UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENSINO DE MATEMÁTICA
MESTRADO PROFISSIONAL EM ENSINO DE MATEMÁTICA

BRUNO SILVEIRA CORRÊA

PROGRAMANDO COM SCRATCH NO ENSINO FUNDAMENTAL: UMA POSSIBILIDADE PARA A CONSTRUÇÃO DE CONCEITOS MATEMÁTICOS

PORTO ALEGRE
2021

BRUNO SILVEIRA CORRÊA

PROGRAMANDO COM SCRATCH NO ENSINO FUNDAMENTAL: UMA POSSIBILIDADE PARA A CONSTRUÇÃO DE CONCEITOS MATEMÁTICOS

Dissertação de mestrado elaborada como requisito parcial para a obtenção do título de Mestre em Ensino de Matemática, pelo Programa de Pós-Graduação em Ensino de Matemática da Universidade Federal do Rio Grande do Sul.

Orientadora: Dra. Márcia Rodrigues Notare Meneghetti

PORTO ALEGRE

2021

BRUNO SILVEIRA CORRÊA

PROGRAMANDO COM SCRATCH NO ENSINO FUNDAMENTAL: UMA POSSIBILIDADE PARA A CONSTRUÇÃO DE CONCEITOS MATEMÁTICOS

Aprovado em 24 de março de 2021.

BANCA EXAMINADORA

_____

Prof. Dr. Marcus Vinicius de Azevedo Basso
(IME-PPGEMAT-UFRGS)

_____

Prof. Dr. Rodrigo Sychocki da Silva
(IME-PPGEMAT-UFRGS)

_____

Prof. Dr. Marcus Vinícius Maltempi
(PPGEM-UNESP)

_____

Orientadora - Profa. Dra. Márcia Rodrigues Notare Meneghetti
(IME-PPGEMAT-UFRGS)

RESUMO

A presente pesquisa propõe-se a analisar os impactos da programação no desenvolvimento de conceitos matemáticos e do pensamento computacional em âmbito escolar. A pesquisa foi desenvolvida com oito estudantes de 8º ou 9º anos do Ensino Fundamental em uma escola de uma rede privada de Porto Alegre. Os estudantes participaram de dez encontros de uma Oficina de Programação em Scratch, em turno inverso ao da carga horária de aula, cuja proposta foi o desenvolvimento de jogos utilizando o *software* Scratch para programá-los. Os encontros foram organizados em dois momentos: cinco encontros para que os estudantes realizassem atividades iniciais, com o propósito de conhecerem e explorarem alguns recursos do *software*; e cinco encontros dedicados ao planejamento e à programação dos jogos. Durante a aplicação do experimento, os estudantes foram observados e questionados, para compreender o desenvolvimento de seus pensamentos. Para a coleta/produção dos dados, foram utilizadas gravações dos áudios das intervenções e dos diálogos entre os estudantes, captura de telas e registros dos estudantes produzidos no Scratch *online*. Trata-se de uma pesquisa qualitativa, utilizando como fundamentação teórica para análise dos dados a teoria do desenvolvimento cognitivo de Jean Piaget, para compreender o desenvolvimento e a construção de conceitos matemáticos dos estudantes e também a teoria Construcionista de Seymour Papert, que inspirou o uso de computadores e da programação na escola, buscando aproximações com a ideia de desenvolvimento do pensamento computacional de Janette Wing. Como conclusão, os resultados da pesquisa mostram que o uso da programação em blocos no Scratch oportuniza a utilização e a construção de conceitos matemáticos, a partir de relações elaboradas pelos estudantes na atividade de programação, assim como desenvolver elementos importantes do pensamento computacional na resolução de problemas que emergem da atividade de programação, como a decomposição de problemas, a generalização de situações semelhantes, a abstração de informações e o uso de algoritmos como forma de representação.

Palavras-chave: Conceitos Matemáticos; Programação; Pensamento Computacional; Scratch.

Abstract

This research proposes to analyze the impacts of programming on the development of mathematical concepts and computational thinking at school. The research was developed with 8 students from 8th or 9th grade of elementary school in a private school in Porto Alegre. The students participated in 10 meetings of a Scratch Programming Workshop, in a reverse shift to the class load, whose proposal was the development of games using the Scratch software to program them. The meetings were separated into two moments: five meetings to carry out initial with the purpose of knowing and exploring some of the software's resources; and five meetings dedicated to the planning and programming of the games. During the application of the experiment, students were observed and questioned, to understand the development of their thoughts. For the collection / production of the data, recordings of the audios of the interventions and dialogues between students were used, screen capture and student records produced in Scratch online. It is a qualitative research, using as a theoretical basis for data analysis Jean Piaget's theory of cognitive development, to understand the development and construction of students' mathematical concepts and also the Constructionist theory of Seymour Papert, which inspired the use of computers and programming at school, seeking approximations with the idea of developing Janette Wing's computational thinking. In conclusion, the research results show that the use of block programming in Scratch provides opportunities for the use and construction of mathematical concepts, based on relationships developed by students in the programming activity, as well as to develop important elements of computational thinking in problem solving that emerge from the programming activity, such as the decomposition of problems, the generalization of similar situations, the abstraction of information and the use of algorithms as a form of representation.

Keywords: Mathematical concepts; Programming; Computational Thinking; Scratch software.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

BNCC	Base Nacional Comum Curricular
CAPES	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
CINTED	Centro Interdisciplinar de Novas Tecnologias
GPTEM	Grupo de Pesquisa em Tecnologias na Educação Matemática
PPGEMAT	Programa de Pós-Graduação em Ensino de Matemática
PPGIE	Programa de Pós-Graduação em Informática na Educação
RENOTE	Revista Novas Tecnologias na Educação
UFRGS	Universidade Federal do Rio Grande do Sul

SUMÁRIO

1 INTRODUÇÃO

Minha trajetória com a exploração e utilização de *software*s de Matemática iniciou-se no primeiro semestre de graduação em Licenciatura em Matemática na Universidade Federal do Rio Grande do Sul (UFRGS), em 2010, na disciplina então intitulada “Computador na Matemática Elementar I”. Nessa disciplina, tive contato com o *software* SuperLogo, no qual pude perceber o potencial desse *software* para colaborar no processo de aprendizagem de Matemática, utilizando-se da atividade de programação para auxiliar na construção de conceitos de geometria, trigonometria, álgebra, entre outros. Seymor Papert, um dos criadores da linguagem de programação LOGO, entendia ser possível utilizar uma linguagem de computação para pessoas comuns, especialmente crianças. Dessa forma, visava possibilitar experiências que ele mesmo já havia vivenciado ao iniciar seu contato com computadores.

> Foi nessa situação que eu pensei sobre computadores e crianças. Eu estava brincando como uma criança e experimentando uma vulcânica explosão de criatividade. Por que o computador não poderia proporcionar a uma criança o mesmo tipo de experiência? Por que uma criança não poderia brincar como eu? O que teríamos que fazer para tornar isso possível? (PAPERT, 1994, p.36)

Tive essa mesma inspiração em meus primeiros contatos com a programação no SuperLogo. Minhas primeiras ações foram a de criar polígonos regulares e, percebendo a necessidade do conhecimento matemático necessário para generalizar uma programação que construísse qualquer polígono regular apenas informando o número de lados, também me fez pensar na possibilidade de desenvolver e/ou consolidar os conceitos envolvidos naquela programação na prática, utilizando a Matemática para resolver um problema.

Depois de graduado e com alguns anos lecionando, aventurei-me no uso do *software* Scratch por vontade própria, tendo novamente aquela sensação de inspiração que tive no início da graduação, um dos fatores que motivou o tema dessa pesquisa, aliado à curiosidade de observar o uso de conceitos matemáticos aplicados aos desafios da programação no Scratch. Nesse sentido, entendemos que a programação no *software* Scratch como uma ferramenta ativa e criativa para despertar o interesse pela aprendizagem desses conceitos, proporcionando um

espaço para os estudantes participarem de forma efetiva da construção de seus conhecimentos, como salienta Maltempi (2008, p. 60):

> [...] as tecnologias representam uma oportunidade para mudanças na educação, em especial da prática docente, da centrada no professor (ou tradicional) para a centrada nos alunos, de forma a atender os anseios e demandas de conhecimento destes. De fato, um grande mérito das tecnologias é o de colocar diversos pesquisadores e educadores em um movimento de reflexão sobre a educação frente às modificações pelas quais a sociedade passa em decorrência da crescente inserção das tecnologias no dia-a-dia das pessoas.

A partir programação no Scratch, pode-se considerar que o estudante "ensina" o ator a movimentar-se no palco, atribuindo as ações desejadas por meio da programação em blocos, sem que ela esteja no centro da ação do estudante. Para isso, é necessário que o estudante reflita sobre os processos que deverão ser executados pelo computador execute, tendo em vista que esses movimentos devem ser ensinados na linguagem do *software*, o que atribui significados aos comandos. Esse processo é descrito por Valente no prefácio de Papert (1988, p. 9), afirmando que, no ambiente LOGO, "o aprendizado acontece através de a criança inteligente 'ensinar' o computador burro, ao invés de o computador inteligente ensinar a criança burra".

A partir das ideias discutidas acima, entendemos que o *software* Scratch possa contribuir para o estudo de conceitos matemáticos. Para Azevedo et al. (2018), é necessário encontrar meios para explorar a matemática sem desconsiderar o currículo atual, potencializando a aprendizagem crítica e criativa dos estudantes. Nesse sentido, para que o conteúdo curricular de matemática não seja estável, "[...] é preciso reservar um espaço na programação do currículo para a curiosidade, o questionamento, a descoberta, a imprevisibilidade, a criatividade e a inovação." (AZEVEDO et al. 2018, p. 953). Dessa forma, vemos potencialidades no *software* Scratch, que pode ser utilizado para que a Matemática seja trabalhada de forma diferenciada e não direcionada. Sobre a organização do currículo, Azevedo et al. (2018, p. 954) ainda salientam que o currículo de Matemática deve:

> [...] oportunizar não só o trabalho com o conteúdo conceitual matemático, mas também a mobilização das características do fazer matematicamente, como: abstrair, organizar, induzir, generalizar, interpretar, etc. É perceber o aprendiz como um construtor de conhecimento, e que seja capaz de desenvolver o seu raciocínio matemático e a sua capacidade para criar, elaborar e resolver diferentes tipos de problemas; fazer observações sistemáticas de aspectos quantitativos e qualitativos, ler e interpretar diferentes representações matemáticas; saber selecionar e organizar dados; produzir informações relevantes, para interpretá-las e avaliá-las.

Vimos no *software* Scratch, então, uma possibilidade para proporcionar o desenvolvimento de ações caraterísticas da atividade matemática, utilizando-se dos recursos tecnológicos e da criatividade dos estudantes para desencadear a construção de conceitos e o desenvolvimento de habilidades matemáticas.

Nesse contexto, nesta pesquisa pretende-se responder à seguinte pergunta: *Quais as contribuições do uso da programação no* software *Scratch no desenvolvimento de conceitos matemáticos e de elementos do pensamento computacional com estudantes de Ensino Fundamental?*

Com o propósito de auxiliar na resposta a essa questão traçamos algumas questões adicionais, as quais buscamos responder no capítulo das considerações finais:

- Como os estudantes desenvolvem seu pensamento matemático a partir da ação de programar?

- Quais conceitos matemáticos podem ser construídos e utilizados durante a programação no *software* Scratch?

- O pensamento computacional desenvolvido durante o processo de programação pode contribuir para o desenvolvimento de habilidades matemáticas?

Foram planejados como objetivos desse trabalho:

- Analisar as contribuições do *software* Scratch na aprendizagem de conceitos matemáticos no ensino fundamental.

- Observar no processo de programação dos estudantes como conceitos matemáticos são utilizados ou desenvolvidos.

- Desenvolver questionamentos, inspirados no método clínico, que possam auxiliar o pesquisador a analisar formas de pensar dos estudantes durante o processo de programação.

Temos como objetivo procedimental:

- Elaborar, justificar e construir uma estrutura de procedimentos pedagógicos que possibilite a investigação dos processos de ação e compreensão dos participantes da pesquisa ao utilizar a programação em blocos do Scratch.

Essa dissertação está estruturada em seis capítulos. O capítulo 2 traz o referencial teórico que deu suporte para essa pesquisa. Neste capítulo, é abordado o potencial uso de tecnologias digitais no contexto escolar, em particular os computadores como ferramentas que podem auxiliar na aprendizagem matemática

dos estudantes. O pensamento computacional também é abordado nesse capítulo. A partir das ideias de Janette Wing, discutimos o desenvolvimento do pensamento computacional e suas relações com o desenvolvimento de habilidades matemáticas na atividade de programação em blocos. A teoria do desenvolvimento cognitivo de Jean Piaget foi escolhida para dar suporte à análise dos dados por entender que o desenvolvimento da compreensão se inicia pela ação, partindo para uma conceituação, o que nos possibilita analisar o processo de construção dos conceitos matemáticos a partir do caminho percorrido do fazer ao compreender. Por fim, são apresentados alguns trabalhos correlatos, que inspiraram esse estudo por utilizarem o *software* Scratch ou o pensamento computacional como tema de estudo vinculado à aprendizagem matemática.

O capítulo 3 apresenta o *software* Scratch em seu contexto de criação e sua interface, oferecendo um panorama geral dos principais comandos que foram utilizados durante as atividades propostas na oficina elaborada para essa pesquisa.

O capítulo 4 apresenta os procedimentos metodológicos da pesquisa, trazendo o método clínico de Jean Piaget como inspiração para a produção dos dados. Também são apresentados nesse capítulo os materiais utilizados para a produção dos dados, o cenário no qual foi desenvolvida a aplicação das atividades e os participantes da pesquisa.

No capítulo 5, de descrição e análise dos dados, estão os dados produzidos, organizados por episódios com suas respectivas análises. No capítulo 6, apresentam-se as considerações finais da pesquisa realizada.

Esperamos que o leitor tenha uma boa e produtiva leitura.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são apresentadas as ideias teóricas que dão sustentação ao nosso trabalho. Iniciaremos pelas ideias de Seymour Papert e José Armando Valente, realçando a importância das tecnologias digitais em sala de aula. Em seguida, são apresentadas as ideias de Mitchek Resnick e Jeannette Wing, discutindo o papel do pensamento computacional na educação e as possíveis colaborações do *software* Scratch para o seu desenvolvimento. Por fim, o capítulo aborda a teoria de Jean Piaget do fazer e compreender, discutindo elementos teóricos que sustentarão as análises sobre os processos de aprendizagem a partir da atividade de programação em blocos.

2.1 Computador na sala de aula

De acordo com Basso e Gravina (2012), o computador vem tendo papel cada vez mais fundamental em nossas vidas. Porém, as facilidades e possibilidades proporcionadas pelo computador e pela internet fazem-nos refletir sobre a necessidade de incorporar as tecnologias na rotina escolar, pois “elas também influem nas formas pensar, de aprender, de produzir” (BASSO, GRAVINA, 2012, p. 12). Essa necessidade vem sendo cada vez mais imposta, sendo claramente evidenciada com o momento de pandemia ocasionado pela COVID-19, período em que o uso do computador e da internet foi de grande importância para o encaminhamento do ensino remoto, adotado por muitas escolas. Nesse contexto, percebe-se a importância da constante adaptação e renovação nas formas de se ensinar, assim como afirma Kenski (2012, p. 30), ao dizer que “as velozes transformações tecnológicas da atualidade impõem novos ritmos e dimensões à tarefa de ensinar e aprender. É preciso estar em permanente estado de aprendizagem e de adaptação ao novo”.

Uma das alternativas para implementar o uso das novas tecnologias no ambiente de ensino e aprendizagem nas escolas é por meio da programação, iniciada e defendida por Seymour Papert e a linguagem LOGO. Segundo Valente (1993), a linguagem LOGO, desenvolvida no Massachusetts Institute of Technology (MIT), Boston E.U.A., pelo professor Seymour Papert e colaboradores, tem como objetivo criar uma linguagem para nos comunicarmos com o computador, tendo

duas raízes para explorar seus aspectos de aprendizagem: a computacional, que tem como característica a exploração de atividades espaciais, fácil terminologia e capacidade de criar novos procedimentos, e a pedagógica, que está fundamentada no construtivismo piagetiano, na qual a criança aprende diversos conceitos matemáticos a partir de experiências do cotidiano, interagindo com objetos do ambiente em que ela vive. Sendo assim, o construcionismo de Papert defende a construção do conhecimento mediante a interação entre estudante e ferramenta, nesse caso, o computador.

Em relação ao uso da tecnologia na educação e sobre os impactos do computador nas relações entre humano e máquina, Pea (1987, p. 95, tradução do autor[1]) já afirmava que

> [...] é consistente com questões baseadas em uma imagem bidirecional que outros educadores e pesquisadores de matemática [...] estão fazendo, tais como: Quais são as *novas* coisas que você pode fazer com tecnologias que você não poderia fazer antes ou que não eram *práticas* para fazer? Quando você começa a usar a tecnologia, que coisas totalmente novas você percebe que podem ser *possíveis*? Por "imagem bidirecional" quero dizer que os computadores não afetam apenas as pessoas, mas as pessoas afetam os computadores. Isso é verdade em dois sentidos. Em certo sentido, todos nós afetamos os computadores e as oportunidades de aprendizado que eles oferecem aos alunos na educação pela forma como os *interpretamos* e pelo que definimos como práticas apropriadas com eles; à medida que essas interpretações mudam com o tempo, alteramos os efeitos que os computadores podem ter mudado que fazemos com eles.

Nessa concepção, percebemos que a defesa do uso dos computadores já vem sendo debatida há algumas décadas, mas muitas das reflexões feitas são pertinentes até os dias de hoje. Valente (1993) afirma que o uso dos computadores na educação tende para diferentes perspectivas a partir de suas modalidades de uso, deixando de ser uma “máquina de ensinar” para ser vista como uma mídia educacional que promove o desenvolvimento do pensar do indivíduo: “o computador passa a ser uma ferramenta de complementação, de aperfeiçoamento e de possível mudança na qualidade de ensino” (VALENTE, 1993a, p. 6). Essa mudança deve-se à nossa condição de vida, que é dominada pela informação e pelos processos rápidos e imperceptíveis que estão ocorrendo todos os dias.

---

[1] Citação original: [...] is consistent with questions based in a two-directional image that other mathematics educators and researchers […] are posing, such as: What are the *new* things you can do with technologies that you could not do before or that weren’t *pratical* to do? Once you begin to use the technology, what totally new things do you realize might be *possible* to do? By “two-directional

Valente (1993) já defendia a utilização do LOGO como um espaço que proporciona o processo de aprender com a máquina. No LOGO, o erro conceitual é considerado, ou seja, “o erro deixa de ser uma arma de punição e passa a ser uma situação que nos leva a entender melhor nossas ações e conceitualizações” (VALENTE, 1993, p. 23). Caso o programa não produza o que era esperado, significa que o conceito está errado, o que oportuniza uma análise do erro, possibilitando uma oportunidade de entender o conceito envolvido no problema proposto. Apresentamos as ideias defendidas por Valente como uma perspectiva ainda importante na atualidade, onde o uso dos computadores e da programação ainda é discutido no contexto escolar.

A versão final da Base Nacional Comum Curricular (BNCC), aprovada e homologada em dezembro de 2018, prevê a linguagem de programação como uma das habilidades na área de Matemática e suas Tecnologias: “(EM13MAT405) Utilizar conceitos de uma linguagem de programação na implementação de algoritmos escritos em linguagem coerente e/ou matemática” (BRASIL, 2018, p. 539). A BNCC ainda apresenta em seus “itinerários formativos” (BRASIL, 2018, p. 477) estratégias de organização curricular visando à formação técnica e profissional dos estudantes do Ensino Médio.

> II - matemática e suas tecnologias: aprofundamento de conhecimentos estruturantes para aplicação de diferentes conceitos matemáticos em contextos sociais e de trabalho, estruturando arranjos curriculares que permitam estudos em resolução de problemas e análises complexas, funcionais e não-lineares, análise de dados estatísticos e probabilidade, geometria e topologia, robótica, automação, inteligência artificial, programação, jogos digitais, sistemas dinâmicos, dentre outros, considerando o contexto local e as possibilidades de oferta pelos sistemas de ensino (BRASIL, 2018, p. 477).

Ou seja, essa concepção de aprendizagem em ambiente de programação permanece atual e relevante. Nas pesquisas em Educação Matemática, o *software* Scratch vem tendo cada vez mais reconhecimento, em especial no processo de desenvolvimento de jogos. Azevedo et al. (2018, p. 952) salientam a relevância atual da produção de jogos digitais nas aulas de Matemática, observando que

> O trabalho com jogos digitais no contexto escolar pressupõe uma nova organização de ensino e exige uma nova postura tanto do professor quanto do aluno para a construção de conhecimento. Isso porque, o jogo digital, além de ter um caráter dinâmico e educacional, é uma atividade que se instaura no contexto de ludicidade, regras, confrontos, mas que se sustenta no objetivo comum de aprendizagem, em particular, no campo de matemática.

A partir dessa perspectiva, o uso das tecnologias na Educação Matemática será pensado como uma ferramenta na qual o estudante possa utilizar para criar e pensar Matemática de forma que a tecnologia possa contribuir para o desenvolvimento da aprendizagem. Assim, conforme Pea (1987), as tecnologias levam ao desenvolvimento do pensamento matemático para a resolução de problemas e não para ser utilizada como um fim em si mesmo.

2.2. O *Software* Scratch e o Pensamento Computacional

Segundo Resnick (2007), a rápida mudança no mundo atual requer que as pessoas encontrem soluções criativas para problemas inesperados. O quanto sabemos e o que sabemos já não é o único ponto a ser avaliado para o sucesso, mas sim a capacidade de pensar e agir de forma criativa. Essa perspectiva em que vivemos é denominada por Resnick como “Sociedade Criativa”.

O que vemos na sala de aula, hoje, nos faz pensar se a escola ajuda os estudantes a se desenvolverem criativamente, ou se, mesmo os estudantes com melhor rendimento escolar, aprendem a resolver tipos específicos de problemas, com pouca capacidade de adaptar o seu conhecimento a uma nova situação, habilidade importante em um mundo que enfrenta rápidas e inesperadas mudanças. Nesse sentido, para Resnick (2007) as tecnologias digitais desempenham um papel importante para a sociedade, já que, além de acelerarem as mudanças, acentuando a necessidade de pensamento em diversos aspectos da vida, elas também dispõem de um potencial inerente para colaborar no desenvolvimento de pensadores criativos, sendo uma delas a programação.

Nesse processo de programação, o *software* Scratch é visto como um ambiente na qual “os alunos aprendem conceitos matemáticos em um contexto significativo e motivador” (RESNICK, 2007, p. 5, tradução do autor[2]). O Scratch é um ambiente de programação que permite que os estudantes criem jogos, animações e simulações, podendo compartilhar suas criações com outros programadores de todo o mundo. Nesse contexto, vemos o *software* Scratch como uma estrutura de mídia

[2] Citação original: “[...]students learn important mathematical concepts in a meaningful and motivating context”(RESNICK, 2007, p. 5).

interativa que favorece, também, o pensamento computacional nos jovens, definido pela BNCC (BRASIL, 218, p. 474) como uma habilidade que “envolve as capacidades de compreender, analisar, definir, modelar, resolver, comparar e automatizar problemas e suas soluções, de forma metódica e sistemática, por meio do desenvolvimento de algoritmos”. Para Brennan e Resnick (2012), o pensamento computacional é um termo que tem recebido uma atenção considerável nos últimos anos, porém, ainda não se tem clareza do que o pensamento computacional engloba como também as estratégias para avaliar o seu desenvolvimento. Porém, nos termos da BNCC, o pensamento computacional é visto como uma habilidade a ser desenvolvida a partir de processos de aprendizagem.

> Os processos matemáticos de resolução de problemas, de investigação, de desenvolvimento de projetos e da modelagem podem ser citados como formas privilegiadas da atividade matemática, motivo pelo qual são, ao mesmo tempo, objeto e estratégia para a aprendizagem ao longo de todo o Ensino Fundamental. Esses processos de aprendizagem são potencialmente ricos para o desenvolvimento de competências fundamentais para o letramento matemático (raciocínio, representação, comunicação e argumentação) e para o desenvolvimento do pensamento computacional (BRASIL, 2018, p. 266).

Cuny, Snyder e Wing (2010) explicam que o pensamento computacional é um termo no qual está ligado ao trabalho construcionista de Seymour Papert, sendo um termo apresentado e popularizado em um artigo seminal de Janette Wing, denominado “*Computational Thinking*”. Apesar de o termo “pensamento computacional” ter sido popularizado por Wing, entendemos ser importante salientar que as ideias sobre o pensamento computacional já existiam e foram escritas no artigo “*Twenty things to do with a computer*”, em 1971 por Seymour Papert e Cynthia Solomon (PAPERT e SOLOMON, 1971), porém não foram vinculados a este termo. Contudo, em 1980, Papert utilizou o termo “Pensamento Computacional” em seu livro “*Mindstorms: Children, Computers, And Powerful Ideas*”:

> Na maioria dos casos, embora os experimentos tenham sido interessantes e emocionantes, eles não conseguiram fazer isso porque eles eram muito primitivos. Seus computadores simplesmente não tinham o poder necessário para os tipos de atividades mais envolventes e compartilháveis. Suas visões de como integrar o pensamento computacional na vida cotidiana foi insuficientemente desenvolvido. Mas haverá mais tentativas e mais e mais. E eventualmente, em algum lugar, todas as peças se juntarão e elas serão "capturadas". Pode-se ter certeza disso porque tais tentativas não serão experimentos isolados operados por pesquisadores que podem ficar sem fundos ou simplesmente ficarem desiludidos e sair. Serão manifestações de um movimento social de pessoas interessadas em

> computação pessoal, interessadas em seus próprios filhos e interessadas em educação. (PAPERT, 1980, p. 182, tradução do autor[3]).

Wing (2006, p. 33, tradução do autor[4]) explica que "o pensamento computacional envolve a resolução de problemas, projetar sistemas e compreender o comportamento, baseando-se nos conceitos fundamentais da ciência da computação". A autora também explica a importância do pensamento computacional no decorrer dos avanços tecnológicos e no desenvolvimento da sociedade como um todo. Para ela, o pensamento computacional é uma habilidade fundamental não apenas para cientistas, mas deve também ser acrescentada às capacidades analíticas de cada criança, assim como a leitura, a escrita e a aritmética.

Sendo assim, entendemos o *software* Scratch como uma ferramenta de programação que fornece um ambiente favorável para contribuir para o desenvolvimento do pensamento computacional a quem o utiliza. Pensamento esse convergente ao de Brennan e Resnick (2012):

> A frase pensamento computacional nos ajuda a pensar em aprender com o Scratch e, por sua vez, acreditamos que a programação com o Scratch fornece um contexto e um conjunto de oportunidades, contribuindo para as conversas ativas sobre o pensamento computacional. Estamos interessados nas maneiras nas quais atividades de aprendizagem baseadas em design - em particular, programação de mídia interativa - apoiam o desenvolvimento do pensamento computacional em jovens. Parte desse interesse é alimentado pela crescente disponibilidade de ferramentas que permitem aos jovens projetar suas próprias mídias interativas (p. 2, tradução do autor[5]).

Sobre o pensamento computacional, Wing (2006) estabelece algumas características, como:

---

[3] Citação original: "In most cases, although the experiments have been interesting and exciting, they have failed to make it because they were too primitive. Their computers simply did not have the power needed for the most engaging and shareable kinds of activities. Their visions of how to integrate computational thinking into everyday life was insufficiently developed. But there will be more tries, and more and more. And eventually, somewhere, all the pieces will come together and it will "catch." One can be confident of this because such attempts will not be isolated experiments operated by researchers who may run out of funds or simply become disillusioned and quit. They will be manifestations of a social movement of people interested in personal computation, interested in their own children, and interested in education." (PAPERT, 1980, p. 182).

[4] Citação original: "Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science." (WING, 2006, p. 33)

[5] Citação original: "The phrase computational thinking helps us think about learning with Scratch, and, in turn, we believe that programming with Scratch provides a context and set of opportunities for contributing to the active conversations about computational thinking. We are interested in the ways that design-based learning activities – in particular, programming interactive media – support the development of computational thinking in young people. Part of this interest is fuelled by the growing availability of tools that enable young people to design their own interactive media." (BRENNAN, RESNICK, 2012, p. 2)

1 – A informática não é apenas programação de computadores. Para um pensamento computacional é necessário pensar em múltiplos níveis de abstração.

2 – O pensamento computacional deve ser uma forma humana de resolver problemas e não fazer humanos pensarem como computadores. Wing (2006) define computadores como “maçantes e chatos” e humanos como “inteligentes e imaginativos”, os humanos é que tornam os computadores empolgantes, utilizando a inteligência para resolver problemas que não eram possíveis antes dos computadores, construindo sistemas com funcionalidades limitadas apenas pela nossa imaginação.

3 – Complementa e combina o pensamento matemático com o pensamento da engenharia. “A Ciência da Computação baseia-se no conhecimento matemático, já que todas as ciências em suas bases formais repousam na Matemática” (WING, 2006, p. 35, tradução do autor[6]). A Ciência da Computação também se baseia inerentemente no pensamento de engenharia, já que há a construção de sistemas que interagem com o mundo real. Sendo assim, há a necessidade de se pensar computacionalmente e não apenas matematicamente, podendo assim projetar sistemas além do mundo físico na construção de mundos virtuais.

4 – O pensamento computacional é utilizado para abordar e resolver problemas, gerenciar nossas vidas diárias e na interação com outras pessoas.

Essas características citadas por Wing enfatizam a importância do pensamento computacional como uma habilidade importante para a sociedade lidar com os problemas do futuro. Fazendo uma comparação com a computação: “a computação foi um sonho de ontem que se tornou realidade hoje, pensamento computacional é a realidade de amanhã” (WING, 2006, p. 34, tradução do autor[7]).

Pensamento computacional pode ser confundido de forma errônea como a capacidade de manipular aplicativos e dispositivos eletrônicos. É percebido que os jovens de hoje geralmente são tratados pelas gerações anteriores como a “geração digital”, pela forma como interagem com os dispositivos eletrônicos e pela facilidade na qual operam aplicativos e mídias digitais. Contudo, observa-se que, muitas

---

[6] Citação original: “Computer science inherently draws on mathematical thinking, given that, like all sciences, its formal foundations rest on mathematics.” (WING, 2006, p. 35)

[7] Citação original: “Ubiquitous computing was yesterday’s dream that became today’s reality; computational thinking is tomorrow’s reality.” (WING, 2006, p. 34)

vezes, essa utilização ocorre de forma mecânica e ineficaz para o desenvolvimento de conhecimento que possa colaborar para a resolução de problemas.

Porém, pensamento computacional, como descrito por Wing (2006), está diretamente ligado à forma como se resolvem problemas, ou seja, o pensamento computacional se propõe não apenas a criar bons usuários de tecnologias, mas sim, criadores e desenvolvedores, evidenciando o papel do pensamento computacional como fator importante na aprendizagem de crianças e adolescentes em seu desenvolvimento educacional. “Em tempos atuais, o desafio que se impõe aos usuários é o de criar os seus próprios sistemas (por exemplo programas, jogos) ou modificar os existentes de acordo com as suas necessidades” (BRACKMANN et al., 2016, p. 197).

Grover e Pea (2013, p. 2) apresentam elementos para a compreensão do pensamento computacional, que podem apoiar o desenvolvimento de sua aprendizagem e avaliação. São eles:

- Abstração e reconhecimento de padrões;
- Processamento sistemático da informação;
- Sistema de símbolos e representação;
- Noções de controle de fluxo em algoritmos;
- Decomposição de problemas estruturados;
- Pensamento iterativo, recursivo e paralelo;
- Lógica condicional;
- Eficiência e restrição de desempenho;
- Depuração e detecção de erro sistemático.

Brackmann (2017) apresenta a informação de que, desde 2001, o Reino Unido lidera a implementação da computação no ensino fundamental, incluindo a disciplina de Computação como obrigatória em todos os quatro níveis em seu Currículo Nacional, mesclando os elementos citados por Grover e Pea (2013) para construir os “Quatro Pilares do Pensamento Computacional”. Esses pilares são: Decomposição, Reconhecimento de padrões, Abstração e Algoritmos. Na seção seguinte, apresentamos esses pilares, articulando esses conceitos com algumas possibilidades do *software* Scratch.

2.3 Os quatro pilares do pensamento computacional

Considerando o desenvolvimento do pensamento computacional em estudantes de nível básico, que leva em consideração recursos computacionais aplicados a resoluções de problemas, os quatro pilares - Decomposição, Reconhecimento de Padrões, Abstração e Algoritmos - são vistos como fundamentais para que esse desenvolvimento ocorra. Segundo Brackmann (2017), o pensamento computacional, no contexto da resolução de problemas:

> [...] envolve identificar um problema complexo e quebrá-lo em pedaços menores e mais fáceis de gerenciar (DECOMPOSIÇÃO). Cada um desses problemas menores pode ser analisado individualmente com maior profundidade, identificando problemas parecidos que já foram solucionados anteriormente (RECONHECIMENTO DE PADRÕES), focando apenas nos detalhes que são importantes, enquanto informações irrelevantes são ignoradas (ABSTRAÇÃO). Por último, passos ou regras simples podem ser criados para resolver cada um dos subproblemas encontrados (ALGORITMOS) (p.33).

Dessa forma, realçamos a importância do desenvolvimento de cada um desses pilares, mesmo que de forma inconsciente pelos estudantes, no processo de resolução de um problema central a qual o estudante se depara como, por exemplo, a programação de um jogo utilizando os blocos do Scratch. Na sequência, apresentamos cada um dos quatro pilares com possíveis situações do *software* Scratch que os exemplificam.

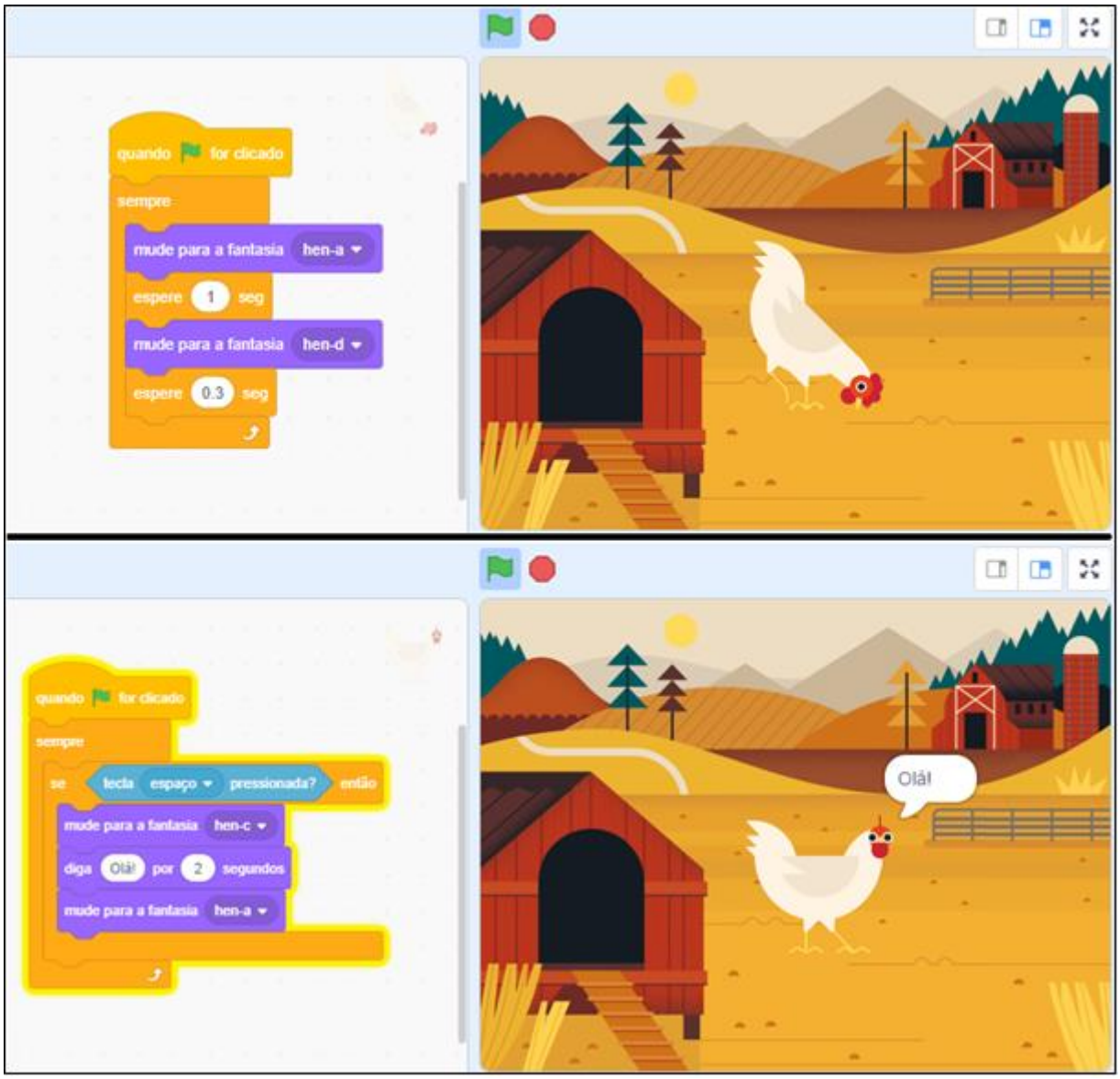
2.3.1 Decomposição

A decomposição de um problema sugere dividi-lo em partes menores para que sejam resolvidas de forma separada, facilitando o processo de resolução. Para Wing (2006), decomposição de um problema sugere atacar uma tarefa considerada complexa, separando-as em tarefas mais simples, de modo a torná-la mais acessível. Desse modo, cada fragmento do problema poderá ser examinado e trabalhado de forma individual, podendo assim modificar um sistema complexo sem a necessidade de influenciar todo o sistema, minimizando o surgimento de possíveis erros.

Por exemplo, ao realizar a programação de um jogo ou de uma animação no *software* Scratch, é possível programar inúmeras ações para um mesmo ator. Por exemplo, se quisermos programar uma animação constante para um ator e realizar

determinada ação dependente do acionamento de uma tecla, é possível decompor o problema em partes. Na Figura 1, mostra-se uma programação no *software* Scratch que satisfaz o problema mencionado, realizando as ações que podem ser observadas de forma independente: a galinha ciscando (animação constante) e a galinha cumprimentando ao pressionar uma tecla definida (animação dependente). Percebemos que, na decomposição, cada ação do personagem pode ser descrita por um algoritmo, mostrando que cada ação pode ser representada como um subproblema do todo.

Figura 1 - Exemplo de Decomposição do problema no Scratch



Fonte: Produção do autor.

Aplicada a decomposição na animação, a programação torna-se mais simples por estar fragmentada (BRACKMANN, 2017), tendo uma programação separada para cada ação do ator. A decomposição dos problemas em algoritmos distintos também permite que as correções e as modificações de um algoritmo sejam feitas de forma independente, não modificando ou prejudicando as demais.

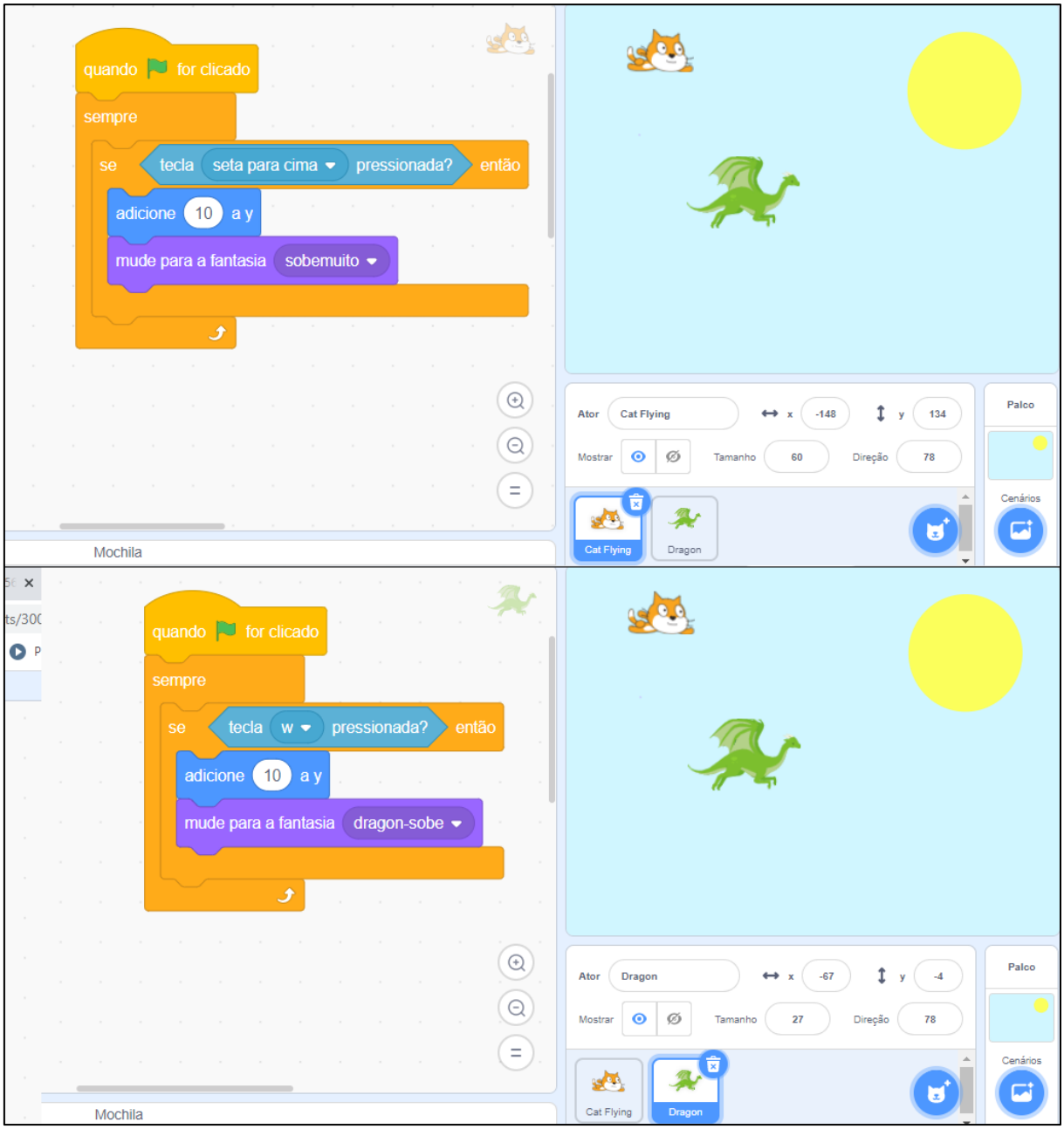
Interessante observarmos e relacionarmos as aproximações da decomposição definida pelo pensamento computacional com a resolução de problemas matemáticos. Ao nos depararmos com problemas de matemática, temos como uma das estratégias para resolvê-los a separação em problemas menores, procurando solucioná-los em etapas, o que se assemelha com o processo de decomposição na programação.

