Um estudo exploratório dos games para introdução ao pensamento computacional

An exploratory study of games for introduction to computational thinking

Sérgio Souza Costa [1], Spartacus Silva Souza[2], Leonardo C. C. Mendes[3], Rosane de F. A. Obregon [4], Luzia Emanuelle R. V. da Silva[5]
Universidade Federal do Maranhão, MA

Evaldinolia Gilbertoni Moreira[6], Jeane Silva Ferreira[7]
Instituto Federal do Maranhão, MA

Resumo

Nos últimos anos, tem crescido o interesse na utilização de games e seus mecanismos para o ensino e aprendizado, principalmente do pensamento computacional. Entretanto, apesar da existência de um volume expressivo de games constata-se uma lacuna quanto as pesquisas que os analisem sistematicamente. Esse tipo de pesquisa é relevante, pois pode apoiar a escolha de um determinado game por educadores e alunos, além de identificar possíveis limitações nos trabalhos recentes da literatura. Assim, o presente artigo objetiva identificar e analisar os games empregados para o ensino e aprendizado do pensamento computacional. O estudo levará em consideração algumas dimensões, como o gênero, habilidades computacionais exploradas e a linguagem utilizada.

Palavras-chave: jogos, programação, aprendizagem.

Abstract

In recent years, there has been growing interest in using games and the mechanisms for teaching and learning, especially of computational thinking. However, despite the existence of a significant volume of games notes is a gap as the research to analyze the systematically. This type of research is relevant because it can support the choice of a particular game by educators and students, and identify possible limitations of the games in the literature. Thus, this article aims to

[1] sergio.costa@ufma.br
[2] spartacus.s.souza@gmail.com
[3] leomds94@gmail.com
[4] rosane.obregon@ufma.br
[5] luziaemanuellee@hotmail.com
[6] evaldinolia@ifma.edu.br
[7] jeane@ifma.edu.br

identify and analyze the games used for teaching and learning of computational thinking. The study will take into consideration some dimensions, such as gender, computer skills and language.

Key words: games, programming, learning.

1. Introdução

Em 2013, o prêmio Nobel de Química foi concedido a três pesquisadores (Karplus, Levitt e Warshel) que lançaram as bases da modelagem computacional de reações químicas de alta complexidade (JORGENSEN, 2013; THIEL; HUMMER, 2013). Diversos outros cientistas têm utilizados modelos e métodos computacionais nas mais distintas áreas do conhecimento. Conway (1970) modelou o movimento de formas simples no espaço (semelhante a organismos vivos) a partir de um autômato celular denominado jogo da vida. Schelling (1971) usou também autômato celular para modelar e simular a segregação a partir de um pequeno conjunto de regras. Estes trabalhos inspiraram diversos cientistas sociais a utilizarem computadores para simular fenômenos sociais (GILBERT, 2008). Além das ciências sociais, geógrafos tem utilizado métodos e ferramentas da ciência da computação para compreender e modelar o espaço geográfico, dando origem a geoinformática (GOODCHILD; YUAN; COVA, 2007). Outra ciência de fronteira, a bioinformática, originou-se a partir da utilização dos computadores na biologia. Além desses, físicos, matemáticos e engenheiros utilizam frequentemente ferramentas e métodos computacionais em seus experimentos e cálculos. Fora dos centros de pesquisa, os computadores estão presentes nas casas e nos bolsos das pessoas. Computadores de mesa, *notebooks*, *tablets*, *smartphones* e *smart-tvs* são os exemplos mais comuns, porém já é possível encontrar óculos e relógios[8] integrados através de software e conectividade.

Todos estes dispositivos compartilham a mesma arquitetura que possibilitou o desenvolvimento do primeiro computador. O ENIAC (*Electronic Numerical Integrator Analyzer and Computer*) apresentou uma arquitetura inovadora que permitia que os programas (uma sequência de instruções e ou símbolos) fossem armazenados na memória do computador da mesma maneira que os dados (GOLDSTINE; GOLDSTINE, 1996). Essa característica tornou os computadores muito mais flexíveis, pois poderiam ser programados para atender a diferentes necessidades. Estas instruções são combinadas através de algoritmos, o que requer um modo específico de pensar, denominado pensamento computacional. Este pode ser definido então como os processos do pensamento envolvidos na formulação de problemas, de modo que suas soluções possam ser representadas como passos e algoritmos computacionais (AHO, 2012). Através do pensamento computacional, criam-se soluções para as ciências, a indústria e o cotidiano. Wing(2006) definiu o pensamento computacional como o conjunto de habilidades intelectuais e de raciocínio que indicam como as pessoas interagem e aprendem a pensar por meio da linguagem computacional. Wing tem inspirado diversos pesquisadores e educadores a criarem metodologias que levam o pensamento computacional para escolas de ensino básico, fundamental e médio. Contudo, esse interesse não é recente, um dos trabalhos mais antigos foi o desenvolvimento da linguagem de programação Logo que tinha como objetivo o ensino de programação para crianças (SOLOMON; PAPERT, 1976). As metodologias mais recentes estão utilizando principalmente jogos eletrônicos (conhecidos também como games) para apoiar o

[8] Estes são alguns exemplos de tecnologias vestíveis.

processo de ensino e aprendizado do pensamento computacional. Entretanto, apesar da existência de um volume expressivo de games[9] e ambientes gamificados, constata-se uma lacuna quanto a pesquisas que os analisem sistematicamente. Este tipo de pesquisa é relevante, pois pode apoiar a escolha de um determinado game por educadores e alunos, além de identificar possíveis limitações nos trabalhos recentes da literatura. Assim, o presente artigo objetiva identificar e analisar os games empregados para o ensino e aprendizado do pensamento computacional. O estudo levará em considerações algumas dimensões, como o gênero, habilidades computacionais exploradas e a linguagem utilizada.

O artigo está estruturado da seguinte maneira: a Seção 2 apresenta e discute os trabalhos relacionados com o objeto de estudo, o que inclui conceitos de games e do pensamento computacional; a Seção 3 apresenta a metodologia utilizada para comparar e analisar os jogos; a Seção 4 apresenta os resultados que sintetizam as dimensões utilizadas para comparar e analisar os jogos; por fim, a Seção 5 apresenta as conclusões do trabalho.

