UNIVERSIDADE ESTADUAL DO PARANÁ - UNESPAR

PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO

NOÇÕES SOBRE PENSAMENTO COMPUTACIONAL EXPRESSAS POR PROFESSORES DE MATEMÁTICA NA CONSTRUÇÃO DE JOGOS NO SCRATCH

Suely Maria de Souza

Programa de Pós-Graduação em Educação Matemática
PRPGEM

Campo Mourão,
2024

PRPGEM
PROGRAMA DE PÓS-GRADUAÇÃO EM EDUCAÇÃO MATEMÁTICA

UNESPAR
Universidade Estadual do Paraná

UNIVERSIDADE ESTADUAL DO PARANÁ - UNESPAR
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM EDUCAÇÃO MATEMÁTICA - PRPGEM

NOÇÕES SOBRE PENSAMENTO COMPUTACIONAL EXPRESSAS POR PROFESSORES DE MATEMÁTICA NA CONSTRUÇÃO DE JOGOS NO SCRATCH

Suely Maria de Souza

Orientador:
Prof. Dr. Sérgio Carrazedo Dantas

Dissertação apresentada ao Curso de Mestrado do Programa de Pós-Graduação em Educação Matemática da Universidade Estadual do Paraná, linha de pesquisa: Tecnologia, diversidade e cultura em educação matemática, como parte dos requisitos necessários à obtenção do título de Mestre em Educação Matemática.

Campo Mourão
Dezembro de 2024

Ficha catalográfica elaborada pelo Sistema de Bibliotecas da UNESPAR e Núcleo de Tecnologia de Informação da UNESPAR, com Créditos para o ICMC/USP e dados fornecidos pelo(a) autor(a).

Souza, Suely Maria de
Noções sobre Pensamento Computacional expressas por professores de Matemática na construção de jogos no Scratch / Suely Maria de Souza. -- Campo Mourão-PR,2024.
104 f.: il.

Orientador: Sérgio Carrazedo Dantas.
Dissertação (Mestrado - Programa de Pós-Graduação Mestrado Acadêmico em Educação Matemática) -- Universidade Estadual do Paraná, 2024.

1. Pensamento Computacional. 2. Scratch. 3. Formação de Professores de Matemárica. I - Dantas, Sérgio Carrazedo (orient). II - Titulo.

Suely Maria de Souza

NOÇÕES SOBRE PENSAMENTO COMPUTACIONAL EXPRESSAS POR PROFESSORES DE MATEMÁTICA NA CONSTRUÇÃO DE JOGOS NO SCRATCH

Comissão Examinadora:

gov.br Documento assinado digitalmente
SERGIO CARRAZEDO DANTAS
Data: 20/03/2025 08:34:57-0300
Verifique em https://validar.iti.gov.br

Professor Doutor Sérgio Carrazedo Dantas– Presidente da Comissão Examinadora
Universidade Estadual do Paraná (Unespar)

gov.br Documento assinado digitalmente
MARIA IVETE BASNIAK
Data: 20/03/2025 08:53:15-0300
Verifique em https://validar.iti.gov.br

Professora Doutora Maria Ivete Basniak - Membro da Banca
Universidade Estadual do Paraná (Unespar)

Professor Doutor Guilherme Francisco Ferreira - Membro da Banca
Universidade Estadual Paulista Júlio de Mesquita Filho (Unesp)

Resultado: Aprovada

Campo Mourão
Dezembro de 2024

RESUMO

A presente pesquisa propõe um estudo teórico e prático sobre a compreensão de um grupo de professores de Matemática sobre Pensamento Computacional. Para isso, inicialmente, foi desenvolvida uma revisão bibliográfica sobre Pensamento Computacional abordando conceito, aspectos históricos, aplicabilidade, seus processos constituintes e sua utilização no contexto educacional. Reconhecemos o Pensamento Computacional como fator determinante do aprendizado amplo, bem como para soluções de problemas diversos, especificamente no contexto educacional, ou mesmo na vida cotidiana. No cenário prático da pesquisa, foi desenvolvida uma oficina sobre construção de Jogos no Scratch com professores do Núcleo de Educação de Apucarana, Paraná. Este estudo apresenta relatos sobre os jogos criados pelos professores durante o processo de formação e como os processos de Pensamento Computacional foram mobilizados por eles. Cada jogo digital, assim como seu vídeo e materiais necessários para sua construção, foram analisados e destacam-se nas análises dos processos mentais de Pensamento Computacional que foram mobilizados para sua construção. Conclui-se que os processos analisados nos jogos digitais são acionados durante sua construção, e mesmo não sendo relatados pelo cursista, no interior dos jogos construídos podemos verificar que foram mobilizados.

Palavras-chave: Pensamento Computacional, Scratch, Formação de Professores de Matemática.

ABSTRACT

This research proposes a theoretical and practical study on the understanding of a group of Mathematics teachers about Computational Thinking. Thereunto, initially, a bibliographic review on Computational Thinking was developed, addressing the concept, historical aspects, applicability, its constituent processes and its use in the educational context. We recognize Computational Thinking as a determining factor in broad learning, as well as for solving various problems, specifically in the educational context, or even in everyday life. In the practical scenario of the research, a workshop on building Games in Scratch was developed with teachers from the *Núcleo de Educação de Apucarana*, Paraná. This study presents reports on the games created by teachers during the training process and how Computational Thinking processes were mobilized by them. Each digital game, as well as its video and materials needed for its construction, were analyzed and stood out in the analyses of the mental processes of Computational Thinking that were mobilized for its construction. It is concluded that the processes analyzed in digital games are triggered during their construction, and even though they are not reported by the student, within the constructed games we can verify that they were mobilized.

Keywords: Computational Thinking, Scratch, Mathematics Teacher Education.

LISTA DE FIGURAS

LISTA DE SIGLAS

BNCC	Base Nacional Comum Curricular
CSTA	*Computer Science Teachers Association*
IFSP	Instituto Federal de Educação, Ciência e Tecnologia de São Paulo
ISTE	*International Society for Technology in Education*
MIT	Massachussets Institute of Technology - Instituto de Tecnologia de Massachusetts
PC	*Personal Computer* – computador pessoal
PNE	Plano Nacional de Educação
SBC	Sociedade Brasileira de Computação
TDIC	Tecnologias Digitais de Informação e Comunicação
UNESPAR	Universidade Estadual do Paraná

SUMÁRIO

1 INTRODUÇÃO

Neste texto introdutório abordo a escolha do tema, o caminho trilhado até aqui, seu desenvolvimento e a apresentação desta pesquisa. Em outras palavras, apresento o desenvolvimento de uma pesquisa que está em curso desde 2021[1].

Em minha atuação como professora por mais de 30 anos, vivenciei muitas transformações no processo educacional, e tenho observado o quanto a tecnologia tem feito parte da vida das crianças e adolescentes em seu dia a dia, e assim, considero importante que os recursos tecnológicos também façam parte de suas vidas escolares, por compreender que tais recursos possam contribuir com a aprendizagem em muitas disciplinas, em especial com a Matemática. Dessa forma, abordar a temática que proponho nesta pesquisa pode contribuir para incentivar a construção do Pensamento Computacional no âmbito docente, e para que as metodologias de ensino sejam ampliadas e favoreçam a motivação e o desenvolvimento de uma aprendizagem com profundidade nesse assunto.

O desenvolvimento tecnológico das últimas décadas reflete diretamente no contexto social, e tem influenciado o âmbito educacional de maneira considerável. Ainda assim, existem muitos professores que visualizam a tecnologia como um obstáculo a ser superado, e diante do receio de não ser possível dominá-lo, muitas vezes há rejeição de se adaptar às mudanças.

Junto ao desenvolvimento tecnológico surgem algumas noções associadas ao uso de tecnologia, e entre elas, o pensamento computacional é um tema emergente, que é o foco de minha pesquisa.

