ANÁLISE DO PENSAMENTO COMPUTACIONAL NA EDUCAÇÃO VOLTADO À SERIOUS GAMES

ANALYSIS OF COMPUTATIONAL THINKING IN EDUCATION AIMED TO SERIOUS GAMES

Fernanda Moreira POUZA

fernandampouza@hotmail.com
Curso de Sistemas de Informação, Centro Universitário Padre Anchieta

Carlos Eduardo CÂMARA

ccamara@anchieta.br
Ciência da Computação, Sistemas de Informação, Centro Universitário Padre Anchieta

Resumo

Este trabalho tem por objetivo analisar a importância da construção do Pensamento Computacional aplicado na educação, propor uma nova metodologia baseada em *serious games* e apresentar, por meio de um protótipo, uma melhor alternativa para um desenvolvimento cognitivo mais favorável em alguns anos. Para verificar a eficiência do método deste projeto, foram analisados diversos artigos acerca do desenvolvimento do Pensamento Computacional, a utilização de *Serious Games* na área da educação, os conceitos de algoritmos e o ambiente de programação Karel. Para validação do protótipo desenvolvido, foram realizados testes de própria autoria e validações de resultados esperados. Com os testes realizados, chegou-se à conclusão que o protótipo é útil para utilização como ferramenta para o aprendizado e desenvolvimento do Pensamento Computacional.

Palavras-Chave

Pensamento computacional; *serious games*; algoritmo; Karel; educação.

Abstract

This work aims to analyze the importance of construction of Computational Thinking applied in education, propose a new methodology based on serious games and present, through a prototype, a better alternative for a more favorable cognitive development in some years. In order to verify the efficiency of the method of this project, several articles about the development of Computational Thinking, the use of Serious Games in education area, concepts of algorithms and the Karel programming environment were analyzed. For validation of the developed prototype, tests of own authorship and validations of expected results were performed. With tests carried out, it was concluded that the prototype is useful for use as a tool for learning and development of Computational Thinking.

Keywords

Computational thinking; serious games; algorithms; Karel; education.

INTRODUÇÃO

Nos últimos anos, o meio tecnológico está modificando e se adaptando à vida das pessoas e, as pessoas tendo de se adaptar a estas novas tecnologias, de uma maneira muito acelerada e acessível.

Segundo uma pesquisa realizada pela *Microsoft* e o *McKinsey & Company's Education Practice*, "*os avanços na tecnologia devem levar a grandes rupturas no mercado de trabalho na medida em que a automação já pode substituir até 50% dos empregos existentes, somente nos EUA*". Isso fará com que até 11.5 milhões de empregos ligados a níveis de baixa escolaridade deixem de ser ocupados por pessoas até 2030, segundo a mesma pesquisa citada anteriormente.

Muito se percebe que a tecnologia está se tornando uma ferramenta intrínseca ao cotidiano das pessoas, fenômeno denominado como computação ubíqua por Weiser (1991).

Visto isso, a maneira com que estamos acostumadas a trabalhar com a tecnologia irá mudar muito rapidamente e será necessário que novas ideias e novas práticas garantam, principalmente, que as crianças de hoje possam desenvolver competências cognitivas que as auxiliem na forma de pensar, aprender, trabalhar, se relacionar, etc.

"Considerando essa presumível relevância do compreender e do saber criar tecnologias digitais em detrimento de apenas consumi-las, é essencial que a educação das novas gerações seja planejada de modo a desenvolver tais habilidades" (SEVERGNINI, 2018).

Atualmente, as técnicas de ensino estão começando a se adequar às tecnologias, mas em alguns anos como deverá ser o ensino para que as pessoas possam se promover e até mesmo se desenvolver?

Para Resnick (2012) a maioria dos jovens possui pouca experiência em projetar e criar com mídias digitais, sentindo-se mais confortáveis em usar a tecnologia ao contrário de criar suas próprias. "*Eles não são verdadeiramente fluentes em tecnologias digitais: é como se eles pudessem ler, mas não escrever*" (RESNICK, 2012).

Deste modo, é possível perceber que em pouco tempo será necessário elevar o nível cognitivo em determinadas áreas para suprir as necessidades que irão exigir maiores competências.

Visto que a automação irá ocupar certos empregos das pessoas, haverá a necessidade de tornar as pessoas mais preparadas e especializadas para lidar com as máquinas.

A partir desse ponto que o presente trabalho está desenvolvido. Apresentar um protótipo baseado em *Serious Games* para aprimorar o desempenho educacional. Com isso, serão definidos alguns conceitos sobre *Serious games*, Pensamento Computacional e técnicas para serem aplicadas no desenvolvimento do protótipo a ser apresentado.

CONCEITOS PRELIMINARES

SERIOUS GAMES

Muito se tem se tem pesquisado e concluído que os jogos digitais estão auxiliando como ferramenta no ensino. Segundo Maia (2017) o aumento do estudo relacionado à S*erious Games*, ou jogos Sérios, pode ser atribuído, em parte, ao crescente interesse do público-alvo em um método alternativo para o aprendizado.

É fato, de acordo com Kafai e Burke (2015), que ainda há um interesse considerável em examinar o potencial educacional que existe em jogar *videogames*.

Pode-se perceber que os jogos têm a capacidade de envolver e cativar a atenção de uma pessoa por longos períodos de tempo, enquanto proporciona alguns outros benefícios, como o desenvolvimento de habilidades de pensamento de alto nível (KENWRIGHT, 2017).

Visto que os jogos podem ser utilizados como uma ferramenta de auxílio para a educação e não um obstáculo, é preciso garantir que essa ferramenta seja utilizada da maneira mais proveitosa, possuindo características educacionais, sem deixar de ser atrativo para o jogador.

"Estes jogos, com propósito e conteúdo específicos, são conhecidos como Serious Games e permitem apresentar novas situações, discutir soluções, construir conhecimentos e treinar atividades particulares" (MACHADO et al., 2011).

Sendo assim, *Serious Games* são os jogos desenvolvidos com o intuito de verificar problemas, analisar a situação e determinar um meio de alcançar uma solução. Ao utilizar os conceitos de *Serious game* como ferramenta para desenvolvimento de um jogo educacional, é possível motivar o aluno a aprender de uma maneira mais descontraída, sem deixar de lado o enfoque principal do aprimoramento do ensino.

O PENSAMENTO COMPUTACIONAL

As disciplinas relacionadas à computação estão presentes no currículo escolar de diversos países, de acordo com Brackmann (2017), onde a introdução dessa ferramenta de ensino ocorre de forma rigorosa.

