Jogos com Propósito para o Ensino de Programação

Rafael Studart Monclar[1,2*] Marcelo Arêas Silva[1,2] Geraldo Xexéo[1,2,3]

Programa de Engenharia de Sistemas e Computação - COPPE/UFRJ[1]

Laboratório de Ludologia, Engenharia e Simulação - LUDES/UFRJ[2]

Departamento de Ciência da Computação - IM/UFRJ[3]

RESUMO

As dificuldades encontradas nos cursos superiores de ciência da computação para ensinar programação têm gerado um afastamento cada vez maior por parte dos estudantes. As taxas de evasão aumentam e nada significativo tem sido feito para reverter esse cenário. Nossa pesquisa, porém, identificou que jogos digitais com o propósito de ensinar esse tipo de conhecimento seria o caminho mais imediato para aumentar o engajamento dos alunos nessa área. Partindo dessa premissa, os autores realizaram uma revisão, desde o ano de 2001 até os dias atuais, buscando identificar jogos que promovam o desenvolvimento desse tipo de competência. Também foram considerados aqueles que fundamentavam os pilares do pensamento computacional, uma vez que são capazes de despertar o interesse em um público mais amplo e mais jovem. São apresentados neste artigo uma lista de 26 jogos, onde se destacam características importantes destes. Nosso intuito, com este trabalho, é estimular iniciativas educacionais para o desenvolvimento ou uso de jogos para ensinar um conhecimento que se mostra relevante, porém, atualmente, desestimulante.

Palavras-chave: programação, educação, ensino, jogos com propósito, pensamento computacional.

1 INTRODUÇÃO

Os alunos costumam enfrentar dificuldades em cursos de ciência da computação, principalmente com a parte da programação de computadores, embora sua familiarização com tais dispositivos pudesse sugerir que o aprendizado de programação se tornaria mais fácil hoje em dia [1].

Pesquisas realizadas entre 1985 e 2005 afirmam que essas dificuldades se mantiveram presentes e os alunos parecem ainda menos interessados em programar [2].

Os desafios encontrados dizem respeito à complexidade do domínio, que inclui conhecimentos elaborados associados a conceitos abstratos, além do desenvolvimento árduo de habilidades cognitivas de cunho prático, como raciocínio lógico, resolução de problemas e abstração de alto nível [3]. Tal somatório de desafios dificulta o ensino de programação de computadores, pois os alunos geralmente caracterizam os cursos como entediantes e difíceis de entender. Além disso, muitos consideram os métodos de ensino utilizados como desinteressantes e afirmam que não são fornecidos exercícios o bastante para praticar os conceitos que aprendem [3].

Ao mesmo tempo em que dificuldades se acumulam sem solução, o número de inscrições no curso de computação diminui e a demanda global por desenvolvedores de software vem aumentando [4]. Como um agravante desse cenário, as taxas de insucesso e evasão são altas, principalmente nos cursos introdutórios de programação de computadores [5].

O ensino e a aprendizagem bem-sucedidos de disciplinas de programação podem ser extremamente benéficos para os alunos da geração do século XXI, o que traz mais importância para que tais dificuldades sejam amenizadas, ou mesmo solucionadas. Esse tipo de ensino permite o desenvolvimento de várias competências, como o pensamento crítico, permitindo que os alunos criem seus próprios programas, além da análise de conceitos e solução de problemas, já que isso é geralmente necessário para decifrar um determinado cenário e traduzi-lo em linhas de código. Também observamos os alunos aprendendo a trabalhar em grupos e colaborar uns com os outros em seus esforços para desenvolver programas executáveis enquanto se exercitam na troca de conhecimentos e na comunicação de ideias. Desse modo, a programação geral de computadores permite que os alunos se tornem aprendizes ao longo da vida, um benefício muito importante para o cenário atual de globalização e interdisciplinaridade, já que essas habilidades podem ser aproveitadas em vários domínios de trabalho futuros (e.g. engenharia, ciência da computação, etc.) [6].

Uma proposta que tem sido utilizada para aliviar os problemas descritos é a incorporação de jogos educativos ou jogos com propósito em cursos de programação de computadores. O termo *jogos educativos* é comumente empregado para descrever jogos de computador que são usados como ferramentas educacionais e fornecem atividades interativas e envolventes que atraem o interesse dos alunos em aprender [7].

Tais jogos são populares em todos os gêneros, etnias e linhas socioeconômicas, ademais seu uso parece estar ligado ao desenvolvimento de habilidades positivas, como atenção, e habilidades visuo-espaciais [4][8], desempenho em sala de aula [9][10], criatividade, disposição para novas atividades e ambientes [11], motivação para o sucesso e engajamento [12].

Já os jogos com propósito, que por vezes são chamados de *jogos sérios*, são aqueles desenvolvidos de acordo com o conceito de "jogos digitais usados para resolver uma ampla gama de desafios sociais relacionados à política, negócios, medicina, assuntos militares e educação inclusiva" [13]. Eles incentivam a aprendizagem por meio de jogos e permitem que o aluno viva experiências a baixo custo e baixo risco que, de outra forma, provavelmente possuiriam um custo elevado e um alto risco. Há muitos campos que são cobertos por jogos com propósito e eles também são destinados a uma variedade de alunos [13].

Entendendo que há um problema na facilitação do aprendizado de disciplinas que ensinam programação e que os jogos com propósito podem ser uma solução viável para resolvê-lo, colocamos como principal objetivo deste artigo a busca e revisão de jogos educacionais existentes que têm como foco ensinar programação de computadores. Em nossa análise buscamos identificar a eficácia com que os jogos apoiam os alunos na conquista dos objetivos educacionais que os professores estabelecem em cada caso [1].

Nas próximas seções apresentamos como essa pesquisa foi realizada (Seção 2), seguida da análise dos jogos educacionais para o ensino de programação, assim como comentários e críticas aos resultados encontrados (Seção 3), finalizando com as conclusões (Seção 4).

*e-mail: rastumon@gmail.com

2 METODOLOGIA DE PESQUISA

A pergunta que norteou nossa busca foi: "Quais são os jogos publicados na literatura acadêmica ou disponíveis comercialmente, que foram criados para auxiliar o ensino de programação?".

Nosso trabalho foi realizado seguindo uma metodologia de revisão narrativa, pois foi nossa primeira abordagem a um projeto que pretende ser maior. Além disso, tivemos dificuldades em encontrar informações suficientes que se enquadrasse em critérios pré-definidos por metodologias como as expostas por Brereton [14].

Pesquisamos uma variedade de bancos de dados acadêmicos, como Web of Science, Scopus, CiteSeer e Google Scholar. As buscas envolveram palavras-chave como: jogos educacionais, programação de computadores, introdução, adolescentes e jovens. Utilizamos como critério de aceitação trabalhos que descreviam jogos utilizados, propostos ou desenvolvidos para dar apoio ao aprendizado introdutório de programação. Não consideramos o ensino de linguagem de programação através da programação de um jogo, ou seja, nossa pesquisa utilizou somente jogos já construídos. Mais de 60 artigos foram originalmente identificados, além de artigos que encontramos a partir das referências. Ao final, após passar pelo critério de aceitação obtivemos 17 artigos para realizar nossa análise.