2.3.2 Reconhecimento de padrões

A partir da decomposição de um problema, surgem novos subproblemas que podem ser relacionados entre si a partir do reconhecimento de padrões comuns de comportamento. No reconhecimento de padrões, problemas podem ser solucionados mais facilmente por compartilharem as mesmas características. Esses padrões podem ser relacionados a uma generalização, ou seja, a identificação de um procedimento que seja capaz de solucionar todos os problemas com o mesmo padrão.

Para Brackmann (2017, p.36), “reconhecimento de padrões é uma forma de resolver problemas rapidamente fazendo uso de soluções previamente definidas em outros problemas e com base em experiências anteriores”. Ou seja, para cada problema, é possível reconhecer padrões utilizados em problemas anteriores, podendo, assim, priorizar as diferenças desse novo problema. No caso da programação no *software* Scratch, podemos perceber, por exemplo, na criação de um jogo, personagens distintos que possuem uma mesma característica ou um mesmo comportamento. A partir do reconhecimento de padrões, é possível replicar a solução de um subproblema, simplificando a solução de um problema maior, modificando apenas as variações particulares de cada um. Na Figura 2, temos um exemplo de programação de um jogo para dois jogadores, com dois personagens.

Figura 2 - Exemplo de Reconhecimento de padrões na programação Scratch.



Fonte: Produção do autor.

Na Figura 2 (códigos de programação em blocos dos movimentos de subir dos dois atores), podemos perceber que as características semelhantes na programação dos atores (padrões), correspondem aos códigos de movimento no palco. Dessa forma, após a programação do primeiro ator e a identificação do padrão de movimento de ambos, pode-se utilizar a solução do primeiro na programação do segundo, otimizando tempo no desenvolvimento do jogo. O que

difere um ator do outro são os comandos do teclado e as suas fantasias, havendo um reconhecimento de padrões entre eles, utilizados para solucionar dois problemas diferentes.

Destacamos que, no desenvolvimento do pensamento matemático, também é usual em diversas áreas da Matemática a necessidade de reconhecer padrões para resolver problemas. Logo, percebemos a generalização do pensamento computacional como uma aproximação com o raciocínio utilizado em situações que exigem pensamento matemático.

2.3.3 Abstração

Segundo Wing (2008), a abstração é a essência do pensamento computacional. Wing explica que, na computação, noções são abstraídas além das dimensões físicas do tempo e do espaço, tornando-se mais ricas e complexas do que as ciências matemáticas e físicas.

> Ao trabalhar com abstrações ricas, definir a abstração 'certa' é fundamental. O processo de abstração - decidir quais detalhes necessitam destaque e os detalhes que podemos ignorar – baseia o pensamento computacional (WING, 2008, p. 3718, tradução do autor[8]).

Já Piaget (1995) define abstração em duas vertentes: a “abstração empírica”, a qual “se apoia sobre objetos físicos ou sobre aspectos materiais da própria ação, tais como movimento, empurrões, etc.” (PIAGET, 1995, p. 5); e a “abstração reflexionante”, que, ao contrário da abstração empírica,

> [...] apoia-se sobre tais formas e sobre todas as atividades cognitivas do sujeito (esquemas ou coordenações de ações, operações, estruturas, etc.) para delas retirar certos caracteres e utilizá-los para outras finalidades (novas adaptações, novos problemas, etc.) (PIAGET, 1995, p. 6)

Sendo assim, percebemos a abstração sendo fundamental, tanto no momento da generalização (reconhecimento de padrões), como também na criação de estruturas que representam o que foi abstraído. Ou seja, a abstração no pensamento computacional pode ser considerada um conjunto de informações ou ideias na qual o sujeito abstrai noções essenciais e os conceitua. Nesse processo,

---

[8] Citação original: “In working with rich abstractions, defining the ‘right’ abstraction is critical. The abstraction process—deciding what details we need to highlight and what details we can ignore—underlies computational thinking.” ((WING, 2008, p. 3718)

elementos que não são necessários são ignorados para que os elementos relevantes sejam priorizados, estruturando a solução do problema em jogo (BRACKMANN, 2017).

Podemos exemplificar as abstrações a partir do exemplo utilizado anteriormente em que, apesar de terem padrões em comum, diferenças são identificadas, retirando o que é pertinente e essencial, para que a programação de cada ator seja resolvida separadamente.

> A abstração consiste, por si mesma, com efeito, numa diferenciação, porquanto separa uma característica para transferi-la, e uma nova diferenciação acarreta a necessidade de integração em novas totalidades, sem as quais a assimilação deixa de funcionar, daí o princípio comum da formação das novidades: a abstração reflexionante conduz a generalizações, por isso mesmo construtivas, e não simplesmente indutivas ou extensivas como a abstração empírica. (PIAGET, 1995, p. 285)

Brackmann (2017) exemplifica a abstração com histórias envolvendo atividades matemáticas, nas quais a abstração das informações pertinentes é necessária para que a equação seja resolvida. No caso do computador, a programação no *software* Scratch proporciona a realização de abstrações de um conjunto de informações para que uma ideia seja possível.

> E assim as porcas e parafusos no pensamento computacional estão definindo abstrações, trabalhando com múltiplas camadas de abstração e entendendo as relações entre as diferentes camadas. Abstrações são as ferramentas "mentais" da computação. O poder de nossas ferramentas “mentais” é amplificado pelo poder de nossas ferramentas “metálicas” (WING, 2008, p. 3718, tradução do autor[9]).

Ou seja, o Scratch possibilita a materialização de uma ideia que dificilmente seria possível realizar no mundo real, utilizando os recursos do *software* para que as abstrações dessa ideia possam ser representadas no computador. Nesse sentido, o sujeito elabora estratégias de solução para um problema utilizando os recursos do software e os processos mentais de abstração.

Interessante observar que o conceito de pensamento computacional tem sido atualizado desde que o termo foi citado por Janette Wing em 2006. Segundo Rocha *et al* (2020, p. 584):

---

[9] And so the nuts and bolts in computational thinking are defining abstractions, working with multiple layers of abstraction and understanding the relationships among the different layers. Abstractions are the ‘mental’ tools of computing. The power of our ‘mental’ tools is amplified by the power of our ‘metal’ tools.” (WING, 2008, p. 3718).

> [...] o conceito de Pensamento Computacional ainda está em construção, e a própria Wing mostra suas evoluções a respeito da temática. Ainda com essas evoluções é possível identificar que no geral todas elas procuram associar ao termo a resolução de problemas e alguma forma de comunicar seja por meio de algoritmos ou outras formas de representação.

Logo, percebemos as aproximações da definição de abstração do pensamento computacional com o uso da abstração no pensamento matemático, em que é relacionada com a forma que representamos e organizamos as estratégias para resolver problemas.

2.3.4 Algoritmos

De acordo com Wing (2008, p. 3718, tradução do autor[10]), "Um algoritmo é uma abstração de um procedimento passo a passo para obter informações e produzir alguma saída desejada", ou seja, um conjunto de algoritmos sugere uma forma de se conceituar as abstrações, sendo um possível meio para expressar generalizações. Já para Brackmann (2017, p. 40), um algoritmo é

> [...] um conjunto de regras para a resolução de um problema, como a receita de um bolo; porém, diferentemente de uma simples receita de bolo, pode-se utilizar diversos fatores mais complexos. Existem algoritmos muito pequenos, que podem ser comparados a pequenos poemas. Outros algoritmos são maiores e precisam ser escritos como se fossem livros, ou então maiores ainda, necessitariam inevitavelmente serem escritos em diversos volumes de livros. Para entender melhor, é possível fazer questionamentos que possam facilitar a compreensão de como gerar e quais as limitações do mesmo, tais como: "É possível solucionar um problema utilizando algoritmos?", ou "Qual a precisão que se necessita para solucionar um problema?".

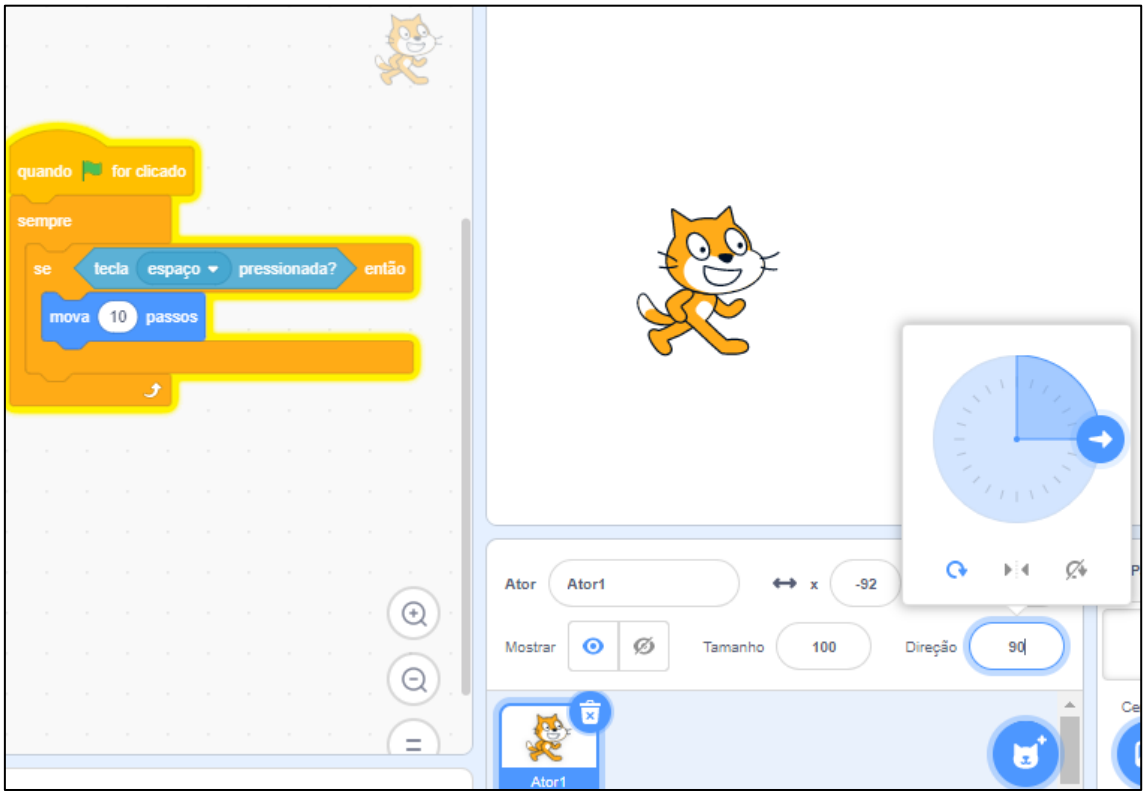
Brackmann (2017) ainda conclui que os algoritmos devem ser entendidos como soluções finais para um problema, já que os processos de decomposição, reconhecimento de padrões e abstração já foram estabelecidos. Esses algoritmos são determinados como uma sequência de informações já definidas, podendo ser utilizados diversas vezes para um determinado fim, funcionando como, por exemplo, uma generalização matemática que pode ser utilizada em diversos problemas semelhantes sem a necessidade de utilizar os mesmos mecanismos para se chegar a essa generalização.

---

[10] Citação original: "An algorithm is an abstraction of a step-by-step procedure for taking input and producing some desired output." (WING, 2008, p. 3718).

Por exemplo, no *software* Scratch, para movimentar um ator no palco, é necessário que o sujeito tenha uma ideia prévia de como deve ser seu movimento. Sendo assim, o sujeito que está programando deve abstrair as informações importantes e pertinentes para que o ator se movimente, organizar essas informações de modo a estruturar a comunicação com a máquina. No caso específico do Scratch, essa estrutura organizada de comunicação utiliza os blocos de programação, que permitem criar uma sequência de códigos para que o computador compreenda as informações, culminando em um algoritmo que coordena o movimento do ator. Na Figura 3, vemos que, para criar um movimento simples horizontal para a direita, foi necessário que estabelecer uma sequência organizada de comandos: um evento (clicar na bandeira verde para dar início à execução das ações do ator; um comando para coordenar o número de vezes que esse movimento se repete (nesse caso, sempre); um comando condicional que depende de uma ação do usuário, para disparar o movimento; e, por fim, o comando que coordena o movimento do ator.

Figura 3 - Exemplo de algoritmo na programação do Scratch.



Fonte: Produção do autor.

A generalização é um elemento do pensamento matemático, já que auxilia na resolução de problemas, utilizando-se da criação de algoritmos para resolver situações semelhantes. Logo, vemos o pensamento computacional como uma alternativa para auxiliar no desenvolvimento da generalização na resolução de problemas que exigem um pensamento matemático.

Apresentamos os quatro pilares do pensamento computacional apresentados por Brackmann (2017), visando identificar elementos que evidenciem o desenvolvimento dessas habilidades nas ações dos participantes da pesquisa, procurando aproximações das suas definições com elementos do pensamento matemático. Na próxima seção, apresentamos a teoria do Fazer e Compreender de Piaget.

2.4 Construção do conhecimento: do fazer ao compreender

Propomos uma investigação analisada à luz da teoria do desenvolvimento cognitivo de Jean Piaget, que busca compreender de que forma a criança se desenvolve e como o conhecimento progride da passagem da ação física para o pensamento (ação mental) que é efetuado a partir das tomadas de consciência. Segundo Piaget (1978),

> [...] existem ações complexas, embora de êxito precoce, que apresentam todas as características de um saber, mas apenas de um "savoir faire" (saber fazer), e que a passagem dessa forma prática de conhecimento para o pensamento se efetuava através de tomadas de consciência, sem se restringir de forma alguma, a uma espécie de esclarecimento, mas consistindo numa conceituação propriamente dita, isto é, numa transformação dos esquemas de ação em noções e em operações [...] (PIAGET, 1978, p. 10).

Dessa maneira, podemos dizer que o desenvolvimento do compreender se inicia pela ação, que busca alcançar um objetivo sem se preocupar com a compreensão dos meios nos quais foram utilizados para realizá-lo, até o nível em que essas ações são realmente compreendidas. Assim, dizemos que a passagem da prática para a conceituação parte da ação para o saber como agir.

> (...) a ação constitui um conhecimento (um *savoir faire*) autônomo, cuja conceituação somente se efetua por tomadas de consciência posteriores e que estas procedem de acordo com uma lei de sucessão que conduz da periferia para o centro, isto é, partindo das zonas de adaptação ao objeto para atingir as coordenações internas das ações (PIAGET, 1978, p. 172).

Nesse processo de desenvolvimento, a ação encontra-se presente em um momento inicial, tendo sua autonomia em relação à conceituação, que apresenta um

atraso em relação à ação. Podemos exemplificar essa situação quando uma criança está aprendendo a brincar com blocos de montar. Inicialmente, a criança brinca encaixando uma peça na outra, sem a necessidade que aquilo tenha uma forma definida, representando a fase da ação. A partir de determinado momento, esses encaixes começam a criar forma e fazer sentido para a criança, iniciando um processo de conceituação, mas ainda se formando a partir das ações. Em certo momento, a conceituação passa a ter influência sobre a ação, dando à ação uma capacidade de previsão, o que possibilita uma programação efetiva da ação a partir da conceituação.

No campo de conceitos matemáticos envolvidos na programação no *software* Scratch, podemos exemplificar a conceituação da movimentação dos atores no palco do Scratch, em que o conceito de plano cartesiano é trabalhado. Inicialmente, a criança testa valores e observa o que acontece na tela, para tentar acertar uma posição específica no palco. Posteriormente, compreende o comportamento do ator e a distribuição dos valores no palco, passando a conceituar o sistema de coordenadas implícito, prevendo a movimentação dos atores no palco antes de programá-los. Sendo assim, o programador consegue movimentar o ator de forma mais consciente, construindo previamente o movimento em suas relações mentais, tendo consciência dos valores a serem utilizados para alcançar seu objetivo. Assim, a conceituação passa a influenciar a ação, tendo a autonomia inicial da ação, mas com a conceituação cada vez mais influenciando as ações do programador.

Dessa forma, observa-se uma nivelação aproximada entre a ação e a conceituação, efetuando trocas constantes entre as duas no processo, oportunizando que a prática se apoie nas teorias. Com o estabelecimento da compreensão a partir dos processos de trocas entre ação e conceituação, percebe-se que a antecipação se torna mais frequente, possibilitando ao sujeito à escolha das ações por meios diferentes, permitindo a passagem do nível do comportamento material para o comportamento da representação.

Essa transição da ação do sujeito para a conceituação é construída a partir de ações coordenadas conduzidas por um aspecto lógico, não existindo uma passagem da não compreensão para o compreender, sendo um processo constante com níveis intermediários. Piaget (1978, p. 173) destaca “que essa transitividade não cai do céu e que ela é produzida pelas próprias ações, antes de se tornar um instrumento operacional generalizado”.

Segundo Piaget (1978) a conceituação se torna uma influência resultante da ação que se efetua a partir de tomadas de consciências posteriores. Piaget (1977) considera que

> (...) a “tomada” de consciência representa algo de diferente e que vai além de uma “tomada”, isto é, de uma incorporação de um campo dado de antemão com todos os seus caracteres e que seria a “consciência”: trata-se na realidade, de uma verdadeira construção, que consiste em elaborar, não “a” consciência elaborada com um todo, mas seus diferentes níveis enquanto sistemas mais ou menos integrados (PIAGET, 1977, p. 9).

A tomada de consciência torna-se um processo complexo que vai além da simples compreensão interna, existindo a necessidade de investigação das leis de conceituação existentes nesses casos. No ramo da psiquiatria, é considerado por Piaget (1977, p. 11) “psicólogos interessam-se sobre tudo em saber em que ocasiões há ou não tomada de consciência, mas negligenciaram demasiadamente a outra questão, que lhe é complementar e consiste em estabelecer “como” ela se processa”. Piaget (1977) complementa que, do ponto de vista epistemológico, nas estruturas operatórias, lógico-matemáticas como causais, encontra-se originada a interiorização das ações. Sendo assim, a tomada de consciência exige mais do que simplesmente uma iluminação, mas sim, um esquema de ações que se transformam em um conceito. Assim, essa tomada de consciência consiste, por sua vez, em uma conceituação.

A conceituação passa a ser um processo posterior à ação, tendo sua influência a partir de determinado nível. “O que a conceituação fornece à ação é um reforço de suas capacidades de previsão e possibilidade, em presença de uma dada situação, de dar um plano de utilização imediata” (PIAGET, 1978, p. 174). Dessa forma, o estabelecimento da compreensão sobre a ação possibilita uma capacidade de antecipação do indivíduo em relação a uma determinada situação, o que favorece a capacidade de escolha entre os diferentes meios de resolução, permitindo assim uma programação da ação a partir da conceituação.

Esses processos mentais podem ser identificados na atividade de programação, pois as atividades iniciais de programação observadas em crianças podem ser caracterizadas por processos de tentativa e erro, que estão relacionados aos processos de desenvolvimento do indivíduo, da ação à compreensão. Para que o *software* execute determinada ação, é necessário que o sujeito que está programando dê um comando que dispara essa ação. Se esse objetivo não é alcançado, ou seja, se o sujeito não obtém êxito na execução da ação, ele pode

experimentar outro comando para verificar se a ação será executada de forma correta, e dar continuidade a essa conduta de testar comandos, caracterizando um processo de tentativa e erro apoiado na ação pura, em que se evidencia uma autonomia da ação.

Contudo, experiências ou conceitos anteriores (ou experiências construídas na situação em foco) podem influenciar na escolha de comandos, fazendo com que o agente programador, a partir das ações, “estabeleça fronteiras entre a prática (“o que fazer para conseguir?”) e o sistema de seus conceitos (“por que as coisas se passam dessa maneira?”)” (PIAGET 1978, p. 174). Assim, quando há a compreensão da ação, o sujeito tem a capacidade da antecipação e, nesse caso, ele poderá antecipar a execução do programa a partir da sequência de comandos escolhida, o que permite identificar os comandos necessários de forma correta e não aleatória, passando do “fazer”, para o “compreender”. Nesse momento, a conceituação passa a ter influência sobre a ação, de modo que as ações são tomadas a partir de construções teóricas.

> [...] fazer é compreender em ação uma dada situação em grau suficiente para atingir os fins propostos, e compreender é conseguir dominar, em pensamento, as mesmas situações até poder resolver os problemas por elas levantados [...] (PIAGET, 1978, p. 176).

Piaget (1978, p. 179) ainda reforça que “compreender consiste em isolar a razão das coisas, enquanto fazer é somente utilizá-las com sucesso”. O fazer caracteriza-se como uma conjuntura primária e essencial para a compreensão, que por sua vez é um saber que ultrapassa a ação, podendo abdicar-se dela.

Desta forma, a tomada de consciência fundamenta-se na passagem da compreensão prática para uma compreensão conceitual, tornando o sujeito capaz da teorização antes da ação, e não mais apenas ação. Na seção a seguir, apresentaremos os trabalhos científicos já realizados com o mesmo tema da presente pesquisa, na busca de aproximações e distanciamentos.

2.5 Trabalhos correlatos

Diante da investigação que se propõe essa pesquisa e da necessidade de conhecer trabalhos científicos relacionados ao tema que já foram desenvolvidos, foi realizada uma busca que objetivou responder à pergunta: Qual o panorama atual de trabalhos que envolvem a programação em blocos no software Scratch para a

aprendizagem de Matemática, o desenvolvimento do pensamento computacional e como a teoria do Fazer e Compreender de Jean Piaget é utilizada na análise de dados produzidos? A partir dessa questão, foi realizada uma busca nos seguintes bancos: banco de teses e dissertações da CAPES, repositório digital da UFRGS Lume e na revista RENOTE.

No banco de teses e dissertações da CAPES, foram utilizadas as palavras-chave “Scratch” e “Pensamento Computacional” para a busca, selecionando o filtro “ensino de matemática”. Foram encontrados noventa e quatro trabalhos que propõem o *software* Scratch e o Pensamento Computacional para o Ensino de Matemática. Destes, foram selecionados três trabalhos, tendo o uso em conteúdos específicos como critério de descarte e a aplicação com estudantes de Ensino Fundamental anos finais e a temática de programação ou uso de jogos como critério de seleção.

Já no repositório digital da UFRGS Lume, foram realizadas duas pesquisas. Na primeira, buscamos trabalhos na categoria Teses e Dissertações, com destaque para os trabalhos em Informática na Educação, utilizando juntas as palavras-chave “Programação” (já que nenhum trabalho foi encontrado inserindo a palavra-chave “Scratch” com a seguinte) e “Pensamento Computacional”. Encontramos cento e quarenta trabalhos relacionados, nos quais escolhemos duas teses, com o critério de aplicação de atividades com estudantes de Ensino Fundamental. Na segunda pesquisa, também buscamos trabalhos que pudessem contribuir para uma melhor compreensão da teoria utilizada. Na categoria Teses e Dissertações, utilizando o filtro multidisciplinar, com o uso da palavra-chave “Jean Piaget”, foram encontrados trezentos e vinte e nove trabalhos, nos quais escolhemos três teses, tendo como critério os temas que se aproximam dos componentes pretendidos no desenvolvimento da pesquisa.

Por fim, fizemos uma busca por trabalhos na Revista Novas Tecnologias na Educação (RENOTE) utilizando também as palavras-chave “Scratch” e “Pensamento Computacional”, encontrando um total de quatro trabalhos. Selecionamos um trabalho, utilizando como critério de descarte os trabalhos com ênfase na formação docente e/ou trabalhos de análise de literatura. O Quadro 1 apresenta um resumo dos trabalhos que serão apresentados a seguir.

Quadro 1 - Trabalhos correlatos

Título	Autor(es)	Ano	Local	Caráter do trabalho	Local de Busca
Programação em Scratch na sala de aula de Matemática: Investigação sobre a Construção do Conceito de ângulo	Rocha	2017	Porto Alegre, RS	Dissertação	Banco de Teses e Dissertações da CAPES
O Software de Programação Scratch na Formação Inicial do Professor de Matemática Por Meio da Criação de Objetos de Aprendizagem	Curci	2017	Londrina, PR	Dissertação	Banco de Teses e Dissertações da CAPES
Contribuições da linguagem Scratch Para o Ensino de Geometria	Queiroz	2018	São Paulo, SP	Dissertação	Banco de Teses e Dissertações da CAPES
Desenvolvimento do Pensamento Computacional através de Atividades Desplugadas na Educação Básica	Brackmann	2017	Porto Alegre, RS	Tese	Repositório digital da UFRGS Lume
Aprendizagem do Pensamento Computacional e Desenvolvimento do Raciocínio	Boucinha	2017	Porto Alegre, RS	Tese	Repositório digital da UFRGS Lume
Espaço de aprendizagem digital da matemática: o aprender a aprender por cooperação	Bona	2012	Porto Alegre, RS	Tese	Repositório digital da UFRGS Lume
Cadeias de Markov e modelagem Matemática: da					

abstração pseudo-empírica à abstração refletida com uso de objetos virtuais	Silva	2015	Porto Alegre, RS	Tese	Repositório digital da UFRGS Lume
Comunicação e aprendizagem matemática *on-line*: um estudo com o editor científico ROODA exata	Notare	2009	Porto Alegre, RS	Tese	Repositório digital da UFRGS Lume
Pensamento Computacional no Ensino Médio: Práticas Mediadoras Utilizando a Linguagem Scratch	Poloni, Soares e Webber	2019	Caxias do Sul, RS	Artigo	Revista RENOTE

Fonte: Produção do autor.

Na descrição a seguir, serão apresentadas as questões norteadoras de cada trabalho, assim como os resultados obtidos, buscando fazer uma análise sobre os aspectos que se aproximam e se distanciam do nosso trabalho.

A primeira proposta analisada foi desenvolvida por Rocha (2017) em sua dissertação de mestrado pelo Programa de Pós-Graduação em Ensino de Matemática da Universidade Federal do Rio Grande do Sul e busca responder à seguinte pergunta: “Quais são as evidências de pensamentos matemáticos e como os estudantes as expressam em atividades de programação envolvendo o conceito de ângulo?” (ROCHA, 2017, p. 17). As atividades propostas para os estudantes envolviam duas etapas: desafios de jogos de labirintos programados previamente pela pesquisadora e a criação de um jogo de labirinto por parte dos estudantes.

Os dados foram analisados a partir de entrevistas inspiradas no método clínico de Piaget, tendo como material de análise vídeos das interações dos estudantes e os arquivos produzidos no *software* Scratch. Também, como suporte para análise dos dados produzidos, a Teoria dos Campos Conceituais de Gerard Vergnaud foi utilizada.

Como resultado, a pesquisadora constatou evidências de que os estudantes desenvolveram o pensamento matemático, mostrando o Scratch como uma ferramenta que possibilita estimular formas de pensar e explorar construções de conceitos matemáticos. Segundo Rocha (2017),

> [...] propõe-se o uso do Scratch como uma ferramenta que possibilita ao aluno entrar em um mundo simulado que lhe permite pensar, descobrir, testar seus teoremas-em-ação e conceitos-em-ação. [...] As tendências de pesquisa com o Scratch publicadas muitas vezes utilizam o *software* como apoio para algum conteúdo que já foi estudado em sala de aula. Essa proposta de pesquisa propõe que o *software* seja utilizado para explorações que auxiliem o aluno na construção do conceito. (ROCHA, 2017, p. 173).

O segundo trabalho analisado foi a dissertação de mestrado de Curci (2017) pelo Programa de Pós-Graduação em Ensino de Matemática da Universidade Tecnológica Federal do Paraná - UTFPR, tendo como questão norteadora: “Programar objetos de aprendizagem no Scratch contribui para que professores de matemática, em formação inicial, ensinem Geometria a partir de métodos inovadores?” (CURCI, 2017, p. 15). O estudo teve como proposta contribuir na formação inicial de estudantes de Licenciatura em Matemática, matriculados na disciplina denominada Mídias Tecnológicas no Ensino de Matemática de uma universidade pública do Estado do Paraná, visando o desenvolvimento de objetos de aprendizagem no formato de jogos digitais, programados no Scratch, para auxiliar no ensino de Geometria.

A pesquisa teve como foco principal a formação de professores no Brasil e a utilização de tecnologias educacionais nessa formação. Para analisar os objetos criados pelos estudantes, foram utilizados critérios definidos pelo Grupo de Pesquisa em Tecnologias na Educação Matemática (GPTEM) para objetos de aprendizagem. O autor diagnosticou que a maioria das produções dos estudantes possuía pelo menos três desses critérios, de modo que esses critérios contribuem para o ensino de Geometria. Também foi constatada pelo autor a contribuição do *software* Scratch para a formação inicial de professores de Matemática.

Analisamos também a proposta elaborada por Queiroz (2018) em sua dissertação de mestrado pelo Programa de Pós-Graduação Stricto Sensu em Matemática em Rede Nacional, Instituto Federal de Educação, Ciência e Tecnologia de São Paulo - IFSP. Não foi identificada uma pergunta norteadora específica, mas observou-se que, como objetivo, a pesquisa “tem a intenção de explorar as potencialidades da linguagem de programação Scratch e a possibilidade de integrá-la ao currículo de Matemática, com foco na Geometria” (QUEIROZ, 2018, p. 12).

A pesquisa teve como intuito apresentar o *software* Scratch como uma ferramenta didática para professores de Matemática que buscam o enriquecimento de suas aulas, apresentando uma sequência didática elaborada para o ensino de

geometria plana. Porém, verificamos que a sequência didática apresentada não foi aplicada.

Verificamos o uso do *software* Scratch no ensino de Matemática com diferentes propostas de abordagem. Percebemos propostas que têm como objetivo o desenvolvimento do pensar matemático do estudante, enquanto outras têm direcionamento para a formação de professores e a utilização do *software* por eles.

Nossa pesquisa tem interesse em investigar o pensamento matemático dos estudantes no ambiente de programação, assim como a proposta utilizada por Rocha (2017). Porém, nosso intuito é identificar e analisar o desenvolvimento de habilidades matemáticas nos estudantes quando estão programando no *software* Scratch, identificando possíveis conceitos matemáticos que emergem nesse ambiente de programação, além de identificar elementos que caracterizam o desenvolvimento do pensamento computacional.

O primeiro trabalho escolhido no Repositório digital da UFRGS Lume foi a tese de doutorado de Brackmann (2017) pelo Programa de Pós-Graduação em Informática na Educação (PPGIE) do Centro Interdisciplinar de Novas Tecnologias na Educação (CINTED) da Universidade Federal do Rio Grande do Sul. O trabalho buscou responder o seguinte problema de pesquisa: “Qual a eficácia da abordagem de ensino desplugado para promover o desenvolvimento do Pensamento Computacional na Educação Básica?” (BRACKMANN, 2017, p. 21). A pesquisa teve como objetivo verificar possibilidades para o desenvolvimento do pensamento computacional a partir de atividades que, exclusivamente, não necessitassem o uso de computadores, possibilitando beneficiar estudantes que não têm acesso aos mecanismos para o uso dessa tecnologia.

As atividades foram desenvolvidas no decorrer de dois projetos pilotos. No primeiro, o instrumento contém duas questões com a inclusão dos quatro pilares do pensamento computacional, obtendo os seguintes resultados:

> Com os resultados analisados, identificou-se que o ensino do Pensamento Computacional poderia ser uma possibilidade válida. Porém, seria necessário utilizar uma ferramenta avaliativa validada e a criação de novas atividades que pudessem exercitar todos os pilares do PC e de maneira mais extensa. Esses dois quesitos foram atendidos no segundo projeto piloto (BRACKMANN, 2017, p. 117).

Já no segundo projeto piloto, a ferramenta válida utilizada, citada anteriormente, foi o “Teste de Pensamento Computacional”, criado por Román-Gonzales *et al*. (2015, apud. BRACKMANN, 2017, p. 118), pois possui uma

abordagem quantitativa e aptitudinal, tendo sua validade comprovada a partir de um rigoroso processo (BRACKMANN, 2017).

Os instrumentos foram aplicados posteriormente em duas escolas em Madrid, na Espanha e em duas escolas no Brasil, na cidade de Santa Maria, RS. Os resultados foram analisados estatisticamente, comprovando os efeitos das atividades em relação ao desenvolvimento do pensamento computacional dos participantes da pesquisa. Os resultados obtidos através de uma abordagem Quase-Experimental em escolas Espanholas e Brasileiras, apresentam dados estatísticos que apontam uma melhoria significativa no desempenho dos estudantes que tiveram atividades de pensamento computacional desplugado em ambos os países.

A segunda pesquisa selecionada foi a tese de doutorado de Boucinha (2017) também pelo Programa de Pós-Graduação em Informática na Educação (PPGIE) do Centro Interdisciplinar de Novas Tecnologias na Educação (CINTED) da Universidade Federal do Rio Grande do Sul. A tese tem como objetivo identificar a relação entre a construção do pensamento computacional e o desenvolvimento do raciocínio (BOUCINHA, 2017, p. 12). Os dados da pesquisa foram produzidos a partir da formação de um curso de extensão de Desenvolvimento de Games, em duas escolas particulares de Porto Alegre, com estudantes do Ensino Fundamental anos finais.

Os dados foram analisados de forma quantitativa, a partir de análises estatísticas dos resultados das avaliações dos instrumentos: Teste de Pensamento Computacional, já citado no trabalho anterior, e provas que compõem a Bateria de Provas de Raciocínio. Como resultado, os dados informam que a construção do pensamento computacional a partir de atividades elaboradas com uma metodologia adequada, podem gerar alterações na capacidade cognitiva dos estudantes, mostrando uma “correlação existente entre o Pensamento Computacional e os demais tipos de raciocínio, reforçando sua importância no desenvolvimento cognitivo” (BOUCINHA, 2017, p. 97).

Podemos observar nos trabalhos apresentados que a criação de atividades e materiais planejados, com o intuito de desenvolver o pensamento computacional nos estudantes, são importantes para o desenvolvimento de habilidades que possibilitam seu uso na resolução de problemas de diversas áreas do conhecimento. Apesar do método de análise dos dados dos trabalhos serem um ponto de divergência, compreendemos que os trabalhos se aproximam com o nosso tema de pesquisa, já

que os objetivos alcançados produziram alternativas para que o pensamento computacional tenha mais espaço no ambiente escolar, assim como buscamos compreender e associar o pensamento computacional com o desenvolvimento de conceitos matemáticos presentes no currículo escolar.

Para ampliar nossa busca por trabalhos correlatos, também foram pesquisados trabalhos relacionados à teoria de Jean Piaget, pois esta é a fundamentação teórica que sustenta as análises da pesquisa.

A primeira análise foi feita sobre a tese de doutorado de Bona (2012), tendo como problema de pesquisa: “Como analisar e compreender o processo de aprendizagem cooperativa dos conceitos de matemática no espaço de aprendizagem digital?” (BONA 2012, p. 25). Juntamente com a teoria de Piaget, o trabalho teve apoio nas ideias de Otto Peters, Seymour Papert, Paulo Freire e Ubiratan D’Ambrosio, tendo a teoria de Piaget como sustentação teórica para a análise dos dados.

No subcapítulo que trata sobre a aprendizagem, a pesquisadora apresenta pesquisas relacionadas às obras de Piaget, enfatizando os conceitos de abstração reflexionante, tomada de consciência e estudos sociológicos. O subcapítulo busca discutir como se dá o processo de conceituação do sujeito, partindo da ação, que consiste em um conhecimento autônomo que pode ser executado sem uma tomada de consciência. A partir da ação com sucesso, sem a consciência dos meios utilizados, busca-se compreendê-las, tendo esse processo de compreensão definido por Jean Piaget como a tomada de consciência.

O processo de conceituação é entendido como uma reconstrução do processo de pensamento, definindo um processo mais elaborado do que a prática obtida pela ação.

> Tal reconstrução é a ação de fazer, refazer, reformular, ajudar, e corrigir de cada estudante individualmente e/ou coletivamente com o grupo de estudantes e com a professora num espaço propício ao desenvolvimento da aprendizagem. (BONA, 2012, p. 37)

Como resultado da pesquisa, a pesquisadora constatou que as tecnologias digitais possibilitam o aprender a aprender de forma cooperativa, valorizando a ação do estudante, assim como a aprendizagem dos conceitos matemáticos, contribuindo para a ressignificação da prática docente. A pesquisadora constatou ao final da sua pesquisa que a teoria de Jean Piaget:

> (...) se faz atual, moderna e interessante a cultura digital, mostrando-se capaz de ser incorporada à prática docente que visa a, através da apropriação das tecnologias digitais, potencializar o processo de aprender a aprender do estudante, seja ele consigo mesmo ou em grupo, tanto em Matemática como em qualquer outra área do conhecimento (...) (BONA, 2012, p. 230)

Dessa forma, nos aproximamos das ideias de Bona (2012) quanto à utilização da teoria do desenvolvimento cognitivo de Piaget para sustentar as análises dos dados em pesquisa envolvendo tecnologia digital e aprendizagem de Matemática.

O segundo trabalho analisado foi a tese de doutorado de Silva (2012), na qual procurava entender como se dava a evolução da abstração refletida na construção de conceitos matemáticos aplicados em uma sequência didática utilizando objetos virtuais, que exploravam situações-problemas envolvendo modelagem matemática em Cadeias de Marcov (SILVA, 2015).

As teorias de tomada de consciência e abstração reflexionante de Jean Piaget foram utilizadas para a análise dos dados. Segundo Silva (2015, p.44), a tomada de consciência consiste em “uma condição necessária e suficiente para a construção do conhecimento na ação do sujeito sobre os objetos”.

Silva (2015) discute a diferenciação proposta por Piaget sobre os termos “insight” e “tomada de consciência”, em que o primeiro consiste em uma inspiração na qual as características de pensamento pelo sujeito não são essenciais para a conclusão da ação, enquanto o segundo processo, no qual apresenta evolução nas fases de pensamento, refere-se às ações coordenadas do sujeito sobre os objetos. Relacionamos essas etapas com a atividade de programação dos participantes da nossa pesquisa, referente ao que esperamos deles frente às ações aplicadas sobre o objeto, o *software*.

Como resultados, Silva (2015) identificou que os participantes criaram e mantiveram uma nova forma de pensamento: o pensamento hipotético contínuo (SILVA, 2015). A utilização das teorias de Jean Piaget apresentou papel fundamental para auxiliar na análise e interpretação dos registros obtidos na pesquisa, contribuindo para a verificação dos “graduais e processuais micro-avanços dos sujeitos na direção do aumento do conhecimento ou capacidade cognitiva”.

Por fim, foi analisada a tese de doutorado de Notare (2009), na qual procurava responder o seguinte problema de pesquisa: “Como uma ferramenta, incorporada em um ambiente virtual de aprendizagem, pode auxiliar na comunicação

científica *on-line* e no processo de construção de conhecimento de conceitos matemáticos?" (NOTARE, 2009, p. 22).

A teoria do desenvolvimento cognitivo de Jean Piaget foi utilizada para analisar os processos cognitivos que surgiram a partir das interações em um ambiente virtual de aprendizagem e comunicação *on-line*, com o propósito de utilizar o editor científico ROODA Exata como suporte para a linguagem científica. No capítulo referente ao desenvolvimento cognitivo pela visão de Piaget, o desenvolvimento da inteligência é apresentado ao olhar do teórico, como uma forma de se adaptar a novas situações, implicando a construção de novas estruturas de pensamento. A tomada de consciência torna-se possível a partir do processo de construção dessas novas estruturas que se formam pela ação do sujeito.

Como resultado da pesquisa, foi evidenciado que o editor ROODA Exata possibilitou uma melhor comunicação matemática *on-line*, o que seria muitas vezes improvável sem a sua utilização, devido à utilização de expressões matemáticas complexas. Nesse processo, a teoria de Piaget forneceu suporte para que os processos cognitivos desencadeados pelas interações com o editor científico pudessem ter sido analisados.

> A utilização de um ambiente de aprendizagem como apoio a disciplinas presenciais mostrou-se eficiente, uma vez que favorece o exercício da comunicação e expressão Matemática. A partir das soluções apresentadas pelos alunos, os mesmos precisaram justificar os meios que levaram aos resultados obtidos. Este exercício de argumentação leva à reflexão de suas ações, conduzindo à tomada de consciência dos conceitos envolvidos na resolução do problema. (NOTARE, 2009, p. 176).

Nesse sentido, percebemos a importância da fala dos participantes no processo de análise dos dados produzidos, observando as construções de pensamento que levam à tomada de consciência, o que se aproxima das intenções de nossa pesquisa.

Ao observarmos a teoria escolhida para análise dos dados produzidos, aplicada aos trabalhos citados e analisados acima, percebemos que a teoria é adequada para as análises dos nossos dados produzidos, contribuindo de forma fundamental para a conclusão e sucesso da nossa pesquisa.

Referente ao trabalho selecionado na revista RENOTE, apresentamos o artigo de Poloni *et al* (2019), da Universidade de Caxias do Sul (UCS). O artigo apresenta "um estudo cujo foco foi identificar formas de mediação possibilitadas pela linguagem Scratch no processo de ensino-aprendizagem de programação" (POLONI

et al, 2019, p. 1). A metodologia utilizada foi o estudo de casos, criado a partir de uma oficina de programação no Scratch. Para análise dos dados, a pesquisa baseou-se na teoria sociointeracionista de Lev Vigotsky.

Como resultados, concluiu-se que o Scratch “cumpre seu papel de mediadora do processo de aprendizagem de programação e contribui para o desenvolvimento de princípios do pensamento computacional” (POLONI *et al*, 2019, p. 9). O artigo também deixa claro, como aspecto resultante da análise, a importância da atuação do professor como mediador para o desenvolvimento progressivo do estudante.

> [...] intervenções problematizadoras e instigadoras realizadas pelo professor. Essas são práticas mediadoras com possibilidades de levar o estudante a dar um “salto”, que é sair da zona potencial de desenvolvimento para a zona de desenvolvimento real. Nessa perspectiva, o uso do Scratch, articulados às intervenções e ações do professor, pode resultar em práticas interativas potencializadoras da aprendizagem de programação. (POLONI *et al*, 2019, p. 9).

Observamos um afastamento referente à conclusão do artigo com a nossa pesquisa, na qual pretende observar as ações dos estudantes em suas construções mentais, evitando a mediação, compreendendo o uso intuitivo da programação no Scratch para que não haja influência no desenvolvimento da conceituação a partir da ação (PIAGET, 1978), baseando nossa postura no método clínico de Jean Piaget. Porém, não descartamos a possibilidade de eventuais intervenções, procurando compreender ao máximo as dificuldades e erros dos participantes, com a mesma importância que daremos às facilidades e acertos.