2. Trabalhos relacionados

Esse trabalho está inserido em um contexto bem mais amplo que é a utilização de games em processos de ensino e aprendizado. Um dos motivos para utilização dos games é o crescente interesse de crianças, jovens e adultos por este entretenimento. De acordo com uma pesquisa realizada pela *Entertainment Software Association* (Associação de Softwares para Entretenimento) a idade média das pessoas que jogam é de 31 anos, sendo que 71% deste público têm mais de 18 anos (ESA, 2014). O game é um tipo mais específico de jogo que utiliza um equipamento eletrônico, como console, computador, celular ou *tablet*. De modo geral, todos os jogos podem ser vistos como elementos da cultura que contribuem para o desenvolvimento social, cognitivo e afetivo dos sujeitos (HUIZINGA, 1971). Contudo, os games possuem algumas características que os distinguem do outros jogos. Salen e Zimmerman (2004) definem um game como um sistema em que os jogadores se envolvem em um conflito artificial, regido por regras, resultando em uma saída quantificável. Em outras palavras, são conflitos não reais onde jogadores são pontuados através de regras bem definidas. Na educação, os games podem ser entendidos como um tipo de ambiente interativo de aprendizagem[10], como proposto por Kapp (2013). Segundo o autor, a gamificação e a simulação são outros dois tipos, sendo importante distingui-los. Kapp (2013) define um game como um sistema em que os jogadores se envolvem em um desafio abstrato, definido por regras, interatividade e *feedback*, que resulta em uma saída quantificável e frequentemente provoca uma reação emocional. Por esta definição, um game ocorre em um espaço abstrato que é usualmente uma simplificação do mundo real. Essa é uma importante característica que diferencia os games dos ambientes gamificados. Este último é entendido como o uso de mecânicas, estéticas e pensamentos dos games para envolver pessoas, motivar a ação, promover a aprendizagem e resolver problemas (KAPP, 2013). Uma definição mais sucinta sobre gamificação é encontrada em (DETERDING et al., 2011). O autor definiu gamificação como a aplicação de elementos de games fora do contexto de games. Na educação alguns destes elementos já são usados há algum tempo, como a distribuição de pontuações para atividades e *feedbacks* frequentes (FARDO, 2013). Exemplos mais recentes de um ambiente gamificado usado na educação inclui Duolingo (GARCIA, 2013)

[9] Neste trabalho utilizaremos o termo games como sinônimo de jogos eletrônicos, a exemplo da literatura que tem empregado este termo.

[10] O termo original é Interactive Learning Environment (ILE)

e Khan Academy (KHAN, 2011). O terceiro tipo de ambiente interativo de aprendizado é a simulação. A simulação é um ambiente realista, de risco controlado, onde os alunos podem praticar comportamentos e experimentar os efeitos das suas decisões. Por exemplo, um futuro piloto de avião precisa ter realizado uma determinada quantidade de horas em simuladores de vôo antes de realizar o primeiro vôo em um avião real. Recentemente, simuladores têm sido utilizados inclusive em autoescolas. Uma grande vantagem é permitir simular situações adversas sem um risco real ao aluno. Simuladores podem inclusive substituir parcialmente laboratórios reais, como de física, química e eletrônica (BALAMURALITHARA; WOODS, 2009).

O objetivo deste trabalho é categorizar apenas os games, ou seja, não são incluídos os ambientes gamificados e nem as simulações. O foco de análise está circunscrito aos games que introduzem o pensamento computacional, definido como o conjunto de habilidades intelectuais e de raciocínio que indica como as pessoas interagem e aprendem a pensar por meio da linguagem computacional (WING, 2006). O pensamento computacional envolve usar métodos, linguagens e sistemas de ciências da computação com objetivo de resolver problemas de qualquer disciplina. Aho (2012) apresenta uma definição mais concisa para o pensamento computacional: processos de pensamento envolvidos na formulação de problemas, de modo que suas soluções possam ser representadas como passos de algoritmos. Por sua vez, um algoritmo pode ser definido como:

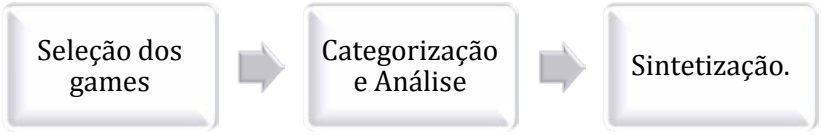
> "Um algoritmo é um método finito, escrito em um vocabulário fixo, regido por instruções precisas, que se movem em passos discretos, 1, 2, 3,..., cuja execução não requer insight, esperteza, intuição, inteligência ou clareza e lucidez, e que mais cedo ou mais tarde chega a um fim" (BERLINSKI, 2002, p.21)

Essa definição captura uma importante característica dos algoritmos: a inteligência está na construção do algoritmo e não na sua execução. Então o algoritmo é uma forma de representar e compartilhar o conhecimento que se tem sobre um determinado problema. O algoritmo pode ser executado por um computador e entendido por qualquer pessoa que compreenda essa linguagem. *The Royal Society* (2012) define o pensamento computacional como processo de reconhecer os aspectos da computação no mundo que nos rodeia, a aplicação de ferramentas e técnicas de Ciência da Computação para entender os sistemas e processos naturais e artificiais. Por estas definições, é possível perceber que o pensamento computacional não deve ficar restrito aos cientistas da computação. Essa constatação inspirou diversas iniciativas para incluir o pensamento computacional desde o ensino básico. Por exemplo, em 2010 líderes de diferentes escolas, em conjunto com a *Computer Science Teachers Association* (CSTA), *International Society for Technology in Education* (ISTE) e *National Science Foundation* (NSF) desenvolveram um conjunto de ferramentas para o ensino do pensamento computacional na educação básica (CSTA.ISTE, 2011). Muitas dessas ferramentas utilizam games para motivar e envolver os alunos. Outra iniciativa importante inclui nomes influentes do mundo tecnológico, como Bill Gates, fundador da Microsoft® e Mark Zuckeberg fundador do Facebook®. Em um dos vídeos, Bill Gates diz que aprender a escrever programas estende sua mente e ajuda a pensar melhor, cria uma maneira de pensar sobre as coisas que é útil em todos os domínios. Além do mundo tecnológico, essa iniciativa conseguiu o apoio de pessoas influentes também na política como Barack Obama, presidente dos Estados Unidos. Essa iniciativa denominada *code.org* já visa estimular o ensino do pensamento computacional desde o ensino básico. No site é possível encontrar diversos games usados para introduzir o pensamento computacional. Esse movimento é atualmente mais forte nos Estados Unidos, porém existem iniciativas