É importante ressaltar que o presente artigo contempla os jogos que se relacionavam de maneira explícita ao ensino de programação, ainda que fossem utilizados com o foco na introdução e desenvolvimento da lógica computacional para crianças. Consideramos importante incluí-los, não só porque eles representaram cerca de 50% dos resultados de nossas buscas, como também porque enxergamos que esse tipo de jogo pode servir como ponte para familiarizar o público mais novo com os conceitos e ambientes da área de desenvolvimento de software. Adicionalmente, com posse desses jogos voltados para pessoas mais novas, seria mais fácil entendermos, a posteriori, como esse tipo de conhecimento seria absorvido pelas pessoas durante as diversas fases de sua vida.

Por fim, buscamos também pelos termos "jogos" e "programação" na AppStore, Steam e Google Play. Porém, tanto para a busca de jogos com viés acadêmico, quanto para o comercial, consideramos em nosso critério de aceitação aqueles em que os desafios, ou as missões, estão incluídas no jogo e podem averiguar se as respostas dos jogadores estão corretas. O que significa que os estudantes podem ser autônomos e jogar, enquanto aprendem, sem a intervenção de um professor [15].

Na seção seguinte apresentamos um pouco da motivação que nos levou a buscar os jogos com o propósito de ensinar programação, bem como a análise destes.

3 JOGOS EDUCACIONAIS PARA ENSINO DE PROGRAMAÇÃO

3.1 Motivação

Cursos de programação frequentemente apresentam altas taxas de reprovação e desistência, sendo que um motivo importante é a dificuldade do ensino de programação, ainda considerado uma tarefa complexa [16].

O ambiente computacional que os alunos usam diariamente, para jogar ou bater papo, por exemplo, é muito diferente do que eles usam para aprender a programar, e eles não veem imediatamente a conexão entre os dois universos: "Quem estuda técnicas pedagógicas concorda que alunos que são novos à ciência da computação considera essa área, tipicamente, cheia de conceitos teóricos, técnicos ou mesmo tediosos " [17].

Outro aspecto bastante comentado é a motivação dos alunos, uma vez que muitos veem a programação como uma atividade cansativa e entediante [18][19][20]. Na revisão sistemática realizada por Souza [21], as categorias de soluções mais sugeridas pelos artigos para esse problema colocaram "Visualização" em primeiro lugar com 21% e "Jogos com propósito" em segundo, com 15%.

Existem pesquisas [22][23][24] sobre o uso de jogos cujas características suportam o ensino e a aprendizagem de programação de computadores. A maioria desses jogos inclui um cenário particular que visa cobrir uma unidade específica da área de programação de computadores, enquanto alguns jogos - em menor número - cobrem múltiplos objetivos de aprendizado e unidades teóricas [25].

Embora existam escolas criando programas inteiros em torno do desenvolvimento de jogos [26][27][28][29][30] para atrair estudantes, nosso foco foi especificamente os jogos que ensinam a programar.

Os currículos que usaram jogos com propósito para se especializarem na aprendizagem de programação encontraram efeitos positivos nos alunos, bem como nos resultados dela [31].

O ponto crítico dos jogos com propósito é a relação entre o jogo e o conteúdo educacional: Zyda escreveu que "A pedagogia deve estar subordinada à história - o componente de entretenimento vem primeiro" [32]. Uma hipótese é que se o jogo é atraente, divertido, estimulante e encoraja o jogador a progredir, então ele vai aprender automaticamente habilidades do jogo e vai absorver uma grande quantidade de informações [33].

Malone considera que as características essenciais de bons jogos de computador e outras situações intrinsecamente agradáveis podem ser organizadas em três categorias: desafio, fantasia e curiosidade [34].

De acordo com [34], a programação de computadores em si é um dos melhores jogos de computador. No "jogo de programação de computadores" há objetivos óbvios e é fácil gerar mais. O *jogador* recebe *feedback* frequente de desempenho. O jogo pode ser jogado em diferentes níveis de dificuldade, e há muitos níveis de objetivos disponíveis, tanto em termos de produto acabado (se funciona, quão rápido ele funciona, quanto espaço requer, etc.) quanto em termos do processo de alcançá-lo (quanto tempo leva para programar, etc.). A autoestima está muito envolvida neste jogo, e provavelmente existem aspectos ocasionais de emoção ou fantasia envolvidos no controle tão completo, mas frequentemente tão ineficiente, do comportamento dessa entidade responsiva. Por fim, o processo de depuração de um programa talvez seja incomparável em sua capacidade de aumentar as expectativas sobre como ele funcionará, tudo isso apenas para fazer tais expectativas serem desapontadas de maneiras que revelam a verdadeira estrutura subjacente do programa [34].

Com base nessa motivação, apresentamos na próxima subseção nossa análise sobre os jogos encontrados.

3.2 Relação dos Jogos e Análise do Resultado

Optamos por dividir nosso relato em duas grandes categorias, de acordo com a faixa etária: "jogos para crianças e adolescentes" e "jogos para universitários". A primeira possui um foco introdutório para criar os alicerces computacionais que posteriormente podem ser utilizados em disciplinas acadêmicas. Já a segunda busca apresentar jogos com um teor ou linguagem mais desenvolvidos para alunos de nível superior, mesmo que também possuam um caráter de apresentação de conteúdo.

Infelizmente, 38% dos jogos encontrados em artigos científicos não está disponível para avaliação. Nesses casos tivemos que levar em conta apenas o que nos foi apresentado pelos autores dos mesmos.

