Programação no Ensino Médio: Uma Abordagem de Ensino Orientado ao *Design* com Scratch

Pasqueline Dantas Scaico, Anderson Alves de Lima, Jefferson Barbosa Belo da Silva, Silvia Azevedo, Luiz Fernando Paiva, Ewerton Henning Souto Raposo, Yugo Alencar, Jão Paulo Mendes[1]

Centro de Ciências Aplicadas e Educação – Universidade Federal da Paraíba (UFPB) - Rua da Mangueira, s/n - CEP 58.297-000 – Rio Tinto – PB – Brasil

{pasqueline, anderson.alves, jefferson.bello, silvia.azevedo, luiz.fernando, ewerton.raposo, yugo.alencar, paulo.mendes)@dce.ufpb.br

Abstract. This paper presents a project for teaching programming in high school which intends to instruct students about what is the Computer Science and develop skills through algorithmic thinking from a design-oriented approach. Also, it's reported one Programming Contest, which served to motivate students and to reveal the knowledge they acquired.

Resumo. Este trabalho apresenta os objetivos de um projeto para o ensino de programação no ensino médio que visa instruir os estudantes sobre o que representa a Computação e desenvolver habilidades através do pensamento algorítmico através de uma abordagem de ensino orientado ao *design*. O artigo também relata uma olimpíada de programação que serviu para motivar os participantes do curso e para revelar os conhecimentos adquiridos.

1. Introdução

A Computação evoluiu imensamente nas últimas décadas e concentrou esforços em diversas áreas de pesquisas que tornaram as tecnologias cada vez mais eficientes e confiáveis. A dependência da sociedade moderna é tamanha que estudos estimam que até o ano de 2018 haja a demanda de um milhão e meio de profissionais na área de tecnologia. Diante deste cenário, já se prevê escassez de mão-de-obra qualificada para assumir tais postos de trabalho (Wilson 2010). Muitos estudos mostram que o desinteresse por parte dos estudantes pela área de Computação é reforçado por informações imprecisas e a existência de estereótipos.

Aprender a programar é extremamente importante, principalmente, ao se considerar que o desenvolvimento de algoritmos é o eixo central para todas as suas áreas relacionadas da Computação. Contudo, não é uma tarefa simples porque requer o domínio de um conjunto amplo de habilidades técnicas, tais como o conhecimento de linguagens de programação, de ambientes para a construção do código, de embasamento matemático, e de outras capacidades que estão mais relacionadas a aspectos cognitivos e psicológicos, a exemplo da habilidade de representar bem o problema e saber decompô-lo, de construir o seu significado, de criar modelos mentais que auxiliem a busca da solução, de abstrair conceitos e reconhecer padrões (Sudol 2011). Assim, o quanto antes essas habilidades puderem ser desenvolvidas, melhor será a visão e as atitudes que os

ISSN: 2316-6541

estudantes que concluem o ensino básico terão sobre a área de tecnologia e suas carreiras.

Ainda são poucos os trabalhos que relatam experiências do ensino de programação no ensino médio. Uma linguagem citada em cursos introdutórios de programação é a Python, que tem a sintaxe simples, blocos delimitados por endentação e feedbacks imediatos, conforme menciona Lima Ribeiro et al (2011, p. 5). Segundo Elkner (1999), ensinar e compreender Python torna-se mais fácil, fazendo com que a sintaxe não seja mais uma barreira para que se construam os algoritmos. Marques et al (2011) cita sua experiência ao conduzir uma oficina de introdução a programação com Python para alunos do ensino médio, em que diversos jogos foram codificados e mostra que o uso de jogos foi um fator bastante motivacional. Por outro lado, Rapkiewicz (2006) afirma que Python ainda é complexo para os iniciantes em programação, mesmo sendo mais simples que diversas outras linguagens. O uso da linguagem Scratch na introdução de conceitos algorítmicos tem sido bem vista. Para Trajano (2010), ela é uma linguagem adequada para entender as principais estruturas, que são a base para a criação de algoritmos.