A primeira menção sobre o Pensamento Computacional apareceu nos trabalhos de Papert em 1980. Porém, em 2006, Wing escreveu um ensaio teórico abordando o assunto e no qual a expressão *Pensamento Computacional* instigou estudiosos, não somente da computação, a debaterem o assunto. Segundo Wing (2006), o Pensamento Computacional é uma habilidade para ser desenvolvida, não somente nos cientistas da computação, mas em todas as pessoas. Para a autora, pensar computacionalmente se traduz em processos que, resumidamente, são: decomposição, reconhecimento de padrões, abstração e produção de algoritmos.

A decomposição consiste em obter problemas menores a partir de um problema maior ou mais complexo. Com isso, é possível concentrar a atenção na resolução de partes específicas do problema, obtendo, ao fim, a solução desse problema.

[1] As partes relacionadas às experiências da pesquisadora estão grafadas em primeira pessoa.

O reconhecimento de padrões pode surgir a partir da decomposição, quando problemas menores podem ser solucionados com base em experiências anteriores ou via repertórios matemáticos/computacionais, ou ainda acontece ao perceber o que é constante ou variável nos dados ou formas associadas ao problema.

A abstração é o processo de concentrar a atenção no que é necessário e suficiente para a resolução de um problema ou subproblemas, desconsiderando dados ou informações irrelevantes.

A produção de algoritmo é o processo de obtenção de passos ou regras de ação desenvolvidos e efetivados durante a resolução de subproblemas ou problemas. Esses algoritmos podem ser (d)escritos em códigos da língua materna, códigos matemáticos, códigos de computação/programação, entre outros.

Com base nas discussões realizadas no grupo de estudos Autômato[2], foram acrescentados dois processos aos já abordados: formulação do problema e depuração.

A formulação do problema consiste em um processo em que o agente humano idealiza, de acordo com a necessidade, objeto e questionamentos, partindo para uma maneira de resolver um problema. Com os processos formulados, consideram-se quais seriam as possíveis variáveis e que ações serão executadas para que possa partir para o plano de ação, utilizando as técnicas necessárias para sua construção.

A depuração é um processo mental que acontece durante todo o tempo em que o agente humano está programando seu jogo digital. Os ajustes necessários para o funcionamento do jogo digital programado perpassam todos os outros pilares não hierárquicos.

A pesquisa realizada buscou investigar e compreender quais processos do Pensamento Computacional são mobilizados por um grupo de professores de matemática na construção de jogos no Scratch, quando envolvidos em processos formativos sobre a temática Pensamento Computacional.

Nossa fonte de informações para esta pesquisa são as produções de cursistas relativas ao projeto final de um curso com o seguinte título: *Pensamento Computacional na Construção de Jogos com o Scratch*. Essa produção final é composta por um arquivo ou um link para um projeto de um jogo construído no Scratch, e um texto descritivo que o cursista escreveu sobre seu projeto. Tanto o projeto como o texto foram lidos por nós com base nas noções de Pensamento Computacional, conforme apresentado no decorrer da revisão bibliográfica.

---

[2] Grupo de pesquisa e Estudos em Educação Matemática da Universidade Estadual do Paraná - Campus de Apucarana, tendo como líder o Prof. Dr. Sérgio Carrazedo Dantas.

O primeiro capítulo da pesquisa configura esta introdução. O segundo capítulo aborda historicamente o Pensamento Computacional. Aspectos como o Pensamento Computacional são apresentados na educação brasileira e na BNCC, além de suas contribuições no contexto escolar.

O capítulo seguinte trata da plataforma Scratch e seus recursos. Além disso, da sua utilização no Brasil e a apresentação de relatos de experiências com a plataforma e os resultados desses relatos.

O quarto capítulo destaca os jogos que foram desenvolvidos no curso ofertado aos professores via ambiente Moodle. Cada jogo digital, assim como seu vídeo e materiais necessários para sua construção foram mencionados no referido capítulo. Os jogos destacam os processos mentais de Pensamento Computacional que foram mobilizados para sua construção.

O quinto capítulo descreve a metodologia utilizada na pesquisa, enfocando delimitação do grupo, procedimentos de coleta de informações, foco da análise e método em ação.

O sexto capítulo traz a análise dos resultados do projeto final dos professores no ambiente Moodle, assim como os processos mobilizados nas construções. Ressalta-se que foram analisados três jogos que apresentaram, em sua construção, alguns elementos relevantes para nosso interesse de pesquisa: criatividade e explicação no fórum dos processos mentais que o cursista acredita ter mobilizado em seu jogo digital. Por último figuram as considerações finais, seguidas pelas referências utilizadas no estudo.

2 TECNOLOGIAS DIGITAIS E O PENSAMENTO COMPUTACIONAL

De acordo com Valente (2016), as tecnologias digitais de informação e comunicação trouxeram (e trazem constantemente) mudanças relevantes na organização da sociedade, em termos econômicos, sociais e culturais. Essas mudanças refletem na forma como as pessoas se comunicam e interagem, na relação direta com as informações e o conhecimento, no comércio, no trabalho e em diversas outras áreas.

A evolução da computação é constante, com a descoberta de novos princípios que tornam os anteriores obsoletos. No âmbito das tecnologias, os jogos são uma tendência crescente, indo além do aspecto do entretenimento, envolvendo aprendizagem e principalmente treinamento e aprimoramento de diversas habilidades. Em outras palavras, através da utilização de jogos, é possível aprimorar o raciocínio o que pode favorecer a construção de habilidades relacionada a resolução de problemas diversos (Denning, 2007).

Dessa forma, os avanços tecnológicos ampliaram as possibilidades de solução de problemas, por mais complexos que sejam. No entanto, a tecnologia tem se tornado fundamental na resolução de questões básicas do dia a dia. Por conta disso, a interação das pessoas com a tecnologia tem sido ampliada, possibilitando que a robótica desempenhe atividades outrora realizadas exclusivamente por seres humanos. Nesse sentido, é crescente a exigência pelas habilidades com recursos tecnológicos, ainda que se trate de um domínio básico de alguma área voltada para um trabalho específico. O conhecimento tecnológico passou a ser um diferencial, que pode ampliar ou reduzir possibilidades de sucesso em contextos diversos (Vicari; Moreira; Menezes, 2018).

Contudo, Valente (2016), citando Buckingham (2007), considera que, apesar de a cultura digital já fazer parte de muitos setores da sociedade, em grande parte do tempo as pessoas utilizam limitadamente os recursos que tecnologias ou os dispositivos oferecem. Explicando de outra forma, não se compreende o quanto os recursos tecnológicos podem oferecer em termos de ferramentas que podem contribuir com a realização de atividades de trabalho, estudo e lazer. “Ainda estamos na fase de entender e explorar essas tecnologias como se fossem sofisticadas máquinas de escrever, de acessar a informação e de se comunicar” (Valente, 2016, p. 866).

Logo, para uma relação mais próxima e útil com a tecnologia, é necessário que as sociedades compreendam que as tecnologias digitais devem ser consideradas recursos fundamentais na resolução de problemas na atualidade, desde o armazenamento e a organização

de dados, bem como a própria forma de os indivíduos analisarem as situações, com olhar mais prático e objetivo.

Ademais, mais recentemente, associado ao tema de tecnologias digitais, surgiu uma discussão sobre a expressão *Pensamento Computacional*, que vem mobilizando estudiosos do Brasil e de outros países.