3.2.1 Jogos para Crianças e Adolescentes

Interfaces mais infantis e ilustradas, na maioria das vezes lidando com linguagem de blocos, e histórias menos elaboradas caracterizam os 13 jogos a seguir. As abordagens encontradas em cerca de 62% deles tem como foco o desenvolvimento do

pensamento computacional [35] para resolução de problemas, porém também encontramos outros mais elaborados. São eles:

A. **Train: Build and Program it** (2011) [22] é um jogo de simulação, onde o aluno constrói ferrovias e projeta os comportamentos de transporte dos trens no sistema de trilhos. O foco do jogo é para compreender a capacidade do aluno em resolver problemas através da sua abstração e decomposição. Esse jogo entra na seara do desenvolvimento do pensamento computacional, mais do que propriamente da programação. Houve uma amostra considerável, com 112 estudantes que avaliaram o jogo. Desses, 64 atingiram o estado de fluxo, 38 ficaram ansiosos e 10 entediados utilizando o jogo como ferramenta para o aprendizado de programação. Quando comparado com o método tradicional de ensino, os resultados foram 30, 78 e 10, respectivamente. A diferença numérica se deu, pois, 3 alunos não participaram da simulação [22]. O jogo em si não é particularmente elaborado, mas serviu para mostrar que os estudantes engajados buscavam soluções mais criativas para resolver seus problemas, enquanto os que se mostraram entediados foram em busca de alternativas mais superficiais;

B. **Program your robot** (2012) [36] é projetado para permitir que os alunos pratiquem o uso de conceitos introdutórios de programação dentro de um ambiente que suporta explicitamente a aquisição de habilidades como construção de algoritmos, depuração e simulação. O objetivo do jogo é ajudar um robô a escapar de uma série de plataformas, construindo um plano de fuga chamado *algoritmo de solução*. Os jogadores constroem seu algoritmo de solução dando vários comandos para o robô executar. Estes são divididos em dois: comandos de ação e comandos de programação. Os de ação têm um efeito direto no robô (*i.e.* avançar, virar à esquerda), enquanto os de programação afetam indiretamente essas ações, suportando a solução desenvolvida pelo jogador (*i.e.* repetição de uma série de comandos ou tomada de decisão através de uma condição). É um jogo introdutório para crianças e sem nenhuma particularidade inventiva, principalmente se considerarmos sua data de publicação, de 2012, que o coloca atrás de outros que o precederam como Light-Bot (2008) e RoboZZle (2009);

C. **Parson's programming puzzles** (2006) [37] busca resolver o treinamento da sintaxe, considerado pelos autores como tedioso. Para tal, criaram esse quebra-cabeças no estilo *arraste-e-solte*, uma vez que identificaram que quebra-cabeças e jogos são importantes para o ensino de ciência da computação tanto em cursos básicos quanto avançados. Eles notaram que esse tipo de abordagem ajuda a acomodar um espectro mais amplo do estilo de aprendizado individual, se comparado com métodos mais tradicionais. Os autores desse jogo consideraram que o foco de um curso que busca ensinar programação precisa ser na sintaxe e no aprendizado mecânico, pelo menos em algum grau [37]. Eles a comparam ao aprendizado de matemática ou línguas estrangeiras, em que técnicas de repetição, exposição e prática excessiva garantem a absorção do conteúdo [38]. Porém, isso parece negligenciar o aspecto criativo que a programação é capaz de desenvolver, além de deixar de lado a semântica, que vem antes mesmo da sintaxe;

D. **Peeps** (2008) [39]. Esse jogo surgiu como parte de um projeto maior, **Rapunsel**, para incluir meninas no mundo da computação [40][41]. Sua principal mecânica é a dança. Um personagem central é atribuído ao jogador, que pode programá-lo para dançar, se mover ou se comportar de várias formas. A linguagem Java é apresentada aos jogadores através de exercícios guiados que lhes permitem criar passos e sequências de dança. A estrutura de recompensa abarca tanto os jogos cooperativos quanto os competitivos. Jogadores podem competir entre si em torneios de dança, ou eles podem optar por coletar e trocar códigos, decorar suas *casas* no jogo, criar músicas para as danças, e contribuir com códigos para a biblioteca compartilhada, onde outros jogadores poderão baixá-los e dar notas. Há recompensas pela realização de lições que introduzem conceitos e procedimentos de programação cada vez mais complexos, assim como a criação de códigos mais sofisticados e originais [42]. Deste modo, em vez de usar a programação para criar jogos inteiros, espera-se que os jogadores aprendam novas habilidades por meio de uma combinação de aulas interativas e modificações, ou criações, de novas ações e conteúdos dentro do jogo. Rapunsel representa uma variação relevante da noção de usar jogos para ensinar programação, principalmente por focar em um público que costuma ser negligenciado pelo mercado há anos. Infelizmente, o projeto foi de 2003 a 2006, o que parece explicar o motivo do jogo não estar mais disponível ou sequer em uso;

E. **Cargo-Bot** (2012) [43] foi desenvolvido para iPad com foco em ensinar a lógica de instruções em blocos de semântica facilmente identificáveis. Os desafios vão crescendo em dificuldade a medida que o jogador avança pelas fases. O jogo pareceu ser mais introdutório, com foco em quem não tem experiência alguma na área de programação, embora entretenha o jogador caso contrário. Os gráficos são bem desenvolvidos e atualmente ainda é possível baixá-lo para jogar;

F. **Catos Hike** (2013) [44] tem como história por trás uma criança chamada Cato, que cai num portal e precisa resolver quebra-cabeças usando a lógica de programação. Segue a mesma linha de *Cargo-Bot* e está disponível para iOS. Segue a premissa do Cargo-Bot em termos de público-alvo, no entanto possui mais possibilidades de uso, principalmente por incluir um editor de mapas. Ainda assim, é mais infantil que o jogo anterior;

Figura 1: Imagem retirada do jogo *Catos Hike*.

G. **Code Combat** (2014) [45] é uma solução fechada que ensina alunos do ensino fundamental e médio a programar usando um RPG. As linguagens usadas são Python e Javascript. Essa pareceu ser a solução mais comercial encontrada, aparentemente focada em venda para colégios, porém com uma opção para a criança jogar pelo sítio online deles de maneira gratuita também, caso queira explorar os conceitos por conta própria. O que encontramos indica que os criadores do jogo o desenvolveram de maneira modular, de tal forma que a escola compra um pacote com a fase que querem desenvolver e com isso vem uma apostila do professor e do aluno. Ou seja, não pareceu haver muito espaço para criatividade, visto que o foco não era conectar professores experientes em programação com alunos iniciantes. A impressão é de uma preocupação maior com a divulgação do ensino de programação pelo colégio, em detrimento do acompanhamento da evolução do aprendizado das crianças;

H. **Dream Coders** (2012) [46] apresenta um RPG em 2D com missões para que o jogador possa salvar uma princesa. Tais missões envolvem a programação de códigos em Java. Aparentemente surge como um embrião do Code Combat, ainda mais por ser 2D, mas a maneira de inserir a programação no jogo com o uso de scripts distancia o jogador do ambiente do jogo, pela existência de dois modos de interface, quebrando, de certa forma, a sensação de imersão;

I. **Light-Bot 2** (2010) [47] é um jogo menos elaborado que visa ensinar conceitos básicos de pensamento computacional para quem não possui experiência alguma na área de programação. Buscamos maiores informações no site oficial, porém ele não tinha mais nenhuma relação com o jogo. O original também não foi encontrado. Tudo que achamos foi a versão 2.0 do jogo em um sítio agregador de jogos em *Flash*. Com relação ao jogo em si, não há nenhuma novidade em relação aos tantos outros que lidam com a lógica visual de blocos para o encaixe sequencial de instruções buscando resolver problemas;

J. **Machinist-Fabrique** (2013) [48], assim como *Light-Bot 2*, é um jogo de resolução de problema através de uma sequência de comandos, com foco no público infantil. O criador optou por fazer uso de uma linguagem de blocos textuais, provavelmente para facilitar a transição para linguagens de programação como Scratch [49] ou mesmo linguagens de script.

