UNIVERSIDADE DE CAXIAS DO SUL
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENSINO DE CIÊNCIAS E MATEMÁTICA
MESTRADO PROFISSIONAL

MARCELO PUZISKI

O DESAFIO DO DESENVOLVIMENTO DO PENSAMENTO COMPUTACIONAL NA ESCOLA:
VIVENCIANDO EXPERIÊNCIAS E CONSTRUINDO HABILIDADES

CAXIAS DO SUL, RS
2019

UNIVERSIDADE DE CAXIAS DO SUL
PROGRAMA DE PÓS-GRADUAÇÃO EM ENSINO DE CIÊNCIAS E MATEMÁTICA

O DESAFIO DO DESENVOLVIMENTO DO PENSAMENTO COMPUTACIONAL NA ESCOLA:
VIVENCIANDO EXPERIÊNCIAS E CONSTRUINDO HABILIDADES

Dissertação apresentada ao Programa de Pós-Graduação em Ensino de Ciências e Matemática da Universidade de Caxias do Sul, sob orientação da Profa. Valquíria Villas Boas Gomes Missell e coorientação da Profa. Carine Geltrudes Webber, como requisito parcial para a obtenção do título de Mestre em Ensino de Ciências e Matemática.

CAXIAS DO SUL
2019

AGRADECIMENTOS

Este trabalho de pesquisa é dedicado a todos aqueles que trabalham na área da Educação de maneira inclusiva. Em especial, é dedicado aos educadores que reservam parte do seu trabalho para incentivar a presença cada vez maior das minorias sociais nas Ciências.

O trabalho aqui realizado surgiu da minha vontade de aprimorar conhecimentos acerca do Ensino de Ciências e Matemática, primeiramente para aprimoramento pessoal. Assim, agradeço àqueles que me inspiram buscar criar e adaptar atividades diferentes e significativas em minhas aulas: meus alunos.

Agradeço também a todas as mulheres que me inspiraram e fizeram deste projeto uma realidade: Márcia Elisa Butignol Puziski, Valquíria Villas Boas Gomes Missell, Carine Geltrudes Webber, Daiane Rech, Alexandra Valença Colvara e Renata Da Pieve.

Agradeço ao apoio de Gabriel Méndez Piccoli, essencial nas etapas finais desta pesquisa, por me inspirar a querer ser uma pessoa melhor.

Por fim, agradeço aos leitores deste trabalho, pois se chegaram até aqui, significa que estão buscando diferentes perspectivas para a sala de aula e por isso, são excelentes professores e/ou pesquisadores, e espero que esta leitura acrescente coisas positivas ao seu trabalho!

Obrigado!

RESUMO

Neste documento apresenta-se uma pesquisa sobre o desenvolvimento das habilidades do Pensamento Computacional por crianças do 4º ao 7º anos do Ensino Fundamental. Esta pesquisa surgiu da reflexão sobre as necessidades de mudanças no sistema escolar, tendo em vista o novo perfil de aluno do século XXI. Um levantamento bibliográfico apresenta o Pensamento Computacional como uma das tendências na área da Educação para estes alunos. O Pensamento Computacional compreende um conjunto de habilidades fundamentais para o cidadão da sociedade atual. É a maneira de pensar como um cientista da computação para a resolução de problemas, utilizando as habilidades de decomposição, abstração e modelagem. A pesquisa em questão teve como objetivo avaliar o desenvolvimento das habilidades do Pensamento Computacional por meio de atividades realizadas no *software Scratch*, que foi desenvolvido para o ensino de programação para crianças. Embasado no Construcionismo de Seymour Papert e nas habilidades do Pensamento Computacional, cada encontro gerou dados que foram analisados e discutidos à luz dessas teorias. Os resultados das análises mostraram que atividades de programação realizadas no *Scratch* são uma maneira efetiva de colaborar no desenvolvimento de habilidades do Pensamento Computacional por crianças e adolescentes no Ensino Fundamental. A compilação destes dados deu origem a um produto educacional, que consiste em uma sequência didática, organizado na forma de um guia, com seus objetivos, atividades e instrumentos de avaliação, de maneira que possa ser utilizado e adaptado para qualquer escola com um laboratório simples de informática.

Palavras-chave: Pensamento Computacional. Construcionismo. Scratch. Ensino de Programação. Informática na Educação.

ABSTRACT

The following document presents a research about the development of Computational Thinking abilities on children from the 4th to 7th years of Elementary School. This research emerges from a reflection on the needs of changes on the school system, regarding the new profile of students from the 21st century. A literature review shows Computational Thinking as one of the trends on Education for these students. Computational Thinking theory, which also serves as basis to this research, presents this kind of thinking as a fundamental set of skills for the citizen of current society. It is the ability to think like a computer scientist towards problem-solving, and it relies on the abilities of decomposition, abstraction and design. The present project has as a main objective to evaluate the development of those skills through activities executed on the *Scratch* software, which was created for the teaching of programming for children. Based on Seymour Papert's Constructionism and the afore mentioned Computational Thinking theory, each encounter has produced data, which were analyzed and discussed under the light of these Theories. The results from the analysis have shown that the coding activities os *Scratch* are an effective way of developing Computational Thinking skills on students from Elementary School. The collected data have originated an educational product, which is a teaching guide, with its objectives, activities and evaluation methods, in such a way that it can be used and adapted to any school with a simple computer lab.

Keywords: Computational Thinking. Constructionism. Scratch. Programming Teaching. Computing in Education.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

A internet e as novas tecnologias dão acesso aos mais variados tipos de informação em qualquer lugar. A computação possibilita que grandes volumes de dados sejam produzidos, armazenados e acessados, tornando o conhecimento amplamente acessível e menos seletiva a sua produção (POZO, 2004).

Os estudantes das escolas brasileiras também são afetados por essas mudanças. Ainda assim, observa-se que o currículo das escolas está muito mais centrado em conteúdos e na retenção de informação do que nas habilidades necessárias para lidar com elas, conforme apontam Coutinho e Lisbôa (2011, p.5):

> O desafio imposto à escola por esta nova sociedade é imenso; o que se lhe pede é que seja capaz de desenvolver nos estudantes competências para participar e interagir num mundo global, altamente competitivo que valoriza o ser-se flexível, criativo, capaz de encontrar soluções inovadoras para os problemas de amanhã, ou seja, a capacidade de compreendermos que a aprendizagem não é um processo estático mas algo que deve acontecer ao longo de toda a vida.

Para que a escola possa superar este desafio, diversas reflexões sobre o currículo atual devem ser feitas, considerando as mudanças comentadas acima. Especialmente quanto à influência da computação na sociedade e que o computador já faz parte da cultura das crianças e adolescentes há algum tempo. Para Papert (2008, p.15),

> No mundo inteiro, as crianças assumiram um apaixonante e duradouro caso de amor com os computadores, utilizando-os de modo tão variado quanto suas atividades. A maior parte do tempo é dedicada aos jogos (...). Elas usam computadores para escrever, desenhar, comunicar-se, obter informações. Algumas os utilizam para relacionamentos sociais; outras para isolar-se. Em muitos casos, a dedicação ao computador é tanta que a palavra “vício” vem à mente de pais preocupados.

Este fenômeno cultural implica que o computador faz parte da realidade da maioria dos alunos das escolas, públicas e privadas, de países onde o acesso ao computador e, principalmente à internet, é possível.

No Brasil, estima-se que 54,4% dos brasileiros acima de 10 anos de idade teve acesso à internet nos três meses que antecederam ao levantamento realizado no ano de 2014 (IBGE, 2015). Desses usuários, 80,6% acessaram a internet pelo computador (IBGE, 2015).

Outro importante fator que amplificou o acesso das crianças e adolescentes à informação foi o surgimento smartphones. As tecnologias móveis crescem de maneira acelerada e, em 2016, o número de pessoas com mais de 10 anos de idade que tinham acesso a conexões

móveis já passava de 61% e, apesar destes dispositivos aumentarem os fluxos informacionais, eles também favorecem a busca e uso de informações de maneira superficial (CARNEIRO et al, 2018).

Em contrapartida a todos estes dados, a Ciência da Computação ainda é socialmente exclusiva. É uma ciência com baixo envolvimento de mulheres e pessoas de classes econômicas inferiores, o que é motivado pelo preconceito e discriminação (PAPERT, 1992; ANGOTTI, AUTH, 2001; BANDERA, 2008).

Segundo os autores Coutinho e Lisbôa (2011, p.5), a internet e as tecnologias digitais fizeram emergir um novo paradigma social. Este novo paradigma é, na verdade, a sociedade da informação, ou sociedade do conhecimento (HARGREAVES, 2003), ou sociedade da aprendizagem (POZO, 2004), ou, ainda, sociedade em rede alicerçada no poder da informação (CASTELLS, 2003). Essa nova sociedade nos insere, ainda segundo Coutinho e Lisbôa (2011, p.5), em

> Um mundo onde o fluxo de informações é intenso, em permanente mudança, e "onde o conhecimento é um recurso flexível, fluido, sempre em expansão e em mudança" (Hargreaves, 2003, p. 33). Um mundo desterritorializado, onde não existem barreiras de tempo e de espaço para que as pessoas se comuniquem.

A informação é matéria prima, estando na base do conhecimento e da comunicação entre as pessoas (COUTINHO, LISBÔA, 2011, p.5). A maneira como o indivíduo lida com a informação, no entanto, precisa permear o currículo das escolas na atual configuração da sociedade. O acesso à informação está fácil, então o enfoque pode mudar da retenção desta para o desenvolvimento de habilidades necessárias para utilizá-la na resolução de problemas.Além de analisar o contexto em que esta pesquisa está inserida, também precisa-se analisar quem são os sujeitos desta pesquisa. Quem são os estudantes do século XXI e da sociedade da informação?

O termo "nativos digitais" parece ser o mais usado para descrever o perfil dos estudantes das escolas atualmente. Este termo, cunhado por Marc Prensky em 2001 (PRENSKY, 2001; KENNEDY et al., 2008) refere-se à geração de estudantes do século XXI, nascidos depois de 1980[1], que passou toda sua vida cercada de computadores, *videogames*, *smartphones* e todas as outras ferramentas e brinquedos da era digital.

Essa nomenclatura sugere que os usuários de tecnologias digitais que não fazem parte desta geração são imigrantes digitais: pessoas que não nasceram na era digital, mas aprenderam,

[1] Considera-se o ano de 1980 o principal divisor entre as gerações, mas essa data pode variar se levarmos em conta estudos (KENNEDY et al., 2008) que identificam diferenças grandes no uso da tecnologia ligadas a condições socioeconômicas, fatores culturais/étnicos e gênero.

em algum momento da vida, a lidar com as novas tecnologias (PRENSKY, 2001). Os imigrantes digitais são, então, aqueles que nasceram antes de 1980, ou que não tiveram acesso às citadas ferramentas da era digital desde sua infância, mas sim tiveram de aprender a lidar com elas posteriormente.

Prensky (2001, p.1-2, tradução nossa) alerta: “os estudantes de hoje não são mais as pessoas que o nosso sistema educacional foi criado para ensinar”, e, ainda, “os professores, imigrantes digitais, que falam um idioma desatualizado, estão sofrendo para ensinar uma população que fala um idioma completamente novo”. Pozo (2004) afirma que “mudar as formas de aprender dos estudantes requer também mudar as formas de ensinar de seus professores”. Essas afirmações sinalizam a necessidade de mudança no atual sistema educacional, especialmente em relação aos processos de ensino e de aprendizagem. O conceito de aula, professor e estudante são questionados ao levantarmos tais discussões.

Os usos das Tecnologias da Informação e Comunicação (TICs) por estes jovens os diferencia das gerações anteriores. Para Bennet, Maton e Kervin (2008, p. 775-776, tradução nossa), “essas diferenças são tão significativas que a natureza da própria educação deve mudar para acomodar suas habilidades e interesses”.

A geração de estudantes que está na sala de aula atualmente vive em uma realidade cercada por sistemas controlados por computadores (veículos, casas, escola, a “internet das coisas”) e utiliza *softwares* variados para entretenimento (jogos, redes sociais, comunicação). Entretanto, mesmo com o predomínio e a preferência por recursos tecnológicos, os nativos digitais ainda têm limitados conhecimentos tecnológicos. O nível de conhecimento tecnológico que eles mobilizam raramente ultrapassa o que, em termos de domínio cognitivo, corresponde ao apontar, reconhecer, repetir, registrar, marcar (BLOOM et al, 1956). Desta forma, evidencia-se o uso dos recursos tecnológicos, sem a devida compreensão, aplicação ou análise.

Um conjunto de habilidades chamado de Pensamento Computacional parece suprir essa necessidade. O Pensamento Computacional consiste em um raciocínio análogo ao que os programadores empregam nos computadores, mas executados por humanos (WING, 2006). É o processo de reconhecer os aspectos da Computação no mundo para entendê-lo sob essa perspectiva (ROYAL SOCIETY, 2012).

Sem avanço no domínio cognitivo, os nativos digitais não alcançarão nenhum desenvolvimento ou criação tecnológica, habilidades tão necessárias nos dias atuais. Neste sentido, uma intervenção se torna necessária a fim de propor atividades e métodos de ensino que favoreçam aprendizagens do Pensamento Computacional, que contempla uma variedade de habilidades cognitivas para resolução de problemas que são o tema central desta dissertação.

Para atender as necessidades desse novo perfil de estudante, em nível global encontramos diversas ações para promover mudanças na educação. Destacamos aqui três movimentos que estão acontecendo: um nos Estados Unidos da América, outro no Reino Unido e um terceiro, a nível global. Essas ações apontam as principais preocupações dos professores, pesquisadores e demais especialistas da área. Também apontam as tendências da educação em dois países superdesenvolvidos.

A escolha destas ações tem a finalidade de estabelecer diretrizes para este trabalho, mostrando as principais forças que movimentam os sistemas educacionais para as quais queremos nos direcionar.

A coligação estadunidense P21 (*Partnership for 21st Century Learning*[2]), fundada em 2002, contou com o apoio de professores, líderes da educação e empresas para definir uma estrutura (*framework*) de habilidades e conhecimentos que os estudantes do século XXI precisam para obter sucesso no trabalho, na vida e na sua cidadania.

Para ilustrar a magnitude deste projeto, destacamos, entre as empresas que apoiam o projeto, algumas que são notórias pelo seu interesse e preocupação com a educação, como a *LEGO Education, Apple Incorporation, Pearson, The Walt Disney Company, Fisher-Price* e a *Intel Corporation.* Além do apoio dessas empresas, o governo dos Estados Unidos também apoia diretamente a parceria, através de seu Ministério da Educação.

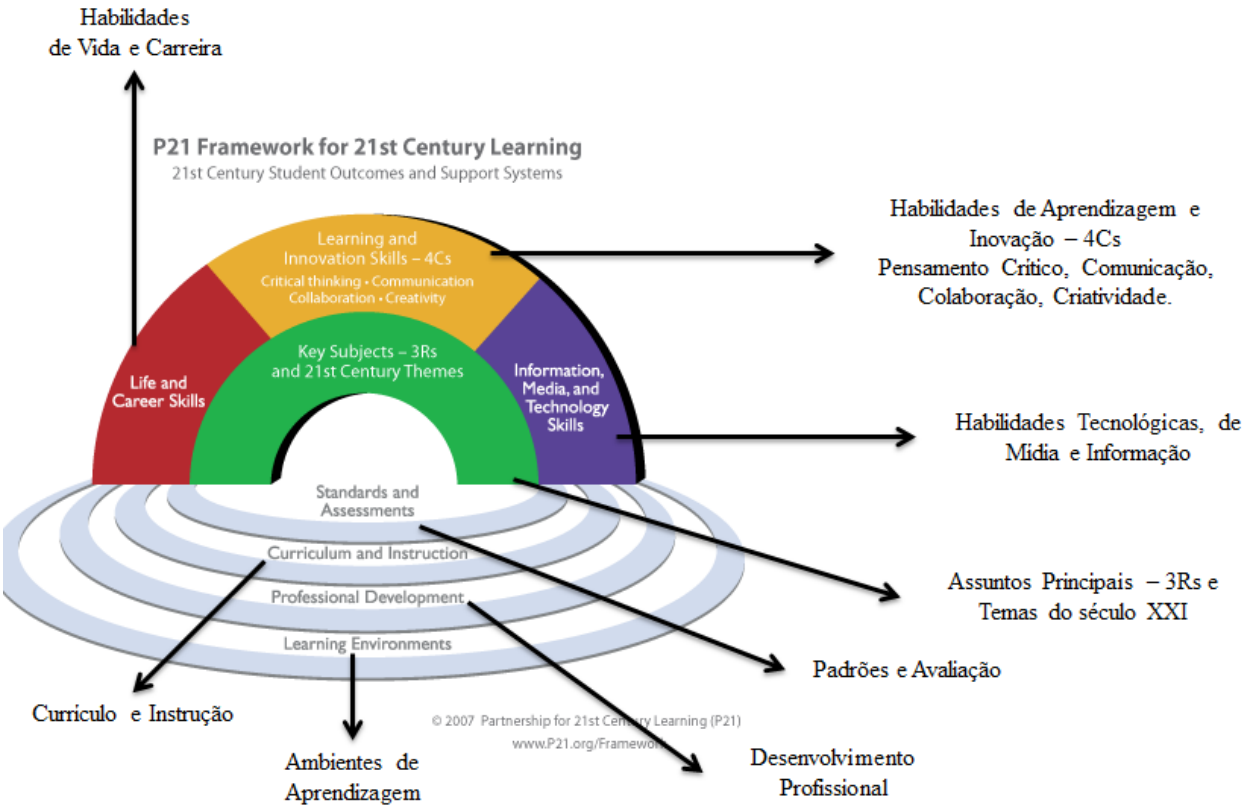
A Figura *1* representa a estrutura construída pela coligação (P21, 2016), onde destacamos os campos "*Learning and Innovation Skills*" e "*Information, Media and Technology Skills*":

"Habilidades de aprendizagem e inovação são o que diferenciam estudantes que estão preparados para uma vida progressivamente complexa" (P21, 2016, p. 2, tradução nossa). Essas habilidades são o que a P21 chamou de 4 Cs:

- Criatividade e Inovação;
- Pensamento Crítico e Resolução de Problemas;
- Comunicação; e
- Colaboração.

[2] Parceria pela Aprendizagem no Século 21. http://www.p21.org

Figura 1- Estrutura pela aprendizagem do século XXI (P21, 2016, tradução nossa).



Quanto às habilidades tecnológicas, de mídia e informação, a P21 destaca que "hoje vivemos em um ambiente conduzido pela mídia e tecnologia, marcado pelo acesso a uma abundância de informação" (P21, 2016, p.2, tradução nossa) e também destaca a importância das habilidades de:

- Alfabetização[3] informática;
- Alfabetização midiática; e
- Alfabetização nas TICs.

A *Royal Society*[4], instituição que existe desde 1600 e cria ações para promover o desenvolvimento das Ciências no Reino Unido também está constantemente envolvida em diversos projetos que visam a adequação dos sistemas de ensino e da Educação como um todo para os cidadãos do século XXI.

[3] O termo "alfabetização" foi traduzido do termo "*literacy*" que pode ter significados levemente variados, conforme o contexto. Aqui, entende-se que ele representa o domínio, ou algum grau de instrução elevado sobre determinado assunto.

[4] Academia nacional de ciências do Reino Unido. https://royalsociety.org

Destacamos aqui dois projetos atuais da *Royal Society:* "*Vision for Science and Mathematics Education*"[5] (ROYAL SOCIETY, 2014) e "*After the reboot: computing education in UK schools*"[6] (ROYAL SOCIETY, 2017).

O primeiro dos projetos destacado tem como objetivos, ou visão, que todos os jovens estudem Matemática até a idade de 18 anos, que o currículo e sua avaliação sejam estabilizados e deem apoio a um excelente processo de ensino e de aprendizagem, que a profissão professor seja um *status* profissional elevado e que exista um grande número de especialistas em Ciências e Matemática (ROYAL SOCIETY, 2014). Mais do que isso, o projeto propõe uma reforma na maneira com que se ensina e se aprende Ciências e Matemática. A criação deste projeto se justifica com um de muitos estudos que mostram que o crescimento tecnológico e as mudanças nos modelos empresariais ocasionaram a necessidade de contínua adaptação de habilidades para a participação do cidadão no mercado de trabalho (UKCES, 2014). Além disso, a *Royal Society* (2014, p. 16, tradução nossa) afirma: "O Reino Unido não está sozinho ao enfrentar esses desafios de habilidades. A resposta global tem sido a de analisar e reformar sistemas de educação".

Dentre os diagnósticos das mudanças necessárias na Educação, a Royal Society (2014, p. 36, tradução nossa) também afirma que "Se torna evidente que os últimos avanços científicos e tecnológicos vão exigir que as pessoas sejam altamente aptas em lidar e analisar dados, incluindo habilidades do Pensamento Computacional". O Pensamento Computacional, que será melhor explicado no capítulo 3, é um dos conceitos nos quais se baseiam a nossa pesquisa.

Já o projeto "*After the reboot: computing education in UK schools*" apresenta uma análise da introdução à computação como uma disciplina nas escolas do Reino Unido, que foi realizada em 2014, e aponta as tendências e o futuro desta disciplina. A disciplina computação nas escolas do Reino Unido abrange três linhas: tecnologia da informação, alfabetização digital e Ciência da Computação. Destacam também a habilidade do Pensamento Computacional como uma componente central do currículo (ROYAL SOCIETY, 2017).

O *software Scratch*, desenvolvido pelo Massachusetts Institute of Technology (MIT) em 2007, foi projetado especialmente para crianças entre 8 e 16 anos criarem projetos por meio da programação.

A construção de algoritmos pode ser algo desafiador para crianças e adolescentes, visto que o ensino das linguagens de programação e desenvolvimento de algoritmos é, de certa forma, um processo de alfabetização (DELGADO et al., 2004). O *Scratch*, no entanto, é uma

[5] Tradução: Visão para a Ensino de Ciências e Matemática.
[6] Tradução: Depois de reiniciar: Ensino de computação nas escolas do Reino Unido

ferramenta didática adequada para crianças, por ter um design e um caráter menos intimidador (HARVEY; MÖNIG, 2010). Segundo Vidal e colaboradores (2015, p.24, tradução nossa),

> O *Scratch* facilita a aprendizagem de regras sintáticas e semânticas das linguagens de programação tradicionais, e facilita, através do uso de elementos multimídia como imagem e som, a visualização de elementos algorítmicos tais como movimento, condições e repetições de ações.

No processo de criar e compartilhar criações e projetos do *Scratch*, crianças não só aprendem conceitos importantes de Matemática e Ciência da Computação, mas também desenvolvem habilidades de aprendizado importantes, como a criatividade, comunicação efetiva, análise crítica, experimentação sistemática, design iterativo e aprendizagem contínua (MONROY-HERNÁNDEZ, RESNICK, 2010).

A escolha do computador como instrumento de aprendizagem também leva em conta que "o computador se encontra entre o mundo de sistemas formais e o mundo de coisas físicas; ele tem a habilidade de fazer o abstrato concreto" (PAPERT, 1992, p.2). Especialmente por meio do Scratch, o pensamento pode ser transformado em uma representação visual, o que possibilita novas estratégias de ensino e aprendizagem.

Levando em conta o perfil dos estudantes do século XXI, que necessita de habilidades adequadas a sua realidade, e as tendências mundiais de especialistas e cientistas da Educação, sinaliza-se uma grande necessidade de mudança no sistema escolar atual.

Por meio dos dados e discussões levantados nesta introdução, foram consideradas diferentes possibilidades de desenvolver habilidades diferentes na escola. Mais especificamente, as habilidades do Pensamento Computacional.

Em particular, com esta pesquisa, levanta-se a seguinte questão: **De que maneira atividades de programação realizadas no *Scratch*[7] podem influenciar no desenvolvimento do Pensamento Computacional de crianças e adolescentes que estão no Ensino Fundamental?**

Os objetivos deste trabalho dividem-se em objetivo geral e objetivos específicos. Este trabalho teve como objetivo geral:

[7] Software educacional com linguagem de programação própria, desenvolvido para fins educacionais pelo Massachusetts Institute of Technology (MIT). Será a ferramenta principal desta pesquisa e é melhor apresentada no capítulo 3.

- Promover o desenvolvimento das habilidades do Pensamento Computacional de crianças do quarto ano escolar ao sétimo ano escolar, por meio de atividades desenvolvidas semanalmente por meio do software *Scratch*.

A realização desta pesquisa teve como objetivos específicos:

- Planejar e realizar encontros direcionados para estudantes do turno integral do quarto ao sétimo ano escolar, com a intenção de promover o desenvolvimento de habilidades do Pensamento Computacional;
- Inspecionar os produtos, programas e registros escritos destes encontros para verificar o desenvolvimento dos estudantes em relação às práticas e habilidades do Pensamento Computacional;
- Elaborar, como produto educacional, um guia didático para implantação destes encontros em qualquer escola que tenha um laboratório de informática.

Este trabalho está organizado em seis capítulos.

No capítulo 2, apresentamos o referencial teórico, que serve como base conceitual ao trabalho.

No capítulo 3 são apresentados os procedimentos metodológicos, ou seja, a caracterização da pesquisa, o contexto de realização da mesma, os recursos utilizados, os métodos de análise de dados e o desenvolvimento das atividades realizadas.

Apresentamos, no capítulo 4, o produto educacional desta pesquisa, que é um guia didático com as atividades que foram desenvolvidas.

O capítulo 5 é destinado à análise dos resultados da pesquisa.

Finalmente, no capítulo 6, realizamos uma reflexão sobre este trabalho e o que ele representa, perante a situação atual da educação.

2. REFERENCIAL TEÓRICO

Neste capítulo, abordamos a teoria construcionista de Seymour Papert, exploramos conceitos e definições do Pensamento Computacional, como o Pensamento Computacional se relaciona com a escola, apresentamos características do estudante do século XXI e a Taxionomia de Bloom.

2.1. Trabalhos Relacionados e Diferenciais

No Brasil, o movimento pelo desenvolvimento do Pensamento Computacional na Escola é amplamente discutido, mas pouco colocado em prática. Barcelos e Silveira (2012) trazem reflexões sobre a relação entre o Pensamento Computacional e a Matemática na Educação Básica, apresentando três competências do Pensamento Computacional na Matemática: "articulação de símbolos, identificação de padrões e regularidades e construção de modelos representativos e explicativos" (BARCELOS, SILVEIRA, 2012, p. 32).

Andrade e colaboradores (2013) trazem sugestões de jogos físicos que podem ser utilizados em atividades voltadas para o desenvolvimento do Pensamento Computacional no Ensino Fundamental. São três atividades, que consistem em um jogo de adivinhação (cara-a-cara), uma caça ao tesouro e a organização de uma festa. Os autores trazem modelos de avaliação como sugestão e um guia de aplicação, porém não houve a coleta de dados e análise de resultados.

Rodriguez e colaboradores (2015) utilizam o *Scratch* com 7 alunos de 14 a 16 anos de idade, em um projeto de pré-Iniciação Científica. Os estudantes desenvolveram durante 6 meses projetos colaborativos no *Scratch*, na forma de jogos educativos, a fim de desenvolver habilidades de programação. Sobre os resultados, os autores destacam que "como resultado direto dos procedimentos adotados nas atividades de exploração dos recursos do Scratch, destaca-se a possibilidade de estimular o raciocínio lógico e a resolução de problemas de forma lúdica e dinâmica, além do desenvolvimento de noções básicas de programação" (RODRIGUEZ et al., 2015).

Dentre outros trabalhos que utilizaram o Scratch para ensinar a lógica de programação, destacamos o trabalho de Oliveira e colaboradores (2014), realizado com 20 alunos do 9º ano do Ensino Fundamental, na forma de 10 encontros de 2h, utilizando o *Scratch* para desenvolver jogos e animações. Os resultados obtidos pelos autores mostram que é possível incluir a

temática “lógica de programação” no cotidiano escolar. Também avaliaram o uso do *Scratch* como ferramenta auxiliar, onde todos os alunos alegaram interesse e afinidade com o *software*.

O projeto de Schoeffel e colaboradores (2015) trouxe o *Scratch* e a robótica para alunos do 7º a 9º anos do Ensino Fundamental com uma metodologia baseada na resolução de problemas (*Problem Based Learning*) para a introdução de conceitos de lógica de programação. O objetivo era desenvolver nestes alunos habilidades de programação e prepará-los para participar da Olimpíada Brasileira de Informática. Os autores encontraram evidências de um desempenho superior nos alunos participantes na Olimpíada Brasileira de Informática em comparação com aqueles que não participaram da atividade, destacando que as questões desta Olimpíada são exercícios de lógica.

Dentre os trabalhos analisados, o de Brackmann (2017) foi o mais coeso quanto à metodologia e resultados. É uma tese de doutorado que contou com aplicações práticas no Brasil e na Espanha para o desenvolvimento do Pensamento Computacional em crianças do 5º e 6º anos do Ensino Fundamental. Utilizando a metodologia de atividades desplugadas (espécie de programação sem computador) realizadas em sala de aula, aliadas a atividades no *Scratch*, realizadas no laboratório de informática, Brackmann encontrou sucesso em suas atividades e trouxe grandes reflexões. “A questão agora não é se devemos desenvolver o Pensamento Computacional de crianças nas escolas e, sim, como e a partir de que idade/ano. É um caminho sem volta.” (BRACKMANN, 2017, p. 167).

Foram analisadas também dois levantamentos de trabalhos sobre o Pensamento Computacional (SCHOEFFEL et al., 2015; ARAÚJO et al., 2016). Schoeffel e colaboradores (2015, p. 1477), após sua análise, destacam que:

> Com relação aos relatos de ensino de Pensamento Computacional e raciocínio lógico por meio da programação, diversos trabalhos foram encontrados, porém a maioria deles descreve ações isoladas e de curta duração.

Araújo e colaboradores (2016) analisaram 15 artigos brasileiros sobre o desenvolvimento do Pensamento Computacional. Nestes artigos, 10 deles utilizavam a programação como instrumento principal, e o *software Scratch* foi determinante em 7 destes. Sobre as habilidades avaliadas, as de programação e algoritmo foram as principais e em dois dos trabalhos foram utilizados os conceitos computacionais de Brennan e Resnick (2012) para a avaliação. Sobre o instrumento de avaliação, a maioria dos artigos (9) utilizaram testes. Outro instrumento que foi bastante utilizado é a avaliação por meio da análise dos códigos e projetos. Esse método foi utilizado em 5 dos trabalhos analisados.

A partir da análise realizada nos trabalhos citados foi possível observar padrões nos trabalhos brasileiros que envolvem *Scratch* e Pensamento Computacional. A maioria dos trabalhos utiliza o *Scratch* para introduzir conceitos de programação, mas não focam suas análises em habilidades como o Pensamento Computacional. Pensando nisto, o trabalho aqui proposto visa desenvolver habilidades do Pensamento Computacional nos estudantes e propor atividades passíveis de reprodução em qualquer escola do Brasil.

Os documentos encontrados e analisados foram artigos de revistas científicas e também apresentados em congressos, com exceção da tese de doutorado de Brackmann (2017). Assim, relatos de experiência com análises mais complexas sobre o Pensamento Computacional na escola brasileira tornam-se necessárias.

Levando em conta os trabalhos acima mencionados, este trabalho propos-se a apresentar resultados que possam auxiliar e suprir essas carências na literatura avaliada e contribuir com novas possibilidades de discussões e, especialmente, análise de resultados para trabalhos científicos da área.

2.2. O Construcionismo

O Construcionismo é a filosofia do aprendizado mão-na-massa (*hands-on learning*), onde a aprendizagem acontece quando o aprendiz constrói algo (KURTI, KURTI, FLEMING, 2014). Foi idealizado por Seymour Papert, famoso pesquisador das áreas da Educação, Matemática e Ciências da Computação, do *Massachusetts Institute of Technology* (MIT).

A teoria Construcionista parte dos pressupostos construtivistas da teoria de Jean Piaget, onde o processo de aprendizagem é visto como construção de estruturas do conhecimento, independente das circunstâncias do aprendizado, e complementa a sua teoria com a ideia de que esse aprendizado acontece de maneira especialmente efetiva onde o aprendiz está conscientemente engajado em construir uma entidade pública, “seja ela um castelo de areia na praia ou a Teoria do Universo” (PAPERT, HAREL, 1991).

Segundo Ackermann (2001, p. 4, tradução nossa, grifo do autor), “a abordagem de Papert ajuda a compreender *como ideias são formadas e transformadas quando expressas através de diferentes meios, quando analisadas em contextos particulares, quando trabalhadas por mentes diferentes*.”.

Sobre educar para a matemática, Papert (1971, tradução nossa) afirma “(...) ser um matemático, como ser um poeta, ou um compositor ou um engenheiro, significa fazer, ao invés

de conhecer ou entender.” Isso significa que, no Construcionismo, o trabalho “mão-na-massa” (não necessariamente com material concreto) é altamente estimado como situação de aprendizagem. Segundo Papert (1971, p. 1-2), sobre o trabalho criativo matemático,

> É (...) pressuposto que poucas pessoas podem trabalhar criativamente em matemática. Eu acredito que existe uma conspiração acidental de psicólogos e matemáticos em manter essa presunção. Os psicólogos contribuem a ela por genuína ignorância do que o trabalho matemático criativo possa ser. Os matemáticos, muito frequentemente, o fazem pelo seu elitismo, na forma de uma forte convicção de que a criatividade matemática é privilégio de uma minoria.

O processo de quebra dessa presunção é muitas vezes referido como o “processo de desmistificação da matemática” (MIGUEL, MIORIM, 2013), especialmente na fala corriqueira dos professores. Oferecer oportunidades justas e possibilitar o acesso à informação e ao conhecimento a todos permeia as ideias do construcionismo.

Especialmente na computação, Turkel e Papert (1990, p. 129, tradução nossa) identificam a importância e a presença da pluralidade epistemológica: “A diversidade de abordagens à programação sugere que acesso igualitário até mesmo aos elementos mais básicos da computação exige a aceitação da validade de diversos jeitos de saber e pensar, uma pluralidade epistemológica”.

Grandes ferramentas educacionais atuais surgiram com as ideias construcionistas, como é o caso de peças LEGO voltadas para a educação[8], a robótica educacional, os Espaços Maker e FabLabs, as impressoras 3D, softwares como o LOGO e o *Scratch*, entre outros.

2.2.1. Maker Education

Trabalhos de Papert comprovam avanços matemáticos em uma turma de sétima série, mesmo que essa turma não tenha tido uma aula formal de matemática, apenas atividades de programação LOGO[9] (PAPERT, HAREL, 1991; PAPERT, 1971, 1980). No Construcionismo, a sala de aula e os materiais pedagógicos se transformam. Não mais é necessário o quadro-negro e o livro didático, e esses são substituídos por outras plataformas onde o aluno pode construir e expressar seu conhecimento, como é o caso dos computadores e *tablets,* materiais de robótica,

[8] Atualmente a LEGO conta com uma ramificação chamada de LEGO *Education Solutions*, voltada a produzir produtos de uso educacional para sala de aula, incluindo a linha *Mindstorms,* que é voltada à robótica educacional.

[9] LOGO é uma linguagem de programação educacional desenvolvida por Seymour Papert na década de 1980. É considerada a primeira linguagem de programação desenvolvida para fins educacionais.

impressoras 3D, peças LEGO, material de desenho, tecidos e qualquer material que possa ser utilizado para criar e construir.

Os espaços de aprendizagem, além da sala de aula, podem tomar a forma de laboratórios, oficinas e ateliês. Estes ambientes são chamados de "espaço *maker*"[10], e caracterizam a *Maker Education*, que é "um ramo da filosofia construcionista que trata o aprendizado como um esforço pessoal que exige que o estudante, ao invés do professor, inicie o processo de aprendizagem" (KURTI, KURTI, FLEMING, 2014, p. 8, tradução nossa).

A *Maker Education* encoraja os estudantes a participar de trabalhos de resolução de problemas, aprendizagem auto-dirigida e de colaboração, tudo isso criando projetos tangíveis para apresentar aos colegas (HSU, BALDWYN, CHING, 2017).

Godhe, Lilja e Selwyn (2019) levantam a questão de que a introdução das tecnologias *maker* nas escolas é insuficiente para a integração da cultura *maker* ao ambiente formal de educação. É necessário mudança e análise crítica por parte dos educadores. Neste contexto, o presente trabalho de pesquisa torna-se mais um argumento para essa discussão.

2.3. Conceituando o Pensamento Computacional

O termo "Pensamento Computacional" (em inglês "*Computational Thinking"*) é um conceito que está gradativamente evoluindo, desde o século XX. O termo foi utilizado por Seymour Papert pela primeira vez em um artigo publicado em 1996. A conceituação moderna do termo foi feita por Jeannette Wing (WING, 2006, 2017), que o apresentou como um conjunto de habilidades e atitudes que pode ser aprendido e aplicado por todos, não somente cientistas da computação (KALELIOGLU; GÜLBAHAR; KUKUL, 2016).

Wing (2006, p.33, tradução nossa) destaca o Pensamento Computacional como "[um conjunto de habilidades que] se constroem nos poderes e limites da computação, seja ela executada por um humano ou por uma máquina". Fundamentalmente, o Pensamento Computacional requer pensar como um cientista da computação. É fazer uso das habilidades de abstração, reformulação de problemas, organização e análise de dados, automação, eficiência e generalização (BARR; HARRISON; CONERY, 2011).

Este tipo de pensamento exige mais do que a habilidade de programar um computador. Ele requer o pensamento em múltiplos níveis de abstração (WING, 2006). Ele é definido pela

[10] A tradução literal de "*maker*" é "criador", assim podemos traduzir espaço *maker* (ou *makerspace*) como um espaço de criação. Outras variações destes ambientes podem ser encontradas, como *FabLab*, *Hackerspace* ou *TechShop*. Estes últimos totalmente voltados a fins tecnológicos.

autora (WING, 2017, p.3, tradução nossa) como “o processo de pensamento envolvido em formular um problema e expressar sua solução de maneira que um computador – humano ou máquina – possa efetivamente resolvê-lo”.

Para Barr e Stephenson (2011, p.115, tradução nossa), “o poder do Pensamento Computacional é que ele se aplica a todos os outros tipos de raciocínio”. Wing (2006, p. 35) o apresenta como um pensamento que “complementa e combina o Pensamento Matemático e Pensamento de Engenharia”. A percepção de que é um conjunto de habilidades útil exclusivamente para cientistas da computação, apesar do seu nome, não compreende todas as possibilidades que o Pensamento Computacional permite ao ser humano.