similares em diversas partes do mundo, como o *Code Club*, que é uma rede mundial de atividades extracurriculares gratuitas, completamente gerenciada por voluntários, com o objetivo de ensinar programação de computadores às crianças. Similar ao *code.org*, eles disponibilizam material e metodologias de apoio ao ensino de programação usando games. Vale ainda citar alguns artigos acadêmicos como Barcelos e Silveira (2012); França (et al., 2012); Lauyse (et al., 2014). Nesses trabalhos os autores exploram o uso de games no ensino de computação no ensino fundamental e médio no Brasil, estimulando a criação de diversos games com propostas similares. O objetivo central de tais iniciativas é despertar a motivação das crianças para o aprendizado do pensamento computacional, através do desenvolvimento ou utilização dos games. Nesse sentido, foi possível sintetizar esses–trabalhos de acordo com a metodologia apresentada na Seção 3.

3. Metodologia

Para alcançar o objetivo proposto neste estudo foram adotadas três etapas:



A seleção foi utilizada para delimitar o escopo do trabalho devido a grande variedade de games. O foco deste trabalho compreende apenas os games definidos por (KAPP, 2013). Não foram incluídos gamificação e simulação que são outros exemplos de ambientes interativos de aprendizagem. Para o ensino do pensamento computacional, existem diversos ambientes gamificados, dentre os quais, destacam-se: Khan Academy, CodeAcademy, TreeHouse e Code Avengers e CodePuPil. Além destes, existem alguns ambientes que tem como foco principal as competições entre programadores. Mesmo não sendo classificados como ambientes de aprendizado gamificado, eles possuem algumas características de jogos, como pontuações, placares e evolução. Destacam-se também: Project Euler, Spoj, Hacker Hank, CodeEval, Topcoder, hackerearth, codingame, checkios e coderbyte. As simulações são frequentemente usadas para demonstrar o comportamento de estruturas de dados e algoritmos clássicos da literatura como árvores e listas e algoritmos de ordenação. Para o ensino do pensamento computacional, existem ainda ambientes que ensinam a partir do desenvolvimento de games. Alguns deles utilizam linguagens visuais que facilitam seu uso por crianças, como o Scratch, Pocket Code, Alice, Stencyl, Gamefroot, DesignBlocks, Hopscotch e Android App Inventor. Estes também não serão analisados neste trabalho.

A categorização e análise utilizou um conjunto de dimensões adaptadas de Connolly (et al., 2012). Esse teórico categorizou diversos jogos sérios considerando as seguintes dimensões: gênero, plataforma, disciplina, digital ou não digital e propósito principal. Algumas destas dimensões não são relevantes para este trabalho, pois iremos categorizar um grupo mais específico de games. Deste modo, foi realizada uma adaptação para incluir apenas as dimensões que julgou-se mais relevante ao estudo. Assim, foram consideradas aquelas que ajudam na escolha do game mais adequado as necessidades e limitações dos jogadores, a saber: gênero, plataforma, habilidades e linguagem utilizada.

A dimensão gênero utilizou a taxonomia proposta por (HERZ, 1997). Mesmo não sendo a taxonomia padrão aceita pela literatura, apresenta similaridade com as utilizadas pela indústria de games tornando mais fácil a categorização (CONNOLLY et al., 2012; KIRRIEMUIR; MCFARLANE, 2004). Contudo, alguns games podem se encaixar em mais de um destes gêneros. Herz (1997) considerou a taxonomia para categorização dos games apresentada na Tabela 1.

Tabela 1. Descrição dos gêneros dos games segundo a taxonomia de Herz.

Gênero	Descrição
Ação	Aquele em que a maioria dos desafios apresentados são testes de habilidades e coordenação física do jogador. Enigmas, conflito tático, e os desafios de exploração são comuns (ADAMS, 2013). Neles o jogador tipicamente controla o avatar de um protagonista. O avatar tem que navegar um nível, coletando objetos, evitando obstáculos e lutando contra inimigos com vários ataques.
Aventura	Aquele que tem como ênfase uma história interativa sobre um personagem principal, que é desempenhado pelo jogador. Narrativas e exploração são elementos essenciais do jogo. Enigmas e desafios conceituais compõem a maioria da jogabilidade. Combate, gestão econômica e desafios de ação são reduzidos ou inexistentes (ADAMS, 2013).
Luta	Aquele que enfatiza os combates de personagens controlados pelo computador, ou aqueles controlados por outros jogadores (KIRRIEMUIR, MCFARLANE 2004).
Quebra-cabeça	Aquele que enfatiza a resolução de enigmas. Eles podem testar muitas habilidades para resolver problemas, incluindo seguir um raciocínio lógico, reconhecer padrões, resolver sequências e completar palavras (KIRRIEMUIR, MCFARLANE 2004).
RPG	Aquele em que o jogador controla um ou mais personagens, geralmente projetados pelo jogador, e os orienta em uma série de missões gerenciadas pelo computador. Para vencer é necessário completar as missões. A evolução do poder e as habilidades do personagem é uma característica fundamental no jogo. Desafios típicos incluem combate tático, logístico, crescimento econômico, exploração e resolução de enigmas. Os desafios de coordenação física são raras, exceto em híbridos de ação (ADAMS, 2013).
Simulação	Aquele onde o jogador precisa ter sucesso dentro de alguma recriação simplificada de um lugar ou situação, por exemplo, um prefeito de uma cidade controlando finanças e construção de obras (KIRRIEMUIR, MCFARLANE 2004).
Esporte	Aquele que simula algum aspecto de um esporte atlético real ou imaginário. Pode ser jogado como: partidas, gerenciamento de equipe, de carreira, ou ambos (ADAMS, 2013).
Estratégia	Aquele em que a maioria dos desafios são estratégicos, onde o jogador pode escolher entre uma grande variedade de ações ou movimentos potenciais. Para vencer é necessário um planejamento superior e tomada de decisões ótimas; a sorte não desempenha um grande papel. Outros desafios, como os táticos logísticos, econômicos e de exploração, também podem estar presentes. Desafios de coordenação física tem pouca ou nenhuma influência (ADAMS, 2013).