2.1 Pensamento Computacional

Para abordar noções relacionadas ao Pensamento Computacional, é preciso reconhecer uma relação direta com análises numéricas e a prática de cálculos, desde Newton, que possibilitou a evolução de atividades de processamento e o avanço tecnológico como um todo. Muitos foram os estudos e ações que contribuíram para a evolução tecnológica. Como Charles Babbage, que criou a Máquina Diferencial, oferecendo-a ao governo britânico, em 1820, favorecendo a elaboração de tabelas de navegação com dados de qualidade, reduzindo naufrágios. Já no século XX, na década de 1920, o exército norte-americano custeou pesquisas sobre computadores analógicos, e na década de 1940, fez o mesmo para computadores digitais, buscando cálculos mais precisos para o disparo de projéteis (Tedre; Denning, 2016).

O Pensamento Computacional, mesmo antes de ser assim denominado, sempre foi uma prática comum, mesmo anteriormente ao surgimento da ciência da computação, pois vários cientistas realizavam análises numéricas e tabulação em larga escala, em práticas que, atualmente, seriam denominadas Pensamento Computacional. Depois do surgimento da computação moderna, a ciência valeu-se dela para seus estudos. Assim, as práticas associadas ao Pensamento Computacional ampliaram possibilidades de estudos científicos e o surgimento de novos recursos e de novas tecnologias (Tedre; Denning, 2016).

Para Nardelli (2019), o pensamento computacional é um conceito que precisa ser compreendido como recurso fundamental para a sociedade, de forma geral. Contudo, é inegável a dificuldade de propor uma descrição precisa sobre o tema, em decorrência de sua complexidade, sendo possível discorrer sobre seu funcionamento e demandas sem se voltar de maneira aprofundada aos aspectos epistemológicos. O autor destaca quão necessária é a abordagem do pensamento computacional no ambiente educacional, contribuindo de diversas maneiras para a construção do conhecimento.

A primeira menção efetiva sobre o Pensamento Computacional apareceu nos trabalhos de Papert, em 1980. Desde então, a expressão Pensamento Computacional tem sido abordada e

discutida em vários níveis de ensino e em áreas de conhecimento, como na Educação e na Computação (Beleti Júnior; Sforni, 2021).

Para o desenvolvimento do conceito de Pensamento Computacional, Papert e seus colaboradores criaram experiências buscando discutir maneiras através das quais as crianças pudessem controlar o movimento de um robô, com comandos da linguagem *Logo*. Assim, no robô, era acoplada uma caneta, que delineava seu trajeto no papel, desenvolvendo formas específicas. Com base nessas atividades, análises e observações, Papert e colaboradores criaram a teoria construcionista, que foi abordada por Papert, em 1980, em seu livro *Mindstorms*, com a proposta da programação como estímulo de ideias. Na visão de Papert, os computadores precisariam ser usados com o intuito de que as pessoas *pensassem com* as máquinas, e para que pudessem refletir sobre o próprio ato de pensar (Valente, 2016).

Anos mais tarde, Wing (2006) argumentou que o Pensamento Computacional é uma habilidade para ser desenvolvida, não somente pelos cientistas da computação, mas por todas as pessoas. Para a autora, pensar computacionalmente traduz-se em processos que, resumidamente, são: decomposição, reconhecimento de padrões, abstração e produção de algoritmos. Depois do ensaio teórico sobre Pensamento Computacional publicado por Jeannette Wing, em 2006, muitos trabalhos sobre o tema foram desenvolvidos e publicados (Beleti Júnior; Sforni, 2021).

Conforme descrevem Beleti Júnior e Sforni (2021), Wing conceitua o Pensamento Computacional e define suas características como:

- Conceitualizar, não programar: assim, a informática não é uma programação de computadores. Dessa forma, pensar como um cientista informático envolve reflexão em vários sentidos;
- Competência fundamental e não mecanização: a competência fundamental é um pensamento que vai além da rotina;
- Um modo como os humanos pensam, não os computadores: ou seja, o Pensamento Computacional ocorre quando humanos pensam com criatividade e dinamismo;
- Complementa e combina o pensamento matemático e de engenharia: informática e matemática estão intimamente ligadas, pautadas nos conhecimentos da engenharia, ao construir sistemas que fazem parte do mundo. Logo, essas características envolvem o mundo real e virtual, de maneira complementar;

- Ideias, não artefatos: não são softwares e artefatos de hardware que fazem o Pensamento Computacional, mas os conceitos computacionais usados na solução de problemas que contribuem para a vida cotidiana; e
- Para todos, em todo o lado: quando o Pensamento Computacional estiver inserido na vida das pessoas, não exigirá explicações e definições.

Segundo Wing (2006), o Pensamento Computacional é uma habilidade imprescindível para quaisquer pessoas, e não está restrito aos envolvidos com informática. Entre práticas como leitura, escrita e aritmética, é preciso inserir o Pensamento Computacional para que as crianças, ao aprenderem os conceitos e aplicabilidades, possam fazê-los de maneira analítica.

De maneira crescente, tem se considerado o pensamento computacional como uma nova disciplina, com aspectos distintos da ciência da computação. Buscando uma precisão no conceito que não foi atingida por Wing em seus ensaios teóricos de 2006 a 2017, alguns estudiosos destacam outros aspectos.

Segundo Vicari, Moreira e Menezes (2018, p. 29), o Pensamento Computacional

> [...] está em constante evolução e sua definição, bem como os seus limites, igualmente evoluem. Assim, como era de se esperar, não foi encontrada uma definição precisa ou formal do termo, pois o PC não envolve apenas conceitos e resultados formais de Ciência da Computação, também agrega práticas de projetar sistemas, entender o comportamento humano e o pensamento crítico (WING, 2010) (ou seja, esse conjunto de definições representa o estado atual do PC). Esse fato não limita o surgimento de novas propostas suportadas por outros conjuntos de referências.

A *Computer Science Teachers Association* (CSTA) e a *International Society for Technology in Education* (ISTE), com auxílio de colaboradores, elaboraram conceitos e dimensões para o Pensamento Computacional que deveriam ser direcionamentos para a Educação Básica. Assim, definiu-se o Pensamento Computacional como processo de resolução de problemas envolvendo as características apresentadas a seguir:

> [...] formulação de problemas de uma forma que seja possível usar um computador e outras ferramentas para ajudar a resolvê-los; organização lógica e análise de dados; representação de dados através de abstrações, como modelos e simulações; automatizações de soluções através do pensamento algorítmico (uma série de passos ordenados); identificação, análise e implementação de soluções possíveis com o objetivo de alcançar a combinação mais eficiente e eficaz das medidas e recursos; generalização e transferência desse processo de resolução de problemas para uma grande variedade de problemas (Bordini, 2017, p. 02).

Nesse sentido, o Pensamento Computacional está diretamente relacionado com a capacidade de solucionar problemáticas, pela análise de conceitos que são imprescindíveis na ciência da informática. “O Pensamento Computacional inclui um leque de ferramentas mentais que reflete a amplitude do ramo das ciências informáticas” (Torres; Figueiredo, 2021, p. 02).

Diante de um problema a ser solucionado, é preciso que se compreenda o problema para que seja possível pensar em como solucioná-lo. Assim, a ciência informática contribui com embasamento sólido para buscar respostas sobre a dimensão do problema e a sua dificuldade de solução. Para que seja possível delimitar a dificuldade de um problema, é preciso considerar o poder subjacente da máquina, que é o dispositivo informático com condições de solucionar a questão. Isso leva à necessidade de analisar as instruções da máquina, limitações de seus recursos e seu ambiente de operações (Wing, 2006).

Brackmann (2017, p. 33) afirma que "o Pensamento Computacional utiliza [...] 'Quatro Pilares': (Decomposição, Reconhecimento de Padrões, Abstração e Algoritmos), para atingir o objetivo principal: a resolução de problemas". Quando observados, esses pilares contemplam e inclusive ampliam os argumentos da Base Nacional Comum Curricular (BNCC) sobre o tema.