A Royal Society (2012, p. 29, tradução nossa) definiu o Pensamento Computacional como “o processo de reconhecer aspectos da Computação no mundo que nos cerca, e utilizar as ferramentas e técnicas da Ciência da Computação para entender e raciocinar sobre sistemas e processos, tanto naturais quanto artificiais”. Mais recentemente, em 2017, ela se posicionou sobre o termo em um de seus projetos, destacado na introdução deste trabalho:

> No seu núcleo está a ideia de que as soluções para muitos problemas não são facilmente quantificadas como uma resposta direta, mas sim como algoritmos que levam à resposta; soluções para uma classe inteira de problemas codificadas em um conjunto de instruções que podem ser seguidas por computadores ou humanos. Além disso, [o Pensamento Computacional] engloba a ideia de que, usando técnicas algorítmicas, sistemas computacionais podem modelar muitos fenômenos, da mudança climática até o jeito que nossos cérebros funcionam e o funcionamento de células cancerígenas. O Pensamento Computacional permite o desenvolvimento de soluções algorítmicas úteis até para problemas complexos (ROYAL SOCIETY, 2017, p. 13, tradução nossa).

Diversos pesquisadores tentaram entrar em acordo com uma definição final para o Pensamento Computacional (WING, 2006; BARR, STEPHENSON 2011; BRENNAN, RESNICK, 2012; KALELIOGLU; GÜLBAHAR; KUKUL, 2016). Este consenso nunca foi atingido. Apesar disso, todos os esforços convergiram a grupos similares de habilidades e atitudes.

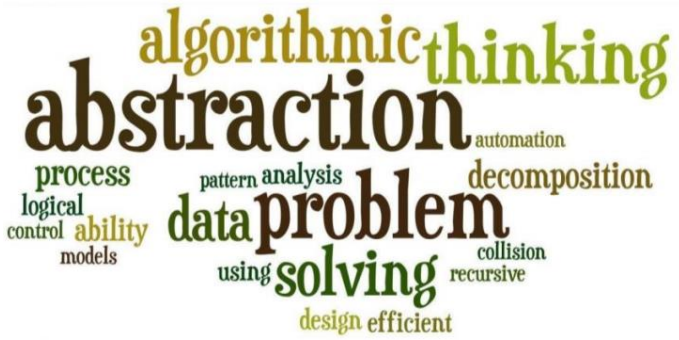
Kalelioglu, Gülbahar e Kukul (2016), com a intenção de aproximar e compilar as diferentes concepções de Pensamento Computacional presentes na literatura, realizaram um estudo analisando 274 artigos publicados entre 2006 e 2014, retirados de seis *databases* e bibliotecas digitais. Como resultado dessa análise, desenvolveram um *wordle*[11] (Figura *2*) com as palavras que mais aparecem ao definir Pensamento Computacional.

[11] Wordle é uma nuvem de palavras, onde aparecem as palavras com mais frequência de uso nos textos analisados. As palavras maiores são aquelas que apareceram com maior frequência.

Analisando a Figura 2 e o sentido das palavras que a compõem, pode-se considerar que a abstração e resolução de problemas são fatores centrais do que se entende por Pensamento Computacional.

Tendo em vista a necessidade de uma revisão na escola brasileira e adequação às necessidades do estudante do século XXI, destacadas na introdução deste trabalho, é notável que as habilidades e atitudes do Pensamento Computacional podem fazer parte de um novo modelo de currículo para a escola.

Figura 2- *Wordle* com as principais palavras usadas nas definições de Pensamento Computacional (KALELIOGLU; GÜLBAHAR; KUKUL, 2016).



2.4. O Pensamento Computacional na Escola

Em 2009, a *Computer Science Teacher Association*[12] (CSTA) e a *International Society for Technology in Education*[13] (ISTE) lançaram um projeto com o objetivo de desenvolver uma definição operacional do Pensamento Computacional para o período *K-12*[14], juntando os conceitos do Pensamento Computacional com o currículo da escola. No entanto, os envolvidos no projeto chegaram ao consenso de que uma mudança educacional é consideravelmente mais complexa do que suspeitavam, e que era necessário desvincular o Pensamento Computacional da Ciência da Computação (BARR, STEPHENSON, 2011; ISTE, 2011).

[12] Associação de Professores de Ciências da Computação: uma organização com mais de 7000 membros, educadores da área da ciência da computação. Cuja missão é apoiar e promover o ensino de ciências da computação e áreas afins no período primário e secundário escolar. https://www.csteachers.org/

[13] Sociedade Internacional pela Tecnologia na Educação: organização criada para promover o uso das tecnologias para a solução de problemas na educação. https://www.iste.org

[14] Termo utilizado em alguns países para se referir ao período escolar da educação básica, que vai do *kindergarten* (4-6 anos) ao último ano escolar (17-19 anos).

A noção de que o Pensamento Computacional deve ser desvinculado da Ciência da Computação vem ao encontro da ideia de Wing (2006, p. 33, tradução nossa), quando ela afirma que o Pensamento Computacional "representa um conjunto de habilidades e atitudes universais, que todos, não apenas cientistas da Computação, estariam ávidos para aprender e utilizar".

Barr e Stephenson (2011, p. 118) listaram conceitos do Pensamento Computacional que os participantes do projeto destacaram no contexto de capacidades, disposições e predisposições, e cultura de sala de aula. Dentre as capacidades, que descrevem o que os estudantes devem fazer para colocar em prática o Pensamento Computacional, encontramos:

- Encontrar soluções para problemas;
- Testar e depurar erros;
- Modelar, realizar simulações e análises de sistemas;
- Refletir sobre a prática e comunicação;
- Reconhecer abstrações e mover entre os níveis de abstração;
- Inovar, explorar e ter criatividade nas disciplinas;
- Resolver problemas em grupos; e
- Empregar estratégias de aprendizagem diversas.

Entre as disposições e predisposições, categoria que surgiu como uma tentativa de capturar os valores, motivações, sentimentos e atitudes aplicáveis ao Pensamento Computacional, destacaram-se:

- Confiança em lidar com a complexidade;
- Persistência em trabalhar com problemas difíceis;
- Habilidade de lidar com a ambiguidade;
- Habilidade de lidar com problemas "*open-ended*"[15];
- Deixar de lado diferenças para trabalhar com outros a fim de atingir um objetivo comum;
- Reconhecer suas fraquezas e fortalezas quando trabalhando com outros.

Os participantes da pesquisa também tentaram definir o tipo de cultura de sala de aula mais propício para o desenvolvimento do Pensamento Computacional, identificando estratégias e características:

- Uso do vocabulário computacional, quando apropriado para descrever problemas e soluções;

[15] Problemas que apresentam diversas respostas corretas e diversas maneiras e encontra-las.

- Aceitação de tentativas frustradas de solução reconhecendo que o erro pode ajudar a encontrar o caminho do sucesso;
- Trabalho em equipe com os estudantes, com o uso explícito de:
    - Decomposição: decompor problemas em partes pequenas, mais fáceis de resolver;
    - Abstração: simplificação do concreto ao geral, enquanto as soluções são desenvolvidas;
    - Negociação: grupos trabalhando juntos para unir as partes da solução para uma solução geral;
    - Construção de consensos: trabalhar para construir solidariedade do grupo para uma ideia ou solução.

O projeto para definir detalhes e planos de ação para a implementação deste modelo estava em andamento no momento desta revisão bibliográfica, mas já demonstram ser possível utilizar estas capacidades, disposições, predisposições e cultura de sala de aula como um ponto de partida para promover o Pensamento Computacional na escola (BARR, STEPHENSON, 2011; ISTE, 2011).

Analisando outros trabalhos, encontramos Grover e Pea (2013, p. 39, tradução nossa), que trazem sete "grandes ideias"[16] sobre o processo de computação (que destacam como uma perspectiva que representa o significado de Pensamento Computacional), especialmente para o currículo escolar do ensino médio. Essas ideias foram retiradas do curso de Princípios da Ciência da Computação[17], organizado pela National Science Foundation (NSF):

- Computação é uma atividade humana de criatividade;
- A abstração reduz informação e detalhes para focar em conceitos relevantes para o entendimento e resolução de problemas;
- Dados e informação facilitam a construção de conhecimento;
- Algoritmos são ferramentas para desenvolver e expressar soluções de problemas computacionais;
- Programação é um processo criativo que produz artefatos computacionais;
- Dispositivos digitais, sistemas e as redes que os interconectam habilitam e fomentam abordagens computacionais para a resolução de problemas; e

[16] Em inglês, provavelmente uma referência ao termo "*big idea*", de Seymour Papert.
[17] http://www.csprinciples.org/

- Computação habilita a inovação em outros campos, incluindo ciências, ciências sociais, áreas humanas, artes, medicina, engenharia e administração (business).

Outra abordagem comum para o Pensamento Computacional na Escola vem sendo feita no Reino Unido, por meio de quatro pilares (*cornerstones*): decomposição, reconhecimento de padrões, abstração e algoritmos (BRACKMANN, 2017).

2.5. Avaliando o desenvolvimento do Pensamento Computacional

A avaliação do desenvolvimento do Pensamento Computacional não é um processo trivial, e é um dos tópicos mais discutidos por educadores e pesquisadores da área (MORENOLEON, ROBLES, 2015, 2017; WING, 2017; GROVER, 2015, 2017; BASU et al., 2018).

Brennan e Resnick (2012) apontam o que deve ser observado ao realizar tal avaliação. Divididos em "conceitos encontrados", "práticas desenvolvidas" e "perspectivas formadas", os principais itens que identificam o desenvolvimento do Pensamento Computacional são listados abaixo (Tabela *1*, Tabela *2* e Tabela *3*).

Tabela 1 - Conceitos Computacionais (BRENNAN, RESNICK, 2012)

Conceito	**Descrição**
Sequências	Expressar uma atividade particular ou uma tarefa como uma série de passos.
Laços (*loop*)	Mecanismo utilizado para repetição de sequências de maneira mais sucinta.
Eventos	Situações que dão início a outras. (Um famoso exemplo é o evento de colisão de objetos. Em muitos jogos, quando dois objetos colidem, o jogador perde uma vida. A situação de colisão de objetos é o evento que dá início a perda de vida do jogador.)
Paralelismo	Sequências que acontecem ao mesmo tempo.
Condicionais	Habilidade de fazer decisões baseado em certas condições, havendo a possibilidade de múltiplos resultados. Destacam-se os condicionais lógicos "se... então..." e "se e somente se".
Operadores	Operadores matemáticos (adição, subtração, multiplicação, etc), lógicos (e, ou) e operadores de cordas (strings).
Dados (*data*)	informação que o computador armazena, recupera e atualiza.

Tabela 2- Práticas computacionais (BRENNAN, RESNICK, 2012)

Prática 1: Ser incremental e iterativo. Programar não é um processo sequencial. É um processo adaptativo no qual o plano pode mudar de acordo com as necessidades.
Prática 2: Testar e depurar erros.
Prática 3: Reutilizar e *remixar* (em cima do trabalho de outros). Uma prática comum na programação é utilizar o programa de outras pessoas e adaptá-los para suas necessidades.
Prática 4: Abstrair e modularizar. Construir algo maior, juntando pequenas partes.

Tabela 3- Perspectivas computacionais (BRENNAN, RESNICK, 2012)

Perspectiva	
Expressar	"Um pensador computacional vê a computação como mais do que algo a ser consumido: a computação é algo que ele pode utilizar para se expressar" (BRENNAN, RESNICK, 2012).
Conectar	Compartilhar programações e a possibilidade de ter acesso à programação de outras pessoas abre novas perspectivas de conexão entre os pensadores computacionais.
Questionar	"Questionar envolve duvidar do que é dado, respondendo a essa dúvida por meio do design (de algoritmos)" (BRENNAN, RESNICK, 2012).

Neste trabalho, a avaliação do desenvolvimento do Pensamento Computacional foi realizada verificando a presença destes conceitos e práticas, e os objetivos serão elaborados relacionando o Pensamento Computacional e a Taxionomia de Bloom revisada, que será descrita abaixo.

2.6. Taxionomia adaptada ao Pensamento Computacional

Taxionomia é a ciência da classificação. Na área da Educação, encontramos algumas taxionomias específicas desenvolvidas para classificar, identificar e hierarquizar objetivos educacionais, como é o caso da Taxionomia de Bloom (BLOOM et al, 1956), da Taxionomia de Bloom Revisada (ANDERSON et al, 2001), da Taxionomia SOLO (BIGGS e COLLINS, 1982) e da Taxionomia do Aprendizado Significativo (FINK, 2013).

Ferraz e Belhot (2010, p. 422) destacam duas vantagens de se utilizar taxionomias no contexto educacional:

- Oferecer a base para o desenvolvimento de instrumentos de avaliação e utilização de estratégias diferenciadas para facilitar, avaliar e estimular o desempenho dos estudantes em diferentes níveis de aquisição de conhecimento; e
- Estimular os educadores a auxiliarem seus discentes, de forma estruturada e consciente, a adquirirem competências específicas a partir da percepção da necessidade de dominar habilidades mais simples (fatos) para, posteriormente, dominar as mais complexas (conceitos).

A seguir, apresentamos a Taxionomia de Bloom e suas relações com o Pensamento Computacional.

2.6.1. Taxionomia de Bloom

A Taxionomia de Bloom é o resultado de esforços da *American Psychological Association* para produzir um sistema de classificação hierárquica de objetivos educacionais (COFFEY, 2008; SEAMAN, 2011). Deste trabalho, consideraram-se três domínios específicos do desenvolvimento: cognitivo, afetivo e psicomotor e, apesar de todos os domínios terem sido amplamente discutidos, o domínio cognitivo é o mais conhecido e utilizado (FERRAZ e BELHOT, 2010).

O domínio cognitivo da Taxionomia de Bloom está relacionado ao aprender, dominar um conhecimento. Ele envolve o desenvolvimento intelectual, de habilidades e de atitudes. Nesse domínio, os objetivos foram agrupados em seis categorias (Figura *3*) que estão organizadas em uma hierarquia: para ascender à próxima, é necessário ter um desempenho adequado na categoria anterior (FERRAZ, 2010; BLOOM et al, 1956).

Figura 3 - Categorias do domínio cognitivo da Taxionomia de Bloom.

A categoria *Conhecimento* diz respeito a lembrar de informações já aprendidas, como fatos, datas, palavras, métodos, entre outros (FERRAZ e BELHOT, 2010; WILSON, 2016).

A categoria *Compreensão* é a habilidade de compreender e construir significado do conteúdo (WILSON, 2016). É a capacidade de "entender a informação ou fato, de captar seu significado e de utilizá-la em contextos diferentes" (FERRAZ e BELHOT, 2010, p. 426).

A categoria *Aplicação* é "a habilidade de utilizar conteúdo aprendido, ou de implementar conteúdo em situações novas e concretas" (WILSON, 2016, p.2, tradução nossa).

A categoria *Análise* consiste em "subdividir o conteúdo em partes menores com a finalidade de entender a estrutura final" (FERRAZ e BELHOT, 2010, p. 426).

Síntese consiste em agregar e juntar partes com a finalidade de criar um novo todo (FERRAZ e BELHOT, 2010) e *Avaliação* é a habilidade de "julgar, verificar e até criticar o valor de um conteúdo para certo propósito" (WILSON, 2016, p.2, tradução nossa).

2.6.2. Taxionomia de Bloom Revisada

Em 2001, foi publicada uma revisão sobre a Taxionomia de Bloom (ANDERSON, et al, 2001). Wilson (2016, p. 1, tradução nossa) afirma, sobre a revisão da Taxionomia de Bloom: "a maior diferença está nos acréscimos mais úteis e compreensivos de como a Taxionomia intersecta e age sobre diferentes tipos e níveis de conhecimento – factual, conceitual, procedural e metacognitivo".

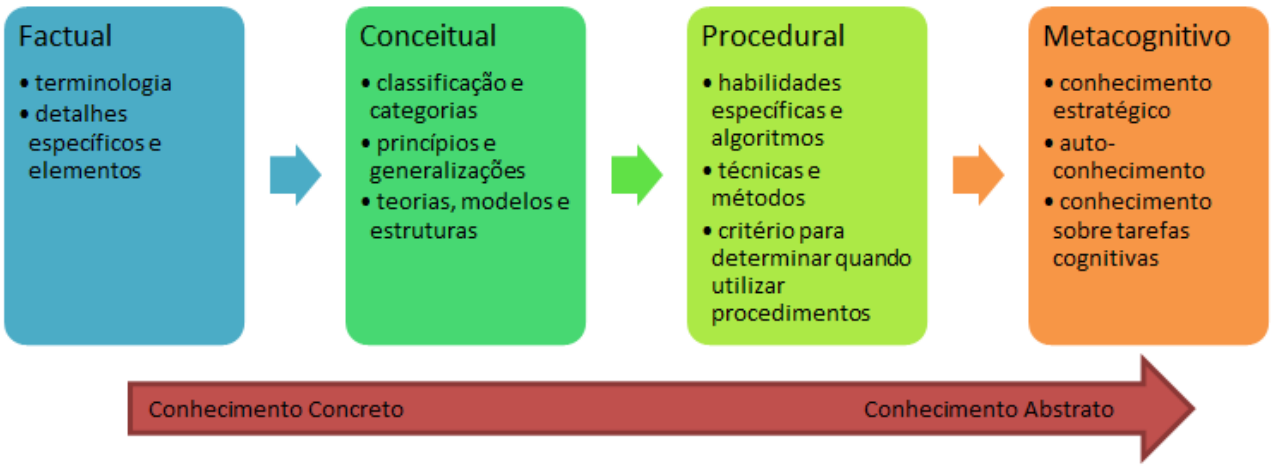
Isso deu origem a uma mudança na categoria "conhecimento", integrando os conceitos de conhecimento efetivo, conhecimento conceitual, conhecimento procedural e conhecimento metacognitivo à Taxionomia de Bloom. Segundo Ferraz e Belhot (2010, p. 428), na Taxionomia Revisada de Bloom, o conhecimento pode ser dividido em:

- Conhecimento Factual/Efetivo: relacionado ao conteúdo básico que o discente deve dominar a fim de que consiga realizar e resolver problemas apoiados nesse conhecimento; relacionado aos fatos que não precisam ser entendidos ou combinados, apenas reproduzidos como apresentados; conhecimento da terminologia; e conhecimento de detalhes e elementos específicos.
- Conhecimento Conceitual: relacionado à inter-relação dos elementos básicos em um contexto mais elaborado que os discentes seriam capazes de descobrir. Elementos mais simples foram abordados e agora precisam ser conectados. Esquemas, estruturas e modelos foram organizados e explicados. Nessa fase, não é a aplicação de um modelo que é importante, mas a consciência de sua existência; conhecimento de classificação e categorização; conhecimento de princípios e generalizações; e conhecimento de teorias, modelos e estruturas.
- Conhecimento Procedural: relacionado ao conhecimento de "como realizar alguma coisa" utilizando métodos, critérios, algoritmos e técnicas. Nesse momento, o conhecimento abstrato começa a ser estimulado, mas dentro de um contexto único e não interdisciplinar. Conhecimento de conteúdos específicos, habilidades e algoritmos; conhecimento de técnicas específicas e métodos; e conhecimento de critérios e percepção de como e quando usar um procedimento específico.
- Conhecimento Metacognitivo: relacionado ao reconhecimento da cognição em geral e da consciência da amplitude e profundidade de conhecimento adquirido de um determinado conteúdo. Em contraste com o conhecimento procedural, esse conhecimento é relacionado à interdisciplinaridade. A ideia principal é utilizar

conhecimentos previamente assimilados (interdisciplinares) para resolução de problemas e/ou a escolha do melhor método, teoria ou estrutura. Conhecimento estratégico; conhecimento sobre atividades cognitivas incluindo contextos preferenciais e situações de aprendizagem (estilos); e autoconhecimento.

A Figura *4* compila os principais conhecimentos de cada subcategoria.

Figura 4 - A dimensão Conhecimento (adaptada de KRATHWOHL, 2002).



Dessa maneira, na Taxionomia de Bloom Revisada, a categoria "conhecimento" passa a ser uma dimensão, transversal à dimensão do domínio cognitivo da Taxionomia de Bloom, que também sofre modificações.

As categorias do domínio cognitivo também sofrem mudanças em relação à Taxionomia de Bloom original (1956), conforme apresentado na Tabela *4*.

Como na Taxionomia de Bloom original, a Taxionomia de Bloom Revisada é hierárquica (na ordem de complexidade: relembrar, entender, aplicar, analisar, avaliar e criar). No entanto, essa hierarquia não é tão rigorosa, permitindo que as categorias se sobreponham (KRATHWOHL 2002, p. 215).

Tabela 4- Estrutura da dimensão do processo cognitivo da Taxionomia Revisada de Bloom (adaptada de KRATHWOHL, 2002, tradução nossa).

Categoria	Descrição e subcategorias
1. Relembrar	Recordar um conhecimento relevante da memória de longo prazo. 1.1. Reconhecendo 1.2. Recordando
2. Entender	Determinar o significado de mensagens instrucionais, incluindo a comunicação oral, escrita e gráfica. 2.1. Interpretando 2.2. Exemplificando 2.3. Classificando 2.4. Sumarizando 2.5. Inferindo 2.6. Comparando 2.7. Explicando
3. Aplicar	Realizar ou utilizar um procedimento em uma situação proposta. 3.1. Executando 3.2. Implementando
4. Analisar	Decompor um material a suas partes constituintes e detectar como as partes se relacionam entre si e com uma estrutura ou propósito geral. 4.1. Diferenciando 4.2. Organizando 4.3. Atribuindo
5. Avaliar	Fazer julgamentos baseado em critérios e padrões. 5.1. Checando (ou verificando) 5.2. Criticando
6. Criar	Juntar elementos para formar um novo e coerente todo ou produzir um produto original. 6.1. Gerando 6.2. Planejando 6.3. Produzindo

Uma pesquisa realizada por Garavaglia e colaboradores (1999) mostrou que os estudantes lembram mais quando aprenderam a lidar com o assunto nos níveis mais altos da taxionomia. Para tanto, isto não requer que tenham necessariamente se apropriado dos níveis

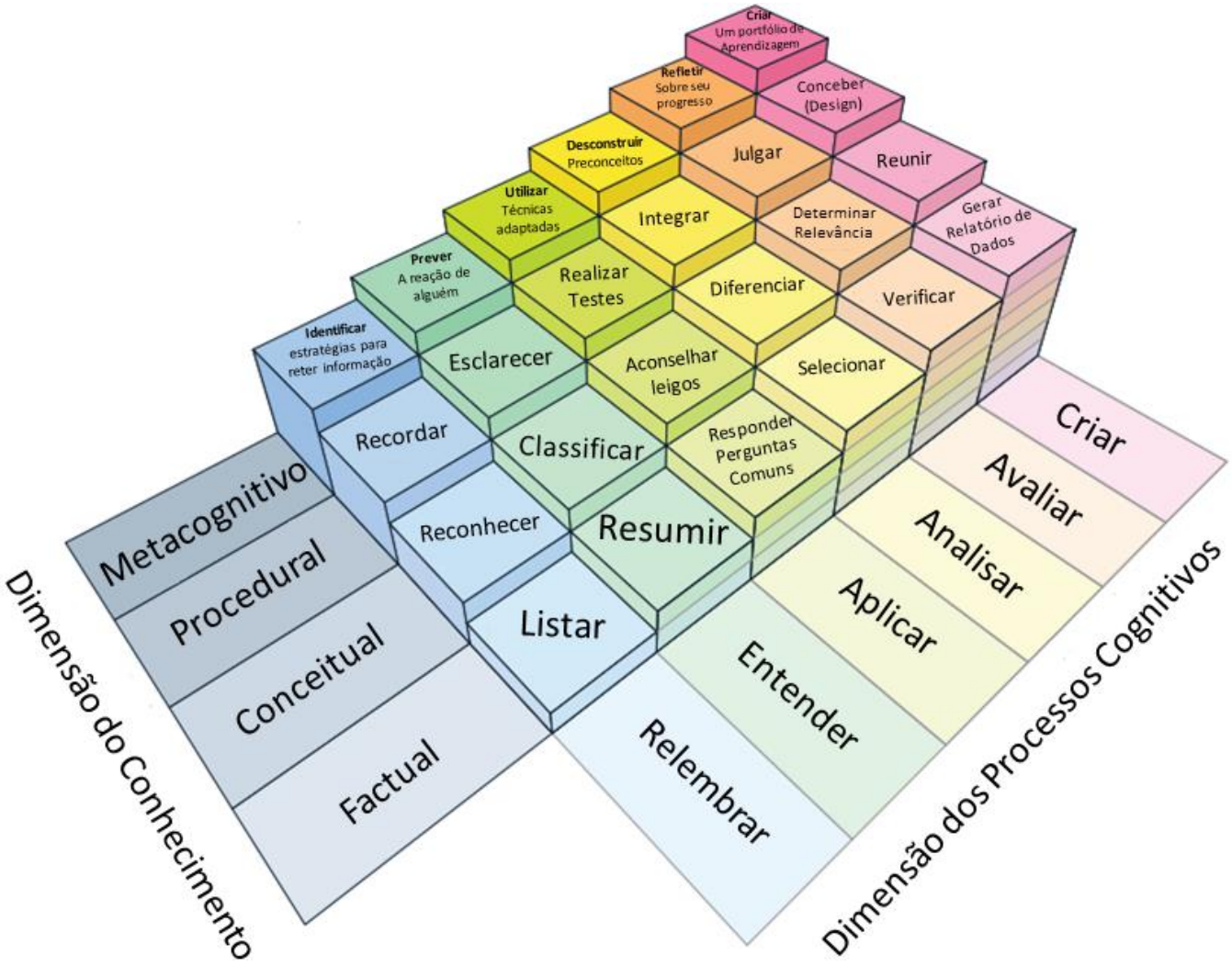
anteriores. Com efeito, isso pode ocorrer após terem atingido os níveis mais altos. Em outras palavras, um nível mais alto pode ser atingido antes de ter atingido o anterior.

Ferraz e Belhot (2010) chamam a atenção para o fato de que:

> "embora a nova taxionomia mantenha o "design" hierárquico da taxionomia original, ela é flexível, pois possibilitou considerar a possibilidade de interpolação das categorias do processo cognitivo quando necessário, devido ao fato de que determinados conteúdos podem ser mais fáceis de serem assimilados a partir do estímulo pertencente a uma mais complexa. Por exemplo, pode ser mais fácil entender um assunto após aplicá-lo e só então ser capaz de explicá-lo."

A Figura *5* mostra um modelo, adaptado de Heer (2012), que mostra como a dimensão conhecimento se relaciona com a dimensão do processo cognitivo na Taxionomia de Bloom Revisada. Cada bloco colorido mostra um exemplo de objetivo de aprendizagem que corresponde à combinação das dimensões do conhecimento e dos processos cognitivos.

Figura 5 – Modelo bidimensional da Taxionomia de Bloom Revisada (adaptado de HEER, 2012).



2.6.3. Relações entre a Taxionomia de Bloom e o Pensamento Computacional

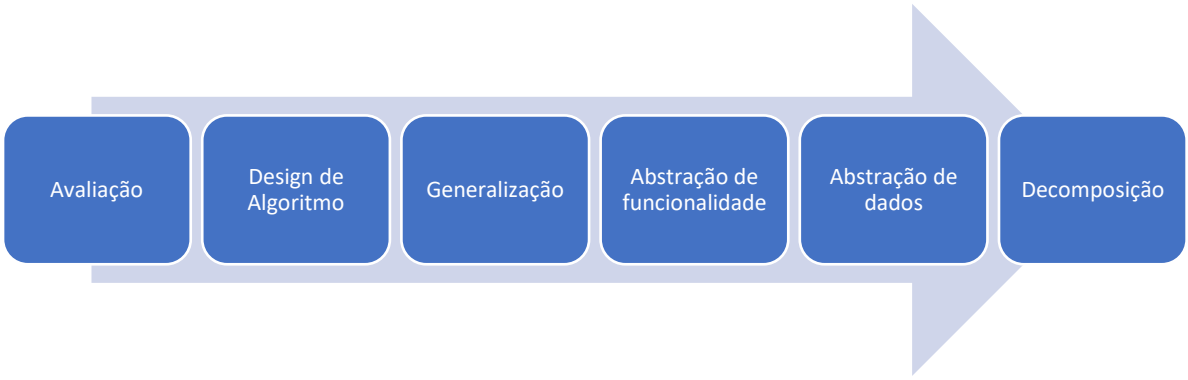
Selby (2015) apresenta as relações entre a Taxionomia de Bloom (1956) e o Pensamento Computacional no contexto do ensino da programação (pedagogia da programação). Esta relação está apresentada na Tabela *5* e e identifica a relação entre as categorias do domínio cognitivo de Bloom e os itens do ensino da programação.

Tabela 5 - Níveis de Bloom no ensino da programação (SELBY, 2015, p.85, tradução nossa)

Taxionomia de Bloom	**Ensino da Programação**
Avaliação	Avaliar, testar
Síntese	Criar programas, design de algoritmos
Análise	Abstrair, decompor, discriminar
Aplicação	
Compreensão	Estruturas, constructos, tipos
Conhecimento	

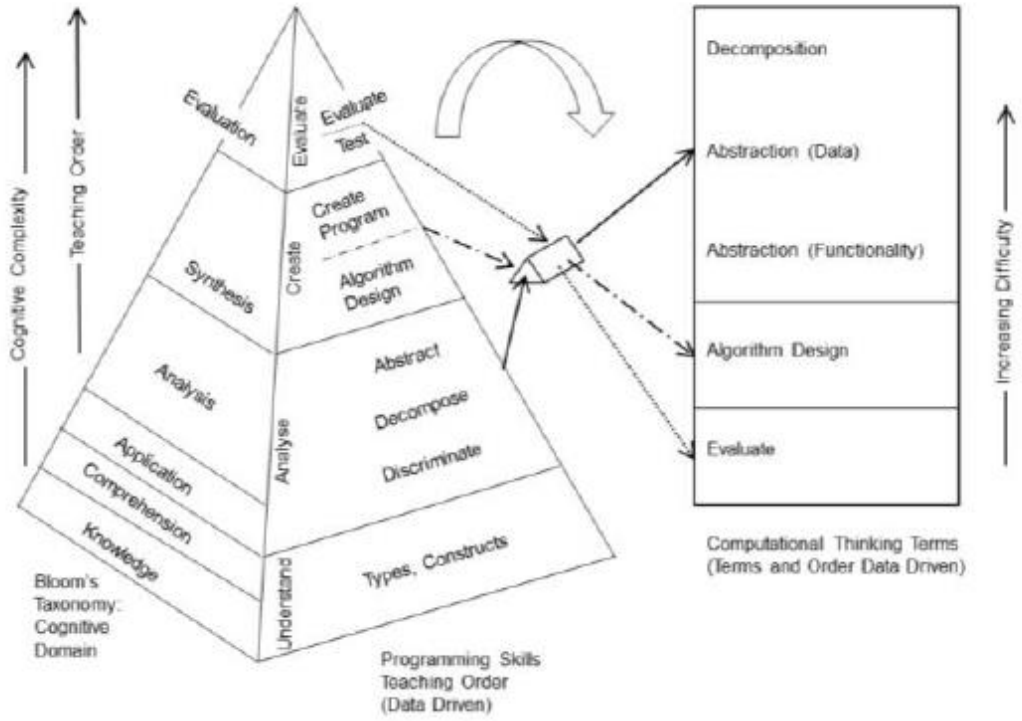
Analisando habilidades do Pensamento Computacional, Selby (2015, p. 85, tradução nossa) também identificou aquelas que são compreendidas como mais difíceis de dominar (Figura *6*), possibilitando uma relação entre a hierarquia de Bloom (1956) e as habilidades do Pensamento Computacional.

Figura 6 - Ordem de dificuldade do domínio das habilidades do Pensamento Computacional.



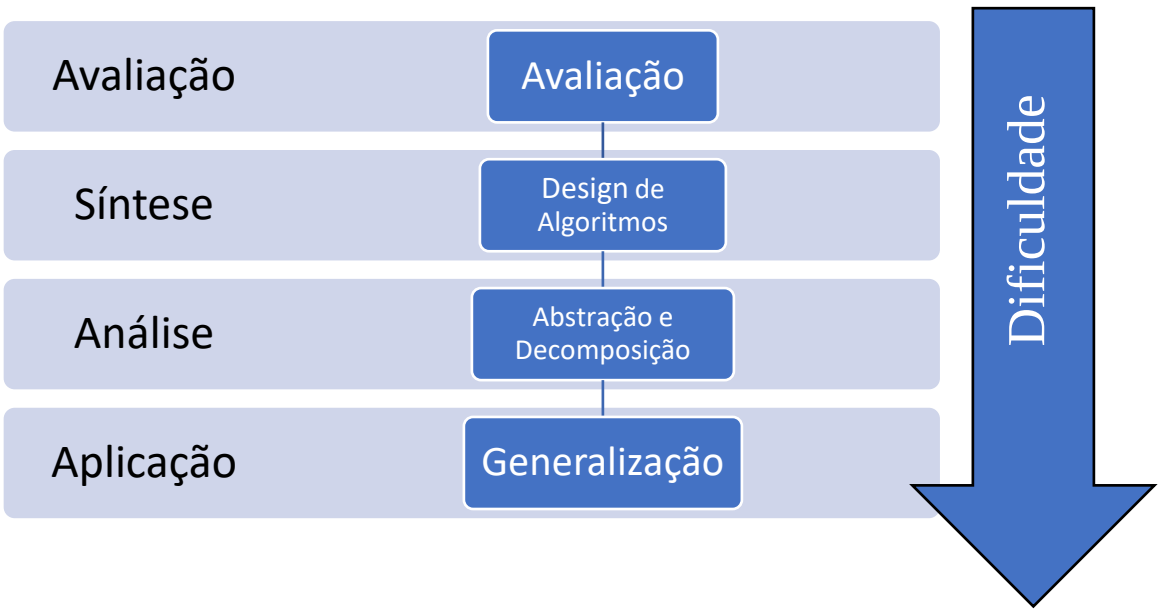
Contrariamente ao esperado, a ordem crescente de dificuldade observada para as habilidades do Pensamento Computacional é inversa às dos níveis de Bloom no ensino da programação, dando origem ao modelo apresentado na Figura *7*.

Figura 7 - Modelo: Pensamento Computacional, ensino da programação e a Taxionomia de Bloom (SELBY, 2015, p. 86).



Assim, é possível formalizar uma relação entre a Taxionomia de Bloom e as habilidades do Pensamento Computacional (SELBY, 2015), apresentada na Figura *8*.

Figura 8 - Relação entre a Taxionomia de Bloom e as habilidades do Pensamento Computacional.



A relação entre a Taxionomia de Bloom e o Pensamento Computacional vai além. Falloon (2016) desenvolveu em seu trabalho uma estrutura (Tabela *6* e Tabela *7*) que relaciona os conceitos, práticas e perspectivas do Pensamento Computacional de Brennan e Resnick (2012) com as da Taxionomia de Bloom Revisada (ANDERSON, et al, 2001).

Tabela 6 - Estrutura para análise de Fallon, parte 1 (adaptada, 2016, p. 7, tradução nossa)

	Conceitos do PC	**Práticas do PC**	**Perspectivas do PC**	**Atividade de Programação**
Relembrar (lembrando informações relacionadas ao que precisa ser feito)		Conceituando (a tarefa).	Compartilhando	Lembrando informação sobre a tarefa ou especificações.
Relembrar (lembrando informações relacionadas às funções, ferramentas e operações do aplicativo)		Conceituando (as ferramentas e recursos).	Compartilhando	Lembrando informações sobre as ferramentas, blocos ou funções.
Entender (desconstruindo a tarefa ou problema em etapas para ajudar no entendimento de sua resolução)		Conceituando (a tarefa).	Questionando Compartilhando	Entendendo ou esclarecendo as etapas necessárias para completar a tarefa.
Entender (interpretando atributos de resoluções bem-sucedidas e como estes serão avaliados)		Conceituando (a tarefa).	Questionando Compartilhando	Entendendo ou esclarecendo as especificações da tarefa e critérios de sucesso.
Aplicar e Criar (utilizando a estratégia tentativa e erro)	Sequências. Eventos. Trabalhar com dados, valores e variáveis.	Criando e testando programas utilizando estratégias não-incrementais e/ou não-iterativas.	Compartilhando	Aplicando conhecimento para criar e testar programas (estratégia "tentativa e erro").
Aplicar e Criar (utilizando uma estratégia incremental)	Sequências. Eventos. Trabalhar com dados, valores e variáveis.	Criando e testando programas utilizando estratégias incrementais e/ou iterativas.		Aplicando conhecimento para criar e testar programas (estratégia "passo-a-passo").

Tabela 7- Estrutura para análise de Fallon, parte 2 (adaptada, 2016, p. 7, tradução nossa)

Analisar (utilizando pensamento geral e conhecimento de computadores para entender os desafios, e o pensamento preditivo para identificar e corrigir possíveis erros, antes de criar e testar o programa)	Trabalhar com dados, valores e variáveis.	Conceituando tarefas e critérios, e identificando possíveis erros, antes de testar. Depurando erros, antes de testar.	Questionando	Analisando possíveis erros do programa e suas soluções antes de testar (preditivo)
Analisar (utilizando pensamento geral e conhecimento de computadores para analisar e corrigir erros depois de testar o programa)	Trabalhar com dados, valores e variáveis.	Conceituando tarefas e critérios, e identificando possíveis erros, depois de testar. Depurando erros, depois de testar.	Questionando	Analisando possíveis erros do programa e suas soluções depois de testar.
Avaliar (analisando o quão bem o resultado atende ao critério de sucesso)		Avaliando o resultado do programa.	Questionando	Avaliando (sem mudanças ou decisões).
Avaliar (analisando o quão bem o resultado atende ao critério de sucesso e modificar o programa, se necessário)		Avaliando o resultado do programa e sua modificação, se necessário.	Questionando	Avaliando (com mudanças ou decisões).

A existência de tais estruturas (Figura *7*, Tabela *5* e Tabela *6*) evidencia a possibilidade de utilizar a Taxionomia de Bloom para estruturar os objetivos de aprendizagem de atividades para desenvolver o Pensamento Computacional.

3. PROCEDIMENTOS METODOLÓGICOS

O método utilizado para esta pesquisa está detalhado a seguir, para subsequente análise dos resultados.

3.1. Delineamento da Pesquisa

A pesquisa desenvolvida foi uma abordagem qualitativa, aplicada na área de Ensino de Ciências e Matemática. Sua natureza é exploratória, pois se pretende por meio dela obter maior familiaridade com o problema investigado. Ela é também de natureza explicativa, pois se pretende identificar os aspectos que contribuem para o desenvolvimento do Pensamento Computacional, que é o fenômeno estudado (GERHARDT, SILVEIRA, 2009).