A segunda dimensão utilizada foi a indicação da plataforma onde o game é executado. Segundo Apperley (2006), plataforma refere-se aos hardware em que o jogo é jogado . Isso inclui computadores pessoais, vários consoles Sony (*PlayStation*, Nintendo, Xbox, etc.), bem como dispositivos móveis. Contudo, este trabalho irá referir como plataforma os sistemas operacionais utilizados nos computadores (Windows®, Linux e OSX®), nos dispositivos móveis (iOS®, Android® e Windows Phone®) e a Web.

A terceira dimensão tem como objetivo listar as habilidades do pensamento computacional exploradas nos games estudados. Ressalta-se que estas referem-se às atividades requeridas para a formulação de problemas em termos de conversões entre entrada/saída e a construção dos algoritmos que realizam essas conversões. Foram delimitadas cinco habilidades utilizadas por alguns autores para demonstrar as habilidades do pensamento computacional explorados em seus games (BERLAND; LEE, 2011; KAZIMOGLU et al., 2012), conforme descrito na Tabela 2:

Tabela 2: Habilidades do pensamento computacional.

Habilidade	**Descrição**	**Exemplo**
Lógica condicional	Consiste na habilidade de compreender as consequências dos valores verdadeiro e falso. Usualmente através da utilização da construção se-então-senão.	Dados 3 números *x*, *y* e *z*. Se *x* é maior que *y* e *y* é maior que *z*. Então, sabemos que *x* é o maior. Senão, o maior valor será *y* ou *z*. (tabela verdade)
Construção de algoritmos	Consiste na habilidade de resolver um determinado problema, utilizando um conjunto de lógicas condicionais em uma abordagem passo a passo.	Dados 3 números *x*, *y* e *z*. Primeiro verifica-se se *x* é maior que *y* e *z*. Se for verdade, então *x* é o maior. Senão, preciso verificar se *y* é maior que *z*. Se for verdade então *y* é maior. Se não *z* é o maior.
Depuração	Consiste no ato de encontrar erros lógicos em um algoritmo que não funciona como esperado.	O algoritmo pode não comportar como esperado para uma determinada entrada. Em sala de aula é explorada através de atividades que envolvam encontrar um erro dentro de um algoritmo.
Simulação	Consiste no ato de modelar ou testar um algoritmo. São usadas tanto para a depuração quando para a construção de algoritmos.	Dado um algoritmo que retorna o maior valor, podemos simular o algoritmo para diferentes combinações de valores de *x*, *y* e *z*. Em sala de aula é explorada através de teste de mesa e execuções passo a passo.
Socialização	Refere aos aspectos sociais do pensamento computacional. Onde a solução de um problema pode ser alcançada e compartilhada entre uma ou mais pessoas.	A solução de um problema pode ser alcançada pela divisão e distribuição do problema para uma ou mais pessoas. Dojo de programação é um exemplo de metodologia que enfatiza essa habilidade (SATO et al., 2008). Ela enfatiza a programação em pares que codificam e compartilham com todos as suas soluções.

A quarta dimensão utilizada foi a linguagem, por ser o principal meio para o ensino e aprendizado do pensamento computacional. Em computação, os algoritmos são construídos através de uma linguagem visual ou textual, formal, não ambígua e computacionalmente tratável. Muitos jogos tem utilizado linguagens visuais por serem mais fáceis aos programadores novatos, pois evitam erros comuns de sintaxe. Por exemplo, o não fechamento de parênteses dentro de uma expressão matemática. Nas linguagens visuais a programação envolve arrastar e soltar blocos, podendo ser bem exploradas por dispositivos com telas sensíveis ao toque. Contudo, existem jogos que utilizam linguagens textuais populares na indústria de software como Java, JavaScript e Python. Estas linguagens tendem a ser menos restritas do que as visuais, sendo aplicadas em uma gama maior de problemas.

Na última etapa do estudo, foram utilizadas as cinco dimensões para comparar e analisar os games selecionados. Os resultados desta análise são apresentados na próxima seção.

4. Resultados

Utilizando a metodologia para seleção, incluímos neste trabalho os seguintes games: CargoBot, HardCoder, Lightbot, RoboZZle, RoboMind, Tynker, CodeSpells, Codemancer, CodeCombat, CodeHunt e FightCode. Esta seleção incluiu todos os games que durante a pesquisa tinham

documentação, podiam ser testados e não fossem projetos antigos e ou abandonados. Como apresentado na metodologia, não foram incluídos também os ambientes gamificados e simulações.

Os primeiros games analisados tem como objetivo básico movimentar um objeto a fim de solucionar um desafio. O primeiro que analisamos foi o Cargobot, um game para a plataforma iOS (CARGOBOT, 2014; TESSLER et al., 2013). Importante ressaltar aqui que ele foi desenvolvido diretamente nesta plataforma utilizando uma ferramenta para criação de jogos denominada Codea. O game é bem simples, mas com bons gráficos. O objetivo do game é movimentar as caixas que estão empilhadas em um estado inicial para um final. Essa tarefa é realizada por um braço mecânico que é movimentado através de quatro comandos básicos: para cima, para baixo, lado direito e esquerdo. Para realizar o desafio é necessário construir um algoritmo composto pela sequência dos comandos básicos. A Figura 1 mostra a tela principal do game dividido em três partes. Na parte superior mostra o desafio, na inferior é codificado o algoritmo e no meio é a simulação do funcionamento do braço mecânico. Existem diferentes desafios, que podem exigir a utilização de muitos comandos. Quanto menos comandos o jogador efetuar para alcançar o objetivo, mais pontos ele irá ganhar. Uma forma de reduzir a quantidade de comandos é através da modularização e de funções recursivas[11]. O usuário pode criar até quatro funções que podem ser chamadas a partir do módulo principal ou a partir de outras funções. Não existe comando explícito de repetição, porém é possível usar a recursividade para alcançar o mesmo resultado. Um ponto negativo é estar disponível apenas para a plataforma iOS.

