Planejando um *serious game* para a prática de Programação

Cárlisson Borges Tenório Galdino[1], Sebastião Rogério da Silva Neto[1], Evandro de Barros Costa[1]

[1]Instituto de Computação – Universidade Federal de Alagoas (UFAL)
Caixa Postal 57072-900 – Maceió–AL – Brasil

carlisson@nti.ufal.br, srsn@ic.ufal.br, ebcosta@gmail.com

Abstract. *This article presents some decisions made when planning a serious game to practice of programming. The "KidCoder" is a proposal for a serious game as a practice environment, to motivate students, presenting challenges with gradual difficulty, wrapped in an immersive storyline. In form of a bi-dimensional JRPG, it uses visual programming as game mechanic, used by player to solve battles and puzzles. It is expected to the article can help both to understand the project under development and to assist in decisions of anyone that will eventually develop similar project.*

Resumo. *Este artigo apresenta algumas das decisões tomadas durante o planejamento de um serious game para a prática de Programação. O "KidCoder" é uma proposta de serious game na forma de um ambiente de prática para motivar alunos, apresentando desafios com dificuldade gradual, envoltos em um enredo imersivo. Na forma de um JRPG em duas dimensões, utiliza-se programação visual como mecânica de jogo, a ser utilizada pelo jogador para resolver combates e quebra-cabeças. Espera-se, com o artigo, ajudar tanto a compreender o projeto ora em desenvolvimento quanto auxiliar nas decisões de quem eventualmente venha a desenvolver projeto semelhante.*

1. Introdução

Cursos de Ciência da Computação e afins tem apresentado um índice de aprovação abaixo do desejado em disciplinas que constituem seus pilares: Algoritmo e Programação. São vários os motivos que levam a isso. Alguns dos fatores, segundo [Silva 2009], incluem a dificuldade dos alunos de lidar com conceitos abstratos, o hábito de decorar conteúdos de disciplinas, trazido da vida escolar anterior, desânimo e falta de motivação. Desta forma, os alunos terminam não dedicando atenção suficiente à prática, diante deste tipo de disciplina.

Esta dificuldade, se não tratada, tende a levar a dois efeitos nocivos aos cursos de tecnologia: a evasão do curso e estudantes que enfrentarão dificuldades sempre que se depararem com qualquer disciplina do curso que envolva Programação.

Além do conhecimento teórico, é fundamental que o estudante pratique, de modo a dominar melhor o ato de programar, complementando o conhecimento teórico

DOI: 10.5753/cbie.wcbie.2015.1164

obtido em sala de aula com um entendimento do processo de criação de algoritmos e solução de problemas, internalizado através da prática.

Para estimular a prática de programação desses alunos, há várias possibilidades, desde a cobrança de trabalhos e exercícios até o uso de softwares mais específicos. Dentre estes, os que tentam estimular o estudante através da diversão.

KidCoder é um jogo que está sendo desenvolvido de modo a oferecer ao aluno um ambiente de prática divertido e envolvente. Este artigo apresenta o projeto, através das decisões que foram tomadas da sua concepção à implementação. Pretende-se que seja útil para quem tenha curiosidade a respeito do KidCoder ou mesmo sirva de auxílio a quem pretende desenvolver um serious game[1].

De acordo com os aspectos acima delineados, este trabalho se organiza da seguinte forma: a Seção 2 apresenta trabalhos relacionados, com ideia semelhante à do presente trabalho; a Seção 3 apresenta a proposta KidCoder e seus conceitos; na Seção 4 é apresentado o Planejamento e Desenvolvimento da proposta; por fim, na Seção 5, são apresentadas as considerações finais.

2. Trabalhos Relacionados

Dentre os trabalhos existentes para finalidade similar, temos como gênero mais popular o combate de robôs em uma arena. Neste gênero (que chamaremos aqui de RWL - RobotWar[Sch] Like, em homenagem ao jogo lhe deu origem em 1970, apresentado na figura 1), cada jogador deve programar o comportamento de seu robô digital. Após isso, os robôs são executados em uma arena virtual para duelarem. O gênero se popularizou com o Robocode[Lar13], criado por Mathew Nelson em 2001, tendo hoje diversos jogos para as mais variadas linguagens e plataformas.