Ainda segundo Brackmann (2017), o Pensamento Computacional consiste em uma distinta capacidade criativa, crítica e estratégica humana sobre saber utilizar os fundamentos da computação nas diversas áreas do conhecimento, com a finalidade de identificação e resolução de problemas, sendo de maneira individual ou colaborativa, através de passos claros que uma pessoa ou máquina possam executá-los de forma eficaz (Vee, 2013).

Nesse sentido, Brackmann (2017) e Brackmann *et al.* (2016) salientam que o pensamento computacional tem como pressupostos a identificação de problemas complexos e a sua divisão em partes menores e mais simples, que sejam fáceis de gerenciar.

Em nosso grupo de pesquisa, adotamos seis processos mentais que constituem o que chamamos de pensamento computacional. Nossos estudos sobre Brackmann *et al.* (2016), Brackmann (2017), Cavalheiro (2020) e Wing (2006) e outros estudiosos nos levaram a concordar que o Pensamento Computacional é constituído por: decomposição, reconhecimento de padrões, abstração e produção de algoritmos, mas somados a esses, incluímos outros: formulação do problema e depuração. Eles foram abordados em pesquisas realizadas por integrantes do grupo Autômato, como as de Dantas (2023), Lirio e Prado (2023), Plewka e Dantas (2023) e Santos *et al.* (2023).

Assim, os processos do Pensamento computacional podem ser resumidos da seguinte forma.

- **Formulação do problema**: é o processo em que o problema é elaborado em termos de necessidade, objetivo e pergunta;
- **Decomposição**: é o processo de quebrar um problema grande em partes menores e mais gerenciáveis. Quando os professores de matemática criam jogos no Scratch,

eles decompõem o problema em tarefas menores, como criar objetos, definir parâmetros e criar lógicas de jogo. Assim a decomposição consiste em obter problemas menores partindo de um problema maior ou mais complexo, sendo possível concentrar a atenção na resolução de partes específicas do problema, obtendo, ao fim, sua resolução;

- **Reconhecimento de padrões**: é o processo de monitorar e avaliar dados para identificar tendências e padrões. Pode surgir partindo da decomposição, quando os problemas menores podem ser solucionados com base em experiências anteriores, com matemática ou computação, com constante ou variável nos dados ou formas associadas ao problema;
- **Abstração**: é o processo de identificar e focar atenção nos principais elementos de um problema, ignorando os detalhes desnecessários. Na resolução de um problema, a abstração contribui com a identificação de variáveis e operações necessárias para a resolução, ignorando os detalhes menores. Em outras palavras, é o processo de concentrar no que é necessário e suficiente para a resolução do problema, desconsiderando o que não tem relevância;
- **Produção de Algoritmos**: Um algoritmo é uma sequência específica de instruções para resolver um problema. Produzir um algoritmo é o processo de construção de um modelo abstrato que resolve parte de um problema. A resolução de um problema geralmente é a integração de pequenos algoritmos que funcionam em sequência ou em dependência, e que julgam estados e condições de entrada; e
- **Depuração**: é o processo de busca de erros na resolução em curso, o que se traduz na verificação do que é implementado e como essa possível solução é pensada. Trata-se de um movimento de mão dupla em que, em um sentido, o que é apresentado em tela pode modificar o que o sujeito pensa sobre sua resolução; e no outro sentido, como o que sujeito reflete (pensa) sobre a resolução leva a modificações nos processos e algoritmos que resultam no que é apresentado em tela.

O processo de depuração envolve busca, reconhecimento e correção do erro. Como explica Espadeiro (2021), o pilar da depuração envolve testes, verificação, aprimoramento e potencialização da resolução do problema. Assim, avaliar e corrigir são fundamentais, mas otimizar os processos possibilita reavaliação e potencialização da eficácia.

2.2 Pensamento computacional no contexto educacional

No cenário educacional, utilizou-se o termo pensamento computacional no ano de 1967, em relação à linguagem de programação *Logo*. Seymour Papert, Cynthia Solomon e Wally Feurzeig, em 1967, aplicaram esse termo de forma pioneira. O objetivo da programação visava a favorecer a compreensão relacionada às formas de funcionamento da linguagem computacional, dando início à reflexão sobre a ampliação do pensamento da criança, guiado por questões, sobre como e o porquê de programar (Gonçalves; Portella; Luz, 2019).

De acordo com Barros *et al.* (2018), o Pensamento Computacional pode ser abordado em ações pedagógicas de várias maneiras, em decorrência de seu caráter inovador, podendo ser utilizado em diversos aspectos da educação como instrumento de aprendizagem. Portanto, a inserção do Pensamento Computacional na Educação Básica favorece o engajamento dos discentes em relação ao aprendizado de tecnologias. Assim, em relação às áreas utilizadas para a difusão do pensamento computacional, informática e matemática são as mais recorrentes nas práticas docentes.

O Pensamento Computacional amplia possibilidades de aprendizado em várias áreas, assim como acontece com os jogos tecnológicos, que contribuem para o desenvolvimento do raciocínio rápido e aprimora habilidades. Nesse sentido, *a computação é um jogo infinito*; com essa linha de raciocínio, é possível encantar os alunos para o desenvolvimento do Pensamento Computacional em sala de aula, através de estratégias utilizadas em jogos que os motivam na construção do conhecimento através da busca de solução de problemas (Denning, 2007).

Com os avanços tecnológicos sendo incorporados pela sociedade, alguns estudiosos acreditaram que a ideia de programação fosse alterada conforme o letramento computacional fosse assimilado na rotina de vida das pessoas. Os conhecimentos da tecnologia de informação possibilitam a abordagem de seus conteúdos no ambiente escolar, mas as aulas sempre focaram no uso dos recursos da informática, nem mesmo se aproximando de conceitos e funcionamento (Tedre; Denning, 2016).

A discussão acerca do papel da compreensão sobre a computação de uma maneira mais ampla e aprofundada trouxe críticas e reflexões, e os profissionais das ciências cognitivas do desenvolvimento e psicologia da programação destacaram que a relação superficial com os recursos de informática poderia até mesmo reduzir a habilidade de solucionar problemas. Ainda, foi reconhecido que compreensão e aprendizado sobre programação estavam diretamente relacionados às habilidades matemáticas, raciocínio lógico, condicional, memória, pensamento processual e habilidades de raciocínio temporal. Para isso, é necessário que se

ensine computação ao mesmo tempo em que se aprende a utilizá-la. Dessa forma, o processo torna-se significativo, e leva o aluno à assimilação e reflexão (Tedre; Denning, 2016).

No Brasil, têm se ampliado as propostas de introduzir o Pensamento Computacional na educação. As publicações são crescentes sobre o tema, na Comissão especial de Informática na Educação e no Workshop de Educação em Computação. Muitas dessas publicações mostram resultados promissores sobre o envolvimento de estudantes com as tecnologias e seu potencial interdisciplinar. Essas práticas também são incentivadas por outras que são realizadas em diferentes países, como nos Estados Unidos, que têm impulsionado a implantação de conhecimentos da computação para o contexto da Educação Básica (Raabe *et al*., 2015).

> O conjunto de capacidades cognitivas de ler, de escrever e de fazer operações matemáticas que foram fundamentais para o exercício da cidadania até o século passado, necessita ser ampliado, acrescido da habilidade do pensamento computacional, essa capacidade de descrever, de explicar, de operar com situações complexas. As concepções quanto à alfabetização e o letrametro digital têm sido ampliados radicalmente. Dos processos de instrumentalização para o uso de recursos computacionais, passa-se à necessidade de desenvolver habilidades exigidas para atuar na sociedade do século XXI, em especial, utilizar saberes e dispositivos tecnológicos para construir respostas a problemas (Conforto *et al*., 2018, p. 03).

O Plano Nacional de Educação (PNE) traz, em sua meta 6, a questão da educação integral. Nesse contexto, os alunos permanecem na escola por um período igual ou superior a sete horas, com o oferecimento de atividades multidisciplinares, que podem abranger o contato com conhecimentos diversos, como informática, podendo ter um período voltado para o desenvolvimento de noções sobre pensamento computacional.