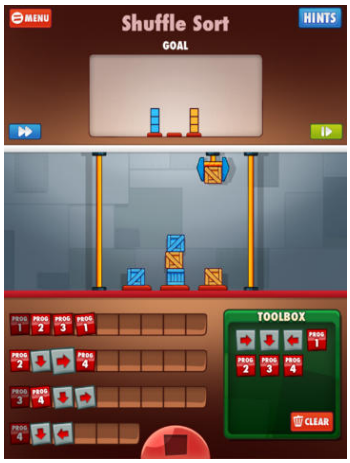


Figura 1: Exemplo de tela do CargoBot.
Fonte: Viana, 2012.

O Lightbot (LIGHTBOT, 2014) é outro game que tem um conceito similar, além de contar com bons gráficos. Desenvolvido para diversas plataformas, ele utiliza cinco comandos para movimentar um robô em um espaço virtual. Os comandos são: seguir em frente, virar a esquerda, virar a direita, pular e acender uma luz. Estes comandos são representados como ícones e são arrastados para montar o algoritmo. Conforme o jogador vai completando missões, através do alcance do objetivo, mais funções são liberadas. O objetivo do jogo é movimentar um robô dentro de um espaço quadriculado iluminando os quadrados que são solicitados. Similar ao CargoBot, existem suportes a procedimentos e recursividade. Existe ainda uma versão adaptada para crianças entre 4 e 8 anos, denominada LightBot Jr. Outro game similar, porém com gráficos mais simples e menos atrativo é o HardCoder. Ele tem como objetivo

[11] Em programação, funções recursivas são aquelas que chamam a sí própria.

recolher todas as frutas que aparecem na tela (NADAF, 2013). Para isso é usado uma sequência de comandos que movimenta uma seta nas direções cardeais (norte, sul, leste e oeste) e colaterais (nordeste, sudeste, noroeste e sudoeste). Existem 180 desafios com diferentes níveis de dificuldade, garantindo assim algumas horas de entretenimento e aprendizado. O HardCoder tem suporte a modularização de um modo similar ao LightBot e CargoBot. Contudo, existe suporte explícito de comandos de repetição, onde o jogador pode determinar a quantidade de vezes que um comando será executado. O HardCoder também é um game para dispositivos móveis, desenvolvido apenas para a plataforma Android.

Para ser jogado em computadores de mesa, porém ainda com uma dinâmica similar, nós estudamos o RoboZZle e RoboMind. O RoboZZle tem como atrativo poder ser jogado *online*, ele é denominado pelos criadores como um quebra-cabeça social, pois dá acesso direto a bate papos, fóruns, placares e a possibilidade de criar e compartilhar um desafio (OSTROVSKY, 2013). O objetivo de jogo é controlar um robô para coletar objetos localizados dentro de um espaço quadriculado Para isso são utilizados três comandos básicos: mover, girar e pintar. O game tem suporte a modularização, recursividade e comandos condicionais. O RoboMind tem um conceito similar, pois várias fases tem como objetivo fazer um robô virtual movimentar dentro de um espaço pintando determinadas regiões de branco ou preto (RESEARCH KITCHEN, 2005). Os comandos mais básico são movimentos, como norte, sul, leste e oeste, e pintar de branco ou preto. Possui suporte a modularização, repetição e comandos condicionais que verificam se existe um obstáculo a frente, a direita ou esquerda. É possível ainda saber qual a cor do quadrado onde o robô se encontra. Com estes comandos é possível construir algoritmos muito mais elaborados do que CargoBot, LightBot e HardCoder. Além disso, o game é muito bem documentado, com material de apoio para os professores e permite controlar robôs reais Lego Mindstorms. É um produto tem muita qualidade, porém o fato de ser pago pode ser visto como uma limitação para algumas escolas públicas. Todos os games citados anteriormente, permitem visualizar passo-a-passo a execução dos comandos. Deste modo, o jogador acompanha o que acontece para cada comando do código, ajudando a depurar o algoritmo. Eles não usam uma linguagem de programação convencional, os códigos são montados através de elementos gráficos que representam os comandos. Essa característica possibilita seu uso por públicos de todas as idades e sem necessidade de ter conhecimentos de programação. Entretanto, devido a seus comandos limitados eles são indicados apenas para a introdução ao pensamento computacional.

Os próximos games analisados utilizam linguagens mais próximas das convencionais, porém ainda visuais. Eles utilizam blocos que são agrupados de modo similar a um quebra-cabeça, como os utilizados no ambiente de programação Scratch (MALONEY et al., 2010). Uma das vantagens deste tipo de linguagem é evitar os erros de sintaxe, pois são muito comuns nas linguagens convencionais. O CodeSpells é um exemplo de game que utiliza programação em blocos. Ele iniciou como um projeto de pesquisa de doutorado de Sarah Esper e Stephen Foster (THOUGHTSTEM, 2014). É um dos poucos games em 3D usados para ensino de programação. Nele, o jogador controla um mago que encontra pelo caminho missões com problemas a serem resolvidos, ou inimigos para enfrentar. Todos os poderes e ações do personagem são controlados através da programação, aprimorando os conhecimentos de construção de algoritmos enquanto joga. A Figura 2 demonstra a utilização da programação baseada em blocos para controlar o personagem. É um game que tem como foco os públicos jovem e adulto. O jogo não segue um roteiro fixo, e as possibilidades de ação também são diversas, já que são dependentes do jogador. A versão final do jogo está prevista para setembro 2015 e custará 20 dólares. Pelos vídeos e versões anteriores, o game parece ser bem divertido, com várias

possibilidades de enredo, jogo online e múltiplos jogadores. Além de desenvolver o pensamento computacional e a criação de algoritmos, também estimula a criatividade. O fato de ser pago, porém, é um dos pontos negativos para o emprego em escolas públicas.