Os procedimentos desta pesquisa a caracterizam como participante, devido ao fato de que o pesquisador esteve ativamente envolvido de modo cooperativo nas situações pesquisadas (FONSECA, 2002).

3.2. Materiais

Para a realização da pesquisa utilizou-se como amostra uma turma de turno integral da escola Caminho do Saber. A escola e a turma com a qual foi realizada a pesquisa estão detalhadas na seção seguinte, juntamente com o *software* selecionado para as atividades.

3.2.1. A Escola

A pesquisa foi realizada com uma turma do turno integral da escola Caminho do Saber, uma escola particular de Ensino Fundamental e Médio, localizada na cidade de Caxias do Sul. A escola conta com aulas de educação tecnológica e robótica no currículo até o oitavo ano do Ensino Fundamental.

Com 1200 estudantes matriculados no turno curricular, é uma das maiores escolas da cidade. É uma escola que preza pela inovação e inserção de novas tecnologias nos processos de ensino e aprendizagem e, por isso, conta com lousas digitais instaladas em todas as salas de aula, laboratórios de informática, sala *Apple* (sala com *iPads* para uso pedagógico), ambiente virtual para postagem de trabalhos e materiais, laboratório de educação tecnológica e laboratório de robótica.

A escola trabalha com dois sistemas de ensino: Dom Bosco (Educação Infantil até oitavo ano do Ensino Fundamental) e Sistema Ari de Sá (nono ano do Ensino Fundamental até a terceira série do Ensino Médio). O Sistema Dom Bosco se alicerça em 3 perspectivas: o interacionismo, a interdisciplinaridade e o pensamento complexo. O Sistema Ari de Sá tem como missão prover soluções educacionais com excelência.

A escola Caminho do Saber é declaradamente sociointeracionista, com a seguinte proposta pedagógica (retirada do seu website):

> Baseada na teoria Sociointeracionista, os estudantes interagem uns com os outros, usufruindo de diferentes espaços de aprendizagem e recursos pedagógicos disponibilizados pela escola. Para que ocorra uma aprendizagem significativa, valorizamos o vínculo afetivo entre professor e estudante. Levamos em consideração os conhecimentos prévios dos estudantes: o professor associa os fatos da atualidade e organiza os conteúdos de forma interdisciplinar. Os estudantes aprendem, construindo e interagindo em todos os momentos da rotina escolar sem desprezar o que o ensino tradicional tem consolidado. Procuramos inovar o ensino de hoje para transformar o amanhã. O Sociointeracionismo é a melhor opção, pois mescla construção e interação em um ambiente propício ao desenvolvimento intelectual, onde o professor planeja, desafia, direciona, executa, avalia e corrige cada etapa de aquisição de conhecimento. (CAMINHO DO SABER, 2019)

3.2.2. Perfil do Estudante - Amostra

A turma escolhida para ser a amostra da pesquisa foi a turma do turno integral do quarto ao sétimo ano do Ensino Fundamental.

O turno integral[18] acontece no turno oposto ao curricular. Estudantes que frequentam a aula curricular pela manhã participam do turno integral no período da tarde, e vice-versa. Destacamos que a implementação da escola em turno integral abre espaço para o desenvolvimento de atividades diferenciadas que não afetem diretamente o conteúdo da aula curricular, e já está prevista no texto da Lei de Diretrizes e Bases da Educação Nacional (BRASIL, 1996): “Serão conjugados todos os esforços objetivando a progressão das redes escolares públicas urbanas de Ensino Fundamental para o regime de escolas de tempo integral.”.

A pesquisa foi realizada com 14 estudantes, que estudam no turno da tarde e participam do turno integral durante a manhã. Quanto à idade e ao ano escolar, os estudantes estão divididos conforme a Tabela *8*.

[18] Também chamado de “turno inverso”, “turno oposto”, entre outros.

Tabela 8 – Relação de estudantes: idade e ano escolar.

Idade	**09-10**	**10-11**	**11-12**	**12-13**
Ano Escolar	**4º**	**5º**	**6º**	**7º**
Número de Estudantes	8	2	3	1

Dos estudantes descritos na tabela, apenas aqueles do sexto e sétimo ano tiveram contato com experiências de programação, durante o turno curricular, nas atividades no laboratório de robótica.

Dentre as outras atividades desta turma de turno integral, estão: recreação e brincadeiras no pátio, horário para estudos e temas, biblioteca e hora da leitura, informática (recreativa), projetos sociais, horário de esportes, envolvimento em projetos da escola.

3.2.3. Scratch – A Ferramenta

Conforme mencionado na introdução, o *software Scratch* é uma ferramenta didática adequada para o trabalho com crianças. Sendo uma das principais ferramentas para o uso da programação em escolas, por ser gratuito e didaticamente adequado, o Scratch é a principal plataforma utilizada nesta pesquisa.

Para utilizar o *Scratch*, o usuário pode realizar o download para uso *offline* ou acessar pelo próprio navegador. Para ambos os casos, deve-se acessar o link https://scratch.mit.edu/.

Tutoriais sobre seu uso estão disponíveis no mesmo link, desenvolvido e traduzidos pela própria equipe do MIT.

3.3. Método

A partir das reflexões apresentadas na introdução deste documento, que evidenciaram a importância das habilidades do Pensamento Computacional para os estudantes do século XXI, foi selecionado o tema “Pensamento Computacional” como o fator determinante para esta pesquisa.

O desenvolvimento do Pensamento Computacional pode ser realizado por meio de ferramentas e plataformas de ensino de programação. As plataformas para o ensino de programação vêm sendo utilizadas em todo o mundo de maneira integrada ao ensino das

Ciências e da Matemática, mas também de outras áreas do conhecimento. A possibilidade de desenvolver animações, jogos e aplicações simples, porém interessantes, torna a ferramenta *Scratch* apta para ser usada por crianças em vários cenários.

O método empregado nesse trabalho resultou no desenvolvimento de uma sequência didática para o desenvolvimento do Pensamento Computacional. Algumas das atividades propostas foram adaptadas do guia de currículo *Creative Computing*, disponibilizado pelos membros da equipe de pesquisa *ScratchEd*, da *Harvard Graduate School of Education*. (BRENNAN, BALCH, CHUNG, 2014).

A escolha do grupo amostral como uma turma do turno integral foi natural, tendo em vista que as atividades propostas não foram pensadas para se encaixar nos conteúdos do calendário curricular proposto pela escola, mas sim para desenvolver habilidades nos estudantes. Isso possibilitou a fuga da sala de aula padronizada (idades iguais) e com conteúdos específicos, sem comprometer a expectativa de aprendizagem destes estudantes.

Para desenvolver as habilidades do Pensamento Computacional, uma revisão bibliográfica levou à escolha do software *Scratch.* Em seguida, as atividades da sequência didática foram elaboradas e logo aplicadas.

Foram avaliados os resultados obtidos pela aplicação da sequência em termos de aprendizado e desenvolvimento das habilidades do Pensamento Computacional, verificando a presença dos conceitos, práticas e perspectivas (Tabela *1*, Tabela *2* e Tabela *3*). Em cada encontro, foram analisados os programas desenvolvidos e as abordagens utilizadas pelos estudantes.

3.4. Descrição dos Encontros

Os encontros estão organizados na forma de plano de ensino. Este plano de ensino descreve os processos de ensino e de aprendizagem de uma sequência didática voltada ao desenvolvimento do Pensamento Computacional para crianças do turno integral de uma escola de Ensino Fundamental (estudantes dos anos escolares 4º ao 7º ano).

Os encontros foram realizados semanalmente com os estudantes. O tempo necessário estimado para cada atividade foi de 1 hora e 30 minutos, podendo variar em cada encontro.

Os objetivos de aprendizagem (Tabela *9* e Figura *9*) foram escritos levando em conta as duas dimensões da Taxionomia de Bloom Revisada (ANDERSON et al, 2001). A localização dos objetivos de aprendizagem na estrutura bidimensional da Taxionomia de Bloom foi

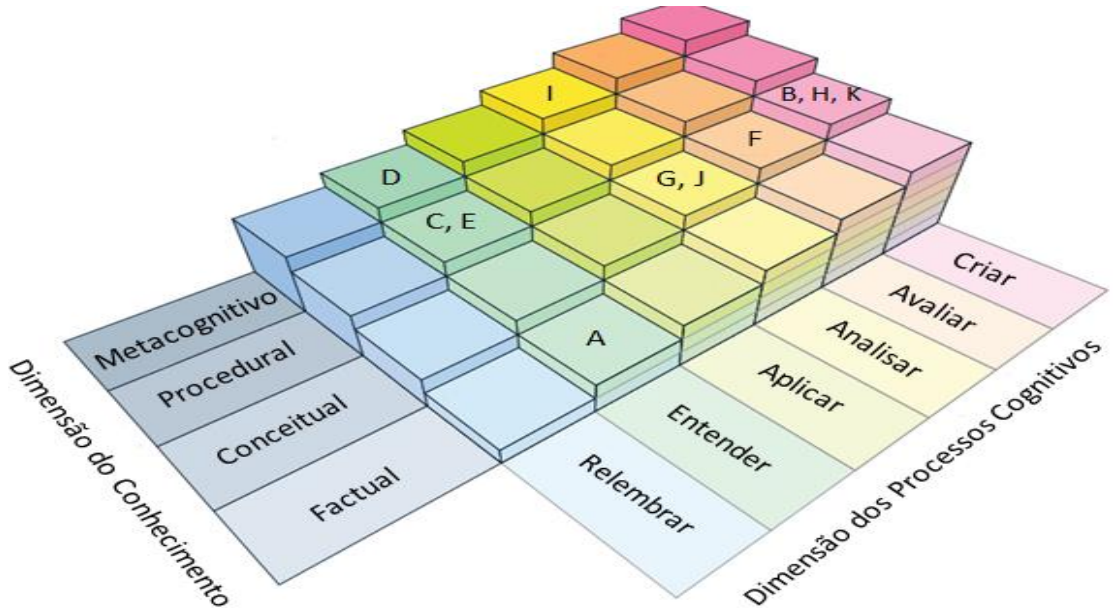
realizada a partir das dimensões dos Processos Cognitivos (Tabela *4*), e do Conhecimento (Figura *4*).

Por exemplo, o objetivo A (Localizar e identificar blocos básicos de programação do *Scratch*) caracteriza determinar o significado de uma comunicação gráfica (Entender), interpretando os blocos básicos de programação (Conhecimento Factual).

Tabela 9- Organização dos Encontros e Objetivos de Aprendizagem

	Espera-se que os estudantes estejam aptos para
Encontro 1 – Meu primeiro programa	A) Localizar e identificar blocos básicos de programação do *Scratch*. B) Programar uma sequência básica no *Scratch*. C) Descrever como se constrói um algoritmo.
Encontro 2 – Let's Dance	D) Discutir a importância de ser claro e objetivo. E) Traduzir uma ação como uma sequência de passos.
Encontro 3 – Debug it!	F) Avaliar o funcionamento de algoritmos básicos no *Scratch*. G) Testar algoritmos no *Scratch*.
Encontro 4 – Jogo de Corrida	H) Desenvolver um jogo de corrida no *Scratch*.
Encontro 5 – Hora do Código	I) Examinar e comparar outras plataformas de programação em blocos.
Encontro 6 – Debug it! 2.0	F) Avaliar o funcionamento de algoritmos básicos no *Scratch*. G) Testar algoritmos no *Scratch*.
Encontro 7 – Projeto Colaborativo	J) Examinar a programação de outras pessoas, com a finalidade de continuá-las.
Encontro 8 – Jogo do Labirinto	K) Programar um jogo de labirinto no *Scratch*.

Figura 9 - Localização dos Objetivos de Aprendizagem na estrutura bidimensional da Taxionomia de Bloom Revisada.



3.5. Encontro 1 – Introdução: Construindo nosso primeiro programa

Ao iniciar uma sequência didática é importante apresentar aos estudantes a proposta das atividades, o seu objetivo, o método a ser utilizado e, especialmente se tratando de atividades que envolvem um software específico, é necessário que ele seja apresentado aos estudantes de maneira que não se sintam intimidados.

3.5.1. Descrição da Atividade

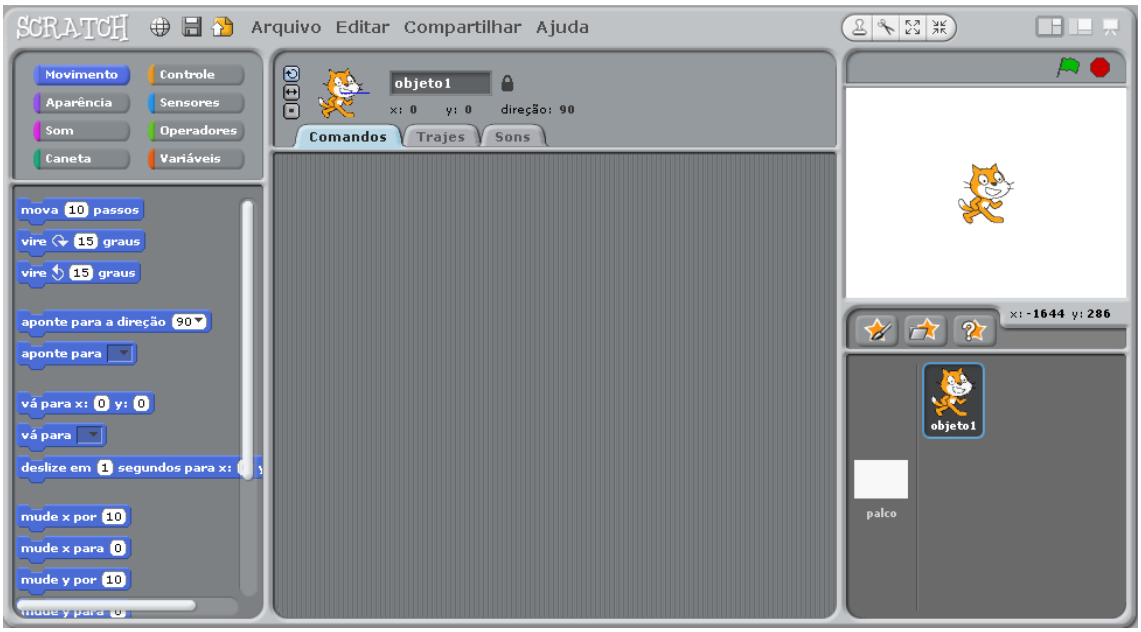
3.5.1.1. Introdução aos encontros

Aos estudantes foi explicado o funcionamento dos encontros: quando acontecerão, como acontecerão, por que estamos realizando essas atividades (objetivo) e o que é esperado deles. Da mesma maneira, os estudantes foram questionados em relação a suas expectativas.

3.5.1.2. Introdução ao *Scratch*

Utilizando um projetor foi apresentada aos estudantes a interface do software *Scratch* (Figura *10*).

Figura 10 - Interface do Scratch.



Foi realizada uma apresentação breve sobre como funciona o software e sua linguagem de programação em blocos.

3.5.1.3. Definindo o Problema

Foi a hora de construir o primeiro programa destes estudantes (daqueles que nunca tiveram experiências em programação): uma animação simples com a personagem gato do *Scratch*.

O programa foi construído por todos, com acompanhamento do professor, na tela que projetada. Em consenso, a turma decidiu fazer com que o gato do *Scratch* se movesse e, em seguida, falasse “Oi! Meu nome é Scratch!”

O professor falou aos estudantes sobre a estratégia de decompor o problema. Neste caso, a animação foi dividida em duas partes: movimentar a personagem e a fazer ela falar. Durante o processo de construção do programa, o professor encorajou os estudantes a utilizarem habilidades do Pensamento Computacional com questionamentos (vamos testar?, como podemos separar esta animação em partes?, algo está errado, será que conseguimos consertar?).

3.5.1.4. Parte 1: Movimentando o Objeto

O professor pediu aos estudantes que procurem nos blocos de programação e encontrem algum que possa fazer com que o objeto (gato) se movimente. Intuitivamente, os estudantes encontraram o bloco de movimento (Figura *11*).

Figura 11 - Bloco de movimento.

Na sequência, realizaram o teste, dando dois cliques no bloco. Eles observaram que isso movimentou a personagem no sentido que ela está virada. Testaram, então, o que acontece quando aumentam o número de passos.

Foi observado que ao mover o objeto desta forma, ele desaparece de onde estava e aparece na posição mais adiante, dependendo do valor estabelecido. Portanto, apenas este bloco não é suficiente para fazer a personagem caminhar.

Testaram, todos juntos, algumas sugestões, como o exemplos apresentado na Figura 12.

Figura 12 - Exemplo de programação.

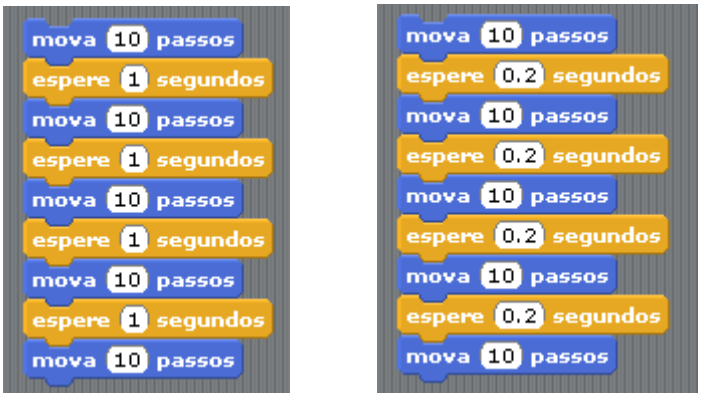
Depararam-se, então, com outro problema: é preciso que a personagem dê uma pausa antes de dar o próximo passo. Para isso, os estudantes foram desafiados a encontrar uma solução.

O bloco de espera foi uma sugestão dos estudantes. Então, tentaram colocar um bloco de espera entre os passos (Figura *13*).

Figura 13 - Programação com bloco de espera.

Agora a personagem parece estar caminhando. Os estudantes também decidiram ajustar o tempo de espera (Figura 14).

Figura 14 - Ajustes para a programação.



O professor ajudou a turma a descobrir que este processo pode ser sintetizado com o bloco de repetição (Figura *15*).

Figura 15 - Programação simplificada.

3.5.1.5. Parte 2: Fazendo a Personagem Falar

Para realizar a segunda parte do nosso problema, a turma explorou o software para encontrar uma solução. A solução da turma foi o bloco apresentado na Figura *16*.

Figura 16 - Bloco de fala.

Alternativamente, poderiam ter sido utilizados os blocos de som, com gravações.

3.5.1.6. Unindo as partes (síntese)

Fazendo a síntese das partes do problema, os estudantes construíram o programa proposto. Este programa está apresentado na Figura *17*.

Figura 17 - Síntese das partes.

3.5.1.7. Recursos Extras

De acordo com o ritmo da turma e a atenção que o professor pretende dar a cada detalhe, é possível que sobre tempo para explorar outras funções do programa.

Uma das funções que foi explorada é a de trajes (Figura *18*), para fazer o movimento parecer mais natural. Assim, a cada repetição, parece que o gato realizou o movimento de dar um passo.

Figura 18 - Função "trajes".

Foi possível também explorar os blocos de comando (Figura *19*) para iniciar o programa sem precisar dar dois cliques no programa.

Figura 19 - Blocos de comando.

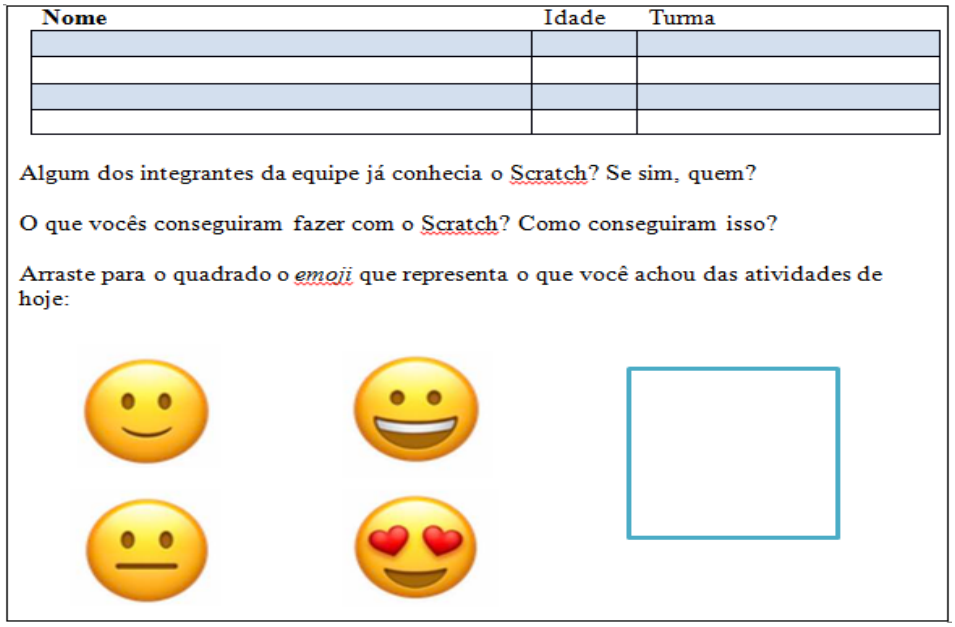
3.5.1.8. Exploração Livre

Depois que a turma programou uma pequena animação, foram ser encorajados a modificar esta, ou até mesmo criar uma diferente, com os recursos que quiserem. Foi o momento de explorarem os recursos do software e sua capacidade intuitiva de programação.

3.5.1.9. Apresentação e Relatório

Finalmente, os estudantes apresentaram seus programas para os colegas. Foram encorajados a explicar o que utilizaram de diferente, se encontraram muitas dificuldades e o que acharam de ser programadores.

Em seguida, foram solicitados a preencher um relatório para a coleta de dados.



3.6. Encontro 2 – Let's Dance

O Pensamento Computacional é uma habilidade que está presente também quando não estamos programando. Esta atividade iniciou com o desafio de descrever uma sequência de passos para uma dança (o termo utilizado com os alunos foi "programar uma pessoa"), e seguiu com a transposição deste desafio para a interface do *Scratch*.

A ideia de programar uma pessoa é simplesmente estabelecer uma sequência de passos para resolver um problema. Nesse caso, o problema foi ensinar uma dança a uma pessoa sem utilizar elementos visuais, apenas utilizando palavras.

3.6.1. Descrição da Atividade

Separados em duplas, foi designado a cada estudante um papel: programador ou programa. Cada dupla foi composta por um programador e um programa.

3.6.1.1. Vamos dançar!

Aos programadores foi mostrado um vídeo de dança. Cada programador recebeu o desafio de passar ao programa as instruções através da fala. Foi proibido ao programador utilizar gestos ou mostrar a dança de alguma maneira que não por comandos verbais.

O outro integrante, o programa, teve de realizar a dança conforme entendeu, e depois compararam seu resultado com a dança original.

Foi realizada uma pequena discussão sobre a importância da clareza quando estamos programando e sobre as diversas interpretações que certos termos podem ter.

3.6.1.2. Festa do *Scratch*!

Ainda separados em duplas, os estudantes foram desafiados a programar uma "festa do *Scratch*". Para isso, cada dupla seguiu o tutorial[19] do *Scratch* onde programam a animaç uma festa e, em seguida, foram encorajadas a personalizar seu projeto adicionando objetos, alterando a dança, o pano de fundo e explorando os sons. O desafio foi personalizar a animação de uma festa de dança.

No final, cada grupo apresentou sua animação para a turma.

[19] https://scratch.mit.edu/projects/editor/?tip_bar=getStarted

3.7. Encontro 3 – *Debug it*!

Uma das habilidades do Pensamento Computacional é a habilidade de depurar: encontrar erros e consertá-los. Com este objetivo, neste encontro foram propostos desafios, na forma de programas que estão com erros. Coube a cada grupo encontrar os erros e consertá-los.

3.7.1. Descrição da Atividade

Separados em grupos, os estudantes receberam os programas descritos abaixo, com seus respectivos erros (*bugs*). Eles tiveram de encontrar a parte do programa que está causando este erro e consertá-la.

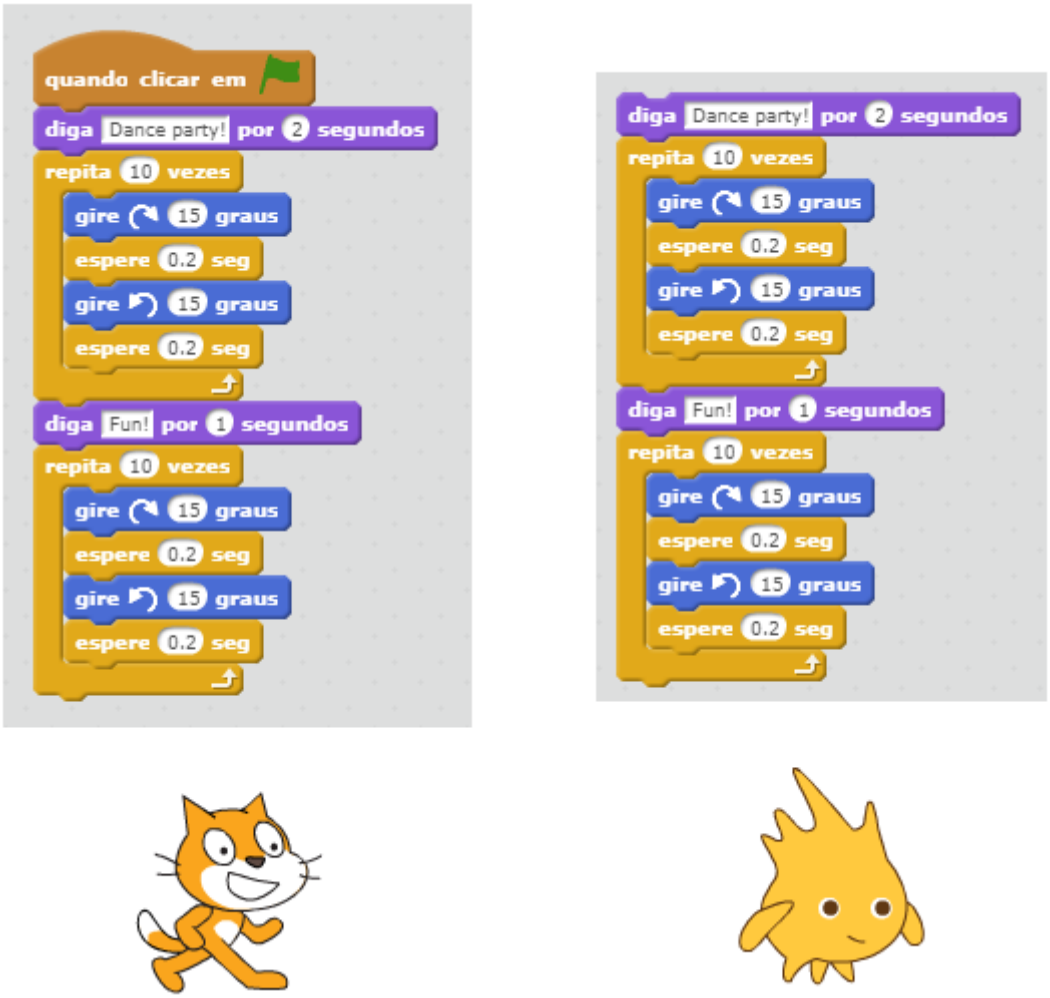
Foi solicitado que eles registrassem em um pequeno formulário qual era o erro do programa e como ele foi corrigido. Foi incluída neste texto uma possível solução, para facilitar que o leitor visualize a proposta.

3.7.1.1. Gobo está Parado!

Neste programa (Figura *20*), tanto o gato do *Scratch* quanto o Gobo (personagem amarelo) deveriam dançar quando a bandeira verde é clicada, mas isso não acontece. Cabe a cada equipe encontrar o erro e consertá-lo, do seu jeito.

Possível solução: O principal erro está na ausência do bloco "Quando clicar em bandeira verde" na programação do Gobo. Isso significa que não há um evento que dá início ao programa, que não vai fazer o Gobo dançar. Para resolver isso, uma alternativa simples é adicionar o bloco em questão no início da programação.

Figura 20 - Gobo está parado!



3.7.1.2. *Scratch* está com Preguiça?

O programador deste algoritmo (Figura *21*) queria que o gato do *Scratch* virasse uma cambalhota ao pressionar a tecla “espaço”, mas ele simplesmente não se move.

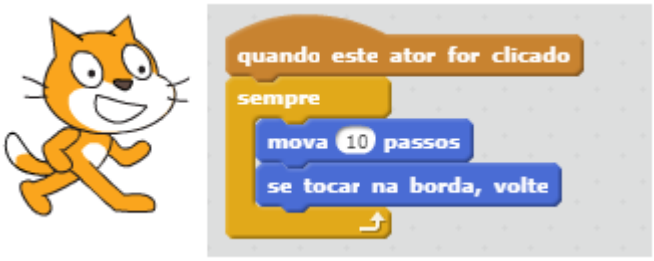
Figura 21 - Scratch está com preguiça?

Possível solução: Cada equipe deve compreender que o programa está certo, o gato do *Scratch* está girando 360°, mas ele faz isso tão rápido que não conseguimos perceber. Isso pode ser consertado, por exemplo, adicionando intervalos de tempo entre cada “gire 90 graus”.

3.7.1.3. De Cabeça para Baixo!

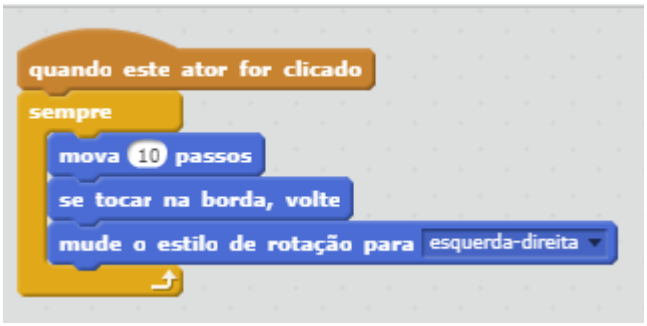
Neste projeto (Figura *22*), o gato do *Scratch* deveria ir de um canto ao outro da tela. O problema é que ele faz isso e quando volta, está de cabeça para baixo!

Figura 22 - De cabeça para baixo!



Possível solução: o principal problema deste programa pode ser resolvido com um bloco que caracteriza seu movimento (Figura *23*).

Figura 23 - Possível solução.



3.7.1.4. Meow!

A intenção do programa (Figura *24*) era que o gato do *Scratch* miasse de duas formas: com o balão de fala e através do som, que repetiria o miado 3 vezes. Algo no programa está fazendo com que ele reproduza o som somente depois do balão, e não ao mesmo tempo, e o som está sendo reproduzido somente uma vez.

Figura 24 - Programa *Meow!*

Possível solução: o bloco “toque o som meow” deve estar dentro do comando “repita 3 vezes”. Além disso, deve ser trocado pelo bloco “toque o som meow até o fim”, para que o programa não execute 3 miados ao mesmo tempo. Para fazer com que o som seja executado ao mesmo tempo dos balões, existem algumas alternativas como dividir o programa em dois, ou trocar o bloco “Diga meow, meow, meow! por 2 segundos” por “Diga meow, meow, meow!”.

3.8. Encontro 4 – Jogo de Corrida!

A criação de jogos como uma ferramenta educacional e motivacional foi um dos recursos a ser explorado neste encontro. A proposta foi simples: criar um jogo de corrida onde existirão duas ou mais personagens, e, quando uma determinada tecla for pressionada, essa personagem irá se mover.

3.8.1. Descrição da Atividade

A turma inteira e o professor realizaram a atividade em conjunto, com o auxílio do projetor.

Começaram escrevendo uma lista de coisas que devem acontecer neste jogo de corrida e descrevendo como ele vai funcionar.

Utilizando a habilidade de decomposição de problemas, dividiram o projeto em pequenas tarefas, para que depois elas formem a solução.

3.8.1.1. Determinando os Competidores e seus Movimentos

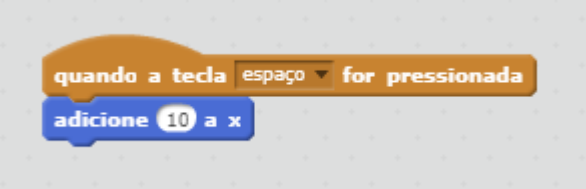
As personagens do jogo de corrida são os competidores. Eles podem ser representados por personagens do próprio *Scratch* ou podem ser desenhados com a função de edição do programa.

A turma optou por um círculo preto e um círculo verde, respectivamente objeto1 e objeto2 (Figura *25*).

Figura 25 - Objetos 1 e 2.

Cada objeto deve se movimentar uma determinada distância a cada vez que uma tecla for pressionada. Isso foi feito facilmente com o comando “adicione 10 a x” (Figura *26*).

Figura 26 - Comando: adicione 10 a x.



Para cada competidor, foi determinada uma tecla diferente do teclado: espaço e seta para a direita.

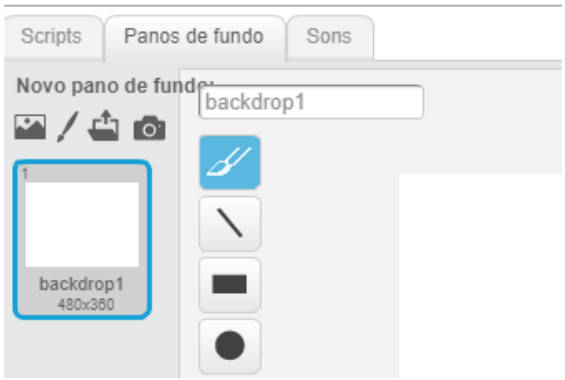
3.8.1.2. Pista de Corrida

Agora os competidores já "correm" pela tela. Foi necessário então, fazer a pista de corrida. Para isso, acessaram o palco, clicando ao lado dos atores, no local indicado (Figura *27*).

Figura 27 - Localização do palco.

Em seguida, selecionaram a aba "Planos de Fundo" (Figura *28*) e pintaram a pista de corrida.

Figura 28 - Aba de planos de fundo.



Pensando como deveria ser a pista de corrida, os estudantes levantaram alguns pontos importantes, como as cores, o formato e o tamanho, e como o programa irá detectar qual competidor ganhou a corrida.

Dessa maneira, concluíram que a pista de corrida deve ter uma cor que determine a linha de chegada, pois é a partir da cor que o programa vai detectar qual objeto encostou primeiro na linha. Optaram por um exemplo simples de pista de corrida: um cenário de uma cor com um retângulo de cor diferente (Figura *29*).

Figura 29 – Exemplo de pista de corrida.

3.8.1.3. Mecânica do Jogo

Definidos o cenário e os competidores, foi necessário determinar como estes irão interagir um com o outro. Mais especificamente, foi preciso encontrar uma maneira de determinar o vencedor.

Uma forma simples de fazer isso foi utilizando o sensor de cor do *Scratch*, que determina quando o ator está encostando na cor especificada, neste caso, a cor magenta. A turma e professor, então, exploraram os blocos de sensor e chegaram ao programa apresentado na Figura *30*.

Figura 30 - Utilizando o sensor de cor.

A turma escolheu que quando o competidor tocasse na cor magenta, um som fosse tocado.

3.8.1.4. Customização

A turma criou o "jogo base" de corrida em conjunto. Agora o desafio foi para que, separados em grupos de três componentes, os estudantes customizassem seus jogos com outras funções. No final da aula, cada grupo apresentou as suas mudanças.

Algumas sujestões de customização foram propostas:

- Adicionar função de voltar os competidores para a posição inicial (Figura *31*);

Figura 31 – Exemplo de customização para voltar à posição inicial.

- Adicionar anúncio de que o jogador venceu através de um evento, que repercute nos scripts dos outros objetos (Figura *32*);

Figura 32 – Exemplo de customização com anúncio do vencedor.

- Adicionar a possibilidade de se movimentar para cima e para baixo;
- Pista de corrida com curva.

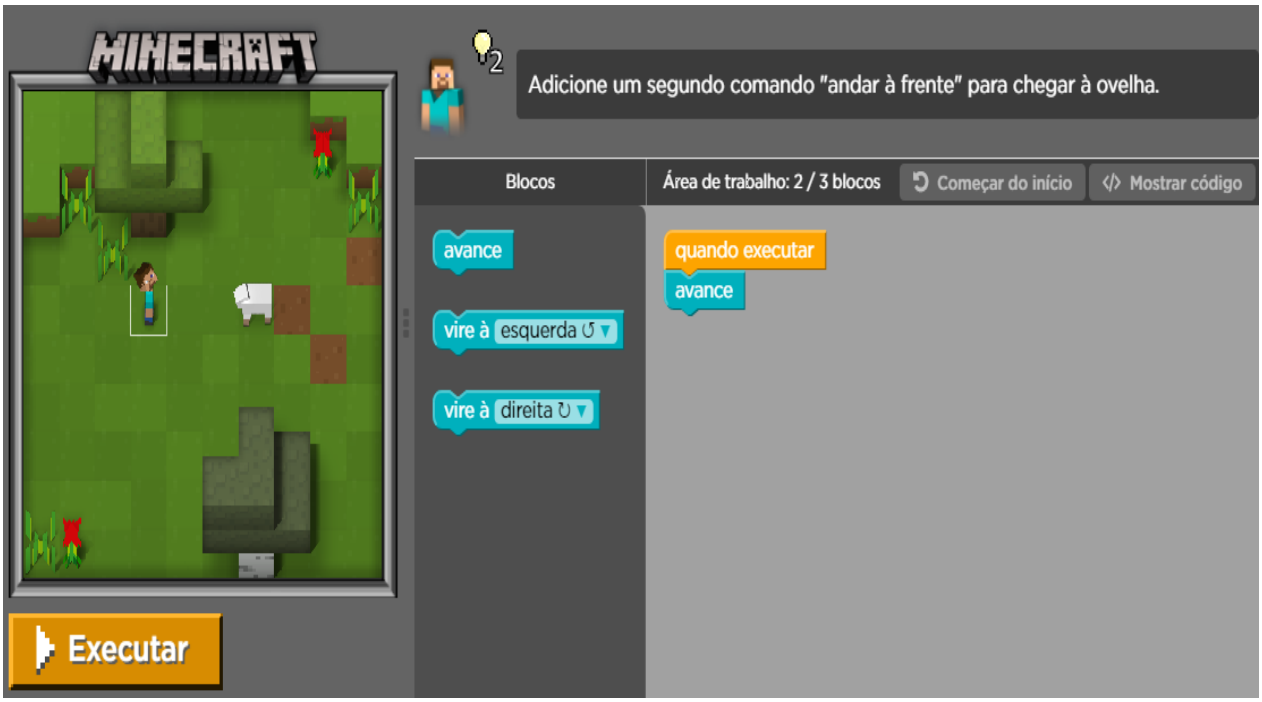
3.9. Encontro 5 - Hora do Código

A organização *code.org* é uma organização não lucrativa que se dedica a fazer a Ciência da Computação cada vez mais acessível nas escolas e ambientes de aprendizagem. Em seu *website*[20] encontramos diversas atividades, entre elas a Hora do Código.