A implementação dessa disciplina como obrigatória traz benefícios educacionais, onde as habilidades para raciocínio e solução de problemas se mostram muito mais aprimoradas.

Uma pesquisa feita por Balanskat et. al. (2015), apontou que na Europa o ensino de Ciência da Computação já está integrado no currículo da Educação Básica de 15 países, sendo eles: Áustria, Bulgária, República Tcheca, Dinamarca, Estônia, França, Hungria, Irlanda, Lituânia, Malta, Espanha, Polônia, Portugal, Eslováquia e Inglaterra.

No Brasil, o Pensamento Computacional não está definido como parte do currículo das escolas do Ensino Básico (Brackmann, 2017). Os documentos existentes relacionados à tecnologia estão restritos à abordagem de letramento e inclusão digital.

O termo "Pensamento Computacional" começou a ser popularizado no ano de 2006 por meio do artigo "*Computational Thinking*" de Jeannette Wing (2006) e definido de várias maneiras, entre elas, a de que "Pensamento computacional envolve a resolução de problemas, projeção de sistemas, e compreensão do comportamento humano, através da extração de conceitos fundamentais da ciência da computação" (WING, 2006). Ainda segundo Wing (2006), o pensamento computacional é a reformulação de um problema que consideramos difícil em um problema que sabemos como resolver.

Desde então, as definições estão se aperfeiçoando, como a de que o Pensamento Computacional "*É uma abordagem usada para solução de problemas utilizando o que se sabe sobre Computação*" *(GOOGLE FOR EDUCATION, 2015)* e mais atualmente sendo caracterizado por Brackmann (2017) como sendo o Pensamento Computacional:

> *[..] Uma distinta capacidade criativa, crítica e estratégica humana de saber utilizar os fundamentos da Computação, nas mais diversas áreas do conhecimento, com a finalidade de identificar e resolver problemas, de maneira individual ou colaborativa, através de passos claros, de tal forma que uma pessoa ou uma máquina possam executá-los eficazmente.*

O Pensamento Computacional não é uma habilidade que pode ser ligada somente às pessoas que estão relacionadas ao desenvolvimento da tecnologia (BOUCINHA, 2017). O processo de resolução de problemas utilizados em áreas como Ciência da Computação pode ser generalizado e aplicado para diferentes áreas de pesquisa e até mesmo para a vida cotidiana (WING, 2006) e, como todos sabemos já está atuando em todas as áreas do conhecimento, como uma disciplina básica.

A grande ideia para a resolução de qualquer problema dentro do Pensamento Computacional se baseia, principalmente, na abstração e decomposição de uma tarefa grande ou complexa (WING, 2006).

Um estudo desenvolvido pela Code.Org (2016), Liukas (2015) e *BBC Learning* (2015), com elementos citados por Grover e Pea (2013) e o guia da *Computer at School* (CSIZMADIA et al., 2015), chegaram aos "Quatro Pilares do Pensamento Computacional": Decomposição, Reconhecimento de Padrões, Abstração e Algoritmos. (Brackman, 2017 Grover, 2013))

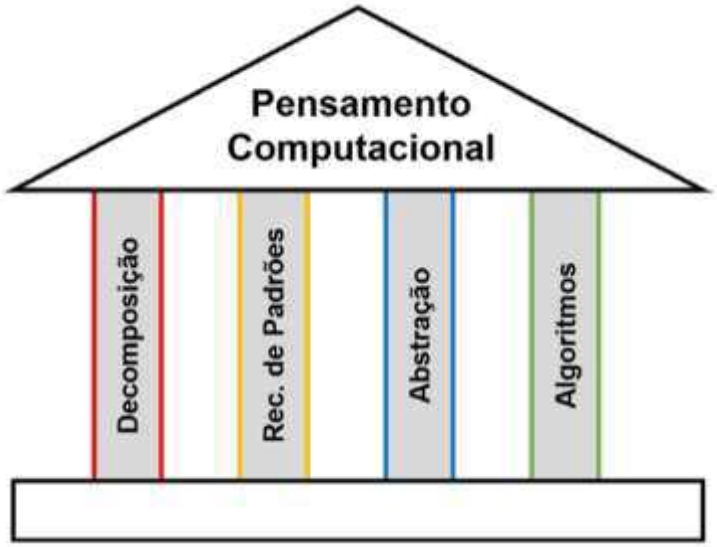
No seguinte tópico são abordados os conceitos básicos dos Pilares citados anteriormente.

QUATRO PILARES DO PENSAMENTO COMPUTACIONAL

Brackmann (2017) cita o Pensamento Computacional como uma forma de identificar um problema complexo e dividi-lo em pedaços, ou seja, fazer sua decomposição. Cada pedaço pode ser analisado individualmente com maior profundidade, verificando os problemas que já foram vistos anteriormente e, assim, poder fazer o reconhecimento de padrões. Os detalhes mais importantes devem ter um enfoque maior, enquanto as informações irrelevantes devem ser descartadas, proporcionando a abstração das informações. Por último, uma sequência de passos pode ser montada para levar a uma resolução à cada um dos subproblemas verificados, tendo assim, um algoritmo.

Os pilares que formam a base do Pensamento Computacional podem ser observados na figura a seguir.

Figura 1. Os Quatro Pilares do Pensamento Computacional



Decomposição

O Dicionário Aurélio nos dá as seguintes definições para a palavra decomposição: "*Examinar por partes. Separar ou separar-se (um corpo ou um conjunto) em seus elementos ou partes; [...]*" (Aurélio).

O Dicionário Michaelis também nos dá os seguintes significados para a mesma palavra: "*Ação, processo ou efeito de decompor. Separação de um todo em suas partes constitutivas.*" (Michaelis).

No caso do Pensamento Computacional, a decomposição é o processo onde os problemas são separados em partes menores, para uma melhor compreensão ao serem examinados separadamente.

É possível citar, como fez Brackmann (2017), a ligação que a decomposição tem com os programadores, que utilizam essa técnica para dividir um problema em instruções, baseadas em funções, métodos, objetos que compõem um algoritmo e seu consequente, código-fonte. Quando essa técnica é aplicada no desenvolvimento de programas, o entendimento e a correção de possíveis erros no código acabam se tornando um processo mais fácil e rápido, já que o erro pode se encontrar em apenas um modulo do projeto. Isso não seria possível caso a técnica de decomposição não fosse aplicada, pois seria necessário alterar várias linhas diferentes de códigos pelo projeto.