Os conceitos da ciência da computação têm sido abordados justificando atividades desenvolvidas para o desenvolvimento de habilidades como pensamento crítico e computacional, contribuindo para a compreensão das tecnologias digitais em sua produção, além do seu uso habitual. São conhecimentos imprescindíveis para a formação de indivíduos na atualidade, para que possam exercer sua cidadania. Nesse contexto, tem sido propostas transformações curriculares em vários países, com introdução de programação e da ciência da computação no início da Educação Básica (Valente, 2016).

> Na maior parte das propostas implantadas ou dos estudos realizados a ideia é reavivar a programação por meio de atividades como *coding computer science* ou *computer programming*, objetivando a criação de condições para o desenvolvimento do pensamento computacional. Outros países têm uma preocupação muito mais ampla do que simplesmente aprender a programar e estão buscando novas maneiras de explorar os conceitos computacionais no sentido de criar condições para o desenvolvimento do pensamento computacional (Valente, 2016, p. 867).

Certamente é importante ressaltar que a forma como se pretende implantar o estudo da ciência da computação e do Pensamento Computacional no contexto da educação precisa considerar recursos e os níveis das turmas. Isso possibilita que os docentes tenham autonomia para a implantação dessas temáticas da maneira que melhor se adequar à realidade de sua instituição.

Buscando se inserir na discussão sobre o Pensamento Computacional mais relacionada à formação na fase da Educação Básica, a BNCC (Brasil, 2018, p. 474) afirma que "Pensamento Computacional envolve as capacidades de compreender, analisar, definir, modelar, resolver, comparar e automatizar problemas e suas soluções, de forma metódica e sistemática".

Documento normativo, a BNCC apresenta um conjunto orgânico e progressivo dos conhecimentos fundamentais para todos os alunos durante as etapas da Educação Básica, para que tenham seu direito de aprendizagem e desenvolvimento garantidos, seguindo o Plano Nacional de Educação, bem como o que se apresenta nas Diretrizes Curriculares Nacionais da Educação Básica (Reis; Barichello; Mathias, 2021).

De acordo com Barichello (2021), a BNCC aborda o desenvolvimento do pensamento computacional, sendo um dos objetivos da área da Matemática dos Anos Finais do Ensino Fundamental até o Ensino Médio. Apesar do termo não estar entre as competências nem nas habilidades, é apresentado de forma recorrente nas discussões presentes no documento, no componente curricular de Matemática. De forma ampla, há sugestões sobre o desenvolvimento do pensamento computacional como um aspecto que potencializa o aprendizado de Matemática.

No contexto da Matemática e suas tecnologias, a BNCC traz propostas de formação inovadoras em relação a esse componente curricular, aplicando-o à realidade, diversificando cenários e aplicabilidades. É fundamental que sejam analisadas as experiências dos alunos para que se possa ampliar, aprofundar e consolidar os aprendizados necessários no Ensino Fundamental. No fim do Ensino Médio, o aluno precisa ter habilidade de formular e solucionar problemas em vários cenários, de maneira proativa e através de recursos matemáticos. Para que isso aconteça, o ensino de Matemática precisa favorecer o desenvolvimento de formas de pensar, comunicar-se e argumentar, por meio de reflexões e conclusões atingidas em conjunto (Reis; Barichello; Mathias, 2021).

Em relação à Educação Básica, a BNCC define que os alunos precisam ter desenvolvido dez competências gerais, que garantem o seu direito de aprendizagem e desenvolvimento. Assim, é importante ressaltar que essas competências gerais da Educação Básica estão relacionadas e envolvem os conhecimentos adquiridos nas três etapas da Educação Básica,

sendo elas a Educação Infantil, o Ensino Fundamental e o Ensino Médio, com articulação de saberes e habilidades que já foram descritas na Lei de Diretrizes e Bases da Educação Nacional em 1996 (Brasil, 2018).

Durante o Ensino Médio, busca-se a garantia legal do aprofundamento de conhecimentos que foram construídos no decorrer do Ensino Fundamental. A BNCC estrutura-se, no contexto do Ensino Médio, com base em competências e habilidades, e não se limita aos componentes curriculares, diferentemente do Ensino Fundamental, período em que se trabalham competências ligadas aos seus respectivos componentes curriculares (Reis; Barichello; Mathias, 2021).

Para o Ensino Médio, com base na BNCC, o conhecimento matemático precisa ser desenvolvido através de cinco competências específicas:

> 1. Utilizar estratégias, conceitos e procedimentos matemáticos para interpretar situações em diversos contextos, sejam atividades cotidianas, sejam fatos das Ciências da Natureza e Humanas, das questões socioeconômicas ou tecnológicas, divulgados por diferentes meios, de modo a contribuir para uma formação geral.
> 2. Propor ou participar de ações para investigar desafios do mundo contemporâneo e tomar decisões éticas e socialmente responsáveis, com base na análise de problemas sociais, como os voltados a situações de saúde, sustentabilidade, das implicações da tecnologia no mundo do trabalho, entre outros, mobilizando e articulando conceitos, procedimentos e linguagens próprios da Matemática.
> 3. Utilizar estratégias, conceitos, definições e procedimentos matemáticos para interpretar, construir modelos e resolver problemas em diversos contextos, analisando a plausibilidade dos resultados e a adequação das soluções propostas, de modo a construir argumentação consistente.
> 4. Compreender e utilizar, com flexibilidade e precisão, diferentes registros de representação matemáticos (algébrico, geométrico, estatístico, computacional etc.), na busca de solução e comunicação de resultados de problemas.
> 5. Investigar e estabelecer conjecturas a respeito de diferentes conceitos e propriedades matemáticas, empregando estratégias e recursos, como observação de padrões, experimentações e diferentes tecnologias, identificando a necessidade, ou não, de uma demonstração cada vez mais formal na validação das referidas conjecturas (Brasil, 2018, p. 531).

De acordo com Barichello (2021), essas competências têm conexão direta com a prática de raciocínio lógico, representação, comunicação e argumentação. Assim, na BNCC define-se competência como mobilização de conhecimentos, habilidades, ações e valores que contribuem para solucionar problemas do cotidiano.

É fundamental considerar que crianças e adolescentes que frequentam o ambiente escolar já nasceram imersos em uma realidade tecnológica, e as Tecnologias Digitais de Informação e Comunicação (TDIC) são os meios de comunicação mais utilizados. Além disso, são a forma que muitos têm como meio de entretenimento. Essas mudanças trouxeram novos comportamentos entre crianças e adolescentes, diferentes de alguns anos, gerando maior

questionamento e obstáculos para que sua atenção se prenda a algo que considera desinteressante (Vicari; Moreira; Menezes, 2018).

A Matemática precisa se apoiar nos recursos tecnológicos como forma de ampliar possibilidades de aprendizado, contextualizando e favorecendo uma compreensão ampla e com sentido, com o uso das tecnologias como forma de auxílio na solução de problemáticas e, consequentemente, na assimilação de conceitos matemáticos.

Segundo a Sociedade Brasileira de Computação (SBC), o Pensamento Computacional está ligado a conceitos abstratos, de análise, também automação. Assim, tem conexão direta com a computação. A SBC também liga a computação com a ideia de cultura digital, bem como com sua contribuição para a construção de definições, como fluência digital, ética digital e computação e sociedade. Essa fluência digital tem conexão direta com plataformas digitais, e consequentemente, está ligada ao Pensamento Computacional *plugged*, que se vale das tecnologias digitais na educação. Nessa definição da SBC, o mundo digital também tem ligação com a computação. O mundo digital engloba temas como codificação, processamento e distribuição de informações, que novamente estão relacionados ao Pensamento Computacional (Vicari; Moreira; Menezes, 2018).