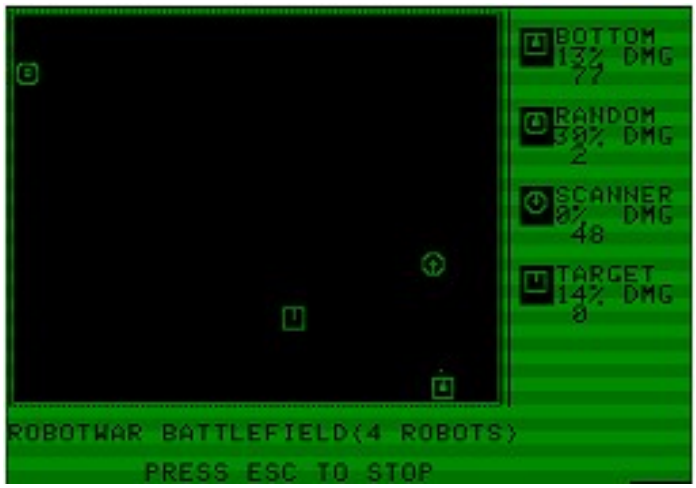

Figura 1. RobotWar, precursor do gênero de arena de robôs virtuais

1 *Serious Game*: jogo criado tendo como propósito primário outro que não o entretenimento.

Além dos RWL, outros gêneros de jogos são eventualmente utilizados, geralmente ligados à solução de puzzles, como é o caso do experimento relatado por [JESUS, 2010].

O software Logo (mostrado na figura 2) permite o desenho através da movimentação de uma tartaruga pela tela, sendo uma ferramenta utilizada em estágio inicial do aprendizado de programação, especialmente para crianças, dada a simplicidade de seu conceito.

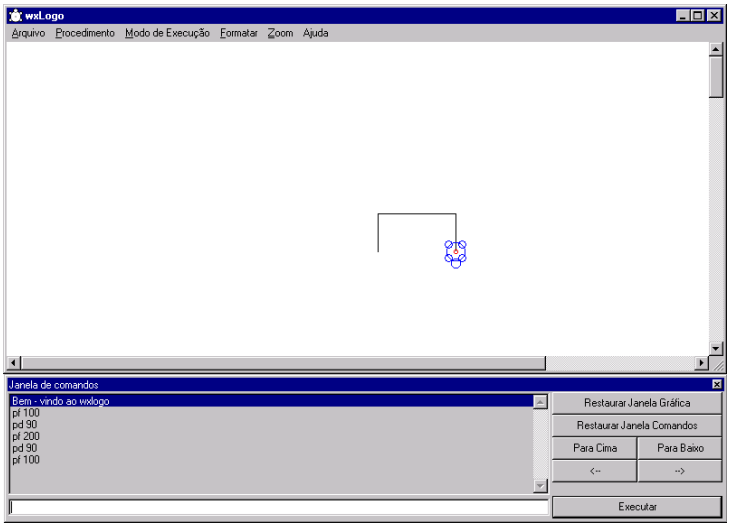

Figura 2. Tela incial SuperLogo

O artigo [SILVA, R. M. 2008] expõe a utilização da linguagem de programação Logo na educação de crianças, bem como as vantagens deste uso para formação do seu intelecto. Torna-se evidente que a linguagem de programação Logo é ótima para o desenvolvimento das habilidades intelectuais da criança, que ao interagir com o computador através da "tartaruga" constrói as estruturas mentais de forma privilegiada, fazendo com que o raciocínio lógico se desenvolva, favorecendo a aprendizagem das matérias curriculares principalmente da matemática.