No próximo capítulo apresentaremos o software Scratch.

3 O *SOFTWARE* SCRATCH

Scratch é um ambiente de programação desenvolvido no MIT, Massachusetts Institute of Technology. O Scratch segue a linha construcionista de Papert, tendo as mesmas funcionalidades do ambiente LOGO[11], porém, com uma interface gráfica mais intuitiva e mais atrativa para crianças com a utilização de blocos, parecidos com os blocos do Lego®, que minimiza o uso de códigos.

De acordo com Resnick et al. (2009), quando os computadores pessoais foram introduzidos no final dos anos 1970 e 1980, a possibilidade de ensinar todas as crianças a programar foi vista com entusiasmo. Porém, o Logo não conseguiu cumprir totalmente a tarefa de introduzir programação para as crianças devido a suas primeiras linguagens de programação, que eram difíceis de usar e as crianças não conseguiam dominá-las (RESNICK, 2009). Outros fatores que influenciaram na desistência da introdução da programação foram: o contexto no qual a programação foi introduzida, no qual ninguém poderia oferecer orientação quando ocorriam erros na programação, ou incentivar a exploração quando a programação ocorria de forma correta, e a introdução da programação em atividades que não eram ligadas aos interesses dos jovens, como gerar listas de números primos ou desenhar linhas simples.

Resnick et al. (2009) ainda explica que Papert argumentava que a linguagem de programação deveria ter um andar baixo, ou seja, um fácil começo, um teto alto, que representa as oportunidades de criar cada vez mais projetos complexos com o tempo, e paredes largas, para poder suportar diferentes tipos de projetos para que pessoas com diferentes interesses e estilos de aprendizagem possam ser incluídos. Dessa forma, ao ser criado o Scratch, três princípios foram estabelecidos para o seu design: tornar-se mais fácil de utilizar, mais significativo para o indivíduo que o utiliza, encontrando algo que o faça se identificar no contexto, e mais social do que outros ambientes de programação (RESNICK, 2009).

---

[11] A linguagem Logo foi desenvolvida na década de 60 no MIT (*Massachussets Institute of Technology)* em Cambridge, Massachusetts, Estados Unidos, pelo educador matemático Seymour Papert. A linguagem LOGO propõe facilitar a comunicação entre usuário e o computador, voltada para o ambiente educacional. É proposto que o estudante seja ativo na construção do seu conhecimento, desenvolvendo sua capacidade intelectual.

Nesse sentido, o *software* Scratch oportuniza ao estudante utilizar conceitos básicos de programação, possibilitando o incentivo ao estudo de conceitos matemáticos e computacionais. Na sequência do capítulo, detalharemos a interface do Scratch, suas principais ferramentas, assim como uma breve apresentação do site elaborado pelos criadores do Scratch.

3.1 O site do Scratch

Resnick et al. (2009) ressaltam o desejo de desenvolver uma abordagem para a programação que despertaria o interesse de pessoas que não se imaginavam como programadoras.

> Queríamos facilitar para todos, de todas as idades, origens e interesses, para programarem suas próprias histórias interativas, jogos, animações e simulações, e compartilharem suas criações entre si. Desde o lançamento público em maio de 2007, o Scratch Web site (http://scratch.mit.edu) tornou-se uma comunidade online vibrante, com pessoas compartilhando, discutindo e remixando um ao outro de seus projetos. (RESNICK et al., 2009, p. 60, tradução do autor[12])

Ao acessar o site, nos deparamos com um grande número de animações, histórias, jogos, entre outros, produzidos por pessoas das mais variadas características e de diversos lugares. No site é possível realizar um perfil de usuário a partir de um cadastro, possibilitando a postagem de suas produções, nas quais os demais usuários, visando a troca de ideias e o auxílio colaborativo, podem fazer comentários, fazer *download* e fazer sugestões de modificação dos projetos postados.

O site também disponibiliza o *download* do Scratch e uma versão online, além de possuir espaços de tutoriais e fóruns para a troca de ideias entre usuários. Atualmente o site dispõe de um cadastro especial para professores, em que, mediante esse cadastro, que é posteriormente analisado por uma equipe responsável, é possível o professor usuário criar turmas, cadastrar estudantes e inserir materiais.

---

[12] Citação original: “We wanted to make it easy for everyone, of all ages, backgrounds, and interests, to program their own interactive stories, games, animations, and simulations, and share their creations with one another. Since the public launch in May 2007, the Scratch Web site (http://scratch.mit.edu) has become a vibrant online community, with people sharing, discussing, and remixing one another’s projects.” (RESNICK et al., 2009, p. 60).

3.2 A Interface do *Software*

A interface do Scratch é composta por quatro áreas, sendo uma interface intuitiva. Detalhamos a interface do Scratch online (já que será a ferramenta na qual os participantes da pesquisa irão explorar), numerando a imagem da Figura 4 com as seguintes legendas:

1. Palco: Local onde os objetos são inseridos e onde é possível ver os resultados da programação criada;
2. Lista de *Sprites*: Área onde os *sprites* (atores) que fazem parte de projeto são localizados;
3. Blocos de comandos: Lista de Categorias e blocos de comandos;
4. Área de comandos: Local em que os comandos são conectados e editados.

Figura 4 - Interface do Scratch online

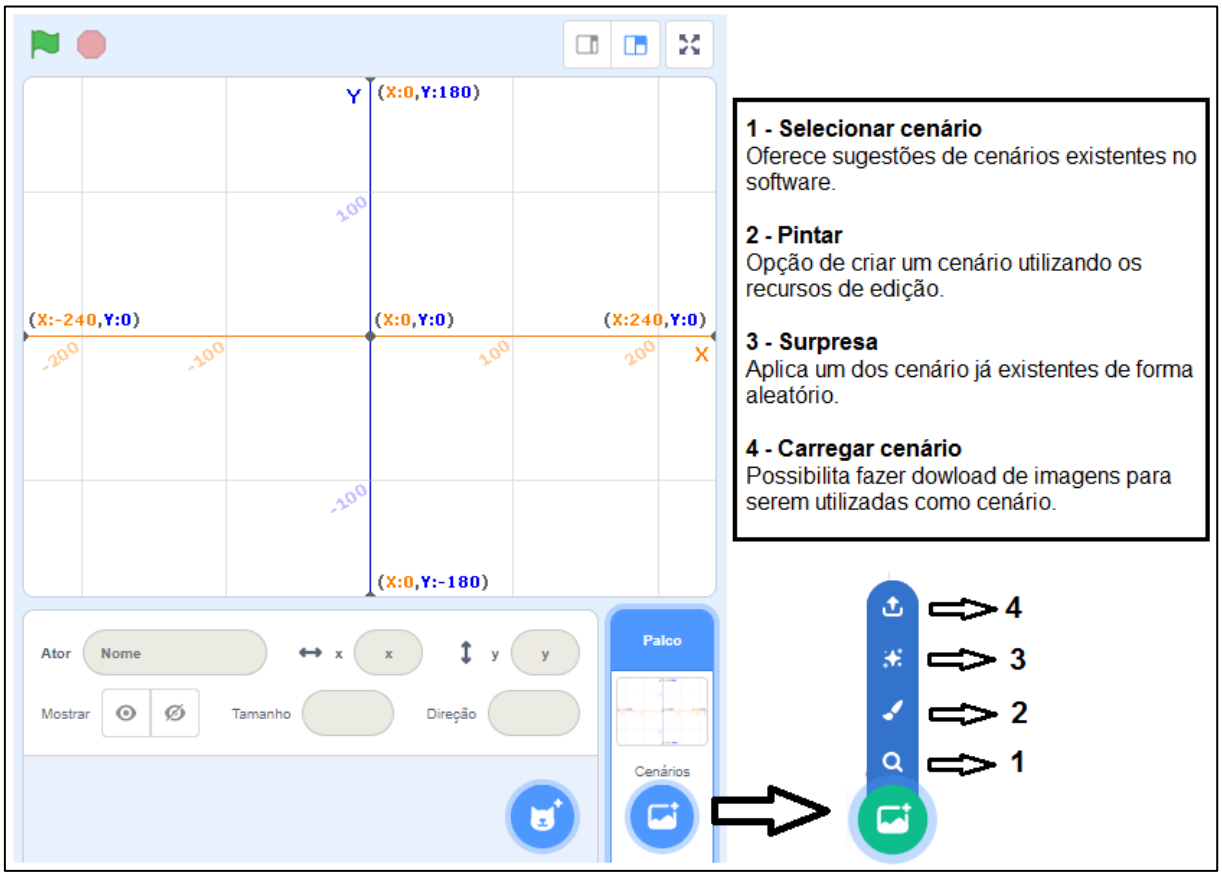
Fonte: Produção do autor.

Com o objetivo de compreender melhor as funções do Scratch e suas possibilidades, nas subseções seguintes, vamos apresentar as quatro áreas da interface (palco, blocos de comando, área de comando e lista de *scripts*) com mais detalhes e as ferramentas de recursos de edição do *software*, que possibilitam a criação de novos *sprites* ou edição dos já existentes.

3.2.1 Palco

O palco do Scratch possui um plano de coordenadas cartesianas sem eixos visíveis. O centro do palco localiza-se na posição de origem do sistema (0,0), sendo que o intervalo do eixo x que está visível na tela é entre -240 e 240 e o intervalo do eixo y é entre -180 e 180. As unidades de medidas utilizadas no Scratch são chamadas passos, ou seja, o palco possui 480 passos de largura e 360 passos de altura. A Figura 5 exemplifica o plano cartesiano existente no palco e as opções disponíveis para se utilizar o palco.

Figura 5 - Palco do Scratch.



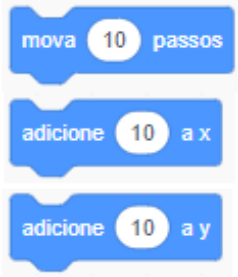
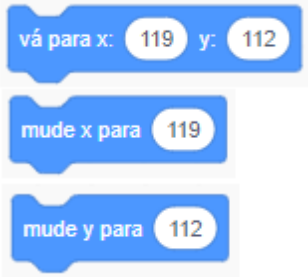
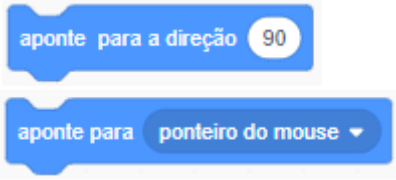
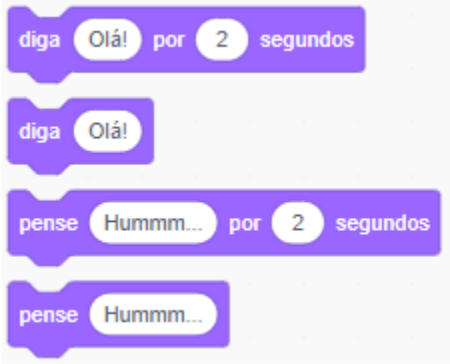
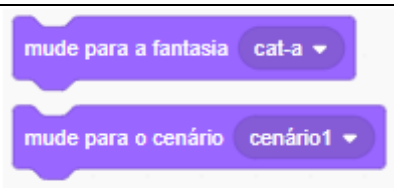
Fonte: Produção do autor.

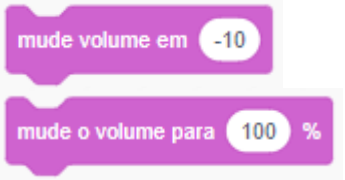
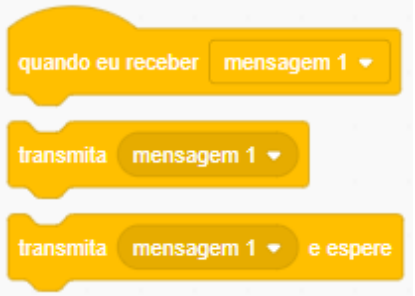
3.2.2 Blocos de comando

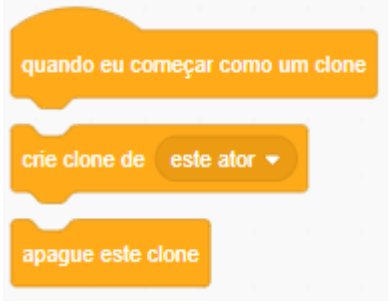
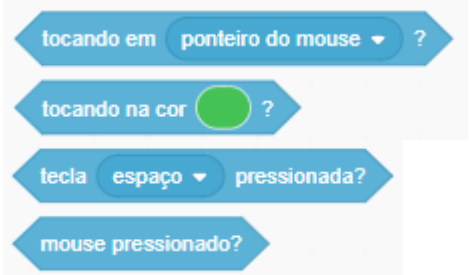
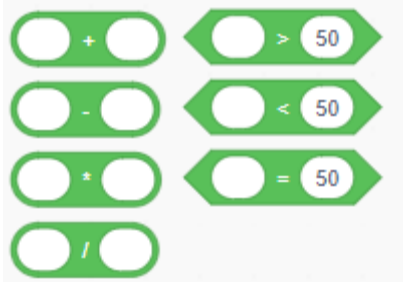
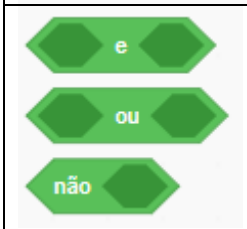
São nove categorias em que os blocos do Scratch online estão divididos. Os blocos são organizados por cor para ajudar a identificar os blocos que se relacionam por categoria. No Quadro 2, apresentamos as categorias com alguns dos seus

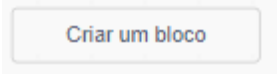
blocos. Esses blocos foram escolhidos por terem sido utilizados com frequência pelos estudantes durante os encontros.

Quadro 2 - Comandos do Scratch e suas funções

CATEGORIA	BLOCO	AÇÃO
Movimento		Movimenta o *sprite* de acordo com o número de passos inseridos no campo, ou para direção apontada ou adicionando unidades em sua coordenada.
		Gira o *sprite* para a direita ou para a esquerda o valor, em graus, que é inserido no campo.
		Desloca o *sprite* para a uma nova posição (x,y).
		Gira o *sprite* para a direção indicada no campo ou em direção ao ponteiro do mouse.
Aparência		Um balão de fala/pensamento é exibido acima do *sprite*, sendo possível estipular um tempo para que o balão fique visível.
		Altera a fantasia do *sprite* ou o cenário.

		Esconde ou mostra o *sprite*.
Som		Opções para inserir sons.
		Aumenta ou diminui o volume do som escolhido.
Eventos		Executa um conjunto de procedimentos a partir de determinada ação ou momento.
		Permite enviar mensagens a um ou mais *sprites* que servem para coordenar ações. Ao receber uma mensagem um ou mais *sprites* executam as ações.
Controle		Determina um espaço de tempo para que um comando seja executado.
		Repete os comandos de acordo com o número de vezes que se deseja repeti-los.
		Repete sempre um comando ou um conjunto de comandos.

		Realiza ações a partir de uma condição específica, podendo executar outra ação caso a condição anterior não ocorra.
		Para todos os *sprites.*
		Cria clones de um determinado *sprite* podendo estabelecer um ou mais comandos para que os clones executem. Caso necessário, para que todos os clones não continuem na programação, é possível excluir os clones a partir de determinada condição.
Sensores		São condições para que o *sprite* execute uma ou mais ações. São inseridos dentro dos blocos de controle e de comando.
Operadores		Operadores matemáticos que são inseridos dentro de outros blocos para realizar operações ou estabelecer critérios de igualdades ou desigualdades.
		São conjunções utilizadas para executar ações quando duas situações ocorrerem ao mesmo tempo ou que pelo menos uma ocorra. Também pode ser utilizada para executar uma ou mais ações quando determinada situação não

		ocorrer. São inseridos dentro dos blocos de controle e de comando.
Variáveis		Possibilita a criação de variáveis para um ou mais *scripts*. Essas variáveis são utilizadas para, por exemplo, executar ações diferentes para o mesmo *script* em determinadas situações diferentes, ou também, estabelecer número de vidas para um personagem de um jogo.
Meus blocos		Cria um novo bloco que representa a programação de um conjunto de ações.

Fonte: Produção do autor.

3.2.3 Área de comandos

Para que os *sprites* façam algo no palco, é necessário arrastar blocos, da aba de blocos de comando, para a área de comandos, encaixando-os e construindo, assim, programações que atribuam ações para o ator. Esses conjuntos de blocos unidos formarão o que chamamos de *script.* Os blocos encaixam-se de determinadas maneiras, impedindo que haja erros que tendem a ocorrer, principalmente em linguagens de programação que exigem a digitação dos comandos. Marji (2014) observa que os *scripts* podem ser testados à medida que forem criados, sem a necessidade de que estejam completos. É possível clicar sobre o primeiro bloco de uma sequência de blocos para que essa sequência seja imediatamente executada. Também é possível arrastar para a área de comando, blocos desconectados para testá-los individualmente, fazendo uma análise de suas funções dentro da programação. Os testes possibilitam ao sujeito que está programando encontrar *bugs* (erros) na sua programação, partindo para um processo de *debugging*, termo utilizado para indicar a análise para encontrar e solucionar um *bug.* Quando um bloco ou conjuntos de blocos são executados, é

possível visualizar a execução do ator no palco, possibilitando uma análise do que está sendo programado.

Para cada *sprite*, o Scratch permite programações paralelas que possibilitam eventos distintos que podem acontecer simultaneamente (Figura 6).

Figura 6 - Programação paralela para um mesmo *sprite*.

Fonte: Produção do autor.

Quando a bandeira verde é pressionada, o *script* da Figura 6a é executado, fazendo com que o *sprite* execute os comandos listados na sequência do evento. Se a tecla “espaço” for pressionada, o *script* da Figura 6b executará o comando em paralelo com o comando já em execução, fazendo com que o ator possa, nesse caso, mover-se pelo palco e dizer “olá” ao mesmo tempo.

3.2.4 Sprites

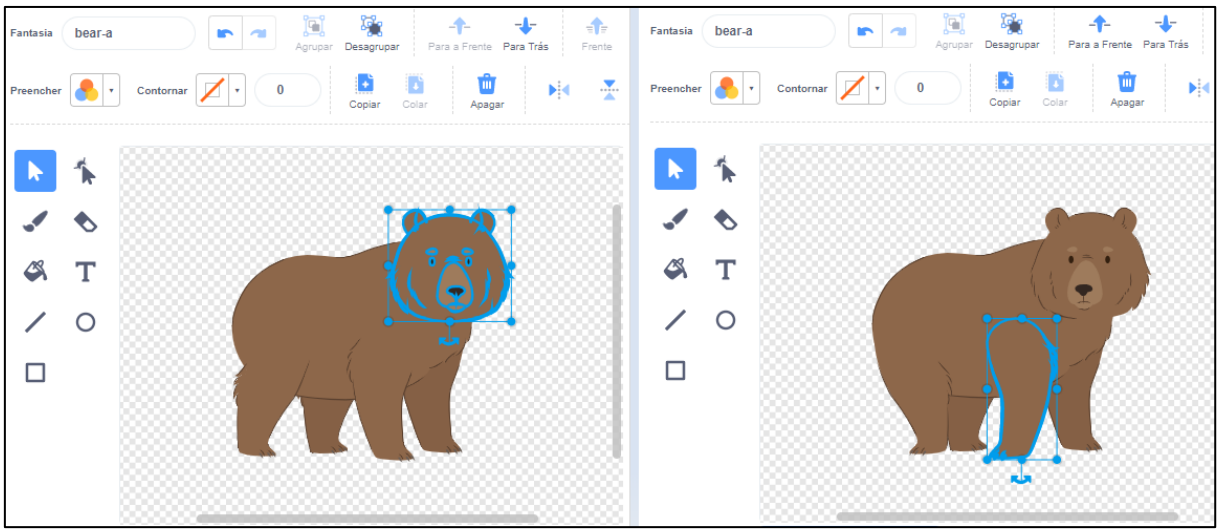
Em um mesmo projeto, é possível ter mais de um *sprite,* sendo que cada *sprite* terá a sua programação própria, ou seja, cada ator no palco terá uma lista de comandos a serem executados, mesmo que *sprites* diferentes executem os mesmos movimentos, deverá ser criado um conjunto de comandos para cada um deles.

Interessante ressaltar que o Scratch possui uma lista com vários *sprites* para que possam ser utilizados como atores. Alguns desses *sprites* já possuem mais de uma fantasia para que as animações sejam facilitadas.

3.2.5 Recursos de Edição de Imagem (Paint Editor)

O Scratch possui um espaço para a criação de *sprites* e cenários, possibilitando também a edição do material já disponibilizado. Essas edições e criações podem ser realizadas clicando em  (pincel), localizado como opção ao selecionar um ator ou selecionar um cenário. Ao clicar no pincel para criar ou editar um novo ator, a aba “Fantasias”, que se encontra acima das opções de blocos, é acessada, ou na aba “Cenários”, caso seja desejado criar ou editar um cenário. Na Figura 7 podemos observar a edição de um ator já disponível no Scratch (grande parte dos *sprites* oferecidos pelo Scratch é construída em camadas, possibilitando a edição por camadas desses *sprites*).

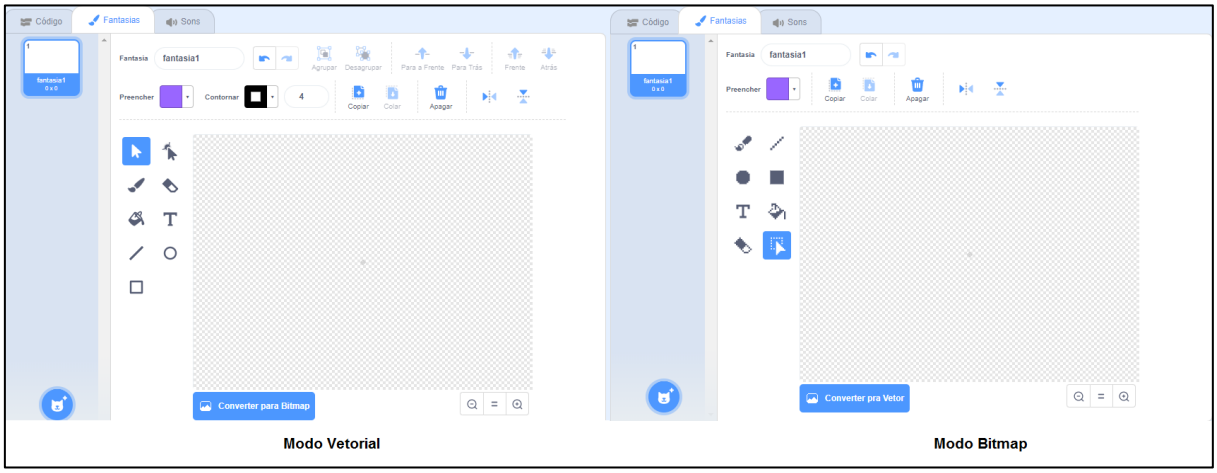
Figura 7 - Edição de *sprites* no Scratch.



Fonte: Produção do autor.

Na Figura 7, vemos um exemplo de desenho sendo trabalhado no formato vetorial, que permite trabalhar imagens a partir de descrições geométricas, sendo essas descrições feitas a partir de vetores. Também é possível utilizar desenhos no formato bitmap, em que as imagens são formadas a partir de pontos minúsculos chamados pixels. Na Figura 8 mostramos a interface e as ferramentas disponíveis nos dois formatos de desenho.

Figura 8 - Recursos de edição de *sprites* do Scratch



Fonte: Produção do autor.

No capítulo seguinte, serão apresentados os procedimentos metodológicos que direcionaram a pesquisa.

4 PROCEDIMENTOS METODOLÓGICOS

Para respondermos ao nosso problema de pesquisa: "*Quais as contribuições do uso da programação no software Scratch no desenvolvimento de conceitos matemáticos e do pensamento computacional com estudantes de Ensino Fundamental?",* foi elaborada e oferecida uma oficina para estudantes do oitavo e do nono ano do Ensino Fundamental. Os estudantes, organizados em grupos, foram desafiados a programar um jogo utilizando o Scratch online. Nesse processo de programação de jogos, buscamos identificar e analisar quais conceitos matemáticos foram desenvolvidos e utilizados no decorrer da oficina, observando e analisando como os participantes pensam e quais as estratégias utilizadas para as suas programações, verificando se características do pensamento computacional encontram-se presentes durante as ações dos estudantes.

> A produção do jogo é uma proposta pedagógica que une não somente a criatividade e o desenvolvimento de habilidades específicas, como imaginar, criar e inventar, cenários, personagens, a trama e as etapas (fases), mas também propõe, em sua própria dinâmica e organização, a possibilidade do estudante estabelecer ideias, produzir significados e traçar diferentes estratégias para construí-los a partir da análise, interpretação e composição dos mais variados algoritmos lógicos, relacionais e operatórios matemáticos. (AZEVEDO et al. 2018, p. 953)

Assim, apresentamos neste capítulo a estruturação da pesquisa. Iniciamos trazendo características da investigação qualitativa, observando a relação dos dados produzidos com o objeto de pesquisa, definindo o caráter da investigação. Seguimos abordando aspectos do método clínico, utilizado como inspiração metodológica para a produção e análise dos dados da pesquisa. Em seguida, descrevemos o local no qual foram produzidos os dados, os participantes da pesquisa, os recursos utilizados para produção dos dados e, finalmente, uma breve apresentação dos encontros da oficina.

4.1 Pesquisa Qualitativa

A presente pesquisa tem como norte para a produção de dados a abordagem qualitativa, na qual:

> Alguns investigadores movimentam-se nas escolas munidos de blocos de apontamentos para registrarem os dados. Outros recorrem ao equipamento de vídeo na sala de aula e não seriam capazes de conduzir uma

> investigação sem ele. Outros ainda elaboram esquemas e diagramas relativos aos padrões de comunicação verbal entre os alunos e professores. No entanto, todos eles têm em comum o seguinte: o seu trabalho corresponde à nossa definição de investigação qualitativa e incide sobre diversos aspectos da vida educativa (BOGDAN, BIKLEN ,1991, p. 47).

Bogdan e Biklen (1991) ainda definem a investigação qualitativa como possuindo cinco características, deixando claro que nem todos os estudos dispõem de todas essas características, podendo não possuir uma ou mais delas

A primeira característica citada pelos autores, corresponde à relevância do meio em que a pesquisa está sendo desenvolvida, reconhecendo o local e o contexto, fatores importantes para a observação e coleta dos dados investigados. Sendo assim, para um investigador qualitativo, é entendido que “as ações podem ser melhor compreendidas quando são observadas no seu ambiente habitual de ocorrência'' (BOGDAN E BIKLEN ,1991, p. 48), fato que justifica a presença do investigador nos locais da pesquisa.

Como a investigação qualitativa tem como dados coletados e/ou produzidos palavras e imagens, não números, a segunda característica define a investigação qualitativa como descritiva. Sendo assim, os dados produzidos, que podem ser transcrições de entrevistas, notas de campo, fotografias, vídeos, documentos pessoais ou outros registros oficiais, são analisados respeitando a forma que são registrados ou transcritos, partindo da “ideia de que nada é trivial, tudo tem potencial para constituir uma pista que nos permita estabelecer uma compreensão mais esclarecedora do nosso objeto de estudo” (BOGDAN E BIKLEN ,1991, p. 49).

Já a terceira característica define que “os investigadores qualitativos se interessam mais pelo processo do que simplesmente pelos resultados ou produtos” (BOGDAN E BIKLEN ,1991, p. 49). Segundo Bogdan e Biklen (1991), a interação entre professores e crianças tem sua relevância no processo, fazendo parte dos resultados produzidos, tendo a ênfase no processo um aspecto muito útil na investigação educacional.

A quarta característica define que os dados da pesquisa são utilizados para a construção das abstrações durante o processo de agrupamento dos dados recolhidos, sem o objetivo de confirmar ou negar hipóteses pré-estabelecidas. Ou seja, os dados não são analisados de forma que deem sentido a um resultado conhecido, mas sim para chegar a um resultado que vai se construindo na medida em que os dados são recolhidos e examinados. Sendo assim:

> O investigador qualitativo planeja utilizar parte do estudo para perceber quais as questões mais importantes. Não presume que se sabe o suficiente para reconhecer as questões importantes antes de efetuar a investigação (BOGDAN, BIKLEN ,1991, p. 50).

A quinta característica aborda a importância do significado na abordagem qualitativa, observando a importância de reconhecer os fatores e significados que estão no contexto da pesquisa, fazendo com que o investigador busque compreender a perspectiva dos diferentes participantes da pesquisa. Ou seja, “o processo de condução de investigação qualitativa reflete uma espécie de diálogo entre os investigadores e os respectivos sujeitos” (BOGDAN E BIKLEN ,1991, p. 51).

Dessa forma, a partir das características da investigação qualitativa definidas por Bogdan e Biklen (1991), norteamos nossa pesquisa, tratando-se assim de uma investigação qualitativa na qual buscamos investigar as contribuições da programação no Scratch no desenvolvimento de conceitos matemáticos.

4.2 Método clínico

Para a produção dos dados, buscamos inspiração nas entrevistas do método clínico de Jean Piaget. Segundo Delval (2002, p. 53):

> A essência do método clínico consiste na intervenção constante do experimentador em resposta à atuação do sujeito, com a finalidade de descobrir os caminhos que segue seu pensamento, dos quais o sujeito não tem consciência e que, portanto não pode tornar explícitos de maneira voluntária. Por isso, essa intervenção é orientada pelas hipóteses que o experimentador vai formulando acerca do significado das ações do sujeito.

O método clínico foi esboçado por Jean Piaget em 1921, passando por diversas etapas de aprimoramento. A primeira etapa, considerada como esboço do método, foi utilizada por Piaget até 1926, e se constituía em observações de pequenas experiências, sendo puramente verbal, utilizando-se da observação ao lado de conversas abertas com as crianças sobre alguns problemas. A segunda etapa é a constituição do método, utilizado entre 1926 e 1932, na qual o método tem sua formulação explícita contendo suas características e dificuldades. Piaget busca desconsiderar a entrevista puramente verbal e apoiar-se na ação do sujeito, com o intuito de estudar os conteúdos do pensamento, observando as concepções das crianças a partir do estudo de suas ideias morais, concepção de normas e compreensão da justiça.

Na sequência, no período de 1932 a 1940, Piaget modifica seu objeto de interesse. Em suas obras, observa-se que seu objeto de pesquisa passa a ser as origens da inteligência, dando início a etapa do método não-verbal, aplicando o método em participantes que ainda não falam, tornando os métodos puramente verbais inúteis nesse caso. A próxima etapa do método é denominada manipulação e formalização, etapa na qual nos inspiramos para produzir os dados. Foi utilizada por Piaget entre 1940 e 1955, tendo como estratégia de validação do método a explicação como complemento da ação, ou seja, o sujeito resolve tarefas mediante sua ação, pedindo explicações do que se fez. A partir de 1955, alguns desenvolvimentos posteriores tem a tentativa de utilizar dados estatísticos, não tendo representações significativas de mudanças no método.

Ao decorrer das mudanças e variações do método, temos como características a intervenção sistemática do experimentador a partir da ação do sujeito, seja qual for essa ação, seja ela verbal, de manipulação de um objeto seguido de explicação ou por si mesma. Essas características supõem que o método clínico é um procedimento de investigação do sujeito a partir das ações ou das palavras, sendo muitas vezes confundidas como um método de entrevista, a partir de conversas com a criança. Porém, para Delval (2002), o método clínico difere-se pelo fato de a interação com o sujeito estar relacionada com o tipo de atividade do experimentador.

> Partimos do suposto de que o método clínico é um procedimento para investigar como as crianças pensam, percebem, agem, e sentem, que procura descobrir o que não é evidente no que os sujeitos fazem e dizem, o que está por trás da aparência de sua conduta, seja em ações ou palavras. Dado que muitas vezes o método clínico consiste em conversas com o sujeito, tende a ser identificado frequentemente com um método de entrevista verbal, de puras conversas com as crianças. Contudo, como já assinalamos, isso não é verdade e presume apenas uma visão superficial, visto que a essência do método não está na conversa, mas sim no tipo de atividade do experimentador e de interação com o sujeito (DELVAL, 2002, p. 67)

Nesse contexto, é dito que o método clínico se difere dos outros métodos na observação direta e na intervenção constante do experimentador (professor pesquisador) diante das atuações do sujeito, ou seja, enquanto os estudantes realizam ações, procuramos analisar o que está acontecendo e atribuir significados para essas ações a partir das constantes intervenções.

> Fixa-se uma série de aspectos da conduta do sujeito e, à medida que este vai se produzindo, realiza intervenções motivadas pela atuação do sujeito, que têm como objetivo esclarecer qual o sentido do que ele está fazendo.

> Isso supõe que o experimentador tenha de se perguntar a cada momento qual o significado da conduta do sujeito e a relação com as suas capacidades mentais. Como geralmente isso não fica claro, trata de realizar intervenções que ajudem a desvendar seu sentido. Isso exige que a intervenção do experimentador seja extremamente flexível e também muito sensível ao que o sujeito está fazendo (DELVAL, 2002, p. 68).

Como pressuposto, o método clínico baseia-se na ideia de que o sujeito possui uma coerente estrutura de pensamento, construindo o mundo a sua volta a partir de representações que são apresentadas por ele ao longo de entrevistas ou de suas ações (DELVAL, 2002). Desta forma, Piaget considera cada criança como um sujeito único, com uma coerência interna, porém, procurando observar esse sujeito com objetivo central em seu universal e não nas suas peculiaridades, verificando nessa criança um produtor de conhecimento. Sendo assim, o objetivo do método se constitui em, a partir de um sujeito individual, observar as características gerais que aquele sujeito pode explicar ou resolver um problema.

Assim, utilizamos o método clínico para inspirar as intervenções do pesquisador, com o objetivo de investigar a conduta dos estudantes perante os objetos em questão, no nosso caso, o computador e as ações de programação. Nesse sentido, julgamos que as raízes do método clínico podem colaborar para a compreensão das contribuições da programação no *software* Scratch no desenvolvimento do pensamento matemático.

Para entender o processo de pensamento do sujeito sobre determinado fenômeno, é necessário produzi-lo e pedir para ele explicá-lo. A partir das explicações do sujeito, que podem ser mais ou menos incompletas, é tarefa do pesquisador esclarecer ao máximo o sentido das razões do sujeito, podendo, inclusive, provocar outras situações que complementem suas explicações, esclarecendo ou contradizendo-as, para que uma melhor compreensão do fenômeno seja obtida.

Desta forma, como pesquisadores, a partir das explicações do sujeito sobre suas ações, devemos nos perguntar por que ele explica dessa forma, quais os significados estabelecidos aos termos utilizados e se os significados são semelhantes aos empregados conceitualmente ou possuem significado diferente. Sendo assim, para análise dos dados produzidos, a cada momento da atividade, buscamos compreender o porquê das explicações do sujeito, tendo o dinamismo na mudança das perguntas ou das situações do experimento em relação à postura e à conduta dada pelo sujeito ou em suas repostas. Assim como Delval (2002, p. 72)

explica, “essa intervenção sistemática do experimentador com hipóteses que vai formulando durante o processo é o que caracteriza o método clínico”.

4.3 Cenário da pesquisa

A produção de dados para essa pesquisa foi realizada no período de abril a julho de 2019 em uma escola particular da cidade de Porto Alegre situada no centro da cidade, que atende por volta de 480 estudantes de 1º a 9º ano do Ensino Fundamental, sendo que anos inicias são atendidos durante o turno da tarde e anos finais no turno da manhã.

A escola possui um espaço virtual, que foi utilizado para a realização da oficina e produção dos dados da pesquisa. O espaço virtual, conhecido na escola como Sala de Informática, é composto de 31 computadores cujo sistema operacional é Windows 7. O espaço pode ser reservado pelos professores para utilização durante o seu período de segunda-feira a quinta-feira, tendo a sexta-feira para uso reservado dos estudantes em turno inverso. É também utilizado pelos estudantes nos projetos realizados pela escola. Conta com dois profissionais que atendem como suporte técnico e monitoria.

Para a realização da pesquisa na escola, previamente contatamos o diretor da escola para que o projeto de pesquisa fosse apresentado e para que fosse entregue a carta de apresentação, conforme Apêndice A. Tendo a proposta aceita pelo diretor, foram organizados os horários para atendimento aos estudantes e a realização da oficina de programação.

4.4 Participantes da pesquisa

Para a pesquisa, foram selecionados oito estudantes, três de 8º ano e cinco de 9º ano, sendo duas meninas e seis meninos, mediante Termo de Consentimento Informado (Apêndice B) e Termo de Assentimento Informado (Apêndice C). Os estudantes foram selecionados pela demonstração de interesse em participar da oficina e de acordo com a disponibilidade de presença no turno inverso do horário habitual das aulas regulares. Todos os participantes da pesquisa têm o pesquisador como professor titular de Matemática.

Os estudantes foram divididos em dois grupos de quatro componentes cada. Os cinco primeiros encontros foram divididos em dois momentos de uma hora cada, com um grupo em cada horário. Essa decisão foi tomada para permitir ao pesquisador acompanhar e intervir cada grupo separadamente, com o objetivo de compreender a conduta e pensamento dos estudantes. Nos cinco últimos encontros, foi organizado apenas um momento com os dois grupos, com duração de duas horas. Observamos que as experiências com ambos os grupos foram muito semelhantes nos cinco encontros iniciais. Na etapa de produção do jogo, o segundo grupo produziu um trabalho muito semelhante com o que foi desenvolvido nos encontros iniciais. Dessa forma, optamos por analisar o desenvolvimento dos estudantes do primeiro grupo.

Os estudantes foram autorizados pelos seus responsáveis a participar da pesquisa mediante um Termo de Consentimento Informado, disponibilizado no Apêndice B.

4.5 Produção de dados

Os encontros foram gravados em áudio a partir de um celular que ficava junto aos estudantes durante o processo investigativo, não havendo necessidade de levar o celular até os estudantes, já que eles ficaram próximos uns aos outros. Outros recursos utilizados para a produção dos dados foram: o caderno de campo, utilizado pelo pesquisador para anotar as situações pertinentes ocorridas; cadernos de anotações dos grupos de estudantes, para que eles pudessem anotar procedimentos, ideias ou rascunhos; fotos produzidas durante o experimento, para registrar imagens das construções dos estudantes nas diversas situações que ocorriam durante o processo; e as produções finais dos estudantes salvas em seus cadastros do site do Scratch.

4.6 Descrição da Oficina e elaboração das atividades: Uma possibilidade introdutória

A oficina foi estruturada em duas etapas: cinco encontros para abordar uma possibilidade inicial de familiarização com o software e com a programação em

blocos, com duração de uma hora para cada grupo, com o objetivo geral de proporcionar aos participantes conhecer e se familiarizar com os blocos do Scratch, assim como apresentar ferramentas e recursos interessantes para a construção de seus projetos na segunda etapa, observando e analisando a construção de conceitos matemáticos durante o processo; e cinco encontros de duas horas, com o objetivo geral de observar e analisar os participantes na programação de seus projetos, investigando as ações dos participantes na tentativa da compreender os conteúdos de seus pensamentos no processo de suas programações. O planejamento e a organização das atividades da oficina foram sustentados pelo referencial teórico dessa pesquisa.

Os objetivos da pesquisa nos encontros iniciais são:

- Questionar e analisar a forma de comunicação entre participante e computador.
- Observar e analisar os conceitos matemáticos desenvolvidos e utilizados durante o processo de programação.
- Analisar os pilares o desenvolvimento e o uso do pensamento computacional durante o processo de programação.

No Quadro 3 a seguir, apresentamos a organização da pesquisa para cada um dos cinco encontros iniciais, indicando os objetivos da pesquisa particulares para cada encontro e os objetivos de cada encontro.

Quadro 3 - Planejamento dos encontros.

	Objetivos da pesquisa	**Objetivos do encontro**
Encontro 1: Apresentação do software Scratch – Comunicação com a máquina: Movimentação do ator no palco.	- Observar as ações dos participantes na tentativa de movimentação do ator no palco. - Analisar as primeiras relações feitas pelos participantes na organização dos comandos para o movimento do ator.	- Apresentar os componentes do software: palco, área de comando e blocos de comando. - Escolher um ator no palco para iniciar a movimentação no palco. - Explicar a Importância dos blocos de “eventos” para se comunicar com o computador. - Usar os blocos “se, então” e

		“sempre” na programação mostrando a sua importância na comunicação com o computador. - Oportunizar a construção do conceito de plano cartesiano durante o processo de movimentação do ator no palco.
Encontro 2: Animação do ator no palco.	- Observar e analisar as ações dos participantes na troca de fantasia dos atores para dar movimento. - Observar a compreensão dos estudantes em relação a organização dos algoritmos para programação envolvendo o tempo.	- Apresentar os recursos de edição de imagem do Scratch para criar as fantasias dos atores. - Usar os blocos de troca de fantasias para animar os atores. - Verificar a necessidade de um intervalo de tempo para possibilitar a visualização das trocas de fantasias e caracterizar a animação. - Utilizar como recurso o comando “espere” para delimitar um tempo para as fantasias trocarem.
Encontro 3: Criação de atores com o comando clone	- Observar e analisar as possíveis generalizações para a resolução de problemas utilizando comandos e algoritmos dos encontros anteriores. - Questionar e analisar as possíveis soluções desenvolver as programações dos clones.	- Utilizar algoritmos já aprendidos para iniciar a programação dos atores, aplicando as modificações necessárias para cada caso. - Utilizar as ferramentas do software para criar múltiplos sprites, e características de um ator a partir do código “crie um clone de”. - Programar os clones utilizando o bloco “quando eu começar como um clone” - Controlar os clones para que desapareçam em um

		certo momento na programação.
Encontro 4: Colisão entre atores.	- Observar e analisar as ações dos estudantes na programação de colisão entre atores. - Questionar as intenções e ações dos estudantes durante a programação dos algoritmos já construídos nos encontros anteriores e suas possíveis adaptações.	- Utilizar os recursos do Software para programar as colisões dos atores. - Criar fantasias para representarem os atores quando colidirem. - Criar a animação para as colisões entre os atores.
Encontro 5: Uso do recurso “variável” no Scratch.	- Questionar e analisar as relações feitas pelos estudantes durante o uso do recurso “variável” no Scratch. - Observar e analisar a compreensão dos estudantes em relação aos blocos do Scracth e de como o computador os interpreta. - Observar e analisar a compreensão dos estudantes ao relacionar os algoritmos com as representações desejadas no palco do *software.*	- Utilizar os recursos de criação de “variável” para inserir nas programações. - Explicar a função desse recurso na programação. - Questionar em quais momentos da programação esse recurso é adequado, utilizando projetos já realizados nos encontros anteriores. - Utilizar os blocos de operações para criar sentenças de verdadeiro ou falso para inserir nos algoritmos. - Utilizar o conceito de valor numérico no uso da programação adicionado ou retirando valores a partir de uma condição.