Considerando a necessidade de refletir sobre a motivação dos alunos na construção do conhecimento, o Pensamento Computacional mostra-se um início que pode ser trabalhado primeiramente na Educação Infantil. Quando se pensa no ensino de Matemática apenas em etapas posteriores da Educação Básica, acredita-se que o Pensamento Computacional tende a ser abordado de maneira breve e rasa, uma vez que a temática é oferecida aos alunos de maneira relacionada a experiências de outras pessoas. Entretanto, se há uma busca pela prática do Pensamento Computacional pelos alunos, uma vez que se apresenta o tema de maneira superficial, considerando estudos e experiências vivenciadas por outras pessoas que não fazem parte da realidade do aluno, não se favorece a prática (Navarro, 2021).

Acredita-se que, em razão de os alunos não aprenderem a inserir sua realidade na atividade prática abordada, principalmente através de experiências realizadas em seu cotidiano, assimilando conceitos, torna-se mais complexa a compreensão da abstração que a teoria. Logo, o pensamento torna-se genérico e não pessoal, configurando-se em algo que demanda maior atenção e concentração para ser entendido (Sousa; Miota; Carvalho, 2011).

Dessa forma, a relação entre pensamento teórico e pensamento científico é inegável. Para isso, o Pensamento Computacional na Educação Matemática precisa ser compreendido de maneira ampla. Ao considerar que os documentos oficiais da educação não apresentam recursos e metodologias através das quais se abordaria o Pensamento Computacional em sala de aula, a

questão torna-se relevante sob o aspecto dessa temática como crucial no Ensino de Matemática, em busca de práticas que vão além da teoria, ampliando as conexões de conceitos trabalhados nesse componente curricular de forma ampla e útil para a vida do indivíduo (Navarro, 2021).

Valer-se dos conteúdos matemáticos e da forma como são ensinados para potencializar e contribuir com as vivências dos alunos, construindo novas maneiras de ação diante de problemáticas variadas pode ser uma das formas através das quais a Matemática pode contribuir, de forma prática, para o cidadão.

Nesse contexto, são necessários instrumentos para o ensino de Matemática, além de uma abordagem das atividades matemáticas como ações humanas, que podem envolver erros e acertos. Assim, o componente traz contribuições diretas para a formação do cidadão, com capacidade para identificar diferentes funções da matemática em vários contextos (Barichello, 2021).

O aprendizado de Matemática envolve muito mais que regras e fórmulas, engloba a compreensão e a demanda de construir conceitos através da forma de pensamento. Assim, o ensino de matemática vai além da visão técnica e mecanizada, que há anos se apresentava como a melhor opção. Existe a necessidade de apropriação de conceitos, de compreender e considerar questões que vão além da prática, como temáticas políticas, sociais, entre outras (Navarro, 2021). Nesse sentido, o Pensamento Computacional é uma temática imprescindível para a ampliação do papel da matemática como treinamento das habilidades de solução de problemas diversos.

As políticas públicas têm papel importante na promoção e defesa do Pensamento Computacional, de acordo com a BNCC. Através de investimentos em infraestrutura, recursos educacionais e outras iniciativas, as políticas públicas têm contribuído para o desenvolvimento do Pensamento Computacional e para a formação de uma geração de alunos bem-preparados para o mundo digital.

É inegável a necessidade de formação docente que se volte para a ampliação das próprias habilidades e dos alunos, favorecendo a utilização do Pensamento Computacional no contexto de aprendizagem em sala de aula. Como aponta o estudo realizado por Almeida e Miranda (2023), destaca-se a relevância da formação docente para o uso do Pensamento Computacional e seus conceitos no processo de aprendizagem, para que possam compreender os conceitos e sua aplicabilidade no contexto educacional.

Portanto, investimentos em infraestrutura que foram feitos para promover e defender o Pensamento Computacional de acordo com a BNCC incluem a implementação de equipamentos eletrônicos, como computadores, laptops, tablets e outros dispositivos, além da

oferta de cursos e palestras sobre o assunto, bem como a disponibilização de materiais educativos e de recursos para a pesquisa.

Na revista da Sociedade Brasileira de Educação Matemática, regional do Paraná, no texto intitulado *Geometria Escolar e os pensamentos, matemático e computacional***,** do professor doutor Sérgio Carrazedo Dantas, foi apresentada uma crítica à BNCC, pois ela apresenta o pensamento computacional somente através de fluxogramas. O autor apresenta sete proposições de uso dos pensamentos matemático e computacional integrados à Geometria Escolar (Dantas, 2023).

De acordo com Dantas (2023), ao ler a BNCC, percebe-se rapidamente que ela propõe uma centralização curricular alicerçada em avaliações em larga escala. Um posicionamento que é passível de críticas quanto a vários aspectos, como ao que consta na parte destinada à Matemática do Ensino Fundamental e do Ensino Médio quanto à utilização de tecnologias digitais e quanto ao pensamento computacional.

Ao contemplar o uso de tecnologias digitais, a BNCC trata como é utilizado em outros países (Estados Unidos, Austrália e Inglaterra), as preocupações e anseios desses países e quais os processos para a utilização das tecnologias para um verdadeiro desenvolvimento de competências gerais para a Educação Básica.

O Pensamento Computacional envolve conhecimentos amplos e diversificados em aplicabilidade que podem favorecer o aprendizado de diversas maneiras:

> Pensamento Computacional no ensino de Matemática pode ser uma abertura para o desenvolvimento de saberes e um caminho para exercitar a resolução de problemas. O Pensamento Computacional pode ser desenvolvido tanto na abordagem plugada quanto na desplugada, sendo que a abordagem plugada faz uso de tecnologias digitais e a abordagem desplugada pode ser utilizada sem o uso de computador ou qualquer recurso eletrônico (Lopes, 2022, p. 20).

Portanto, o desenvolvimento do Pensamento Computacional pode ser trabalhado de maneiras variadas, visando a contribuir para ampliar as formas como problemas podem ser solucionados, potencializando as práticas de raciocínio dos alunos, e favorecendo a busca de conhecimentos que sejam determinantes para uma vida com dinamismo e agilidade de pensamento, em consonância com a realidade da sociedade contemporânea. Assim, garantem-se oportunidades para o indivíduo, dentro e fora do ambiente escolar.

3 O SCRATCH

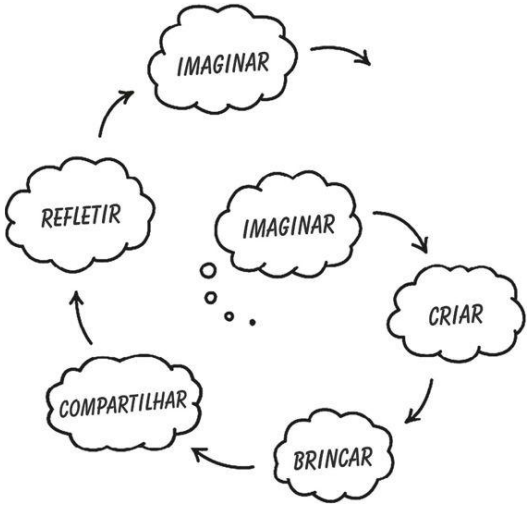
Em 1982, Resnick conheceu Papert, que trabalhava como jornalista e foi a uma palestra com ele, pois estava escrevendo sobre Ciência e Tecnologia. Os computadores pessoais (*personal computer*, em inglês - PC) tinham acabado de surgir e Papert percebeu que ele gerava curiosidade nas crianças. Ele tinha criado a linguagem *Logo* e observou como incentivar as crianças a criarem na linguagem Logo. Utilizando como base a linguagem Logo, Resnick trabalhou com crianças e adolescentes em contraturno, em um projeto de título *Computer Clubhouse*, plataforma em que os jovens podiam explorar, experimentar e se expressar, criando seus próprios jogos.