K. **Move the Turtle** (2012) [50] é um jogo de iOS, também voltado para o ensino dos primeiros passos de programação para crianças, onde o jogador precisa escolher a sequência de passos que uma tartaruga irá fazer para cumprir os desafios da fase em que ela se encontra. O jogo apresenta uma interface sofisticada que repete a filosofia das primeiras aulas de um curso da linguagem de programação Logo [51]. Ele está devidamente adaptado e nivelado com a estrutura atual de jogos como Angry Birds [52] ou Cut the Rope [53], em que os desafios vão sendo incrementais e você pode ganhar mais ou menos estrelas na fase que se encontra, dependendo de como você a solucione.

L. **Robo Logic** (2014) [54] também é para iOS e com o mesmo princípio dos anteriores, seguindo uma programação em blocos para guiar o personagem principal. Sem nenhuma inovação em particular, é um exemplar da mesma linha que ensina a lógica de instruções sequencias para crianças;

M. **RoboZZle** (2009) [55]. Os quebra-cabeças desse jogo tem como objetivo a coleta de todas as estrelas que aparecem na fase. Para isso, o jogador deve usar a lógica de uso dos comandos sequenciais em blocos gráficos. Apresenta gráficos menos elaborados que os demais jogos de quebra-cabeça cuja a resolução é feita com o sequenciamento de blocos que realizam ações. Esse exemplar teve micro-inovações como o uso de cor para ensinar instruções condicionais.

3.2.2 Jogos para Universitários

Nossa busca identificou para alunos mais maduros, em cursos universitários, jogos com um foco maior em sintaxe e linguagem de script, mesmo aqueles mais introdutórios. Desde jogos do início do século XXI até um RPG de 2016, todos tentam envolver os estudantes em universos fictícios para que sejam capazes de compreender fundamentos de programação. São eles:

A. **Robocode** (2001) [56], desenvolvido pela IBM, é um dos primeiros ambientes desenvolvidos como um jogo educacional de código aberto para suportar Java. O objetivo é desenvolver um algoritmo para controlar tanques em uma batalha. Cada jogador programa seu tanque. Os alunos desenvolvem sua estratégia de guerra usando a programação Java ou .Net e as batalhas acontecem interativamente quando todos os jogadores completam a programação de seus algoritmos. Através deste processo, os alunos aprendem comandos básicos de programação procedural de computadores, programação orientada a objetos, herança e polimorfismo. A avaliação, realizada por 83 participantes sorteados aleatoriamente no fórum online da comunidade Robocode foi vaga, já que relataram apenas que os alunos pareceram gostar de jogar. Desde 2005 o jogo foi portado para a SourceForge [57], com código aberto e vem tendo atualizações. Existe, inclusive, uma liga brasileira que organiza torneios;

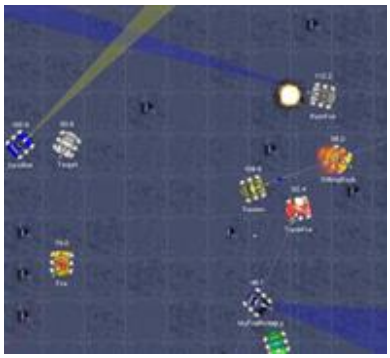

Figura 2: Imagem de uma partida de *Robocode*.

B. **Kernel Panic** (2010) [58] usa metáforas da ciência da computação (*i.e.* bits e ponteiros) como objetos gráficos que são controlados pelo jogador. É um RTS (jogo de estratégia em tempo real) simplificado com as seguintes características: não há gerenciamento de recursos, exceto tempo e espaço; todas as unidades são livres para criar; tem uma pequena árvore de melhoria tecnológica com menos de 10 unidades; e usa gráficos vetoriais de baixo custo que combinam com o universo. O jogo em si, possui características que o tornam mais focado no desenvolvimento do pensamento computacional do que

na programação. É um jogo, com gráficos rudimentares, que funciona como metáfora para o que ocorre quando programamos. Porém, o artigo não deixa claro a capacidade de até que ponto ele seria capaz de ensinar programação por si só;

C. **Prog&Play** (2011) [58] é um jogo com propósito adaptável para diversas escolhas de aprendizado (imperativa, orientada a objeto, funcional, gráfica). Ele é um RTS que permite interações entre os usuários. Seu multiplayer foi desenvolvido usando Kernel Panic como base. Usado de maneira educacional, o jogo permite que alunos programem seus personagens e tentem formar alianças para vencer. Os estudantes podem selecionar o idioma com o qual desejam programar, dentre opções

D. **Marvin's Arena** (2009) [58] é um jogo de programação gratuito com visual muito semelhante ao do Robocode. Nota-se que houve bastante inspiração neste inclusive, porém o grande diferencial é a opção de jogá-lo em 3D. Ele é projetado para tentar ensinar de maneira fácil a programação em qualquer linguagem compatível com o .NET. É adequado para novos programadores que querem ganhar experiência com esse tipo de linguagem. No jogo, o aluno cria um robô e o deixa lutar contra um ou mais oponentes em um torneio. O primeiro robô pode ser escrito em apenas alguns minutos, porém robôs complexos podem levar mais tempo. Desenvolvedores mais experientes podem, inclusive, criar robôs que trabalhem em equipe para lutar contra outras equipes. Os robôs podem se mover, procurar por oponentes e disparar balas e foguetes para sobreviver na batalha;

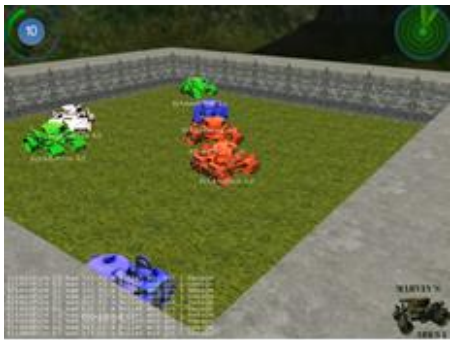

Figura 3: O visual 3D em *Marvin's Arena*.