O Kodu Game Lab foi desenvolvido pelo laboratório de pesquisas FUSE (Future Social Experiences) Labs, mantido pela Microsoft. Segundo [SOUZA et al. 2013] este software é utilizado para criar jogos, em num ambiente tridimensional e multimídia cujo principal objetivo é estimular em sala de aula a aprendizagem lúdica da programação de computadores moldada nos paradigmas da orientação à objetos

Esses trabalhos apresentam proposta semelhantes ao KidCoder, na próxima seção é apresentado o diferencial da proposta do presente trabalho.

3. KidCoder e Conceitos

A proposta KidCoder nasceu como trabalho de mestrado e vem sendo desenvolvida. Embora a implementação atual seja pouco abrangente, parte das possibilidades de expansão já está prevista.

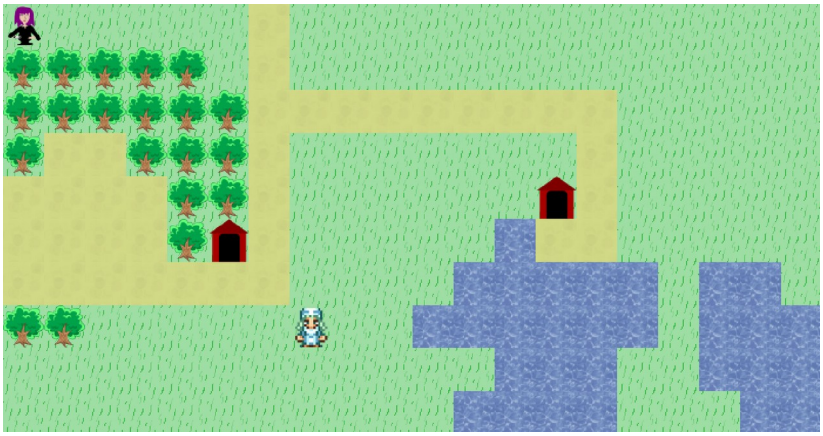

Figura 3. Protótipo atual do KidCoder

Em sua essência, KidCoder é um RPG[2] que utiliza montagem de algoritmos como forma de resolução de conflitos e problemas. Tanto os combates quanto os puzzles do jogo são resolvidos com o jogador montando algoritmos. Alheio a esta diferença crucial (que é o que torna KidCoder um serious game, de utilidade para quem quer desenvolver suas habilidades como programador), o jogo apresenta uma história envolvente, com um objetivo maior a ser cumprido pelo protagonista.

Está sendo criado visando, em primeiro momento, auxiliar estudantes de cursos técnicos e superiores que precisem aprender sobre Programação. Pretende-se, futuramente, que seja útil a outros públicos, mesmo que totalmente leigos na matéria. Este objetivo pendente, em particular, precisa que o jogo ofereça um bom modo tutorial para que possa ser alcançado.

Os desafios implementados no KidCoder serão mapeados com tópicos comuns à cadeira de Programação ministrada em cursos de nível superior. Tal medida terá pouca consequência prática inicial, mas abrirá caminho para bons usos futuros como, por exemplo, a possibilidade de o jogo perceber quais os tópicos nos quais o jogador demonstra mais dificuldades para recomendar seu estudo. A progressão do nível de dificuldade virá tanto pelo acréscimo de novos elementos no decorrer do jogo como da necessidade de proficiência em tópicos mais avançados - ou da combinação de tópicos - para solucionar os quebra-cabeças e combates do jogo.

4. Planejamento e Desenvolvimento

Desde o planejamento, decisões tiveram de ser tomadas para que o projeto prosseguisse. Aqui trataremos das principais delas.

4.1 Como tratar a diversão

Há três formas de se utilizar diversão para auxiliar o ensino ou aprendizado, através de um software: gamification, serious game e interação jogável. [XU 2011]

2 *Role Playing* videogame: gênero de jogo onde se controla as ações de um personagem principal imerso em um mundo bem-definido.

Na primeira abordagem, tem-se um software de finalidade técnica, mas que faz uso de recursos de game design para melhorar a experiência do usuário ou se tornar mais atrativo. O Waze[3] é um exemplo de gamification, considerando que se trata de uma ferramenta de geolocalização e definição de rotas, mas que acrescenta elementos como pontos e recompensas para seus usuários.