Figura 2: Exemplo de tela do CodeSpells.
Fonte: Thoughtstem, 2014

O Codemancer é outro exemplo de game 3D que ainda será lançado para Windows, OSX, iOS e Android (CODEMANCER, 2014). Segundo seus criadores, o Codemancer foi desenvolvido para ensinar a magia por trás da programação para crianças entre 9 e 14 anos de idade. O game apresenta uma história sobre uma menina tentando crescer e fazer o bem, apesar dos obstáculos incríveis em mundo de fantasia cheio de feiticeiros rivais e seus *minions*. A programação é realizada através de blocos. Tynker é outro exemplo de game que utiliza montagem de algoritmos em blocos (FUEL, 2014). Ele possui vários programas de ensino específicos para o aprendizado de programação. O objetivo do game são realizações de missões, resultando em pontos e conhecimentos ao usuário. Todos os cursos oferecidos iniciam com a introdução aos códigos em blocos, assim como a utilização dos mesmos. Logo após esta introdução, o usuário pode desfrutar de uma aprendizagem cheia de recursos interativos como vídeos explicativos e mini-games, ajudando assim a manter o jogador motivado para a realização do curso completo. Além disso, há tutores embutidos no jogo que ensinam a criança ou jovem a aplicar os conceitos passo-a-passo, evitando as possíveis desistências. A utilização de blocos para construir algoritmos foi muito explorada nos diversos games disponíveis no code.org[12]. Existem games para serem jogados dentre de um período aproximado de uma hora, sendo denominados a hora do código. Após a sua conclusão, o ambiente indica aos jogadores uma sequência de pequenos games que irão introduzir diversos conceitos de programação. Todos eles apresentam um desafio a ser completado através da codificação em blocos o que requer aproximadamente 20 horas. Todos os games são inspirados em personagens infantis ou jogos conhecidos como Angry Birds, Plants vs Zombie e Flappy Birds. Uma característica importante é os jogos estarem estruturados dentre de um ambiente gamificado. Um professor cadastrado pode criar turmas, adicionar e acompanhar a evolução dos alunos dentro do ambiente.

Além dos jogos que utilizam linguagens baseadas em blocos, analisou-se também aqueles que utilizam linguagens de programação convencionais. CodeCombat é um destes games no qual o

[12] O code.org (http://code.org) é um ambiente gamificado que inclui diversos games que ensinam o pensamento computacional. É possível criar turmas e acompanhar a evolução de cada aluno.

jogador controla um protagonista que vai se tornando cada vez mais forte a medida que os desafios vão evoluindo (CODECOMBAT, 2014). O controle do personagem é realizado por uma linguagem de programação escolhida pelo jogador dentre as seis suportadas. Antes de começar um desafio, são apresentados os comados e exemplos de como usá-los, como mostrado na Figura 3. Deste modo, torna-se mais fácil o uso por jogadores sem conhecimento prévio de programação. Destacam-se ainda, diversos recursos existentes em grandes games populares tais como: a escolha de personagens, o acumulo de pontos para avançar de nível, a compra de itens e equipamentos e a possibilidade de desafiar jogadores reais. O fato de usar linguagens de programação convencionais, possibilita ao jogador aprender a sintaxe de diversas linguagens, criar algoritmos e depurar os códigos com seus mecanismos de correção de erros. Outra característica em destaque é ser software aberto. Ele cumpre bem o objetivo de ensinar de um jeito divertido. Pode ser usado para ensinar programação para jovens e iniciantes, além de ser um ótimo passatempo para os mais experientes. Sua interface é bem atraente, o jogo é animado, cheio de possibilidades e muito interativo.

Figura 3. Tela inicial do CodeCombat usando a linguagem de programação Python.

CodeHunt é um outro game que utiliza linguagens de programação, onde o desafio é a depuração de códigos escritos em C# ou Java (MICROSOFT RESEARCH, 2014). Ele tem como foco jogadores com algum conhecimento nas linguagens utilizadas. Porém, o estilo do game e o número de níveis faz com que o jogador consiga desenvolver conhecimentos avançados nelas. Há ainda a possibilidade de criar seus próprios níveis e compartilhar seus resultados nas redes sociais. Ele é um bom game para quem já tem um conhecimento básico e quer se aprimorar nas linguagens suportadas. Não garante a mesma diversão do CodeCombat, mas os desafios dos níveis, sistema de pontos, rankings e a possibilidade de interação com as redes sociais ajuda a manter a motivação dos jogadores. Uma característica importante é o foco na depuração, ou seja, na correção de erros de programas. Por último, o Fightcode é um game onde o objetivo é programar um tanque robô de batalha contra outros usuários (FIGHTCODE, 2013). A programação requer um conhecimento prévio na linguagem de programação utilizada, neste caso, JavaScript. Destaca-se a socialização e a competição com outros jogadores. Contudo, não existem recursos explícitos para simulação e depuração. A Tabela 3 apresenta a sintetização de todos os games analisados a partir das quatro dimensões discutidas na Seção 2.

Tabela 3. Síntese dos jogos estudados
Fonte: dos autores

Nome	Gênero	Plataforma	Habilidades	Linguagem
CargoBot	Quebra-cabeças	iOS	Construção de Algoritmos, Depuração	Visual
HardCoder	Quebra-cabeças	Android e iOS	Construção de Algoritmos, Simulação	Visual
Lightbot	Quebra-cabeças	OSX, Windows, Web, iOS, Android e Não Digital	Construção de Algoritmos, Simulação	Visual
RoboZZle	Quebra-cabeças	Web, iOs, Android, Windows Phone	Lógica condicional, Construção de Algoritmos, Simulação, Socialização	Visual
RoboMind	Simulação	Windows XP/7/8, Mac OS X e Linux	Lógica condicional, Construção de Algoritmos, Simulação, Socialização	Textual e Visual
Tynker	Aventura	Web, iOS, Android	Lógica condicional, Construção de algoritmos, Depuração, Simulação	Baseado em blocos.
CodeSpells	Aventura	Windows e OSX.	Construção de Algoritmos, Simulação, Socialização	Baseado em blocos
Codemancer	Aventura	Windows, OSX, Android, iOS	Construção de Algoritmos, Simulação, Socialização	Baseado em blocos
CodeCombat	Luta	Web	Lógica condicional, Construção de algoritmos, Depuração, Socialização	JavaScript, CoffeeScript, Lua, Python, Io, Clonjure
CodeHunt	Simulação	Web	Lógica condicional, Construção de algoritmos, Depuração, Simulação e Socialização	Java, C#
FightCode	Luta	Web	Lógica condicional, Construção de algoritmos, socialização	JavaScript