E. **Gun-Tactyx** (2008) [63] como Marvin's Arena, porém usando SMALL [64], uma linguagem simples de script para ensinar conceitos básicos de programação;

F. **Robot Battle** (2002) [58] segue a mesma linha dos dois anteriores, porém com uma linguagem de script específica. Tanto este quanto os últimos dois jogos são muito parecidos em termos de proposta e remetem ao Robocode. Seria interessante se existisse a possibilidade do usuário escolher o tipo de linguagem que gostaria de praticar;

G. **Colobot** (2007) [58], ou *Colonize with Bots* é um jogo de estratégia em tempo real com gráficos em 3D. O objetivo do jogo é encontrar um planeta para colonizar. Originalmente lançado como um jogo comercial desenvolvido especificamente para ensinar programação de computadores, atualmente é possível encontrar a versão gratuita dele com o nome de Colobot: Gold Edition [65]. Os jogadores comandam diferentes veículos escrevendo pseudocódigos numa linguagem de programação específica do jogo (semelhante a C ++) como Ada [59], C [60], Java [61], OCaml [62], Scratch [49]. Os conteúdos de aprendizagem e a usabilidade não estão incluídos no jogo e devem ser adicionados pelos professores. Com relação à avaliação do jogo, os resultados iniciais indicam que a maioria dos estudantes o achou motivacional, embora alguns professores que participaram de seu estudo tenham relatado percepções negativas, porque acharam que ele poderia representar erroneamente a ciência da computação como sendo apenas feita de videogames [58]. A vantagem em relação ao Kernel Panic, no qual foi baseado, é que se pode utilizar uma variedade maior de linguagens de programação;

para completar várias tarefas. Apesar de apresentar um jogo interativo, ele não pode ser modificado de acordo com um currículo específico e não é gratuito, o que dificulta sua adoção;

H. **Catacombs** (2007) [4] é um jogo de RPG cujo objetivo é familiarizar os alunos com a definição de variáveis e uso de condicionais e loops. O jogador se torna um mago e responde a perguntas a fim de preencher um código corretamente, utilizando uma mini-linguagem especificamente projetada para o jogo. A ideia do jogo é boa, e realmente parece inserir os alunos no universo da lógica de programação com uma pseudolinguagem de script. Porém, a avaliação do jogo pelos alunos parece insuficiente, já que apenas dois grupos de alunos responderam, ambos grupos pequenos, com 13 e 8 pessoas. Não obstante, eles julgaram a experiência divertida e interessante [4]. Todos que responderam já tinham conhecimentos básicos em ciência da computação;

I. **Saving Sera** (2007) [4] tem um viés educacional onde o jogador tenta salvar uma princesa chamada Sera, que foi raptada pelo monstro Gargamel em seu 16º aniversário. Os jogadores são obrigados a passar por várias missões, seja preenchendo as unidades de código adequadas em exercícios específicos ou classificando os segmentos de código em seu devido lugar para que possam avançar para a próxima etapa. Essas atividades permitem que os alunos aprendam e pratiquem recursão e estruturas condicionais. O jogo é dos mesmos autores de Catacombs, e surgiu de um artigo [4] cuja ideia era criar jogos para aumentar a retenção na disciplina de introdução à programação. A avaliação do jogo foi realizada pelos mesmo grupos de alunos do jogo anterior e, da mesma forma, nos pareceu insuficiente. A maioria achou o jogo divertido e educacional;

J. **Elemental: The Recurrence** (2009) [9] tem, inicialmente, uma breve demonstração do que é recursão para introduzir os alunos à teoria. Posteriormente, os estudantes precisam aplicar o algoritmo de busca em profundidade, movendo o protagonista através de uma árvore binária e completando três missões. É um jogo tridimensional que ajuda os alunos a aprender recursão e o algoritmo de busca em profundidade, utilizando a linguagem de programação C#. Barnes novamente apresenta uma alternativa de jogo com o propósito de ensinar programação, a qual a maioria dos 43 alunos que participaram da pesquisa não só gostaram como se sentiram motivados a querer aprender conceitos de programação ainda mais complexos. É importante ressaltar que todos estavam cursando ou concluíram o curso *Estruturas de Dados e Algoritmos* [9]. Esse jogo

tem um foco bem específico, já que ensina recursão e busca em profundidade usando C#, portanto não é tão introdutório quanto os anteriores. Elementos típicos de jogos não foram plenamente desenvolvidos, como

K. **Wireless Intelligent agent Simulation Environment – WISE** (2004) [66] - ele combina atividades das versões virtual e física do jogo *Wumpus World*, que pode ser jogado cooperativamente ou competitivamente. Também habilita agentes distribuídos fisicamente a jogar um jogo interativo e provê um ambiente dinâmico de aprendizagem que pode melhorar diversos cursos de computação. Ele pode ser usado como um meio de demonstrar técnicas durante as aulas, e os estudantes podem realizar trabalhos práticos que testem, expandam, ou modifiquem o simulador. Uma adaptação de uma ideia antiga, porém o que foi apresentado é apenas algo que pode ser usado, mas sem nenhum resultado concreto de uso para a melhoria do aprendizado de competências computacionais de programação em sala de aula. A versão física ainda requer um grande número de recursos, como espaço, robôs e assim por diante;

L. **CMX** (2016) [25] é um jogo de MMORPG projetado e desenvolvido para melhorar o aprendizado e ensino de programação. A ideia dos desenvolvedores é que ele seja usado em sala de aula como uma ferramenta de apoio [67]. O principal ambiente do CMX replica uma fábrica que polui o ecossistema com lixo tóxico, e o último pedaço de terra que resta também está sob o risco de contaminação. Para isso, uma equipe de indivíduos chamados *crackers* são ativistas que tentam invadir a fábrica para desligar seu servidor principal e, assim, impedir que ele polua ainda mais o ambiente. No entanto, a fábrica está equipada com funcionários chamados *hackers*, que são pagos para proteger o servidor e a operação da fábrica em andamento. Em cima desse cenário, os jogadores são treinados por personagens chamados Senseis que desbloqueiam novos desafios e controlam o fluxo do jogo. Esse foi o jogo mais atual que encontramos. Entretanto, a linguagem utilizada nessa versão apresentada no artigo é o C. Apesar dessa aparente limitação, ele permite feedbacks, recompensas e pontuação. Pareceu pouco sofisticado, mas, ainda assim, a experiência é válida por ser um jogo que ensina um tópico específico que não é exatamente trivial;
customizações, inserindo outras possibilidades de linguagem ou mesmo de perguntas feitas pelos Senseis. Porém, analisando atentamente, ele parece ser mais um quiz elaborado do que propriamente um jogo que entretém, embora os autores do artigo tenham apontado o contrário em sua pesquisa com os 76 alunos que o testaram [25];

M. **Ruby Warrior** (2013) [68]. O jogador controla um guerreiro que precisa chegar ao final da fase. Esse controle é feito através de instruções sequenciais em script na linguagem Ruby [69]. É um jogo web que deveria servir para treinar o jogador na linguagem Ruby, porém se mostrou bem superficial, por se manter restrito aos conceitos básicos da linguagem. Tem efeito maior como divulgação da empresa, que desenvolve um bootcamp para quem quer se tornar programador, do que objetivo educacional. Além disso, ele possui um incômodo, já que para continuar jogando, depois da segunda fase, é necessário estar conectado com sua conta do Facebook.