A Hora do Código é um curso desenvolvido para que estudantes dediquem uma hora do seu dia para aprender a programação através de desafios simples que envolvem figuras de sua cultura (como, por exemplo, o jogo *Minecraft* e as personagens da animação *Frozen*).

A proposta desta atividade foi que os estudantes realizem a atividade "Aventureiro de *Minecraft*"[21]. Esta atividade é guiada pelo próprio jogo em sua interface (Figura *33*), que lança os desafios e explica como funciona o ambiente de programação.

Figura 33 - Interface *code.org*



O objetivo desta atividade foi exemplificar a diversidade de possibilidades com programação – dentre elas a programação de jogos muito populares. A inserção desta atividade na sequência didática surgiu da necessidade de mostrar outras formas de programar e para apresentar esta plataforma aos estudantes que demonstrarem interesse em explorar outros tipos de programação em casa.

[20] https://code.org/
[21] https://studio.code.org/s/mc/stage/1/puzzle/1

3.10. Encontro 6 – Debug it! 2.0

Com objetivos similares aos do terceiro encontro, foram propostos novos desafios para que os estudantes encontrem e depurem os erros de alguns programas

3.10.1. Descrição da Atividade

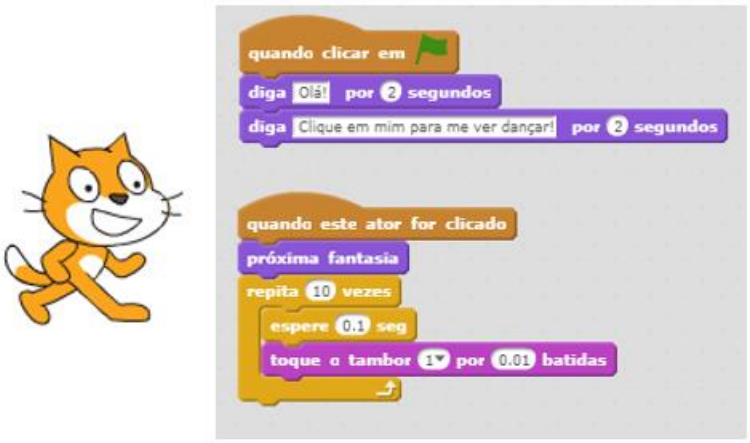
Separados em grupos, os estudantes receberam os programas descritos abaixo, com seus respectivos erros (*bugs*). Eles tiveram de encontrar a parte do programa que está causando este erro e consertá-la.

Foi solicitado que eles registrassem em um pequeno formulário qual era o erro do programa e como ele foi corrigido. Foi incluído neste texto uma possível solução, para facilitar que o leitor visualize a proposta.

3.10.1.1. Dança?

Na programação (Figura *34*), o gato do *Scratch* convida o usuário do programa para ver sua dança, mas ao clicar nele, ele apenas se move uma vez, e o som de tambores continua com ele parado. Como podemos consertar isso?

Figura 34 - Dança?

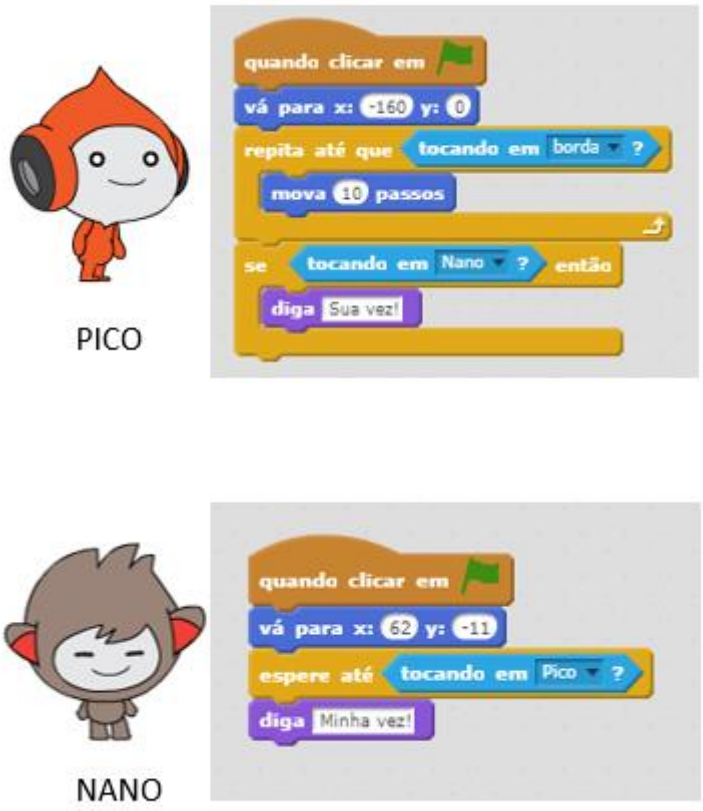


Possível solução: Podemos simplesmente colocar o bloco “próxima fantasia”, que dá o movimento ao gato, dentro do ciclo “repita 10 vezes”.

3.10.1.2. Pega-Pega

Na programação (Figura *35*), Pico e Nano estão brincando de pega-pega. Quando Pico encosta em Nano, deveria dizer “Sua vez!”, mas isso não acontece. O que deu errado?

Figura 35 - Pega-pega.

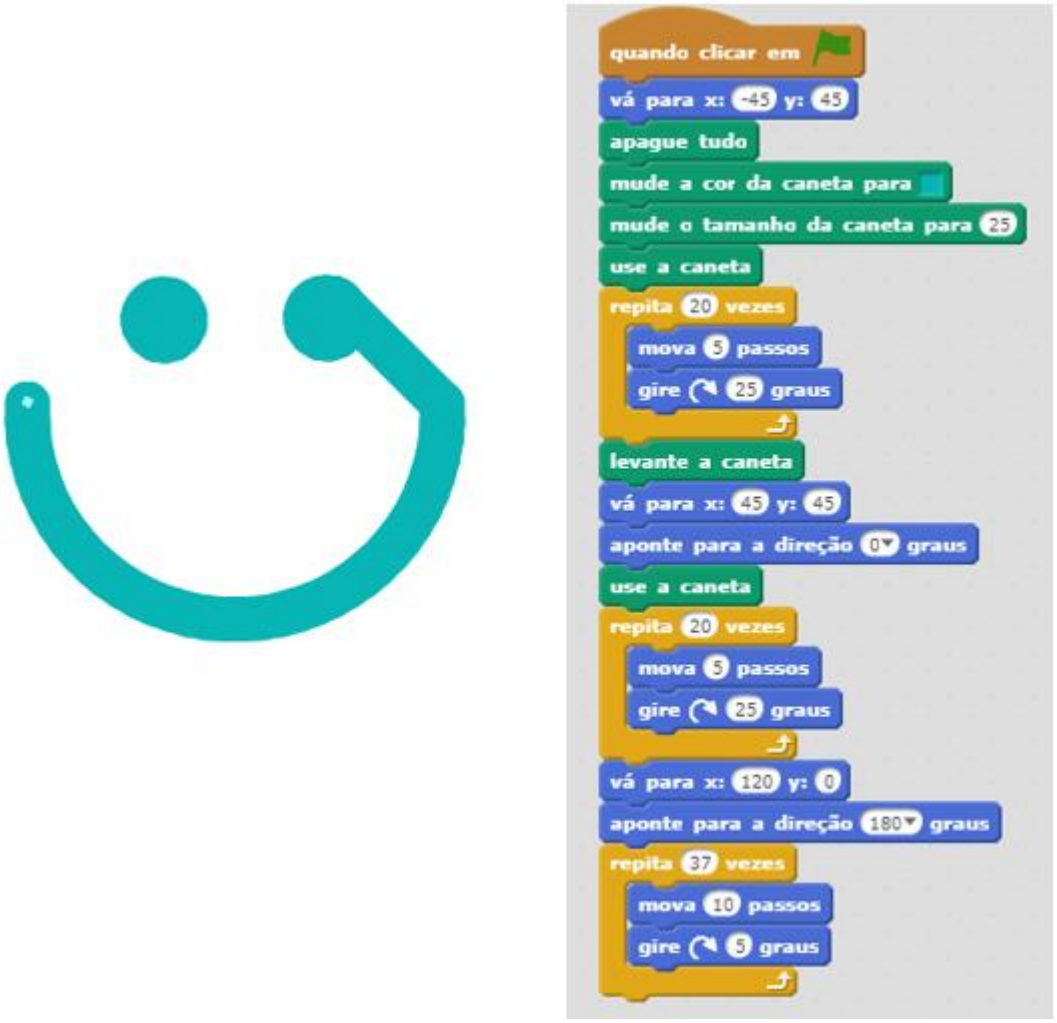


Possível solução: Pico estará “preso” no ciclo “repita até que tocando em borda”, e assim não executará a parte do programa “se tocando em Nano, então” até que seja tarde demais (depois de tocar na borda). Basta colocar o ciclo “se tocando em Nano, então diga Sua vez!” para dentro do ciclo “repita até que tocando em borda”.

3.10.1.3. Rosto Feliz

Este programa (Figura *36*) deveria desenhar um rosto feliz, mas ele está conectando um dos olhos à boca. O que podemos fazer?

Figura 36 - Rosto feliz.

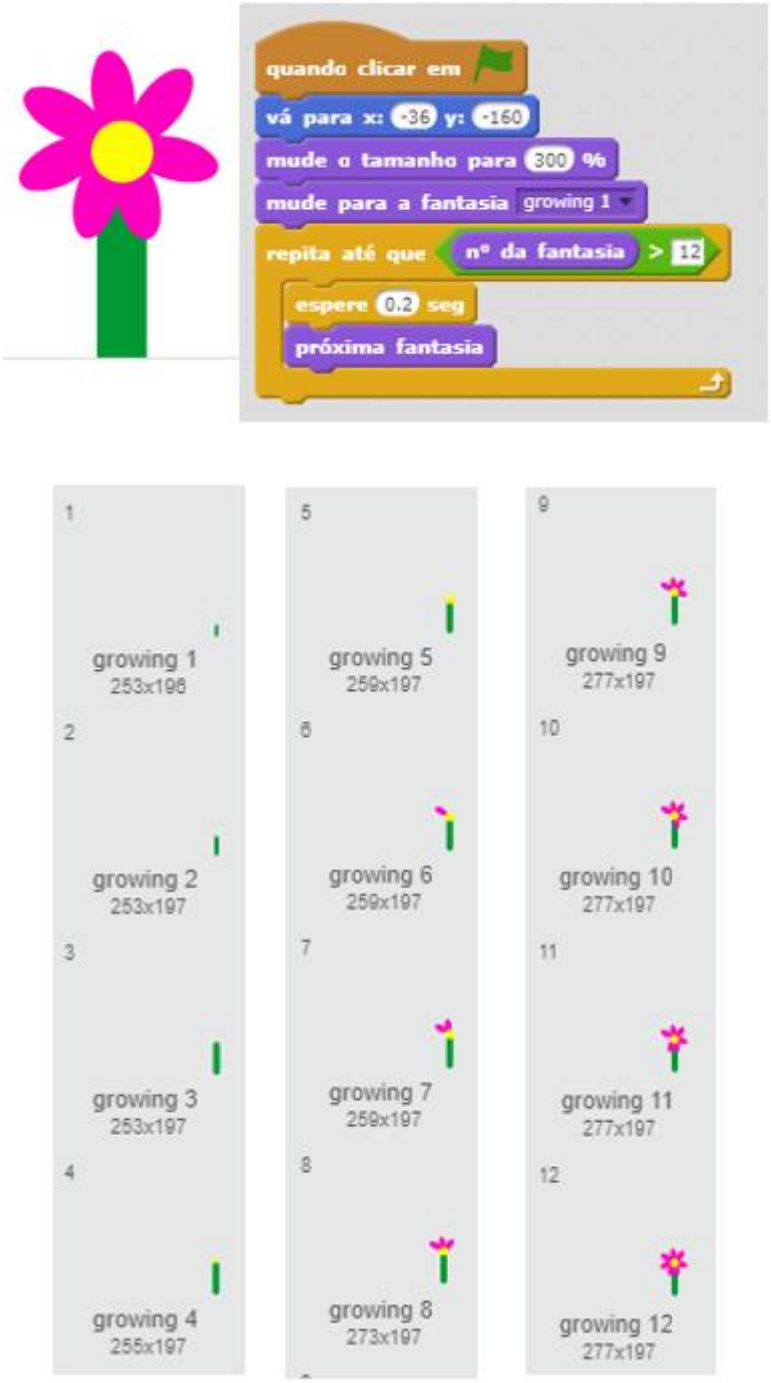


Possível solução: é preciso levantar a caneta ao ir da posição do olho até a posição onde a boca será desenhada. Para isso, inserimos blocos de “levante a caneta” e “use a caneta” nos locais adequados.

3.10.1.4. Florescendo

Este programa (Figura *37*) deveria parar quando a flor está crescida, mas ele continua repetindo a animação sem fim. Como podemos solucionar este problema?

Figura 37 – Florescendo.



Possível solução: A troca das fantasias da personagem será executada até que o número da fantasia seja maior do que 12. Como temos somente 12 fantasias, isso nunca irá acontecer. Para corrigir o problema, trocamos o número 12 por 11.

Podemos também trocar o bloco de operação “maior que” por “igual a”. Assim, quando o número da fantasia for exatamente 12, o programa irá parar.

3.10.1.5. Feliz Aniversário!

Esta animação (Figura *38*) deveria tocar o tema “parabéns pra você” e, ao acabar a música, deveria aparecer a instrução de “Clique para apagar as velas!”, mas essa instrução aparece durante a música. Como podemos consertar este erro?

Figura 38 - Feliz Aniversário!

Possível solução: trocar o bloco “toque o som birthday” por “toque o som birthday até o fim”.

3.11. Projeto Compartilhado

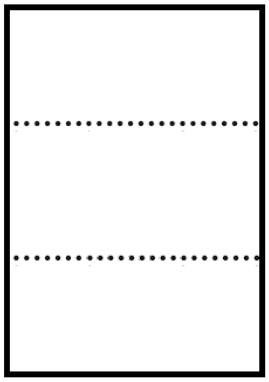
Nesta atividade, os estudantes foram desafiados a criar animações com a turma toda e cada um pode adicionar um pouco do que conhece e contribuir com cada animação. Antes disso, uma atividade introdutória foi realizada.

3.11.1. Descrição da Atividade

Para iniciar, os estudantes foram separados em duplas e trios. Cada grupo ficou com um computador para si. Foi distribuído para cada equipe uma folha tamanho A4 e uma caneta hidrográfica colorida (cada caneta com uma cor diferente).

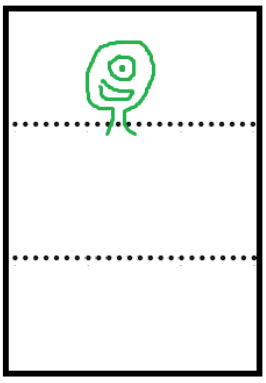
Os estudantes foram orientados a dobrar a folha em 3 partes (Figura *39*).

Figura 39 - Dobraduras da folha.



Nesta folha, foram solicitados a usar sua imaginação e desenhar na primeira parte a cabeça de um monstro, de maneira que o final do pescoço fique na segunda parte, conforme exemplo (Figura *40*).

Figura 40 - Exemplo de desenho de monstro.



Em seguida, tiveram que dobrar a folha, de maneira que seu desenho esteja escondido, e a turma fez um rodízio de desenhos. O desenho foi entregue ao próximo grupo. O grupo não podia espiar o trabalho do anterior, e teve agora que fazer um corpo para este monstro a partir do pescoço que foi deixado.

De maneira semelhante, fizeram até a cintura do monstro, deixando as pernas para a próxima equipe finalizar.

Ao final da atividade, foi feita uma reflexão de como o trabalho em grupo pode ser divertido, e sobre os desafios de continuar e finalizar o trabalho de outra pessoa.

Em seguida, os estudantes foram desafiados a fazer algo parecido, mas no *Scratch*! Os grupos deveriam iniciar uma animação no *Scratch*. Tiveram 10 minutos para iniciar o trabalho. Ao final destes 10 minutos, o professor anunciou que acabou o tempo e que deveriam trocar de mesa e continuar o trabalho do outro grupo. Repetiram o rodízio até retornarem ao seu computador.

No final, apresentaram no projetor os projetos, e os grupos que iniciaram cada animação descreveram sua ideia original e o que acharam do resultado final. Também falaram sobre a dificuldade de interpretar o projeto dos outros.

3.12. Encontro Final: Jogo do Labirinto

Foi lançado um desafio que une os conceitos que foram trabalhados durante os encontros anteriores. Nesse sentido, este encontro também serviu como uma avaliação do desenvolvimento da turma em relação uso do software e desenvolvimento de algoritmos.

Para este encontro, cada grupo deveria criar um jogo de labirinto no *Scratch*: um jogo onde você controla uma personagem que deve solucionar o caminho de um labirinto sem encostar na parede.

3.12.1. Descrição da Atividade

Separados em duplas e trios, os estudantes receberam o desafio de criar um jogo de labirinto, conforme orientações:

Uma personagem controlada pelo usuário deve percorrer um labirinto. Ao chegar no final, deve ser anunciado que o jogador venceu. Se o jogador encostar na parede, deve retornar ao início.

Os estudantes criaram os cenários e a personagem e utilizaram as cores para diferenciar a parede do labirinto. Foram estimulados a fazer o máximo que conseguirem e a customizar o jogo conforme acharem melhor.

Durante esta atividade o professor passou nos grupos para auxiliar, tomando cuidado para que apareçam as aprendizagens e conceitos das atividades anteriores.

4. RESULTADOS

Neste capítulo são apresentados os resultados desta pesquisa, a partir da realização da sequência didática descrita. No final de cada análise envolvendo o uso do software Scratch apresenta-se os conceitos, práticas e perspectivas do Pensamento Computacional (BRENNAN, RESNICK, 2012) identificadas em cada encontro.

Para avaliar estes conceitos, práticas e perspectivas, Brennan e Resnick (2012) apontam métodos como a Análise de Projetos, Entrevistas e Cenários de Design (*Design Scenarios*). Segundo os mesmos autores, nenhum destes, no entanto, se mostra suficiente para avaliar as três dimensões do Pensamento Computacional proposta por eles. A recomendação é, então, utilizar uma combinação dos métodos. Nesta dissertação, foram utilizadas a Análise de Projetos e Cenários de Design da maneira descrita a seguir:

Para avaliar os conceitos computacionais, através da Análise de Projetos, foi utilizada a tabela de conceitos computacionais de Brennan e Resnick (Tabela *1*), e a verificação foi realizada nos programas apresentados no final dos encontros, avaliando se o conceito está presente e também se foi utilizado de maneira correta.

As práticas computacionais (Tabela *2*) estão presentes enquanto as crianças estão criando o seu projeto (BRENNAN, RESNICK, 2012). A avaliação das práticas foi realizada durante as atividades por meio de Cenários de Design: cada encontro apresentava desafios particulares, com grau crescente de complexidade e, durante a resolução destes desafios, foram verificadas a presença ou não destas práticas, conforme a sua definição.

Avaliar as perspectivas (Tabela 3) apresenta desafios maiores, por se tratar de uma mudança interna, perceptível por meios sociais e psicológicos (BRENNAN, RESNICK, 2012). Nesse trabalho, as atividades da forma como foram propostas não contemplaram essa avaliação.

De forma complementar, descreve-se o desenvolvimento das atividades e percepções dos estudantes.

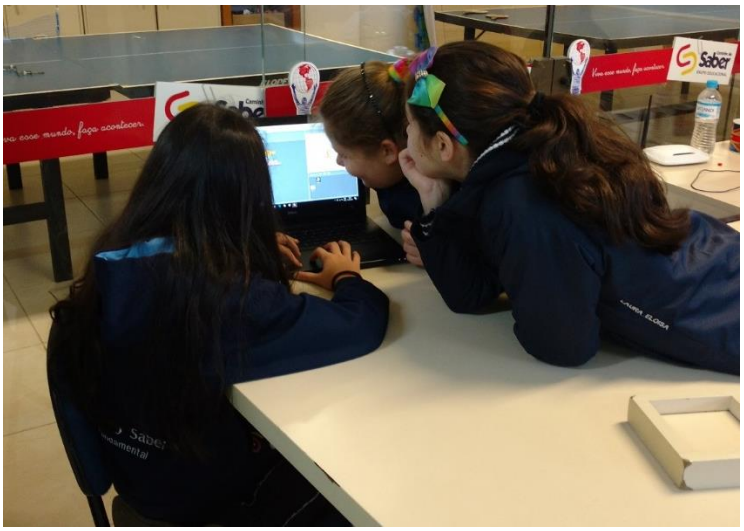
4.1. Encontro 1

Inicialmente foram apresentados os objetivos da proposta. Em seguida, foi apresentado o software *Scratch* e o conceito de programação e algoritmo. Junto com a turma, foi desenvolvido um programa básico: fazer o gato andar para frente e para trás, conforme ilustrado na Figura *41*.

Figura 41 - Programa desenvolvido pela turma, com o auxílio do professor.

Os estudantes foram divididos em grupos de 3 ou 4 integrantes. O restante do tempo foi destinado à exploração livre do *Scratch*, com o encorajamento de experimentação e criação de programas.

Figura 42 - Estudantes explorando o *Scratch.*

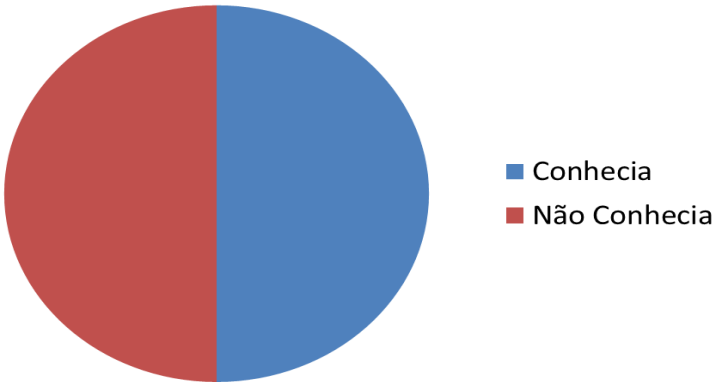


Em seguida, os estudantes preencheram o relatório proposto. O resultado deste relatório está descrito a seguir.

1 - Algum dos integrantes já conhecia o Scratch? Se sim, quem?

Fazendo o levantamento desta questão, encontramos que metade dos estudantes conhecia o *Scratch*, conforme apresentado na Figura *43*. No entanto, ao questioná-los sobre o grau de familiaridade com o *software*, constatou-se que o conhecimento era superficial.

Figura 43 - Gráfico de representação da experiência dos estudantes com o *Scratch.*



2 – O que vocês conseguiram fazer com o *Scratch*? Como conseguiram isso?

As respostas a esta questão estão indicadas na Tabela *10*.

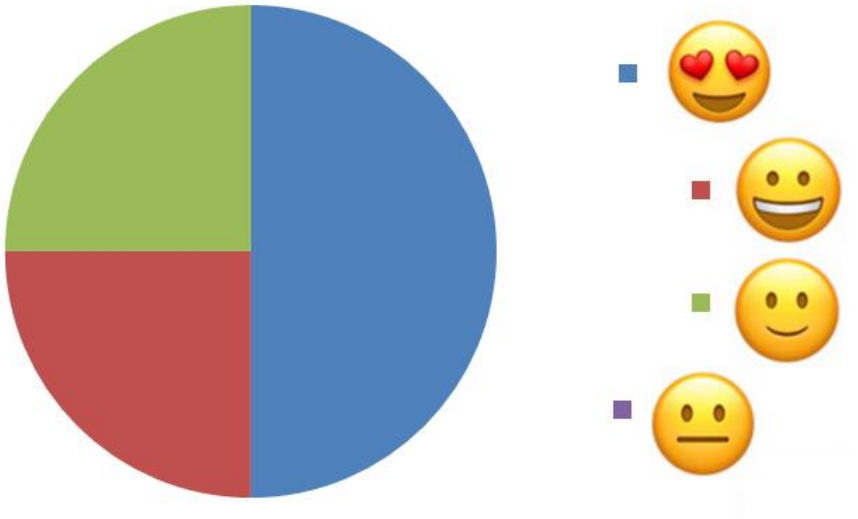
Tabela 10 - Respostas da questão 2 do relatório do encontro 1.

Grupo	**Resposta**
1	Gato andar e falar.
2	Ele falar e andar.
3	Muita coisa com a programação
4	Nós gravamos, programamos, editamos e rimos muito. Com as instruções do professor.

3 – Arraste para o quadrado o *emoji* que representa o que você achou da atividade de hoje.

Todas as respostas foram positivas, conforme apresentado na Figura *44*, mostrando entusiasmo em fazer atividades deste tipo.

Figura 44 - Respostas do item 3 do relatório do encontro 1.



A seguir, encontra-se a avaliação deste encontro. Levando em consideração que foi um encontro introdutório, onde grande parte da atividade foi realizada com o professor, não é possível afirmar o desenvolvimento de todas as práticas e perspectivas por parte de todos estudantes, mas destacam-se alguns pontos importantes.

Foi realizada a análise a partir do programa desenvolvido pela turma (*Figura 41*). É importante destacar que todos os estudantes conseguiram construir este programa sem ajuda, quando o projetor foi desligado. Os conceitos presentes nos programas foram sequências e laços. As sequências constituem uma série de passos no programa. Brennan e Resnick (2012, p. 3, tradução nossa) escrevem que "como uma receita, uma sequência de programação especifica o comportamento ou ação que deve ser produzido". No caso deste programa simples, a ação produzida foi a de mover o gato e fazê-lo falar. Quanto ao uso do laço, observou-se que o bloco "repita 5" foi utilizado de maneira correta. Quanto a este conceito, Brennan e Resnick (2012, p. 4, tradução nossa) destacam que "laços são mecanismos para executar a mesma sequência múltiplas vezes".

Com relação às práticas desenvolvidas, apesar de ser o primeiro encontro, foi possível perceber a prática de "teste e depuração de erros" no momento em que a turma precisou encontrar uma solução para o problema do gato do *Scratch* não se movimentar suavemente (descrito no item 3.5.1.2. Parte 1: Movimentando o Objeto). Para Brennan e Resnick (2012, p. 5, tradução nossa) "as coisas raramente funcionam como o planejado e é crítico desenvolver estratégias para lidar com e antecipar problemas".

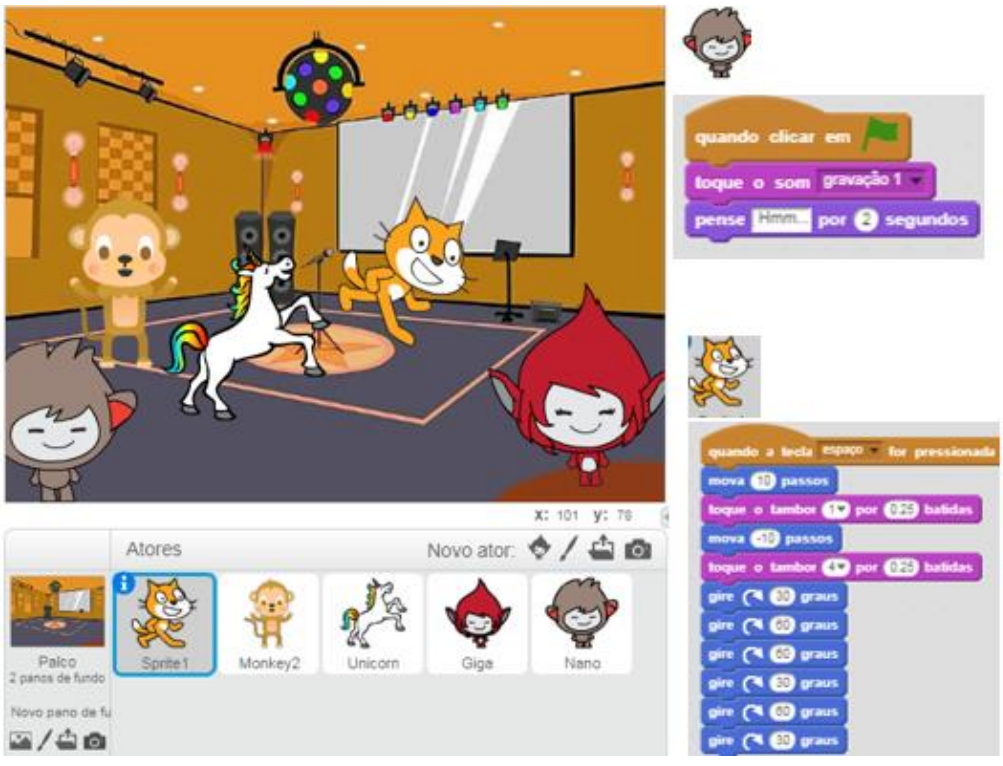
Com relação à prática denominada abstrair e modularizar, observou-se na solução apresentada que a turma aplicou a abstração. A turma identificou duas tarefas que o gato do *Scratch* deveria executar: deslocar-se e falar. Sendo duas tarefas independentes, elas foram divididas, programadas e testadas separadamente. Por fim, os programas foram integrados em um, representando a solução completa. O conceito de modularização não foi empregado nessa atividade.

4.2. Encontro 2

A dinâmica de dança aconteceu de maneira tranquila e pareceu despertar o interesse dos estudantes pela atividade. Na segunda parte, foram instruídos a construir uma animação de dança seguindo o tutorial do *Scratch*[22]. Concluído o tutorial, foram desafiados a modificar a animação da maneira que achassem mais interessante.

A animação do grupo 01 (Figura *45*) consiste em várias personagens estáticas. Apenas duas personagens estão programadas: Nano, que toca o som que o grupo gravou para a dança, e o gato do Scratch, que se move com som de tambor e depois gira.

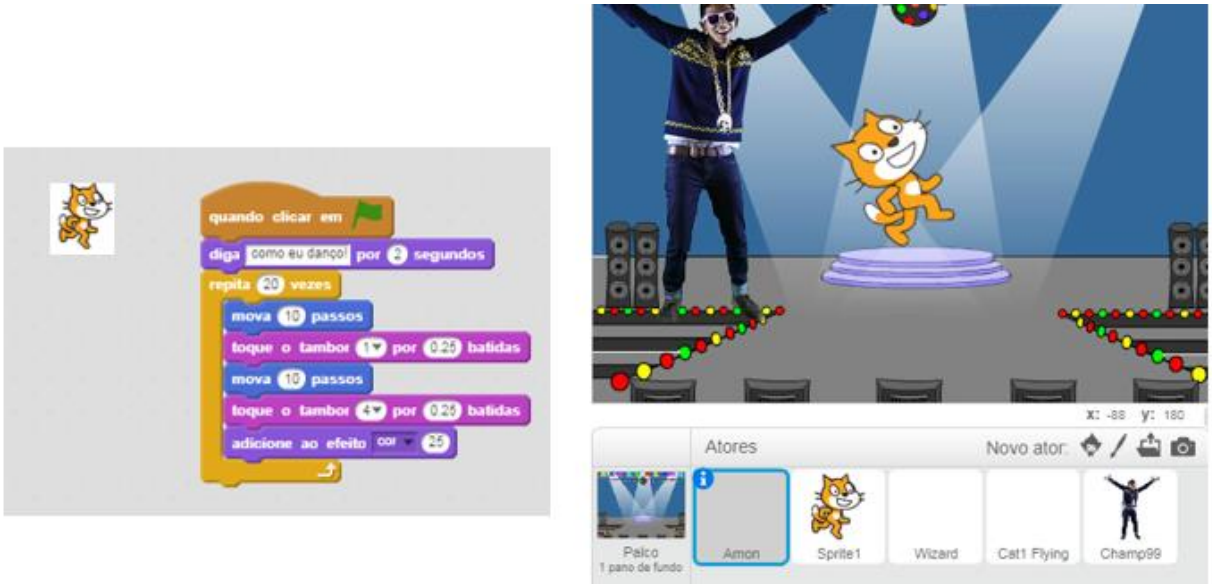
Figura 45- Projeto do grupo 01.



[22] https://scratch.mit.edu/projects/97071142/

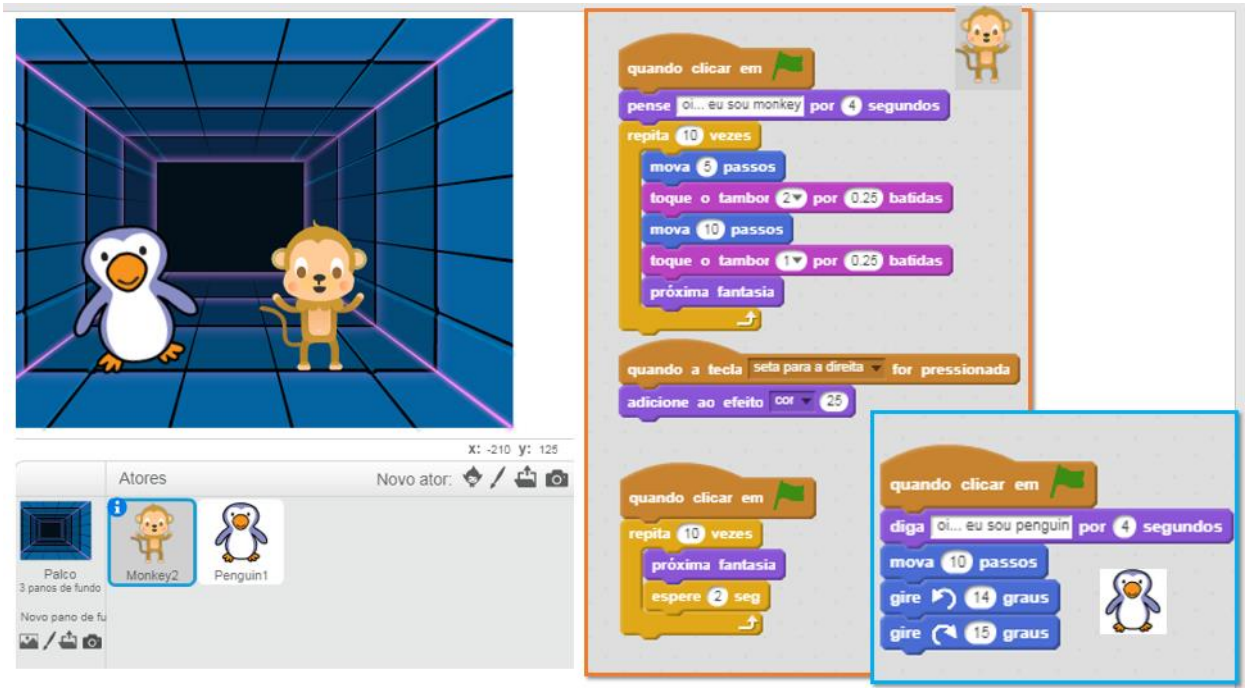
O projeto do grupo 02 (Figura *46*) mostra que também exploraram o uso de várias personagens. No entanto, na versão final, apenas o gato do *Scratch* estava programado. Aqui observamos a presença de um *loop* (ciclo).

Figura 46 - Projeto do grupo 02.



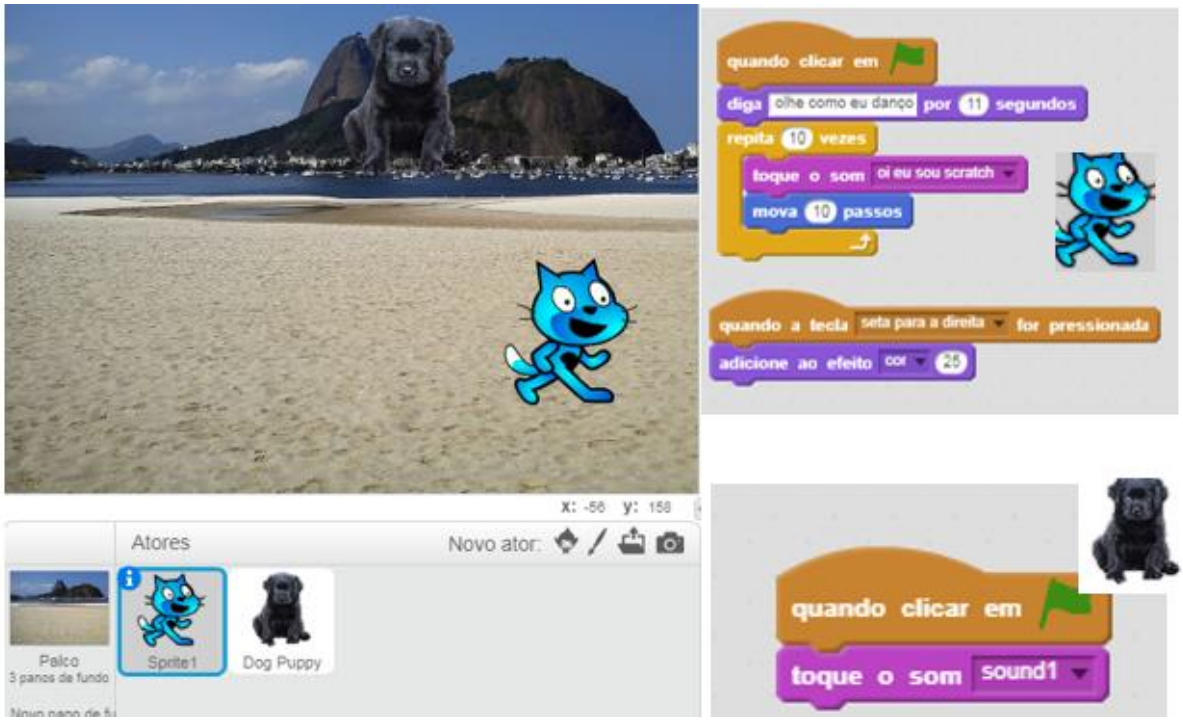
O grupo 03 conseguiu, em seu projeto (Figura *47*), utilizar alguns conceitos do Pensamento Computacional. Observa-se especialmente o paralelismo de 3 *scripts* e a presença de *loops*.