Fonte: Produção do autor.

A organização dos cinco primeiros encontros (que serão detalhados a seguir) originou o produto didático dessa dissertação, que se encontra no Apêndice D.

Questionamentos iniciais foram elaborados para orientar a intervenção do pesquisador. Contudo, novos questionamentos podem ser formulados ao longo dos

experimentos, a partir da observação do pesquisador, para analisar a conduta dos estudantes e atribuir significados para suas ações, na busca por compreender o pensamento dos participantes a partir de explicações que podem complementar suas ações. Apresentaremos as situações que ocorreram em cada encontro, seguidos dos questionamentos elaborados e uma sequência de possíveis blocos utilizáveis em cada situação, com base na organização e formação das atividades da oficina.

1º Encontro: Apresentação do software Scratch – Comunicação com a máquina: Movimentação do ator no palco.

O primeiro encontro foi elaborado para apresentar o Scratch aos estudantes e trabalhar com os comandos básicos de movimentação de um ator no palco. Nesse encontro, foram apresentadas as ferramentas e funcionalidades do Scratch, com o objetivo de familiarizar os estudantes com o *software*. Ainda, dois conceitos previstos durante a organização da oficina foram trabalhados: o conceito de plano cartesiano a partir dos eixos cartesianos que fazem parte do palco do Scratch e o conceito de condição. A seguir, apresentamos as três etapas do encontro.

Etapa 1 – Comunicação com a máquina.

Movimentar o ator no palco utilizando um ou mais blocos.

Possíveis questionamentos:

Q1 – Como movimentar o ator no palco?

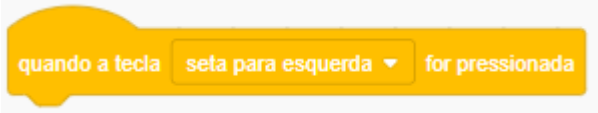
Q2 – O que é necessário para o computador fazer o movimento solicitado?

Q3 – Quais blocos são necessários para que o computador entenda que é necessário fazer o movimento escolhido?

Q4 – Para que servem os blocos de evento?

Possíveis blocos a serem utilizados (mais prováveis de serem escolhidos) e suas funções (Quadro 4):

Quadro 4 - Lista de blocos 1

BLOCO	AÇÃO
	Todo ator inicia como um vetor de movimento que vai da esquerda para a direita. Esse bloco já está programado para seguir esse vetor, podendo mudar de direção apenas quando o ator for girado.
	Blocos que giram o ator no sentido horário ou anti-horário.
	Bloco que comunica o evento para que a programação aconteça. Nesse caso, quando a bandeira verde é clicada.
	Bloco que comunica o evento para que a programação aconteça. Possibilita o uso de teclas para comunicar o evento.

Fonte: Produção do autor.

Etapa 2 – Movimentar o ator horizontalmente utilizando as setas direcionais do teclado.

Solicitar que os estudantes que movimentem o ator horizontalmente utilizando o bloco de evento  .

Possíveis questionamentos:

Q1 – Como podemos movimentar o ator para a direita e para a esquerda utilizando as setas direcionais do teclado com o evento da bandeira verde?

Q2 – Em quais outros blocos os sensores podem se encaixar?

Q3 – O que o computador está entendendo com o algoritmo construído?

Q4 – Como organizar os comandos para que o computador compreenda o que deve ser feito?

Q5 – Há outro possível bloco para que, quando a tecla direita for pressionada, o ator se movimente?

Q6 – Qual a função do bloco sempre?

Q7 – Se o número que está dentro do bloco o faz ir para a direita e vocês querem ir para a esquerda, ou seja, movimento oposto, o que pode ser feito?

Q8 – Qual o comportamento dos números colocados dentro do bloco de movimento?

Q9 – Como fazer para o ator se movimentar mais rápido ou mais lento?

Novos blocos importantes e suas funções nessa etapa do encontro (Quadro 5).

Quadro 5 - Lista de blocos 2

BLOCO	AÇÃO
	Bloco de sensor, definindo a ação utilizada para o comando.
	Bloco de condicional, que comunica a ação que o computador deve fazer a partir de uma condição.
	Utilizado para comunicar o computador que uma ação deve se repetir sempre a partir de um evento.

Fonte: Produção do autor.

Etapa 3 – Movimentar o ator verticalmente utilizando as setas direcionais do teclado para cima e para baixo e outros blocos de movimento.

Movimentar o ator verticalmente utilizando os blocos que adicionam valores aos eixos das abscissas e das ordenadas.

Possíveis questionamentos:

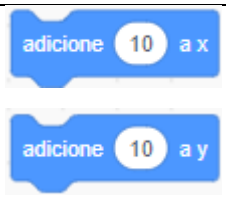
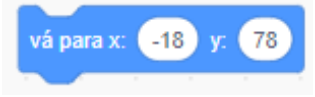
Q1 – Por que o bloco que adiciona valor a x faz o movimento horizontal?

Q2 – Como movimentar o ator verticalmente?

Q3 – O que representam o x e o y no palco?

Possibilidades de novos blocos e suas funções nessa etapa do encontro (Quadro 6).

Quadro 6 - Lista de blocos 3

BLOCO	AÇÃO
	Adiciona valores aos eixos das abscissas e das ordenadas, movimentando o ator na horizontal e na vertical. Pode ser utilizado para a compreensão do plano cartesiano que está vinculado ao palco.
	Movimenta o ator para um ponto específico do palco. Pode ser utilizado para compreender melhor o plano cartesiano que está vinculado ao palco.

Fonte: Produção do autor.

2º Encontro: Animação do ator no palco.

No segundo encontro, o foco foi animar o ator no palco, por meio da criação de fantasias e programação de movimentos no palco. Inicialmente, foi apresentado o recurso de edição de imagem do Scratch. Como as atividades da oficina foram pensadas de forma aberta, duas possibilidades seriam possíveis para esse encontro: utilizar o projeto iniciado no encontro anterior, para dar prosseguimento às programações iniciais, de modo a torna-las cada vez mais elaboradas; dar início a um novo projeto.

- Animação de um personagem.

Animar um ator a partir da troca de fantasias utilizando o comando “espere” para dar o tempo de cada animação.

Possíveis questionamentos:

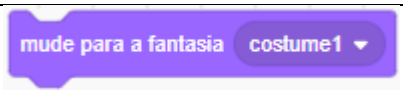
Q1 – Por que utilizar os blocos de troca de fantasia em sequência não anima o ator?

Q2 – O que é necessário para que o computador possa tornar a animação visível?

Q3 – Qual a função do comando espere?

Novos blocos importantes e suas funções nesse encontro (Quadro 7).

Quadro 7 - Lista de blocos 4

BLOCO	AÇÃO
	Comando que troca a fantasia do ator.
	Determina um tempo para que a fantasia troque ou para que o programa mude a fantasia

Fonte: Produção do autor.

3º Encontro: Criação de atores com o comando clone.

O terceiro encontro foi elaborado com o objetivo de utilizar a programação para a criação de clones de um ou mais atores para aparecerem diversas vezes. Nesse encontro, utilizamos blocos de operadores para que os clones possam surgir de lugares aleatórios e com um intervalo de tempo definido ou aleatório. A seguir, apresentamos as duas etapas desse encontro.

- Etapa 1: Criação de clones 1

Criar clones de um personagem para surgirem de uma das bordas, sugerimos da borda direita para a borda esquerda.

Possíveis questionamentos:

Q1 – Como podemos fazer para um mesmo personagem aparecer de forma indeterminada, sem a necessidade de criar mais atores?

Q2 – Qual bloco podemos utilizar para criar o clone de um ator?

Q3 – Como podemos programar apenas os clones e não o ator clonado?

Q4 – Como fazer para determinar um intervalo de tempo para que os clones apareçam.

Q5 – Como podemos apagar os clones que percorreram toda a tela para que o computador não continue processando todos os clones? (Mesmo que o clone saia da tela, ele ainda estará sendo processado)

Q6 – O que podemos fazer para que os clones apareçam de forma aleatória, tanto na posição da borda, quanto no tempo?

Etapa 2: Criação de clones 2

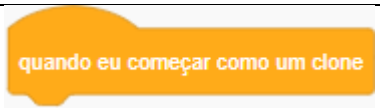
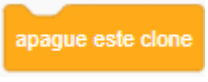
Criar um ataque para um personagem utilizando os mesmos recursos da etapa anterior.

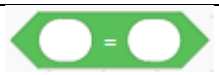
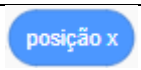
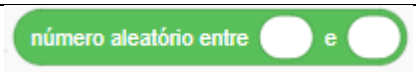
Questionamentos:

Q1 - Quais blocos utilizar para que o computador entenda o momento de criar um clone, ou seja, o momento de atacar?

Novos blocos importantes e suas funções nas duas etapas de criação de clones (Quadro 8).

Quadro 8 - Lista de blocos 5

BLOCO	AÇÃO
	Utilizado na programação do ator para criar clones com as mesmas características.
	Bloco utilizado para iniciar a programação apenas dos clones, não interferindo no ator clonado.
	Apaga o clone a partir de um momento definido.

	Operador de igualdade, utilizado para condicionar uma ação a um valor especifico.
	Pode ser utilizado dentro do bloco de operadores. Utilizado para condicionar a uma ou mais posições específicas.
	Utilizado para escolher um intervalo para que o programa possa escolher aleatoriamente.

Fonte: Produção do autor.

4º Encontro: Colisão entre atores.

O quarto encontro foi destinado para a programação de colisão de *sprites*, criando animações para os contatos e ações posteriores.

- Animação de atores colidindo.

Criar situações para que os atores possam se colidir, como por exemplo, um inimigo “encostando-se” ao personagem principal ou um ataque “atingindo” um inimigo.

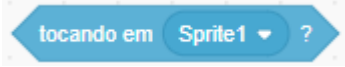
Possíveis questionamentos:

Q1 – Quais blocos podemos utilizar para representar a colisão entre os atores?

Q2 – Como podemos animar esse contato?

Bloco utilizado e sua função para determinar a colisão entre os atores (Quadro 9):

Quadro 9 - Lista de blocos 6

BLOCO	AÇÃO
	Bloco que condiciona a colisão

Fonte: Produção do autor.

5º Encontro: Uso do recurso “variável” no Scratch.

O quinto encontro sugere a utilização do projeto do encontro anterior, para propor situações nas quais será necessária a utilização de variáveis que permitam um melhor controle da programação (por exemplo, criar invulnerabilidade em um dos atores quando ele reaparecer no jogo depois de “morrer”). Outro exemplo a ser trabalhado será um limite de colisões para o ator, representando as chances que o ator tem para colidir, podendo ser trabalhado como as “vidas” do ator. A seguir, apresentamos esses exemplos separados em etapas.

- Etapa 1: Invulnerabilidade do ator.

Em um jogo, quando o personagem principal é atingido ele recebe um tempo para que fique imune a ataques. Para programar essa situação, utiliza-se a ferramenta variável para criar sentenças de verdadeiro e falso para que o algoritmo seja utilizado.

Possíveis questionamentos:

Q1 – Quando o ator fica invulnerável, quais ações ou situações que não podem ocorrer?

Q2 – Como podemos estabelecer condições para que o computador utilize ou não um determinado algoritmo?

Q3 – Onde, no algoritmo, devemos incluir as condições com a variável?

Q4 – Para que um algoritmo seja executado, a sentença determinada pela variável deve ser verdadeira ou falsa?

Q5 – Como podemos representar o personagem para evidenciar que ele está invulnerável?

Etapa 2: Contagem de vidas.

Criar uma variável para iniciar o número de vidas do ator. Programar uma situação para que ele perca as vidas até um limite.

Questionamentos:

Q1 – Como podemos inserir um valor para a variável criada?

Q2 – Como inserimos essa variável na programação?

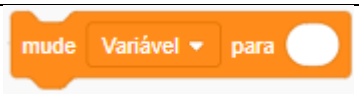
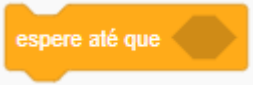
Q3 – Em qual momento a variável deve diminuir o número de vidas?

Q4 – Como dar um limite para que o número de vidas não seja negativo?

Q5 – O que pode ser programado para evidenciar o final do jogo?

Novos blocos importantes e suas funções nas duas etapas do encontro (Quadro 10).

Quadro 10 - Lista de blocos 7

BLOCO	AÇÃO
	Bloco variável para inserir em outros blocos compatíveis.
	Define a sentença, que pode ser verdadeira ou falsa, ou até mesmo mudar a variável para algum número.
	Utilizado para que o algoritmo inicie a partir de uma condição específica na programação.
	Para todas as animações e trava os algoritmos.

Fonte: Produção do autor.

Com esse último encontro, finalizamos as atividades planejadas para a primeira etapa da oficina. A partir desse encontro, iniciamos a segunda etapa da oficina, em que os estudantes planejam e programam os seus projetos.

O sexto encontro foi planejado para projetar os trabalhos finais. Nesse encontro os estudantes trabalharam na sala interdisciplinar da escola, em que foi possível utilizar as mesas para trabalhos em grupos, possibilitando um melhor ambiente para pensarem em seus projetos. Foram fornecidos materiais para que os estudantes pudessem esboçar seus projetos com a possibilidade do uso de tecnologia digital para pesquisar *sprites* para o jogo. Os estudantes, nesse encontro, deveriam apresentar suas ideias contendo título, cenário de abertura e organização

de etapas para o jogo, assim como personagens. A expectativa foi de que nesse encontro os estudantes elaborassem todas as ideias iniciais para projetar seus jogos, com possibilidades de adaptação das ideias durante a execução do projeto.

Os encontros sete, oito, nove e dez (o último sendo um encontro extra) foram planejados para a execução do projeto, ou seja, a programação do jogo no Scratch. Nesses encontros, procuramos acompanhar e identificar a conduta dos estudantes no processo de programação, observando as principais dificuldades e desafios enfrentados, como a correção de *bugs* possíveis, observando os conceitos de Matemática que apareceriam durante o processo junto às estratégias utilizadas por eles para compreendê-los e observando elementos de pensamento computacional evidenciados ao longo dos encontros de programação. Ao finalizar o projeto, pretendíamos que os estudantes apresentassem o seu jogo, mostrando a programação envolvida, assim como as dificuldades e estratégias para o fechamento do projeto.

No capítulo a seguir, apresentamos a descrição da oficina e as análises.

5 DESCRIÇÃO E ANÁLISE

Neste capítulo, serão apresentadas as análises dos dados. A descrição e análise estão organizadas em episódios ocorridos durantes os encontros do experimento, selecionados a partir de critérios que indicam maior riqueza de dados com possibilidades para análise e peculiaridades que favorecem e enriquecem o desenvolvimento da pesquisa. As identidades dos estudantes foram preservadas, sendo utilizadas letras maiúsculas das iniciais dos seus nomes ou sobrenomes para identificá-los.

A seguir, os relatos dos episódios e as análises dos dados.

5.1. Comunicação com a Máquina: Primeiras Ações

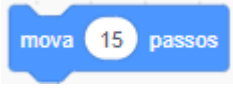
Para compreender os processos ocorridos nos primeiros momentos da oficina, escolhemos analisar as ações de uma das duplas do primeiro grupo de estudantes. Os estudantes desse grupo organizaram-se em duplas, uma composta pelos estudantes I e S, e a outra pelos estudantes A e M. A oficina tem início com a solicitação do cadastro dos estudantes no site do Scratch.

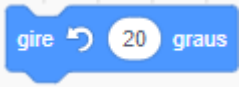
A interface do *software* foi apresentada aos estudantes, com o objetivo de explorar a janela de comandos, a janela de programação e o palco. Discutiu-se sobre a troca de ator e a mudança de palco, para ampliar as possibilidades de utilização do Scratch durante as atividades propostas e na criação de seus procedimentos. Um momento inicial de exploração e familiarização ocorreu, no qual os estudantes escolheram atores e palco para iniciar a exploração dos blocos de comandos do *software*.

Para provocar a primeira ação de programação, foi solicitado às duplas que movimentassem os seus respectivos atores no palco (ressaltamos que esses estudantes não tinham experiência com programação). O objetivo inicial era a experimentação de blocos para programar ações para o personagem. Para isso, oportunizou-se uma discussão com os estudantes sobre a comunicação entre programador e máquina, a partir de ordens precisas, detalhadas e organizadas, de forma que o programador fornecesse instruções corretas para o computador. Segundo Papert (1994, p. 109), “Os computadores, de fato, fazem o que se manda

que façam. Entretanto, quando lhe mandamos fazer, nem sempre lhes estamos mandando fazer o que pensamos. Nem o que eles fazem é sempre o que parece ser". A categoria Evento foi destacada e explorada para que a relação entre programação e execução fosse estabelecida.

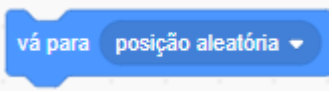
A dupla escolhida para análise foi a dupla A e M, observando os processos de exploração e ambientação dos recursos do software. A proposta era movimentar o ator no palco. Como hipótese inicial da dupla para o movimento do ator, o bloco que caracteriza o evento 

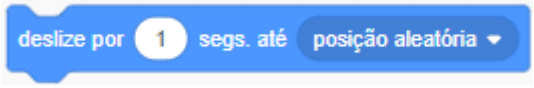
é encaixado ao bloco 

, o que fez o ator mover-se para a direita. Em uma ação experimental e exploratória para fazer o ator retornar, o bloco 

foi encaixado com os demais e fez com que ocorresse um movimento de looping na tela enquanto a tecla espaço estava sendo pressionada, situação na qual A e M não conseguiram explicar. Para esclarecer a compreensão dos estudantes sobre a situação que se apresentou, o pesquisador interviu solicitando uma explicação que complementasse suas ações:

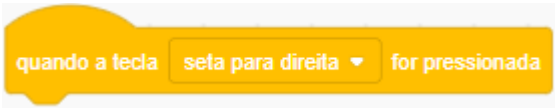
> Pesquisador: Por que o ator faz esse movimento?
>
> Estudante A: Porque eu não sei, acho que é o "gire 20 graus", mas é um *looping.* Mas depois desse movimento, como eu faço para ele fazer outro comando?
>
> Estudante M: Mova para frente.
>
> Estudante A: Entendi! Daí eu pego outro evento, que é quando colocamos outros blocos.

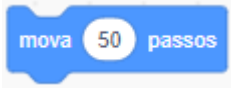
O diálogo revela que A identifica que o bloco "gire 20 graus" é responsável pelo *looping*, mesmo sem compreender as razões para esse comportamento. A partir da sugestão de M, a dupla experimenta os blocos 

e 

, para observarem as ações do ator na tela do computador. Observa-se nesse momento um processo de ação pura, na qual os

blocos são testados para um determinado fim, movimentar o ator, mesmo que esse fim seja uma forma provisória de resolver o problema e que ainda não seja totalmente compreendido pelos estudantes A e M. Para Piaget (1978), as ações possibilitam o erro e a análise do erro, que antecedem as tomadas de consciência, compreendendo em ação a situação, caracterizada pelo fazer.

Por fim, a dupla recorre ao bloco 

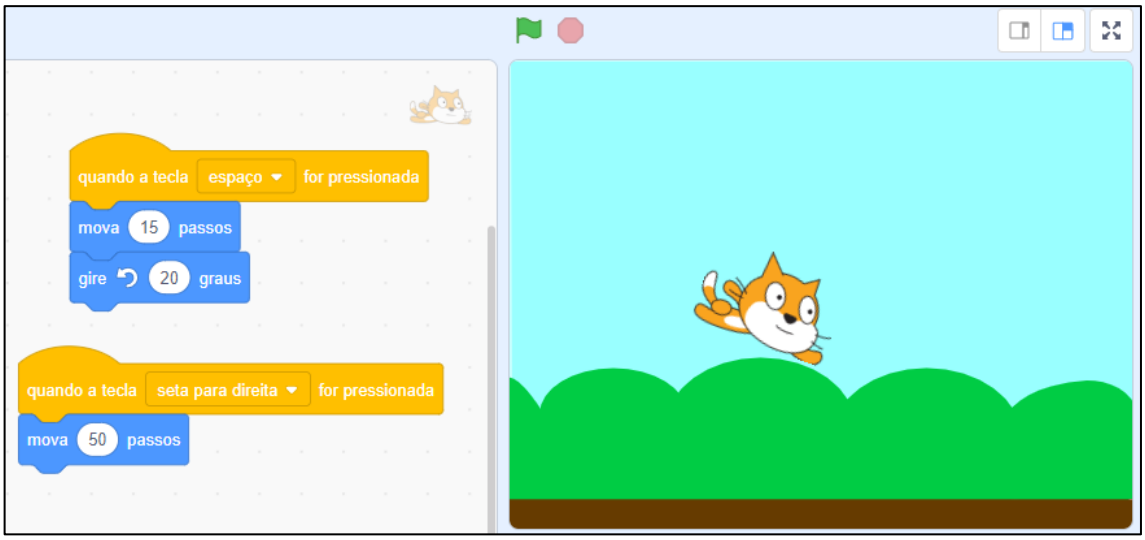
como segundo evento, encaixando o bloco 

para movimentar o ator para frente, movimento esse já explorado anteriormente. Para complementar os dados para análise e compreender o pensamento dos estudantes, o pesquisador questiona:

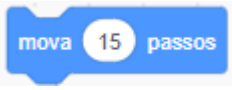
> Pesquisador: Qual é o movimento do ator?
>
> Estudante A: Com a seta direita ele pode ir para frente, quando ele chega na borda da tela (palco) pressionamos espaço para ele fazer a volta.

A Figura 9 mostra a programação inicial de movimento do ator no palco feita pela dupla A e M.

Figura 9 - Códigos utilizados para a movimentação do ator no palco.



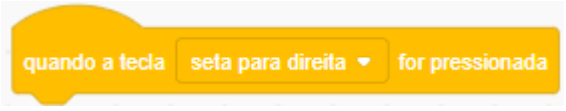
Fonte: Acervo do autor.

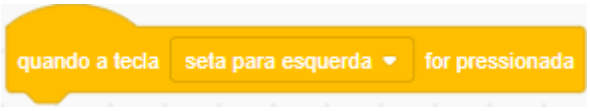
Nessa experiência, percebemos que, inicialmente os estudantes estavam em uma etapa de exploração dos recursos e funcionamento do Scratch. Ao encaixar o bloco 

, A e M perceberam que o ator se deslocava apenas para a direita. Assim, os estudantes depararam-se com um novo problema: como fazer o ator voltar, sem precisar arrastá-lo com o mouse? Na tentativa de fazê-lo voltar, testaram o bloco 

e perceberam que o ator se movimentava em uma trajetória que lembra a forma de círculos ao pressionar a tecla espaço continuamente. Percebemos as tentativas utilizadas para resolver essa situação, que levou a dupla a abstrair as características do movimento do ator para elaborar um algoritmo que o faça movimentar, mesmo que não seja da forma desejada inicialmente. Assim, a dupla concluiu que precisa de um novo evento para que o ator possa se movimentar somente para frente, percorrendo palco em movimentos que avançam e retornam. De acordo com Piaget (1978), os estudantes iniciam o processo de ação e exploração, promovendo soluções provisórias para cumprir o desafio de movimentar o ator, porém em um patamar suficiente para alcançar tal propósito, mesmo que ainda não seja totalmente compreendido por eles.

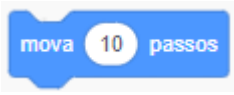
5.2 Movimento dos Atores no Palco: Atribuindo Movimento Consciente

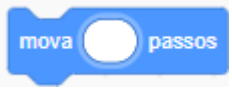
Como um novo desafio, foi solicitado às duplas que movimentassem o ator em diferentes sentidos no palco. O objetivo dessa atividade exigia dos estudantes a ideia de localização no palco, envolvendo o conceito de coordenada cartesiana, compreendendo os movimentos do ator no palco de forma consciente e controlada. Como ações iniciais da segunda dupla, identificados por I e S, são escolhidos os blocos de comandos 

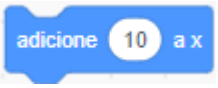
e 

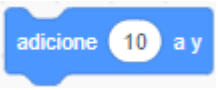
para os movimentos de deslocamento para a direita e para a esquerda. Destaca-se aqui que os estudantes já identificaram um padrão para que o computador realize as funções solicitadas e, dessa forma, as

ações utilizadas nesse momento foram previamente pensadas, tomando consciência da necessidade de um evento para que a programação ocorra e antecipando suas ações (PIAGET, 1978).

Em ambos os eventos, os estudantes adicionaram o comando 

. Porém, no momento de execução, ao pressionar, tanto seta esquerda, como seta direita, o ator movimentava-se sempre para a mesma direção, o que surpreendeu os estudantes e provocou um desequilíbrio momentâneo. Isso porque os estudantes acreditavam que a tecla “seta direita” movimentaria o ator para a direita e que a tecla “seta esquerda” movimentaria o ator para esquerda. Observando as ações de I e S, o estudante A questiona se o número digitado dentro do bloco de 

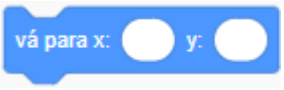
não deveria ser negativo, resgatando seus conhecimentos prévios sobre números inteiros e aplicando nessa situação específica, utilizando um conhecimento-em-uso, no sentido discutido por Papert (1994, p. 96) no qual afirma que “parte deliberada da aprendizagem consiste em fazer conexões entre entidades mentais já existentes”. O estudante S observou os blocos 

e 

e comentou: “Ah, entendi! O x é o comprimento e o y é a altura”, apontando para o seguinte espaço da tela (Figura 10). Nesse momento, a ideia de sistema de coordenadas cartesianas (conceito ainda desconhecido) está sendo construída por S, a partir de ações e experimentações no ambiente de programação. Para Papert (1994), essas podem ser situações de aprender-em-uso, que liberam os estudantes para aprender de uma forma pessoal, recebendo orientações ocasionais.

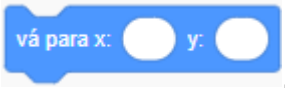
Figura 10 - Informações na interface do Scratch.

Fonte: Acervo do autor.

O estudante I tenta explicar para os demais que, no bloco 

o y é a altura e o x é o comprimento, mostrando na tela a posição em que o ator estaria. Para investigar a compreensão dos estudantes sobre os papéis de x e y nos comandos utilizados, o diálogo a seguir se estabelece a partir de intervenções do pesquisador:

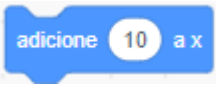
> Pesquisador: Então, o que vocês acham que representam o x e o y?
> Estudante A: É uma espécie de coordenada.
> Pesquisador: Isso! E o que quer dizer esse bloco 'adicione 10 a x'?
> Estudante A: Adicione 10 passos da posição que ele está.
> O pesquisador pede, então, para que o estudante A utilize o comando.

Na tentativa de movimentar o ator e explorando os blocos de comandos, o estudante S descobre que também poderia utilizar o comando 

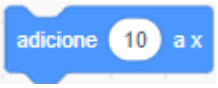
, e assim o faz, arrastando os blocos para a área de programação. Ao executá-lo, os estudantes perceberam que estavam realizando um comando que estabelecia um local fixo para o ator cada vez que eles pressionavam a tecla programada, ou seja, o ator movimentava-se para o local de coordenadas (x, y) na primeira vez que a tecla era pressionada. Porém, nas tentativas seguintes, o ator continuava na mesma posição (x, y) e os estudantes precisaram analisar a situação, compreender o significado do comando utilizado, tanto do ponto de vista da programação, quanto do ponto de vista do conceito matemático inerente a ele, para identificar o erro e solucioná-lo, tornando o processo operatório.

Explorando as possibilidades de comandos e a movimentação do ator no palco (novamente nível da ação), os estudantes descobriram que o ator tem uma posição no palco que é definida por x e por y, sendo que cada lugar do palco tem uma posição x e y diferente e específica. O conceito de posição no plano cartesiano está sendo explorado e, aos poucos, construído em ação. Nesse momento, os estudantes puderam, a partir da programação e análise dos resultados, compreender a ideia de coordenadas cartesianas e par ordenado como um par de

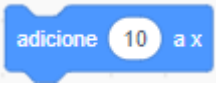
números ordenados que localizam um ponto no plano. Mesmo que os estudantes não tenham conseguido atingir o objetivo inicial (dar movimento contínuo para o ator no palco), descobriram uma nova possibilidade para movimentar o ator para uma posição definida do palco, a partir da compreensão do papel de suas coordenadas x e y.

Na sequência, os estudantes experimentaram o comando 

relacionado ao evento “quando a tecla seta esquerda for pressionada”.

Na exploração do comando 

, foi possível observar as tomadas de consciência dos estudantes sobre a distribuição dos números inteiros no eixo das abscissas, pois analisaram as possibilidades de substituir valores em x, tanto positivos, como negativos e descobriram os respectivos resultados, como destacado no diálogo em que o pesquisador pergunta o que acontece ao adicionar 10 a x e a estudante A responde que o movimento terá uma adição de 10 referente ao x atual.

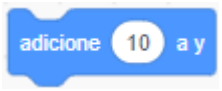
A partir da fala de A, evidenciamos a compreensão do movimento que ocorre com a utilização do bloco. Dessa forma, os estudantes acrescentaram o comando 

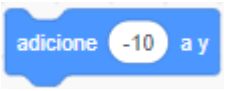
junto ao evento “quando a tecla seta esquerda for pressionada” e executaram o programa. O pesquisador questiona:

> Pesquisador: Então, clicando na seta para a esquerda, estão vendo que ele está indo para a direita?
>
> Estudante A: Sim, pois toda vez que ele vai (que a tecla é pressionada) o x aumenta 10.
>
> Pesquisador: Ok, mas vocês querem que ele vá para o outro lado, então o que vocês terão que pensar?
>
> Estudante M: Menos!

Nesse momento, os estudantes percebem que o valor atribuído ao comando representa o quanto deve ser adicionado à coordenada x para o ator andar para a direita e o quanto deve ser retirado da coordenada x para que ele vá para a esquerda, proporcionando a construção da distribuição dos números inteiros no eixo

das abscissas. Analisando com mais profundidade, o uso do número negativo para que o ator percorra uma distância, mesmo que seja para trás, traz implícito o conceito de módulo (ou valor absoluto) de um número, já que a distância do ponto inicial ao ponto final do ator representa um valor positivo, ou seja, o seu valor absoluto.

Dessa forma, o palco pode ser conceituado como um plano com coordenadas, observando que x representa o movimento do ator para a direita quando adicionado um valor e para a esquerda quando subtraído um valor. Ao serem questionados sobre o movimento vertical do ator, eles descobriram que devem usar o comando 

e 

, podendo assim também movimentar o ator para cima e para baixo (Figura 11). O entendimento do deslocamento vertical ocorreu de forma mais imediata, pois a ideia de movimento no palco já estava construída e foi abstraída para ser utilizada nessa nova situação, generalizando a ideia de adição e subtração, agora para subir e descer.

Figura 11 - Programação para movimentar o ator.

Fonte: Acervo do autor.

Destaca-se a mudança das ações dos estudantes na programação do movimento horizontal do ator, passando para um processo de conceituação, que

mesmo ainda apoiado na ação de testar como o ator iria para cima ou para baixo, já estava se consolidando. A partir do momento em que os estudantes compreenderam o movimento vertical do ator, podemos concluir que, a partir dos esquemas de ações utilizados nos movimentos, e que evoluíram para um processo de conceituação, os participantes compreenderam a movimentação total do ator no palco, sabendo antecipadamente como fazer para que o ator se movimente para a direita, para a esquerda, para cima e para baixo, não havendo mais a necessidade de testar blocos. Piaget (1978) afirma que há uma fase em que a ação e a conceituação têm relevância similares, efetuando-se trocas constantes entre elas. Posteriormente, a ação recebe um reforço da conceituação, possibilitando uma previsão do que deverá ser feito em determinada situação. Decorrente disso, desencadeou-se um processo de compreensão de conceitos matemáticos como plano cartesiano, eixos de coordenadas, par ordenado e valor absoluto.

Podemos perceber nesse processo de tentativa e erro para movimentar o ator no palco que os estudantes tiveram uma conduta apoiada em ações para verificar os movimentos do ator a partir dos comandos selecionados, sem saber quais seriam os resultados de suas ações. A partir dessas ações, os sujeitos puderam compreender como cada bloco influenciaria no movimento do ator. A partir de então, já não era mais necessário testar blocos para saber como o ator poderia se movimentar na horizontal ou na vertical. Logo, os estudantes passaram a um processo de compreensão e um domínio do problema de movimentação no palco, chegando ao nível da conceituação do que foi feito, transformando esquemas de ações em operações. Sendo assim, podemos caracterizar os conceitos representados pelos estudantes como uma tomada de consciência, passando do processo de ação pura para uma conceituação (PIAGET, 1978).

Dando continuidade ao trabalho, notou-se que os estudantes ficaram limitados em utilizar um evento para cada situação. Então, para finalizar o encontro, foi apresentado um recurso que possibilita utilizar apenas um evento - o comando condicional se/então. A utilização do conceito de condicional é importante na linguagem de programação. O conceito matemático de condicional pode ser muito bem explorado no Scratch, já que é um conceito que auxilia na tomada de decisões durante a programação, ou seja, possibilita a escolha de um grupo de ações e estruturas a serem executadas quando determinadas condições são ou não satisfeitas.

Ao testar o procedimento elaborado com o bloco 

, os estudantes perceberam que os comandos não eram executados quando a tecla espaço era pressionada após clicarmos na bandeirinha verde. Assim, os estudantes passaram a analisar o procedimento para identificar o problema, e identificaram que os comandos eram realizados se a tecla espaço fosse pressionada enquanto a bandeira verde fosse clicada, o que indicou a necessidade de um novo bloco para ser explorado: o comando 

. Os estudantes criaram, então, um novo algoritmo (Figura 12), que diz “Olá!” sempre que a tecla espaço é pressionada depois da bandeira verde ser clicada.

Figura 12 - Algoritmo utilizando o comando sempre.

Fonte: Acervo do autor.

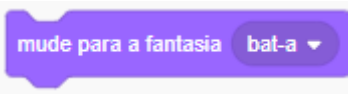
Registramos que, nesse momento, foi necessária maior intervenção do pesquisador para auxiliar os participantes na compreensão do bloco 

, caracterizando um momento de orientação ocasional, conforme apontado por Papert (1994). O algoritmo criado no final do encontro teve o objetivo de disponibilizar uma nova ferramenta para as programações futuras (o bloco sempre), respeitando as diferentes circunstâncias em que esse comando será utilizado, assim como as adaptações necessárias para resolver um futuro problema.

5.3. Animação dos Atores: Troca de Fantasias e Diálogos

Para compreender esse episódio, serão analisadas as ações da dupla I e S, cujo objetivo era movimentar um morcego no palco, observando a exploração de novos blocos para a programação. Utilizando o evento 

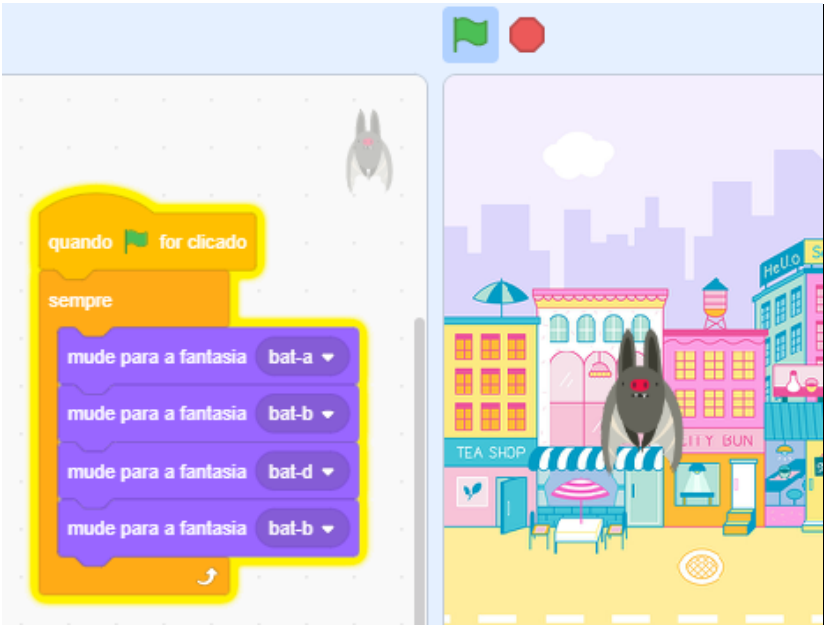
, seguido do comando 

, lembrando das discussões do episódio anterior e identificando um possível padrão para iniciar o algoritmo, os estudantes identificaram na categoria Aparência do bloco 

, utilizando-o dentro do comando 

. A proposta da dupla era que o movimento do morcego tivesse quatro fantasias para completar um ciclo de movimentos de bater as asas, repetindo-se posteriormente o mesmo ciclo. Na Figura 13, apresentamos o algoritmo inicial elaborado pela dupla para a animação do personagem.

Figura 13 - Tentativa de troca de fantasia do ator.



Fonte: Acervo do autor.

Após clicar em bandeirinha verde, os estudantes perceberam que nada acontecia com a animação planejada. Nesse momento, foi necessário discutir com os estudantes que a mudança de fantasia estava acontecendo, porém, na velocidade de processamento do computador, imperceptível ao nosso olhar. O diálogo a seguir se estabeleceu, para tentar compreender a situação:

> Estudante I: Por que ele não está se movendo, sor?
>
> Pesquisador: Por que vocês acham que isso está acontecendo?
>
> Estudante I: Não sei, ele não faz o que eu mando.
>
> Pesquisador: Mas a programação está feita, ele está fazendo a troca de fantasias, mas por que não podemos ver?
>
> Estudante S: Por que ele faz muito rápido (as trocas de fantasias) temos que dar um tempo para ele fazer as trocas.
>
> Pesquisador: Isso aí, computador é muito rápido e ele faz as trocas mas não conseguimos ver.

Papert (1994) relata uma experiência em que crianças, utilizando a linguagem LOGO, pretendiam programar uma luz piscando. Inicialmente, utilizando o comando repita seguido de dez vezes, observaram a luz piscar apenas uma vez. Na tentativa de aumentar o número de vezes, utilizando o comando REPITA 1000, observaram que o mesmo ocorria. Vendo o computador fazendo a mesma ação se "confundindo" em ambas situações, se questionam do porquê, chegando à conclusão da rapidez do computador. Papert (1994, p. 110) denomina esse processo como "originar *insight* sugerindo perguntas frutíferas".

Assim, os estudantes reconheceram a necessidade de incluir um tempo determinado entre uma fantasia e outra, para que fosse possível observar a animação da movimentação do personagem. A dupla pesquisou e identificou na lista de comandos o bloco 

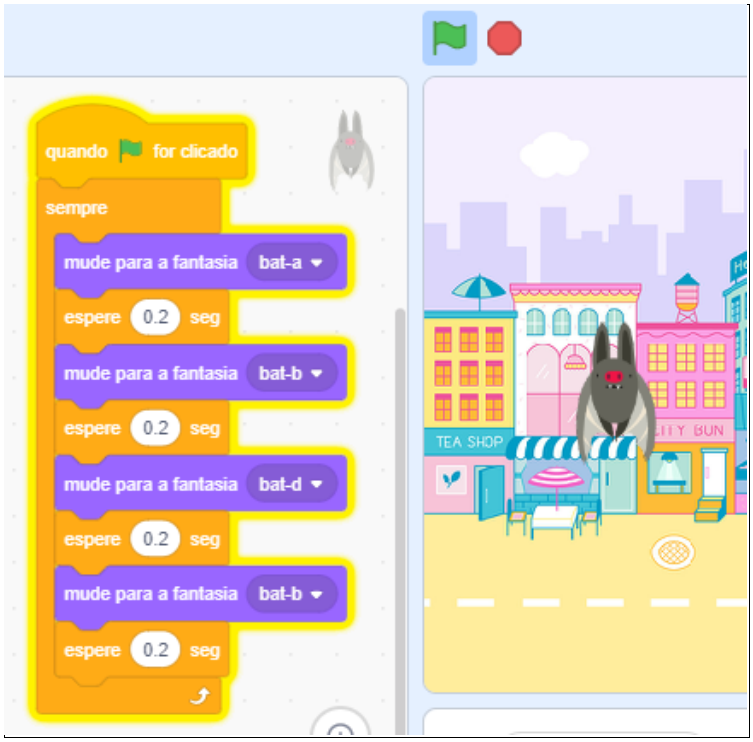
e o utilizou entre uma fantasia e outra. Na primeira tentativa, os estudantes colocaram o comando 

apenas entre as duas primeiras fantasias, utilizando 1 segundo de intervalo entre as trocas de fantasia, revelando ainda falta de compreensão prévia sobre o fluxo do algoritmo. Ao

executar o procedimento, imediatamente perceberam que a animação não estava completa e que deveriam encaixar o comando 

entre todos os blocos de troca de fantasias. Além disso, a escolha inicial de 1 segundo de espera deixou o movimento muito lento e não correspondia à animação esperada. Após algumas experimentações para definir um tempo adequado, a dupla chegou ao algoritmo de programação ilustrado na Figura 14 e à animação de bater asas do morcego esperada. Fazer o morcego bater asas pode parecer uma tarefa de programação simples, mas os estudantes estavam engajados em obter o resultado desejado e alcançar esse resultado promoveu uma sensação de euforia, sendo possível observar as expressões de alegria e entusiasmo ao ver o morcego batendo asas do jeito que havia sido planejado. Para Papert (1985, p. 18) a criança, ao concluir uma determinada tarefa com sucesso, “[...] adquire um sentimento de domínio sobre um dos mais modernos e poderosos equipamentos tecnológicos e estabelece um contato íntimo com algumas das ideias mais profundas da ciência, da matemática e da arte de construir modelos intelectuais”.

Figura 14 - Algoritmo para animação do ator utilizando o comando "espere".



Fonte: Acervo do autor.

Como os estudantes não sabiam de antemão o que aconteceria ao executar o procedimento, tiveram que partir para as ações de experimentações, testes, erros, reflexões, novos testes, até obter o resultado desejado. Segundo Piaget (1978), compreender em ações faz parte do fazer, em um grau suficiente para atingir um propósito. Dessa forma, ações podem ser repetidas até que o sujeito consiga dominar, em pensamento, as mesmas situações, passando a compreender o processo.