Este trabalho apresenta uma experiência que tem introduzido o ensino de programação em uma escola pública de ensino médio. Todavia, a abordagem utilizada concentrou os esforços não apenas em ensinar uma linguagem de programação, mas estabelecer situações onde os alunos pudessem começar a desenvolver capacidades importantes para que no futuro, possam se tornar bons programadores. Uma das principais ações do projeto culminou com uma olímpiada de programação entre os estudantes, que apresentou resultados bastante positivos e motivadores para eles e para a equipe que desenvolveu o projeto.

O trabalho está organizado da seguinte maneira: a Seção 2 procura explicar a motivação para a escolha da linguagem de programação utilizada. A Seção 3 descreve a metodologia que conduziu o planejamento e execução das atividades do curso. A Seção 4 descreve em detalhes uma das principais do projeto, uma olimpíada de programação e seus desafios, que exploraram diversas capacidades dos alunos e serviu para validar a metodologia de ensino utilizada. Por fim, a Seção 5 trata das considerações finais.

2. A Linguagem Scratch

Aprender uma linguagem de programação é uma tarefa desafiadora. Todavia, tornar o ensino de programação mais acessível para um maior número de indivíduos é algo importante porque é capaz de estimular muitas capacidades cognitivas e para que aquele que aprende possa aplicar as técnicas utilizadas na programação na resolução de diversos tipos de problemas, nas mais distintas profissões.

Segundo Pausch e Kelleher (2005), desde o início dos anos 1960, pesquisadores tentam desenvolver linguagens de programação e ambientes com a intenção de tornar a programação acessível para um número maior de pessoas. A principal dificuldade encontrada pelos programadores iniciantes é a questão da sintaxe, pois eles precisam compreender a linguagem que o computador entende e também o idioma, já que algumas sintaxes que possuem a clareza suficiente para que o novato entenda a funcionalidade do uso de seus comandos. Esses autores também afirmam que é preciso simplificar a sintaxe das linguagens, para que estas fiquem mais próximas da linguagem natural.

ISSN: 2316-6541

O ambiente Scratch é uma linguagem que contribui para o aprendizado de programação através de um conceito inovador de desenvolvimento orientado ao *design*, que privilegia a Computação Criativa, a qual reconhece o conhecimento e as práticas que os jovens precisam desenvolver para criar software que sejam provenientes dos seus interesses pessoais. Apenas para efeito de ilustração do potencial pedagógico dessa linguagem, a sintaxe de um comando de repetição na linguagem Scratch, Python e Pascal é apresentada na Figura 1.

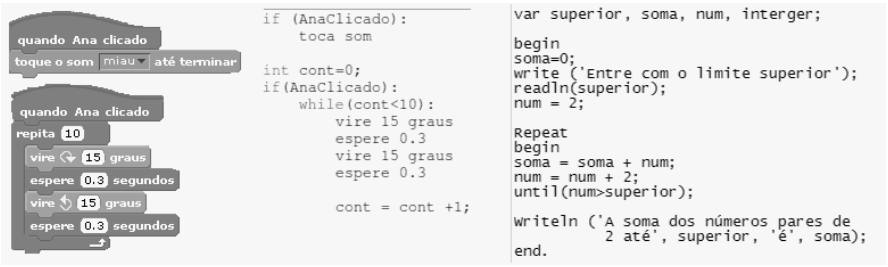

Figura 1. Laço de repetição no Scratch, Python e Pascal

É possível notar como Scratch pode ser mais didática para iniciantes. Ao analisar o trecho de código, percebe-se que o fato dos comandos em Scratch se basearem em uma estrutura de blocos de montar, o aluno pode concentrar seu esforço na busca pela construção do algoritmo, o que não é o caso das outras duas linguagens, cuja sintaxe para o comando de repetição requer uma série de detalhes, que podem dificultar o processo de compilar o algoritmo. Assim, é possível supor que uma parte do esforço esteja na correção da sintaxe e não na lógica de programação, por exemplo, o que pode ser um fator desmotivador para o aprendiz.