Figura 47 - Projeto do grupo 03.



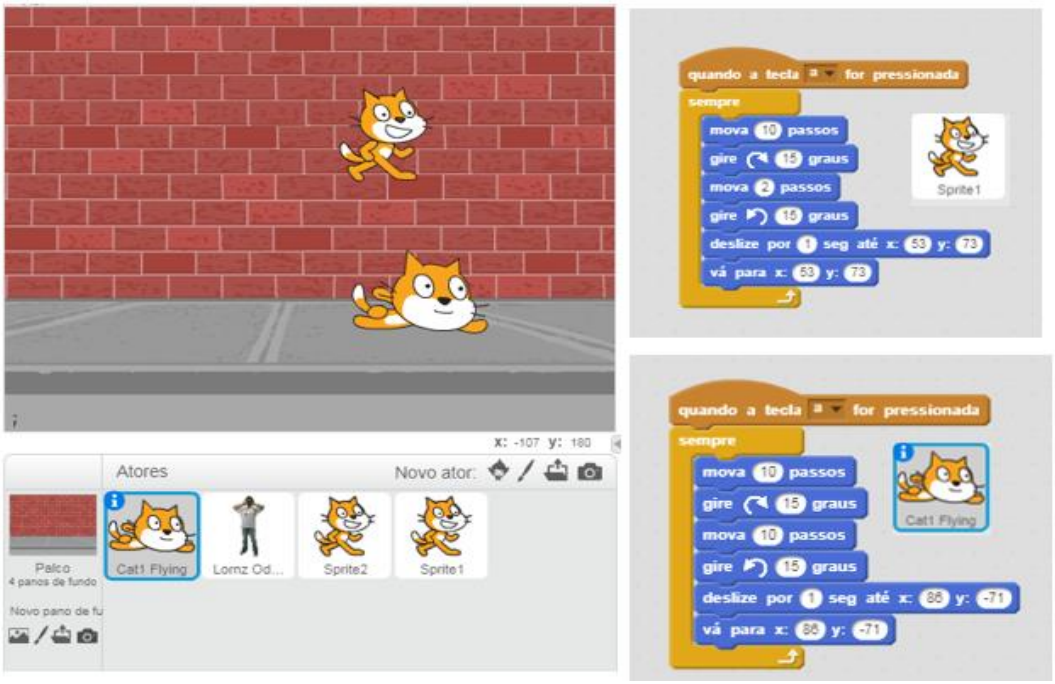
O grupo 04 utilizou duas personagens em seu projeto (Figura *48*). O gato do *Scratch* teve um comando especial para trocar de cor (seta para direita).

Figura 48 - Projeto do grupo 04.



O grupo 05 fez uso de comandos muito parecidos, alterando apenas os valores de posição, para animar seu projeto (Figura *49*).

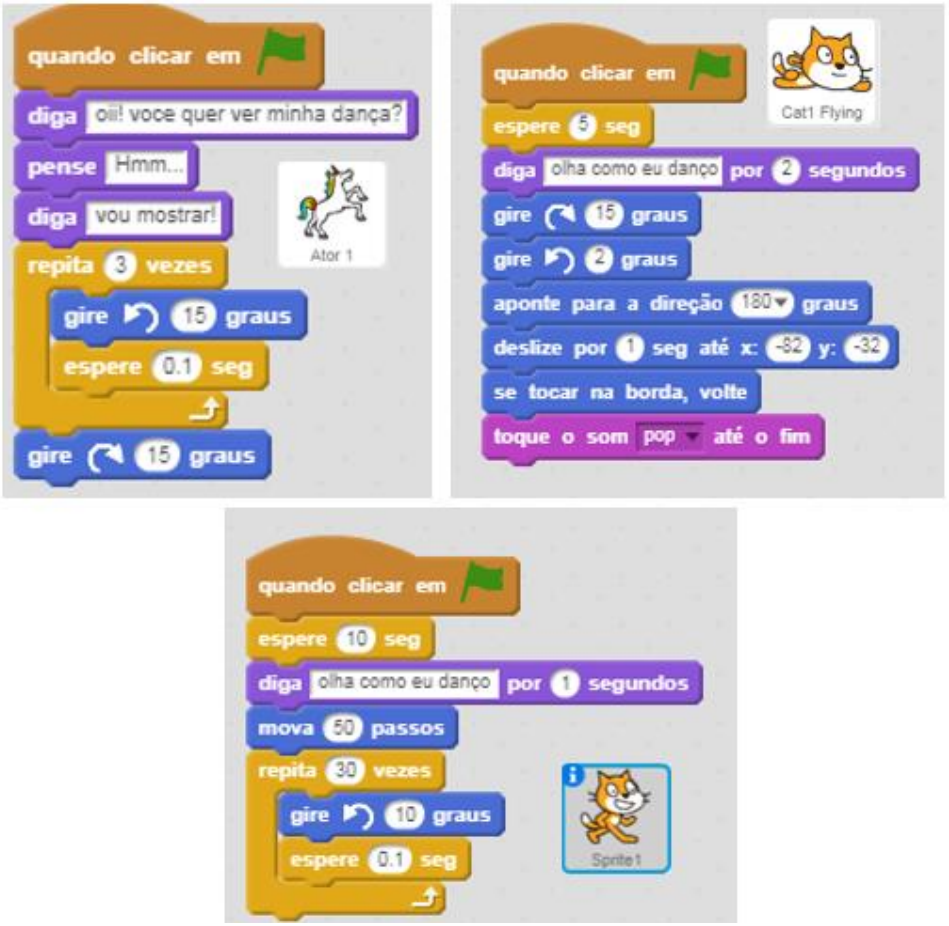
Figura 49 - Projeto do grupo 05.



O projeto do grupo 06 (Figura *50* e Figura *51*) contou com três *scripts* paralelos.

Figura 50 - Projeto do grupo 06 (parte 1).

Figura 51 - Projeto do grupo 06 (parte 2).



O grupo 07 (Figura *52* e Figura *53*) conseguiu fazer um programa com a função “enviar mensagem” para melhor sincronizar suas personagens. Aqui também encontramos o paralelismo de programas.

Figura 52 - Projeto do grupo 07 (parte 1).

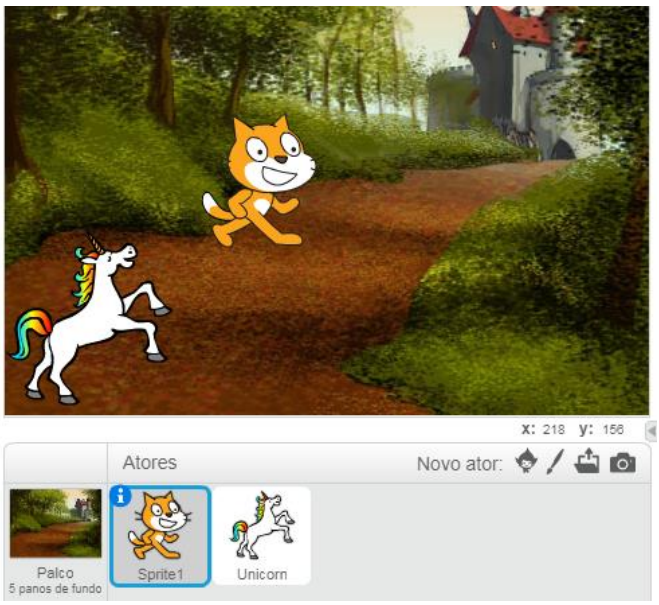
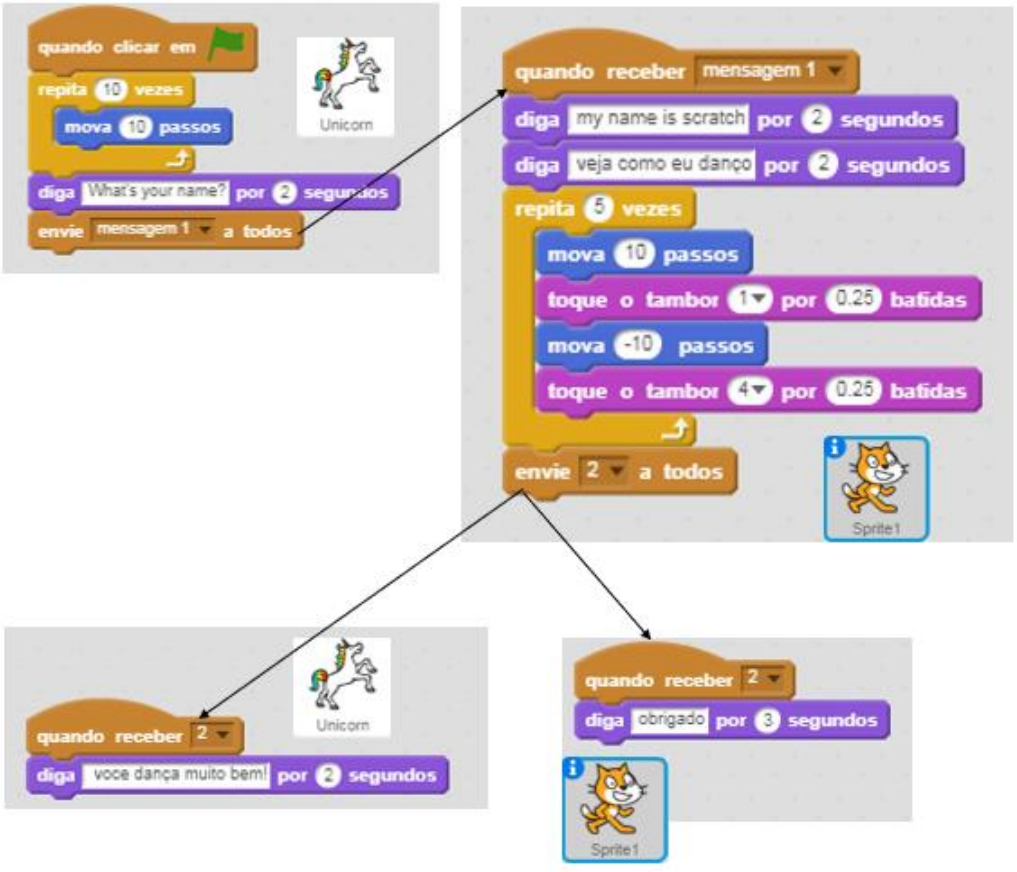


Figura 53 - Projeto do grupo 07 (parte 2).



O grupo 08 conseguiu utilizar mais de uma personagem (Figura *54* e Figura *55*), mas ainda com algumas dificuldades, como os blocos de programação soltos: não faziam parte da sequência e ficaram sobrando. Quando questionados, disseram que não sabiam como deletar os blocos.

Figura 54 - Projeto do grupo 08 (parte 1).

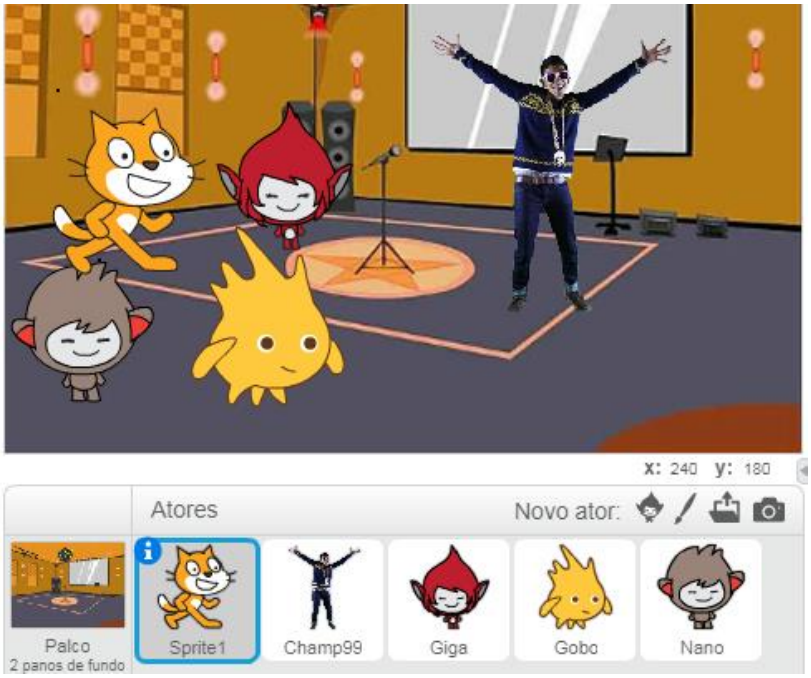
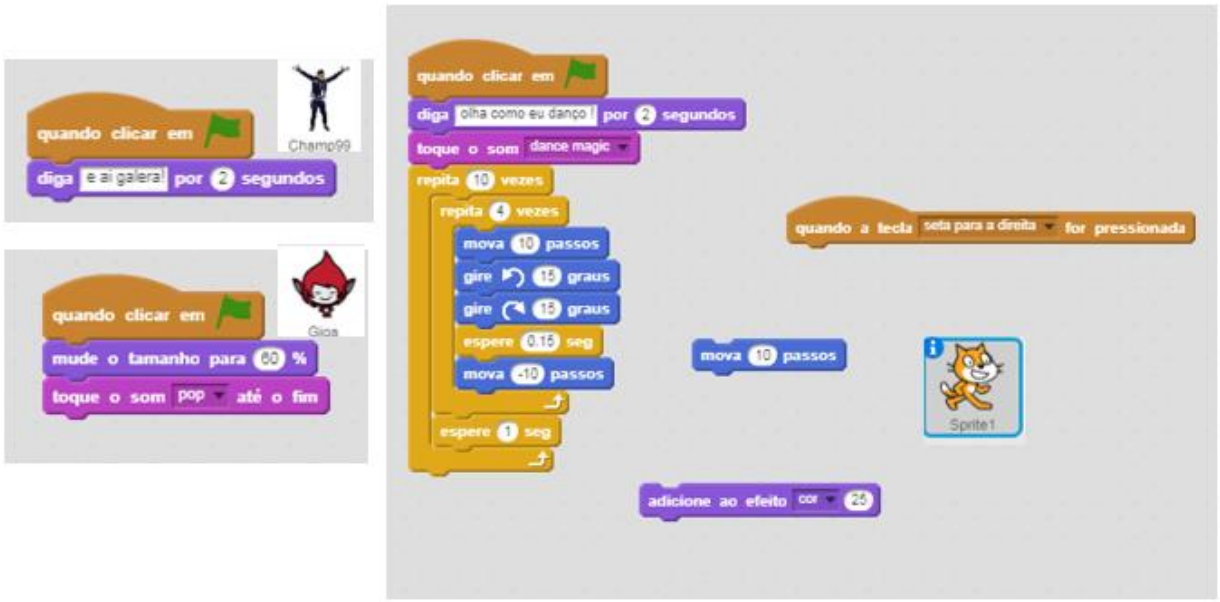


Figura 55 - Projeto do grupo 08 (parte 2).



Finalizando seus projetos, cada grupo apresentou para a turma sua animação.

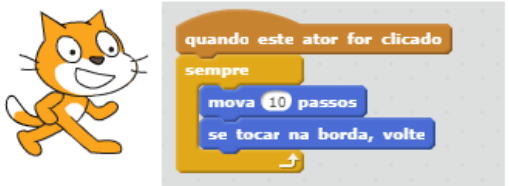
A fim de atestar os conceitos computacionais presentes, procedeu-se com a identificação, baseada em Brennan e Resnick (2012) e constatou-se que os programas de todos os grupos expressaram a festa como uma sequência de passos de maneira bem sucedida. Identificou-se também o uso de laços de repetição (instrução repita) em quase todos os projetos. Os grupos demonstraram facilidade em compreender o uso de eventos para casos simples (clique na bandeira verde). Contudo, eles não souberam utilizar os mecanismos de eventos para organizar a festa. Percebeu-se nos programas que foram utilizados diferentes blocos de espera para sincronização das personagens da festa. Somente o grupo 07 buscou criar eventos partindo do uso de mensagens sincronizadas. O uso de eventos e dos recursos de paralelismo são inerentes a essa atividade, devendo estar presentes para provocar o efeito esperado de uma festa. Nos programas também estava presente o conceito de paralelismo, facilmente observado quando uma personagem interagia com outra. As sequências aconteciam ao mesmo tempo, e foram sincronizadas de maneira correta.

Quanto às práticas desenvolvidas, durante a realização da atividade, muitos testes tiveram de ser feitos. A tendência inicial dos estudantes foi de pedir ajuda e pensar que seu trabalho estava errado quando não funcionava de acordo com o esperado e, então foram estimulados a testar seus trabalhos e tentar depurar os possíveis erros. Foi perceptível que, quando se deparavam com uma falha no programa, muitos se frustravam, o que deu abertura para trabalhar a cultura de sala de aula proposta com o projeto-parceria da CSTA e ISTE: “aceitação de tentativas frustradas de solução, reconhecendo que o erro pode ajudar a encontrar o caminho do sucesso” (ISTE, 2011). Para Brennan e Resnick (2012, p. 7, tradução nossa), “As coisas raramente funcionam como imaginado e é crítico desenvolver estratégias para lidar com – e antecipar – problemas” e isso se caracteriza com a prática de testar e depurar erros. Confirma-se então nessa etapa a ocorrência da prática de testar e depurar erros. Além disso, os estudantes dividiram o problema (programar a animação de uma festa) em partes, atribuindo um comportamento para cada personagem. A abstração nessa atividade esteve presente à medida em que os grupos programaram as personagens da festa, uma a uma (visão local), e somente depois de toda programação concluída puderam observar a festa (visão global). Bernnan e Resnick (2012, p. 9, tradução nossa) caracterizam essa habilidade como “construir algo grande juntando coleções de partes menores”.

4.3. Encontro 3

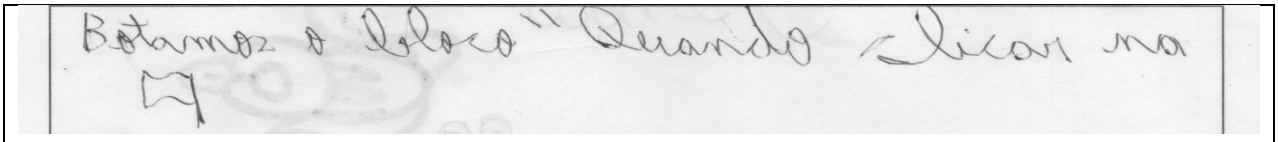
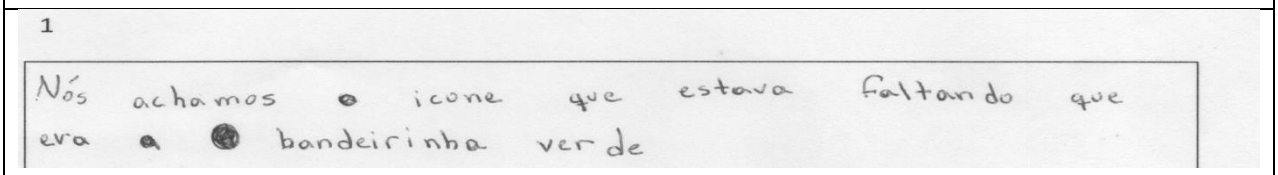
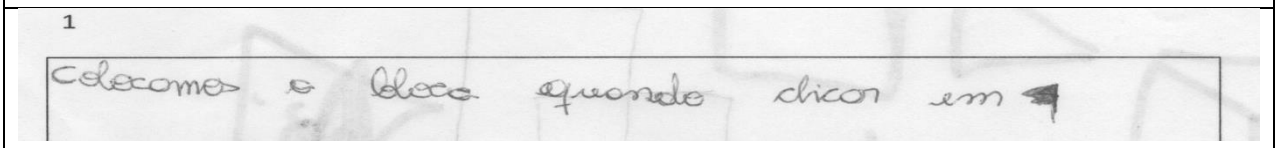
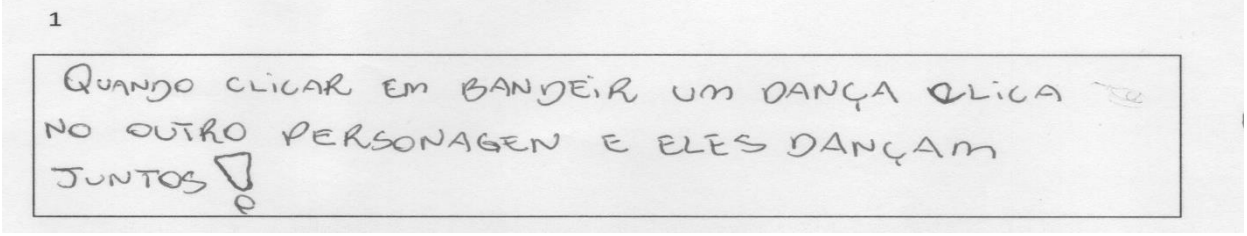
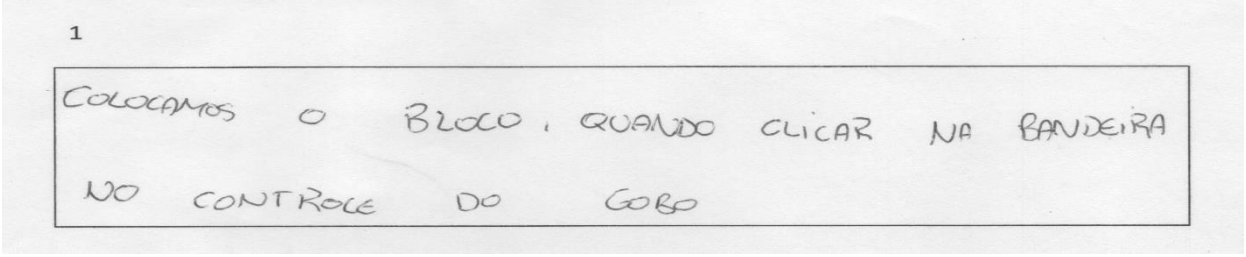
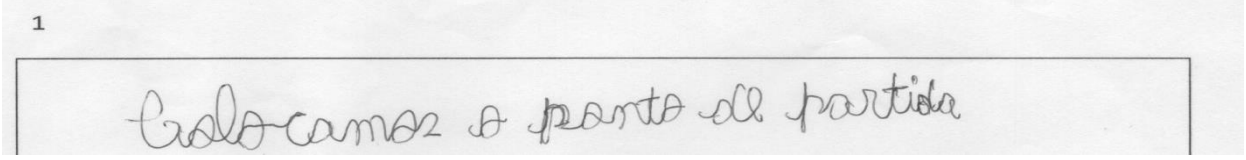
Neste encontro, os estudantes foram desafiados a depurar códigos e registrar suas soluções, conforme a Tabela *11*. O registro encontra-se no Apêndice A.

Tabela 11 - Lista 1 de desafios para depurar.

1	Neste programa, tanto o gato do *Scratch* quanto o Gobo (personagem amarelo) deveriam dançar quando a bandeira verde é clicada, mas isso não acontece. 
2	O gato do *Scratch* deveria dar uma cambalhota ao pressionar espaço, mas ele fica parado. 
3	O gato do *Scratch* deveria andar de um lado para o outro da tela, mas algo deu errado e ele está voltando de cabeça para baixo! 
4	A intenção do programa era que o gato do *Scratch* miasse de duas formas: com o balão de fala e através do som, que repetiria o miado 3 vezes. Algo no programa está fazendo com que ele reproduza o som somente depois do balão, e não ao mesmo tempo, e o som está sendo reproduzido somente uma vez. 

O desafio 1 foi solucionado por todos. Na Tabela *12* transcrevemos as soluções apresentadas. Observamos que a solução unânime foi a de colocar o bloco "quando clicar em bandeira verde" na programação do Gobo.

Tabela 12 - Soluções para o desafio 1.


"Botamos o bloco "Quando clicar na bandeira""

"Nós achamos o ícone que estava faltando que era a bandeirinha verde"

"Colocamos o bloco quando clicar em bandeira"

"Quando clicar em bandeira um dança clica no outro personagem e eles dançam juntos!"

"Colocamos o bloco quando clicar na bandeira no controle do Gobo"

"Colocamos o ponto de partida"

Abaixo fazemos uma análise das soluções propostas para o desafio 2 (Tabela *13*).

Tabela 13 - Soluções para o desafio 2.

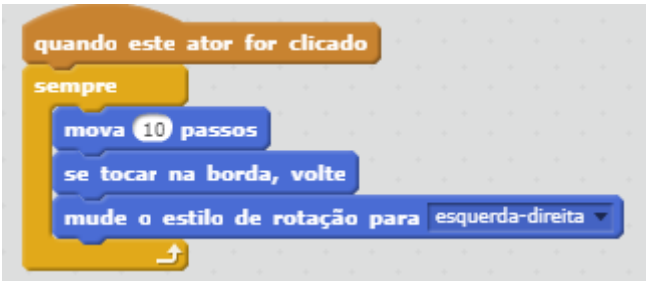
1		**2**	
3		**4**	

A solução 1 mostra que não existe entendimento do conceito de ângulo. Os grupos não colocaram o ciclo (*loop*), e para que a solução funcione é necessário segurar a barra de espaço.

As soluções 2, 3 e 4 realizam o que o programa estava proposto a fazer. No caso das soluções 2 e 4, existe a repetição do bloco “gire 10 graus” que é desnecessária e poderia ter sido sintetizada em um único bloco.

O desafio 3 foi solucionado por apenas uma equipe, conforme apresentado na Figura *56*. As demais não apresentaram solução. Foi, então, realizada uma discussão sobre a solução do grupo com a turma e como ela poderia ter sido feita. Os estudantes que não conseguiram realizar o desafio, então, exploraram a solução, a fim de aprender novos recursos.

Figura 56 - Solução do desafio 3.



O desafio 4 foi solucionado por todas equipes de maneira semelhante (Figura *57*). Uma das equipes gravou seu próprio som de miado, com a função de gravar áudios do *Scratch*.

Figura 57 - Solução do desafio 4.

Todos os programas tratavam de arrumar sequências, fazendo com que o entendimento do conceito estivesse presente em todos os desafios. A presença de laços também aconteceu em todos os desafios propostos, especialmente no desafio 2, em que três grupos utilizaram o laço para consertar o programa. Eventos simples, que dão início aos programas estavam presentes em todos os desafios. Em especial, no desafio 1 era necessário identificar que não havia um evento que era condição de início para o movimento de uma das personagens. Constata-se dessa forma a presença do conceito de eventos, que consiste em um comando que dá inicio a outra ação (BRENNAN, RESNICK, 2012). O conceito de paralelismo também esteve presente no desafio 1, onde foi necessário compreender que os programas deveriam ser executados ao mesmo tempo.

Em termos de práticas desenvolvidas, a prática de testar e depurar erros era inerente ao desafio, já que a proposta era testar o programa e corrigir o erro. Reutilizar e remixar também foi parte da atividade, que consistia em adaptar programas que não foram criados pelos próprios estudantes e caracteriza outra habilidade do Pensamento Computacional (BRENNAN, RESNICK, 2012). Observou-se também que ainda era muito presente a ideia de que existe apenas uma solução certa, e foi difícil para alguns estudantes entenderem que podem existir diversas soluções certas para cada cenário apresentado. Nesse caso, houve a oportunidade de

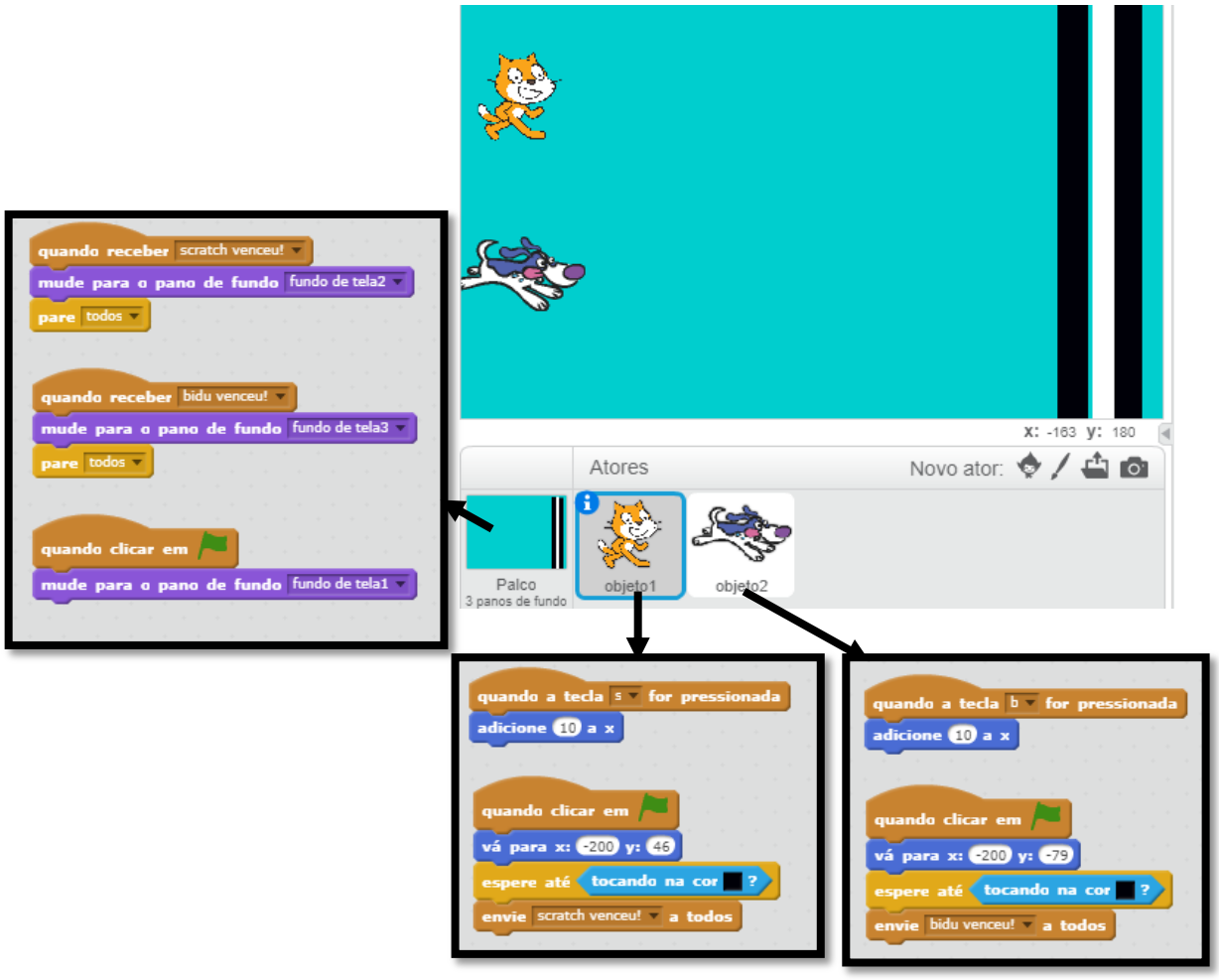
trabalhar o que Papert e Turkel (1990) chamam de *Epistemological Pluralism*[23], em que as formas de conhecimento e de conhecer podem ser variadas. A maneira como cada grupo encontrou suas soluções dependeu de como cada grupo interpreta e se relaciona com a proposta.

.

4.4. Encontro 4

A Figura *58* ilustra um dos jogos de corrida criados pelos estudantes. Os jogos dos outros grupos ficaram muito semelhantes ao apresentado, utilizando como ilustração para os resultados do encontro. Um dos grupos também conseguiu adicionar mais um competidor ao jogo, utilizando outra tecla para realizar seu movimento.

Figura 58 - Jogo de Corrida 1.



[23] Epistemological Pluralism pode ser traduzido como Pluralidade Epistemológica. Não foi encontrado na literatura em língua portuguesa nenhuma adaptação para este termo.

O jogo (Figura *58*) funcionou corretamente e utilizou o sistema de mensagens para comunicar eventos. Os estudantes fizeram uso do paralelismo, com códigos simultâneos. Também conseguiram apresentar o conceito de condicionais pela primeira vez: quando uma personagem toca na cor preta, o plano de fundo muda para uma mensagem que anuncia o vencedor. Estabeleceram a condição de vitória e, a partir disso, traduziram para o jogo.

O uso das mensagens para realizar eventos deixa claro que os estudantes conseguem comunicar, a esta altura, quando algo de seu programa deve iniciar e traduzir isso para a linguagem de programação.

Sobre as práticas desenvolvidas, destacam-se o teste e depuração de erros, que fez parte de todo o encontro até o resultado final, e reutilizar e remixar, especialmente no caso que os estudantes puderam reutilizar parte de seu código para criar o código do outro competidor, mudando apenas valores de partida e a tecla que indicará seu movimento.

4.5. Encontro 5

Como forma de avaliação, foi verificado que todos estudantes conseguiram concluir a atividade proposta, que explora diversos conceitos da programação e pensamento computacional.

Nas atividades da “Hora do Código”, estavam presentes os conceitos computacionais de Brennan e Resnick (2012). Quanto ao conceito de sequências, estava presente quando os estudantes faziam uma série de passos para que a personagem do Minecraft se movimente, tose as ovelhas, corte as árvores e realize outras ações. O conceito de laços também fez parte da atividade, quando as personagens repetiam ações como “andar para frente e colocar um bloco de madeira” para construir uma casa. Os condicionais também estavam muito presentes, em momentos onde as personagens deveriam verificar se havia lava na sua frente e, nesse caso, colocar um bloco de pedra para que não pisassem nela.

A principal prática (BRENNAN, RESNICK, 2012) presente na atividade foi a de reutilizar e remixar. Cada desafio era mais complexo que o anterior e muitos precisavam que se reutilizasse o código que criou anteriormente. Também foi bastante presente a prática de testar e depurar erros, e o próprio ambiente verificava a necessidade de refazer o código.

4.6. Encontro 6

Similar ao encontro 3, os estudantes foram desafiados a depurar códigos (Tabela *14*, Tabela *15*, Tabela *16* e Tabela *17*) e registrar suas soluções (Apêndice B).

Tabela 14 - Lista de desafios para depurar (1 e 2).

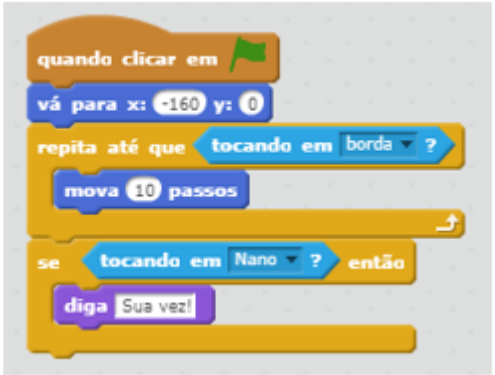
	Desafio
1	Na programação, o gato do *Scratch* convida o usuário do programa para ver sua dança, mas ao clicar nele, ele apenas se move uma vez, e o som de tambores continua com ele parado. Como podemos consertar isso?  
2	Na programação, Pico e Nano estão brincando de pega-pega. Quando Pico encosta em Nano, deveria dizer “Sua vez!”, mas isso não acontece. O que deu errado?  PICO   NANO 

Tabela 15 - Desafio 3 para depurar.

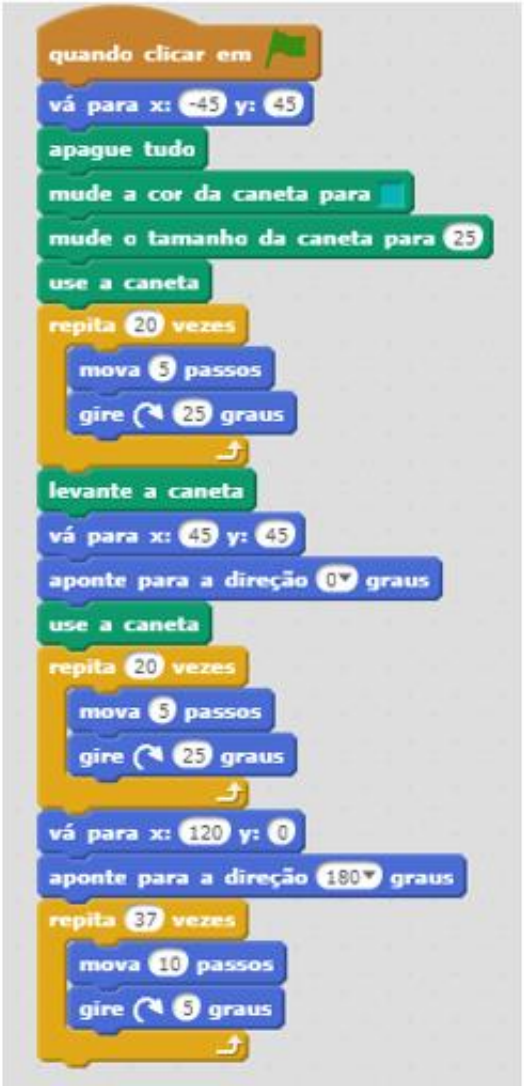
3	Este programa deveria desenhar um rosto feliz, mas ele está conectando um dos olhos à boca. O que podemos fazer?  

Tabela 16 - Desafio 4 para depurar.

4	Este programa deveria parar quando a flor está crescida, mas ele continua repetindo a animação sem fim. Como podemos solucionar este problema?