3.2.3 Análise comparativa

As duas subseções apresentadas totalizam os 26 jogos encontrados em nosso levantamento, porém, como pudemos notar, muitos deles eram bem parecidos, seja na estética, mecânica ou dinâmica.

Indo dos jogos mais elaborados aos mais simples, apresentamos a análise de cada um deles, ainda que com limitações por conta da dificuldade em acharmos maiores informações sobre muitos, conforme havíamos mencionado anteriormente.

As Tabelas 1 e 2, apresentadas a seguir, confrontam outras características de cada jogo e nos permite visualizar de forma comparativa. Cada coluna verifica um atributo ou faz uma pergunta relativa ao jogo em questão. Em relação à coluna Linguagem Ensinada, algumas são pseudo-linguagens criadas especificamente para o jogo. Nesse caso, identificamos como linguagem proprietária.

Tabela 1: Comparativo dos Jogos Encontrados para Crianças e Adolescentes

Jogo	Artigo analisa o jogo	Para escola ou aluno	Em rede ou Local	Ensina ou apoia	Existe em 2018	Gratuito, Comercial, Código aberto	Linguagem ensinada	Bem acabado
A	Sim	Aluno	Local	Apoia	Não	Gratuito	Proprietária	Sim
B	Sim	Aluno	Local	Apoia	Não	Gratuito	Blocos	Sim
C	Sim	Aluno	Local	Apoia	Não	Gratuito	-	Não
D	Não	Escola	Local	Apoia	Não	Gratuito	Proprietária	Não
E	-	Aluno	Local	Apoia	Sim	Gratuito	Blocos	Sim
F	-	Aluno	Local	Apoia	Sim	Comercial	Blocos	Sim
G	-	Escola/Aluno	Rede	Ensina	Sim	Gratuito	JS/Python	Sim
H	Sim	Aluno	Local	Ensina	Não	Gratuito	Java	Não
I	-	Aluno	Rede	Apoia	Sim	Gratuito	Blocos	Sim
J	-	Aluno	Local	Apoia	Sim	Comercial	Blocos	Sim
K	-	Aluno	Local	Apoia	Sim	Comercial	Blocos	Sim
L	-	Aluno	Local	Apoia	Sim	Comercial	Blocos	Sim
M	-	Aluno	Local	Apoia	Sim	Gratuito	Blocos	Sim

Tabela 2: Comparativo dos Jogos Encontrados para Universitários

Jogo	Artigo analisa o jogo	Para escola ou aluno	Em rede ou Local	Ensina ou apoia	Existe em 2018	Gratuito, Comercial, Código aberto	Linguagem ensinada	Bem acabado
A	Sim	Aluno	Local	Apoia	Sim	Cód. aberto	Java	Sim
B	Sim	Escola	Rede	Apoia	Sim	Cód. aberto	Várias	Não
C	Sim	Escola	Rede	Apoia	Não	Cód. aberto	Várias	Não
D	Não	Aluno	Local	Apoia	Sim	Gratuito	.NET	Não
E	Não	Aluno	Local	Apoia	Sim	Gratuito	SMALL	Não
F	Não	Aluno	Local	Apoia	Sim	Gratuito	Proprietária	Não
G	Não	Aluno	Local	Apoia	Sim	Cód. aberto	Proprietária	Sim
H	Sim	Escola	Local	Ensina	Não	Gratuito	Proprietária	Sim
I	Sim	Escola	Local	Ensina	Não	Gratuito	Proprietária	Não
J	Sim	Escola	Local	Ensina	Não	Gratuito	C#	Não
K	Não	Escola	Rede	Apoia	Não	Gratuito	Natural	Não
L	Sim	Escola	Rede	Apoia	Sim	Cód. aberto	Várias	Sim
M	-	Aluno	Rede	Ensina	Sim	Gratuito	Ruby	Sim

A partir da consolidação dos dados nas tabelas podemos perceber que 61% dos jogos voltados para crianças e adolescentes usam mais linguagens visuais de blocos, o que era esperado, uma vez que essa abordagem ajuda na construção da lógica computacional. Também é possível observar que 74% são jogos que rodam localmente, o que neutraliza o componente de engajamento social.

Somente 23% dos jogos analisados se preocupam, de fato, com o ensino da linguagem. A maioria dos jogos se limitam a apoiar o aprendizado. Isso mostra que é desejável o acompanhamento de perto do aprendizado por um professor, com conhecimentos da linguagem de programação.

Aproximadamente 85% dos jogos analisados são gratuitos ou de código aberto, o que acreditamos que pode facilitar a adoção por parte dos alunos ou professores. Com isso, seguimos para a última seção deste artigo em que expomos as considerações finais de nossa pesquisa.

4 Conclusão

Conseguimos encontrar 26 jogos com o propósito de ensinar programação. Apesar de nosso intuito original ter tido como foco alunos universitários, notamos a importância de considerar também os jogos que desenvolviam o pensamento computacional em estudantes de faixas etárias menores. Por esse motivo, os jogos foram classificados em duas categorias.

De maneira geral, a maior parte dos jogos educativos encontrados apresentaram soluções válidas que podem aumentar o desempenho de quem quer aprender a programar. No entanto, eles se concentram em unidades específicas de aprendizado e suportam atividades limitadas de programação.

Vale ressaltar que também encontramos evidências de que os jogos desenvolvidos com os processos atuais podem não contribuir para o desenvolvimento de pensamento lógico, cognitivo e social dos alunos [70]. Isso acontece, principalmente, porque as ferramentas de ensino encontradas, por si só, não são capazes de engajar os alunos por um longo período. A partir deste ponto, consideramos que um trabalho futuro seria identificar como os jogos poderiam atingir esse objetivo de maneira mensurável.

Como muitos autores constataram [71], diversos programadores iniciantes não persistem diante de desafios. Foi verificado que os novatos querem evitar a complexidade, enquanto os especialistas querem gerenciá-la [72]. Como resultado disso, o novato é menos propenso a fazer mais de uma tentativa na hora de depurar seu programa. O que reforça a necessidade de jogos com o propósito de desenvolver esse tipo de resiliência nos jovens desenvolvedores, porém nenhum dos trabalhos que encontramos demonstrou ter.

Nossa busca, inclusive, chegou a nos revelar uma metodologia para a criação de jogos educacionais para o ensino de computação, o ENgAGED [73], que integra o design instrucional e de jogos, buscando equilíbrio entres ambos os aspectos. Contudo, não quisemos nos aprofundar nessa metodologia, uma vez que não era o foco de nossa pesquisa atual, embora achemos relevante mencioná-lo nessa seção, já que pode ser um excelente ponto de partida para nossos trabalhos futuros.

Por fim, é notável que pouco foi feito nos últimos vinte anos com o intuito de envolver o aluno universitário no ensino de programação, uma disciplina que possui tantas desistências e resistências. Esse cenário é, no mínimo, curioso e surpreendente, pois a maioria dos alunos gosta de jogos digitais e poderia se beneficiar de jogos educativos com esse enfoque.