Dentre estes jogos o LightBot está presente para diferentes públicos alvos e plataformas. Além disso ele pode ser jogado inclusive utilizando apenas recortes de papéis sem a necessidade de um computador ou dispositivo móvel. O CodeHunt se destaca por trabalhar todas as habilidades, contudo tem como público alvo jogadores com algum conhecimento prévio em programação. Por outro lado, o CodeCombat suporta uma quantidade maior de linguagens de programação. Ele ainda conta com uma comunidade bem ativa, que se comunicam através de diversos canais que são acessadas diretamente pelo portal do game. O CodeSpells tem gráficos muito bons e enredo similar a grandes jogos populares, porém ainda não foi lançado oficialmente. Por fim, o CargoBot é um game que leva a impressão de poder ser jogado mesmo por aqueles que não tenham como objetivo principal o aprendizado de programação.

5. Considerações Finais

Os computadores sempre foram reconhecidos como importantes ferramentas para fins científicos, militares e empresariais. Mais recentemente, eles passaram a fazer parte do dia a dia das pessoas. Porém, saber como programá-los os tornam muito mais flexíveis, pois não ficamos limitados a usá-los através dos aplicativos já existentes. Para isso é necessário uma forma diferente de pensar, denominada pensamento computacional. Este trabalho discutiu o ensino e aprendizado do pensamento computacional através de games. Neste primeiro estudo não foi incluído gamificação e nem simulação. Teve como foco apenas aqueles que pudessem ser jogados em computadores ou dispositivos móveis, como *tablets* e *smartphones*, não foi incluído jogos de console como o Playstation ou Xbox. Eles foram analisados a partir de quatro principais dimensões: gênero, plataforma, habilidades e linguagem. Dentre os games analisados, a maioria tem como foco crianças, em alguns casos a partir dos quatro anos de idade. Eles estão presentes em diversas plataformas, incluindo a Web, Android e iOS. Alguns utilizam linguagens mais limitadas, enquanto outros utilizam linguagens de programação convencionais, como JavaScript e Python. O gênero mais comum é o de quebra-cabeças, usualmente jogado a partir de uma linguagem visual e simplificada. A escolha do game como processo de ensino e aprendizado irá depender do público alvo e dos conceitos que serão trabalhados. Um professor pode usar diferentes games no desenvolver do pensamento computacional, começando com os mais limitados, por exemplo, o CargoBot e o LightBot. Avançando para linguagens visuais baseados em blocos, até chegar em games que utilizam linguagens de programação convencionais. Espera-se que este trabalho sirva tanto para aqueles que pretendem usar games no ensino, quanto aqueles que estão interessados em iniciar suas pesquisas. Uma lacuna encontrada nesta pesquisa foi a falta de estudos que demonstram as metodologias de como integrar estes games as unidades curriculares de cursos de ensino técnico e ou superior. Muitos deles tem como foco crianças, porém acredita-se que eles possam ser integrados também ao ensino técnico e ou superior. Em uma experiência recente foi observado um grande interesse por estes games por discentes do curso Bacharelado Interdisciplinar em Ciência e Tecnologia (BICT) da Universidade Federal do Maranhão. Os alunos concluíram muito mais níveis do que os solicitados pelo professor. Essa constatação motiva a fazer nestes estudo futuro sobre a integração destes games aos processos de ensino e aprendizados em cursos. Neste estudo futuro poderá ser incluído ainda os ambientes gamificados, de competição e de ensino baseado no desenvolvimento de games.

Referências Bibliográficas

ADAMS, E. **Fundamentals of Game Design**. Berkley, CA: New Riders, 2013.

AHO, A. V. Computation and computational thinking. **Computer Journal**, v. 55, n. 7, p. 832–835, 2012. Disponível em: <http://link.springer.com/article/10.1007/BF00413693>. Acesso em: 10/12/2014.

APPERLEY, T. H. Genre and game studies: Toward a critical approach to video game genres. **Simulation & Gaming**, v. 37, n. 1, p. 6–23, 2006. Disponível em: <http://sag.sagepub.com/cgi/doi/10.1177/1046878105282278>. Acesso em: 15/7/2014.

BALAMURALITHARA, B.; WOODS, P. C. Virtual laboratories in engineering education: the simulation lab and remote lab. **Computer Applications in Engineering Education**, v. 17, p. 108–118, 2009.

BARCELOS, T.; SILVEIRA, I. Pensamento Computacional e Educação Matemática: Relações para o Ensino de Computação na Educação Básica. XX Workshop sobre Educação em Computação. **Anais...** , 2012. Curitiba - PR.

BERLAND, M.; LEE, V. R. Collaborative Strategic Board Games as a Site for Distributed Computational Thinking. **International Journal of Game-Based Learning**, v. 1, n. 2, p. 65–81, 2011. Disponível em: <http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/ijgbl.2011040105>. Acesso em: 3/11/2014.

BERLINSKI, D. **Advento do Algoritmo**. Rio de Janeiro, RJ: Globo, 2002.

CARGOBOT. Cargobot. Disponível em: <http://twolivesleft.com/CargoBot/>. Acesso em: 10/1/2014.

CODECOMBAT. CodeCombat. Disponível em: <http://codecombat.com>. Acesso em: 5/10/2014.

CODEMANCER. Codemancer. Disponível em: <http://codemancergame.com>. Acesso em: 10/11/2014.

CONNOLLY, T. M.; BOYLE, E. A.; MACARTHUR, E.; HAINEY, T.; BOYLE, J. M. A systematic literature review of empirical evidence on computer games and serious games. **Computers & Education**, v. 59, n. 2, p. 661–686, 2012. Elsevier Ltd. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0360131512000619>. Acesso em: 11/7/2014.