Especialmente para a realidade das escolas públicas brasileiras, o Scratch também é bem mais adequado pelo fato de que usa o idioma nativo, que é o português. Sendo assim, o caráter mais didático do Scratch projeta no aluno a possibilidade dele se concentrar no exercício do pensamento algoritmo e na criatividade para a construção das soluções. Não ter que se preocupar com questões importantes, mas acessórias à programação, como é o caso da correção de problemas de sintaxe, é algo que instiga o aluno a se empenhar mais na criação dos códigos.

3. Descrição da metodologia utilizada

O trabalho descrito neste artigo é o desdobramento de um projeto que está sendo desenvolvido há um ano em uma escola pública no interior da Paraíba, que tem como objetivo introduzir o pensamento computacional e algorítmico para as turmas de 2º e 3º ano do nível médio. Antes de iniciarem o curso de programação, os estudantes participaram de diversas atividades que trataram de temas relacionados à tecnologia e que estimularam o pensamento computacional através de competições de equipe, a partir de *atividades desplugadas* (Scaico et al, 2012).

A metodologia utilizada para proporcionar o ensino de programação para alunos que não possuíam qualquer tipo de experiência com a Computação baseou-se em uma proposta que procurou estimulá-los a desenvolver pequenos softwares, como jogos simples e animações.

Foram utilizados alguns princípios de uma abordagem de ensino de programação que se baseia no conceito de *design*, onde se 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) na elaboração dos algoritmos. Pode-se dizer, então, que a aprendizagem baseada no conceito de design é particularmente adequada para a Computação Criativa, que enfatiza que o conhecimento e as técnicas que os jovens precisam desenvolver para criar um software devem ser provenientes dos seus interesses pessoais. Para assegurar que os estudantes teriam a atenção necessária em sala de aula, formaram-se quatro turmas contendo cada uma delas oito alunos, que ficaram sob a responsabilidade de uma dupla de professores. É válido mencionar que os professores são estudantes do curso de Licenciatura em Computação da Universidade Federal da Paraíba. O público-alvo do curso de programação foi um grupo de 32 alunos com idade entre 15 a 19 anos, interessados em aprofundar os estudos e que não possuíam conhecimentos prévios sobre a área de Computação, apenas eram usuários de informática e da Internet.

O curso foi elaborado com carga horária de 20 horas (que foi distribuída ao longo de dez semanas, sendo, portanto, duas horas de aulas semanais). O objetivo principal da disciplina foi o apresentar aos estudantes as principais estruturas de uma linguagem de programação e de praticar técnicas utilizadas na programação, como é o caso do uso da abstração, depuração de erros, testes e melhoria de algoritmos simples.

Cada aula teórica foi seguida de duas aulas práticas, onde os estudantes foram estimulados a praticar os comandos do Scratch e a desenvolver capacidades relacionadas à descoberta de erros nos códigos, trabalho colaborativo (completar códigos parcialmente implementados) e criar projetos usando a sua criatividade, percebendo e se expressando através da construção de animações sobre aquilo o que estava à sua volta. Os planos de aula, exercícios utilizados, os códigos desenvolvidos e detalhes da metodologia utilizada podem ser acessados no endereço web (http://www2.ccae.ufpb.br/pibid).

Apesar de existirem quatro turmas com professores diferentes, todos seguiram os mesmos planos de aula. Em relação à avaliação, utilizaram-se os mesmos exercícios de fixação para que se pudesse identificar as disparidades entre as turmas. Um ambiente colaborativo foi estabelecido através de uma lista de discussão para que as experiências fossem trocadas semanalmente entre os professores. Além disso, criou-se um repositório de conteúdo, que podia ser acessado através de uma conta do Dropbox. Ainda para efeito de avaliação de aprendizagem foi organizada uma olimpíada de programação, que está detalhada na Seção 4.