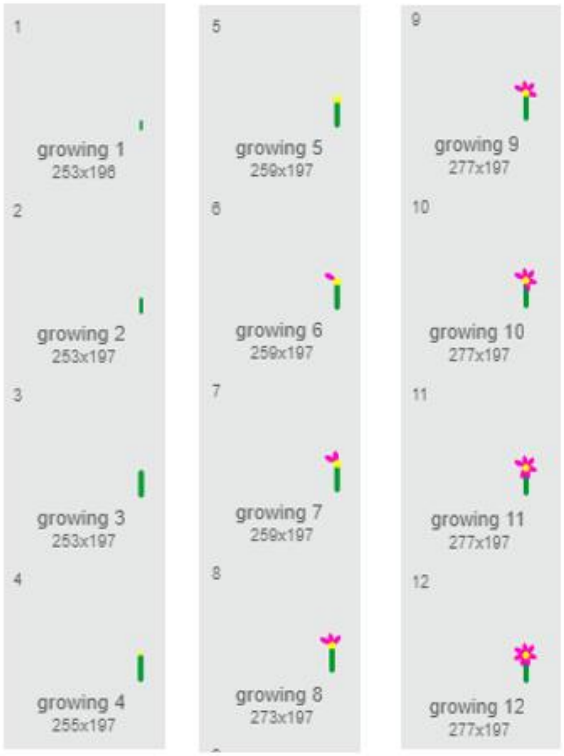


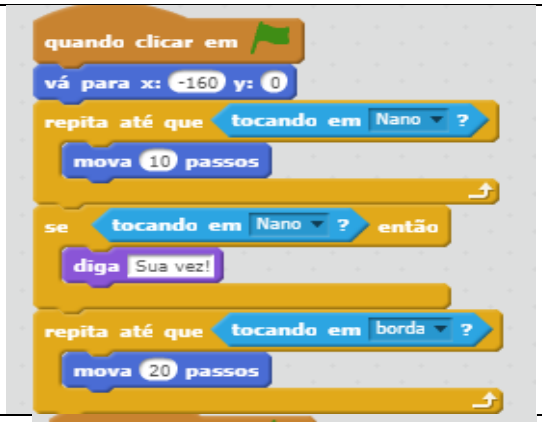
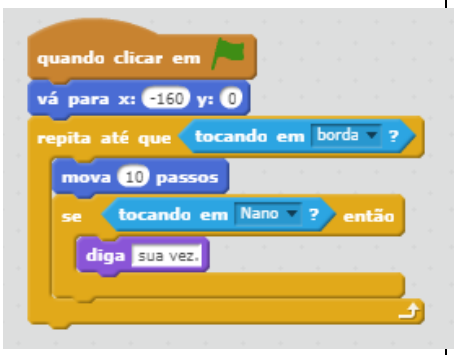
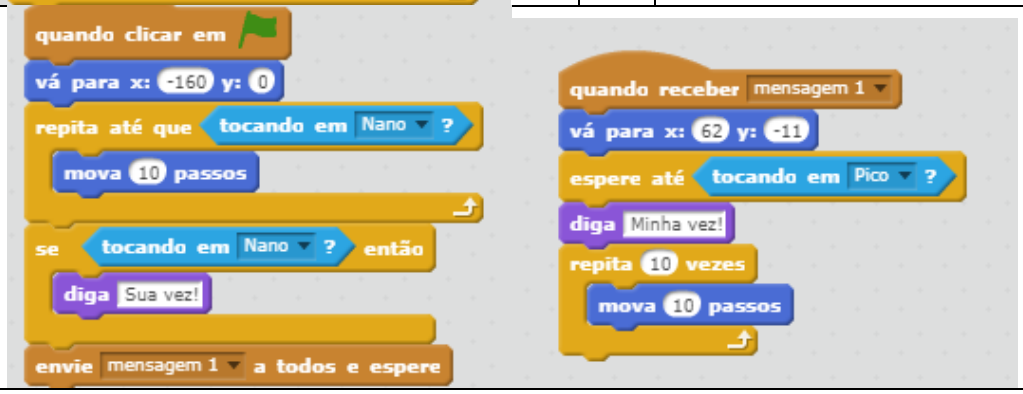
Tabela 17- Desafio 5 para depurar.

5	Esta animação deveria tocar o tema “parabéns pra você” e, ao acabar a música, deveria aparecer a instrução de “Clique para apagar as velas!”, mas essa instrução aparece durante a música. Como podemos consertar este erro?   

No desafio 1, todos os grupos conseguiram depurar e corrigir o código, colocando o bloco de trocar fantasia dentro do ciclo. Uma das equipes também personalizou sua dança, fazendo o gato do *Scratch* girar ao invés de trocar de fantasia.

O desafio 2 mostrou uma pluralidade de soluções, conforme apresentadas na Tabela *18*.

Tabela 18 - Soluções do desafio 2.

1		**2**	
3			

O desafio 3 foi resolvido por todas as equipes adicionando os blocos de “levantar a caneta” e “abaixar a caneta” no local adequado. Alguns grupos conseguiram fazer a caneta trocar de cor.

O desafio 4 gerou duas resoluções. A mais comum foi trocar o número 12 por 11, fazendo com que a flor parasse na fantasia 12, já que é maior do que 11. A outra solução foi de trocar o sinal de “maior que” por “igual a”.

O desafio 5 foi solucionado adicionando um bloco de espera entre a música e o pedido de clicar no bolo. Uma das equipes não conseguiu solucionar o desafio.

O conceito de sequência estava presente em todos os programas, que tratavam de uma série de passos a serem seguidos (BRENNAN. RESNICK, 2012). Coube aos estudantes encontrarem em qual destes passos havia um erro. Especialmente no desafio 3 foi necessário que os estudantes entendam como a sequência de passos é organizada para que interpretem o momento certo de adicionar o bloco “abaixar a caneta”.

Os condicionais também foram um conceito desenvolvido e bastante presente na atividade. O conceito consiste em traduzir a habilidade de tomar decisões baseado em certas condições (BRENNAN, RESNICK, 2012). Nos desafios 2 e 4 os estudantes precisaram encontrar e arrumar a condição que não estava sendo atingida para que o programa executasse o que era esperado.

Os conceitos de laço, eventos e paralelismo também estavam presentes nas soluções dos desafios.

As práticas desenvolvidas foram a de testar e depurar erros, por se tratar de um desafio de corrigir erros nos programas, percebeu-se que a primeira coisa que os estudantes fizeram foi testar o programa para ver o que deu errado. Também praticaram o ato de reutilizar e remixar, utilizando os códigos errados para criar soluções adequadas.

4.7. Encontro 7

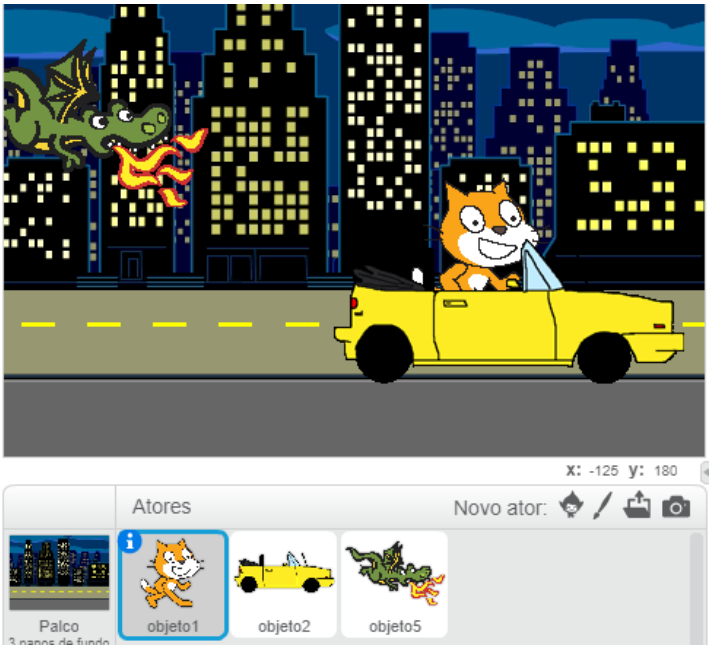
Os estudantes realizaram a atividade de desenhar um monstro em conjunto, sem saber como a parte anterior do monstro era. O resultado desta atividade está no Apêndice E.

Após esta atividade, os estudantes foram desafiados a criar uma animação. Cada grupo teve um tempo de 10 minutos para criar sua animação. No final deste tempo, fizeram o rodízio entre os computadores para continuar a animação do grupo vizinho. O processo foi repetido até que o grupo voltasse ao seu computador original.

4.7.1. Animação 1

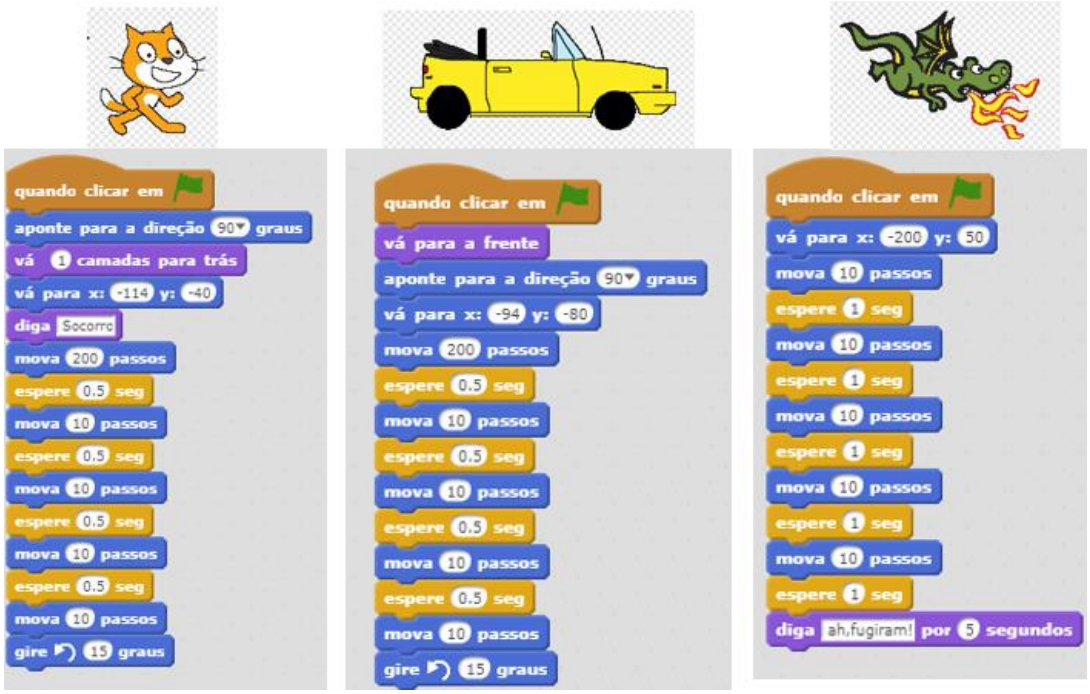
A animação da Figura *59* mostra o gato do *Scratch* dirigindo um carro e fugindo de um dragão.

Figura 59 - Animação 1.



A turma utilizou 3 scripts paralelos, utilizando a espera em segundos para sincronizar a animação (Figura *60*).

Figura 60 - Programação da Animação 1.



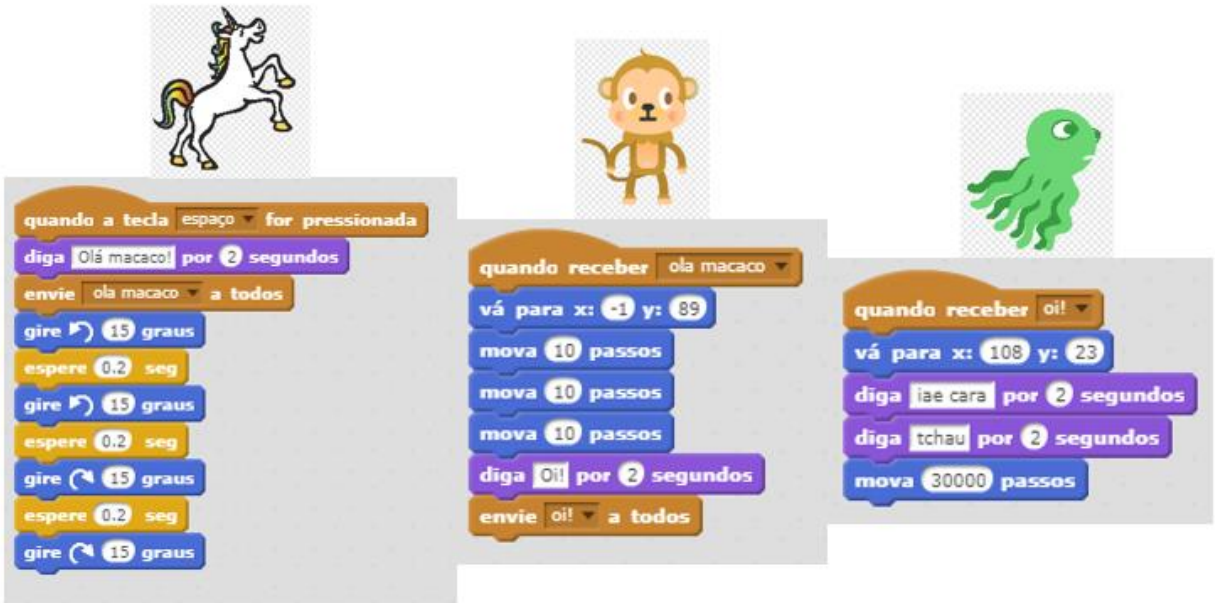
4.7.2. Animação 2

A animação mostra um breve diálogo entre um macaco, um unicórnio e um polvo.

Figura 61 - Animação 2.

A turma utilizou 3 scripts paralelos, sincronizados com o sistema de enviar mensagens/eventos do *Scratch* (Figura *62*).

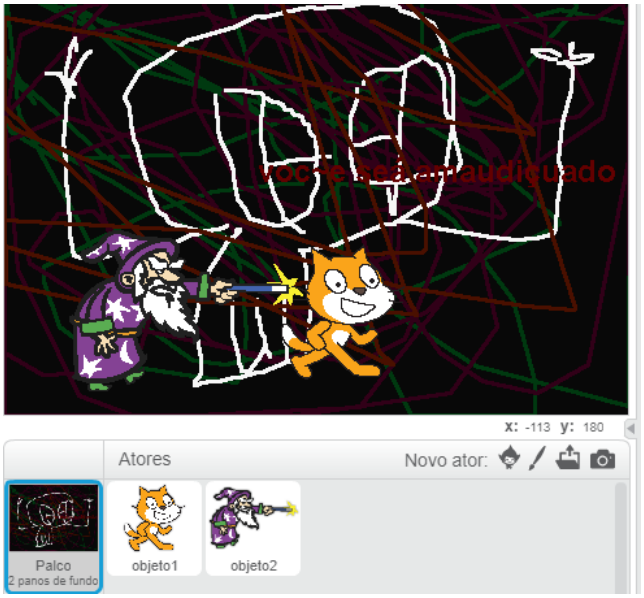
Figura 62 - Programação da Animação 2.



4.7.3. Animação 3

Nesta animação (Figura *63*), o bruxo lança uma maldição sobre o gato do *Scratch*, que tenta fugir, mas acaba sendo transformado em um sapo.

Figura 63 - Animação 3.



A turma colocou para que, ao clicar na bandeira verde, todas as personagens voltassem ao seu local original e para que o bruxo amaldiçoasse o gato. No palco, colocaram a execução de uma risada maléfica, gravada pela turma. Ao final da risada, o gato diz “Ah não!” e corre, mas é tarde demais, e ele muda a fantasia para um sapo.

Figura 64 - Programação da Animação 3.

Esta atividade foi uma oportunidade para ver o que os estudantes conseguiam criar e quais conceitos computacionais (BRENNAN, RESNICK, 2012) conseguiam mostrar domínio sem intervenção externa. Em todas animações encontramos sequências de passos e também o conceito de paralelismo, pois contam com mais de uma personagem animada, que interagem uma com a outra.

Os laços foram utilizados na animação 3 e os eventos na animação 2 e 3. Como a característica do encontro era criar uma animação livre, foi uma oportunidade de testar muita coisa que nos encontros anteriores os estudantes não tiveram oportunidade.

As práticas realizadas foram a de testar e depurar erros e principalmente a de reutilizar e remixar, pois fazia parte da atividade. Segundo Brennan e Resnick (2012, p. 8, tradução nossa), "Reutilizar e remixar apoia o desenvolvimento de capacidades críticas de interpretação de códigos". Ao trabalhar em cima do código construído por outros, os estudantes se encontraram na situação de ter de interpretar o código a fim de adicionar corretamente suas contribuições para melhorar a animação, sem desconstruir o que o grupo anterior havia deixado.

4.8. Encontro 8

O desafio de construir o labirinto foi lançado e a turma aceitou. Dividiram-se em duplas e trios e começaram a construir seu jogo conforme as orientações.

Nesta atividade, a intervenção do professor foi mínima e foi uma grande oportunidade para os estudantes mostrarem seu domínio dos conceitos computacionais e colocar em ação as práticas desenvolvidas.

4.8.1. Grupo 01

Utilizando conceitos de paralelismo e condicionais, fazem um jogo funcional de labirinto. Ao iniciar o jogo, o objeto gato do *Scratch* se posiciona sobre a bola azul. Utilizando os comandos "seta para cima", "seta para baixo", "seta para direita" e "seta para esquerda", o objeto se movimenta e, ao tocar na cor das paredes, manda o objeto de volta para o começo. Caso o objeto toque na cor magenta, é declarada a vitória.

Figura 65 - Jogo do Labirinto 1 (Código).

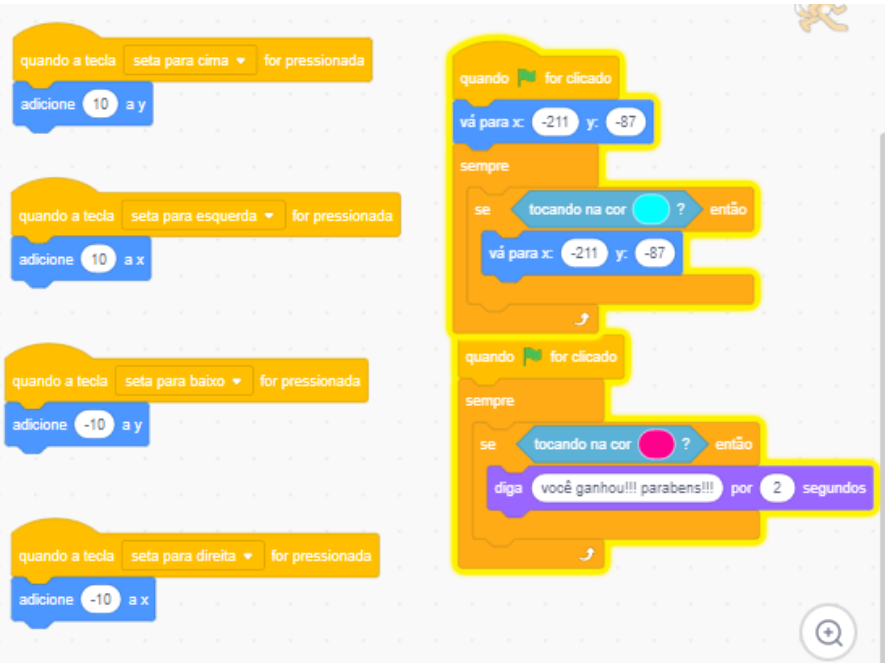
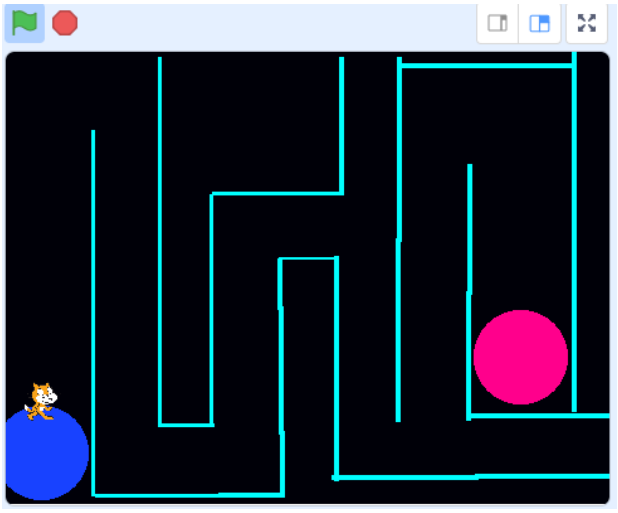


Figura 66 - Jogo do Labirinto 1 (Interface).



4.8.2. Grupo 02

Utilizando conceitos parecidos com o grupo 01, este programa também funciona conforme o solicitado. A novidade é que, ao conseguir a vitória, o "plano de fundo" troca para uma tela de vitória.

Figura 67 - Jogo do Labirinto 2 (Código).

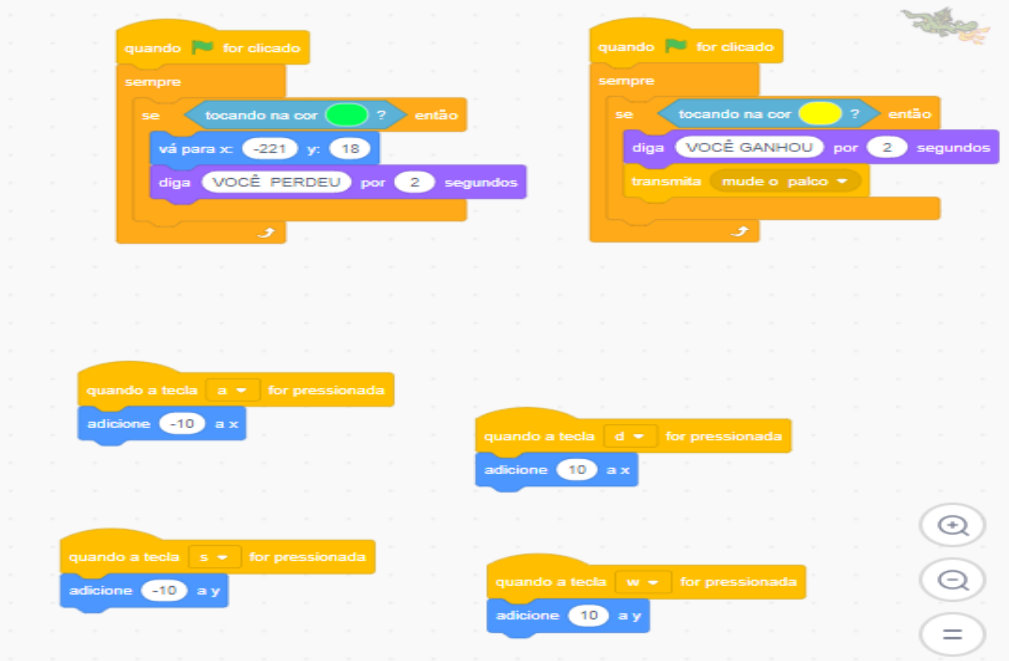
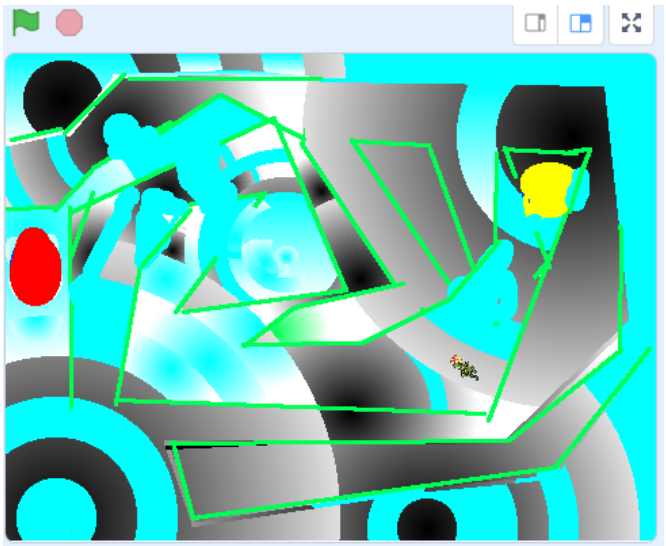


Figura 68 - Jogo do Labirinto 2 (Interface).



4.8.3. Grupo 03

O grupo 03 conseguiu realizar o desafio, que está apresentado nas Figuras Figura *69* e Figura *70*. Assim como o grupo anterior, o jogo funciona como proposto.

Figura 69 - Jogo do Labirinto 3 (Código).

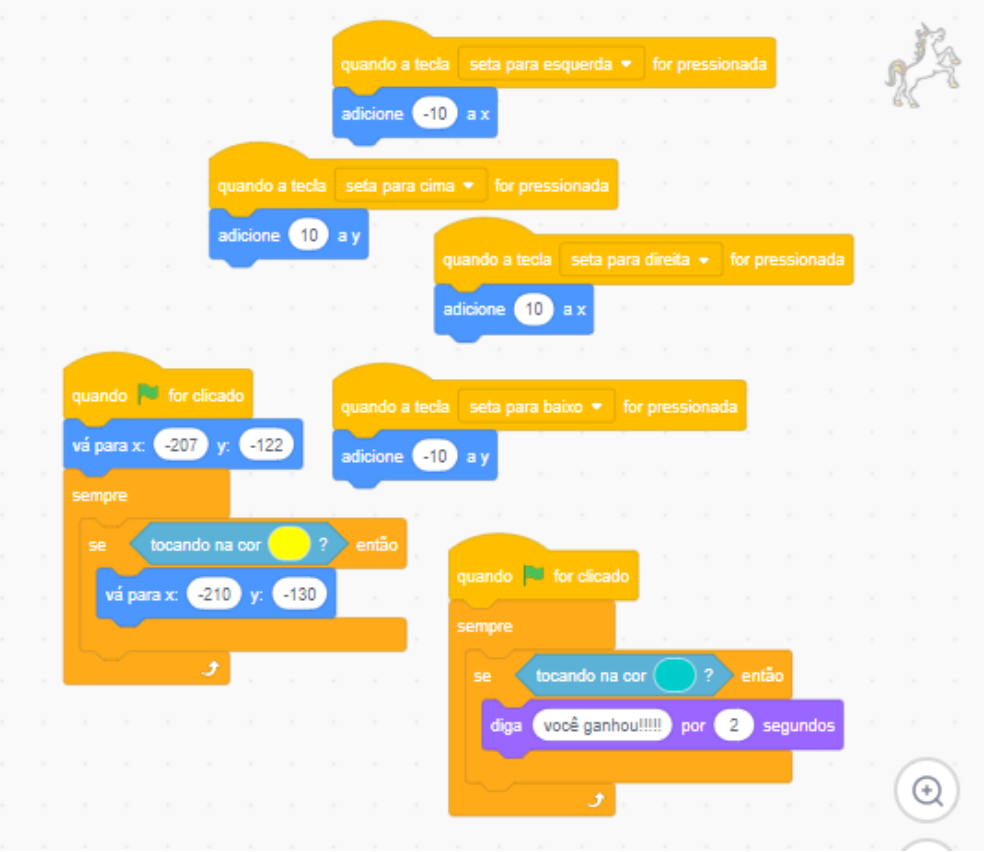
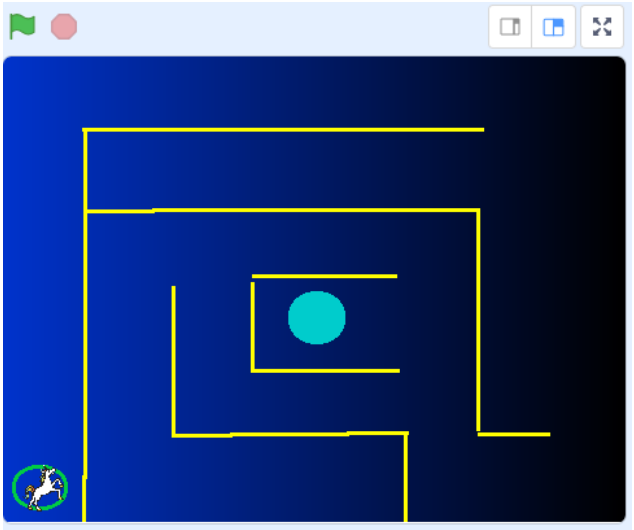


Figura 70 - Jogo do Labirinto 3 (Interface).



4.8.4. Grupo 04

De maneira análoga, o grupo 04 conseguiu realizar o desafio, apresentado nas Figuras 71 e 72. Como o grupo anterior, utilizaram as setas para movimentar o objeto e estabeleceram a condição de vitória por meio dos sensores.

Figura 71 - Jogo do Labirinto 4 (Código).

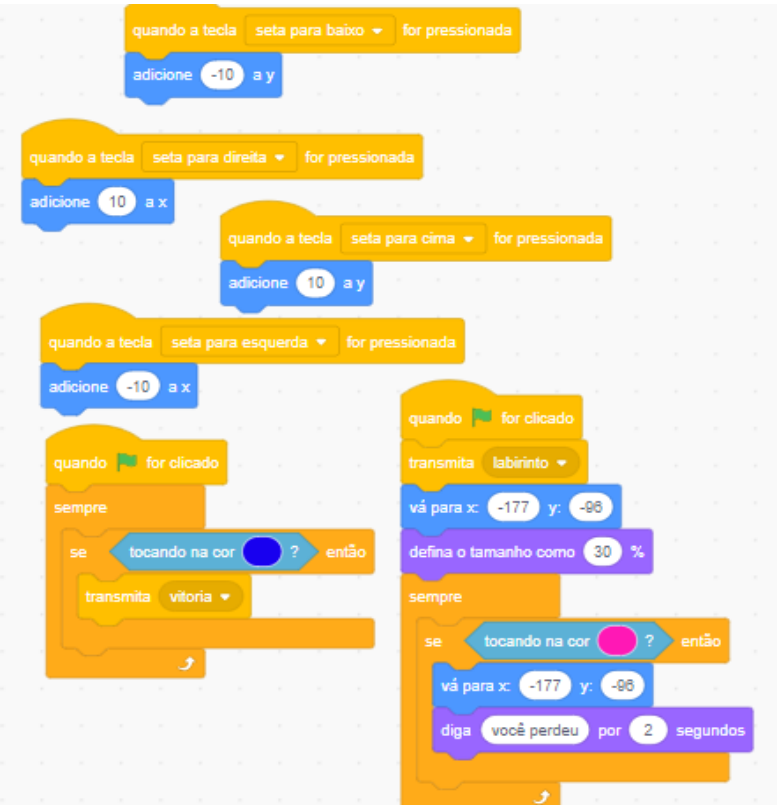
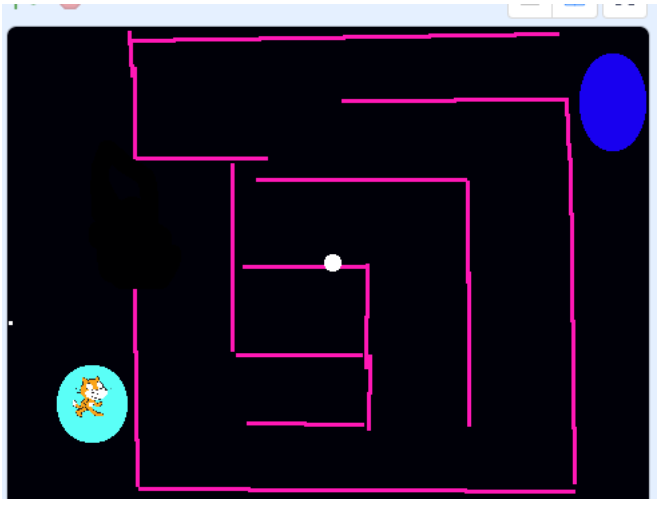


Figura 72 - Jogo do Labirinto 4 (Interface).



A construção deste jogo como encontro final foi satisfatória. Os programas mostram o desenvolvimento e o domínio dos conceitos e práticas que foram propostos.

Tiveram de utilizar os conceitos apresentados nos encontros anteriores e as práticas desenvolvidas. Foram utilizados os conceitos de sequência, laço, eventos, paralelismo e condicionais para a criação dos jogos.

As práticas desenvolvidas foram a de reutilizar e remixar e a de abstrair e modularizar. Muitos elementos do programa foram inspirados ou reutilizados de códigos que a turma já tinha utilizado nas outras atividades, como, por exemplo, a condição de tocar em uma cor determinada para estabelecer a vitória no jogo, que também havia sido utilizada na atividade do jogo de corrida. Isso caracteriza a habilidade reutilizar e remixar (BRENNAN, RESNICK, 2012).

Segundo Brennan e Resnick (2012, p.9, tradução nossa), "No *Scratch*, os usuários aplicam a abstração e modularização em múltiplos níveis, desde o conceito inicial do trabalho, de conceituar o problema, até traduzir a resolução do problema em códigos". No projeto proposto, essa prática esteve muito presente. A conceituação do que seria o jogo, como ele funcionaria e quais as partes que precisam ser programadas para funcionar juntas foi realizada pelos estudantes com muito sucesso, caracterizando uma boa prática computacional.

5. PRODUTO EDUCACIONAL

Como produto educacional, foi desenvolvido um plano de ensino (Apêndice D), na forma de sequência didática, que será disponibilizado no website do Programa de Pós-Graduação em Ensino de Ciências e Matemática, junto com esta dissertação, para professores e educadores com interesse no desenvolvimento de habilidades do Pensamento Computacional em estudantes.

Segundo Zabala (2007, p. 18), uma sequência didática é "um conjunto de atividades ordenadas, estruturadas e articuladas para a realização de certos objetivos educacionais, que têm um princípio e um fim conhecido tanto pelos professores como pelos estudantes".

Tendo em vista que "o plano de ensino deve ser norteado pelo perfil do estudante que o curso vai atender" (SPUDEIT, 2014, p. 2), a adaptação de algumas atividades pode ser necessária, caso seja pensado em outra aplicação da sequência didática. As oito atividades utilizadas nesta pesquisa estão presentes no guia didático, com objetivos específicos detalhados e com algumas possibilidades de adaptação para educadores.

Exemplos de adaptação são o trabalho em grupos maiores, caso o número de computadores não seja suficiente, o uso do Scratch offline, caso não haja conexão com a internet e o direcionamento das atividades conforme os interesses da turma.

Acredita-se que este produto educacional é um documento importante e uma possibilidade para a reflexão sobre as práticas educacionais atuais.

6. CONSIDERAÇÕES FINAIS

Tendo em vista os resultados, em relação aos conceitos e práticas visando o Pensamento Computacional, retornamos para nossa questão de pesquisa **"De que maneira atividades de programação realizadas no *Scratch* podem influenciar no desenvolvimento do Pensamento Computacional em crianças e adolescentes que estão no Ensino Fundamental?"**

É claro que existe uma pluralidade de caminhos que podem ser percorridos para este objetivo. As atividades propostas neste trabalho refletiram de maneira positiva no desenvolvimento do Pensamento Computacional de crianças e adolescentes do Ensino Fundamental e, levando em consideração o caminho desenvolvido nesta pesquisa, encontramos sucesso em nossas expectativas.

Esta pesquisa, junto com seu produto educacional, são um esforço para trazer para as escolas brasileiras práticas que visem o desenvolvimento de habilidades para o estudante do século XXI. O produto educacional foi concebido de maneira que qualquer escola com um laboratório de informática básico possa adotar e adaptar este desafio ao seu corpo discente: desenvolver habilidades do Pensamento Computacional.

Uma forte corrente de educadores vem defendendo o desenvolvimento destas habilidades no currículo das escolas, e a tendência do uso das tecnologias para a educação já é realidade no mundo todo. Além disso, foi muito importante adotar uma abordagem que permita que isso seja acessível para o maior número de instituições de ensino possível. Isso tudo inspirou na escolha do *software Scratch*, que é gratuito.

A divulgação da sequência didática aqui desenvolvida será feita de maneira digital, on-line e gratuita para que escolas de qualquer lugar do Brasil ou falantes de língua portuguesa possam se beneficiar destes resultados. O bom educador vai mais longe: vai pensar e repensar as reflexões apresentadas neste documento e adaptá-las para a sua realidade, e com base nisto tem a possibilidade de criar momentos importantes de aprendizagem com as suas turmas.

Dentro deste contexto, o produto educacional e os resultados da pesquisa apresentada poderão trazer uma possibilidade nova para educadores das escolas que estão engajados em desenvolver habilidades do século XXI em seus alunos. São contribuições para algo maior: o processo de mudança que a Educação está passando. Professores, nativos digitais ou não, podem trabalhar com os resultados apresentados e também criar histórias de sucesso.

Muitas possibilidades também surgem deste trabalho. Uma das principais dificuldades encontradas foi a falta de material de qualidade – especialmente em português. Nesta dissertação

foi utilizado grande parte do material em inglês, pois considerou-se que este possuía uma qualidade geralmente superior aos materiais em português encontrados.

Das reflexões feitas aqui, algumas possibilidades de trabalhos futuros surgem. A avaliação dos encontros se deu, principalmente, à luz dos conceitos e práticas de Brennan e Resnick (2012), já quanto às perspectivas computacionais, os próprios autores colocam que existe muita dificuldade em realizar a sua avaliação. A avaliação do Pensamento Computacional é, na verdade, uma das maiores dificuldades que apareceram – tanto para encontrar materiais teóricos sobre o assunto, quanto para adequar o que existe para as atividades propostas. Essa é uma área que precisa de mais pesquisadores para ganhar força.

A Educação *Maker* também foi grande influenciadora para o material desenvolvido. Uma área em ascensão, especialmente com a robótica educacional se popularizando em escolas particulares, o Brasil tem grande potencial para desenvolver pesquisas e pode se tornar autoridade na área.

O caminho já foi traçado e as dificuldades são muitas. Este trabalho é, então, um esforço para trazer aos ambientes educacionais do Brasil cada vez mais oportunidades de inovação. É, também, uma grande reflexão sobre as prioridades da Escola. Por fim, é um convite para que cada vez mais pesquisadores e educadores se juntem para solucionar o desafio que é implementar o desenvolvimento do Pensamento Computacional nas atividades das escolas brasileiras.

7. REFERÊNCIAS

ACKERMANN, Edith. Piaget's constructivism, Papert's constructionism: What's the difference. **Future of learning group publication**, v. 5, n. 3, p. 438, 2001.

ANDERSON, L.W. , KRATHWOHL, D.R. , AIRASIAN, P.W., CRUIKSHANK, K.A., MAYER, R.E., PINTRICH, P.R., RATHS, J., & WITTROCK, M.C. **A taxonomy for learning, teaching, and assessing: A revision of Bloom's Taxonomy of Educational Objectives** (Complete edition). New York: Longman, 2001.

ANDRADE, Daiane et al. Proposta de atividades para o desenvolvimento do pensamento computacional no ensino fundamental. In: **Anais do Workshop de Informática na Escola**. 2013. p. 169.

ANGOTTI, José André Peres; AUTH, Milton Antonio. Ciência e tecnologia: implicações sociais e o papel da educação. **Ciência & Educação** (Bauru), v. 7, n. 1, p. 15-27, 2001.

ARAUJO, Ana Liz; ANDRADE, Wilkerson; GUERRERO, Dalton. Um mapeamento sistemático sobre a avaliaçao do pensamento computacional no brasil. In: **Anais dos Workshops do Congresso Brasileiro de Informática na Educação**. 2016. p. 1147 – também disponível na pasta "Referências de última última hora").

BANDEIRA, Lourdes. A contribuição da crítica feminista à ciência. **Estudos Feministas**, v. 16, n. 1, p. 207-228, 2008.

BARCELOS, Thiago Schumacher; SILVEIRA, Ismar Frango. Pensamento computacional e educação matemática: Relações para o ensino de computação na educação básica. In: XX Workshop sobre Educação em Computação, Curitiba. Anais do XXXII CSBC. 2012. p. 23.

BARR, David; HARRISON, John; CONERY, Leslie. Computational thinking: A digital age skill for everyone. **Learning & Leading with Technology**, v. 38, n. 6, p. 20-23, 2011.