Reconhecimento de Padrões

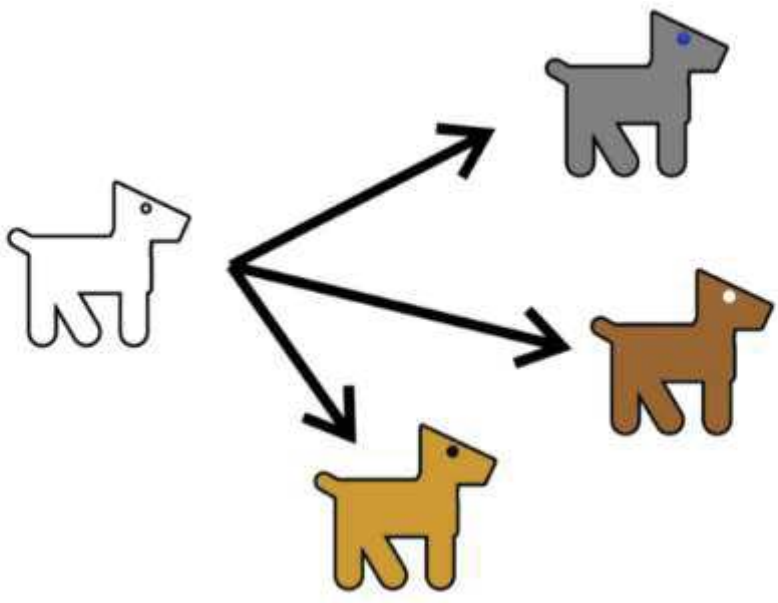
O Reconhecimento de Padrões se dá a partir da finalização da etapa de decomposição. Ao quebrar o problema em partes menores é possível verificar a existência de padrões. Segundo Brackmann (2017), é uma forma de resolver problemas rapidamente, fazendo uso de soluções previamente definidas em outros problemas e com base em experiências anteriores.

Visto isso, é possível perceber que pelo reconhecimento de padrões, é possível tornar uma solução de um problema mais fácil e aplicá-la a problemas semelhantes.

Um exemplo que pode ser utilizado para ilustrar como se dá o reconhecimento de padrões é pelas características de um cachorro. Pode-se identificar que os cachorros possuem características semelhantes, algumas delas sendo os olhos, pelagem, rabo e orelhas.

Identificado um padrão inicial, como dado no exemplo anterior, é visto que caso haja necessidade de identificar um outro cachorro, seria preciso replicar este padrão como um modelo genérico para os padrões que forem encontrados, como é possível verificar na Figura 2.

Figura 2. Reconhecimento de Padrões e Replicação



Abstração

A abstração é uma das bases citadas inicialmente por Wing (2006) como sendo um dos conceitos principais para o Pensamento Computacional.

De acordo com Brackmann (2017), este pilar abrange a filtragem dos dados, onde deve-se ignorar elementos que não sejam necessários e que se possa concentrar a atenção, em dados que sejam mais importantes para desenvolver a ideia do problema a ser resolvido.

A abstração está muito presente na área da computação, onde os profissionais precisam criar abstrações de problemas para desenvolver sistemas. Podemos citar como um exemplo de abstração um algoritmo, onde o processo recebe uma entrada, executa uma sequência de passos e produz uma saída (BRACKMANN, 2017).

Algoritmos

Guimarães e Lages (1994) apresentam a definição de algoritmo como sendo uma "*descrição de um padrão de comportamento, expressado em termo de um repertório bem definido e finito de ações "primitivas", das quais damos por certo que elas podem ser executadas*".

Outra definição de algoritmos apontada por Farrer et al. (2008) é dada como uma "*descrição de um conjunto de comandos que, obedecidos, resultam numa sucessão finita de ações*".

Em outras palavras, um algoritmo é um texto contendo um conjunto de instruções, regras ou passos que devem ser executados em determinada ordem para atingir um resultado ou solução de certo problema.

Os algoritmos estão muito presentes no dia a dia das pessoas, porém sem que elas mesmas se deem conta disso. A comparação mais comum é com uma receita de bolo. Uma receita possui, um problema inicial a ser resolvido, a lista dos ingredientes, alguns fixos e outros variáveis, uma sequência de passos a serem seguidos e, ao final, se todos os passos forem seguidos corretamente, haverá um bolo pronto, ou seja, o problema estará resolvido.

A compra de um produto pela Internet, a substituição de uma lâmpada queimada e até mesmo as ações ao acordar podem seguir um algoritmo, conforme o dia da semana é possível tomar a decisão de sair da cama em um determinado horário ou dormir por mais algum tempo.

Entretanto, existem algoritmos muito mais complexos do que uma receita de bolo e que também estão muito presentes no cotidiano das pessoas. É o caso de como as publicações das redes sociais aparecem na linha do tempo. Entretanto, esses modelos de algoritmos não serão tratados no presente trabalho.

KAREL

"Na década de 70, o então estudante Richard E. Pattis decidiu projetar um ambiente para ensinar de uma maneira mais simples os fundamentos das linguagens de

programação. Surgiu então um projeto onde deve-se ensinar um robô a resolver problemas simples. Para o robô, deu-se o nome de Karel, em homenagem ao tcheco Karel Capek, cujo em um dos trabalhos mencionou pela primeira vez a palavra "robot", que posteriormente popularizou-se pelo mundo" (ROBERTS, 2005).

O Karel é um robô que vive em um mundo bem simples, quadriculado, plano e infinito (GRATÉROL; MARÍN, 2006) onde irá executar uma série de instruções, sendo possível direcioná-lo para determinadas tarefas dentro deste mundo.

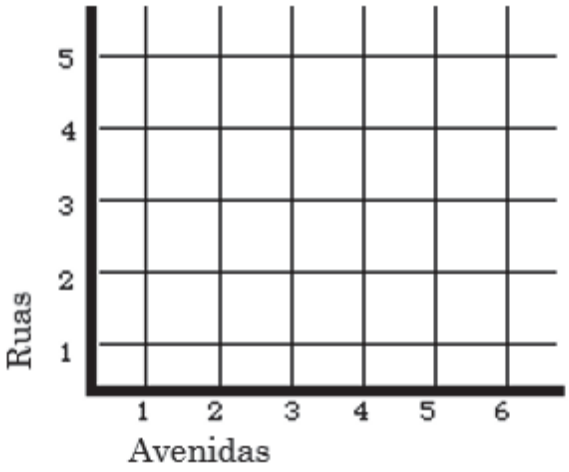
METODOLOGIA E DESENVOLVIMENTO DO PROTÓTIPO

O protótipo a ser desenvolvido se caracteriza por um *serious game* baseado no ambiente de programação chamado Karel, o robô, com o objetivo de aprimorar o processo de aprendizado dos alunos, proporcionando um estudo mais dinâmico a partir dos jogos e motivando-os com técnicas diferenciadas de ensino. Para a construção da proposta do jogo, foram utilizadas como base algumas mecânicas do Mundo de Karel, descritas a seguir.