4. Olimpíada Interna de Programação com Scratch (OIPS)

A competição foi organizada nos moldes das outras olimpíadas científicas existentes, como a Olimpíada Brasileira de Informática (OBI). O objetivo da OIPS foi o de despertar nos alunos o interesse por uma ciência importante para a sua vida prática e sua formação básica, no caso, a Computação, além de registrar os resultados de aprendizagem dos alunos e as suas principais dificuldades.

A olimpíada foi realizada em uma única modalidade e os alunos competiram individualmente. A prova foi composta por cinco questões, que possuíam diferentes níveis de dificuldade, desde desafios simples de lógica até implementações completas que exigiam raciocínio, conhecimentos técnicos e criatividade. Cada questão teve seu peso definido em função da sua complexidade. A competição teve duração de três

ISSN: 2316-6541

horas. Para que se possa entender como diferentes habilidades e técnicas de programação foram exigidas na OIPS, e como elas foram o reflexo do que aconteceu nas aulas, as subseções a seguir descrevem cada questão utilizada na competição em detalhes.

4.1. Questão 1: Ordenação de Instruções

A ideia da questão "Ordenando as instruções" foi baseada em questões do ensino fundamental, onde o professor apresenta um conjunto palavras desordenado para os alunos e estes devem criar frases com sentido real usando o conjunto fornecido. Ao aluno foram fornecidos vários fragmentos de códigos, em que alguns executavam comandos explicados durantes as aulas e outros, estavam erroneamente empregados no código fornecido. Esta questão exigia que o aluno identificasse que comandos eram imperativos para a correta implementação do código e ele que ordenasse os comandos, de forma que o código passasse a executar uma ação com sentido (Figura 1). O intuito desse desafio foi avaliar o grau de abstração dos alunos com relação ao que era necessário para a solução do problema, que é uma atitude fundamental para a formação de um bom programador.

Figura 1. Exemplo de possíveis algoritmos que poderiam ser formados

Para executar a questão, um conjunto de regras foi definido: i. só era permitido usar códigos que faziam parte dos comandos distribuídos, ii. nem todos os comandos fornecidos eram necessários para a construção da solução e iii. a ordenação dos comandos não era única, visto que existia mais de uma maneira de implementar um determinado algoritmo. Para a construção do algoritmo era essencial que o mesmo fosse o mais limpo possível. Os alunos aplicaram conceitos trabalhados durante as aulas que tratavam muito superficialmente do desempenho de algoritmos, mas que despertavam a importância deste assunto para a Ciência da Computação. A avaliação da questão foi realizada através da criatividade de codificação do aluno, se a solução fornecida obedecia ao esperado e se o código produzido utilizaria apenas os comandos necessários.

4.2. Questão 2: Reproduza a animação

Para resolver a questão o competidor tinha acesso a um vídeo com uma animação feita no Scratch. Depois, exigia-se que o aluno reproduzisse a animação vista, utilizando estruturas e comandos existentes no ambiente Scratch (Figura 2). Esse desafio teve como objetivo apurar a capacidade dos alunos em identificar, analisar e reproduzir uma solução para um problema com um resultado já pré-definido, capacidade essa muito exigida em olímpiadas de programação. A animação tinha características importantes

que deveriam ser bem analisadas para a correta reprodução, tais como estrutura de repetição e sincronismo no diálogo dos personagens.

Figura 2. Animação disponibilizada em vídeo, que deveria ser reproduzida

Definiu-se que esta questão possuía nível intermediário de dificuldade e para a sua análise considerou-se se a solução fornecida continha ou não todos os detalhes necessários para a correta reprodução do vídeo mostrado aos competidores.