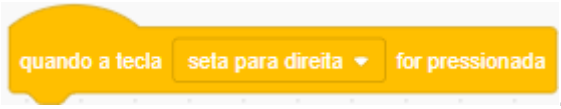
Verificamos nesse processo que o uso do padrão inicial para a programação foi importante para desencadear uma sequência de ações que foram responsáveis para que os estudantes conseguissem chegar a um fim, visto que durante o processo de encaixar o comando “espere” entre as mudanças de fantasia foi se conceituando como um processo necessário para que animação fosse possível, ou seja, os participantes tomaram consciência de que, para conseguir executar a animação desejada para um algoritmo que desempenhe essa função, é necessária a utilização do comando “espere” entre todos os blocos de troca de fantasia, trazendo noções iniciais sobre controle de fluxo de algoritmos (elemento do pensamento computacional apontado por Grover e Pea (2013)). Consideramos que os estudantes, a partir das ações de programar a animação no primeiro momento, em que a troca de fantasias acontecia na velocidade do processador do computador, perceberam a necessidade de outro fator para resolverem o problema.

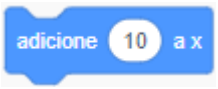
Observar a tentativa de encaixar o comando 

entre os blocos das fantasias nos mostra as ações para chegar ao êxito como um saber fazer, proporcionando uma reflexão durante as ações, transformando os esquemas das ações em noções conscientes, proporcionando um pensamento a partir da forma prática de conhecimento (PIAGET, 1978).

A noção de tempo também foi uma construção interessante: 1 segundo pode parecer demasiadamente rápido, mas para animar o movimento de asas do morcego, mostrou-se lento. Utilizar 0.2 segundos possibilita que os estudantes pensem no tempo de forma mais fragmentada do que o usual, observando que o tempo pode ser representado de forma decimal, ampliando as possibilidades de aprendizado desse conceito aplicado na prática de programar uma animação. Sobre a velocidade de processamento do computador, a estranheza inicial dos estudantes em verificar que, mesmo utilizando os blocos de troca de fantasia, a animação se

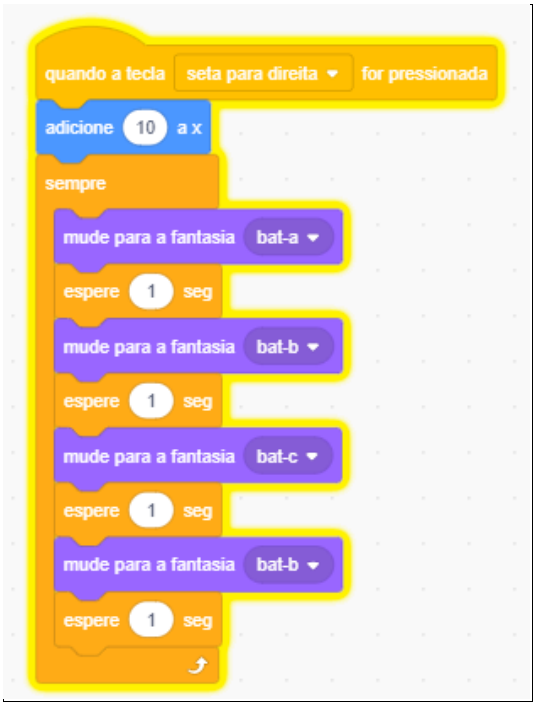
mostra congelada, fornece uma ideia desmesurada do conceito de fragmentação do tempo.

No desafio seguinte, a atividade tinha como proposta animar o personagem durante a movimentação no palco, ou seja, juntar as ideias trabalhadas nos procedimentos anteriores em um único procedimento. Como o objetivo era movimentar o personagem (novamente o morcego), a dupla utilizou o evento 

, juntamente com o bloco 

. Destacamos que, dessa vez, os estudantes sabiam de antemão exatamente quais blocos encaixar para que o personagem se movimentasse para a direita, o que fortalece a hipótese de conceituação do primeiro encontro e que os conceitos de plano cartesiano e par ordenado foram compreendidos pelos sujeitos, tornando-se operatório. Nesse sentido, a conceituação de coordenadas no sistema de coordenadas cartesianas ultrapassa a ação, possibilitando uma antecipação dos comandos sem a necessidade de testes, evidenciando a capacidade de previsão e de planejamento da ação (PIAGET, 1978). A Figura 15 mostra o algoritmo da dupla.

Figura 15 - Tentativa de programação para animar o ator em paralelo com o movimento.



Fonte: Acervo do autor.

No momento de executar o procedimento, ao pressionarem o botão seta direita, os estudantes foram surpreendidos com o seguinte resultado: o morcego

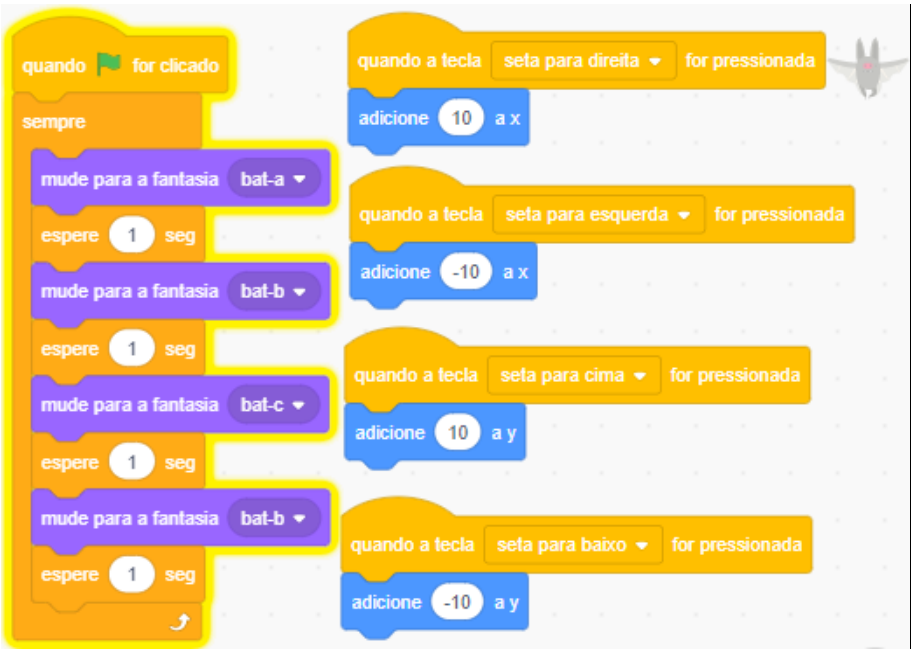
movimentou-se dez passos para a direita e, posteriormente, fez os movimentos de animação, não se movendo mais para a direita, e evidentemente não era esse o resultado esperado. Questionados sobre o porquê o personagem teve aquele comportamento, a dupla imediatamente identificou que a sequência de comandos ordenava que o personagem se movimentasse 10 passos para direita apenas uma vez e, em seguida, animasse a batida de asas com a troca de fantasias, não necessitando mais ir para a direita para fazer o movimento. Os estudantes testaram e observaram também que, retirado o bloco 

do procedimento, cada vez que a tecla seta direita fosse pressionada, o personagem realizava a sequência de movimentos até o fim, tendo que esperar o movimento completo descrito na sequência para que executasse novamente.

Depois de algumas experimentações, a dupla decidiu utilizar novamente o evento 

, utilizando os algoritmos do encontro anterior para movimentar o morcego, apresentando um evento para cada direção de movimento (Figura 16). Dessa forma, a dupla optou por decompor a programação em duas partes (eventos) independentes (uma para animar o bater de asas e outra para movimentar o ator no palco) e ficou satisfeita com o resultado. Podemos observar o uso criativo dos estudantes para resolver o problema, revelando a habilidade de decomposição. Apesar de não utilizarem um algoritmo apenas para representar o movimento do morcego, percebemos que os estudantes encontraram a forma deles resolverem o problema, retomando a afirmação de Resnick (2007), que salienta a importância de ser criativo para encontrar soluções para os problemas repentinos.

Figura 16 - Algoritmos com evento paralelos de animação e movimento.



Fonte: Acervo do autor.

Consideramos dois pontos interessantes a se destacar: o primeiro é o uso do comando 

na programação. É possível compreendê-lo a partir da alteração do número de vezes que se pretende repetir as ações (nível da ação), até que houve uma compreensão de seu conceito, considerando um recurso importante a ser explorado em atividades futuras. O segundo ponto refere-se ao número de algoritmos, que poderiam ser generalizados, de modo a se obter apenas um algoritmo para um único propósito. Esse aspecto foi explorado em encontros futuros.

Percebemos que o êxito ao completar as tarefas foi, por vezes, a partir das ações de exploração, tentativa e erro, observando problemas a serem solucionados. A partir de ações que pudessem colaborar para a realização das tarefas, os estudantes puderam reconhecer e generalizar alguns conceitos matemáticos (e de programação) que foram compreendidos durante o processo. Mas também percebemos que conceitos já aprendidos foram resgatados, como o resgate dos algoritmos de movimento do ator no palco. Piaget (1995) explica que, na abstração reflexionante, o início de uma generalização vem de uma repetição de uma solução encontrada, sendo a generalização também o resultado de uma abstração reflexionante, ou seja, a retomada de conceitos como o resgate de algoritmos

possibilita a abstração dos conceitos envolvidos para que uma generalização seja construída como resultado do processo.

5.4. Criação de clones: Atacando Inimigos

Para a criação de clones[13], dois momentos foram planejados e propostos aos participantes para abordar essa temática:

- Primeiro momento: escolher um personagem para realizar um “ataque” via arremesso de um objeto ou via “poder” para atacar os personagens inimigos. Para isso, os estudantes deveriam criar um segundo personagem para representar esse ataque, que seria acionado ao pressionar uma tecla.

- Segundo momento: criar inimigos no jogo, aparecendo com aleatoriedade no palco e em intervalos de tempo também aleatórios, dentro de um período de tempo determinado. Como objetivo, os estudantes deveriam explorar os blocos “quando eu começar como um clone”, “crie clone de (selecionar ator)” e “apague este clone”.

Vamos analisar o processo de criação do segundo grupo, composto pelos estudantes A, D, V e J, para representar o primeiro momento. O objetivo dos participantes era programar um pinguim que joga bolas de neve. Destacamos a importância da liberdade de escolha do que os estudantes querem programar, para proporcionar maior engajamento. Maltempi (2005) destaca a relevância do significado para o estudante no Construcionismo:

> O Construcionismo postula que o aprendizado ocorre especialmente quando o aprendiz está engajado em construir um produto de significado pessoal (por exemplo, um poema, uma maquete ou um website), que possa ser mostrado a outras pessoas. Portanto, ao conceito de que se aprende melhor fazendo, o Construcionismo acrescenta: aprende-se melhor ainda quando se gosta, pensa e conversa sobre o que se faz (MALTEMPI, 2005, p. 3).

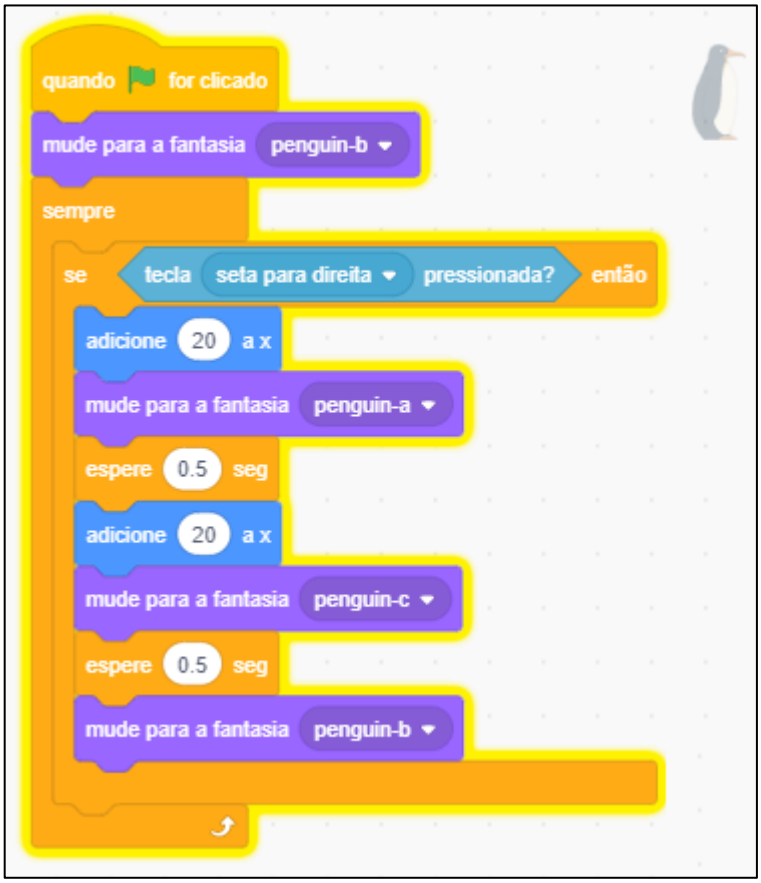
O primeiro passo foi a programação do movimento do pinguim. Os estudantes decidiram que o movimento desse ator seria apenas na horizontal, para a esquerda

[13] No *software* Scratch, a criação de clones possibilita criar vários atores com as mesmas características, ou seja, mesma fantasia, comportamento e atributos. Basta criar clones de um ator específico e posteriormente programa-los.

e para a direita, trocando de fantasias para simular que o pinguim dava dois passos a cada movimento a partir do acionamento das setas direcionais do teclado. Para realizar essa programação, os estudantes partiram do código elaborado e descrito no episódio anterior, acrescentando modificações na troca de fantasia. Ao realizar essa programação, os estudantes optaram por utilizar o comando sempre, observando que poderiam utilizá-lo para movimentar o ator ao mesmo tempo em que ele troca de fantasia, já que as duas ações eram simultâneas. Percebemos nessa situação, ao optarem por juntar duas programações, que os estudantes obtiveram certo entendimento sobre como esse algoritmo funciona e o computador processa os blocos, demonstrando uma compreensão do fluxo do algoritmo ao unir em um mesmo algoritmo as ações de trocar de fantasia e se movimentar, já que os dois eventos deveriam acontecer com a mesma tecla sendo pressionada, o que difere da programação do movimento do morcego no episódio anterior, em que a troca de fantasias era independente do movimento do ator. Na Figura 17, podemos observar a programação elaborada pelos estudantes para movimentar o pinguim para a direita. Já na Figura 18, vemos uma sequência de movimentos do pinguim no palco.

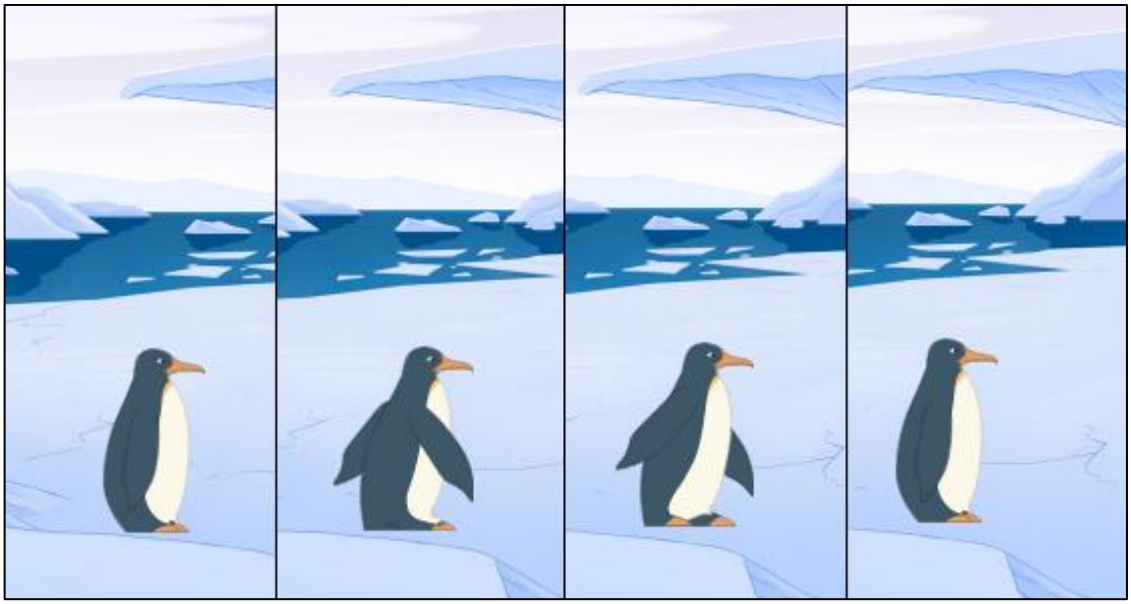
Destaca-se aqui o processo de abstração dos participantes no momento de programar essa ação do ator, pois assim que os estudantes imaginam o movimento do pinguim utilizando a troca de fantasias e o deslocamento, eles observam as principais características desse movimento, andar para frente enquanto utiliza uma fantasia durante um intervalo de tempo e andar para frente enquanto utiliza outra fantasia durante mais um intervalo de tempo, utilizando os blocos do Scratch como representações de cada etapa desse movimento, criando assim o algoritmo que o represente o todo.

Figura 17 - Programação do movimento do ator.

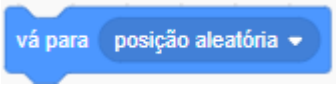


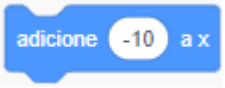
Fonte: Acervo do autor.

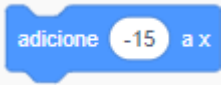
Figura 18 - Representação do movimento do ator.



Fonte: Acervo do autor.

Uma vez programado o ator pinguim, agora o grupo precisava programar a bola de neve para “atacar”. A programação do ator bola de neve desencadeou debates sobre algumas questões importantes. Primeiro, era necessário decidir sobre a posição inicial da bola de neve. Investigando os blocos de programação disponíveis no Scratch, o grupo encontrou o bloco 

. Esse bloco permite que o ator vá para uma posição aleatória, ou para o ponteiro do mouse, ou para um ator. O grupo, naturalmente, escolheu direcionar a bola de neve para o pinguim, para dar início ao “ataque”. Porém, ao utilizar esse bloco de programação, a bola de neve não se direcionava exatamente para o local que os estudantes esperavam. A expectativa era que a bola de neve estivesse posicionada na nadadeira do pinguim, mas ela surgia em frente ao peito do pinguim, deixando o grupo insatisfeito com o resultado. Para resolver o problema, os estudantes perceberam que deveriam programar um deslocamento da bola de neve para a nadadeira do pinguim, dando evidências de compreensão da atividade de programação e da noção de algoritmo, em que tudo deve ser explicitado e comandado ao computador pela linguagem de programação utilizada. Dessa forma, acrescentaram uma movimentação da bola de neve no palco utilizando os comandos 

e 

para que a mesma se movesse para a esquerda no eixo x e para baixo no eixo y. A determinação da coordenada que satisfez a posição adequada do ator bola de neve não foi imediata, já que os estudantes precisaram encontrar o distanciamento correto por meio de observação e ajustes. Contudo, foi notório que eles já mostravam familiaridade e controle em relação às posições no plano cartesiano representado pelo palco, visto que os blocos escolhidos foram exatamente os necessários para o sucesso da ação.

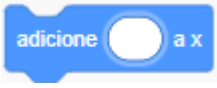
Uma vez ajustada a posição da bola de neve, o grupo deu início à programação do movimento da bola. Movimentar atores no palco já estava tornando-se uma tarefa simples. A partir de experimentos, os concluíram que o movimento da bola não poderia depender de uma sequência de acionamentos da tecla espaço (representando um movimento discreto, ou seja, com início e fim definidos) mas, uma vez a tecla pressionada, o ator (bola de neve) deveria se deslocar em um movimento contínuo (sempre adicionando um valor a x). Assim, os estudantes

tiveram que utilizar duas vezes o bloco 

: o primeiro para que, sempre que a tecla espaço fosse pressionada, acionasse o segundo sempre, que adiciona sempre um valor ao x (Figura 19). Então, cada vez que a tecla espaço fosse pressionada, a bola de neve surgia na mão do pinguim e se movimentava horizontalmente para a direita.

Figura 19 - Programação do movimento da bola de neve.

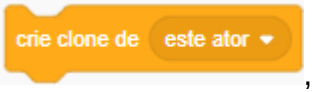
Fonte: Acervo do autor.

Outra discussão interessante foi relacionada ao tempo que a bola de neve levaria para chegar até a borda da tela em relação ao número inserido no bloco 

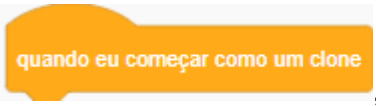
. Na primeira tentativa de programação, os estudantes utilizaram valores altos, fazendo com que o ator bola de neve se movesse muito rápido. Dessa forma, por experimentações, testes e erros, os estudantes foram ajustando os valores para encontrar a velocidade esperada, inserindo números maiores e menores, observando que, quanto maior o número, mais rapidamente o ator se movia e, consequentemente, quanto menor o número, se movia mais lentamente, explorando a ideia de proporcionalidade.

Nesse processo de verificação de valores a serem inseridos no bloco, percebemos que os estudantes utilizam o conceito de velocidade de forma inconsciente e espontânea, como um conhecimento-em-uso. Ao observar o tempo que o ator percorre a tela em uma distância fixa, os estudantes trabalham esse conceito de forma intuitiva, testando e verificando que o número adicionado ao x é

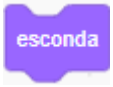
inversamente proporcional ao tempo que ele leva para chegar ao outro lado, trabalhando de forma concreta conceitos formais (velocidade e proporcionalidade inversa), utilizados com um propósito específico e de interesse dos estudantes, nesse caso, o de arremessar uma bola de neve.

O próximo desafio foi criar outras bolas de neve para que, cada vez que a tecla espaço fosse pressionada, uma bola de neve fosse "arremessada" pelo pinguim. A ideia inicial do grupo foi duplicar o ator, criando um número de bolas de neve suficientes para o pinguim arremessar. Ao testar a programação, os estudantes perceberam que a ideia era inviável, pois o número de bolas de neve necessárias durante o jogo não era fixo e previsível, o que motivou uma troca de estratégia. O grupo passou a vasculhar os blocos de programação, procurando inspiração para resolver o problema. Observando os blocos disponíveis, o estudante S identificou o bloco 

, e passou a investigar meios de inseri-lo na programação. Para provocar os estudantes sobre a utilização desse bloco e verificar a compreensão, o pesquisador questionou sobre o momento em que o clone deveria ser criado na programação. Logo, os estudantes perceberam que o clone deveria ser criado logo após a tecla espaço ser pressionada, para que o "ataque" fosse efetuado.

Os estudantes inseriram o bloco na programação e perceberam que toda vez que a tecla espaço era pressionada, uma nova bola de neve surgia na mão do pinguim, mas ainda sem movimento. O próximo passo foi, então, fazer com que o clone se movesse, como o ator verdadeiro. Nesse momento, o pesquisador indicou para os estudantes o bloco 

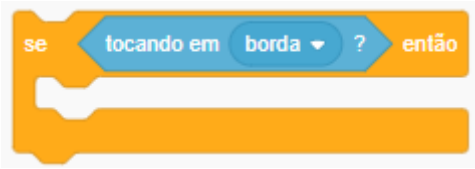
, para que entendessem que o movimento deveria ser inserido no clone. Dessa forma, foi imediato para os estudantes deslocarem os blocos que davam movimento para o ator e encaixarem no bloco indicado.

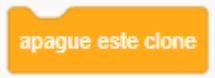
Um novo problema surgiu: quando a tecla espaço era pressionada, a bola de neve que originava os clones ficava visível. Para resolver essa situação, os estudantes utilizaram o bloco 

. Porém, esse bloco escondia também os

clones. Foi utilizado então o comando 

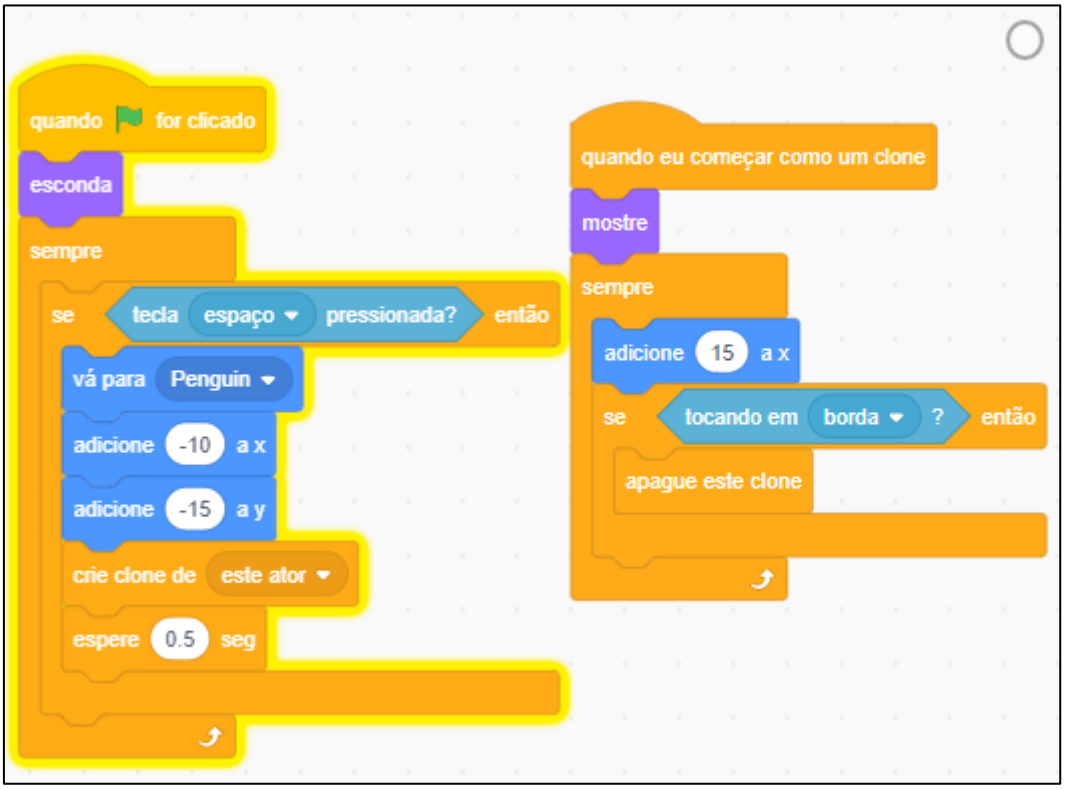
na programação dos clones, assim, quando o ator começava como um clone, o clone era mostrado, resolvendo o problema. Esses movimentos de ajustes no código de programação a cada imprevisto identificado revelam uma conduta cada vez mais autônoma na ação de programar desses estudantes. A capacidade de resolver pequenos problemas que compõem um desafio maior indicam habilidades de adaptação do conhecimento em novas situações. Para Brennan e Resnick (2012, p. 7) “as coisas raramente (ou nunca) funcionam como imaginamos; é fundamental para os designers desenvolverem estratégias para lidar com - e antecipar - problemas”. Sendo assim, salientamos esses ajustes como um processo importante no desenvolvimento da programação.

Por fim, o grupo teve o desafio de resolver mais um problema: na borda onde o clone batia, ficava um pedaço do personagem visível (devido à própria configuração do *software*). Portanto, os blocos de programação 

e 

foram utilizados para resolver o problema. Na Figura 20, podemos visualizar a programação do ator bola de neve (descrito como ator) utilizando a criação de clones.

Figura 20 - Programação dos clones do ator bola de neve.



Fonte: Acervo do autor.

A programação do movimento do pinguim com trocas de fantasia ocorreu sem empasses, pois os participantes já dominavam esse tipo de programação, ou seja, tomaram consciência do processo de localização do ator no palco e do comando espere para a troca de fantasias, de modo que esses conceitos fornecem um reforço para a ação e permitem de antemão uma previsão da animação programada (PIAGET, 1978). Para a programação dos clones, foi necessário mostrar a similaridade entre a programação de movimento feita anteriormente para ator original com a que deveria ser feita nos clones, pois eles deveriam assumir o mesmo comportamento, ou seja, seguir um padrão. Compreendendo essa semelhança e identificando o padrão de movimento, os estudantes programaram os clones pensando apenas no movimento que o ator original fazia ao ser arremessado de modo consciente e adaptado, revelando um processo de abstração, que considera apenas os detalhes importantes e despreza aqueles que não são relevantes para a solução (BRACKMANN, 2017). Observando a presença de componentes do pensamento computacional, podemos analisar esse processo de criação de clones a partir de uma abstração dos elementos importantes que movimentavam o ator

original, reconhecendo os padrões entre ele e os clones, utilizando os algoritmos de movimento da mesma forma que usados em situações anteriores, porém com adaptações específicas para a criação de clones.

A dupla I e S programou um morcego como personagem principal e clones de uma borboleta, que representaram os inimigos do morcego. Neste momento, os estudantes já sabiam como criar um clone de um personagem (escondendo o ator principal) e programar os clones utilizando o comando “quando eu começar como um clone”, então o primeiro desafio foi decidir o local de partida dos clones.

Como os estudantes não sabiam as coordenadas dos limites do palco, realizaram alguns testes para determinar a coordenada certa, pois o objetivo era que o clone saísse de fora da tela no sentido direita para esquerda. A partir da análise de alguns valores, os estudantes conseguiram abstrair da tela suas dimensões. Assim, determinaram o valor da coordenada de x que satisfazia a condição desejada. Em seguida, a dupla I e S testou também as coordenadas dos limites verticais do palco, e determinou o valor desejado, utilizando o comando “vá para x: (nº) y (nº)”. Ao testar o programa, os estudantes observaram que os clones das borboletas surgiam em um determinado tempo, escolhido por eles via comando “espere”, porém sempre na mesma posição. O pesquisador provocou os estudantes com a pergunta:

> Pesquisador: Há como os clones surgirem em lugares distintos?
>
> Estudante S: Deveria ter um comando no qual pudéssemos escolher um intervalo.

O pesquisador, fazendo uma orientação ocasional, sugeriu aos estudantes explorarem os blocos de comandos existentes, até que o estudante S visualizou na categoria de blocos operadores o comando 

.

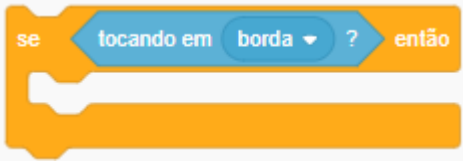
Dessa forma, a coordenada y poderia ser aleatória entre os valores identificados anteriormente para encontrar os limites das coordenadas y no palco.

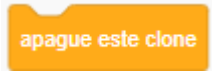
O conceito de intervalo numérico foi utilizado pelo estudante S, pois ele utiliza um valor (aleatório) dentro de um conjunto de valores com limites inferior e superior, que determinam o intervalo, caracterizando um processo de aprendizagem-em-uso (PAPERT, 1994). A partir de experimentações, os estudantes estimaram as coordenadas dos limites do palco (uma representação do plano cartesiano),

possibilitando controle da localização do espaço bidimensional em que os atores podem se movimentar.

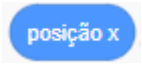
A utilização de um intervalo de tempo para o aparecimento das borboletas, utilizando o bloco 

dentro do comando espere, ocasionando uma aleatoriedade no intervalo de tempo do surgimento dos clones revela o uso do conceito de intervalos numéricos. Ao saber que é possível colocar um número para estipular um tempo fixo no comando espere, o estudante S, a partir de regulações ativas, reconhece a existência de um bloco que utiliza intervalos, utilizando-se de um conhecimento matemático entendido por ele e adaptado para solucionar o problema (RESNICK, 2007). Podemos observar a relação que deve ser feita do conceito matemático de intervalo com a função do bloco que é utilizado para escolher um número aleatório entre dois números, em que a mesma ideia é observada em duas linguagens diferentes, possibilitando o estudante uma ampliação do conceito matemático na prática (conhecimento-em-uso).

Determinar o momento de apagar o clone foi o novo desafio do grupo. Desafio porque, ao ultrapassar a borda esquerda, o software mostra uma ponta do personagem, dando a impressão visual de que o personagem parou nessa posição. A primeira tentativa do grupo foi utilizar o comando 

, juntamente com o bloco 

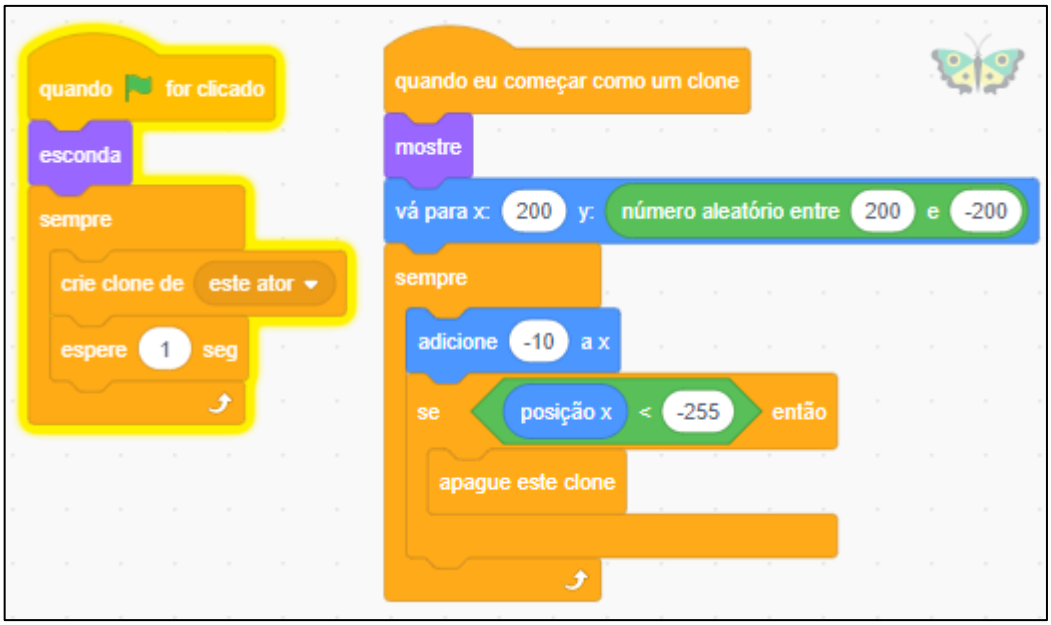
. Porém o resultado não deixou os estudantes programadores satisfeitos, pois o clone deveria passar pela borda (desaparecer aos poucos) e não desaparecer completamente assim que a tocava. Como solução, a dupla percebeu que a borboleta deveria permanecer visível enquanto a coordenada x não chegava em -255 (valor utilizado após alguns testes para que o clone continuasse na tela e tivesse parte de sua imagem permanecendo visível), para dar a ilusão visual de que o ator passou da borda, sendo apagado nesse momento. Em outras palavras, os estudantes concluíram que, se a coordenada fosse menor que -255, o software deixaria o clone da borboleta com um pedaço à mostra. Utilizaram o bloco de operadores 

utilizando o bloco azul, 

e o número -255. Dessa forma, quando o clone atingisse um valor de

coordenada menor do que -255, seria apagado, resolvendo o problema inicial. Na Figura 21 podemos observara programação criada para os clones.

Figura 21 - Programação para apagar os clones.



Fonte: Acervo do autor.

Percebemos um elemento de improvisação por parte dos estudantes para fazer o programa funcionar, remetendo a ideia trazida por Papert (1994) de *bricolagem* em seu livro *A Máquina das Crianças*, fazendo referência à obra O Pensamento Selvagem, de Lévi-Strauss, no qual ele utiliza-se do termo para se referir “a como as sociedades ‘primitivas’ conduzem uma ‘ciência do concreto’” (PAPERT, 1994, p. 128). Papert (1994, p. 138) utiliza-se do termo *bricolage* como:

> [...] uma metáfora para os estilos do antigo João-faz-tudo, que bate de porta em porta oferecendo-se para consertar o que quer que esteja estragado. Face em uma tarefa, o “arrumador” remexe em sua sacola de ferramentas sortidas buscando uma que se adaptará ao problema à mão e, se a ferramenta não funciona para a tarefa, ele simplesmente tenta outra [..] (PAPERT, 1994, p. 128).

Papert (1994, p. 128) ainda utiliza a *bricolage* como uma metodologia para a atividade intelectual, em que seus princípios básicos são: “use o que você tem, improvise, vire-se”. Interpretamos a caixa de ferramentas dos estudantes como os blocos de programação e as ideias matemáticas (limites de intervalos, desigualdades, deslocamentos) para obter os resultados desejados. Quando a utilização dos códigos que fazem apagar os clones não acarreta na solução prevista, os estudantes recorrem ao que eles tinham disponível e de conhecimento próprio,

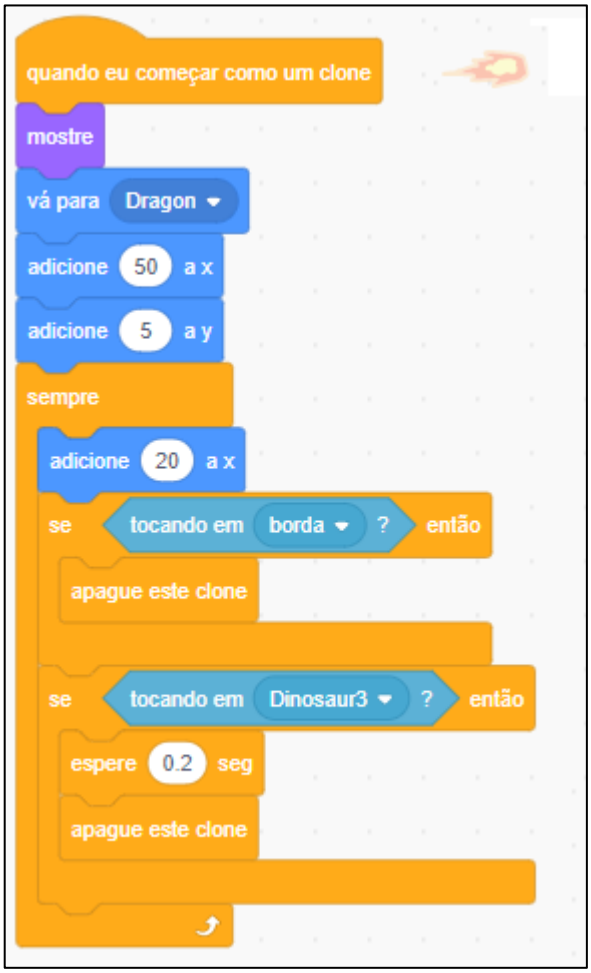
reconhecendo as desigualdades nos blocos do Scratch e utilizando-as para resolver o problema.

5.5. Colisão de atores: Atores Reagindo ao Contato

Nesse episódio, escolhemos analisar a programação do segundo grupo. A partir dos recursos abordados nos encontros anteriores, foi planejado o seguinte desafio: programar um personagem principal (os estudantes escolheram os *sprites* de um dragão já disponível no Scratch), com um "poder" (os estudantes criaram um ator utilizando os recursos de edição de imagem do software, para representar uma bola de fogo que sairia da boca do dragão), e um inimigo (os estudantes utilizaram os *sprites* de um pterodactyl, também disponível no Scratch). O objetivo principal do desafio era programar as colisões dos atores e, para isso, os estudantes deveriam pensar na colisão da bola de fogo com o pterodactyl e também na colisão do pterodactyl com o dragão.

Para isso, duas situações foram observadas: a primeira foi incorporar a programação de colisão no programa que já estava sendo elaborada; a segunda, mostrar que houve uma colisão. Para a primeira situação, exemplificaremos a bola de fogo ao tocar no personagem pterodactyl. Na Figura 22, percebemos que os estudantes programaram para a bola de fogo (especificado na programação como Ator 1) tocar na borda e adaptaram para quando o personagem tocar no pterodactyl (especificado na programação como Dinosaur3). Sendo assim, o grupo utilizou um algoritmo já programado anteriormente e o adaptou para a nova situação, facilitando a nova programação. Novamente percebemos os pilares da generalização e da abstração (BRACKMANN, 2017) no processo de programação dos estudantes, apresentando o uso do pensamento computacional para resolver o problema, na qual os estudantes conseguem abstrair as características semelhantes de problemas anteriores e conseguem adaptar com a inclusão de características particulares, em que novamente a abstração do problema é necessária para a seleção do que é importante e do que é novo.

Figura 22 - Programação do ataque do personagem.



Fonte: Acervo do autor.

No segundo momento, escolhemos novamente a colisão entre a bola de fogo (Ator 1) e o pterodactyl (Dinosaur3), porém com a diferença de que queremos mostrar que o pterodactyl foi atingido. Para isso, o grupo decidiu mudar a fantasia do ator para simular que havia sido atingido, revelando que a utilização dos blocos de comandos de trocas de fantasias já estava incorporada em suas "caixas de ferramentas" para resolver os pequenos desafios de programação que compõe o desafio maior. Após encaixar o bloco de mudança de fantasia e executar o programa, os estudantes perceberam que a mudança não estava acontecendo. Analisando os blocos de programação, lembraram que é preciso estipular um tempo para que a fantasia possa aparecer antes de apagar o clone, e resolveram o problema. Na Figura 23, mostramos a programação dos clones do pterodactyl.

Figura 23 - Programação de colisão do ator

Fonte: Acervo do autor.

Na Figura 24 observamos três momentos durante o jogo. O primeiro momento (Figura 24 (a)), em que não há colisão entre os atores, o segundo momento (Figura 24 (b)) em que há a colisão da bola de fogo com o pterodactyl (depois de mudar a fantasia, o clone espera um momento antes de ser apagado) e o terceiro momento (Figura 24 (c)) em que há a colisão entre o dragão e o pterodactyl, em que o dragão muda de fantasia (depois de mudar a fantasia, o dragão é escondido e mostrado novamente em uma outra coordenada definida pelos estudantes na programação).

Figura 24 - Representações dos movimentos dos atores na programação.

(a) (b) (c)

Fonte: Acervo do autor.

Nesse episódio, não percebemos a construção ou utilização de novos conceitos matemáticos durante a programação, além daqueles já apontados anteriormente e que foram resgatados aqui (deslocamento, coordenadas, desigualdades, tempo, movimentos discretos e movimentos contínuos, lógica condicional e fluxo de algoritmo). Porém, percebemos que os pilares do pensamento computacional apontados por Brackmann (2016) estavam sendo desenvolvidos e utilizados durante esse processo. A adaptação de um algoritmo já programado em uma nova situação fica evidente nessa programação, o que evidencia o processo de generalização a partir do reconhecimento de padrões sendo utilizado pelos participantes. Segundo Brackmann (2017, p. 35), “Padrões são similaridades ou características que alguns dos problemas compartilham e que podem ser explorados para que sejam solucionados de forma mais eficiente”.