Quando é definido o processo de especificação desses comandos, inicialmente o robô Karel entende muito pouco desses comandos, portanto, não é indicado, a princípio, que sejam definidos objetivos muito complexos. Porém, conforme se dá o processo de programação, o robô pode ir aprimorando sua capacidade e estender o nível de dificuldade das tarefas a serem executadas a cada novo objetivo que possa surgir.

Os comandos devem ser bem especificados, como será mostrado mais adiante, para que o robô consiga interpretar qual a tarefa a ser cumprida e também para desenvolver os quatro pilares do Pensamento Computacional no aluno.

Figura 3. O mundo de Karel



A interface do mundo de Karel é bem simples, como visto na Figura 3. Relembrando um plano cartesiano, o mundo possui as linhas horizontais como sendo as “Ruas” e as linhas verticais sendo as “Avenidas”. Os pontos onde uma Rua e uma Avenida se cruzam, são chamados de esquina ou nós. Dentro desse mapa o robô irá percorrer os nós para chegar ao objetivo final.

MODO DE JOGO

Tendo como base o mundo de Karel, foi desenvolvido um protótipo de *Serious Game* com o intuito de aprimorar o pensamento computacional, propondo que o jogador entenda os conceitos e execute, por meio dos pilares, os passos para alcançar o objetivo de cada fase. Também foram utilizados como base para este *Serious game*, conceitos de Algoritmos para Busca de Caminho (A*). Esses algoritmos são baseados na ideia de sair de um ponto inicial, fazendo uma estimativa de movimentação ao próximo nó que será mais viável, acumulando o custo real do caminho já percorrido, para alcançar o estado final da maneira mais rápida e, com menor custo.

No jogo em questão, o robô iniciará cada fase em um ponto específico do mapa (0,0) e movimenta-se pelas linhas das ruas e avenidas do mapa, percorrendo o caminho através das intersecções.

O jogador deverá escolher executar dentre quatro comandos disponíveis, tais como: mover para frente, rotacionar para esquerda, rotacionar para a direita ou retornar, para encontrar a solução do problema proposto e chegar ao ponto final desejado. Deve-se cumprir os objetivos para avançar de nível, e assim, avaliar o conteúdo colocado em prática por meio dos resultados obtidos.

Durante a partida, conforme o jogador realizar as movimentações, um painel com cartas irá apresentar informações de jogadas para combinar as movimentações anteriores com as possíveis jogadas que irão suceder, ou seja, a cada movimentação, o painel irá apresentar algumas opções de movimentações que podem ser feitas e assim estabelecer um critério de pontuação onde, caso o jogador faça uma movimentação que deixe o robô mais perto do objetivo, sua pontuação aumente e, caso faça um movimento em que o robô tenha que percorrer mais que o desejado, sua pontuação diminua.

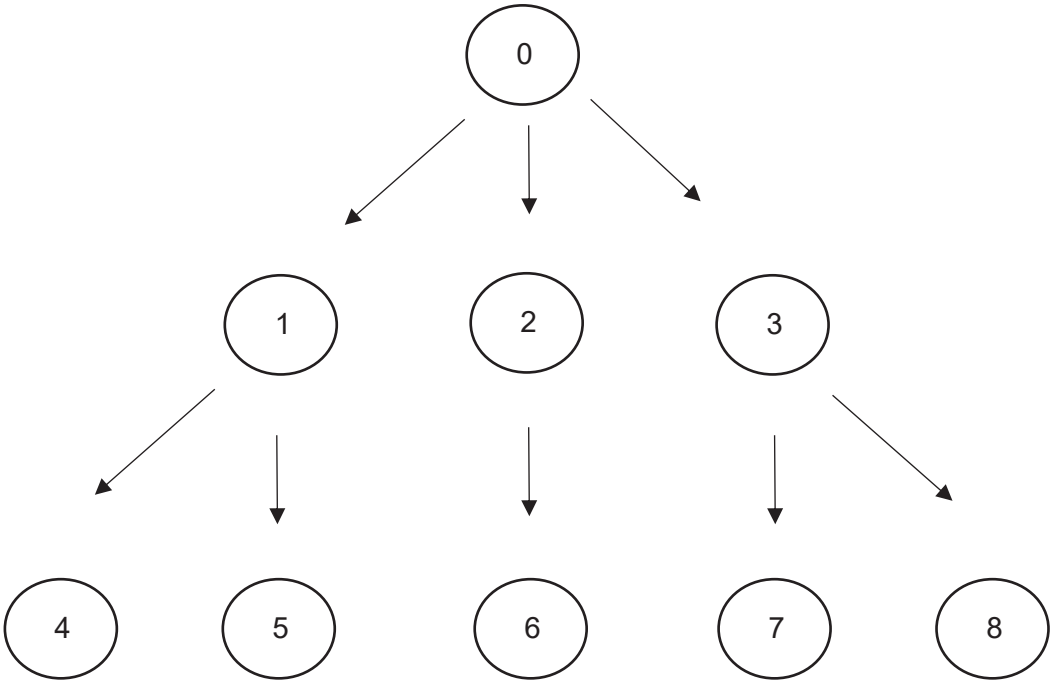
Ao completar cada objetivo, o nível de dificuldade das próximas fases é aprimorado, passando a conter mais números de movimentações do robô, mais elementos no ambiente de jogo e cenários diferentes onde o robô poderá diminuir sua pontuação, ou perder vidas, caso passe por algum elemento que cause danos a sua estrutura. Nessas próximas fases o jogador deverá aplicar novas instruções de algoritmos, como os laços de repetição, comparações, encadeamento de instruções, para construir repetições sequentes de instruções, na busca por atingir um objetivo. Um exemplo apresentado é, enquanto o robô não chegar ao ponto final, o jogador deverá executar mais um movimento.

Também há a possibilidade de determinação de tempo para atingir determinado ponto do objetivo, deixando o jogo com certa dificuldade após algumas fases de jogo.

Uma forma de solucionar o problema é representando o ambiente em forma de um grafo, onde o jogador deve verificar qual será o menor caminho que o robô poderá percorrer a partir do ponto inicial até o ponto final.

Para isso, serão considerados alguns elementos básicos, como o ponto inicial, ponto final, os comandos de direção (avançar, rotacionar para a esquerda, rotacionar para a direita e voltar) a serem utilizados para movimentar o robô pelo percurso.