4.3. Questão 3: Jogo de toca

O *jogo de toca* é uma brincadeira infantil muito conhecida. De modo geral, o jogo consiste em dois tipos de jogadores: os pegadores e os que devem evitar ser apanhados. Na Olimpíada do Scratch foi solicitado que os alunos programassem parte desse jogo como desafio para a competição. O objetivo principal dessa questão foi testar o conhecimento dos competidores em relação aos comandos relacionados às estruturas de controle do Scratch e de aparência.

Foi disponibilizado para cada competidor um código que possuía fragmentos de parte da implementação do jogo, restando a ele apenas completar o comportamento que faltava. No código fornecido já estava implementada a movimentação do protagonista de um personagem, a fuga do outro personagem e o sistema de pontuação, responsável por alterar o placar. Esperava-se dos competidores que eles montassem a estrutura condicional que permitiria a reação dos personagens conforme as regras que foram estipuladas no inicio do jogo: i. quando a pontuação do jogador fosse zero, ii. o personagem (gato) deveria emitir uma mensagem informando que o jogo estaria começando, iii. quando o *score* fosse 10 informar que já tinha 10 pontos e lançar uma mensagem incentivando o jogador continuar a jogar e iv. aos 15 pontos ele deveria emitir um aviso informando que o jogo estava finalizado, fazendo o outro personagem (bruxa) sumir, como se pode ver na Figura 3.

Figura 3. Imagens da animação "Jogo de toca"

Essa questão foi classificada com nível de dificuldade intermediário. A avaliação da mesma ocorreu através da análise da criatividade durante a codificação do competidor e se o seu código criado atendia às condições exigidas.

4.4. Questão 4: O fantástico sapo mágico

A questão o *fantástico sapo mágico* foi apresentada aos alunos como sendo um sapo inteligente, capaz de realizar algumas operações básicas matemáticas: soma, subtração, divisão, multiplicação e raiz quadrada. A cada competidor foi fornecido o código que realizava quatro dessas operações (Figura 4). A questão exigia que o participante fosse capaz de depurar o código, descobrir que operação não estava implementada e fornecer a implementação faltante.

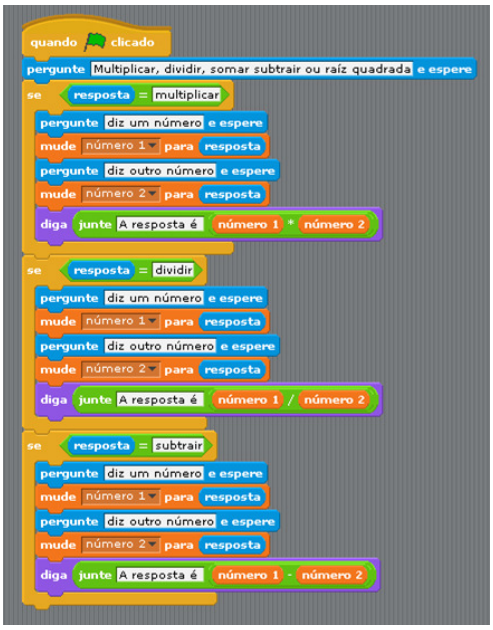

Figura 4. Código da calculadora utilizada na questão

Para que a questão fosse realizada com sucesso era necessário que o aluno tivesse um bom conhecimento dos comandos de controle, sensores, operadores e variáveis e o mais importante: que os alunos soubessem ler e entender cada parte do código. Então o aluno só completava a questão se conseguisse fosse capaz de identificar que a operação de soma estava faltando e que a completasse corretamente para que o código executasse corretamente (Figura 5).

Figura 5. Implementação da operação de soma

ISSN: 2316-6541

A atividade foi classificada com nível de dificuldade avançado. Em relação ao julgamento de desempenho, este foi analisado ao se verificar se os alunos identificaram, compreenderam e resolveram o erro inserido no desafio.