Um serious game segue o caminho oposto: trata-se de um jogo primeiramente, mas que apresenta um objetivo secundário, além da diversão. Os conceitos de gamification e serious game podem se confundir em um primeiro momento. Uma boa forma de diferenciá-los é com uma simples pergunta: "se retirarmos todos os elementos de game design deste programa, o que resta?". Caso, na situação hipotética, o programa ainda tivesse utilidade, estamos tratando de gamification (o Waze sem elementos de game design continuaria sendo uma ferrametna de geolocalização e definição de rotas), caso contrário, lidamos com um serious game.

Interação jogável se trata da elaboração de produtos e serviços interativos, frequentemente objetivando uma mudança de comportamento.

As duas primeiras abordagens são promissoras para a demanda. Inicialmente pensou-se em um editor de códigos gameficado, utilizando testes de unidade para dar feedback instantâneo. A concepção do KidCoder veio logo depois, mostrando-se um conceito suficientemente interessante para a mudança de foco. Convém lembrar que, para a criação de um jogo, há que se levar em conta os elementos apresentados, se tem potencial de resultar em um jogo que divirta e, desta forma, atinja os objetivos pretendidos. Em uma análise inicial, KidCoder aparentou preencher este requisito.

4.2 Competição ou Cooperação

Pode-se classificar como jogos competitivos aqueles cujo objetivo do jogador é superar outro jogador (ou outros jogadores). Jogos não competitivos podem ser divididos em dois grupos: jogos cooperativos e jogos individuais. Embora, a princípio, haja uma distinção entre os três tipos, na prática pode-se misturá-los. Ao criarmos uma disputa de times, por exemplo, misturamos a cooperação e competição. Até mesmo um jogo individual pode despertar o espírito competitivo se aplicada pontuação e um ranking.

Competições têm o efeito de motivar, induzindo seus participantes a tentarem superar os demais. Tal efeito é perceptível ao se pensar em esportes, quando até mesmo a simples torcida por um time é capaz de empolgar. Acontece, porém, que jogos competitivos trazem dois efeitos colaterais:

- Primeiro, que não é capaz de empolgar a todos. À parte dos que se empolgam e motivam com a competição, há os que evitam confrontos e preferem situações cooperativas ou individuais, onde não precise haver perdedores.

3 Um aplicativo de trânsito e navegação baseado em uma comunidade https://www.waze.com/pt-BR/

- O segundo efeito é que o espírito competitivo, quanto mais forte se apresente, mais tende a elevar o nível da disputa. Logo, isso se torna um obstáculo para os que não acompanhem o ritmo de preparações e treinamentos. Tomemos como exemplo o Xadrez, que apresenta regras simples de movimentação de peças. Para se tornar competitivo em Xadrez, porém, é preciso um vasto estudo e prática para se desenvolver estratégias de ação que levem à vitória ou à frustração dos planos do oponente. O lado positivo é que aqueles que são afeitos à competição tendem a buscar esse conhecimento adicional, evoluindo em entendimento das regras e sutilezas do jogo. Outras áreas onde se apliquem jogos tem questionado a aplicação de jogos competitivos.

Traduzindo para o âmbito do ensino e aprendizado de Programação, competições de Programação tendem a ser altamente benéficas para parcela da turma, estimulando seu desenvolvimento. Há a tendência, porém, de que um grupo se torne cada vez mais alheio às disputas, sem envolvimento.

Jogos cooperativos trazem como vantagem a preparação para trabalho em equipe, qualidade desejável em profissionais. Por outro lado, alguns alunos podem tentar evitar a participação prática, buscando apenas crédito por fazer parte do grupo. A experiência individual elimina esses pontos, mas a falta de participação social impõe uma outra dificuldade: é mais difícil despertar interesse e motivação nos alunos do que nos casos anteriores. O sucesso de um jogo individual depende fortemente de quanto o mesmo se tornou atrativo, pelo enredo, mecânica e demais elementos de game design.