Nesse sentido, identifica-se uma positiva familiaridade com os algoritmos utilizados pelos participantes, em que os padrões encontrados por eles nas programações realizadas nesse episódio com situações anteriores possibilitaram encontrar soluções para os problemas com mais facilidade e agilidade pois, “quanto mais padrões se consegue encontrar, mais dinâmico e rápido a macro solução é encontrada” (BRACKMANN, 2017, p. 37).

A atividade de abstração também é verificada, já que os estudantes mostram facilidade em identificar os códigos importantes para a criação dos novos algoritmos, assim como a abstração das características dos novos problemas, representando as ações dos atores no palco a partir dos códigos desenvolvidos por eles. Sendo assim, percebemos dois momentos importantes em que a abstração foi utilizada pelos

estudantes, momentos esses descritos por Wing (2006), na escrita dos algoritmos e suas interações e na seleção dos dados importantes.

5.6 Variável no Scratch: Contagem de “vidas” e condicionando o ator

Nesse episódio, daremos continuidade à análise do segundo grupo. Um dos desafios que o grupo estava enfrentado era que, quando o ator (dragão) desaparecia após o contato com o inimigo e surgia novamente em um local que representava um novo início, se um inimigo estivesse nesse local no momento em que ele aparecesse, o personagem morreria instantaneamente. Sendo assim, foi necessária a utilização de uma variável que deixasse o personagem invulnerável por um intervalo de tempo, impedindo que o ator fosse morto nesse intervalo e pudesse escapar assim que ressurgisse. Para resolver o problema, o grupo precisa programar o ator para duas situações possíveis: a primeira para que o ator dragão fosse vulnerável às colisões dos outros atores envolvidos no jogo (bola de fogo e pterodactyl); a segunda para que ele ficasse invulnerável por um intervalo de tempo, dando oportunidade para o jogador movimentá-lo sem ser atingido pelo inimigo. A divisão do problema em duas situações revela a decomposição, um dos pilares do pensamento computacional, habilidade que permite quebrar um problema maior em problemas menores e mais fáceis de resolver (BRACKMANN, 2017).

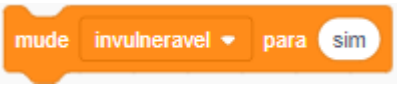
Assim, para resolver o problema, os estudantes utilizaram o recurso de criar "variável" para controlar o comportamento do dragão nas diferentes situações previstas. Identificamos a necessidade de intervenção do pesquisador para utilizar esse recurso, que ocorreu por meio de questionamentos e provocações durante o processo de programação das variáveis, apoiados nas ações dos estudantes.

No seguimento da oficina, a primeira variável criada, utilizada para dar invulnerabilidade ao ator dragão, foi denominada 

. Criada essa variável, os estudantes deveriam compreender onde e como ela deveria ser inserida na programação. A partir do questionamento sobre quando o ator deveria ficar invulnerável, foi estabelecido o seguinte diálogo:

Pesquisador: Quando começa o jogo, o ator está invulnerável?

Estudante V: Não, só quando ele morrer.

Pesquisador: Ótimo, então colocaremos o bloco “mude ‘invulnerável’ para (sim)” (

) se tocar (no outro ator), ele já tem que ficar invulnerável, quando que ele vai deixar de ficar invulnerável?

Estudante V: Ah! Dá pra deixar uns “segundinhos”.

Pesquisador: Certo, então ele vai trocar a fantasia, esperar um segundo, esconder, esperar mais um segundo, ir para a posição que escolhemos e mostrar. Então ele tem que ficar invulnerável depois de mostrar também. Então, vamos colocar mais um tempo entre tocar e ele não ficar mais invulnerável. Como podemos fazer pra ele ficar invulnerável, como mostra nos jogos que vocês jogam?

Estudante D: Podemos deixar ele piscando por um tempo.

Pesquisador: Como podemos fazer para ele ficar piscando?

Estudante D: Podemos o fazer mostrar e esconder.

Pesquisador: Certo, ele já está escondido, então ele vai mostrar, esperar um tempo, como ele tem que piscar bem rápido, quanto tempo vocês acham que ele deve utilizar? Primeiro temos que saber quanto tempo ele vai ficar invulnerável, quanto tempo vocês gostariam.

Estudante V: Uns dois segundos.

Pesquisador: Ok, quanto tempo podemos então fazer com que ele “mostre” e “esconda”.

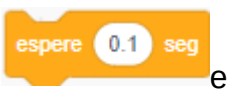
Estudante D: Pensei em deixar ele mudando de fantasia durante 0,1 segundos, para ficar piscando bem rápido, então se queremos 2 segundos, tem que fazer ele piscar 20 vezes.

Pesquisador: Ok, vamos programar, então.

Como o ator já estava escondido, os estudantes adicionaram o bloco 

seguido do bloco 

, bloco 

, 

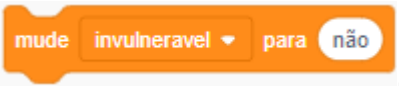
e, por fim, bloco 

.

Pesquisador: Por enquanto, esse processo deu 0.2 segundos, muito pouco, queremos mais tempo, o que podemos fazer?

Estudante D: Dá para colocar um bloco que eu havia visto, “repetir (10) vezes”, para ficarem os 2 segundos.

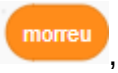
Interessante observar as noções dos participantes de fluxo de algoritmo programado ao utilizarem o bloco com a função de repetir, dando mais eficiência e otimização na programação, verificando que os estudantes compreendem os códigos utilizados e entendem o funcionamento dos blocos, mesmo que em níveis mais elementares.

Finalizada a programação da animação que mostra o ator invulnerável, os estudantes acrescentaram o bloco 

, para que o ator pudesse ser atingido novamente. Na Figura 25, vemos a programação do ator ao tocar no inimigo.

Figura 25 - Uso do recurso variável na programação.

Fonte: Acervo do autor.

Em seguida, os estudantes testaram sua programação. Nesse momento, o grupo percebeu que, quando o dragão trocava a fantasia que representava que ele havia sido atingido, o jogador ainda podia movimentar o dragão utilizando as setas direcionais. Além disso, o dragão também conseguia utilizar seu poder enquanto estava “morto”. Para solucionar o problema, os estudantes reconheceram a necessidade de estabelecer um momento para que as teclas parassem de funcionar e que a bola de fogo (poder do ator) não pudesse ser utilizada. Para isso, os estudantes perceberam a necessidade de uma nova variável, na qual foi denominada variável 

, como mostra o diálogo a seguir.

Pesquisador: Quando que o ator pode atirar (o poder)?

Estudante V: Quando ele estiver vivo.

Pesquisador: E quando ele não pode atirar?

Estudante V: Quando ele morrer.

Pesquisador: Ok, logo vamos criar a variável “morreu”. Começando o jogo, ele estará vivo ou morto?

Estudantes V e D: Vivo!

Pesquisador: Certo, onde que ele morre?

Estudante V: Quando o inimigo toca nele.

Pesquisador: Então é a mesma coisa que a variável “invulnerável”, logo vamos colocar o bloco “mude ‘morreu’ para (sim)”, junto com o bloco “mude ‘invulnerável’ para (sim)”. E quando ele pode começar a atacar?

Estudante V: Quando ele estiver vivo.

Pesquisador: E quando isso acontece na programação?

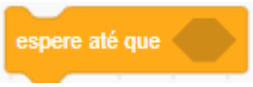
Estudante D: Quando a invulnerabilidade passar.

Pesquisador: Mas ele não pode atirar antes disso? Ele não pode atirar enquanto estiver piscando? Ou quando ele aparece já pode atirar?

Estudante D: Então logo depois que ele aparecer, que fica antes do repita.

Pesquisador: Isso mesmo.

Após o diálogo, que foi conduzido no sentido de provocar o entendimento sobre o fluxo do algoritmo e a ordem dos blocos de comandos, os estudantes implementaram a ideia discutida. Porém ao executarem o jogo, o dragão continuava atirando a bola de fogo, mesmo quando atingido pelo inimigo, assim era possível movimentar o ator depois de ser tocado pelo inimigo. O pesquisador questionou os estudantes sobre o motivo para essa situação estar ocorrendo. Como os estudantes não identificaram que a programação deveria ser para o ator bola de fogo (e não para o dragão), foi necessária uma intervenção do pesquisador. Em relação ao problema do ataque do ator, a programação com a variável 

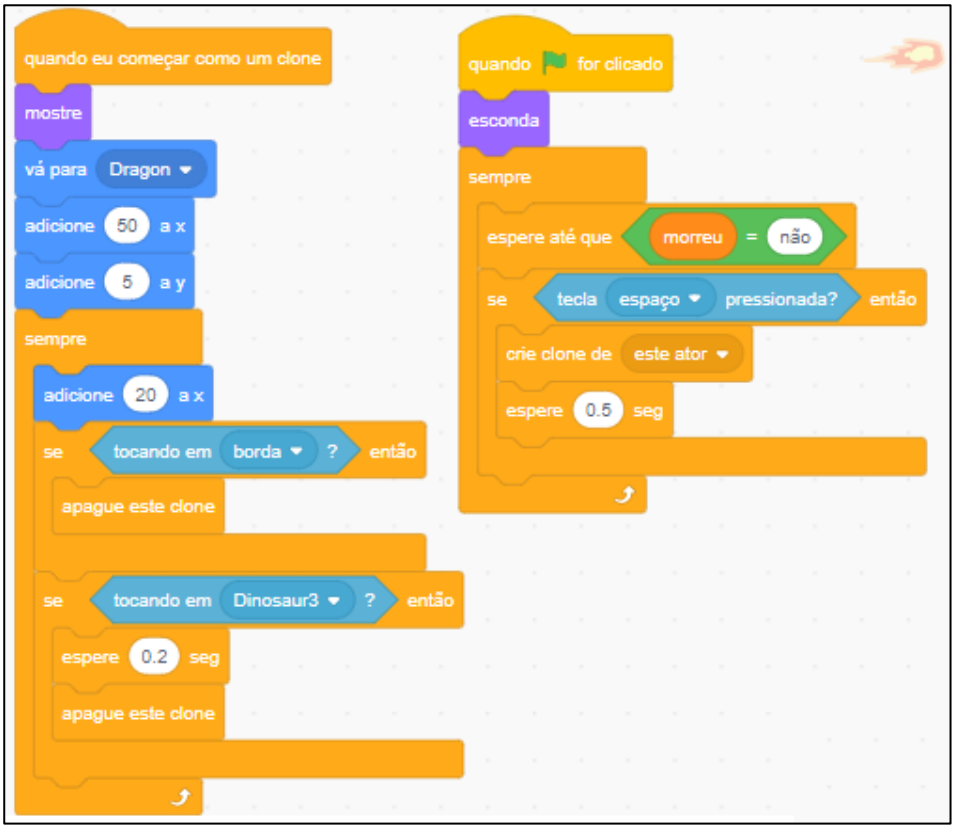
estava associada à programação de atirar, então foi necessário modificar a programação do ataque dele. Para isso, os estudantes utilizaram o bloco 

, com o bloco operador 

, para que a programação que permite o uso da

tecla espaço para atirar seja habilitada apenas quando o ator não estiver morto, obtendo a programação a seguir (Figura 26).

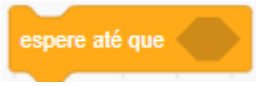
Figura 26 - Programação com o uso do recurso variável.



Fonte: Acervo do autor.

Do ponto de vista matemático, vemos a utilização das duas variáveis invulneravel e morreu como a criação de condicionais lógicas para que o computador entenda quando um algoritmo deve ser executado ou não, ou seja, uma ação só será executada se determinada condição for satisfeita. Dessa forma, podemos dizer que as variáveis criadas nesse procedimento funcionam como “espaços abertos”, que dependem de determinadas condições serem satisfeitas, para o computador executar ou não determinado algoritmo. A utilização de lógica condicional é um dos elementos apontados por Grover e Pea (2013) como importantes para o desenvolvimento do pensamento computacional. Ainda, a noção de variável está implícita aqui, uma vez que as variáveis criadas podem assumir diferentes valores (neste caso, valores lógicos SIM ou NÃO)

Já para resolver o problema da movimentação do ator depois de ser tocado pelo pterodactyl, os estudantes analisaram o problema anterior e identificaram a necessidade de incluir dentro do bloco 

o bloco 

, juntamente com o bloco 

no algoritmo que determinava o movimento (Figura 27), de modo a esperar o momento adequado do jogo para ser executado, quando o ator não estivesse morto. Observamos que para essa ação, os estudantes utilizaram um pensamento semelhante ao ataque do ator, analisando as ações propostas para ele quando ele estivesse “vivo”, sugerindo um processo de identificação de padrão. Analisamos nas ações dos estudantes uma convergência com alguns dos pilares do pensamento computacional descrito por Brackmann (2017). Portanto, entendemos que os estudantes abstraem as ações do ator nos algoritmos criados, compreendendo o modo que esses algoritmos influenciam o ator no palco. Ao mesmo tempo, ao verificar um padrão de funcionamento dos algoritmos, que devem funcionar quando o ator está vivo, os participantes utilizam-se da mesma estratégia para “ensinar” o computador quando esses algoritmos devem ser requisitados. Ao “ensinar” o computador que ele deve esperar até que a varável esteja favorável para realizar a programação do algoritmo, ou seja, quando o ator não estiver morto, os estudantes estabelecem sentenças de verdadeiro ou falso para o computador, desenvolvendo conceitos de lógica aplicados na programação.

Figura 27 - programação com o recurso variável.

Fonte: Acervo do autor.

Em determinado momento da oficina, os estudantes perceberam que, ao apertar o botão “pare” ( ) logo após o ator morrer, as variáveis morreu e invulneravel ficavam habilitadas como “sim”, sendo que, ao iniciar o jogo elas deveriam estar habilitadas como “não”. Quando a bandeira verde era pressionada novamente, o ator não se movia, pois, como a variável morreu estava habilitada como “sim” e os algoritmos que programavam o movimento e o lançamento do ataque estavam programados para esperar que essa variável fosse igual a “não”, era necessário que o ator morresse para que o algoritmo que estabelece a mudança na variável para “sim” pudesse se completar e mudar para “não”. Para resolvermos isso, o estudante D sugeriu colocar um bloco que, ao iniciar o jogo, habilitasse as variáveis para “não”. Sendo assim os estudantes adicionaram esses blocos juntamente com os blocos que determinavam o início do jogo (fantasia, cenário e posição inicial do ator), como mostra a Figura 28.

Figura 28 - Algoritmo com variáveis.

Fonte: Acervo do autor.

A próxima variável criada pelo grupo de estudantes foi a variável vida, com comportamento diferente das variáveis anteriores, ou seja, podendo assumir diferentes valores numéricos para indicar a contagem de vidas. Inicialmente, os estudantes decidiram que o jogo inicia com três vidas (incluindo a vida zero), ou seja, o jogador iria começar com a variável vida assumindo o valor numérico dois. Criada a variável 

, os estudantes deram início à programação de um novo algoritmo, inserindo o bloco 

seguido do bloco 

, com o propósito de estabelecer um limite para o número de vidas. Para provocar nos estudantes um processo de antecipação da organização dos blocos de comandos (e verificar se os estudantes seriam capazes de realizar essa antecipação), o pesquisador iniciou o seguinte diálogo sobre o momento em que o número de vidas diminui:

Pesquisador: O que deve acontecer para a vida diminuir?

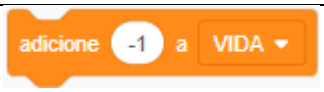
Estudante V: Quando ele morrer (colidir com o outro ator).

Pesquisador: Na nossa programação, onde ele morre?

Estudante D: Quando ele toca no pterodactyl.

Pesquisador: Certo, então utilizaremos qual bloco para retirar uma vida?

> Estudante D: Não tem um bloco de subtrair mas podemos utilizar o bloco adicionar e inserir o número -1, para adicionar um número negativo.

Assim, o grupo utilizou o bloco 

no algoritmo que programa a colisão com o inimigo juntamente com os outros blocos de variáveis. Ao executar o programa, os estudantes perceberam que o objetivo de diminuir as vidas do ator funcionou conforme suas expectativas. Porém as vidas poderiam ser retiradas infinitamente, precisando assim de um momento para finalizar o jogo, ou seja, um *game over*. Não temos registro se os estudantes já haviam antecipado esse novo problema, porém acreditamos que o conhecimento dos números inteiros pudesse causar essa dúvida nos participantes, já que ao zerar o número de vidas e não haver nenhuma restrição para que o computador parasse de adicionar a quantia de -1 ao número de vidas, há a existência dos números negativos.

Para esse propósito, os estudantes retomaram a programação da contagem de vidas do ator igual a 2. Eles precisam estipular um modo de mostrar para o jogador o *game over*. Sendo assim, os estudantes escolheram uma imagem para mostrar que o jogo tinha chegado ao fim e precisavam programar o aparecimento dessa tela quando o jogador perdesse as três vidas. Utilizaram os operadores de desigualdade para esse propósito, ou seja, quando a variável vida fosse menor do que zero, o jogo deveria parar e mostrar a tela *game over*. Sendo assim, dentro do bloco 

, os estudantes acrescentaram o bloco se 

então, seguido do bloco 

para que a programação mudasse de cenário. Porém, ao aparecer a tela de *game over*, os atores ainda se movimentavam na tela. Contudo, o problema foi facilmente resolvido inserindo o bloco 

, obtendo a seguinte programação (Figura 29).

Figura 29 - Algoritmo para contabilizar a vida do ator.

Fonte: Acervo do autor.

Finalizado esse episódio, concluímos a primeira etapa da oficina em que foram trabalhados recursos e estratégias para que os estudantes pudessem programar seus próprios jogos. A seguir, analisaremos os episódios em que os estudantes criaram seus próprios algoritmos e como detectaram e resolveram os *bugs* nas suas programações.

5.7 Programando o Jogo: Novos Mecanismos, Novos Desafios

Nesse subcapítulo, analisaremos as ações dos participantes na programação de seus jogos, observando o desenvolvimento de ideias e estratégias na programação do jogo. Vamos analisar o desenvolvimento do jogo do grupo dos participantes A, M, S e I, já que esse grupo se engajou mais em utilizar novos mecanismos, diferentes dos que foram trabalhados na oficina, o que permitiu analisar como os estudantes se comportaram ao se depararem com novas dificuldades e como resolveram cada adversidade.

Logo, iremos relatar e analisar esses momentos separadamente, partindo da elaboração inicial do jogo, seguindo com os episódios em que os estudantes se depararam com o novo.

5.7.1 Elaboração do jogo: Ideias iniciais

Decidimos por organizar um dos encontros da oficina, com duração de duas horas, para que os estudantes pudessem se dedicar ao projeto do jogo. Diferente dos outros encontros, os estudantes organizaram seus projetos na sala multidisciplinar da escola, composta por mesas de formato hexagonal, proporcionando melhores condições para os grupos se comunicarem.

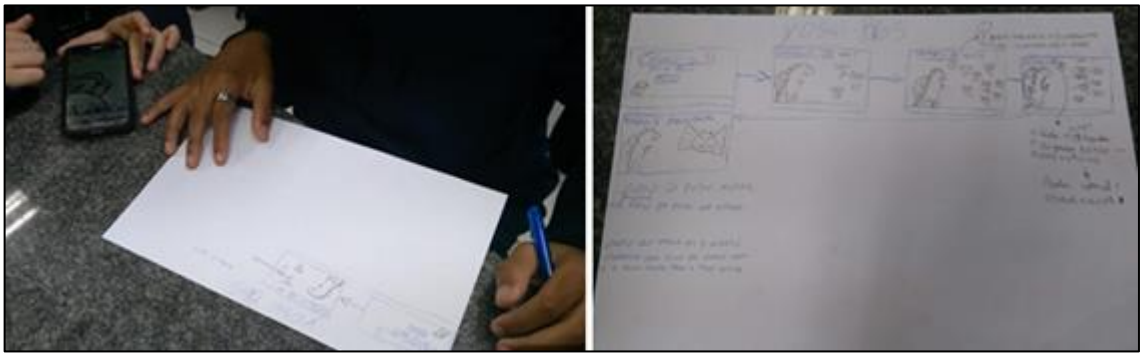
O grupo que analisaremos projetou seu jogo com o nome de “Yoshi Bros” no qual seu personagem principal seria o personagem Yoshi, do jogo Mario Bros.

Vemos na escolha do personagem a importância que há em os estudantes darem significado aos seus projetos. Ao relatar um projeto em que estudantes têm a possibilidade de se engajar em uma atividade de envolvimento social e significativo, Papert (1994, p. 30) afirma:

> [...] aprecio mais é a oportunidade que ela oferece aos estudantes de irromper de sua própria estrutura para engajar-se em atividades mais auto direcionadas. Um modo pelo qual os estudantes rompem com bastante frequência é utilizar a experiência adquirida no projeto para engajar-se [..]

A ideia dos estudantes foi a de programar o personagem com ataques semelhantes ao do jogo oficial e com ataques especiais que surgiriam a cada 1 minuto e 15 segundos e teriam duração de 15 segundos. O jogo foi pensado pelos estudantes para conter uma tela inicial, algumas fases nas quais os inimigos seriam morcegos (referência ao jogo programado durante os encontros anteriores) e uma tela de “game over” caso o jogador perdesse 5 vidas durante o jogo. Nessa tela, o jogador teria a opção de recomeçar o jogo, voltando para a primeira fase, ou para a tela inicial do jogo. Na Figura 30, mostramos os estudantes projetando o jogo e o resultado final do projeto. Observando o engajamento dos estudantes e a empolgação no momento de inserir suas ideias no papel, podemos afirmar que esse momento foi de envolvimento e satisfação e, mesmo com a ausência do computador, os estudantes perceberam a importância do planejamento e compartilhamento de ideias.

Figura 30 - Participantes organizando o projeto do jogo.



Fonte: Acervo pessoal.

Sobre os movimentos do ator, os estudantes pensaram em uma movimentação horizontal, com a possibilidade de saltos, respeitando a ideia de gravidade no jogo, primeiro grande desafio do grupo, já que inserir a ideia de gravidade na programação seria uma novidade para eles.

Consideramos importante esta aula fora do ambiente informatizado, pois possibilitou aos estudantes um momento para que pudessem imaginar os seus projetos, para otimizar os encontros dedicados à programação, já que estarão com ideias bem definidas sobre o que irão criar, assim como analisar a organização e planejamento dos participantes.

As ações de planejamento utilizadas pelos estudantes sem a utilização do computador também são caracterizadas como conceitos que contribuem para a aprendizagem computacional:

> Muitos tópicos importantes da Computação podem ser ensinados sem o uso de computadores. A abordagem desplugada introduz conceitos de hardware e software que impulsionam as tecnologias cotidianas a pessoas não-técnicas. Em vez de participar de uma aula expositiva, as atividades desplugadas ocorrem frequentemente através da aprendizagem cinestésica (*e. g.* movimentar-se, usar cartões, recortar, dobrar, colar, desenhar, pintar, resolver enigmas, etc.) e os estudantes trabalham entre si para aprender conceitos da Computação. (BRACKMANN, 2017, p. 50)

Analisamos que durante o planejamento dos jogos, os pilares do pensamento computacional também foram perceptíveis nas ações dos participantes, mesmo que desplugados. Ao planejarem um jogo, no qual há uma complexidade de elementos, a abstração do problema juntamente com a sua decomposição em subproblemas são características predominantes nesse encontro.

> Pensamento computacional é usar abstração e decomposição ao atacar uma tarefa grande e complexa ou projetar um sistema complexo e grande.

> É a separação de interesses. É escolher uma representação apropriada para um problema ou modelagem dos aspectos relevantes de um problema para torná-lo tratável. (WING, 2006, p. 33, tradução do autor[14])

Logo, ao organizar as fases do jogo, tela inicial e tela de *game over*, movimentações do ator e dos demais personagens, os estudantes entram na fase de decomposição do problema, seguindo de uma abstração que é colocada no papel, definindo os detalhes a serem destacado e quais detalhes ignorar, o que, segundo Wing (2008) fundamenta o pensamento computacional. Por fim, o planejamento em conjunto dos estudantes possibilita "exercitar o trabalho colaborativo, uma das aptidões do Pensamento Computacional". (BRACKMANN, 2017, p. 57).

5.7.2 Programando o salto do ator

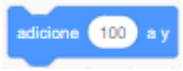
No desenvolvimento da programação do ator principal do jogo, os participantes do grupo tinham como objetivo fazer com que o ator saltasse para frente e para trás, realizando um movimento que lembrasse uma parábola (termo não utilizado pelos estudantes). O grupo decidiu dividir tarefas, e essa incumbência ficou direcionada aos estudantes A e M, enquanto os estudantes I e S assumiram outras responsabilidades.

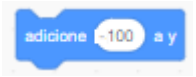
Inicialmente, os estudantes observaram os códigos de programação de outros jogos criados no Scratch (no site do Scratch é possível verificar o código de outros projetos clicando em "visualizar interior"), para que pudessem se inspirar na programação do salto do ator. Porém, as programações observadas em alguns dos jogos envolvia o uso de variável para criar gravidade no Scratch, o que não foi compreendido por eles. Como o grupo gostaria de criar o seu próprio algoritmo e código de programação, a ideia de trabalhar com gravidade foi descartada.

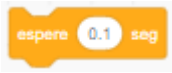
Junto da programação que movimentava o ator, os estudantes inseriram a programação do salto ao pressionar a tecla seta para cima. Para criar este

---

[14] Citação original: "Computational thinking is using abstraction and decomposition when attacking a large complex task or designing a large complex system. It is separation of concerns. It is choosing an appropriate representation for a problem or modeling the relevant aspects of a problem to make it tractable." (WING, 2006, p. 33).

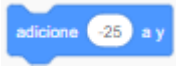
algoritmo, os estudantes iniciaram pelos blocos 

e 

e perceberam que o ator realizava o movimento muito rápido, dificultando a observação do salto. Para contornar a situação, os estudantes inseriram o comando 

, para que fosse possível observar o ator no alto do salto. Contudo, o movimento executado pelo ator ainda não correspondia ao movimento que os estudantes queriam. A ideia seria com que a descida do salto fosse mais lenta e, para isso, os estudantes inseriram um comando para que o ator pudesse descer de forma que fosse possível observar essa animação, separando um movimento único em quatro etapas utilizando os blocos 

, para dar a ilusão de movimento mais lento. Ao executar as alterações, os estudantes perceberam que o ator continuava fazendo o mesmo movimento anterior. O estudante A identificou que era necessário inserir um intervalo de tempo de espera para que fosse possível observar o movimento, assim como visto em encontros anteriores. Para resolver essa situação, os estudantes utilizaram o comando 

depois do bloco adicione 

. O movimento do ator ficou semelhante a um salto e os estudantes ficaram satisfeitos com o resultado: o ator subia rapidamente e descia aos poucos. Porém, apesar de o tempo de reação do ator estar de acordo com as expectativas dos estudantes, as paradas que o ator fazia no ar enquanto descia eram muito visíveis. Para melhorar esse efeito na programação do ator, houve o seguinte diálogo.

Pesquisador: Como poderíamos fazer para as paradas do ator ficassem menos visíveis?
Estudante M: Poderíamos aumentar o número de vezes em que o ator para no ar, para ele ir mais rápido.
Pesquisador: E quanto vocês aumentariam?
Estudante M. Podemos tentar dobrar (repita 8 vezes)
Pesquisador: Teríamos que modificar mais alguma coisa na programação?

Estudante M: Sim, teríamos que diminuir o valor que retiramos de y, para que no final ele volte para a posição. Como estamos dobrando o número de vezes que o movimento vai se repetir, temos que dividir o valor que vamos retirar do y por 2.

Pesquisador: Quanto devemos colocar no “adicionar a y”?

Estudante M: Se é a metade, então seria -12,5.

Analisando o raciocínio da estudante M, percebemos uma relação matemática de proporcionalidade inversa. Esperava-se que a estudante pensasse em dividir o número 100 (valor que foi adicionado a y) pelo número 8, para encontrar o valor, porém o conceito de medidas inversamente proporcionais foi logo utilizado pela estudante, o que causou uma surpresa positiva durante o processo.

Ao analisar a fala da estudante M “*Poderíamos aumentar o número de vezes*” podemos perceber que o uso da relação distância e tempo, utilizado nos primeiros encontros da oficina. Ela percebe as possibilidades de fragmentação da distância em um espaço e um tempo já definidos, observado na frase da estudante: “*Poderíamos aumentar o número de vezes*”. Logo, ao argumentar: “*teríamos que diminuir o valor que retiramos de y, para que no final ele volte para a posição*” percebemos a forma que a estudante resolve o problema da fragmentação, diminuindo os valores de y para que ele se adapte à distância já definida. Assim que a estratégia está definida, a estudante finaliza a solução do problema utilizando as operações matemáticas, salientada na frase: “*Como estamos dobrando o número de vezes que o movimento vai se repetir, temos que dividir o valor que vamos retirar do y por 2*”. Percebemos nesse contexto a aplicação direta de conceitos matemáticos sendo utilizados na prática da programação do salto do ator, revelando um conhecimento-em-uso para resolver um problema.

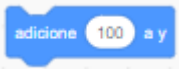
Observando o movimento do ator, os estudantes perceberam que o tempo de descida do ator era maior do que antes. Sendo assim, foram feitos alguns questionamentos em relação a essa situação.

Pesquisador: Por que vocês acham que o ator está descendo mais lentamente?

Estudante A: Como aumentamos o número de repetições (do movimento) provavelmente aumentamos o tempo no comando espere.

> Pesquisador: Isso! E como vocês podem arrumar?
>
> Estudante M: Bom, como antes ele repetia quatro vezes, então o tempo de descida era de 0,4 segundos, agora foi para 0,8. Sendo assim, podemos diminuir o tempo no comando espera para que dê os 0,4 segundos que queremos.

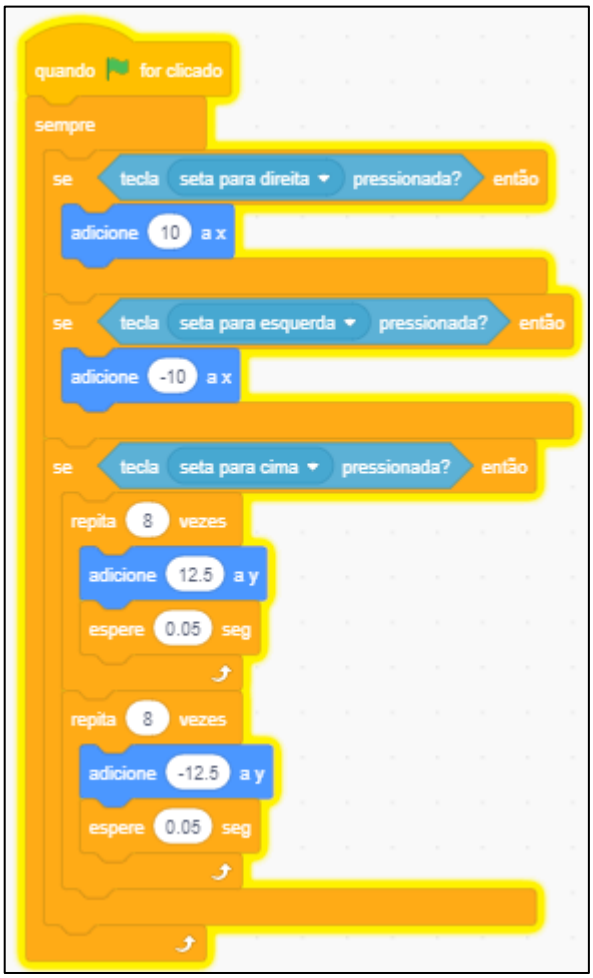
Nesse momento, a estudante M procura o número que multiplicado por 8 daria 0,4 segundos, percebendo em seguida que bastava dividir o 0,1 segundos por 2, utilizando o pensamento anterior, obtendo o valor de 0,05 segundos. Vemos aqui que a estudante M percebeu a proporcionalidade direta entre a fragmentação da distância e, consequentemente, a diminuição do tempo entre as paradas do ator, trabalhando esse conceito dividindo esse tempo por dois, já que a distância também havia sido dividida por dois. No final desse processo, o movimento do ator ficou mais natural, quase não podendo perceber os intervalos de parada do ator durante a descida do salto, agradando os estudantes que estavam programando essa etapa do jogo. Visto que o planejamento inicial era inserir gravidade no jogo, contudo não conseguiram compreender o conceito da variável que envolvia essa programação, os estudantes tiveram que improvisar, utilizando novamente a ideia de *bricolage* imposta por Papert (1994), utilizando seus conhecimentos para simular uma ação que fosse semelhante com o propósito primário do grupo.

Como o movimento da descida estava mais natural que o movimento repentino da subida do salto, os estudantes resolveram utilizar o mesmo movimento para os dois casos, retirando o bloco 

da programação da subida e inserindo o mesmo algoritmo da descida, já que estavam se referindo ao deslocamento para cima. Vemos essa ação imediata dos estudantes como uma generalização do processo anterior que facilitou o processo, utilizando os códigos feitos para a descida do salto do ator, com suas devidas adaptações, também para a subida do salto. Observamos também o processo de abstração dos estudantes na representação do salto do ator em algoritmos que programam esse movimento. Analisando esses pontos em questão, elementos básicos do pensamento computacional nas práticas dos participantes são identificados em suas tomadas de decisões, identificado dois dos pilares do pensamento computacional descritos por Brackmann (2017), a identificação de padrões, podendo ser vista como uma generalização, e a abstração.

Na Figura 31, vemos a programação dos movimentos do ator.

Figura 31 - algoritmo de movimento do ator.



Fonte: Acervo do autor.

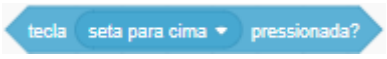
O próximo desafio dos estudantes era fazer com que o ator pulasse para frente ou para trás ao mesmo tempo em que as setas para cima e para direita ou para cima e para esquerda fossem pressionadas, fazendo um movimento semelhante a uma parábola. Inicialmente, utilizando o evento 

, e

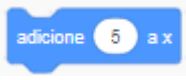
iniciando o algoritmo com o comando 

(trabalhado anteriormente) e o comando 

, os estudantes pensaram em utilizar o operador 

para inserir os sensores 

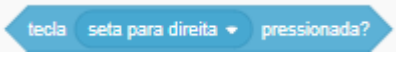
e 

, já que o movimento pretendido seria para cima e para frente ao mesmo tempo, junto com o bloco 

. Identificamos um raciocínio lógico presente na tentativa de programação dos estudantes, pois ao perceber que o movimento deveria ser para frente e para cima, eles concluem que as teclas que programam essas ações devem ser pressionadas ao mesmo tempo. Inicialmente, os estudantes gostaram do resultado, contudo perceberam que, para que o movimento acontecesse, não era possível soltar a tecla seta para cima, já que o ator parava de se movimentar para frente mesmo com a tecla seta para frente pressionada, voltando a ir para frente quando o botão seta para cima voltava a ser pressionado. Analisando o código desenvolvido, os estudantes perceberam a origem desse problema, identificando que o operador 

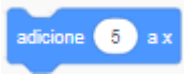
obrigava que as duas teclas fossem pressionadas ao mesmo tempo para que o movimento acontecesse. É possível identificar que os estudantes compreenderam o funcionamento desse operador lógico, pois perceberam que o algoritmo não continuava funcionando se uma das teclas era solta no processo, concluindo que o bloco operador 

limitava o algoritmo, podendo ser utilizado em apenas um caso, as duas teclas pressionadas juntas.

Logo, por tentativa, os estudantes retiraram o operador e deixaram apenas o sensor 

inserido no bloco 

, junto ao bloco



para observar o que acontecia. Surpreendentemente para eles, o ator movia-se da forma como tinham pensado.

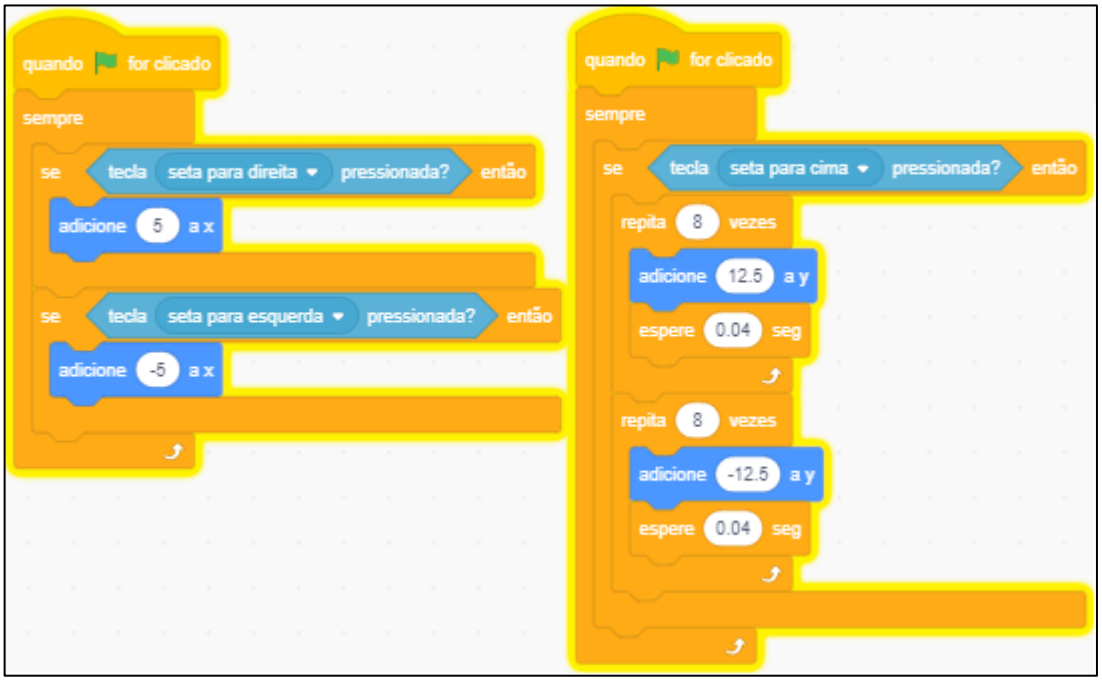
O que causou estranheza é que, no algoritmo anterior, já havia essa programação para ir para frente e para trás, porém não era realizada durante o salto. Ou seja, os estudantes criaram um algoritmo de movimento horizontal separado do algoritmo que programava o salto mas que também já tinha uma programação de movimento horizontal.

Os estudantes levantaram a hipótese de que a programação para saltar pressionando a tecla seta para cima, dentro da mesma programação que possibilita o ator se movimentar na horizontal, poderia interferir no salto para frente e para trás. Sendo assim, decidiram fazer o teste de inserir a programação do salto em outro algoritmo, excluindo o novo algoritmo de movimento horizontal. Feito isso, o ator continuou fazendo o movimento esperado por eles. A situação não foi compreendida pelos estudantes, ou seja, o porquê da interferência do algoritmo do salto inserido no algoritmo do movimento horizontal, porém foi possível observar a reação de satisfação dos estudantes A e M ao concluir essa programação inédita para eles.

Apesar dos estudantes demonstrarem uma tomada de consciência da programação dos movimentos vertical e horizontal, a programação do salto que se assemelha a uma parábola não foi totalmente compreendida no processo de construção, obtendo um sucesso a partir de uma tentativa, ou seja, uma ação. Porém, observamos que as ações não são puras, pois houve antecipações iniciais por parte dos estudantes, impulsionados por conceitos já compreendidos. Esse processo é descrito por Piaget (1978) como influências da conceituação sobre ação, na qual a conceituação oferece um reforço para a ação. Vemos que os estudantes compreendem os conceitos por detrás dos algoritmos, porém não compreendem todos os motivos, particularmente o porquê dos algoritmos juntos não funcionarem, mas, separadamente, sim.

Na Figura 32, são apresentados os algoritmos de movimento do ator principal programados pela dupla, assim como algumas adaptações feitas por eles em relação a valores de tempo e movimento.

Figura 32 - Algoritmos de movimento horizontal separados do algoritmo de salto.



Fonte: Acervo do autor.

Percebemos que o algoritmo de movimento, tanto horizontal como vertical, pode ser resgatado dos encontros anteriores, tendo esses algoritmos como base para a formação de outros (esquemas em desenvolvimento e já desenvolvidos). Observamos um avanço dos estudantes em relação à programação, identificando situações em que pensamento antecede a ação na maior parte do tempo (PIAGET, 1978). Observamos também a Matemática já conhecida por eles sendo utilizada, tanto nas operações com números racionais, como também na proporcionalidade, nesse caso, cálculos com grandezas diretamente e inversamente proporcionais.

5.7.3 Troca de cenários: Tela inicial, tela de *game over* e o *bug* inesperado

Nesse episódio, analisaremos a programação da tela inicial e tela de *game over* feita pelo estudante A e a correção de um *bug* na sua programação. Podemos observar que esta programação foi desenvolvida pelo estudante fora do ambiente da oficina, evidenciando o interesse e o engajamento do estudante em relação aos desafios da programação. Para Papert (1978, p. 112), realizar uma atividade que exerça as próprias escolhas dos estudantes “mobiliza seus próprios pontos fortes e

sua própria confiança numa situação difícil". Sendo assim, percebemos na atitude do estudante A a sua satisfação em participar das oficinas e como o uso do Scratch se caracteriza como uma forma de encarar situações intelectualmente difíceis e/ou desafiadoras.

Inicialmente, vamos explicar o que foi feito pelo estudante. Ao iniciar o jogo na bandeira verde, é apresentada uma tela inicial com a imagem do personagem *Yoshi* e um botão com a palavra "jogar" piscando em azul e laranja, como mostra a Figura 33. Para iniciar o jogo, basta clicar com o *mouse* no botão "jogar".

Figura 33 - Tela inicial do jogo.

Fonte: Acervo do autor.

Para compreender o pensamento do estudante A na sua programação, o pesquisador fez algumas perguntas para que o estudante pudesse explicar os mecanismos utilizados para sua produção.

Pesquisador: Como foi a programação para fazer o botão piscando?

Estudante A: Mudando a fantasia dele. O botão é um personagem (ator) e depois que clica ele esconde. Depois de clicar, ele esconde e vai para o próximo cenário (do jogo).

Pesquisador: E para a troca de cenário?

Estudante A: Existe um bloco que muda o cenário, então coloquei esse bloco depois do evento "quando este ator for clicado" e coloquei para esconder depois disso.

Pesquisador: Certo! E por que tu colocou o bloco "pare outros scripts no ator"?

Estudante A: Quando o cenário trocava ele sumia por um tempo e voltava a piscar, então achei esse bloco que parava ele.

Pesquisador: E por que tu não usou o bloco "pare todos"?