Figura 4. Modelo de grafo



Observando a figura 4 como exemplo, o jogo se encontra na posição inicial 0 e a partir de então são possíveis 3 tipos de movimentação: avançar, seguir para direita e seguir para a esquerda. Seja qualquer posição que se seguir, serão possíveis outras 4 opções de movimentação, as três anteriores e a opção de retornar ao ponto anterior. Assim, cada movimentação abre uma série de sequências que podem ser definidas para avançar e chegar à posição final.

Ao evoluir de fase, é possível verificar que esse grafo também irá evoluir para outro tipo de grafo, conforme a seguinte imagem.

Figura 5. Modelo de grafo 2

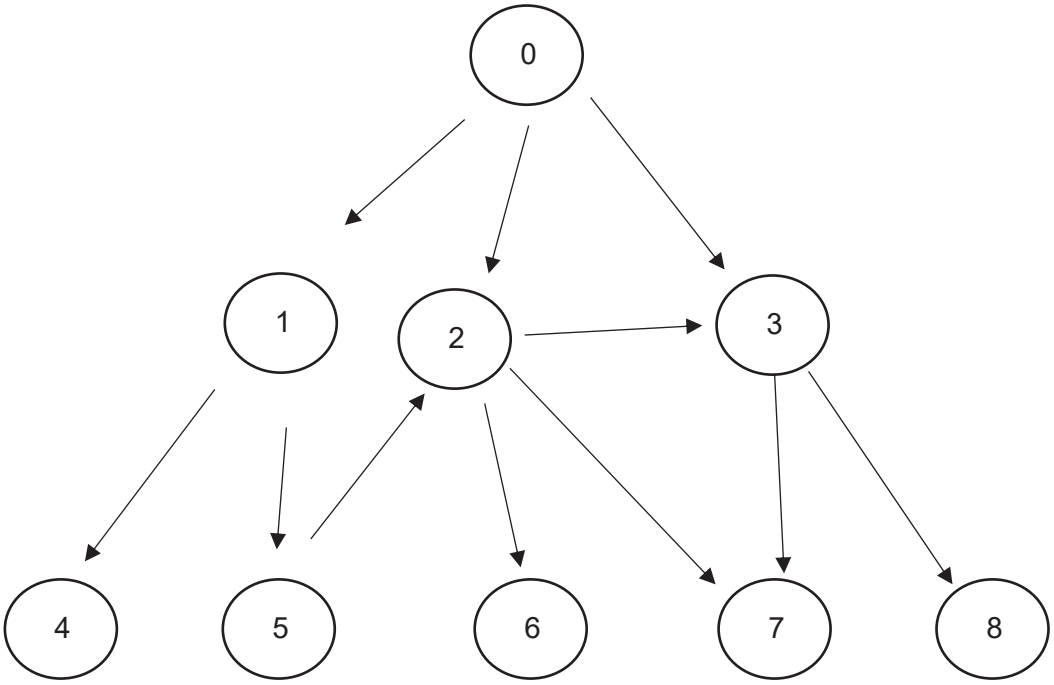
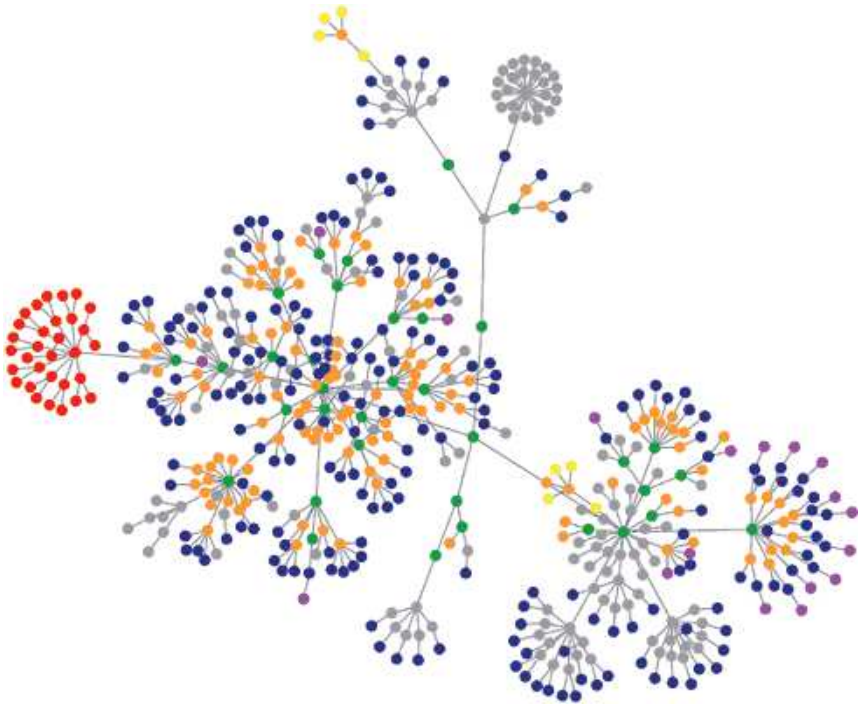


Figura 6. Conjunto de árvores formando uma floresta



TESTES E ANÁLISE DE VIABILIDADE

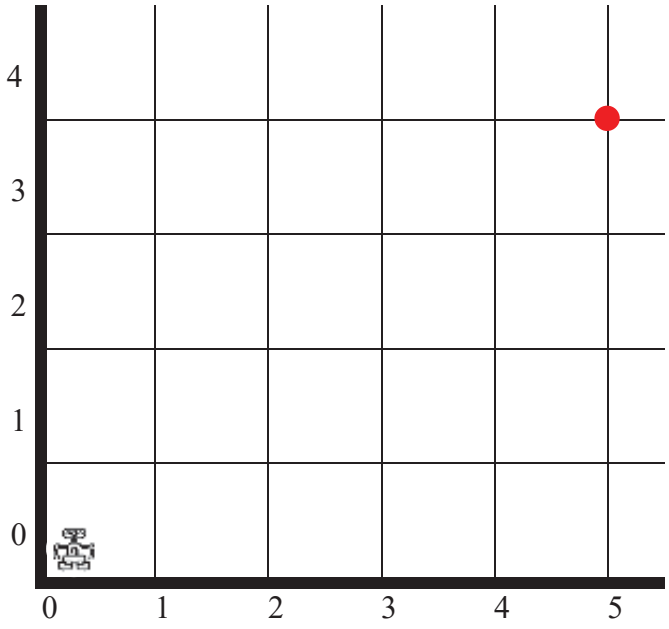
A primeira fase se dá de maneira com que o jogador se familiarize com o ambiente e o modo de jogo, onde ele possa entender como funcionam as jogadas, como movimentar o robô e como atingir o objetivo. Mesmo que simples, essa fase é de suma importância, pois é onde o jogador irá ter a visualização inicial de como poderá aplicar os conceitos básicos nas próximas fases.