Optou-se pela criação de um jogo que oferecesse principalmente uma experiência individual devido à percepção de que havia, como já citado, um bom potencial nos princípios planejados para esse jogo. Ademais, jogos individuais de auxílio ao aprendizado de Programação não são ainda tão frequentes e populares quanto os competitivos. Práticas competitivas em ambientes lúdicos já estão contempladas por jogos RWL.

4.3. Escolha do ambiente de execução

As principais opções de ambiente de execução do jogo seriam desktop e mobile. Uma opção não óbvia e com mais obstáculos para seu uso seriam os consoles de videogame. O ambiente web, por outro lado, é um ambiente capaz de ser utilizado como meio para atingir qualquer dos ambientes citados, a depender da forma como o jogo seja implementado.

Foi pensado o ambiente web para uso em desktops como a principal forma de execução do jogo. Tal escolha não representa a adoção, desde o início, de recursos cliente-servidor: tais recursos podem vir a ser implementados para aprimorar o projeto futuramente.

Outras decisões tomadas (como, por exemplo, o uso de programação visual) facilitarão que seja possível utilizar o jogo em dispositivos com tela sensível ao toque.

4.4. Linguagem de implementação

Devido às qualidades de Python[4] como uma linguagem inicial de aprendizado de Programação (como apresentado por [ELKNER 2012]), Python foi cogitada inicialmente para ser utilizada na implementação do KidCoder. Mesmo com a escolha do ambiente web, ainda seria possível o uso de Python, graças a projetos como Brython[5] e Skulpt[6]. A baixa adoção de soluções assim e a consequente dificuldade em encontrar documentação e apoio foram algumas das razões para que adotássemos outra solução.

Para lidar com HTML5[7], tradicionalmente se codifica algoritmos em JavaScript. Há, porém, diversas outras opções de uso de linguagens alternativas ao codificarmos em HTML5. Desde interpretadores Python e Ruby a linguagens novas, oferecendo funcionalidades que JavaScript não tem, como tipagem estática e maior segurança.

Foi adotada a linguagem CoffeScript[8], que oferece uma sintaxe mais enxuta ao desenvolvimento HTML5, incluindo suporte a classes e encapsulamento de atributos de objetos. O compilador CoffeScript, então, traduz o código escrito para JavaScript.

4.5. Linguagem de programação do jogo

O diferencial de KidCoder, enquanto jogo de RPG, é justamente a necessidade de que o jogador codifique enquanto joga, é a programação como elemento de jogabilidade. Para receber esses comandos do jogador, as possíveis abordagens encontradas foram três: o uso de uma linguagem de programação real, o uso de uma linguagem de programação própria do jogo e o uso de programação visual.

Assim como aconteceu com a Linguagem de Implementação, inicialmente foi escolhida a linguagem Python, devido às facilidades já citadas. A dificuldade em oferecer suporte completo a uma linguagem de programação de uso geral terminou guiando o desenvolvimento para outra solução: uma linguagem própria do jogo, mas que implementaria um subconjunto de Python.

Esta solução foi substituída posteriormente pela adoção de Programação Visual, na qual o usuário arrasta blocos correspondentes a comandos e partes de comandos, encaixando-os para criar, assim, o algoritmo. A figura 4 mostra um exemplo de algoritmo simples composto desta forma.