4.5. Questão 5: Seja Criativo

Este quesito tinha como ideia principal a liberdade de criação. Com base nesse princípio esta questão era a oportunidade dos alunos demonstrarem o quanto conseguiram aprender durante o curso. Tendo como regra fundamental utilizar sua criatividade, era exigido do aluno que o mesmo reproduzisse alguma lembrança, desejo ou que imaginasse alguma história e reproduzisse com a maior quantidade de detalhes possível através de técnicas e comandos aprendidos no curso.

Esta questão foi classificada com nível de dificuldade avançado. Por ser uma questão muito aberta, onde o aluno poderia seguir qualquer caminho, foi necessário estabelecer alguns critérios para sua avaliação. Buscou-se avaliar os competidores segundo os seguintes critérios: controle dos elementos de repetição e de controle, destreza com os movimentos, mudança de trajes, sincronia nos diálogos, uso correto das cores, associação da história com o plano de fundo, criação de objetos e/ou personagens, utilização dos *sprites* e originalidade.

4.6. Desempenho dos participantes na OIPS

Como foi mencionado, além de servir como um elemento motivacional, a competição objetivou avaliar se a metodologia utilizada havia sido homogênea para todas as turmas do curso. A análise do desempenho dos participantes para os quatro primeiros desafios revelou os dados que estão apresentados na Tabela 1.

Pode-se observar que praticamente todos os alunos conseguiram atingir os mesmos resultados para todas as questões, o que validou a metodologia de trabalho utilizada, no que se refere ao seu caráter pedagógico. Ainda de acordo com a tabela pode-se observar que cerca de 90% dos competidores conseguiram concluir completamente quatro dos cinco desafios propostos, enquanto o restante concluiu parcialmente alguns os desafios. Nenhum competidor deixou alguma questão sem implementar, o que mostra que foi possível contribuir para o desenvolvimento de boas práticas e técnicas de programação, como é o caso da leitura e interpretação de código alheio, de depuração e correção de erros, além da implementação de algoritmos simples.

Tabela 1. Desempenho dos participantes nas questões 1 a 4

Desafio	Concluído	Parcialmente Concluído	Não concluído
Desafio 1: Ordenação de Instrução	88%	12%	0%
Desafio 2: Reproduza a animação	100%	0%	0%
Desafio 3: Jogo de Toca	75%	25%	0%
Desafio 4: Fantástico Sapo Mágico	88%	12%	0%

ISSN: 2316-6541

Como a avaliação da Questão 5 **- Seja Criativo -** utilizou vários critérios para a sua correção, a Tabela 2 ilustra o desempenho dos participantes.

Tabela 2. Desempenho dos participantes na questão Seja criativo

Critérios	Recurso (Não Utilizou) (%)	Recurso (Básica) (%)	Recurso (Intermediária) (%)	Recurso (Avançada) (%)
Elementos de Repetição	0%	25%	50%	25%
Domínio Condicional	24%	38%	0%	38%
Mudança de Trajes	12%	25%	38%	25%
Destrezas com os movimentos	0%	25%	63%	12%
Controle no Diálogo	0%	50%	38%	12%
Cores	12%	25%	63%	0%
Associação da história com o plano de fundo utilizado	12%	38%	50%	0%
Originalidade ao criar e utilizar objetos	12%	50%	38%	0%
Utilização de Sprites	0%	63%	25%	12%
Criatividade	12%	50%	13%	25%
MÉDIA	**8,4%**	**38,9%**	**37,8%**	**14,9%**

É válido mencionar que os participantes da olimpíada relataram, através de questionário, que antes de iniciar o curso a visão que possuíam sobre a Computação era extremamente distorcida e que eles se sentem motivados a continuar aprendendo outras linguagens de programação. O questionário também conseguiu refletir que o conhecimento técnico que os alunos adquiriram sobre as estruturas mais comuns que constituem a maior parte dos algoritmos (como é o caso de variáveis, funções, entrada e saída, laços e controle) foi suficiente para que eles pudessem aprender uma nova linguagem de programação, como é o caso de Python, Pascal ou C.