O ponto inicial do robô é definido na Rua Y, Avenida X. O mundo é simples e o primeiro objetivo possui o nível de dificuldade mais baixo, pois as jogadas são menos complexas e não há obstáculos no campo de jogo.

É possível observar que há mais de uma maneira de chegar aos objetivos, como é visto nos exemplos a seguir. Porém no exemplo 1, o algoritmo possui 10 jogadas, enquanto o exemplo 2 possui 17 jogadas. Evidentemente, o exemplo de algoritmo 1 receberia uma maior pontuação em um caso de jogo, visto que apresentou menos movimentações para atingir o ponto final.

FASE 1

Figura 7. KAREL, FASE 1



A fase 1 inicia-se com o robô no estado (0,0) e o cenário apresentado é de caráter mais básico, sem obstáculos entre o robô e o estado final (4,5).

Cada movimento que o faça chegar mais perto de seu objetivo são somados para contabilizar a progressão à próxima fase. Quanto menor a quantidade de movimentos feitos, maior será a sua pontuação. Alguns alertas podem surgir no painel para advertir ao jogador de que fazer movimentos em que o robô volte estados não são aconselhados. Caso o jogador não atinja uma pontuação suficiente, ele deverá repetir a fase para percorrer um novo caminho.

Exemplo de algoritmo 1: Ponto inicial (r0, a0) e ponto final (r4, a5).

INICIO

Avançar (0,1)

Avançar (0,2)

Avançar (0,3)

Avançar (0,4)

Avançar (0,5)

Virar para a esquerda (0,5)

Avançar (1,5)

Avançar (2,5)

Avançar (3,5)

Avançar (4,5)

FIM ALGORITMO

Exemplo de algoritmo 1 com laço de repetição:

INICIO

Enquanto r < 4 faça

Avançar;

Fim enquanto

Virar para a direita; (4,0)

Enquanto a < 5

Então Avançar;

Fim enquanto

FIM ALGORITMO

Exemplo de algoritmo 2: Ponto inicial (r0, a0) e ponto final (r4, a5).

INICIO

Avançar (0,1)

Virar para a esquerda (0,1)

Avançar (1,1)

Virar para a direita (1,1)

Avançar (1,2)

Virar para a esquerda; (1,2)

Avançar (2,2)

Virar para a direita; (2,2)

Avançar (2,3)

Virar para a esquerda; (2,3)

Avançar (3,3)

Virar para a direita; (3,3)

Avançar (3,4)

Virar para a esquerda; (3,4)

Avançar (4,4)

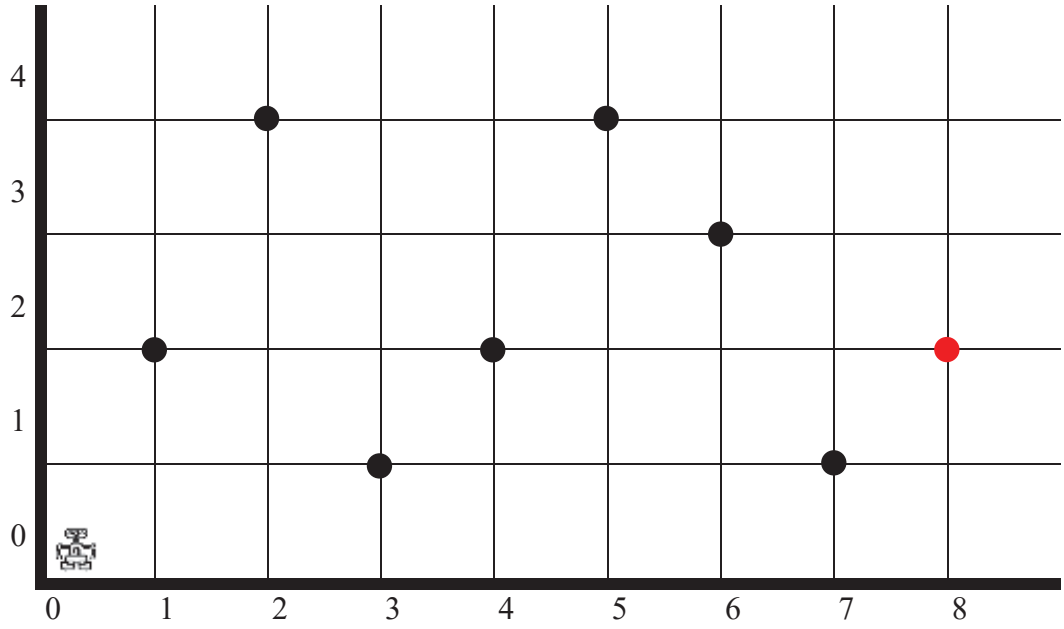
Virar para a direita; (4,4)

Avançar (4,5)

FIM ALGORITMO

FASE 2

Figura 8. KAREL, FASE 2



A fase 2 origina-se com o robô no estado (0,0) e o cenário começa a possuir elementos pelo percurso onde o robô deverá cruzar para agregar pontos a contagem e atingir o estado final a ser alcançado.

As mesmas regras da primeira fase são aplicadas, onde cada movimento é somado para contabilizar a progressão à próxima fase e quanto menor a quantidade de movimentos realizados, maior será a pontuação. Ao mesmo passo, ao movimentar o robô para um estado anterior ao atual, a pontuação do jogador será reduzida, fazendo com que ele evite voltar as jogadas. Para isso, os alertas ainda se farão presentes no painel, advertindo ao jogador que fazer movimentos de retorno não são aconselhados.

Caso o jogador não atinja uma pontuação considerada suficiente, ele deverá repetir a fase para percorrer um novo caminho.

Diante desse cenário de jogo, pode-se dar o seguinte algoritmo como exemplo para chegada ao objetivo.