BARR, Valerie; STEPHENSON, Chris. Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community?. Acm Inroads, v. 2, n. 1, p. 48-54, 2011.

BASU, S., MCELHANEY, K. W., GROVER, S., HARRIS, C. J., & BISWAS, G. (2018). A principled approach to designing assessments that integrate science and computational thinking. International Society of the Learning Sciences, Inc.[ISLS]..

BENNETT, Sue; MATON, Karl; KERVIN, Lisa. The 'digital natives' debate: A critical review of the evidence. **British journal of educational technology**, v. 39, n. 5, p. 775-786, 2008.

BIGGS, J.; COLLIS, K. **Evaluating the quality of learning: the SOLO taxonomy**. New York: Academic Press, 1982.

BLOOM, B. S. et al. **Taxonomy of educational objectives**. New York: David Mckay, 1956. 262 p. (v. 1)

BRACKMANN, Christian Puhlmann. Desenvolvimento do pensamento computacional através de atividades desplugadas na educação básica. 2017.

BRASIL. Lei n. 9.394, de 20 de Dezembro de 1996. **Lei de Diretrizes e Bases da Educação Nacional**, Brasília, DF, dez 1996.

BRENNAN, Karen; BALCH Christian; CHUNG, Michelle. Creative Computing. **Harvard Graduate School of Education**, 2014.

BRENNAN, Karen; RESNICK, Mitchel. New frameworks for studying and assessing the CABRAL, Ronaldo Vieira. O ensino de matemática e a informática: uso do scratch como ferramenta para o ensino e aprendizagem da geometria. **Produção de terceiros sobre Paulo Freire; Dissertações**, 2015.

CAMINHO DO SABER. Proposta Pedagógica, 2018. Disponível em: < http://www.redecaminhodosaber.com.br/portal/metodo> Acesso em: 23 jun. 2018.

CASTELLS, Manuel. A Galáxia da Internet: Reflexões sobre a Internet, os negócios e a sociedade. Rio de Janeiro: Jorge Zahar Editor, 2003.

COFFEY, Heather. Bloom's taxonomy. **Chapel Hill, North Carolina**, 2008.

COUTINHO, Clara Pereira; LISBÔA, Eliana Santana. Sociedade da informação, do conhecimento e da aprendizagem: desafios para educação no século XXI. **Revista de Educação**, v. 18, n. 1, p. 5-22, 2011. Disponível em: <https://repositorium.sdum.uminho.pt/handle/1822/14854>. Acesso em: 19 fev. 2018.

DE FRANÇA, Rozelma Soares; DO AMARAL, Haroldo José Costa. Proposta metodológica de ensino e avaliaçao para o desenvolvimento do Pensamento Computacional com o uso do scratch. In: **Anais do Workshop de Informática na Escola**. 2013. p. 179.

DELGADO, Carla et al. Uma abordagem pedagógica para a iniciação ao estudo de algoritmos. In: **XII Workshop de Educação em Computação**. 2004.

development of computational thinking. In: **Proceedings of the 2012 annual meeting of the** FALLOON, Garry. An analysis of young students' thinking when completing basic coding tasks using Scratch Jnr. On the iPad. **Journal of Computer Assisted Learning**, v. 32, n. 6, p. 576-593, 2016.

FERRAZ, A. P. C. M; BELHOT, R. V. Taxionomia de Bloom: revisão teórica e apresentação das adequações do instrumento para definição de objetivos instrucionais. **Gest. Prod., São Carlos**, v. 17, n. 2, p. 421-431, 2010.

FONSECA, J. J. S. Metodologia da pesquisa científica. Fortaleza: UEC, 2002. Apostila.

GERHARDT, Tatiana Engel; SILVEIRA, Denise Tolfo. **Métodos de pesquisa**. PLAGEDER, 2009.

GROVER, Shuchi. Assessing algorithmic and computational thinking in K-12: Lessons from a middle school classroom. In: **Emerging research, practice, and policy on computational thinking**. Springer, Cham, 2017. p. 269-288.
GROVER, Shuchi. Systems of Assessments" for deeper learning of computational thinking in K-12. In: **Proceedings of the 2015 annual meeting of the American educational research association**. 2015. p. 15-20.

GROVER, Shuchi; PEA, Roy. Computational thinking in K–12: A review of the state of the field. **Educational Researcher**, v. 42, n. 1, p. 38-43, 2013.

HARGREAVES, Andy. O Ensino na Sociedade do Conhecimento: a educação na era da insegurança. **Colecção Currículo, Políticas e Práticas**. Porto: Porto Editora, 2003.

HARVEY, B., MÖNIG, J. Bringing "no ceiling" to Scratch: Can one language serve kids and computer scientists. **Proc. Constructionism**, 2010.

HEER, Rex. A Model of Learning Objectives-based on A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives. **Center for Excellence in Learning and Teaching, Iowa State University**, 2012. Disponível em: <http://www.celt.iastate.edu/wp-content/uploads/2015/09/RevisedBloomsHandout-1.pdf>. Acesso em Junho de 2018.

IBGE - Instituto Brasileiro de Geografia e Estatística. Disponível em: <http://www.ibge.gov.br/>. Acesso em: 19 mai. 2016.

ISTE. Operational Definition of Computational Thinking for K-12 Education. 2011. Disponível em: <http://www.iste.org/docs/ct-documents/computational-thinking-operational-definition-flyer.pdf>. Acesso em: 02 de mar. De 2018.

KALELIOGLU, Filiz; GÜLBAHAR, Yasemin; KUKUL, Volkan. A Framework for Computational Thinking Based on a Systematic Research Review. **Baltic Journal of Modern Computing**, v. 4, n. 3, p. 583, 2016.

KENNEDY, Gregor E. et al. First year students' experiences with technology: Are they really digital natives?. **Australasian journal of educational technology**, v. 24, n. 1, 2008. Disponível em: < https://ajet.org.au/index.php/AJET/article/view/1233>. Acesso em: 20 fev. 2018.

KRATHWOHL, David R. **A revision of Bloom's taxonomy: An overview**. Theory into practice, v. 41, n. 4, p. 212-218, 2002.
KURTI, R. Steven; KURTI, Debby L.; FLEMING, Laura. The philosophy of educational makerspaces part 1 of making an educational makerspace. **Teacher Librarian**, v. 41, n. 5, p. 8, 2014.

MEERBAUM-SALANT, Orni; ARMONI, Michal; BEN-ARI, Mordechai. Learning computer science concepts with scratch. **Computer Science Education**, v. 23, n. 3, p. 239-264, 2013.

MIGUEL, Antonio; MIORIM, Maria Ângela. **História na educação matemática: propostas e desafios**. Autêntica, 2013.

MONROY-HERNÁNDEZ A., RESNICK M. Empowering Kids to Create and Share Programmable Media. **Interactions**, v. 15, n. 2, p 50-53, 2008.

OLIVEIRA, MLS de et al. Ensino de lógica de programação no ensino fundamental utilizando o Scratch: um relato de experiência. In: **XXXIV Congresso da SBC-XXII Workshop de Ensino de Computaç ao,** Brasılia. sn, 2014.

P21. Framework for 21st Century Learning, 2016. Disponível em: <http://www.p21.org/storage/documents/docs/P21_framework_0816.pdf>. Acesso em: 15 abr. 2018.

PAPERT, Seymour. An exploration in the space of mathematics educations. **International Journal of Computers for Mathematical Learning**, v. 1, n. 1, p. 95-123, 1996.

PAPERT, Seymour. **Mindstorms: Children, computers, and powerful ideas**. Basic Books, Inc., 1980.

PAPERT, Seymour. Teaching Children to be Mathematicians vs. Teaching About Mathematics, **Massachusetts Institute of Technology**, Cambridge, MA, 1971.

PAPERT, Seymour; **A máquina das crianças: repensando a escola na era da informática**. Trad. Sandra Costa. ed.rev. Porto Alegre: Artmed, 2008, 224p.

PAPERT, Seymour; HAREL, Idit. Situating constructionism. **Constructionism**, v. 36, p. 1-11, 1991.

POZO, Juan Ignacio. A sociedade da aprendizagem e o desafio de converter informação em conhecimento. **Pátio: Revista Pedagógica**, v. 31, p. 8-11, 2004. Disponível em: <http://decampinasoeste.edunet.sp.gov.br/tics/Material%20de%20Apoio/Coletania/unidade1/A%20sociedade%20da%20aprendizagem%20e%20o%20desafio%20de%20converter%20informa%C3%A7%C3%A3o%20em%20conhecime.pdf>. Acesso em: 10 fev. 2018.

PRENSKY, Marc. Digital natives, digital immigrants part 1. **On the horizon**, v. 9, n. 5, p. 1-6, 2001. Disponível em: < https://www.marcprensky.com/writing/Prensky%20-%20Digital%20Natives,%20Digital%20Immigrants%20-%20Part1.pdf>. Acesso em: 20 fev. 2018.

RODRIGUEZ, Carla et al. Pensamento Computacional: transformando ideias em jogos digitais usando o Scratch. In: **Anais do Workshop de Informática na Escola**. 2015. p. 62.

RODRIGUEZ, Carla et al. Pensamento Computacional: transformando ideias em jogos digitais usando o Scratch. In: **Anais do Workshop de Informática na Escola**. 2015. p. 62.

ROMÁN-GONZÁLEZ, Marcos; MORENO-LEÓN, Jesús; ROBLES, Gregorio. Complementary tools for computational thinking assessment. In: **Proceedings of International Conference on Computational Thinking Education (CTE 2017), S. C Kong, J Sheldon, and K. Y Li (Eds.). The Education University of Hong Kong**. 2017. p. 154-159.

ROYAL SOCIETY. After the reboot: computing education in UK schools, 2017. Disponível em: < https://royalsociety.org/~/media/policy/projects/computing-education/computing-education-report.pdf> Acesso em: 13 abr. 2018.

ROYAL SOCIETY. Shut down or restart: The way forward for computing in UK schools. 2012. Disponível em: <https://royalsociety.org/~/media/education/computing-in-schools/2012-01-12-computing-in-schools.pdf>. Acesso em: 02 mar. 2018.

ROYAL SOCIETY. Vision for Science and mathematics education, 2014. Disponível em: < https://royalsociety.org/~/media/policy/projects/computing-education/computing-education-report.pdf> Acesso em: 13 abr. 2018.

SCAICO, Pasqueline Dantas et al. Programação no ensino médio: uma abordagem de ensino orientado ao design com Scratch. In: **Anais do Workshop de Informática na Escola**. 2012.

SCHOEFFEL, Pablo et al. Uma Experiência no Ensino de Pensamento Computacional e Fomento à Participação na Olímpiada Brasileira de Informática com Alunos do Ensino Fundamental. In: **Anais do workshop do IV Congresso Brasileiros de Informática na Educação** (CBIE 2015). 2015.

SEAMAN, Mark. BLOOM'S TAXONOMY. **Curriculum & Teaching Dialogue**, v. 13, 2011.

SELBY, Cynthia C. Relationships: computational thinking, pedagogy of programming, and Bloom's Taxonomy. In: **Proceedings of the Workshop in Primary and Secondary Computing Education**. ACM, 2015. p. 80-87.

TURKLE, Sherry; PAPERT, Seymour. Epistemological pluralism: Styles and voices within the computer culture. **Signs**, v. 16, n. 1, p. 128-157, 1990.

UKCES - UK Commission for Employment and Skills. The future of work: jobs and skills in 2030. Londres: UKCES, 2014.

VIDAL, Cristian L. et al. Experiencias Prácticas con el Uso del Lenguaje de Programación Scratch para Desarrollar el Pensamiento Algorítmico de Estudiantes en Chile. Formación universitaria, v. 8, n. 4, p. 23-32, 2015.

VON WANGENHEIM, Christiane Gresse; NUNES, Vinícius Rodrigues; DOS SANTOS, Giovane Daniel. Ensino de computaçao com scratch no ensino fundamental–um estudo de caso. **Revista Brasileira de Informática na Educação**, v. 22, n. 3, p. 115-125, 2014.

WILSON, Leslie Owen. Anderson and Krathwohl–Understanding the new version of Bloom's taxonomy. **The second principle. The work of Leslie Owen Wilson. Ed. D,** 2016.

WING, Jeanette M. A Conversation about Computational Thinking. **Future frontiers: Education for an AI world**, p. 127-139, 2017.

WING, Jeannette M. Computational thinking. **Communications of the ACM**, v. 49, n. 3, p. 33-35, 2006.

ZABALA, A. **A prática educativa: como ensinar.** Porto Alegre: Artmed. 2007.

8. APÊNDICE A – REGISTRO DAS RESPOSTAS – DEBUG IT!

Como você resolveu os problemas dos projetos:

1

COLOCAMOS O BLOCO, QUANDO CLICAR NA BANDEIRA NO CONTROLE DO GOBO

2

TIRAR 90 GRAU PORQUE TINHA 4 QUE DANA 360 DAI NÃO DANA PRA VER ELE SE MOVENDO DAI TEVE QUE TIRAR UM PRA DEBUG NELE.

3

NÃO CONSEGUIMOS FAZER O DESAFIO!

4

A GENTE GRAVOU PARA CONSEGUIR RESOLVER O PROBLEMA, E INVERTEU O SOM E O BALÃO.

Como você resolveu os problemas dos projetos:

1

QUANDO CLICAR EM BANDEIR UM DANÇA CLICA NO OUTRO PERSONAGEN E ELES DANÇAM JUNTOS

2

NÓS CHUTAMOS 100 GRAUS E DEU CERTO MAS TENQUE FICAR SIGURANDO O ESPAÇO.

3

NÃO CONSIGUIMOS

4

SO TROCAR DE LUGAR O DIGA MIAU NO LUGAR O TOQUE O SOM

Como você resolveu os problemas dos projetos:

1

Colocamos o bloco quando dicon em 4

2

Colocamos blocos de movimento ~~com~~ e ~~números diferentes~~ pedimos para repitir 10x

3

Não conseguimos

4

Colocamos 3 blocos dentro do repita 3x

Como você resolveu os problemas dos projetos:

1

Nós achamos o icone que estava faltando que era a bandeirinha verde

2

Porque a gente botou 15 graus seusios e ficou clicando no espaso.

Foi no chute que a gente botou 15.

3

4

Botando o "toque o som Meow" Junto com o som diga Meow 3 vezes por 1 segundo

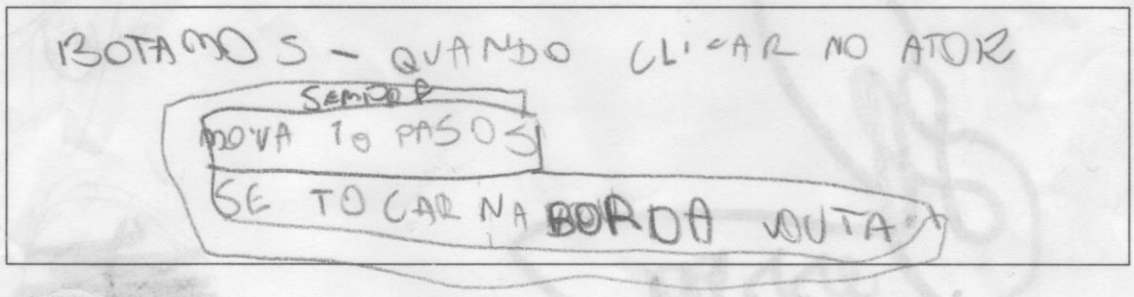
1

Botamos o bloco "Quando clicar na"

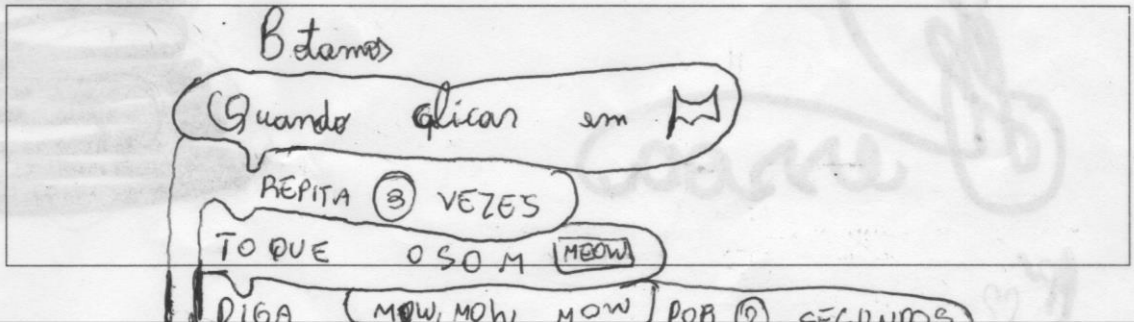
2

Botamos a tecla "Sempre" entre "Quando clicar espaço" e no "Diz 'O gato'"

3



4



Como você resolveu os problemas dos projetos:

1

Colocamos o ponto de partida

2

~~1~~ testando
~~Testamos~~ giros noventa graus e colocamos para
10~~0~~ graus

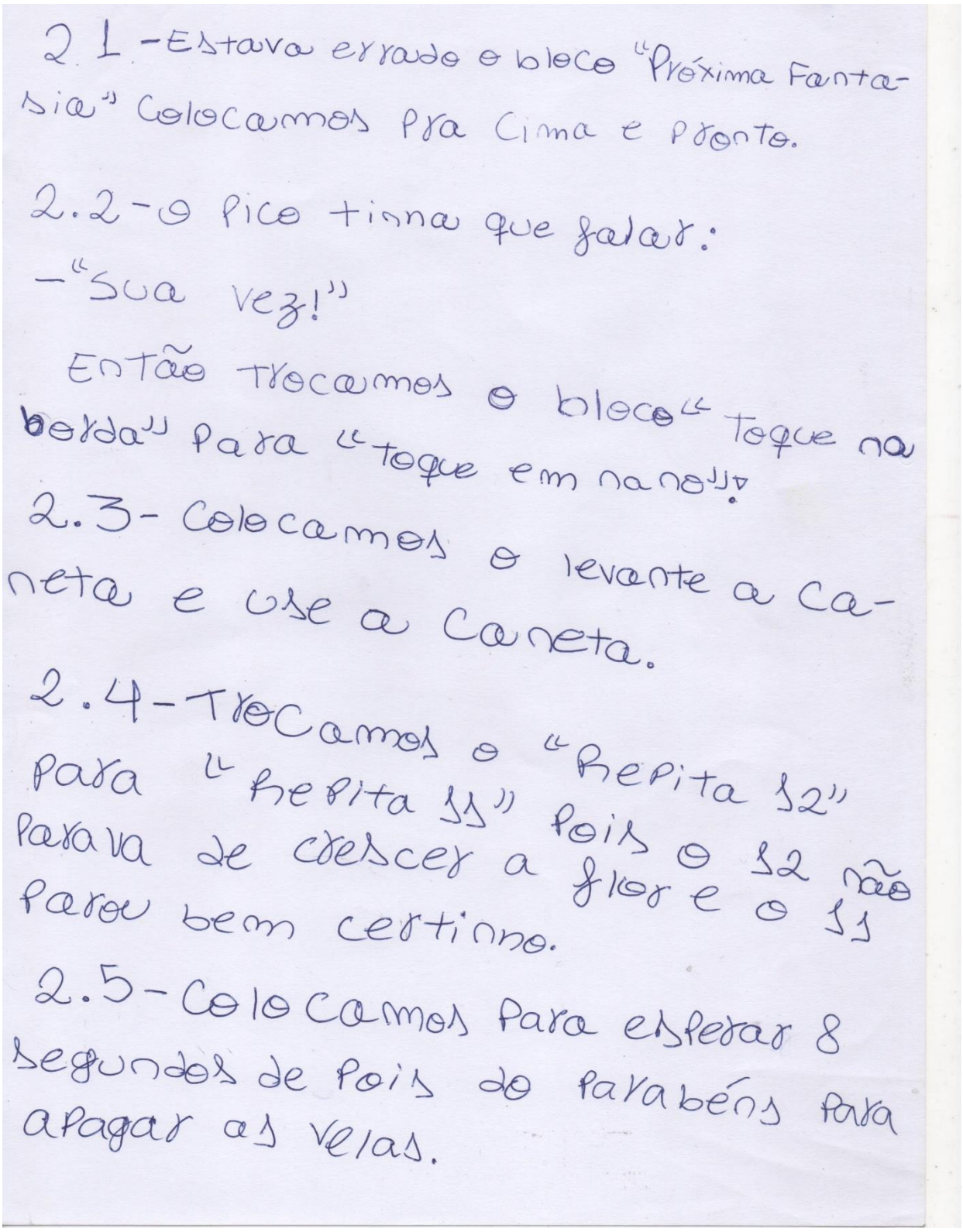
3

~~ROTAMOS~~ BOTAMOS O MUDE O ESTILO DE
DE ROTAÇÃO PARA ESQUERDA-DIREITA

4

Botando atoque o som dentro de repita 3
vezes.

9. APÊNDICE B – REGISTRO DAS RESPOSTAS – DEBUG IT! 2.0



2.1 - Estava errado o bloco "Próxima Fantasia" Colocamos pra Cima e pronto.

2.2 - o Pico tinha que falar:
- "Sua vez!"

Então Trocamos o bloco "toque na borda" para "toque em nano".

2.3 - Colocamos o levante a caneta e use a caneta.

2.4 - Trocamos o "Repita 12" para "Repita 11" pois o 12 não parava de crescer a flor e o 11 parou bem certinho.

2.5 - Colocamos para esperar 8 segundos de pois do parabéns para apagar as velas.

1- NÓS MUDAMOS A MÚSICA, TIRAMOS MUDAR A FANTASIA E MUDAMOS OS GRAUS.

2- COLOCAMOS UM BLOCO DENTRO DO OUTRO. PORQUE ELE TENQUE VERIFICAR SE ESTÁ TOCANDO NO NANO.

3- COLOCAMOS DOIS BLOCOS A MAIS DE "LEVANTE A CANENA" E "USE A CANETA".

4- NÓS MUDAMOS UM BLOCO QUE, [illegible] EM VEZ DE MAIOR QUE 12, IGUAL A 12.

5 - COLOCAMOS PARA ESPERAR 1 SEG. E TOCAR "PARABÉNS PARA VOCÊ".

2.1 - Nós botamos a Palavra "trocar fantasia" para cima, para ele conseguir dançar com a batida
2.2 - Nós colocamos o 2º quadro no primeiro.
2.3 - Nós colocamos "levante a caneta" depois "Use a caneta" e Colocamos
2.4 - NÓS COLOCAMOS O 11 EM VEZES DO 12 PORQUE IA FICAR MELHOR..

2.1: O "TROQUE A FANTASIA" ESTAVA FORA DO "REPITA DEZ VEZES". BOTAMOS O "TROQUE A FANTASIA" ENTRE O "TOQUE O TAMBOR 1 POR 0,01 BATIDAS" E O "ESPERE 0,01 SEGUNDO" E ASSIM FUNCIONOU.

2.2: BOTAMOS O BLOCO "ENVIAR MENSAGEM" E O "RECEBER MENSAGEM"

2.3: Colocamos levante a caneta e abaixe a caneta

2.4 MUDAMOS O NUMERO DA FANTASIA DE 12 PARA 11

2.5 botamos 10 segundos em cima do bloco diga clique para apagar as telas por 3 segundos.

2.1- Se mudarmos o bloco para cima ganhamos

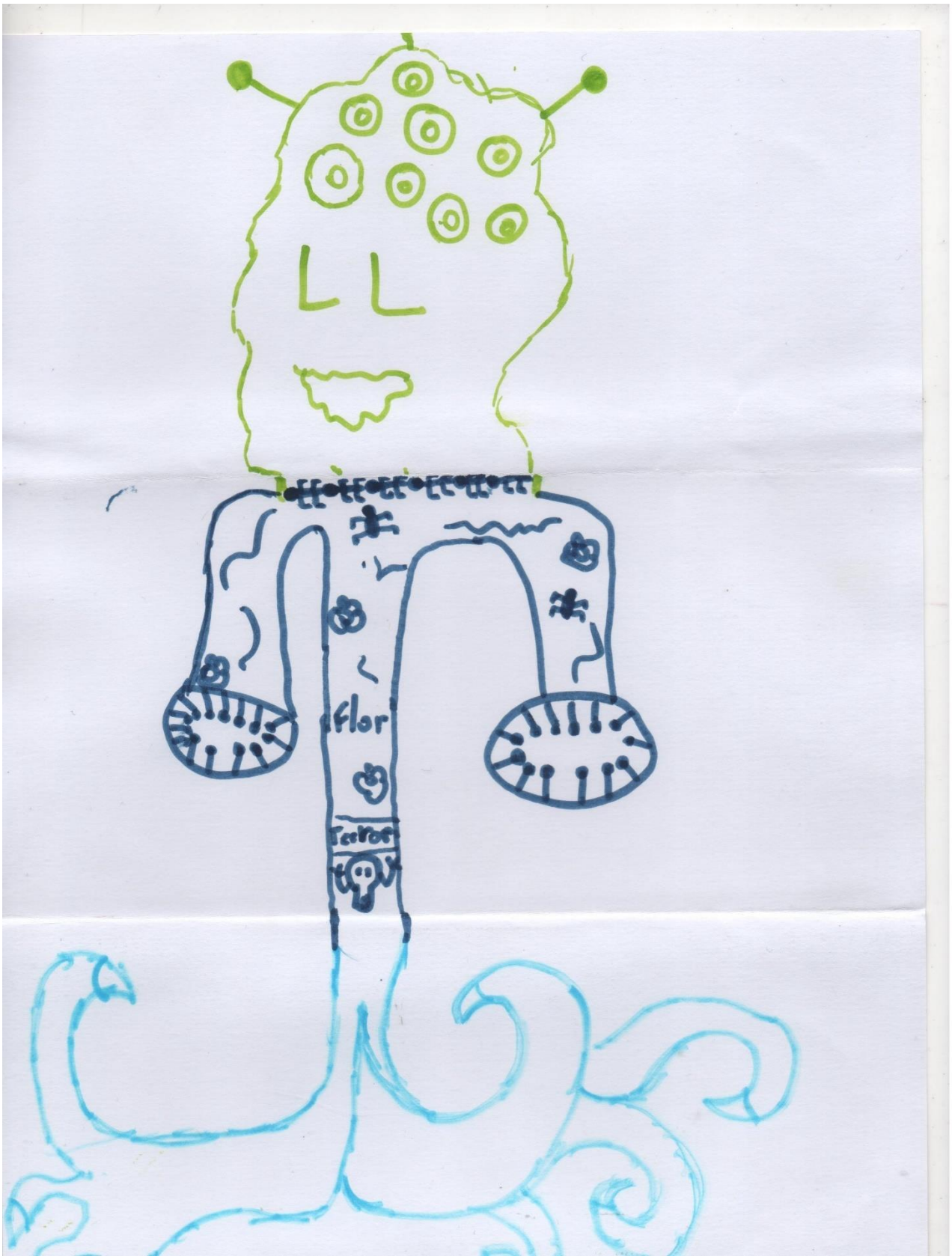
2.2- Mudamos de lugar o bloco diga= Sua vez de lugar e tiramos o bloco se tocando em borda? então.

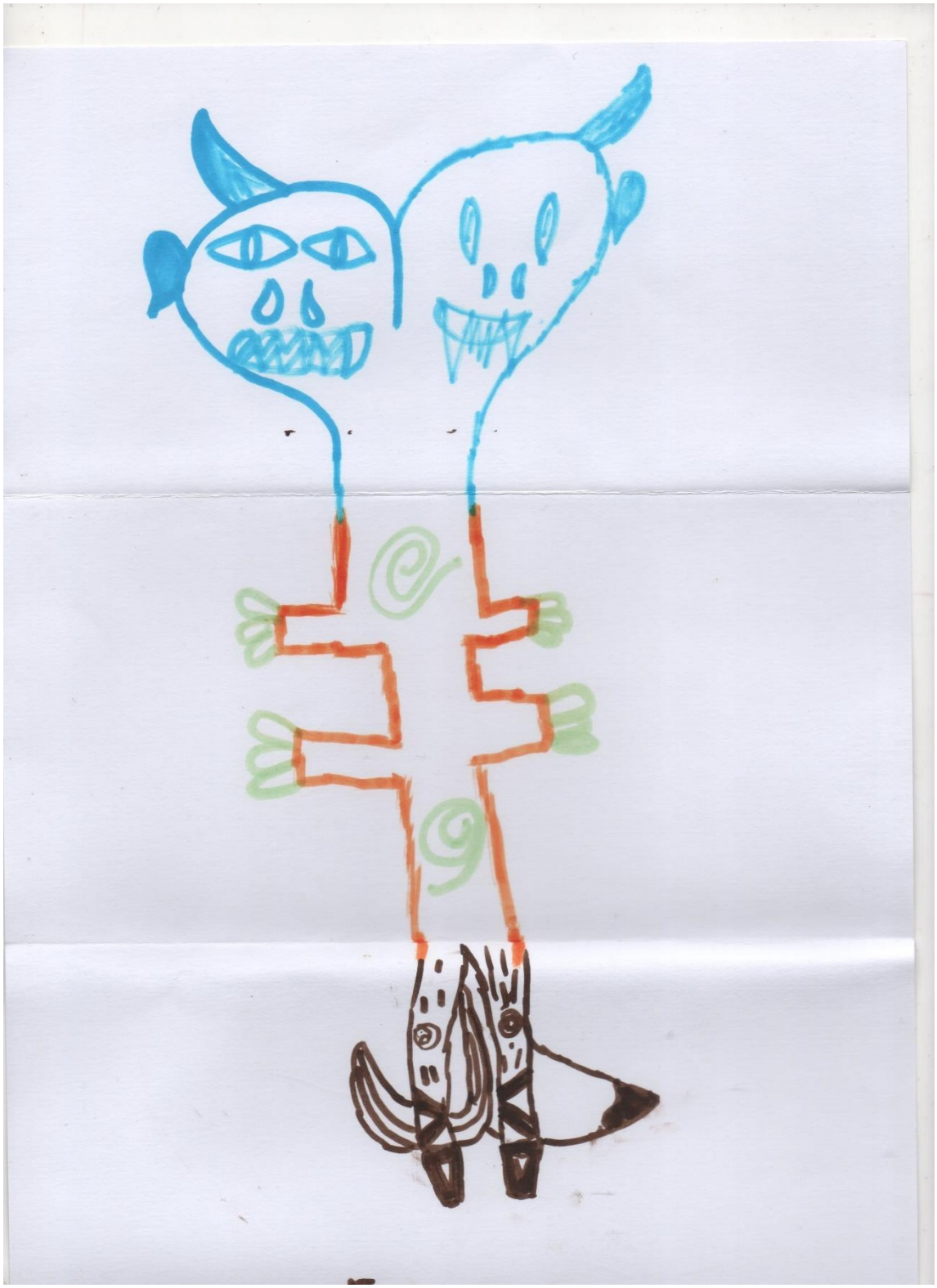
2.3- Colocamos o bloco levante a caneta encima dos blocos vá para o lado e va para tantos graus assim levantou a caneta desenhou e conseguimos

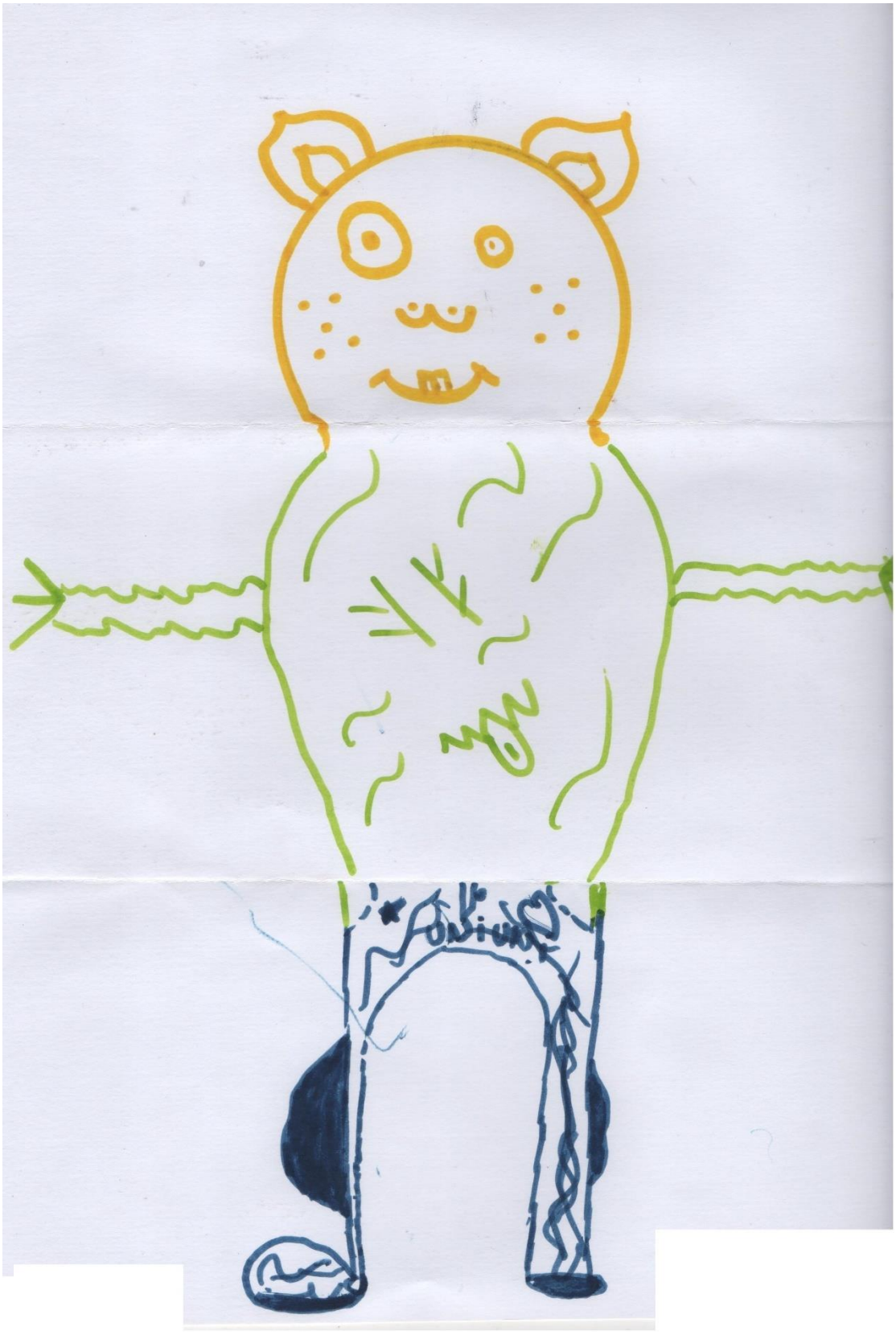
2.4- Coloque 11 vezes espere 0,2 segundos, coloque também 12 vezes (prosima fantasia)

2.5- Trocamos o bloco "toque o som birthday" por "Toque o Som birthday até o fim".

10. APÊNDICE C – PRODUTO DA ATIVIDADE DE CRIAÇÃO COMPARTILHADA







11. APÊNDICE D – PRODUTO EDUCACIONAL

Caro educador!

Este guia é o produto educacional de uma pesquisa do Programa de Pós-Graduação em Ensino de Ciências e Matemática que tem como objetivo o desenvolvimento de habilidades do Pensamento Computacional para estudantes do 4º a 7º ano do Ensino Fundamental.

Abaixo estão descritos os encontros a serem realizados. No total são 8 encontros de aproximadamente uma hora e meia cada um, passível de adaptação.

Para o desenvolvimento das atividades é necessário computadores simples com acesso a internet (um computador para cada grupo de 4 alunos é suficiente) ou com o software Scratch instalado. Também será utilizado um projetor.

Para mais informações sobre o software, visite o website https://scratch.mit.edu/, que conta com o próprio software para download ou para uso on-line, além de tutoriais e guias.

Encontro 1 – Introdução: Construindo nosso primeiro programa

Ao iniciar uma sequência didática é importante apresentar aos estudantes a proposta deste, o objetivo, o método a ser utilizado e, especialmente se tratando de atividades que envolvem um software específico, é necessário que ele seja apresentado aos estudantes de maneira que não se sintam intimidados.

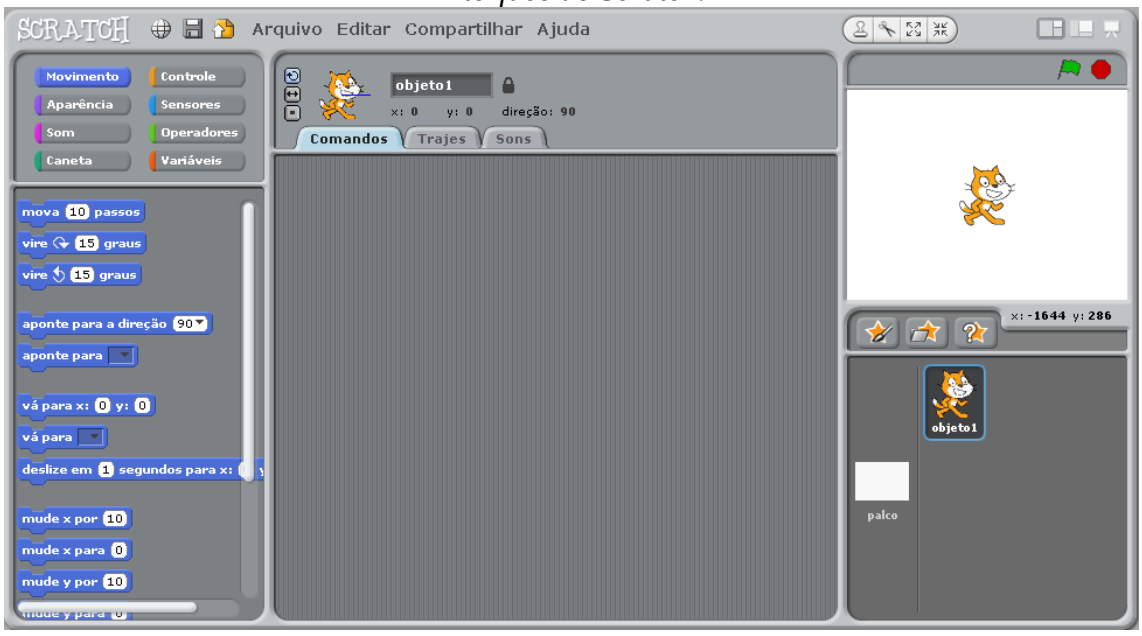
Descrição da Atividade

Aos estudantes, explicar o funcionamento dos encontros: quando acontecerão, como acontecerão, por que estarão realizando essas atividades (objetivo) e o que é esperado deles. Da mesma maneira, os estudantes serão questionados em relação a suas expectativas.