> Estudante A: Eu coloquei esse inicialmente, mas o cara (ator principal) não se mexia, daí testei esse e funcionou, já que eu queria que apenas o botão parasse.

Podemos perceber que a programação do ator “botão jogar” (Figura 34) possui traços semelhantes com algumas ferramentas utilizadas durante a oficina, como a troca de fantasias para o botão piscar. Porém, percebemos a exploração de outro bloco de evento, no qual não havíamos utilizado até o momento, que se trata do bloco 

, assim como os blocos de trocas de cenário, evidenciando autonomia do estudante A na exploração dos blocos do Scratch, possivelmente motivada pelo interesse pessoal em desenvolver o jogo, convergindo com a ideia de Papert (1978) na qual sugere que a realização de um projeto pessoal proporciona um desejo de ir além, o que fica evidente em seu livro *A Máquina das Crianças* ao relatar sua experiência na história de aprendizagem “Maria Constrói uma casa” (PAPERT, 1978, p. 106). Vemos também a utilização do bloco sempre, no qual foi incorporado nas estruturas dos estudantes como um bloco importante para iniciar um algoritmo.

Figura 34 - Programação do botão para iniciar o jogo.

Fonte: Acervo do autor.

Já na tela de *game over* (Figura 35), o estudante A inseriu a pergunta “Recomeçar?” e logo abaixo dois atores  de e  de para representar “sim”

ou “não” respectivamente, sendo que o primeiro botão quando selecionado direcionava o jogador diretamente no cenário do jogo, enquanto que o segundo direcionava o jogador para a tela inicial (já que uma possível ideia futura seria inserir outros recursos na tela inicial, como um menu de opções).

Figura 35 - Tela de *game over*.

Fonte: Acervo do autor.

Os botões da tela de *game over* tiveram programação semelhante ao botão da tela inicial, mostrando uma generalização nos processos de programação desses atores, com as devidas modificações para cada um deles, como o uso do bloco

e a retirada do bloco que inicia o número de vidas do ator.

Ao analisar o comportamento do estudante A nas programações mencionadas, verificamos evidências de que os estudantes já programam sem a necessidade de fazer testes, antecipando suas ações de antemão, fato que Piaget (1978, p. 173) considera como “uma possibilidade de escolha entre meios diferentes, sem limitar-se mais às regulações automáticas através de simples correções compensadoras”, sabendo o que cada bloco utilizado executa, revelando a compreensão de suas funcionalidades. Na Figura 36 vemos a programação dos dois botões.

Figura 36 - Programação dos botões da tela de *game over*.

Fonte: Acervo do autor.

No processo de testes para observar se a programação da tela de *game over* estava correta, os estudantes se deparam com um *bug* ao clicar no botão ✔, já que o jogo ao reiniciar travava o ator principal e não permitia a criação dos clones dos inimigos. Verificando a utilização do botão ✘, também foi observado um *bug* em que, ao ir para a tela inicial do jogo, ao clicar no botão jogar, a tela inicial não mudava para tela do jogo, porém aparecia o ator na tela. Os estudantes do grupo passaram a analisar as duas programações, e a estudante M sugeriu retirar o comando sempre:

> Estudante M: Será que aquele comando sempre não está atrapalhando (referindo-se ao comando inserido na programação do ator ✔ logo depois do bloco 
>
>)? Será que não estamos dizendo para ele (computador) que é para sempre mudar a vida para três e mudar o cenário?

Assim, o estudante A retirou o comando sempre da programação e fez o teste para saber se o problema estava solucionado, sendo observado a eficácia da retirada do bloco.

Estudante A: Não acredito que foi o “sempre”, pois sempre utilizamos o bloco “sempre”.

Pesquisador: Pois é, agora sabemos que devemos cuidar esse comando também quando não utilizá-lo.

Observando essa situação, vemos uma mudança nas verdades provisórias dos estudantes, pois já que o bloco “sempre” foi utilizado em praticamente todos os algoritmos para sugerir uma ação aos atores, observar esse *bug* causado pelo bloco sempre foi desconcertante para eles, porém de muita serventia para que eles pudessem ter mais cuidado nas utilizações dos blocos e, consequentemente, poder antecipar de forma mais eficiente as ações da máquina a partir dos algoritmos criados. Piaget (1978, p. 182) cita que, em caso de fracasso deve-se: “primeiramente encontrar os meios para corrigir a ação [...] e, em seguida, chegar a compreender a razão dos fracassos e (mas talvez com intervalos) a dos sucessos”. Sendo assim, os estudantes tiveram que reorganizar seus esquemas e estruturas relativas ao comando sempre, pois segundo Piaget (1978, p. 182), apesar das suas antecipações incluindo o comando sempre de forma errônea em seu algoritmo, “uma antecipação [...] é apenas uma inferência a partir de informações anteriores, o que não implica em finalidade”. Ou seja, o uso indevido de um bloco que as ações conceituaram o seu propósito, provocou uma volta para um patamar inferior na conceituação do bloco sempre, porém possibilitou novas ações para que as antecipações tenham sucesso em sua finalidade.

Outro ponto observado, foi o posicionamento da estudante M em relação à sua leitura do algoritmo que causava o problema na programação. Percebemos que, apesar de ter sido apenas um teste, não obtendo a certeza do que pensava, a leitura foi suficiente para observar que poderia ter algo errado no código programado, ou seja, as ideias dos estudantes e a forma como foram implementadas não estavam correspondendo. Para Papert (1985, p. 141), “os erros são benefícios porque levam os estudantes a estudarem o que aconteceu, a interpretar o que deu errado, e, através do entendimento, corrigi-lo”. Além disso, o processo de identificação e correção de erros é uma habilidade fundamental para o desenvolvimento do pensamento computacional.

Realizando um contraste com as primeiras experiências de programação realizadas no primeiro dia da oficina, quando era notável uma autonomia da ação, percebemos os avanços obtidos no processo de comunicação entre estudante e máquina, em que o estudante consegue perceber a forma com que a máquina interpreta o encadeamento de blocos (indicando compreensão sobre a noção de fluxo de algoritmos), podendo assim corrigir de forma consciente o código programado, construindo esquemas para uma melhor antecipação da ação aplicada no computador, evidenciando uma ultrapassagem da conceituação sobre a ação. Ou seja, percebemos o fazer dando lugar ao compreender, que segundo Piaget (1978, p. 179), “consiste em isolar a razão das coisas, enquanto o fazer é somente utilizá-las com sucesso, o que é, certamente, uma condição preliminar da compreensão”.

5.7.4 O ataque do personagem: Mudança de estratégia.

Nesse episódio, analisaremos as estratégias utilizadas pelos estudantes para programar as últimas ações do jogo. Na programação realizada até o momento, o jogo já contava com tela de início, tela de game over, um personagem principal com seus movimentos para se locomover na tela, inimigos (que foram programados com a utilização dos recursos vistos durante as oficinas, assim como o contato entre eles e o personagem principal) e contagem de vidas. Sendo assim, para o grupo, faltava apenas a programação do ataque do personagem que, diferente das programações trabalhadas na oficina em encontros anteriores, não lançaria um novo ator, mas sim atacaria com a sua “língua” (assim como o personagem *Yoshi* atacava no jogo *Super Mario Word*). Ou seja, o próprio contato do ator faria com que o ator inimigo fosse eliminado.

A programação desse personagem ficou sob a responsabilidade dos estudantes S e I, que observaram a exigência de uma estratégia diferente da que foi vista anteriormente, visto que o contato do ator principal determinaria se ele seria atingido ou se ele iria atingir o inimigo. Sendo assim, a primeira estratégia dos estudantes S e I foi a de diferenciar o personagem a partir das fantasias (fantasia “o cara”, personagem em posição de repouso, poderia ser atingido pelo inimigo, fantasia “atacando”, personagem em posição de ataque, atingiria o inimigo).

Para a programação da fantasia do ator, foram utilizados os comandos básicos, já vistos na oficina, formados pelo algoritmo da Figura 37.

Figura 37 - Tentativa de programação do ataque do ator.

Fonte: Acervo do autor.

Infelizmente, os estudantes não encontraram (em tempo) uma forma de programar a identificação dos inimigos com as fantasias, fazendo com que o personagem fosse atingido independente da fantasia. Logo, essa ideia foi descartada pelos estudantes. Outro motivo para a dupla descartar essa ideia foi argumentado pela estudante I, assim como a solução para o problema em discussão com o estudante S, como mostra diálogo a seguir.

Estudante I: Se o clone tocar na fantasia de ataque ele some, não importando onde ele encoste, mas ele deveria sumir apenas quando ele encostasse na língua.

Estudante S: Então teríamos que dar um jeito de separar os dois.

EstudanteI: Pensei em fazer a língua como um novo ator.

Estudante S: Mas acredito que vá dar muito trabalho.

Estudante I: Eu pensei aqui em esconder ele e fazer ele aparecer quando a tecla espaço fosse clicada. Colocamos a língua para seguir o ator, fazendo os mesmos movimentos, aparecendo apenas quando o ator troca de fantasia.

Sendo assim, o estudante S ficou responsável por editar a fantasia de ataque do personagem principal, retirando a sua língua, enquanto a estudante I programava o novo ator “língua”. Percebemos, nessa situação, uma decomposição do problema por parte da dupla, que procurou solucionar a situação separando o problema a ser

resolvido do problema que já estava solucionado. Novamente, vemos mais um pilar do pensamento computacional nas estratégias dos estudantes. Brackmann (2017) exemplifica a decomposição, evidenciando sua importância na programação no entendimento do funcionamento de uma bicicleta.

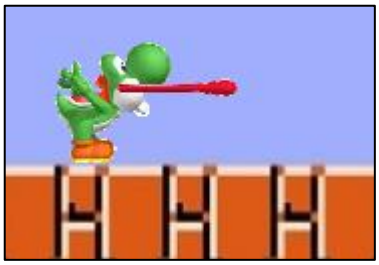
> Quando a decomposição é aplicada a elementos físicos, como por exemplo a bicicleta, a manutenção torna-se mais fácil quando é possível modularizar suas partes. Caso contrário, se o item em questão fosse desenvolvido em uma única peça, seu reparo se tornaria muito difícil e a forma de consertá-lo seria trocada por outra. O mesmo acontece com o desenvolvimento de programas. (BRACKMANN, 2017, p. 35)

O primeiro passo foi programar o personagem língua de forma que ele se mantivesse escondido e aparecesse quando a tecla espaço fosse pressionada. A programação foi iniciada com o bloco 

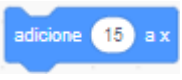
, seguido do bloco 

, deixando a língua escondida. Posteriormente, a estudante I utilizou o algoritmo que faria a língua aparecer em 0,6 segundo em que o personagem ficava em posição de ataque no algoritmo anterior, utilizando 0,1 segundo para que ele ficasse em posição e 0,5 segundo para que a língua aparecesse. Porém, ao testar a programação realizada, a estudante I percebeu que o ator língua ficava em uma posição mais atrás do que deveria, ficando posicionada por cima do personagem principal, como mostra a Figura 38.

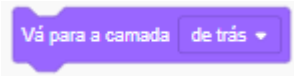
Figura 38 - Representação do ataque do ator em desenvolvimento.



Fonte: Acervo do autor.

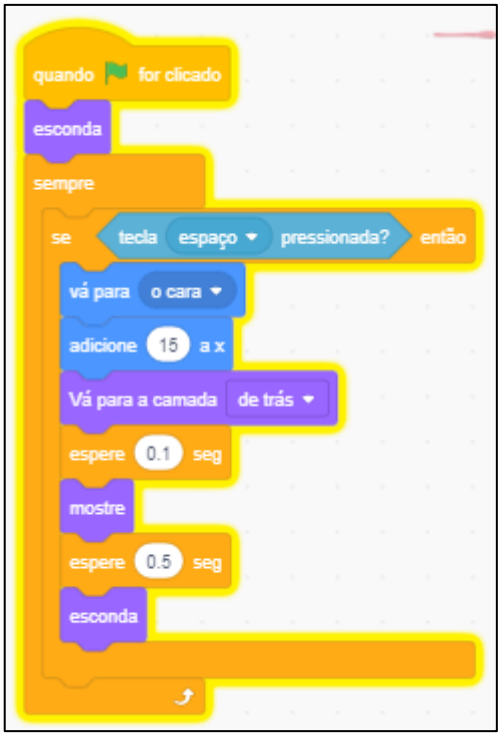
Evidenciando compreensão sobre a localização do objeto (ator) no plano a partir dos esquemas e conceitos já construídos em encontros prévios, a estudante I adicionou o bloco 

, antecipando com sucesso a posição do ator após a modificação. Outro bloco utilizado para solucionar o problema da língua, que deveria

ficar à frente do personagem principal, foi 

, obtendo o seguinte algoritmo (Figura 39).

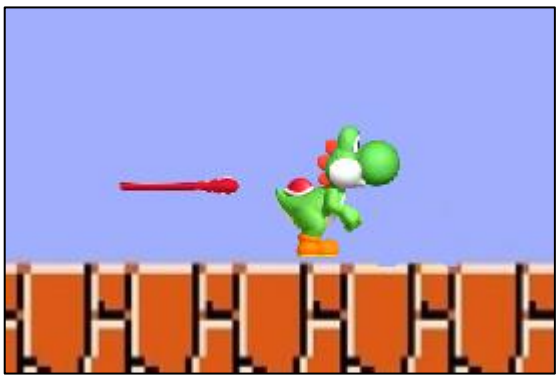
Figura 39 - Algoritmo que programa a língua do ator.



Fonte: Acervo do autor.

Seguindo para a fase de testes, um novo problema surgiu: a movimentação do ator enquanto ele atacava, ou seja, caso uma seta que movimentasse o ator na horizontal fosse pressionada, o ator imediatamente saía da posição de ataque, deixando a língua para trás, como ilustra a Figura 40.

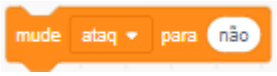
Figura 40 - Problema ocorrido durante a programação.



Fonte: Acervo do autor.

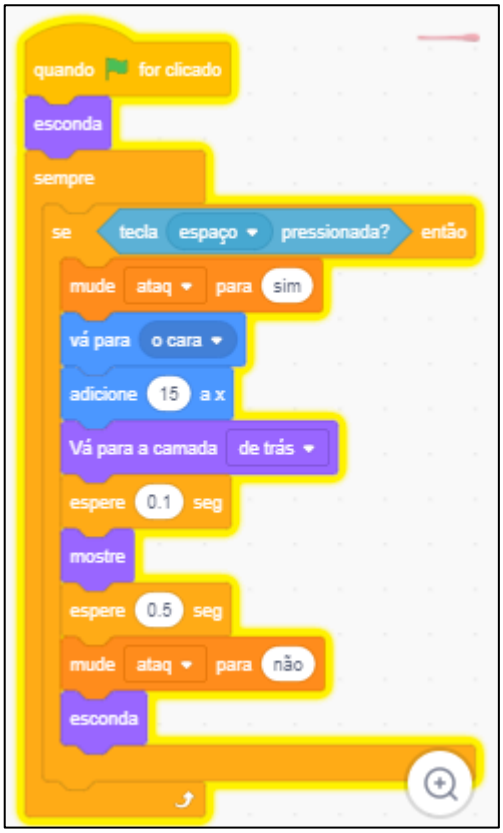
Os estudantes solicitaram auxílio para o pesquisador para resolver o problema. Não sabendo como solucionar o problema de imediato, o pesquisador lembrou da possibilidade de criar variáveis como condições para que um algoritmo seja executado ou não e sugeriu ao grupo. Resgatando ideias sobre o uso de variáveis nos projetos realizados na oficina, a estudante I criou a variável 

, para determinar quando a língua deveria estar escondida e quando ela deveria aparecer, utilizando o conceito de condicional e de variável, já que havia a necessidade de condicionar o momento em que a língua apareceria e, ao mesmo tempo, ao aparecer, era o algoritmo que determinava que os movimentos do ator deveriam ser inutilizados. Nesse caso, foram incluídos na programação (Figura 41) em que a língua aparece, os blocos 

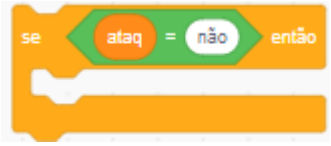
e 

, determinando uma condição para a execução do algoritmo para que a língua fosse utilizada, dando uma possibilidade de condição para a execução de outros algoritmos.

Figura 41 - Programação com o recurso variável inserido.



Fonte: Acervo do autor.

A ideia dessa variável era estabelecer uma condição para o algoritmo de movimento apenas para quando o personagem não estava atacando, inserindo o bloco 

antes dos algoritmos de movimento. Sendo assim, os algoritmos de movimento do personagem principal tiveram a seguinte alteração como mostra a Figura 42.

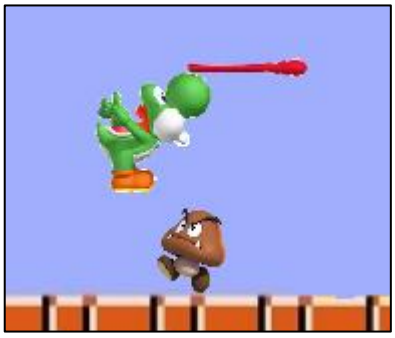
Figura 42 - Algoritmo de movimento do personagem.

Fonte: Acervo do autor.

Dessa forma, o algoritmo que movimenta o ator horizontalmente é executado apenas quando o personagem não está atacando e, quando o personagem está com o ataq “sim”, o algoritmo fica desativado, até que volte a ficar ataq “não”. Observamos a compreensão dos estudantes em relação à função do recurso variável, que pode assumir valores lógicos sim ou não nesse caso, determinando a condição para a execução dos algoritmos.

O último desafio dos estudantes em relação ao ataque do personagem foi relacionado ao salto, pois quando o ator saltava, ao utilizar o ataque, a língua não acompanhava o ator, ficando parada em relação ao solo, como mostra a Figura 43.

Figura 43 - *bug* na programação de ataque durante o salto.



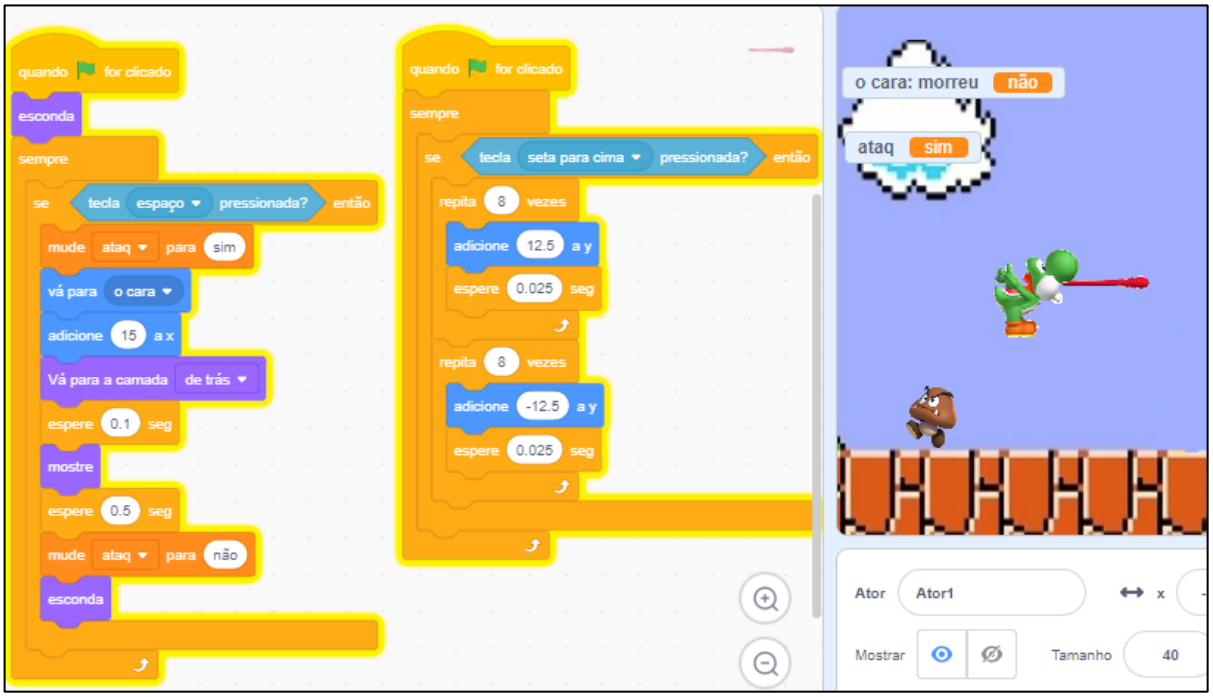
Fonte: Acervo do autor.

Como os estudantes queriam que o personagem tivesse a possibilidade de saltar enquanto atacava, eles não poderiam utilizar a mesma estratégia para solucionar esse problema, já que iria inibir o ator de se movimentar verticalmente enquanto ataca. O estudante S conjecturou a possibilidade de colocar o mesmo algoritmo que programava o salto do ator principal no ator língua, evidenciando compreensão a partir de reconhecimento de padrões entre os movimentos dos dois atores. Com o mesmo movimento, basta utilizar o mesmo algoritmo do ator principal para o ator língua, generalizando as ações de movimento.

> Algoritmos que são responsáveis pela solução de algum problema específico podem ser adaptados para resolver uma variedade de problemas similares. Sempre que necessário, o algoritmo pode aplicar uma solução de forma generalizada (BRACKMANN, 2017, p. 36).

Dessa forma, em uma tentativa inspiradora do estudante S, o grupo solucionou o problema, programando o ataque do personagem principal corretamente (Figura 44).

Figura 44 - Programação final do ator língua.



Fonte: Acervo do autor.

Esse episódio finaliza o experimento, em que pudemos observar as estratégias utilizadas pelos participantes da pesquisa na programação de uma fração do jogo pensado por eles, contendo os principais mecanismos para que ele pudesse ser considerado jogável pelos estudantes, com uma possibilidade futura de continuidade do projeto. No próximo capítulo, vamos apresentar as considerações finais, em que procuramos responder às questões adicionais para contribuírem com a resposta da questão norteadora, assim como possibilidades para encaminhamentos futuros dessa pesquisa.

6 CONSIDERAÇÕES FINAIS

Os dados produzidos durante a pesquisa buscam responder à questão norteadora: *Quais as contribuições do uso da programação no software Scratch no desenvolvimento de conceitos matemáticos e de elementos do pensamento computacional com estudantes de Ensino Fundamental?* Para responder a essa pergunta, é necessário observar e analisar as ações dos participantes durante o processo de programação. Nessa perspectiva, buscamos inicialmente responder às questões adicionais a partir das análises dos resultados da pesquisa.

Como os estudantes desenvolvem seu pensamento matemático a partir da ação de programar? Para esse questionamento, as inspirações do método clínico de Piaget foram fundamentais. Apesar do método não ter sido aplicado em um ambiente favorável, já que o método clínico se caracteriza pela sua aplicação nas ações de um indivíduo por vez (DELVAL, 2002), pudemos identificar e compreender as relações feitas pelos participantes durante as situações propostas e da construção de seus projetos. A partir dos registros recolhidos durante a pesquisa, percebemos nas comunicações entre os participantes e os nossos questionamentos durante o desenvolvimento das ações dos estudantes, foi possível observar o pensamento matemático em diferentes momentos da programação das ações dos personagens.

Ao pensarem em um movimento, os participantes precisaram utilizar blocos de comandos para representá-lo na programação, elaborando um algoritmo ou uma sequência de algoritmos que resolvem a situação. Sendo assim, ao abstrair os dados principais do movimento e implementá-los em uma sequência de blocos, os participantes estão utilizando conceitos matemáticos para essas representações. Ao utilizar números positivos e negativos, por exemplo, para se locomover horizontalmente ou verticalmente, assim como a animação de um salto, percebemos os conceitos de localização e movimento de coordenadas no plano cartesiano sendo utilizados na programação do ator no palco. Ou também, utilizar a proporcionalidade para identificar a relação entre uma repetição de movimentos em um determinado tempo, observando grandezas diretamente ou inversamente proporcionais para solucionar o problema do salto de um ator, desconsiderando uma ideia inicial na qual não era compreendida e utilizando a criatividade juntamente com um conhecimento previamente existente.

Ao conseguir representar essas ações a partir de combinações de códigos formando um algoritmo, os conceitos utilizados pelos participantes são facilmente reconhecidos e aplicados podendo ser utilizados em outras programações ou em situações fora desse ambiente.

Quais conceitos matemáticos podem ser construídos e utilizados durante a programação no software Scratch? Particularmente, durante a oficina preparada para essa pesquisa, evidenciamos como ponto de partida a aprendizagem do conceito de coordenadas cartesianas, o que foi de fundamental importância para os participantes compreenderem o movimento e a localização dos atores no palco. Na iniciativa de movimentar o ator, vemos o uso dos blocos de giros que favorecem o uso do conceito de ângulos.

O conceito de condicional da lógica matemática, que tem grande relevância na área da programação, também pôde ser contemplada como consequência do ambiente de programação em que os estudantes trabalharam. O conceito de condicional pode também ser trabalhado adjacente aos operadores de igualdade e desigualdade, oportunizando na prática o uso desses conceitos que, geralmente, são utilizados em equações e inequações matemáticas.

O conceito de velocidade, mesmo que de forma intuitiva, juntamente com a ideia de grandezas diretamente e inversamente proporcionais, foram identificados durante o processo de programação. A utilização dos números do conjunto dos inteiros, tanto no processo de movimento dos atores, em que também foi possível perceber o conceito de módulo de forma implícita no movimento dos atores, como nas operações matemáticas para a contagem de número de vidas, que ao diminuir a quantidade de vidas deveria se usar o bloco que acrescenta um número, neste caso, um número negativo. Assim como o uso dos números inteiros, a necessidade do uso dos números decimais na utilização do tempo para a troca de fantasia, possibilitando a fragmentação do tempo em espaços muito pequenos, consequentemente, possibilitando uma reflexão sobre a noção de infinito.

Os conceitos matemáticos desenvolvidos com o uso do recurso “variável” do Scratch se apresentaram de duas formas. A primeira, como conceitos de lógica, na qual esse recurso pode se apresentar como uma sentença de verdadeiro e falso (quando a sentença é verdadeira, o computador entende que o algoritmo deve funcionar, já quando for falsa, o computador espera até que ela volte a ser verdadeira). A segunda, como um valor numérico para a contagem de vidas, em que

a vida soma um valor negativo até o um determinado limite, ou seja, acabarem as vidas.

Podemos observar alguns desses conceitos sendo utilizados em mais de um momento durante a oficina, o que nos faz perceber que os conceitos matemáticos sempre estiveram presentes de forma direta ou indireta, consciente ou de forma intuitiva, mas que oportunizou o seu uso na prática da programação com o Scratch. Dessa forma, retomando a afirmação de Resnick (2007), podemos salientar que o *software* Scratch possibilitou o desenvolvimento de conceitos matemáticos.

O pensamento computacional desenvolvido durante o processo de programação pode contribuir para o desenvolvimento de conceitos matemáticos? Visto que o pensamento computacional combina o pensamento matemático com a engenharia para o desenvolvimento de habilidades computacionais para a resolução de problemas (WING, 2006), baseando-nos nos quatro pilares do pensamento computacional de Brackmann (2017), o pensamento computacional contribuiu para o desenvolvimento do pensamento matemático dos estudantes durante os processos de programação, começando pela decomposição de problemas, em que eles subdividiam os problemas para resolvê-los separadamente, observando características semelhantes entre os problemas e criando padrões entre eles. Posteriormente, observamos os estudantes abstraindo ideias pertinentes e criando noções de como resolver o problema, utilizando os blocos do software para concretizar essas ideias em algoritmos, muitas vezes utilizando-se da tentativa e erro para que pudessem construir um algoritmo possível, mas também compreendendo a utilidade dos blocos e desses algoritmos para situações semelhantes, entendemos que o pensamento computacional contribui para desenvolvimento das habilidades matemáticas já que auxiliaram no processo de programar decompondo, abstraindo, generalizando e criando algoritmos.

O pensamento computacional está ligado à habilidade de resolver problemas. Nesse caso, o pensamento computacional possibilita o uso de conhecimentos já adquiridos para isso e, como pudemos observar nos resultados da pesquisa, muitos desses problemas foram solucionados a partir do uso de conceitos matemáticos, o que fortalece a relação do pensamento computacional, não só com o desenvolvimento, mas também, com a consolidação de conceitos matemáticos durante a programação. Nesse sentido, o ato de programar oportuniza o desenvolvimento do pensamento computacional durante as ações dos participantes

em um espaço pedagógico envolvendo programação que, além de desenvolver conceitos matemáticos, oportuniza também que o estudante desenvolva as características inerentes ao pensamento computacional. Sendo assim, concluímos que o pensamento computacional, observado ao olhar dos quatro pilares citados anteriormente, contribui para o desenvolvimento de habilidades matemáticas.

Partindo para a resposta da pergunta norteadora da pesquisa, tendo as respostas das questões adicionais como suporte, podemos concluir que o software Scratch contribui para o desenvolvimento de habilidades matemáticas tanto diretamente como indiretamente, possibilitando a construção de novos conceitos assim como o uso de conceitos já conhecidos, tanto para contribuir no ato de programar quanto para serem consolidados em situações práticas. Sobre o software Scratch e o pensamento computacional, percebemos que a programação no Scratch contribui para o desenvolvimento do pensamento computacional durante a programação de forma intuitiva e natural. Observamos, por exemplo, no último episódio em que os quatro pilares do pensamento computacional estavam presentes durante as ações dos estudantes, decompondo o problema, identificando padrões de códigos que foram utilizados anteriormente e facilitaram a resolução dos problemas, abstraindo os dados mais importantes do problema para, por fim, desenvolver os algoritmos que reproduzem as ações desejadas pelos estudantes programadores.

Nesse contexto, vemos o uso do Scratch como um ambiente que possibilita também o desenvolvimento de conceitos matemáticos, diferente do que percebemos ao analisar algumas pesquisas relacionadas ao Scratch, apropriando-se dessa ferramenta para o auxílio de conceitos já trabalhados anteriormente. Da mesma forma, vemos como uma ferramenta que permite o desenvolvimento do pensamento computacional para que as ideias possam ir além da realidade, abstraindo essas ideias para que possam ser reproduzidas por meio de códigos e algoritmos, convergindo com o pensamento de Wing (2006, p. 35): “ser livre para construir mundos virtuais nos permite engenhar sistemas além do mundo físico”.

A nossa pesquisa teve como proposta buscar como os estudantes compreendem os conceitos matemáticos no contexto da programação, assim como observar o pensamento computacional auxiliando nesse desenvolvimento, procurando contribuir com estudos na área de Educação Matemática e programação com crianças e jovens como meio de aprendizagem. Como possibilidade de

pesquisas futuras, pensamos em analisar os processos de colaboração e cooperação dos estudantes na programação. Sobre aprimoramentos na oficina, sentimos falta de um momento para que os participantes pudessem apresentar os seus projetos. Fato que seria indispensável em uma próxima aplicação.

Com relação a minha formação como professor de Matemática, quero salientar a importância do desenvolvimento desse trabalho. Tanto os resultados da pesquisa, como também o conhecimento construído, puderam reafirmar meu entendimento sobre a relevância do uso da programação no ambiente escolar, assim como conhecer e defender o pensamento computacional como uma habilidade importante para todos na atualidade. Sendo assim, encerramos essa pesquisa convidando o leitor a explorar esse contexto de educação que propõe ao estudante desenvolver Matemática de forma desafiadora, autônoma e relevante.

REFERÊNCIAS

AZEVEDO, G. T. de, MALTEMPI, M. V., MACHADO, J. P. R., LYRA-SILVA, G. M. V. Produção de Games nas Aulas de Matemática: por que não?. **Acta Scientiae**, v. 20, n. 5, p. 950, 2018.

BOGDAN, R.; BIKLEN, S. **Investigação Qualitativa em Educação: uma introdução à teoria e aos métodos.** Porto: Porto Editora. 1991.

BONA, A. S. de. **Espaço de aprendizagem digital da atemática: o aprender a aprender por cooperação**. Tese (doutorado informática na educação) – Programa de Pós-Graduação em Informática na Educação do Centro Interdisciplinar de Novas Tecnologias na Educação. Universidade Federal do Rio Grande do Sul. Porto Alegre. 2012.

BOUCINHA, R. M. **Aprendizagem do Pensamento Computacional e Desenvolvimento do Raciocínio**. Tese (doutorado informática na educação) – Programa de Pós-Graduação em Informática na Educação do Centro Interdisciplinar de Novas Tecnologias na Educação. Universidade Federal do Rio Grande do Sul. Porto Alegre. 2017.

BRACKMANN, C. P. **Desenvolvimento do Pensamento Computacional Através de Atividades Desplugadas na Educação Básica.** Tese (doutorado informática na educação) – Programa de Pós-Graduação em Informática na Educação do Centro Interdisciplinar de Novas Tecnologias na Educação. Universidade Federal do Rio Grande do Sul. Porto Alegre. 2017.

BRACKMANN, C. P.; COUTO BARONE, D. A.; CASALI, A.; HERNÁNDEZ, S. **Pensamento Computacional. Panorama nas Américas.** In F.J. García-Peñalvo & J.A. Mendes (Eds) XVIII Simposio Internacional de Informática Educativa, SIIE. Ediciones Universidad de Salamanca, Espanha, p. 197-202, 2016.

BRASIL. **Base Nacional Comum Curricular: Educação é a Base**. Ministério da Educação: Brasília. 2018. Disponível em: <http://basenacionalcomum.mec.gov.br/images/BNCC_EI_EF_110518_versaofinal_site.pdf>. Acesso em: 21 jan. 2021.

BRENNAN, K., RESNICK, M. **New frameworks for studying and assessing the development of computational thinking.** Paper presented at the Annual American Educational Research Association meeting, Vancouver, BC, Canada, 2012.

CUNY, J., SNYDER, L., WING, J. M. **Demystifying computational thinking for noncomputer scientists**. Unpublished manuscript in progress, referenced. 2010. in: <https://www.researchgate.net/publication/319959476_Demystifying_computational_thinking>

CURCI, A. P. F. **O *Software* de Programação Scratch na Formação Inicial do Professor de Matemática Por Meio da Criação de Objetos de Aprendizagem**. Dissertação (Mestrado em Ensino de Matemática) – Programa de Pós-Graduação

em Ensino de Matemática, Universidade Tecnológica Federal do Paraná, UTFPR, Londrina. 2017.

DELVAL, J. **Introdução à prática do Método Clínico**: descobrindo o pensamento das crianças. Tradução de Fátima Murad. Porto Alegre: Artmed, 2002.

ELOY, A. A. da, LOPES, R. de D., ANGELO, I. M. Uso do Scratch no Brasil com objetivos educacionais: uma revisão sistemática. **RENOTE**, v. 15, n. 1, 2017.

GRAVINA, M. A., BASSO, M. V. A., Mídias Digitais na Educação Matemática. In: GRAVINA, M. A., BÚRIGO, E. Z., BASSO, M.V.A., GARCIA, V.C.V. (org.) **Matemática, Mídias Digitais e Didática - tripé para formação do professor de Matemática.** Porto Alegre: Evangraf, 2012.

GROVER, S.; PEA, R. **Computational Thinking in K-12: A Review of the State of the Field**. Educational Researcher, v. 42, n. 1, p. 38-43, 2013. Disponível em: <https://www.researchgate.net/publication/258134754_Computational_Thinking_in_K-12_A_Review_of_the_State_of_the_Field> Acesso em: 19/07/2019.

KENSKI, V. **Tecnologias e ensino presencial e a distância**. Campinas: Papirus, 2012

MALTEMPI, M. V. **Novas tecnologias e construção de conhecimento: reflexões e perspectivas**. Congresso Ibero-Americano de Educação Matemática, 2005.

MALTEMPI, M. V. Educação matemática e tecnologias digitais: reflexões sobre prática e formação docente/Mathematics education and digital technologies: Reflexions about the practice in teacher education. **Acta Scientiae**, v. 10, n. 1, p. 59-67, 2008.

MARJI, M. **Aprender a programar com Scratch**. Tradução: Lúcia Kinoshita. São Paulo: Novatec, 2014. 284 p.

NOTARE, M. R. **Comunicação e aprendizagem matemática on-line: um estudo com o editor científico ROODA exata**. Tese (doutorado informática na educação) – Programa de Pós-Graduação em Informática na Educação do Centro Interdisciplinar de Novas Tecnologias na Educação. Universidade Federal do Rio Grande do Sul. Porto Alegre. 2009.

PAPERT, S. **A máquina das Crianças: repensando a escola na era da informática**. Tradução de Sandra Costa. Porto Alegre: Artes Médicas, 1994.

PAPERT, S. **Mindstorms: Children, Computers, And Powerful Ideas.** New York: Basic Books, 1980.

PAPERT, S.; SOLOMON, C. **Twenty Things to do with a Computer**. Artificial Intelligence Memo 248, MIT AI Laboratory. Cambridge, MA, 1971. Disponível em: <https://dspace.mit.edu/bitstream/handle/1721.1/5836/AIM-248.pdf?sequence=2&isAllowed=y> Acesso em: 19/07/2019.

PEA, R. **Cognitive technologies for mathematics education.** In A.H. Schoenfeld (Ed.), Cognitive Science and Mathematics Education. p. 89–122, 1987. Hillsdale: Lawrence Erlbaum. <http://web.stanford.edu/~roypea/RoyPDF%20folder/A41_Pea_87b.pdf> Acesso on line em 28 dez. 2018.

PIAGET, J. **A Tomada de Consciência**. São Paulo: Melhoramentos, Editora da Universidade de São Paulo, 1977.

PIAGET, J. **Fazer e Compreender**. São Paulo: Melhoramentos, Editora da Universidade de São Paulo, 1978.

PIAGET, J. **Abstração Reflexionante**. Porto Alegre: Artes Médicas, 1995.

POLONI, L., SOARES, E. M do S., WEBBER, C. G. Pensamento Computacional no Ensino Médio: Práticas Mediadoras Utilizando a Linguagem Scratch. **RENOTE**, v. 17, n. 3, 2019.

QUEIROZ, V. S. **Contribuições da Linguagem Scratch Para o Ensino da Geometria**. Dissertação (Mestrado Profissional em Matemática em Rede Nacional) – Programa de Pós-Graduação Stricto Sensu em Matemática em Rede Nacional, Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, IFSP, São Paulo. 2018.

RESNICK, M. **Sowing the seeds for a more creative society**. Learning and Leading with Technology, p. 18–22, 2007.

RESNICK, M., MALONEY, J., MONROY-HERNÁNDEZ, A., RUSK, N., EASTMOND, E., BRENNAN, K., MILLNER, A., ROSENBAUM, E., SILVER, J., SILVERMAN, B., KAFAI, Y. **Scratch: Programming for all.** Comm. ACM 52, 11, 60–67. 2009.

ROCHA, K. C. da. **Programação em Scratch na sala de aula de matemática: investigações sobre a construção do conhecimento de ângulo**. Dissertação (Mestrado em Ensino de Matemática) – Programa de Pós-Graduação em Ensino de Matemática, Universidade Federal do Rio Grande do Sul, Porto Alegre. 2017.

ROCHA, K. C. da, BASSO, M. V. A., NOTARE, M. R. Aproximações teóricas entre Pensamento Computacional e Abstração Reflexionante. **RENOTE**, v. 18, n. 2, 2020.

ROMÁN-GONZÁLEZ, M.; PÉREZ, J. C.; CARMEN JIMÉNEZ-FERNÁNDEZ. **Test de Pensamiento Computacional: diseño y psicometría general**, 2015. Disponível em: <https://doi.org/10.13140/RG.2.1.3056.5521>. Acesso em: 20/01/2021.

SILVA, R. S. da. **Cadeias de Markov e modelagem Matemática: da abstração pseudo-empírica à abstração refletida com uso de objetos virtuais**. Tese (doutorado informática na educação) – Programa de Pós-Graduação em Informática na Educação do Centro Interdisciplinar de Novas Tecnologias na Educação. Universidade Federal do Rio Grande do Sul. Porto Alegre. 2015.

VALENTE, J. A. **Diferentes usos do computador na educação. Computadores e Conhecimento: repensando a educação,** p. 1-23, 1993. Disponível em: <http://www.nied.unicamp.br/sites/default/files/livros/livro-computadores-e-conhecimento.pdf> Acesso on line em 03 jan. 2019.

WING, J.M. **Computational thinking**. Communications of the ACM 49 (3), p. 33–35, 2006.

WING, J. M. Computational thinking and thinking about computing. **Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences**, v. 366, n. 1881, p. 3717-3725, 2008.

APÊNDICE

Apêndice A: Carta de apresentação

Porto Alegre, 23 de abril de 2019.

Prezado Professor Marcelo Figueiró

Diretor da Escola La - Salle Pão dos Pobres

O professor Bruno Silveira Corrêa **atualmente é mestrando regularmente matriculada no Programa de Pós-Graduação em Ensino de Matemática (PPGEMat) da Universidade Federal do Rio Grande do Sul.**

Como parte das exigências do PPGEMat, o mestrando está desenvolvendo a pesquisa intitulada: Programando Jogos com Scratch: Linguagem de Programação no Ensino Fundamental**, para a conclusão de sua dissertação, na qual é exigida para que possa adquirir o título de Mestre em Educação Matemática.**

A dissertação produzida deve resultar em material didático de qualidade que possa ser utilizado por outros professores de Matemática. Neste sentido, torna-se extremamente importante realizar experimentos educacionais e, por esta razão, estamos solicitando a sua autorização para que este trabalho possa ser desenvolvido na escola sob sua Direção.

Em caso de manifestação de sua concordância, por favor, registre sua ciência ao final deste documento, o qual está sendo encaminhado em duas vias.

Enquanto orientadora responsável, **reiteramos nosso compromisso ético com os sujeitos dessa pesquisa nos colocamos à disposição para quaisquer esclarecimentos durante e após a realização da produção de dados. Para tanto, deixamos à disposição o seguinte telefone de contato: (51)3308.6212 (Secretaria do PPGEMat), e o seguinte e-mail: marcia.notare@gmail.com.**

Agradecemos a sua atenção.

Cordialmente,

Márcia Notare

Professora do PPGEMat

Apêndice B: Termo de Consentimento Informado

TERMO DE CONSENTIMENTO INFORMADO

Eu, _____, R.G. _____, responsável pelo(a) aluno(a) _____, da turma _____, declaro, por meio deste termo, que concordei em que o(a) aluno(a) participe da pesquisa intitulada: **Programando Jogos com Scratch: Linguagem de Programação no Ensino Fundamental**, desenvolvida pelo(a) pesquisador(a) **Bruno Silveira Corrêa.** Fui informado(a), ainda, de que a pesquisa é coordenada/orientada por **Marcia Notare Meneghetti**, a quem poderei contatar a qualquer momento que julgar necessário, pelo e-mail **marcia.notare@gmail.com**.