Resnick percebeu, durante o trabalho com o projeto, que para que as crianças pudessem criar seus próprios jogos e animações, era necessário que aprendessem uma linguagem de programação. O Scratch surgiu para atender às reivindicações do público do *Clubhouse*, uma rede de centros de aprendizagem em contraturno. Para aperfeiçoar a plataforma, foram considerando os feedbacks das crianças, que se envolviam e realizavam seus próprios projetos (jogos, animações, simulações).

Para Resnick (2020), as mudanças devem acontecer devagar, para que o aluno possa construir seus jogos, suas ideias, sua criatividade, transformando seu pensar em algo executável. Ele ressalta que não devem acontecer aulas expositivas, com o professor dando as informações e os alunos absorvendo somente essas informações. Segundo ele, essa técnica não funciona mais para construir um aprendizado de qualidade. Acredita-se que os alunos podem mais, que devem ser pesquisadores e criativos.

A Figura 1 exibe a espiral de aprendizagem criada por Resnick para defender o processo criativo.

Figura 1 - Espiral da aprendizagem criativa



Fonte: Resnick (2017, p. 7).

Resnick (2020) defende que primeiro imaginamos algo, depois criamos e a partir dessa criação, imaginando e brincando com aquilo que foi criado, pode haver compartilhamento e discussão com seus pares, outras pessoas. Realizadas as reflexões a partir do que foi criado, o processo pode se repetir, e por isso vem a imagem em forma de espiral.

Mitchel Resnick argumenta, em seu livro intitulado *Jardim de Infância para a vida toda*, que sua intenção era a criação da plataforma Scratch visando ao engajamento do público infantil, denominando-o aprendizagem criativa, inspirado na maneira como as crianças do jardim de infância aprendem (Resnick, 2020).

O Scratch é uma linguagem de programação visual gratuita que está disponível tanto *on-line* quanto *off-line*. Baseada na linguagem Logo, de Seymond Papert, seu desenvolvimento teve início em 2003, e em 2007 foi lançada ao público. Desde então, o Scratch tem ganhado força e popularidade, tornando-se um dos softwares de programação mais acessíveis e populares na educação.

> Muitas pessoas pensam que o Scratch é uma linguagem de programação, e é mesmo. Mas quem trabalha com o Scratch sabe que ele vai muito além disso. Desde o início, nosso objetivo era criar um tipo de comunidade de aprendizagem on-line em que jovens pudessem criar, compartilhar e aprender de forma colaborativa, no espírito de uma escola de samba (Resnick, 2020, p. 125).

O Scratch pode ser descrito como um ambiente de programação que possibilita a criação de histórias pelos jovens, através de interatividade, com jogos e simulações. Além disso, o Scratch traz a possibilidade de compartilhamento das criações no contexto de uma comunidade virtual, o que permite trocar ideias e conteúdos com outros programadores em qualquer lugar do mundo (Brennan; Resnick, 2012).

O objetivo central é combinar conceitos de programação com elementos de arte e design.

Assim, é importante analisar o Scratch como um recurso para ampliação e solidificação do pensamento computacional através de atividades práticas e um diálogo devidamente contextualizado em relação a esse conceito.

Os autores descrevem que, em seus estudos, observaram atividades expostas na comunidade virtual do Scratch, também em workshops do Scratch, definindo, com base nesse estudo, que o pensamento computacional engloba três dimensões:

> [...] conceitos computacionais (os conceitos que os designers empregam enquanto programam), práticas computacionais (as práticas que os designers desenvolvem à medida que programam) e perspectivas computacionais (as perspectivas que os designers formam sobre o mundo ao seu redor e sobre si mesmos) (Brennan; Resnick, 2012, p. 03).

Segundo os autores, a utilização do Scratch possibilita aprendizado e desenvolvimento do pensamento computacional através da assimilação de conceitos computacionais, práticas que são construídas durante a programação, e ampliando suas perspectivas sobre o universo que cerca seus usuários.

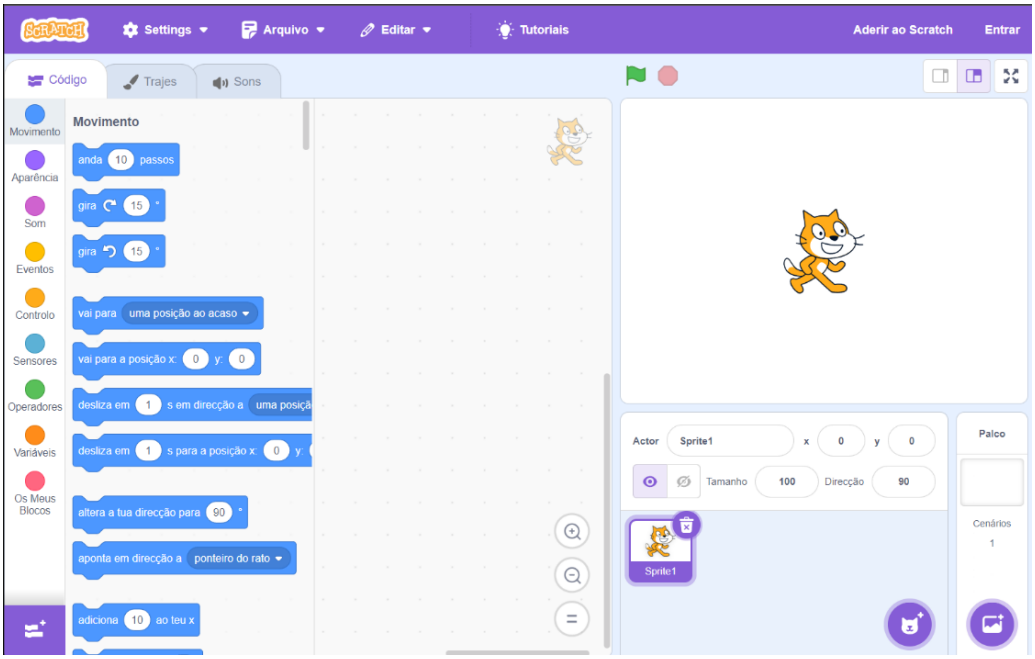
Além dessa questão, em seus estudos, os autores conversaram com *Scratchers*, que pontuaram a ampliação em relação à compreensão de si mesmos, suas relações interpessoais e do universo tecnológico que faz parte de suas vidas (Brennan; Resnick, 2012).

Ainda no contexto de seu estudo, os autores pontuam que, diante de questionamentos sobre as qualidades do Scratch, alguns jovens destacaram que o ambiente de programação tem possibilidades, mas também apresenta limitações. Ainda, que alguns jovens membros da comunidade se reuniram para elaborar uma versão similar ao Scratch com a inserção de blocos, que julgavam importantes no recurso, também desenvolvendo um site para que outras pessoas pudessem ter acesso a essa versão do Scratch. Esse processo reitera o Scratch como um recurso que contribui para a construção do pensamento e para o aprendizado da programação, ensinando sobre flexibilidade e auxiliando na formação de conhecimentos que viabilizam oportunidades de mudar o que se acredita que precisa ser mudado (Brennan; Resnick, 2012). Dessa forma, concretiza-se a ideia de que o pensamento computacional amplia possibilidades e ultrapassa limites para a solução de problemáticas variadas.

3.1 Recursos do Scratch

Para iniciar o Scratch, o usuário deve acessar seu site (http://scratch.mit.edu/) e clicar no link **Criar**. Isso deve conduzi-lo à interface do editor de projetos do Scratch, mostrado na Figura 2.