Introdução ao *Scratch*

Utilizando um projetor, apresentar aos estudantes a interface do software *Scratch*.

Interface do Scratch.



Deve ser feita uma apresentação breve sobre como funciona o software e a linguagem de programação em blocos, específica deste. Não se espera que os estudantes aprendam todos os tópicos comentados, esse aprendizado virá conforme eles agem sobre o objeto.

Definindo o Problema

Agora é a hora de construir o primeiro programa destes estudantes (daqueles que nunca tiveram experiências em programação), que será uma animação simples com a personagem gato do *Scratch*.

Este programa será feito por todos, com acompanhamento do professor, na tela que está sendo projetada. Devem, então, definir um roteiro para a animação.

A animação deve ser algo simples, que pode ser adaptada conforme sugestões dos estudantes. Pode ser, por exemplo, animar a personagem para dar alguns passos para frente e depois dizer "Olá, meu nome é *Scratch*!".

É importante que o professor esteja aberto a sugestões dos estudantes, para customizar[24] o programa e fazer com que este tenha significado para toda a turma. Neste texto, iremos tratar a animação como descrito acima.

Durante a construção do programa, o professor deve sempre encorajar os estudantes a utilizarem habilidades do Pensamento Computacional (vamos testar?, como podemos separar esta animação em partes?, algo está errado, será que conseguimos consertar?). Vamos utilizar essas habilidades, que queremos desenvolver, para a resolução deste problema.

O professor irá encorajar o estudante a **decompor o problema** em partes menores. Neste caso, podemos dividir a animação em duas partes: movimentar a personagem e a fazer falar. Vamos analisar cada parte separadamente.

Parte 1: Movimentando o Objeto

O professor pode pedir aos estudantes que procurem nos blocos de programação, e encontrem algum que possa fazer com que nosso objeto se movimente. Intuitivamente, os estudantes serão levados a encontrar o bloco de mover.

Bloco de movimento.

Na sequência, irão **realizar o teste**: dê dois cliques no bloco em questão. Eles deverão observar que isso movimentou a personagem no sentido que ela está virada. Podem, então, realizar o teste para ver o que acontece quando aumentam o número de passos.

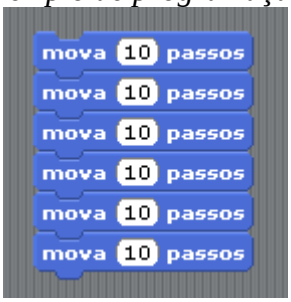
Então, para movimentar o objeto, podem utilizar este bloco. Deve ser observado que ao mover o objeto desta forma, ele desaparece de onde estava e aparece na posição mais adiante,

[24] Customizar é modificar, adaptar ou personalizar algo de modo a adequá-lo às suas necessidades.

dependendo do valor estabelecido. Portanto, apenas este bloco não é suficiente para fazer a personagem caminhar.

O passo mais natural a seguir é o de tentar colocar mais blocos de movimento. É esperado que os estudantes experimentem algumas formas diferentes, e cabe ao professor lidar com elas. Podem testar, todos juntos, algumas sugestões, como o exemplos abaixo.

Exemplo de programação.



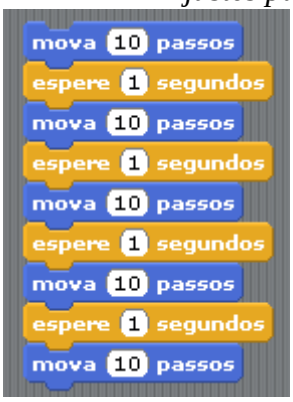
Deparamo-nos, então, com outro problema: é preciso que a personagem dê uma pausa antes de dar o próximo passo. Para isso, desafiamos os estudantes a encontrar uma solução.

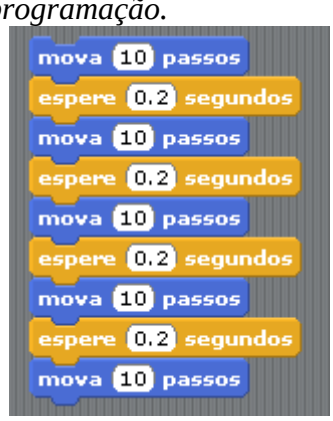
O bloco de espera pode ser uma sugestão dos estudantes. Vamos então, tentar colocar um bloco de espera entre os passos.

Programação com bloco de espera.

Agora a personagem parece que está caminhando. Completamos o nosso programa. Podemos também ajustar o tempo de espera.

Ajustes para a programação.





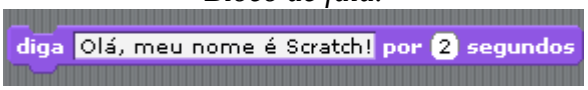
O professor pode ajudar a turma a descobrir que este processo pode ser sintetizado com o bloco de repetição.

Programação simplificada.

Parte 2: Fazendo a Personagem Falar

Para realizar a segunda parte do nosso problema, a turma pode explorar o software para encontrar uma solução. Uma delas pode ser o bloco a seguir.

Bloco de fala.

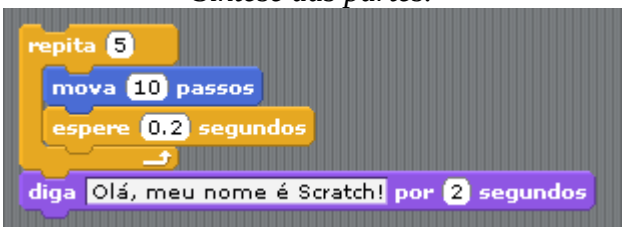


Alternativamente, podem ser utilizados os blocos de som, com gravações. Tudo depende de como a turma encarar o desafio.

Unindo os módulos (síntese)

Fazendo a síntese das partes do problema, conseguimos construir o programa propost.

Síntese das partes.



A realização dessa síntese deve ser discutida também. O conceito de *loop* ainda não foi formalmente apresentado para a turma, e o uso errado do *loop* pode acontecer, como por exemplo colocando o bloco de fala junto com a parte de movimento.

Exemplo de programação com erro (bloco de fala dentro do loop).

Recursos Extras

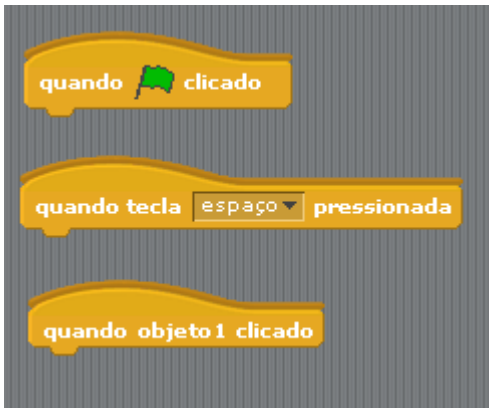
De acordo com o ritmo da turma e a atenção que o professor pretende dar a cada detalhe, é possível que sobre tempo para explorar outras funções do programa.

Uma das funções que pode ser explorada são os trajes, para fazer o movimento parecer mais natural.

Função "trajes".

É possível também explorar os blocos de comando para iniciar o programa sem precisar dar dois cliques no programa.

Blocos de comando.



Exploração Livre

Agora que a turma programou uma pequena animação, devem ser encorajados a modificar esta, ou até mesmo criar uma diferente, com os recursos que quiserem. É o momento de explorarem os recursos do software e sua capacidade intuitiva de programação.

Apresentação

Finalmente, os estudantes irão apresentar seus programas para os colegas. Serão encorajados a explicar o que utilizaram de diferente, se encontraram muitas dificuldades e o que acharam de ser programadores.

Avaliação

A avaliação do encontro será feita de duas maneiras:

a) *Das habilidades do Pensamento Computacional utilizadas*: nos programas apresentados, será feito um levantamento das habilidades e conceitos do Pensamento Computacional que se fizeram presentes. Além disso, durante o encontro essa análise deverá ser feita em todos os momentos pelo professor.

b) *Da validade do encontro para o estudante*: através de um breve relatório, será feito um levantamento de dados sobre os estudantes em relação a suas experiências prévias com programação e primeiras impressões.

Nome	Idade	Turma

Algum dos integrantes da equipe já conhecia o Scratch? Se sim, quem?

O que vocês conseguiram fazer com o Scratch? Como conseguiram isso?

Arraste para o quadrado o *emoji* que representa o que você achou das atividades de hoje:

Encontro 2 – Let's Dance

O Pensamento Computacional é uma habilidade que está presente também quando não estamos programando. Esta atividade inicia com o desafio de "programar uma pessoa", e segue com a transposição deste desafio para a interface do *Scratch*.

A ideia de programar uma pessoa é simplesmente estabelecer uma sequência de passos para resolver um problema. Nesse caso, o problema é ensinar uma dança a uma pessoa sem utilizar elementos visuais, apenas explicando os passos.

Descrição da Atividade

Separados em duplas, será designado a cada estudante um papel: programador ou programa. Cada dupla deverá ser composta de um programador e um programa.

Vamos dançar!

Aos programadores será mostrado um vídeo de dança. Ele deverá passar ao programa as instruções através da fala. Será proibido ao programador utilizar gestos ou mostrar a dança de alguma maneira que não por comandos verbais.

O outro integrante, o programa, deve então realizar a dança conforme entender, e depois irá comparar seu resultado com a dança original.

Será feito uma pequena discussão sobre a importância da clareza quando estamos programando e nas tarefas do cotidiano.

Festa do *Scratch*!

Ainda separados em duplas, os estudantes irão programar uma "festa do *Scratch*". Para isso, cada dupla seguirá o tutorial[25] e, em seguida, será encorajada a personalizar seu projeto adicionando objetos, alterando a dança, o pano de fundo e explorando os sons. O desafio é fazer a animação de uma festa de dança.

Cada grupo irá apresentar sua animação para a turma. Enquanto isso, devem preencher fichas de *feedback*, que serão entregues para a avaliação por pares.

[25] https://scratch.mit.edu/projects/editor/?tip_bar=getStarted

Ficha de feedback.

FICHA DE FEEDBACK	O que você acha que poderia ter sido melhor?	Você achou algo confuso, que poderia ter sido feito de outra forma?	O que você achou legal e que funciona bem no projeto?
DE: grupo ___			
PARA: grupo ___			

Encontro 3 – *Debug it*!

Uma das habilidades do Pensamento Computacional é a habilidade de depurar: encontrar erros e consertá-los. Com este objetivo, neste encontro serão propostos desafios na forma de programas que estão com erros. Cabe a cada grupo encontrar os erros e consertá-los.

Descrição da Atividade

Separados em grupos, os estudantes irão receber os seguintes programas com seus respectivos erros (*bugs*). Eles devem encontrar a parte do programa que está causando este erro e consertá-la.

Será solicitado que eles registrem em um pequeno formulário qual era o erro do programa e como ele foi corrigido.

No final da aula, serão discutidas as soluções.

Desafio 1: Gobo está Parado!

Neste programa, tanto o gato do *Scratch* quanto o Gobo (personagem amarelo) deveriam dançar quando a bandeira verde é clicada, mas isso não acontece. Cabe a cada equipe encontrar o erro e consertá-lo, do seu jeito.

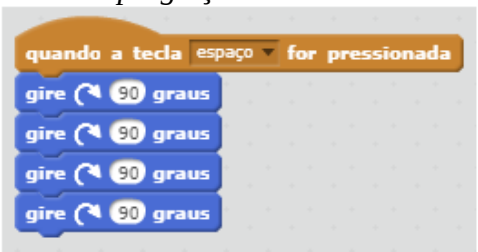
Gobo está parado!

Possível solução: O principal erro está na ausência do bloco “Quando clicar em bandeira verde” na programação do Gobo. Isso significa que não há um evento que dá início ao programa, que não vai fazer o Gobo dançar. Para resolver isso, uma alternativa simples é adicionar o bloco em questão no início da programação.

Desafio 2: *Scratch* está com Preguiça?

O programador deste algoritmo queria que o gato do *Scratch* virasse uma cambalhota ao pressionar a tecla “espaço”, mas ele simplesmente não se move.

Scratch está com preguiça?

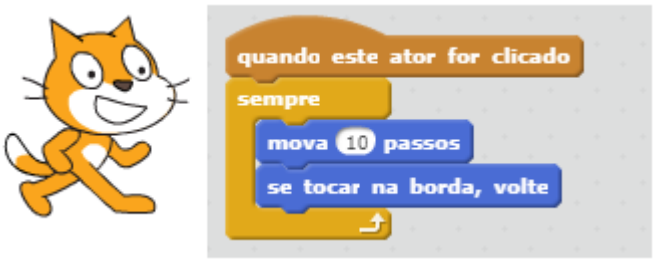


Possível solução: Cada equipe deve compreender que o programa está certo, o gato do *Scratch* está girando 360°, mas ele faz isso tão rápido que não conseguimos perceber. Isso pode ser consertado, por exemplo, adicionando intervalos de tempo entre cada “gire 90 graus”.

Desafio 3: De Cabeça para Baixo!

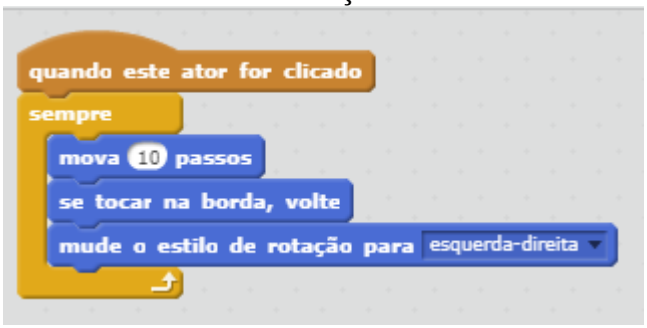
Neste projeto, o gato do *Scratch* deveria ir de um canto ao outro da tela. O problema é que ele faz isso e quando volta, está de cabeça para baixo!

De cabeça para baixo!



Possível solução: o principal problema deste programa pode ser resolvido com um bloco que caracteriza seu movimento.

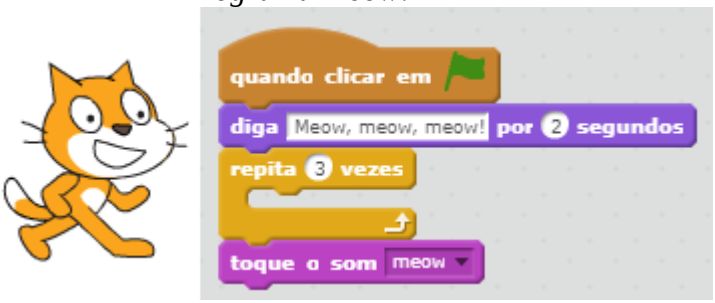
Possível solução.



Desafio 4: Meow!

A intenção do programa era que o gato do *Scratch* miasse de duas formas: com o balão de fala e através do som, que repetiria o miado 3 vezes. Algo no programa está fazendo com que ele reproduza o som somente depois do balão, e não ao mesmo tempo, e o som está sendo reproduzido somente uma vez.

Programa Meow!



Possível solução: o bloco “toque o som meow” deve estar dentro do comando “repita 3 vezes”. Além disso, deve ser trocado pelo bloco “toque o som meow até o fim”, para que o programa não execute 3 miados ao mesmo tempo. Para fazer com que o som seja executado ao mesmo tempo dos balões, existem algumas alternativas como dividir o programa em dois, ou trocar o bloco “Diga meow, meow, meow! por 2 segundos” por “Diga meow, meow, meow!”.

Encontro 4 – Jogo de Corrida!

A criação de jogos como uma ferramenta educacional e motivacional é o recurso a ser explorado neste encontro. A proposta é simples: criar um jogo de corrida onde existirão duas ou mais personagens, e, quando uma determinada tecla for pressionada, essa personagem irá se mover.

Descrição da Atividade

A turma inteira e o professor irão realizar a atividade em conjunto, com o auxílio do projetor, seguindo os preceitos do Pensamento Computacional.

É importante realizar uma lista de coisas que devem acontecer neste jogo de corrida e descrever como ele vai funcionar. Vamos aqui descrever cada parte do projeto, deixando aberto para qualquer alteração que possa vir a ser necessária. A criatividade conta muito!

Utilizando a habilidade de decomposição de problemas, vamos dividir nosso projeto em pequenas tarefas, para que depois elas formem a nossa solução.

Determinando os Competidores e seus Movimentos

As personagens do jogo de corrida são os competidores. Eles podem ser representados por personagens do próprio *Scratch* ou podem ser desenhados com a função de edição do programa.

No nosso caso, fizemos um círculo preto e um círculo verde, respectivamente objeto1 e objeto2.

Objetos 1 e 2.

Cada objeto deve se movimentar uma determinada distância a cada vez que uma tecla for pressionada. Isso pode ser feito facilmente com o comando “adicione 10 a x”.

Comando: adicione 10 a x.

Para cada competidor, será determinada uma tecla diferente do teclado. (Exemplo: espaço e seta para a direita).

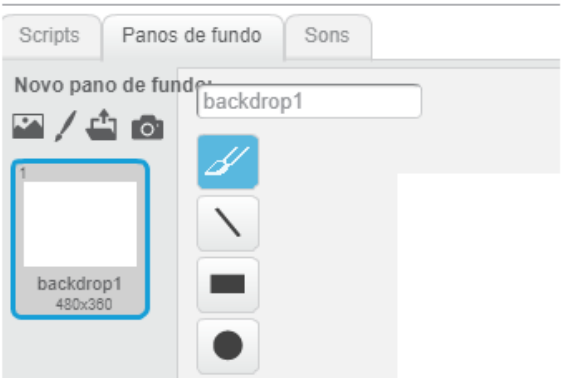
Pista de Corrida

Agora os competidores já "correm" pela tela. É necessário então, fazer a pista de corrida. Para isso, vamos acessar o palco, clicando ao lado dos atores, no local indicado (*Figura 28*).

Localização do palco.

Em seguida, selecionamos a aba "Planos de Fundo" (*Figura 29*) e pintamos nossa pista de corrida.

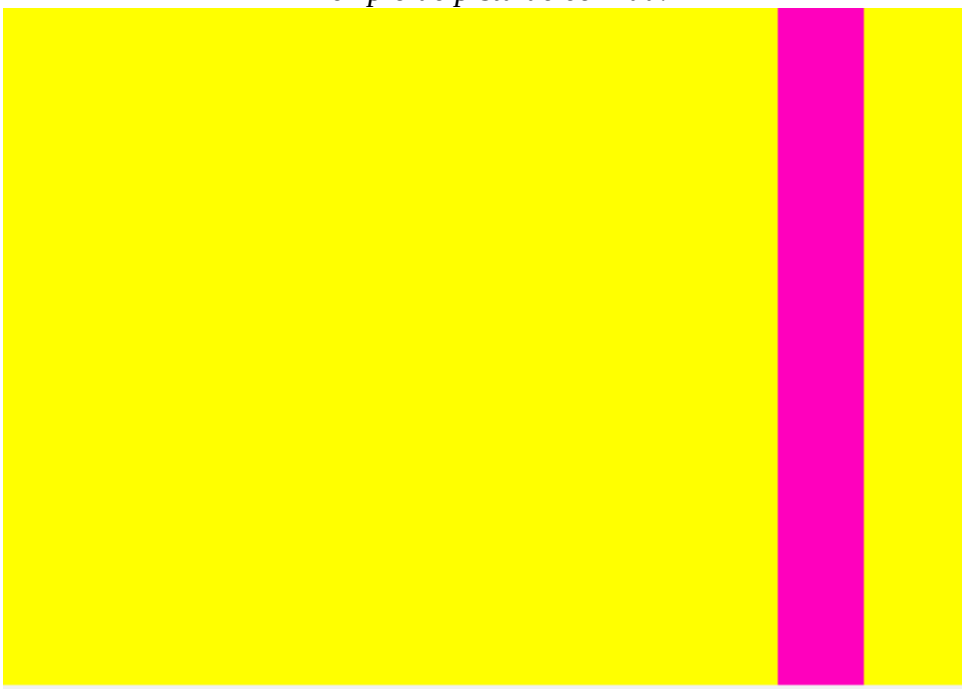
Aba de planos de fundo.



Devemos pensar como deve ser nossa pista de corrida. Discutimos com os estudantes alguns pontos importantes, como as cores, o formato e o tamanho, e, mais especificamente, como o programa vai detectar qual competidor ganhou a corrida.

Dessa maneira, nossa pista de corrida deve ter uma cor que determine a linha de chegada, pois é a partir da cor que o programa vai detectar qual objeto encostou primeiro na linha. Um exemplo simples de pista de corrida é um cenário de uma cor com um retângulo diferente.

Exemplo de pista de corrida.



Mecânica do Jogo

Definidos o cenário e os competidores, precisamos determinar como estes irão interagir um com o outro. Mais especificamente, precisamos encontrar uma maneira de determinar o vencedor.

Uma forma simples de fazer isso é utilizando o sensor de cor do *Scratch*, que determina quando o ator está encostando na cor especificada, neste caso, a cor magenta (*Figura 31*).

Utilizando o sensor de cor.

Agora a turma deve escolher como iremos identificar qual dos competidores ganhou. Algumas sugestões são: utilizar um som para cada competidor, utilizar os blocos de aparência, trocar o plano de fundo.

Customização

A turma criou o "jogo base" de corrida em conjunto, agora o desafio é que, separados em grupos de três componentes, os estudantes customizem seus jogos com outras funções. No final da aula, estes irão apresentar suas mudanças.

Algumas sugestões de desafios:

- Função de voltar os competidores para a posição inicial;

Exemplo de customização para voltar à posição inicial.

- Anúncio de que o jogador venceu através de um evento, que repercute nos scripts dos outros objetos;

Exemplo de customização com anúncio do vencedor.

- Adicionar a possibilidade de se movimentar para cima e para baixo;
- Pista de corrida com curva.

1.Apresentando o Jogo

Se a escola tiver disponibilidade e espaço, a fim de incentivar a divulgação dos resultados dos encontros dos estudantes, este jogo pode ser apresentado pelos criadores para turmas em sala de aula ou até mesmo para a comunidade escolar (pais, professores) em eventos.

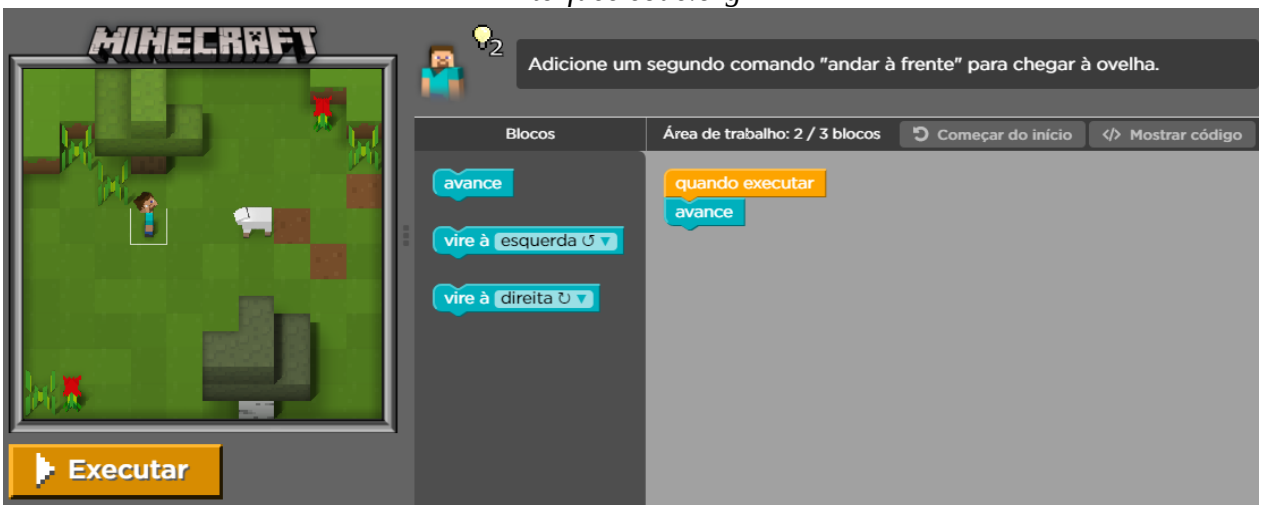
Encontro 5 - Hora do Código

A organização *code.org* é uma organização não lucrativa que se dedica a fazer a Ciência da Computação cada vez mais acessível nas escolas e ambientes de aprendizagem. Em seu *website*[26] encontramos diversas atividades, entre elas a Hora do Código.

A Hora do Código é um curso desenvolvido para que estudantes dediquem uma hora do seu dia para aprender a programação através de desafios simples que envolvem figuras de sua cultura (como, por exemplo, o jogo *Minecraft* e as personagens da animação *Frozen*).

A proposta desta atividade é que os estudantes realizem a atividade "Aventureiro de *Minecraft*"[27]. Esta atividade é guiada pelo próprio jogo em sua interface, que lança os desafios e explica como funciona o ambiente de programação.

Interface code.org



O objetivo desta atividade é mostrar a diversidade de possibilidades com programação – dentre elas a programação de jogos muito populares. A inserção desta atividade na sequência didática surgiu da necessidade de mostrar outras formas de programar e para apresentar esta plataforma aos estudantes que demonstrarem interesse em explorar outros tipos de programação em casa.

[26] https://code.org/
[27] https://studio.code.org/s/mc/stage/1/puzzle/1

Encontro 6 – Debug it! 2.0

Com objetivos similares aos do terceiro encontro, serão propostos novos desafios para que os estudantes encontrem e depurem os erros de alguns programas

Descrição da Atividade

Separados em grupos, os estudantes irão receber programas com seus respectivos erros (*bugs*). Eles devem encontrar a parte do programa que está causando este erro e consertá-la.

Será solicitado que eles registrem em um pequeno formulário qual era o erro do programa e como ele foi corrigido.

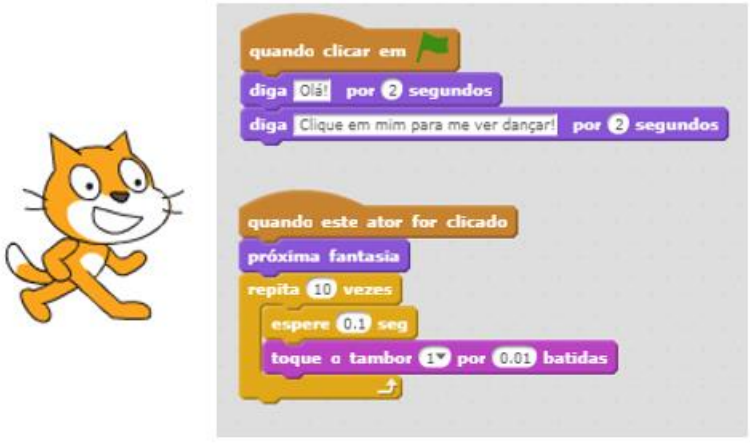
No final da aula, serão discutidas as soluções.

Alguns exemplos de programas com erros são apresentados a seguir:

Desafio 1: Dança?

Na programação, o gato do *Scratch* convida o usuário do programa para ver sua dança, mas ao clicar nele, ele apenas se move uma vez, e o som de tambores continua com ele parado. Como podemos consertar isso?

Dança?

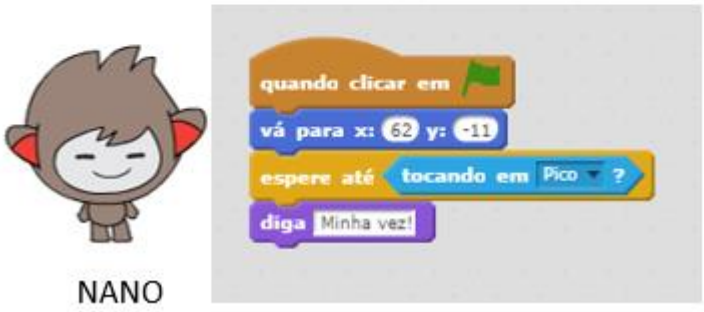


Possível solução: Podemos simplesmente colocar o bloco “próxima fantasia”, que dá o movimento ao gato, dentro do ciclo “repita 10 vezes”.

Desafio 2: Pega-Pega

Na programação, Pico e Nano estão brincando de pega-pega. Quando Pico encosta em Nano, deveria dizer “Sua vez!”, mas isso não acontece. O que deu errado?

Pega-pega.



Possível solução: Pico estará “preso” no ciclo “repita até que tocando em borda”, e assim não executará a parte do programa “se tocando em Nano, então” até que seja tarde demais (depois de tocar na borda). Basta colocar o ciclo “se tocando em Nano, então diga Sua vez!” para dentro do ciclo “repita até que tocando em borda”.

Desafio 3: Rosto Feliz

Este programa deveria desenhar um rosto feliz, mas ele está conectando um dos olhos à boca. O que podemos fazer?

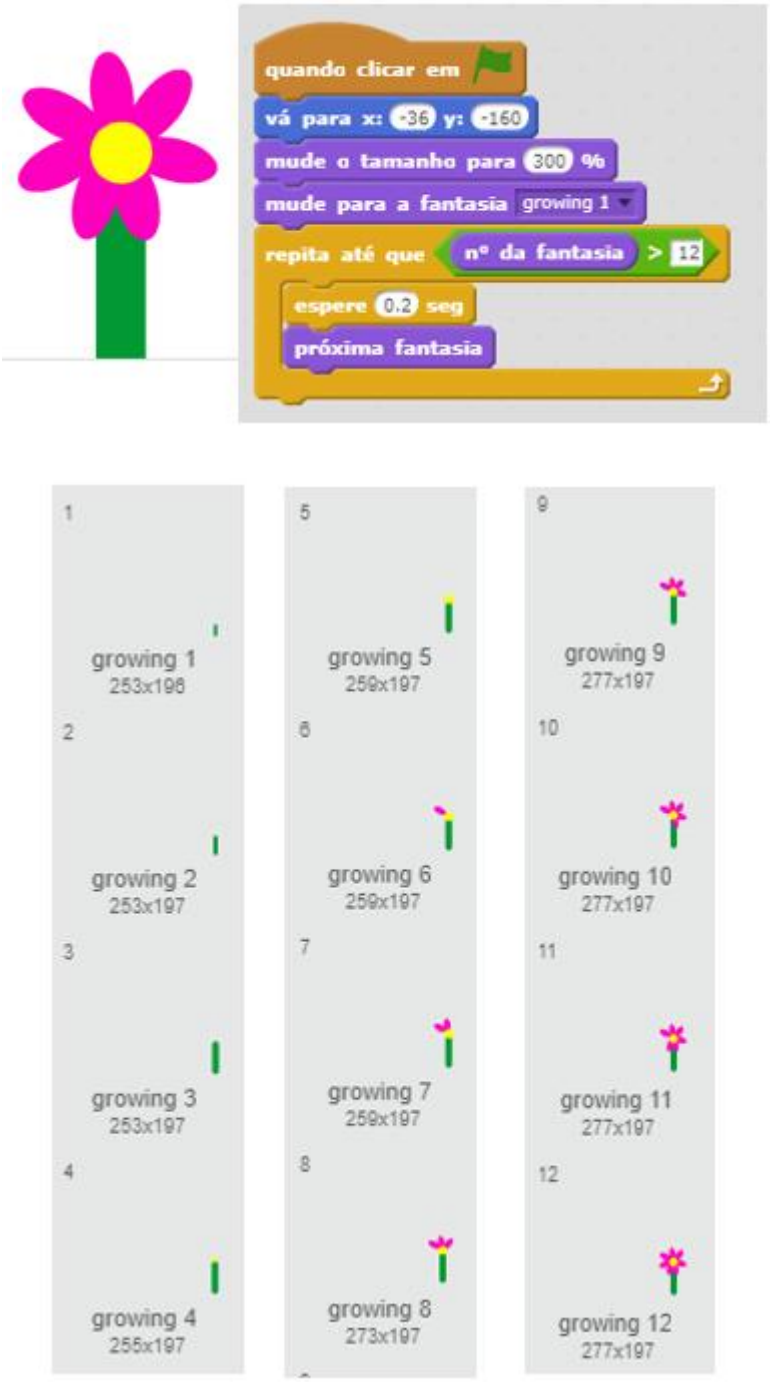
Rosto feliz.

Possível solução: é preciso levantar a caneta ao ir da posição do olho até a posição onde a boca será desenhada. Para isso, inserimos blocos de “levante a caneta” e “use a caneta” nos locais adequados.

Desafio 4: Florescendo

Este programa deveria parar quando a flor está crescida, mas ele continua repetindo a animação sem fim. Como podemos solucionar este problema?

Florescendo.



Possível solução: A troca das fantasias da personagem será executada até que o número da fantasia seja maior do que 12. Como temos somente 12 fantasias, isso nunca irá acontecer. Para corrigir o problema, trocamos o número 12 por 11.

Podemos também trocar o bloco de operação “maior que” por “igual a”. Assim, quando o número da fantasia for exatamente 12, o programa irá parar.

Desafio 5: Feliz Aniversário!

Esta animação deveria tocar o tema “parabéns pra você” e, ao acabar a música, deveria aparecer a instrução de “Clique para apagar as velas!”, mas essa instrução aparece durante a música. Como podemos consertar este erro?

Feliz Aniversário!

Possível solução: trocar o bloco “toque o som birthday” por “toque o som birthday até o fim”.

Encontro 7 - Projeto Compartilhado

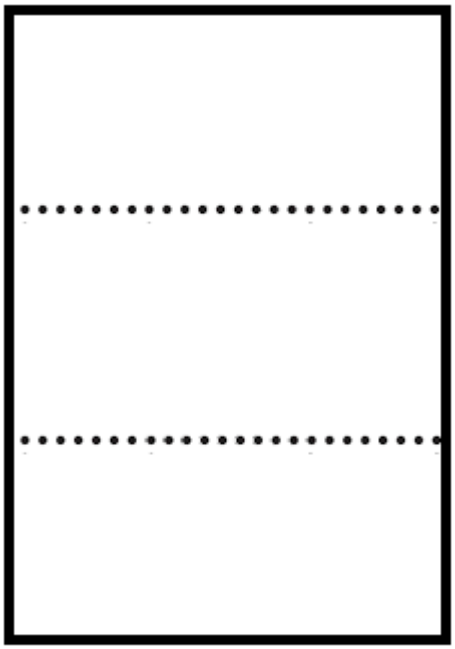
Nesta atividade, os estudantes irão criar projetos onde cada um vai adicionar um pouco do que conhece.

Descrição da Atividade

Para iniciar, os estudantes serão separados em duplas ou trios, dependendo do número de computadores. Cada grupo deve ficar com um computador para si. Será distribuído para cada equipe uma folha tamanho A4 e uma caneta hidrográfica colorida (cada caneta deve ter uma cor diferente).

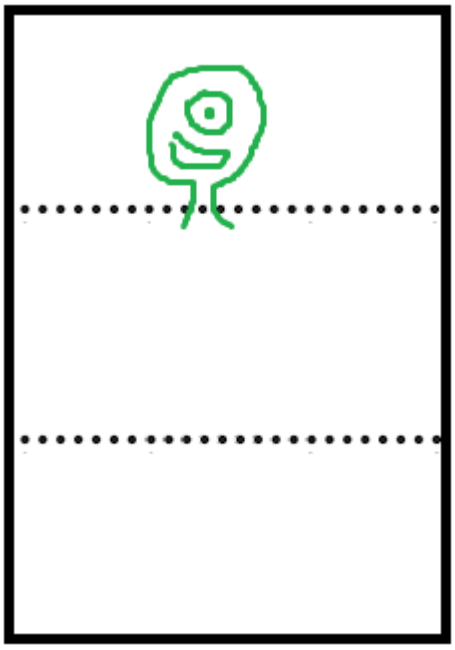
Os estudantes serão orientados a dobrar a folha em 3 partes.

Dobraduras da folha.



Nesta folha, devem usar sua imaginação e desenhar na primeira parte a cabeça de um monstro, de maneira que o pescoço fique na segunda parte, conforme exemplo.

Exemplo de desenho de monstro.



Em seguida, irão dobrar a folha, de maneira que seu desenho esteja escondido, e a turma vai fazer um rodízio de desenhos. O desenho deve ser entregue ao próximo grupo. O grupo não deve espiar o trabalho do anterior, mas deve agora fazer um corpo para este monstro a partir do pescoço que foi deixado.

De maneira semelhante, devem fazer até a cintura do monstro, e deixar as pernas para a próxima equipe finalizar.

Ao final da atividade, deve ser feita uma reflexão de como o trabalho em grupo pode ser divertido, e sobre os desafios de continuar e finalizar o trabalho de outra pessoa.

Agora os estudantes serão desafiados a fazer algo parecido, mas no *Scratch*! Os grupos irão iniciar uma animação no *Scratch*. Terão 10 minutos para iniciar o trabalho. Ao final destes 10 minutos, o professor irá anunciar que acabou o tempo e devem trocar de mesa, e continuar o trabalho do outro grupo. Repetirão o rodízio até que retornem ao seu computador (dependendo do tamanho da turma, pode ser necessário terminar a atividade antes).

No final, serão apresentados no projetor os projetos, e os grupos que iniciaram cada projeto devem descrever sua ideia original e o que acharam do resultado. Também serão estimulados a falar sobre a dificuldade de interpretar o projeto dos outros.

Encontro Final: Jogo do Labirinto

Será lançado um desafio que une os conceitos que foram trabalhados durante os encontros anteriores. Nesse sentido, ele serve como uma avaliação do desenvolvimento da turma em relação aos objetivos de aprendizagem.

Para este encontro, cada grupo deve criar, no *Scratch*, um jogo de labirinto: um jogo onde você controla uma personagem que deve solucionar o caminho de um labirinto sem encostar na parede.

A descrição desta atividade é a mais curta, pois é um desafio para os estudantes criarem o jogo “do zero”. Será um grande teste de suas percepções e estratégias.

Descrição da Atividade

Separados em duplas ou trios, os estudantes receberão o desafio: criar um jogo de labirinto:

Uma personagem controlada pelo usuário deve percorrer um labirinto. Ao chegar no final, deve ser anunciado que o jogador venceu. Se o jogador encostar na parede, deve retornar ao início.

Os estudantes deverão criar os cenários e a personagem e utilizar as cores para diferenciar a parede do labirinto. Devem ser estimulados a fazer o máximo que conseguirem e a customizar o jogo conforme acharem melhor.

Durante esta atividade o professor poderá auxiliar, mas deve cuidar para que apareçam as aprendizagens e conceitos das atividades anteriores.