Exemplo de algoritmo 1:

INICIO

Avançar (0,1)

Virar para a esquerda (0,1)

Avançar (1,1)

Avançar (2,1)

Virar para a direita (2,1)

Avançar (2,2)

Virar para a esquerda (2,2)

Avançar (3,2)

Avançar (4,2)

Virar para a direita (4,2)

Avançar (4,3)

Avançar (4,4)

Avançar (4,5)

Avançar (4,6)

Virar para a direita (4,6)

Avançar (3,6)

Avançar (2,6)

Avançar (1,6)

Virar para a esquerda (1,6)

Avançar (1,7)

Avançar (1,8)

Retornar (1,7)

Virar para a esquerda (1,7)

Avançar (2,7)

Virar para a direita (2,7)

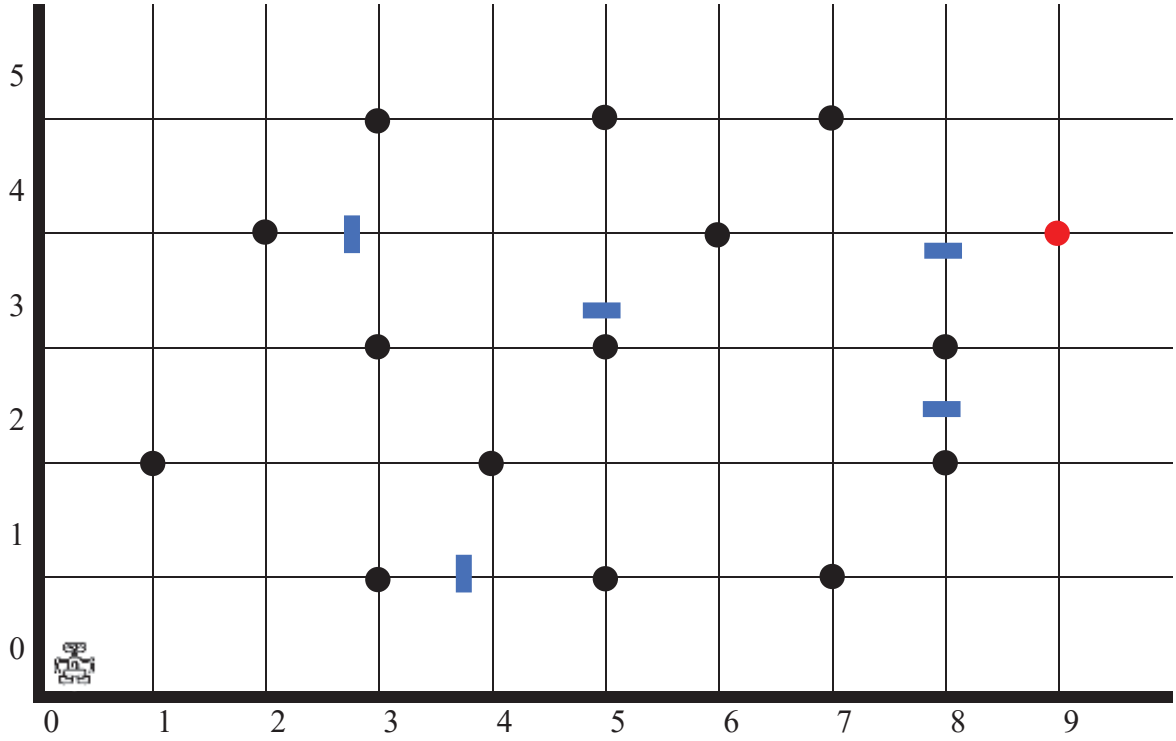
Avançar (2,8)

FIM ALGORTIMO

A partir desse ponto, o jogo começa a expandir e as fases necessitarão de mais raciocínio. A estrutura para os algoritmos a serem desenvolvidos pelo jogador será avaliada e a árvore de resolução, como visto na Figura 6, irá elaborar um conjunto de padrões para avaliar a possibilidade de movimentação do jogador, indicando no painel as combinações que podem ser feitas a partir das movimentações já realizadas com as possíveis movimentações no estado atual.

FASE 3

Figura 9. KAREL, FASE 3



A fase 3 possui alguns elementos no percurso que são barreiras onde o robô não conseguirá atravessar. Esses elementos são os retângulos azuis dispostos nas ruas e avenidas do mundo de Karel. Essas barreiras estão presentes nessa fase de jogo para aprimorar o raciocínio e o pensamento através da inserção de um novo objeto no campo de jogo.

As regras principais continuam, porém como agora há um objeto que impossibilita o robô de cruzar determinados caminhos, o jogador deverá voltar um estado do robô, caso tenha chegado até esse ponto, e percorrer outro caminho para alcançar o estado final. Caso o jogador se atente de antemão a respeito do obstáculo, deverá evitar aquele caminho para não sofrer perda de pontuação.

Nessa fase também o tempo irá interferir na pontuação, fazendo com que se o jogador demorar a chegar ao objetivo, sua pontuação seja menor.

Caso o jogador não atinja uma pontuação considerada suficiente, ele deverá repetir a fase para percorrer um novo caminho.

Diante desse cenário de jogo, pode-se dar o seguinte algoritmo como exemplo para chegada ao objetivo.

Exemplo de algoritmo fase 3:

```
INICIO
        Avance (0,1)
        Virar para a esquerda (0,1)
        Enquanto r < 2 faça
            Avançar;
        Fim enquanto
        Virar para a direita (2,1)
        Enquanto a < 4 faça
            Avançar;
        Fim enquanto
        Virar para a esquerda (2,4)
        Avance (3,4)
        Virar para a direita (3,4)
        Avance (3,5)
        Virar para a direita (3,5)
        Enquanto r != 1 faca
            Avançar;
        Fim enquanto
        Virar para a esquerda (1,5)
        Enquanto a < 8 faça
            Avançar;
        Fim enquanto
        Virar para a esquerda (1,8)
        Avance (2,8)
        Virar para a direita (2,8)
        Avance (2,9)
        Virar para a esquerda (2,9)
        Enquanto r < 4 faça
            Avançar;
```

Fim enquanto

FIM ALGORTIMO

CONCLUSÕES

Este presente trabalho, juntamente com o protótipo desenvolvido para a construção do pensamento computacional, baseado em um *Serious game*, pôde apresentar através de modelos de jogo e a partir dos resultados desse estudo, a validação da ideia para o desenvolvimento de novas metodologias de aprendizado.

Verificando a futura ampliação do ensino de matérias relacionadas à Computação, visto a necessidade que será exigida, é incontestável a aplicação de formas mais atrativas e envolventes de aprendizado, como a apresentada.

Para elaboração deste trabalho, foram consultados diversos artigos, livros e pesquisas referentes ao pensamento computacional e suas respectivas áreas de abrangência, principalmente da área educacional, para avaliar alternativas de uma melhor metodologia para aplicação dessa ideia deste conceito.

Também foram pesquisados conceitos e aplicações a cerca de *Serious Games*, levantadas informações acerca dos Serious games, os modos de aplicações nos diversos projetos encontrados e os resultados após as respectivas aplicações.

A ideia, a princípio, se baseia em implementar o Pensamento Computacional como forma de resolução de problemas na área educacional, entretanto, não é descartada a possibilidade de posteriormente ser expandido para outras áreas de aplicação.