5. Estado atual do trabalho e considerações finais

Percebe-se que os alunos estão se apropriando das informações necessárias para entender o que acontece em um curso superior na área de Computação. Grande parte

ISSN: 2316-6541

dos alunos têm se interessado em aprender a programar, até porque é uma novidade no colégio que estimula curiosidade de muitos. Percebe-se também a aptidão de alguns alunos, que tem mostrado grande potencial como programadores. Provavelmente, sem as informações que receberam, dificilmente ponderariam a escolha de um curso superior na área em função da desinformação sobre as possibilidades profissionais existentes.

Por outro lado, há o caso dos alunos que têm percebido que as atividades existentes na Computação são diferentes do que imaginavam. Esses também estão sendo estimulados a conhecer outras áreas para que a escolha da sua profissão aconteça de maneira mais consciente. Este projeto representa uma mudança na cultura da escola e o fortalecimento da formação dos estudantes da Licenciatura em Computação, que percebem este projeto como um espaço fértil e promissor para a geração de novos conhecimentos.

Tecnicamente, o conhecimento adquirido pelos estudantes lhes permitiu entender as diversas variáveis que influenciam a construção de algoritmos e os modelos de trabalho, que muitas vezes exigem que o programador trabalhe em códigos construídos por outras pessoas, que contém erros, que precisam ser corrigidos. Assim, pôde-se tratar de forma, superficial, mas bastante eficiente os meandros que existem no desenvolvimento de um produto de software. Procurou-se explicar aos estudantes que diferentes linguagens de programação existiam para diferentes propósitos. Atualmente, o curso intermediário de programação está sendo preparado para introduzir a linguagem Python e também noções do paradigma orientado a Objetos.

Referências

Elkner, J. (2001) "Using Python in a High School Computer Science Program". In: 9ª International Python Conference, Long Beach, California U.S.

Pausch, Randy and Kelleher, Caintlin (2005). “Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers”. In: ACM Computing Surveys, Vol. 37, No. 2, pp. 83–137.

Lima Ribeiro et. al. (2011). “LEW: Laboratório de Engenharia Web para ensino, pesquisa e extensão”. In: II Encontro Nacional de Informática e Educação, Campus Cascavel - PR.

Marques, Diego et. al (2011) “Atraindo Alunos do Ensino Médio para a Computação: Uma Experiência Prática de Introdução à Programação utilizando Jogos e Python”. In: XXII SBIE - XVII WIE, Aracaju - SE.

Rapkiewicz, C. E. et al (2006). "Estratégicas pedagógicas no ensino de algoritmos e programação associadas ao uso de jogos educacionais". In: CINTED - Centro Interdisciplinar de Novas Tecnologias na Educação, Rio Grande do Sul.

Scaico, P. D., Corlett, E. F., Paiva, L. F., Raposo, E. H., Alencar, Y (2012). Relato da Utilização de uma Metodologia de Trabalho para o Ensino de Ciência da Computação no Ensino Médio. In: Congresso Brasileiro de Informática na Educação – 18º Workshop de Informática na Escola, Rio de Janeiro.

Sudol L. A. (2011) “Deepening Students’ Understanding of Algorithms: Effects of Problem Context and Feedback Regarding Algorithmic Abstraction,” Carnegie Mellon University.

Trajano, F. H. D, Lins, F. A. A, Melo, J. C B de (2010). “A Informática a serviço do aprendizado”. In: X Jornada de ensino, pesquisa e extensão –UFRPE, Recife/PE.

Wilson C., Sudol L. A, Stephenson C, Stehlik M., Running on Empty: The failure to teach K-12 computer science in the digital age, Association for Computing Machinery, 2010.

ISSN: 2316-6541