4 Linguagem de programação de alto nível, para propósitos gerais https://www.python.org/

5 Uma implementação de Python para a programação web do lado do cliente. http://brython.info/

6 Uma implementação de Python que roda inteiramente no navegador http://www.skulpt.org/

7 http://www.w3.org/TR/html5/

8 http://coffeescript.org/

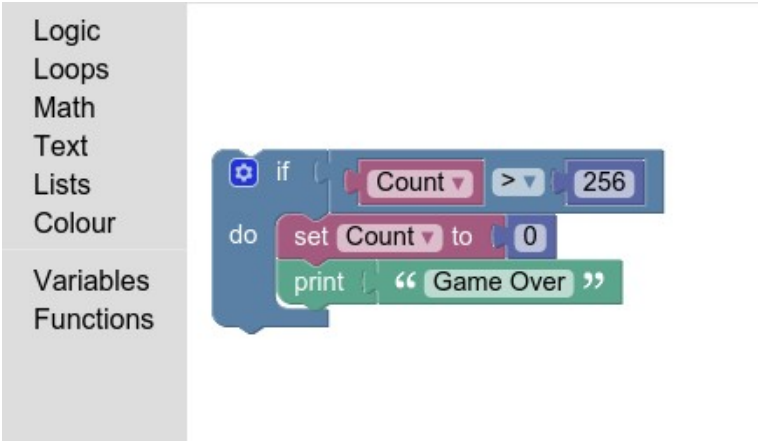

Figura 4. Exemplo de Programação Visual, com a biblioteca Blockly

As vantagens que motivaram a mudança são:

1. Cada desafio apresentará uma API própria para aquela situação. A apresentação das funções como blocos torna o jogo mais interessante visualmente, inclusive torna mais clara essa mudança de API entre os desafios;
2. A criação do algoritmo através do ato de arrastar e unir blocos facilita que o jogo seja adaptado para funcionar também em dispositivos móveis, onde o uso de teclado virtual se apresenta menos prático do que o uso da tela sensível ao toque;
3. O jogador não incorrerá em erros de sintaxe, como falta de indentação ou grafia errada de comandos, podendo focar o algoritmo em si.

Apesar destas vantagens, há que se considerar que há também desvantagens, sendo a principal identificada a oportunidade que se perde de criar uma familiaridade com os rigores de uma linguagem formal.

A adoção de Programação Visual, por outro lado, tende a ser mais atrativa para os jogadores. Os trabalhos de [SANTOS, 2013] e [RIBEIRO, 2012] mostra como seu uso pode ser eficaz na aprendizagem.

4.6. Estratégia de Implementação

Jogos em HTML5 são bastante comuns hoje em dia. Até mesmo jogos mobile e alguns de console, executando aparentemente fora do navegador web, são na verdade criados como aplicativos HTML5. Para este tipo de desenvolvimento, há 4 formas principais (que podem, eventualmente, ser misturadas em uma solução mais complexa). Todas elas utilizam JavaScript para configurar reação e movimentação de elementos, controle de som, efeitos e captura dos comandos do jogador. O que diferencia é a representação visual do jogo e, consequentemente, mudam também os comandos e a forma de lidar com ele através de JavaScript.

Um jogo criado utilizando elementos DOM[9] lida basicamente com elementos simples do HTML posicionados e estilizados via folhas de estilo e/ou comandos JavaScript. Tanto se pode utilizar o próprio HTML para criar previamente os elementos de jogo quanto eles podem ser criados posteriormente, através de JavaScript. A vantagem de uso do DOM é que há ampla documentação sobre como manipular elementos, seja via CSS[10] ou JavaScript, apesar de não ser uma solução frequentemente adotada particularmente para jogos. Pode-se utilizar bibliotecas de uso geral como Jquery[11] amplamente.

A solução mais utilizada para a criação de jogos em HTML5 é com uso de Canvas, elemento de página que cria uma área de desenho, desenho este que pode ser criado via comandos em JavaScript. Inexistente antes da versão 5 do HTML, o Canvas permite que se crie jogos de maneira parecida com o método tradicional de outras linguagens, pelo ciclo resumido, entre outros, por [DJAOUTI, 2008]: verifica se o usuário efetuou algum comando; processa (com ou sem comando); reescreve a tela. A popularidade da abordagem se comprova facilmente pela quantidade de bibliotecas e frameworks para desenvolvimento de jogos em HTML5 que recorrem a ela.

WebGL[12], anteriormente conhecido como Canvas:3D, utiliza um elemento Canvas para criar ambientes tridimensionais, sem necessidade de instalação de plugins, no próprio navegador. Dada a busca por visuais detalhados e fotorrealistas, o WebGL tende a ter cada vez mais adoção.