É esperado que, seja possível fornecer uma nova metodologia para introdução do Pensamento Computacional entre as metodologias de ensino acadêmicas futuras, onde a tecnologia possa ser inserida para facilitar e auxiliar na resolução de problemas.

Vale ressaltar que o protótipo desenvolvido é apenas um modelo, podendo ser modificado e aperfeiçoado, para assim, poder aplicá-lo em uma amostra maior. As fases apresentadas se dão como base para um teste de aplicação onde os pilares do Pensamento Computacional que foram apresentados podem ser adquiridos e aprimorados conforme a manipulação do jogo para resolução das situações, modificando o modo e tempo de raciocínio do jogador.

Para os trabalhos futuros, já é considerada a ampliação do protótipo, onde será desenvolvida com mais cuidado a heurística do jogo e a inclusão de Inteligência Artificial, para verificar padrões de jogada dos usuários, identificar possíveis jogadas “viciadas”, avaliar a possibilidade de saltos entre as fases do jogo. Caso sejam identificadas jogadas mais elaboradas do jogador, habilitar a visão total do mapa somente na introdução do jogo

(ao mapa) e, após o início, a tela poderá se resumir a poucos pontos ao redor do robô e outras funcionalidades interessantes que possam surgir mais adiante.

REFERÊNCIAS BIBLIOGRÁFICAS

BERGIN, Joseph et al. **Karel++:** A Gentle Introduction to the Art of ObjectOriented Programming. JW, 1996.

BERGIN, Joseph. **Karel++ World for Macintosh:** (Preliminary Documentation). 1997-1998. Disponível em: <https://csis.pace.edu/~bergin/KWorld/KWorld.html>. Acesso em: 12 nov. 2018.

BOUCINHA, Rafael Marimon et al. **CONSTRUÇÃO DO PENSAMENTO COMPUTACIONAL ATRAVÉS DO DESENVOLVIMENTO DE GAMES**. **Renote**, v. 15, n. 1, 28 jul. 2017. Universidade Federal do Rio Grande do Sul. http://dx.doi.org/10.22456/1679-1916.75146.

BRACKMANN, Christian Puhlmann. **DESENVOLVIMENTO DO PENSAMENTO COMPUTACIONAL ATRAVÉS DE ATIVIDADES DESPLUGADAS NA EDUCAÇÃO BÁSICA.** 2017. 226 f. Tese (Doutorado) - Curso de Informática na Educação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2017.

FEOFILOFF, Paulo. **Algoritmos para Grafos em C.** 2018. Disponível em: <www.ime.usp.br/~pf/algoritmos_para_grafos/>. Acesso em: 30 nov. 2018.

FARRER, Harry et al. **Algoritmos Estruturados.** 3. ed. Rio de Janeiro: LTC, 2008. 284 p.

GOOGLE FOR EDUCATION. What is Computational Thinking? Computational Thinking for Educators, 2015. Disponível em: <https://computationalthinkingcourse.withgoogle.com/unit?lesson=8&unit=1>. Acesso em: 30/10/2015.

GUIMARÃES, Ângelo de Moura; LAGES, Newton Alberto de Castilho. **Algoritmos e estruturas de dados**. Rio de Janeiro: LTC, 1985-2008. 216 p.

KAFAI, Yasmin B.; BURKE, Quinn. **Constructionist Gaming**: Understanding the Benefits of Making Games for Learning. **Educational Psychologist**, v. 50, n. 4, p.313-334, 2 out. 2015. Informa UK Limited. http://dx.doi.org/10.1080/00461520.2015.1124022.

KENWRIGHT, Ben. Brief review of video games in learning & education how far we have come. **Siggraph Asia 2017 Symposium On Education On - Sa '17**, Bangkok, nov. 2017. ACM Press. http://dx.doi.org/10.1145/3134368.3139220.

MACHADO, L. S. et al. Serious games baseados em realidade virtual para educação médica. Revista Brasileira de Educação Médica. Rio de Janeiro, v. 35, n. 2, p. 254-262, jun. 2011.

MAIA, Estevan A. P. **SERIOUS GAMES:** UMA APLICAÇÃO NO CENÁRIO DA PATOLOGIA CLÍNICA VETERINÁRIA. 2017. 36 f. TCC (Graduação) - Curso de Sistemas de Informação, Centro Universitário Padre Anchieta, Jundiaí, 2017.

MARIN, Hernando Castaneda; GRATÉROL, Wladimir Rodríguez. **Programación Orientada a Objetos en el Micro mundo del Robot Karel.** 2006. Disponível em: <https://www.researchgate.net/publication/235004760_Programacion_Orientada_a_Objetos_en_el_Micro_mundo_del_Robot_Karel_libro_1>. Acesso em: 12 nov. 2018.

MICROSOFT. **O Ensino em 2030 e o aprendizado pronto para a vida:** O Imperativo tecnológico. 2018. Disponível em: <https://info.microsoft.com/LA-DIGTRNS-CNTNT-FY19-08Aug-01-ConhecaaSaladeAulade2030-MGC0002878_01Registration-ForminBody.html>. Acesso em: 13 ago. 2018.

RESNICK, Mitchel. **Mother's Day, Warrior Cats, and Digital Fluency:** Stories from the Scratch Online Community. 2012. Disponível em: <https://web.media.mit.edu/~mres/papers/mothers-day-warrior-cats.pdf>. Acesso em: 17 nov. 2018.

ROBERTS, Eric. **KAREL THE ROBOT LEARNS JAVA**. 2005. Disponível em: <https://web.stanford.edu/class/cs106a/book/karel-the-robot-learns-java.pdf>. Acesso em: 12 nov. 2018.

SEVERGNINI, Luís Filipe. **Alice e o Mistério dos Algoritmos**: um serious game como ferramenta de aprendizagem de lógica de programação para crianças. **Renote**: Revista Novas Tecnologias na Educação, v. 16, n. 1, 2018. Disponível em: <https://seer.ufrgs.br/renote/article/view/86049>. Acesso em: 11 nov. 2018.

WEISER, Mark. **The Computer for the 21st Century.** 1991. Disponível em: <https://www.ics.uci.edu/~corps/phaseii/Weiser-Computer21stCentury-SciAm.pdf>. Acesso em: 19 out. 2018.

WING, Jeannette. **PENSAMENTO COMPUTACIONAL** – Um conjunto de atitudes e habilidades que todos, não só cientistas da computação, ficaram ansiosos para aprender e usar. Revista Brasileira de Ensino de Ciência e Tecnologia, v. 9, n. 2, 2016. Disponível em: <https://periodicos.utfpr.edu.br/rbect/article/view/4711>. Acesso em: 16 ago. 2018.