Destacamos e reiteramos que nosso objetivo futuro é aprofundar essa busca e tentar desenvolver, com base nela, não apenas um jogo que apresente os elementos necessários para a criação das competências utilizadas em programação, como possivelmente uma metodologia que facilite o desenvolvimento desse tipo de jogo.

Agradecimentos

Os autores gostariam de agradecer o apoio da CAPES e do CNPq.

Referências

[1] C. Malliarakis, M. Satratzemi, S. Xinogalos. Educational Games for Teaching Computer Programming. In C. Karagiannidis, P. Politis, I. Karasavvidis, editors, *Research on e-Learning and ICT in Education*, Springer, New York, NY, 2014.

[2] E. Lahtinen, K. Ala-Mutka, H. Jarvinen. A Study of Difficulties of Novice Programmers. In: *Innovation and Technology in Computer Science Education*, pages 14–18, 2005.

[3] R. Moser. A fantasy adventure game as a learning environment: why learning to program is so difficult and what can be done about it. In *ACM SIGCSE Bulletin*, volume 29, number 3, pages 114-116. ACM, 1997.

[4] T. Barnes, E. Powell, A. Chaffin, A. Godwin, H. Richter. Game2Learn: Building CS1 Learning Games for Retention. In *Proceedings of the 12th SIGCSE Conference on Innovation and Technology in Computer Science Education*. Dundee, Scotland: pages 121– 125, June 2007.

[5] A. Gomes and A.J.N. Mendes. Learning to program-difficulties and solutions. In *Proceedings of the International Conference on Engineering Education*. Coimbra, Portugal, September 2007.

[6] K. M.Y. Law, V.C.S. Lee, Y.T. Yu. Learning motivation in e-learning facilitated computer programming courses. In *Computers & Education*, volume 55, issue 1, pages 218-228, 2010.

[7] G.A. Gunter, R.F. Kenny, E.H. Vick. Taking educational games seriously: using the RETAIN model to design endogenous fantasy into standalone educational games. In *Educational Technology Research and Development*, 56(5/6), pages 511-537, 2008.

[8] T. Barnes, H. Richter, A. Chaffin, A. Godwin, E. Powell, T. Ralph, P. Matthews, and H. Jordan. The role of feedback in Game2Learn. In *CHI*, volume 2007, pages 1-5, 2007.

[9] A. Chaffin, K. Doran, D. Hicks, T. Barnes. Experimental evaluation of teaching recursion in a video game. In *Proceedings of the 2009 ACM SIGGRAPH Symposium on Video Games*, 2009.

[10] A. Yusoff, R. Crowder, L. Gilbert, G. Wills. A conceptual framework for serious games. In *Advanced Learning Technologies*. ICALT 2009. Ninth IEEE International Conference on, pages 21-23. IEEE, 2009.

[11] J. O'Kelly, J. Paul Gibson. RoboCode & problem-based learning: a non-prescriptive approach to teaching programming. In *ACM SIGCSE Bulletin 38*, number 3, pages 217-221, 2006.

[12] S. de Freitas, S. Jarvis. A framework for developing serious games to meet learner needs. In *Interservice/Industry Training, Simulation and Education Conference.* Florida. December 2006.

[13] T. Mitamura, Y. Suzuki, T. Oohori. Serious games for learning programming languages. In *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Seoul, pages 1812-1817. 2012.

[14] P. Brereton, B.A. Kitchenham, D. Budgen, M. Turner, M. Khalil. Lessons from applying the systematic literature review process within the software engineering domain. In *Journal of Systems and Software*, volume 80, number 4, pages 571–583. April 2007.

[15] A. Vahldick, A. J. Mendes, M. J. Marcelino. A review of games designed to improve introductory computer programming competencies. In *IEEE Frontiers in Education Conference (FIE) Proceedings*, Madrid, pages 1-7. 2014.

[16] M. Piteira, S. R Haddad. Innovate in your program computer class: An approach based on a serious game. In *ACM International Conference Proceeding Series*, pages 49–54. 2011.

[17] S. Stamm. Mixed nuts: atypical classroom techniques for computer science courses. In *Crossroads*, volume 10, number 4, page 3. 2004.

[18] M. Corney, D. Teague, R. N. Thomas. Engaging students in programming. In *Conferences in Research and Practice in Information Technology Series*, volume 103, pages 63–72. 2010.

[19] R. Ibrahim, A. Jaafar. Using educational games in learning introductory programming: A pilot study on students' perceptions. In *Proceedings of International Symposium on Information Technology - Visual Informatics*, ITSim'10, volume 1. 2010.

[20] J. Shi, S. White. Work-in-progress: Learning to program in a connected world. In *Proceedings of Learning and Teaching in Computing and Engineering*, LaTiCE 2013, pages 229–232. 2013.

[21] D.M. Souza, M.H. da S. Batista, E.F. Barbosa. Problems and Weaknesses in the Teaching and Learning of Programming: A Mapping Review. In *Brazilian Journal of Computers in Education*, [S.l.], page 39. April 2016. ISSN 2317-6121.

[22] C.-C. Liu, Y.-B. Cheng, C.-W Huang. The effect of simulation games on the learning of computational problem solving. In *Computers & Education*, volume 57(3), pages 1907-1918. 2011.

[23] J. Bennedsen, M.E. Carpersen. Exposing the Programming Process. In J. Bennedsen, M.E. Carpersen, M. Kolling, editors, *Reflection on the Theory of Programming: Methods and Implementation*, Springer, Verlag, pages 6-16. 2008.

[24] C. Kazimoglu, M. Kiernan, L. Bacon, L. Mackinnon. Developing a game model for computational thinking and learning traditional programming through game-play. In *Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*. 2010.

[25] C. Malliarakis, M. Satratzemi, S. Xinogalos. CMX: The Effects of an Educational MMORPG on Learning and Teaching Computer Programming. In *IEEE Transactions on Learning Technologies*, volume 10, number 2, pages 219-235. April-June 1 2017.

[26] J. Murray, I. Bogost, M. Mataes, M. Nitsche. Game Design Education: Integrating Computation and Culture. In *IEEE Computer*, volume 39, number 2, pages 3-51. 2006.

[27] M. Zyda. Educating the Next Generation of Game Developers. In *IEEE Computer*, volume 39, number 2, pages 30-35. 2006.

[28] L. Argent, B. Depper, R. Fajardo, S. Ghertson, S. Leutenegger, M. Lopez, J. Rutenbeck. Building a Game Development Program. In *IEEE Computer*, volume 39, number 2, pages 52-61. 2006.

[29] T. Fullerton. Play-Centric Games Education. In *IEEE Computer*, volume 39, number 2, pages 36-42. 2006.

[30] I. Horswill, M. Noval. Evolving the Artist-Technologist. In *IEEE Computer*, volume 39, number 2, pages 53-62. 2006.