Uma vez que SVG[13] é um formato baseado em XML, sua manipulação através de JavaScript é perfeitamente viável. A vantagem do formato é que os elementos que compoem os desenhos são representados internamente como formas geométricas. Apesar do potencial para a criação de jogos, há pouca documentação sobre o uso desta tecnologia, o que dificulta sua adoção para um projeto.

Para o KidCoder, adotamos o uso de DOM devido às vantagens citadas e à pretensão inicial de permitirmos a codificação em Python pelo jogador. Para um jogo sem muitas animações e sem necessidade de renderização 3D, o uso de DOM se mostrou suficiente.

5. Considerações Finais e Trabalhos Futuros

KidCoder continua sendo desenvolvido, rumo ainda à sua implementação inicial. Espera-se que se torne uma ferramenta útil no combate à desmotivação que leva alunos de Programação a abandonarem disciplinas e, algumas vezes, o curso.

9 Document Object Model - convensão de linguagens para representar e interagir com objetos em HTML e XML.

10 Cascading Style Sheets

11 http://jquery.com/

12 https://www.khronos.org/webgl/

13 Scalable Vector Graphic - formato de desenho vetorial, representado em XML

Não previstas de início, algumas características podem ser implementadas no futuro:

- Modo tutorial: um conjunto de desafios que guie o jogador, ensinando como começar a programar, pode tornar KidCoder uma ferramenta adequada também para despertar o interesse por Programação em alunos de ensino médio ou de outras áreas de conhecimento.
- Interatividade com outros jogadores: uma vez que seja implementado um servidor do jogo, KidCoder poderá ser utilizado como ambiente para combate entre robôs programados. Desta forma, o jogo poderá ser utilizado também como uma ferramenta para competições.
- Análise de conhecimento: uma vez que os desafios apresentados ao jogador envolverão conhecimentos distintos, será possível avaliar qualitativamente o desempenho do jogador, apresentando sugestões de assuntos a estudar para que desenvolva melhor suas habilidades, caso tenha gostado de programar.
- Versão para tablets: com alguns ajustes, será possível jogar KidCoder também em tablets.

Referências

ARAUJO, B. O. (2010) "Jogos Cooperativos X Jogos Competitivos: Uma Análise Perceptiva Qualitativa".

DJAOUT,I D. and Alvarez J. and Jessel J. P. and Methel G. and Molinier P. (2008) "A Gameplay Definition through Videogame Classification"

ELKNER, J. and Downey A. B. and Meyers C. (2012) "How to think like a computer scientist".

RIBEIRO, R. S. and Brandão L. O. and Brandão A. A. (2012) "Uma visão do cenário Nacional do Ensino de Algoritmos e Programação: uma proposta baseada no Paradigma de Programação Visual"

SANTOS, R. M. S. (2013) "Ensino da Programação através da Programação Visual"

SILVA, I. F.A. and Silva, I. M. M. and Santos, M. S. (2009) "Análise de Problemas e Soluções Aplicadas ao Ensino de Disciplinas Introdutórias de Programação".

XU, Y. (2011) "Literature Review on Web Application Gamification and Analytics". In: CSDL Technical Report 11-05.

JESUS, Elieser A.; RAABE, André LA. Avaliação Empírica da Utilização de um Jogo para Auxiliar a Aprendizagem de Programação. In:**Anais do Simpósio Brasileiro de Informática na Educação**. 2010.

SILVA, Ronaldo Machado. O uso da linguagem Logo na educação infantil. Disponível em Acesso em: 23 Mar. 2014

SOUZA, Paulo Roberto de Azevedo; DIAS, Lucimeri Ricas. Kodu Game Labs: Apresentação e reflexão sobre os jogos criados e publicados na comunidade Kodu BR. In: Simpósio Brasileiro de Jogos e Entreterimento Digital. 2013.

ELKNER, Jeffrey, Allen B. Downey, and Chris Meyers. How to think like a computer scientist. April 2012.