Tenho ciência de que a participação do(a) aluno(a) não envolve nenhuma forma de incentivo financeiro, sendo a única finalidade desta participação a contribuição para o sucesso da pesquisa. Fui informado(a) dos objetivos estritamente acadêmicos do estudo, que, em linhas gerais, são:

- Aprimorar a aprendizagem dos alunos a partir do software Scratch, além de inseri-los na linguagem de programação do software, apresentando uma nova forma de se pensar Matemática.
- Construir conceitos de Matemática a partir das ações de programação no software Scratch.

Fui também esclarecido(a) de que os usos das informações oferecidas pelo(a) aluno(a) será apenas em situações acadêmicas (artigos científicos, palestras, seminários etc.), identificadas apenas pela inicial de seu nome e pela idade.

A colaboração do(a) aluno(a) se fará por meio de entrevista/questionário escrito etc, bem como da participação em oficina/aula/encontro/palestra, em que ele(ela) será observado(a) e sua produção analisada, sem nenhuma atribuição de nota ou conceito às tarefas desenvolvidas. No caso de fotos ou filmagens, obtidas durante a participação do(a) aluno(a), autorizo que sejam utilizadas em atividades acadêmicas, tais como artigos científicos, palestras, seminários etc, sem identificação. Esses dados ficarão armazenados por pelo menos 5 anos após o término da investigação.

Cabe ressaltar que a participação nesta pesquisa não infringe as normas legais e éticas. No entanto, poderá ocasionar algum constrangimento dos entrevistados ao precisarem responder a algumas perguntas sobre o desenvolvimento de seu trabalho na escola. A fim de amenizar este desconforto será mantido o anonimato das entrevistas. Além disso, asseguramos que o estudante poderá deixar de participar da investigação a qualquer momento, caso não se sinta confortável com alguma situação

Como benefícios, esperamos com este estudo, produzir informações importantes sobre Tecnologias Digitais na Educação Matemática, a fim de que o conhecimento construído possa trazer contribuições relevantes para a área educacional.

A colaboração do(a) aluno(a) se iniciará apenas a partir da entrega desse documento por mim assinado.

Estou ciente de que, caso eu tenha dúvida, ou me sinta prejudicado(a), poderei contatar o(a) pesquisador(a) responsável pelo telefone: **(51)XXXXX-XXXX** ou e-mail: **brunocorrea.mat@gmail.com**.

Qualquer dúvida quanto a procedimentos éticos também pode ser sanada com o Comitê de Ética em Pesquisa (CEP) da Universidade Federal do Rio Grande do Sul (UFRGS), situado na Av.Paulo Gama, 110 - Sala 317, Prédio Anexo 1 da Reitoria - Campus Centro, Porto Alegre/RS - CEP: 90040-060 e que tem como fone 55 51 3308 3738 e email etica@propesq.ufrgs.br

Fui ainda informado(a) de que o(a) aluno(a) pode se retirar dessa pesquisa a qualquer momento, sem sofrer quaisquer sanções ou constrangimentos.

Porto Alegre, _____ de _____ de _____.

Assinatura do Responsável:
Assinatura do(a) pesquisador(a):

Assinatura do Orientador da pesquisa:

Apêndice C: Termo de Assentimento Informado

TERMO DE ASSENTIMENTO INFORMADO

Eu, _____, aluno(a) da turma _____, declaro, por meio deste termo, que concordei em participar da pesquisa intitulada: **Programando Jogos com Scratch: Linguagem de Programação no Ensino Fundamental,** desenvolvida pelo(a) pesquisador **Bruno Silveira Corrêa.** Fui informado(a), ainda, de que a pesquisa é coordenada/orientada por **Marcia Notare Meneghetti**, professora acadêmica da Universidade Federal do Rio Grande do Sul (UFRGS).

Tenho ciência de que a minha participação não envolve nenhuma forma de incentivo financeiro, sendo a única finalidade desta participação, a contribuição para o sucesso da pesquisa. Fui informado(a) dos objetivos estritamente acadêmicos do estudo, que, em linhas gerais, são:

- Aprimorar a aprendizagem dos alunos a partir do software Scratch, além de inseri-los na linguagem de programação do software, apresentando uma nova forma de se pensar Matemática.

- Construir conceitos de Matemática a partir das ações de programação no software Scratch.

Fui também esclarecido(a) de que os usos das informações oferecidas por mim serão apenas em situações acadêmicas (artigos científicos, palestras, seminários etc.), identificadas apenas pela inicial de meu nome ou nome fictício e pela idade.

A minha colaboração se fará por meio de entrevista/questionário escrito etc, bem como da participação em oficina, em que serei observado(a) e minha produção analisada, sem nenhuma atribuição de nota ou conceito às tarefas desenvolvidas. No caso de fotos ou filmagens, obtidas durante a minha participação, autorizo que sejam utilizadas em atividades acadêmicas, tais como artigos científicos, palestras, seminários etc, sem identificação. Esses dados ficarão armazenados por pelo menos 5 anos após o término da investigação.

Cabe ressaltar que a participação nesta pesquisa não infringe as normas legais e éticas. Porém, para que não ocorram constrangimentos, estou ciente de que será mantido o anonimato das entrevistas. Além disso, estou ciente de que poderei deixar de participar da investigação a qualquer momento, caso não se sinta confortável com alguma situação.

Como benefícios, é esperado desde estudo, produzir informações importantes sobre Tecnologias Digitais na Educação Matemática, a fim de que o conhecimento construído possa trazer contribuições relevantes à educação.

Ciente também, de que minha colaboração se iniciará apenas a partir da entrega desse documento por mim assinado.

Porto Alegre, _____ de _____ de _____.

Assinatura do aluno:
Assinatura do(a) pesquisador(a):
Assinatura do Orientador da pesquisa:

APÊNDICE D: Produto Didático

O produto didático da pesquisa visa uma possibilidade de iniciação para quem deseja começar a utilizar o *software* Scracth, ou até mesmo como uma inspiração para que seja aplicado em sala de aula. Esse produto didático está organizado em dois tópicos: (1) um tutorial que apresenta a interface do *software* Scratch, alguns recursos e blocos mais utilizados; (2) a sequência de etapas que podem ser aplicadas em cinco encontros, com o objetivo de programar um jogo simples, apresentando uma possibilidade de etapas para a sua conclusão.

O produto didático traz questionamentos pensados em cada etapa, para contribuir com o desenvolvimento das atividades descritas e orientar professores e estudantes sobre ideias a serem desenvolvidas. Blocos utilizados e suas funções também são apresentados no segundo tópico. Sendo assim, segue o produto didático com o intuito de contribuir para uma exitosa experiência no uso do software Scratch.

TUTORIAL DO SCRATCH

Scratch é um ambiente de programação desenvolvido no MIT, Massachusetts Institute of Technology. O *software* Scratch segue a linha construcionista de Papert. O *software* Scratch oportuniza ao estudante utilizar conceitos básicos de programação, possibilitando o incentivo ao estudo de conceitos matemáticos e computacionais. Na sequência, detalharemos a interface do Scratch, suas principais ferramentas, assim como uma breve apresentação do site elaborado pelos criadores do Scratch.

- Site

Resnick et al. (2009) ressaltam o desejo de desenvolver uma abordagem para a programação que despertaria o interesse de pessoas que não se imaginavam como programadoras.

> Queríamos facilitar para todos, de todas as idades, origens e interesses, para programarem suas próprias histórias interativas, jogos, animações e simulações, e compartilharem suas criações entre si. Desde o lançamento

> público em maio de 2007, o Scratch Web site (http://scratch.mit.edu) tornou-se uma comunidade online vibrante, com pessoas compartilhando, discutindo e remixando um ao outro de seus projetos. (RESNICK et al., 2009, p. 60, tradução do autor)

Ao acessar o site, nos deparamos com um grande número de animações, histórias, jogos, entre outros, produzidos por pessoas das mais variadas características e de diversos lugares. No site é possível realizar um perfil de usuário a partir de um cadastro, possibilitando a postagem de suas produções, nas quais os demais usuários, visando a troca de ideias e o auxílio colaborativo, podem fazer comentários, fazer *download* e fazer sugestões de modificação dos projetos postados.

O site também disponibiliza o *download* do Scratch e uma versão online, além de possuir espaços de tutoriais e fóruns para a troca de ideias entre usuários. Atualmente o site dispõe de um cadastro especial para professores, em que, mediante esse cadastro, que é posteriormente analisado por uma equipe responsável, é possível o professor usuário criar turmas, cadastrar estudantes e inserir materiais.

- Interface

A interface do Scratch é composta por quatro áreas, sendo uma interface intuitiva. Detalhamos a interface do Scratch online (já que será a ferramenta na qual os participantes da pesquisa irão explorar), numerando a imagem da Figura 1 com as seguintes legendas:

5. Palco: Local onde os objetos são inseridos e onde é possível ver os resultados da programação criada;
6. Lista de *Sprites*: Área onde os *sprites* (atores) que fazem parte de projeto são localizados;
7. Blocos de comandos: Lista de Categorias e blocos de comandos;
8. Área de comandos: Local em que os comandos são conectados e editados.

Figura 1 - Interface do Scratch online

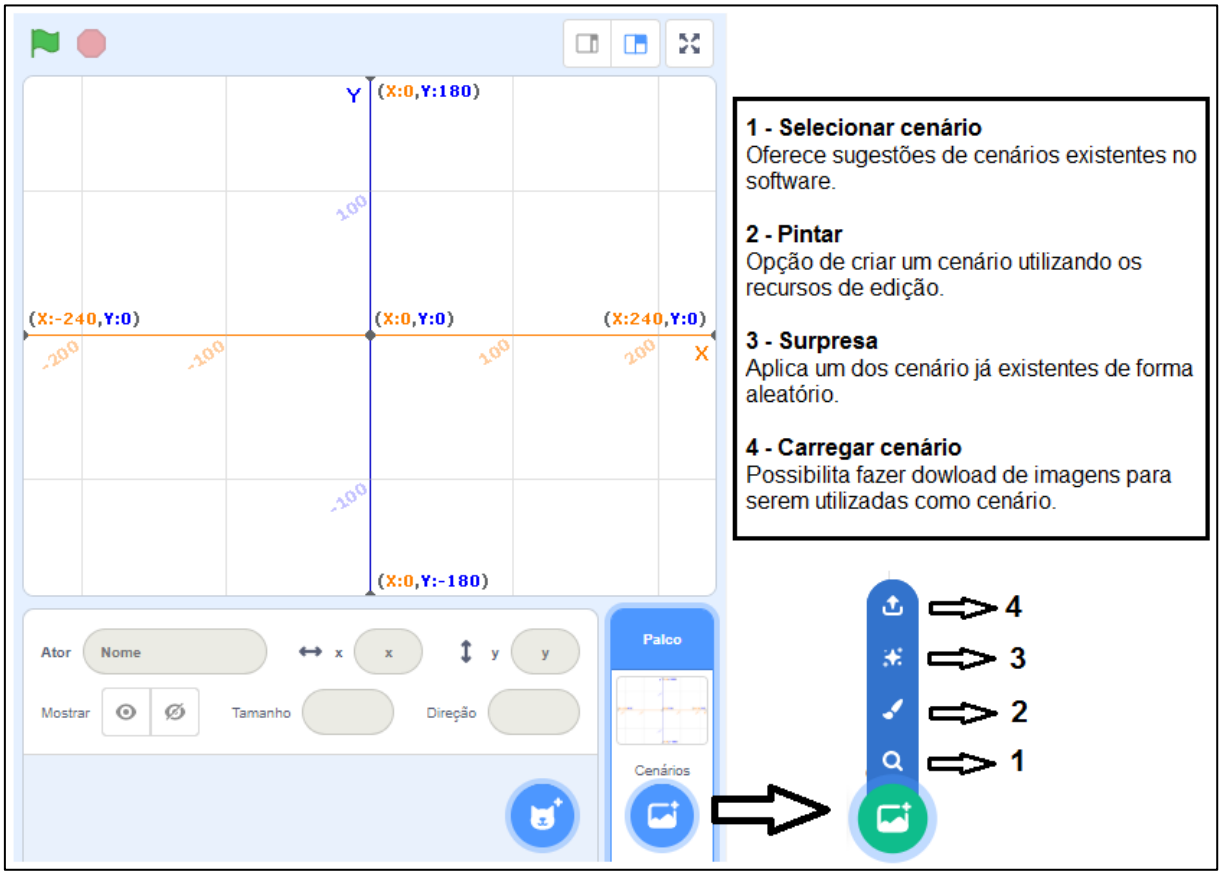
Fonte: Produção do autor.

Com o objetivo de compreender melhor as funções do Scratch e suas possibilidades, nas subseções seguintes, vamos apresentar as quatro áreas da interface (palco, blocos de comando, área de comando e lista de *scripts*) com mais detalhes e as ferramentas de recursos de edição do *software*, que possibilitam a criação de novos *sprites* ou edição dos já existentes.

- Palco

O palco do Scratch possui um plano de coordenadas cartesianas sem eixos visíveis. O centro do palco localiza-se na posição de origem do sistema (0,0), sendo que o intervalo do eixo x que está visível na tela é entre -240 e 240 e o intervalo do eixo y é entre -180 e 180. As unidades de medidas utilizadas no Scratch são chamadas passos, ou seja, o palco possui 480 passos de largura e 360 passos de altura. A Figura 2 exemplifica o plano cartesiano existente no palco e as opções disponíveis para se utilizar o palco.

Figura 2 - Palco do Scratch.



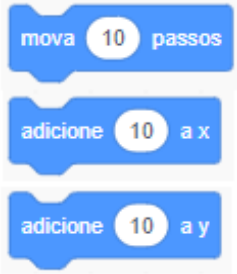
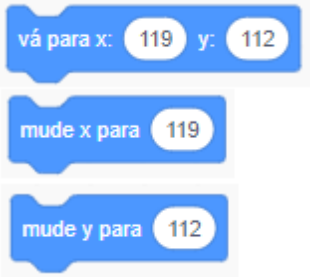
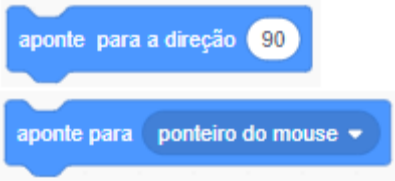
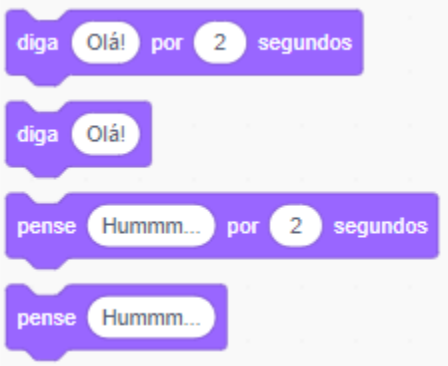
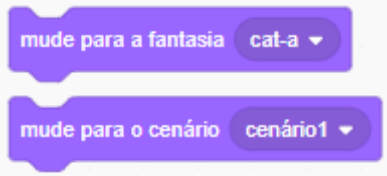
Fonte: Produção do autor.

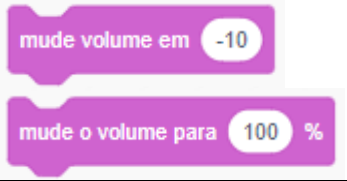
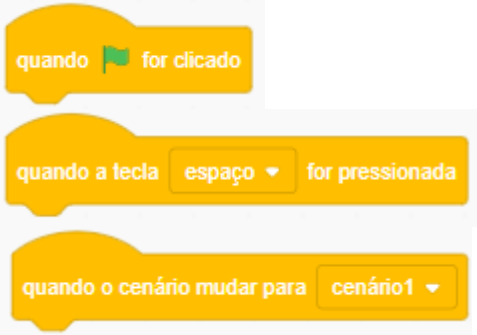
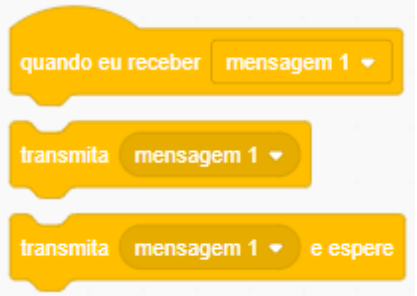
- Blocos de comando

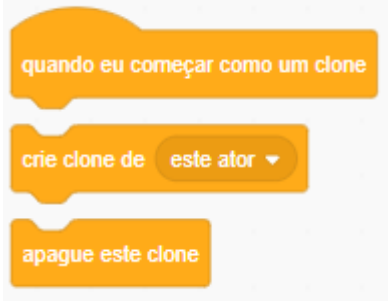
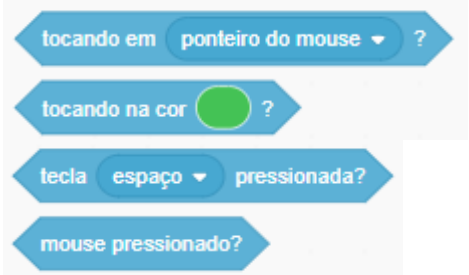
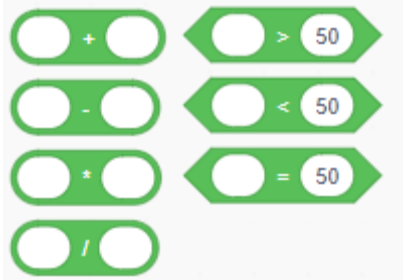
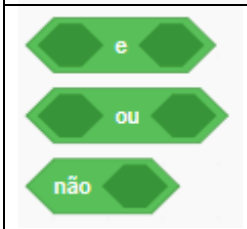
São nove categorias em que os blocos do Scratch online estão divididos. Os blocos são organizados por cor para ajudar a identificar os blocos que se relacionam por categoria. No quadro 1, apresentamos as categorias com alguns dos seus blocos. Esses blocos foram escolhidos por terem sido utilizados com frequência pelos estudantes durante os encontros ou por considerarmos importante incluí-los para a execução de projetos.

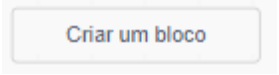
Quadro 1: Comandos do Scratch e suas funções.

CATEGORIA	BLOCO	AÇÃO

Movimento		Movimenta o *sprite* de acordo com o número de passos inseridos no campo, ou para direção apontada ou adicionando unidades em sua coordenada.
		Gira o *sprite* para a direita ou para a esquerda o valor, em graus, que é inserido no campo.
		Desloca o *sprite* para a uma nova posição (x,y).
		Gira o *sprite* para a direção indicada no campo ou em direção ao ponteiro do mouse.
Aparência		Um balão de fala/pensamento é exibido acima do *sprite*, sendo possível estipular um tempo para que o balão fique visível.
		Altera a fantasia do *sprite* ou o cenário.
		Esconde ou mostra o *sprite*.

Som		Opções para inserir sons.
		Aumenta ou diminui o volume do som escolhido.
Eventos		Executa um conjunto de procedimentos a partir de determinada ação ou momento.
		Permite enviar mensagens a um ou mais *sprites* que servem para coordenar ações. Ao receber uma mensagem um ou mais *sprites* executam as ações.
Controle		Determina um espaço de tempo para que um comando seja executado.
		Repete os comandos de acordo com o número de vezes que se deseja repeti-los.
		Repete sempre um comando ou um conjunto de comandos.

		Realiza ações a partir de uma condição específica, podendo executar outra ação caso a condição anterior não ocorra.
		Para todos os *sprites.*
		Cria clones de um determinado *sprite* podendo estabelecer um ou mais comandos para que os clones executem. Caso necessário, para que todos os clones não continuem na programação, é possível excluir os clones a partir de determinada condição.
Sensores		São condições para que o *sprite* execute uma ou mais ações. São inseridos dentro dos blocos de controle e de comando.
Operadores		Operadores matemáticos que são inseridos dentro de outros blocos para realizar operações ou estabelecer critérios de igualdades ou desigualdades.
		São conjunções utilizadas para executar ações quando duas situações ocorrerem ao mesmo tempo ou que pelo menos uma ocorra. Também pode ser utilizada para executar uma ou mais ações quando determinada situação não

		ocorrer. São inseridos dentro dos blocos de controle e de comando.
Variáveis		Possibilita a criação de variáveis para um ou mais *scripts*. Essas variáveis são utilizadas para, por exemplo, executar ações diferentes para o mesmo *script* em determinadas situações diferentes, ou também, estabelecer número de vidas para um personagem de um jogo.
Meus blocos		Cria um novo bloco que representa a programação de um conjunto de ações.

Fonte: Produção do autor.

- Área de comandos

Para que os *sprites* façam algo no palco, é necessário arrastar blocos, da aba de blocos de comando, para a área de comandos, encaixando-os e construindo, assim, programações que atribuam ações para o ator. Esses conjuntos de blocos unidos formarão o que chamamos de *script.* Os blocos encaixam-se de determinadas maneiras, impedindo que haja erros que tendem a ocorrer, principalmente em linguagens de programação que exigem a digitação dos comandos. Marji (2014) observa que os *scripts* podem ser testados à medida que forem criados, sem a necessidade de que estejam completos. É possível clicar sobre o primeiro bloco de uma sequência de blocos para que essa sequência seja imediatamente executada. Também é possível arrastar para a área de comando, blocos desconectados para testá-los individualmente, fazendo uma análise de suas funções dentro da programação. O interessante desses testes é a possibilidade do sujeito que está programando encontrar *bugs* (erros) na sua programação, partindo para um processo de *debugging*, termo utilizado para indicar a análise para encontrar e solucionar um *bug.* Quando um bloco ou conjuntos de blocos são

executados, é possível visualizar a execução do ator no palco, possibilitando uma análise do que está sendo programado.

Para cada *sprite*, o Scratch permite programações paralelas que possibilitam eventos distintos que podem acontecer simultaneamente (Figura 3).

Figura 3 - Programação paralela para um mesmo *sprite*.

Fonte: Produção do autor.

Quando a bandeira verde é pressionada, o *script* da Figura 6a é executado, fazendo com que o *sprite* execute os comandos listados na sequência do evento. Se a tecla “espaço” for pressionada, o *script* da Figura 6b executará o comando em paralelo com o comando já em execução, fazendo com que o ator possa, nesse caso, mover-se pelo palco e dizer “olá” ao mesmo tempo.

- Sprites

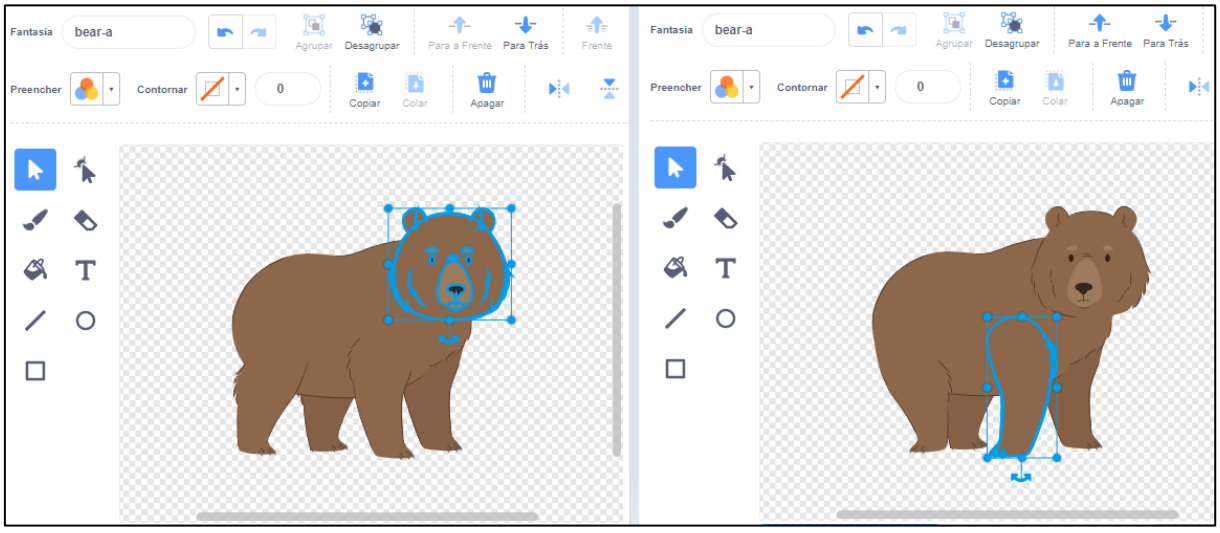
Em um mesmo projeto, é possível ter mais de um *sprite,* sendo que cada *sprite* terá a sua programação própria, ou seja, cada ator no palco terá uma lista de comandos a serem executados, mesmo que *sprites* diferentes executem os mesmos movimentos, deverá ser criado um conjunto de comandos para cada um deles.

Interessante ressaltar que o Scratch possui uma lista com vários *sprites* para que possam ser utilizados como atores. Alguns desses *sprites* já possuem mais de uma fantasia para que as animações sejam facilitadas.

- Recursos de Edição de Imagem (Paint Editor)

O Scratch possui um espaço para a criação de *sprites* e cenários, possibilitando também a edição do material já disponibilizado. Essas edições e criações podem ser realizadas clicando em  (pincel), localizado como opção ao selecionar um ator ou selecionar um cenário. Ao clicar no pincel para criar ou editar um novo ator, a aba “Fantasias”, que se encontra acima das opções de blocos, é acessada, ou na aba “Cenários”, caso seja desejado criar ou editar um cenário. Na Figura 4 podemos observar a edição de um ator já disponível no Scratch (grande parte dos *sprites* oferecidos pelo Scratch é construída em camadas, possibilitando a edição por camadas desses *sprites*).

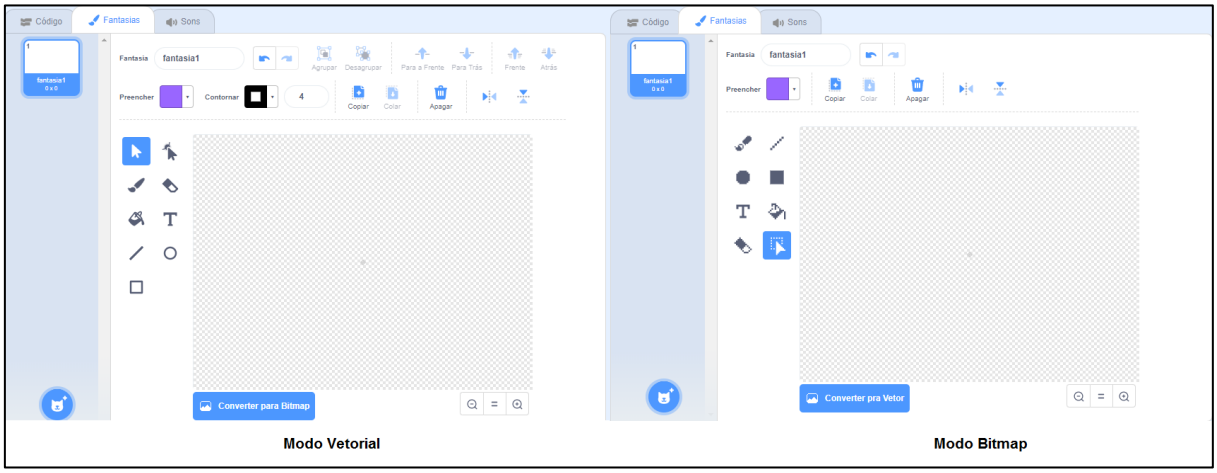
Figura 4 - Edição de *sprites* no Scratch.



Fonte: Produção do autor.

Na Figura anterior, vemos um exemplo de desenho sendo trabalhado no formato vetorial, que permite trabalhar imagens a partir de descrições geométricas, sendo essas descrições feitas a partir de vetores. Também é possível utilizar desenhos no formato bitmap, em que as imagens são formadas a partir de pontos minúsculos chamados pixels. Na Figura 5 mostramos a interface e as ferramentas disponíveis nos dois formatos de desenho.

Figura 5 - Recursos de edição de *sprites* do Scratch



Fonte: Produção do autor.

SEQUÊNCIA DE ATIVIDADES

- 1º Encontro: Apresentação do software Scratch – Comunicação com a máquina: Movimentação do ator no palco.

Apresentar o Scratch aos estudantes e trabalhar com os comandos básicos de movimentação de um ator no palco. Apresentar ferramentas e funcionalidades do Scratch, com o objetivo de familiarizar os estudantes com o *software*.

Dois conceitos matemáticos podem ser trabalhados (entre outros): o conceito de plano cartesiano a partir dos eixos cartesianos que fazem parte do palco do Scratch e o conceito de condição.

A seguir, apresentamos as três etapas do encontro.

Etapa 1 – Comunicação com a máquina.

Movimentar o ator no palco utilizando um ou mais blocos.

Possíveis questionamentos:

Q1 – Como movimentar o ator no palco?

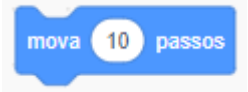
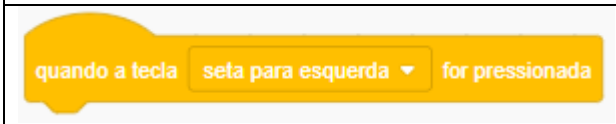
Q2 – O que é necessário para o computador fazer o movimento solicitado?

Q3 – Quais blocos são necessários para que o computador entenda que é necessário fazer o movimento escolhido?

Q4 – Para que servem os blocos de evento?

Possíveis blocos a serem utilizados e suas funções (mais prováveis de serem escolhidos):

Quadro 1 - Lista de blocos 1

BLOCO	AÇÃO
	Todo ator inicia como um vetor de movimento que vai da esquerda para a direita. Esse bloco já está programado para seguir esse vetor, podendo mudar de direção apenas quando o ator for girado.
	Blocos que giram o ator no sentido horário ou anti-horário.
	Bloco que comunica o evento para que a programação aconteça. Nesse caso, quando a bandeira verde é clicada.
	Bloco que comunica o evento para que a programação aconteça. Possibilita o uso de teclas para comunicar o evento.

Fonte: Produção do autor.

Etapa 2 – Movimentar o ator horizontalmente utilizando as setas direcionais do teclado.

Solicitar que os estudantes que movimentem o ator horizontalmente utilizando o bloco de evento 

.

Possíveis questionamentos:

Q1 – Como podemos movimentar o ator para a direita e para a esquerda utilizando as setas direcionais do teclado com o evento da bandeira verde?

Q2 – Em quais outros blocos os sensores podem se encaixar?

Q3 – O que o computador está entendendo com o algoritmo construído?

Q4 – Como organizar os comandos para que o computador compreenda o que deve ser feito?

Q5 – Há outro possível bloco para que, quando a tecla direita for pressionada, o ator se movimente?

Q6 – Qual a função do bloco sempre?

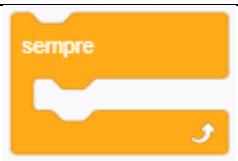
Q7 – Se o número que está dentro do bloco o faz ir para a direita e vocês querem ir para a esquerda, ou seja, movimento oposto, o que pode ser feito?

Q8 – Qual o comportamento dos números colocados dentro do bloco de movimento?

Q9 – Como fazer para o ator se movimentar mais rápido ou mais lento?

Novos blocos importantes e suas funções nessa etapa do encontro.

Quadro 2 - Lista de blocos 2

BLOCO	AÇÃO
	Bloco de sensor, definindo a ação utilizada para o comando.
	Bloco de condicional, que comunica a ação que o computador deve fazer a partir de uma condição.
	Utilizado para comunicar o computador que uma ação deve se repetir sempre a partir de um evento.

Fonte: Produção do autor.

Etapa 3 – Movimentar o ator verticalmente utilizando as setas direcionais do teclado para cima e para baixo e outros blocos de movimento.

Movimentar o ator verticalmente utilizando os blocos que adicionam valores aos eixos das abscissas e das ordenadas.

Possíveis questionamentos:

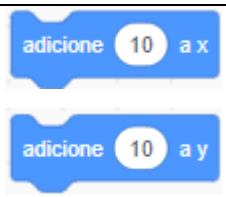
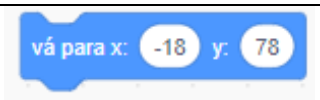
Q1 – Por que o bloco que adiciona valor a x faz o movimento horizontal?

Q2 – Como movimentar o ator verticalmente?

Q3 – O que representam o x e o y no palco?

Possibilidades de novos blocos importantes e suas funções nessa etapa do encontro.

Quadro 3 - Lista de blocos 3

BLOCO	AÇÃO
	Adiciona valores aos eixos das abscissas e das ordenadas, movimentando o ator na horizontal e na vertical. Pode ser utilizado para a compreensão do plano cartesiano que está vinculado ao palco.
	Movimenta o ator para um ponto específico do palco. Pode ser utilizado para compreender melhor o plano cartesiano que está vinculado ao palco.

Fonte: Produção do autor.

- 2º Encontro: Animação do ator no palco.

Animar o ator no palco, por meio da criação de fantasias e programação de movimentos no palco. Apresentar o recurso de edição de imagem do Scratch.

Duas possibilidades são possíveis para esse encontro: utilizar o projeto iniciado no encontro anterior, para dar prosseguimento às programações iniciais, de modo a torna-las cada vez mais elaboradas; dar início a um novo projeto.

A seguir, apresentamos o encontro após apresentar, e os estudantes explorarem, o editor de imagem.

- Animação de um personagem.

Animar um ator a partir da troca de fantasias utilizando o comando “espere” para dar o tempo de cada animação.

Possíveis questionamentos:

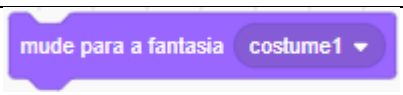
Q1 – Por que utilizar os blocos de troca de fantasia em sequência não anima o ator?

Q2 – O que é necessário para que o computador possa mostrar a animação aos poucos?

Q3 – Qual a função do comando espere?

Novos blocos importantes e suas funções nesse encontro.

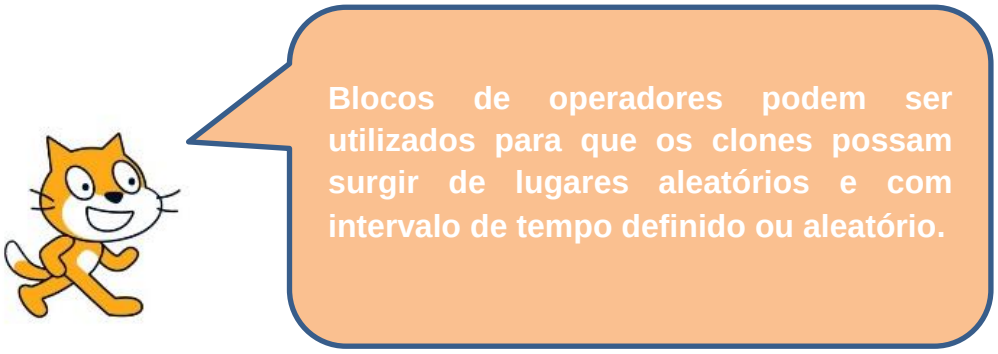
Quadro 4 - Lista de blocos 4

BLOCO	AÇÃO
	Comando que troca a fantasia do ator.
	Determina um tempo para que a fantasia troque ou para que o programa mude a fantasia

Fonte: Produção do autor.

- 3º Encontro: Criação de atores com o comando clone.

Criação de clones de um ou mais atores.



A seguir, apresentamos as duas etapas desse encontro.

- Etapa 1: Criação de clones 1

Criar clones de um personagem para surgirem de uma das bordas, sugerimos da borda direita para a borda esquerda.

Possíveis questionamentos:

Q1 – Como podemos fazer para um mesmo personagem aparecer de forma indeterminada, sem a necessidade de criar mais atores?

Q2 – Qual bloco podemos utilizar para criar o clone de um ator?

Q3 – Como podemos programar apenas os clones e não o ator clonado?

Q4 – Como fazer para determinar um intervalo de tempo para que os clones apareçam.

Q5 – Como podemos apagar os clones que percorreram toda a tela para que o computador não continue processando todos os clones? (Mesmo que o clone saia da tela, ele ainda estará sendo processado)

Q6 – O que podemos fazer para que os clones apareçam de forma aleatória, tanto na posição da borda, quanto no tempo?

Etapa 2: Criação de clones 2

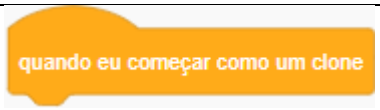
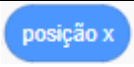
Criar um ataque para um personagem utilizando os mesmos recursos da etapa anterior.

Questionamentos:

Q1 - Quais blocos utilizar para que o computador entenda o momento de criar um clone, ou seja, o momento de atacar?

Novos blocos importantes e suas funções nas duas etapas de criação de clones.

Quadro 5 - Lista de blocos 5

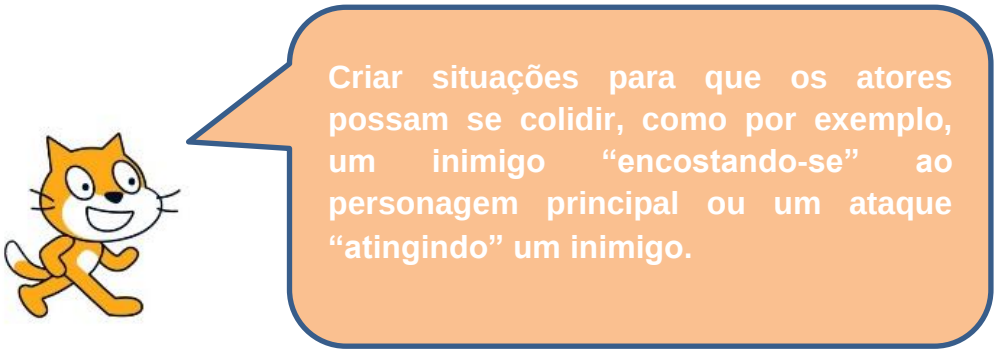
BLOCO	AÇÃO
	Utilizado na programação do ator para criar clones com as mesmas características.
	Bloco utilizado para iniciar a programação apenas dos clones, não interferindo no ator clonado.
	Apaga o clone a partir de um momento definido.
	Operador de igualdade, utilizado para condicionar uma ação a um valor especifico.
	Pode ser utilizado dentro do bloco de operadores. Utilizado para condicionar a uma ou mais posições específicas.
	Utilizado para escolher um intervalo para que o programa possa escolher aleatoriamente.

Fonte: Produção do autor.

- 4º Encontro: Colisão entre atores.

Programação de colisão de *sprites*, criando animações para os contatos e ações posteriores.

- Animação de atores colidindo.



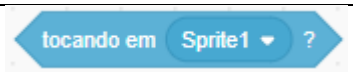
Possíveis questionamentos:

Q1 – Quais blocos podemos utilizar para representar a colisão entre os atores?

Q2 – Como podemos animar esse contato?

Bloco utilizado e sua função para determinar a colisão entre os atores:

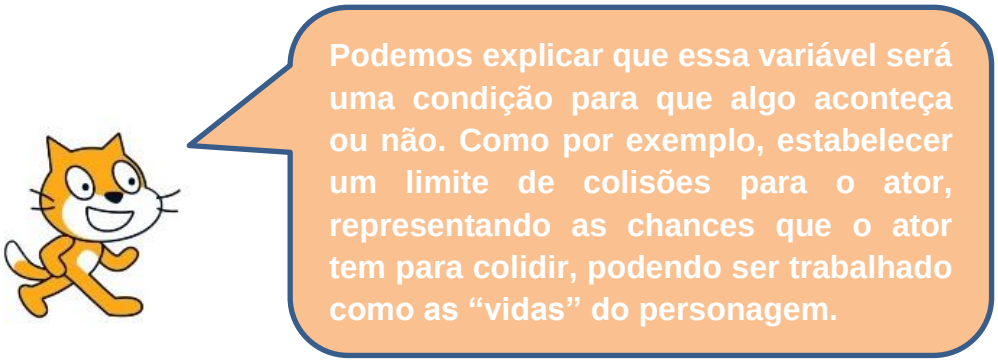
Quadro 6 - Lista de blocos 6

BLOCO	AÇÃO
	Bloco que condiciona a colisão

Fonte: Produção do autor.

- 5º Encontro: Uso do recurso “variável” no Scratch.

A partir do projeto do encontro anterior, propor a utilização de nova ferramenta para controlar a programação, como, por exemplo, criar uma invulnerabilidade em um dos atores quando ele reaparecer no jogo depois de “morrer”. Criar uma “variável”, que pode ser nominada ao ser criada para melhor organização e sem limites de criações, dentro da programação.



A seguir, apresentamos esses exemplos separados em etapas.

- Etapa 1: Invulnerabilidade do ator.

Em um jogo, quando o personagem principal é atingido ele recebe um tempo para que fique imune a ataques. Para programar essa situação, utiliza-se a ferramenta variável para criar sentenças de verdadeiro e falso para que o algoritmo seja utilizado.

Possíveis questionamentos:

Q1 – Quando o ator fica invulnerável, quais as ações ou situações que não podem ocorrer?

Q2 – Como podemos inutilizar os algoritmos que programam essas ações que não podem ocorrer durante o processo?

Q3 – Como podemos dar condições para que o computador utilize ou não um determinado algoritmo?

Q4 – Em quais locais do algoritmo devemos incluir as condições com a variável?

Q5 – Para que um algoritmo ocorra, a sentença determinada pela variável deve ser verdadeira ou falsa?

Q6 – Como podemos representar o personagem para evidenciar que ele está invulnerável?

Etapa 2: Contagem de vidas.

Criar uma variável para utilizar um valor sugestionado para iniciar o número de vidas do ator. Programar uma situação para que ele perca as vidas até um limite.

Questionamentos:

Q1 – Como podemos inserir um valor para a variável criada?

Q2 – Como inserimos essa variável na programação?

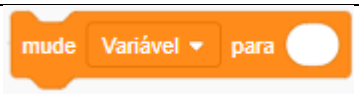
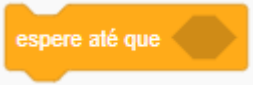
Q3 – Em qual momento a variável deve diminuir o número de vidas?

Q4 – Como dar um limite para que o número de vidas não seja negativo?

Q5 – O que pode ser programado para evidenciar o final do jogo?

Novos blocos importantes e suas funções nas duas etapas do encontro.

Quadro 7 - Lista de blocos 7

BLOCO	AÇÃO
	Bloco variável para inserir em outros blocos compatíveis.
	Define a sentença, que pode ser verdadeira ou falsa, ou até mesmo mudar a variável para algum número.
	Utilizado para que o algoritmo inicie a partir de uma condição específica na programação.
	Para todas as animações e trava os algoritmos.

Fonte: Produção do autor.