[31] A. Ater-Kranov, R. Bryant, G. Orr, S. Wallace, M. Zhang. Developing a community definition and teaching modules for computational thinking: accomplishments and challenges. In *Proceedings of the 2010 ACM conference on Information technology education*. 2010.

[32] M. Zyda. From visual simulation to virtual reality to games. In *Computer*, volume 38, number 9, pages 25–32. 2005.

[33] M. Muratet, P. Torguet, J. Jessel, F. Viallet. Towards a serious game to help students learn computer programming. In *International Journal of Computer Games Technology*, article 3, 12 pages. January 2009.

[34] T. W. Malone. What makes things fun to learn? Heuristics for designing instructional computer games. In *Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on Small systems*, pages 162-169. ACM SIGSMALL, New York, NY, USA, 1980.

[35] Wing, J.. Computational thinking, Communications of the ACM, v.49 n.3, March 2006.

[36] C. Kazimoglu, M. Kiernan, L. Bacon, L. Mackinnon. A Serious Game for Developing Computational Thinking and Learning Introductory Computer Programming. In *Procedia - Social and Behavioral Sciences*, volume 47, pages 1991–1999. January 2012.

[37] D. Parsons, P. Haden. Parson's programming puzzles: a fun and effective learning tool for first programming courses. In *Proceedings of the 8th Australasian Conference on Computing Education* (Hobart, Australia, January 16-19, 2006), pages 157-163. January, 2006.

[38] D.A. Norman, J.C. Spohrer. Learner-centered education. In *Communications of the ACM*, volume 39, number 4, pages 24-27. April 1996.

[39] M. Flanagan, D.C. Howe, H. Nissenbaum. Values at play: Design tradeoffs in socially-oriented game design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Portland), pages 751-760. ACM Press, 2005.

[40] E.R. Hayes, I.A. Games. Making computer games and design thinking. In *Games & Culture*, volume 3, pages 309–332, 2008.

[41] M. Flanagan. Troubling "games for girls": Notes from the edge of game design. In *Unpublished proceedings of Digital Games Research Association* (Vancouver, British Columbia, Canada). 2005.

[42] M. Flanagan, D.C. Howe, H. Nissenbaum. New design methods for activist gaming. In *Unpublished proceedings of Digital Games Research Association* (Vancouver, British Columbia, Canada). 2005.

[43] Cargo-Bot: http://twolivesleft.com/CargoBot. Acessado em 12 de Maio de 2018.

[44] Catos Hike: http://hwahba.com/catoshike/. Acessado em 12 de Maio de 2018.

[45] Code Combat: http://codecombat.com/. Acessado em 12 de Maio de 2018.

[46] J.K.-W. Chang, L.H. Dang, J. Pavleas, J.F. McCarthy, K. Sung, J. Bay. Experience with Dream Coders: developing a 2D RPG for teaching introductory programming concepts. In *Journal of Computing Sciences in Colleges*, volume 28, number 1, pages 227–236. 2012.

[47] Light-Bot 2: http://armorgames.com/play/6061/light-bot-20. Acessado em 12 de Maio de 2018.

[48] Machinist-Fabrique: http://learntocode.biz/. Acessado em 12 de Maio de 2018.

[49] Scratch: https://scratch.mit.edu. Acessado em 27 de Julho de 2018.

[50] Move the Turtle: http://www.geekkids.me. Acessado em 12 de Maio de 2018.

[51] Logo: http://el.media.mit.edu/logo-foundation/what_is_logo/logo_programming.html. Acessado em 27 de Julho de 2018.

[52] Angry Birds: https://www.angrybirds.com/games. Acessado em 25 de Julho de 2018.

[53] Cut the Rope: https://www.cuttherope.net. Acessado em 25 de Julho de 2018.

[54] Robo Logic: https://www.digitalsirup.com/app/robologic/?_lang=en. Acessado em 12 de Maio de 2018.

[55] RoboZZle: http://robozzle.com. Acessado em 12 de Maio de 2018.

[56] J. O'Kelly, J.P. Gibson. RoboCode & problem-based learning: a non-prescriptive approach to teaching programming. In *ACM SIGCSE Bulletin 38*, number 3, pages 217-221. 2006.

[57] Robocode: http://robocode.sourceforge.net. Acessado em 25 de Julho de 2018.

[58] M. Muratet, P. Torguet, F. Viallet, J. Jessel, Experimental feedback on Prog&Play: a serious game for programming practice. In *Computer Graphics Forum*, volume 30, pages 61-73. 2011.

[59] Ada: https://www.adacore.com/about-ada. Acessado em 27 de Julho de 2018.

[60] Linguagem C: https://www.geeksforgeeks.org/c-language-set-1-introduction/. Acessado em 27 de Julho de 2018.

[61] Java: https://docs.oracle.com/javase/8/docs/technotes/guides/language/index.html. Acessado em 27 de Julho de 2018.

[62] OCaml: https://ocaml.org. Acessado em 27 de Julho de 2018.

[63] Gun-Tactyx: http://apocalyx.sourceforge.net/guntactyx/. Acessado em 27 de Julho de 2018.

[64] Small: http://www.csci.csusb.edu/dick/samples/small.html. Acessado em 27 de Julho de 2018.

[65] Colobot: Gold Edition: https://colobot.info. Acessado em 25 de Julho de 2018.

[66] D.J. Cook, M. Huber, R. Yerraballi, L.B. Holder. Enhancing computer science education with a wireless intelligent simulation environment. In *Journal of Computing in Higher Education*, volume 16, number 1, pages 106–127. 2004.

[67] C. Malliarakis, M. Satratzemi, S. Xinogalos. Towards a new massive multiplayer online role playing game for introductory programming. In *Proceedings of the 6th Balkan Conference in Informatics*, pages 156-163. ACM, 2013.

[68] Ruby Warrior: https://www.bloc.io/ruby-warrior. Acessado em 12 de Maio de 2018.

[69] Ruby: https://www.ruby-lang.org/pt/. Acessado em 27 de Julho de 2018.

[70] L.M. Tabuti, R. Nakamura. Métodos para o Desenvolvimento de Jogos Digitais de Lógica: Uma Revisão Sistemática. In *Anais do Simpósio Brasileiro de Informática na Educação*, volume 26, number 1, page 41. October, 2015.

[71] J. Denner, L. Werner, E. Ortiz. Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts?. In *Computers & Education*, volume 58, number 1, pages 240-249. January 2012.

[72] A. Fluery. Student beliefs about Pascal programming. In *Journal of Educational Computing Research*, volume 9, pages 355–371. 1993.

[73] P.E. Battistella, C.G. von Wangenheim. Engaged: Um processo de desenvolvimento de jogos para ensinar computação. In *Proceedings of Brazilian Symposium on Computers in Education*, volume 27, number 1, page 380. 2016.