proinfo, 1999.

ATWOOD, J. *Equiparar programação a ler e escrever não é bom*. Zero Hora, Porto Alegre, 24 Mar. 2013.

CODE CLUB BRASIL. *O que é?*. Disponível em: <http://codeclubbrasil.org/oquee/>. Acesso em: 31 jul. 2014.

CODE.ORG. *Learn Code*. Disponível em: <http://learn.code.org/hoc/1>. Acesso em: 31 jul. 2014.

CORREIA, L. H. A.; SILVA, A. J. de C. *Computador Tutelado*. Lavras: UFLA/FAEPE, 2005.

COSTA, V. R. *Computação para os pequenos*. Disponível em: <http://cienciahoje.uol.com.br/alo-professor/intervalo/2014/05/computacao-para-os-pequenos> . Acesso em: 30 out. 2014.

GOMES, P. *Codecademy ensina programação de graça*. <http://porvir.org/porcriar/codecademy-ensina-programacao-de-graca-pelo-mundo/20120828>. Acesso em: 31 jul 2014.

OBAMA, B. Don't Just Play on Your Phone, Program It. *The White House Blog*. 2013. Disponível em: <http://m.whitehouse.gov/blog/2013/12/09/don-t-just-play-your-phone-program-it>. Acesso em: 31 jul. 2014.

PAPERT, S. Looking at Technology Through School-Colored Spectacles. *MIT Media Lab*, 1996.

PRADO, M. E. B. B. *LOGO* – Linguagem de Programação e as Implicações Pedagógicas. Campinas: UNICAMP, 2000.

RUSHKOFF, D. As 10 questões essenciais da era digital. São Paulo: Editora Saraiva, 2012. SCRACTH. *ABOUT Scratch (Scratch Documentation Site)*. Disponível em: <http://info.scratch.mit.edu/About_Scratch>. Acesso em: 31 jul 2014.

SERRANO, F. Geração Geek. *Revista Exame Informática*. São Paulo: Editora Abril, 2014.

SETZER, V. W. O computador no Ensino: Nova Vida ou Destruição?. In: CHAVES, E. O. C.; SETZER, V. W. *O uso de computadores em escolas*: Fundamentos e Críticas, São Paulo: Ed. Scipione, 1988, p. 69-127. Disponível em: <http://www.ime.usp.br/~vwsetzer/computador-no-ensino.html>. Acesso em: 30 out. 2014.

SETZER, V. W. Computadores na educação: por quê, quando e como. In: SETZER, V. W. *Meios Eletrônicos e Educação*: uma visão alternativa, 2a. ed. São Paulo: Escrituras, 2002, p. 85-134. Disponível em: <http://www.ime.usp.br/~vwsetzer/PqQdCo.html>. Acesso em: 10 nov. 2014.

LOVE, D. *A Conversation With Linus Torvalds, Who Built The World's Most Robust Operating System And Gave It Away For Free*. Jun. 7, 2014. Disponível em: <http://www.businessinsider.com/linus-torvalds-qa-2014-6>. Acesso em: 31 jul. 2014.

VALENTE, J. A. *Computadores e conhecimento: repensando a educação*. Campinas: UNICAMP. 1993.

VALENTE, J. A. (Org). *O computador na sociedade do conhecimento*. Campinas, SP: UNICAMP/NIED, 1999.