CONWAY, J. The game of life. **Scientific American**, v. 223, n. 4, p. 4, 1970. ACM Press.

CSTA.ISTE. Computational Thinking in K–12 Education leadership toolkit. Disponível em: <http://csta.acm.org/Curriculum/sub/CurrFiles/471.11CTLeadershiptToolkit-SP-vF.pdf>. Acesso em: 10/12/2014.

DETERDING, S.; KHALED, R.; NACKE, L.; DIXON, D. Gamification: toward a definition. CHI 2011 Gamification Workshop Proceed. **Anais...** . p.12–15, 2011.

ESA. Essential Facts About the Computer and Video Game. Disponível em: <http://www.theesa.com/wp-content/uploads/2014/10/ESA_EF_2014.pdf>. Acesso em: 12/12/2014.

FARDO, M. L. **A gamificação como estratégia pedagógica: estudo de elementos dos games aplicados em processos de ensino e aprendizagem**, 2013. Universidade de Caxias do Sul.

FIGHTCODE. FightCode. Disponível em: <http://fightcodegame.com>. Acesso em: 8/11/2014.

FRANÇA, R.; SILVA, W.; AMARAL, H. Ensino de ciência da computaçao na educaçao básica: Experiências, desafios e possibilidades. XX Workshop de Educação em …. **Anais...** , 2012.

FUEL, N. Tynker. Disponível em: <http://www.tynker.com>. Acesso em: 10/11/2014.

GARCIA, I. Learning a Language for Free While Translating the Web. Does Duolingo Work? **International Journal of English Linguistics**, v. 3, p. 19–25, 2013. Disponível em: <http://www.ccsenet.org/journal/index.php/ijel/article/view/24236>. .

GILBERT, N. **Agent-Based Models**. Guildford, UK: Sage, 2008.

GOLDSTINE, H. H.; GOLDSTINE, A. Electronic numerical integrator and computer (ENIAC). **IEEE Annals of the History of Computing**, v. 18, n. 1, p. 10–16, 1996.

GOODCHILD, M. F.; YUAN, M.; COVA, T. J. Towards a general theory of geographic representation in GIS. **International Journal of Geographical Information Science**, v. 21, n. 3, p. 239–260, 2007.

HERZ, J. C. **Joystick nation : how videogames ate our quarters, won our hearts, and rewired our minds**. Boston: Little, Brown and Company, 1997.

HUIZINGA, J. **Homo ludens: o jogo como elemento da cultura**. 1º ed. São Paulo: Editora da Universidade de S. Paulo, Editora Perspectiva, 1971.

JORGENSEN, W. L. Foundations of biomolecular modeling. **Cell**, v. 155, n. 6, p. 1199–202, 2013. Elsevier Inc.

KAPP, K. M. **The Gamification of Learning and Instruction Fieldbook: Ideas Into Practice**. New York, New York, USA: John Wiley, 2013.

KAZIMOGLU, C.; KIERNAN, M.; BACON, L.; MACKINNON, L. Learning programming at the computational thinking level via digital game-play. **Procedia Computer Science**, v. 9, n. 0, p. 522–531, 2012. Elsevier Masson SAS. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S1877050912001779>. Acesso em: 5/12/2014.

KHAN, S. Khan Academy. Disponível em: <http://www.khanacademy.org/>. Acesso em: 1/10/2014.

KIRRIEMUIR, J.; MCFARLANE, A. **Literature review in games and learning**. 2004.

LAUYSE, M.; OLIVEIRA, S. DE; SOUZA, A. A. DE; et al. Ensino de lógica de programação no ensino fundamental utilizando o Scratch : um relato de experiência. Congresso da Sociedade Brasileira de Computação. **Anais...** . v. 12, p.1493–1502, 2014.

LIGHTBOT. LightBot. Disponível em: <http://lightbot.com>. Acesso em: 10/12/2014.

MALONEY, J.; RESNICK, M.; RUSK, N.; SILVERMAN, B.; EASTMOND, E. The Scratch Programming Language and Environment. **ACM Transactions on Computing Education (TOCE)**, v. 10, p. 16:1–16:15, 2010.

MICROSOFT RESEARCH. Codehunt. Disponível em: <https://www.codehunt.com>. Acesso em: 8/11/2014.

NADAF, I. Hardcoder. Disponível em: <https://play.google.com/store/apps/details?id=org.iliyas.HardCoder>. Acesso em: 10/10/2014.

OSTROVSKY, I. Robozzle. Disponível em: <http://www.robozzle.com>. Acesso em: 10/10/2014.

RESEARCH KITCHEN. Robomind. Disponível em: <http://www.robomind.net/en/index.html>. Acesso em: 10/10/2014.

SALEN, K.; ZIMMERMAN, E. **Rules of Play: Game Design Fundamentals**. MIT press, 2004.

SCHELLING, T. C. Dynamic models of segregation. **The Journal of Mathematical Sociology**, v. 1, n. 2, p. 143–186, 1971.

SOLOMON, C.; PAPERT, S. A case study of a young child doing Turtle Graphics in LOGO. Proceedings of the June 7-10, 1976, national computer conference and exposition. **Anais...** . p.1049–1056, 1976.

TESSLER, J. J.; BETH, B.; LIN, C.; TESSLER, J. J. Using cargo-bot to provide contextualized learning of recursion. Proceedings of the ninth annual international ACM conference on International computing education research - ICER '13. **Anais...** . p.161, 2013. Disponível em: <http://dl.acm.org/citation.cfm?doid=2493394.2493411>. .

THE ROYAL SOCIETY. Shut down or restart? The way forward for computing in UK schools. , 2012. London, UK: The Royal Society.

THIEL, W.; HUMMER, G. Nobel 2013 Chemistry: Methods for computational chemistry. **Nature**, v. 504, n. 7478, p. 96–97, 2013. Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved. Disponível em: <http://dx.doi.org/10.1038/504096a>.

THOUGHTSTEM. CodeSpells. , 2014. Disponível em: <http://codespells.org/>.

WING, J. M. Computational thinking. **Communications of the ACM**, v. 49, n. 3, p. 33, 2006. Disponível em: <http://portal.acm.org/citation.cfm?doid=1118178.1118215>.