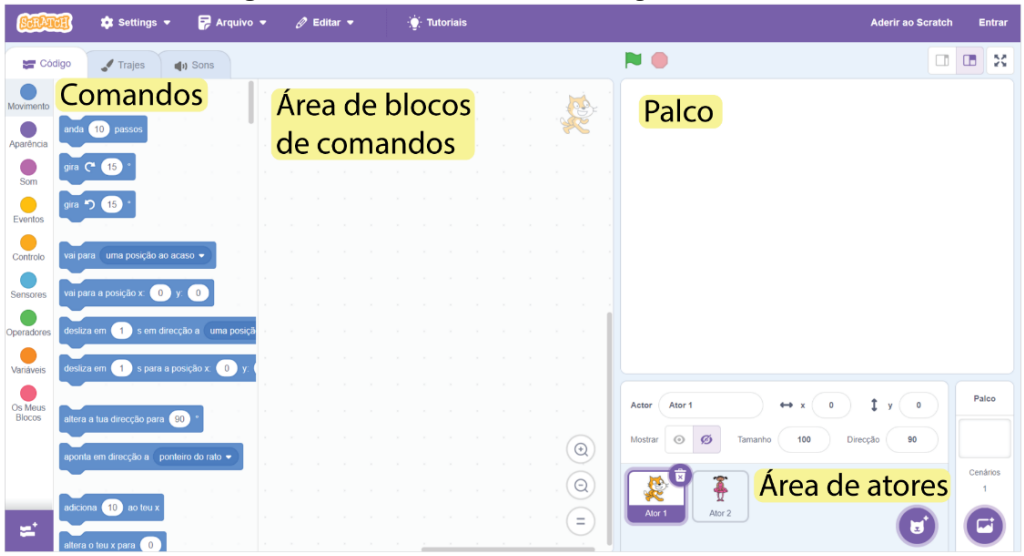
Figura 2 - Interface do Scratch



Fonte: Produzida pela autora.

A Figura 3 apresenta a interface do Scratch com algumas regiões nomeadas.

Figura 3 - Interface do Scratch com regiões nomeadas



Fonte: Produzida pela autora.

Para programar no Scratch, utilizam-se os blocos da seção **Comandos**, os quais estão organizados em categorias: Movimento, Aparência, Som, Eventos, Controles, Sensores, Operadores, Variáveis e Os Meus Blocos.

A seção **Área de blocos de comandos** é o local onde são disponibilizados os blocos encaixados uns aos outros para compor instruções para controlar ações ou aparências dos atores.

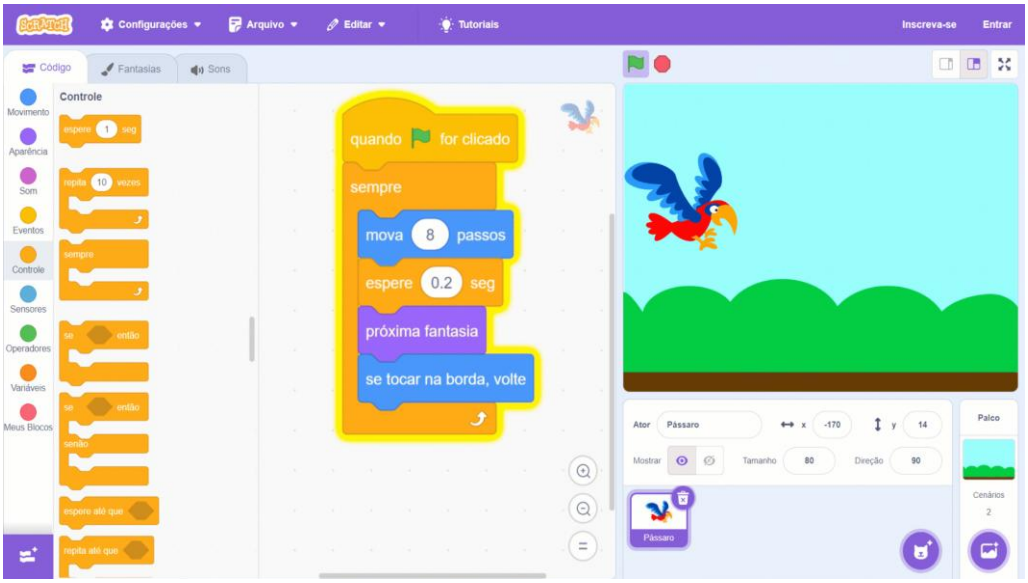
O **Palco** é a região visual do programa. Nesse espaço são disponibilizados atores, textos e cenários.

Os atores são disponibilizados em um subcampo, a **Área de atores**. Para ativá-los e programá-los, primeiramente deve-se clicar sobre eles para tornar sua área de comandos ativa.

Na aba **Fantasias** é que se articulam imagens dos atores para que se produzam estados diferentes em sintonia com movimentos.

Na aba **Sons** podem-se utilizar os sons que já existem no aplicativo, adicionar sons do computador, criar seus próprios sons e até recortar partes de sons para serem adicionados aos projetos.

Figura 4 - Interface do Scratch com regiões nomeadas



Fonte: Produzida pela autora.

A Figura 4 corresponde a uma animação de um pássaro no Scratch que voa de um lado ao outro da tela, de forma ininterrupta. No lado esquerdo da tela, na **Área de blocos de comandos**, há um conjunto de comandos que produzem a animação.

3.2 Conceitos computacionais

Em um texto publicado em 2012 sob o título *New frameworks for studying and assessing the development of computational thinking*, Brennan e Resnik discorreram sobre a perspectiva que têm a respeito do pensamento computacional e como ele pode ser explorado e operacionalizado com a utilização do Scratch. Eles estudaram construções de crianças de 08 a 12 anos publicadas na plataforma virtual do Scratch e, em seguida, coletaram dados via

entrevistas com algumas delas, e construíram uma perspectiva sobre pensamento computacional baseada em três perspectivas-chave. Segundo os autores,

> [...] ao estudar a atividade na comunidade on-line do Scratch e nos workshops do Scratch, desenvolvemos uma definição de pensamento computacional que envolve três dimensões principais: **conceitos computacionais** (os conceitos que os designers empregam enquanto programam), **práticas computacionais** (as práticas que os designers desenvolvem enquanto programam) e **perspectivas computacionais** (as perspectivas que os designers formam sobre o mundo ao seu redor e sobre si mesmos)[3] (Brennan; Resnik, 2012, p. 03, tradução e grifos nossos).

Conceitos computacionais dizem respeito às técnicas utilizadas durante a codificação para obter resultados animados, dinâmicos, estéticos e interativos. No caso do Scratch, a codificação acontece construindo blocos de instruções. Segundo os autores, há sete conceitos computacionais operacionais no Scratch: *sequências*, *loops*, *paralelismo*, *eventos*, *condicionais*, *operadores* e *dados* (Brennan; Resnick, 2012),

Com base nesses autores, construímos exemplos e discorremos sobre os sete conceitos computacionais. O objetivo foi argumentar sobre como a prática desses conceitos computacionais colocam em evidência processos constituintes do pensamento computacional, a saber: formulação de problemas, decomposição, reconhecimento de padrões, abstração, produção de algoritmo e depuração.

3.2.1 Sequências

Segundo Brennan e Resnick (2012, p. 03, tradução nossa),

> Um conceito fundamental na programação é que uma atividade ou tarefa específica é expressa como uma série de etapas ou instruções individuais que podem ser executadas pelo computador. Como uma receita, uma sequência de instruções de programação especifica o comportamento ou a ação que deve ser produzida[4].

Para exemplificar, considere que seja construída no Scratch uma animação de repetição ilimitada, em que um gato caminha na tela, para, mia, aguarda alguns segundos, volta a caminhar e o ciclo de ações se repete. Porém, ao tocar em uma das bordas verticais do palco, o gato vira para o lado oposto e continua a repetir a sequência. O conceito *sequência* é utilizado para obter tal conjunto de ações integradas e em série.

---

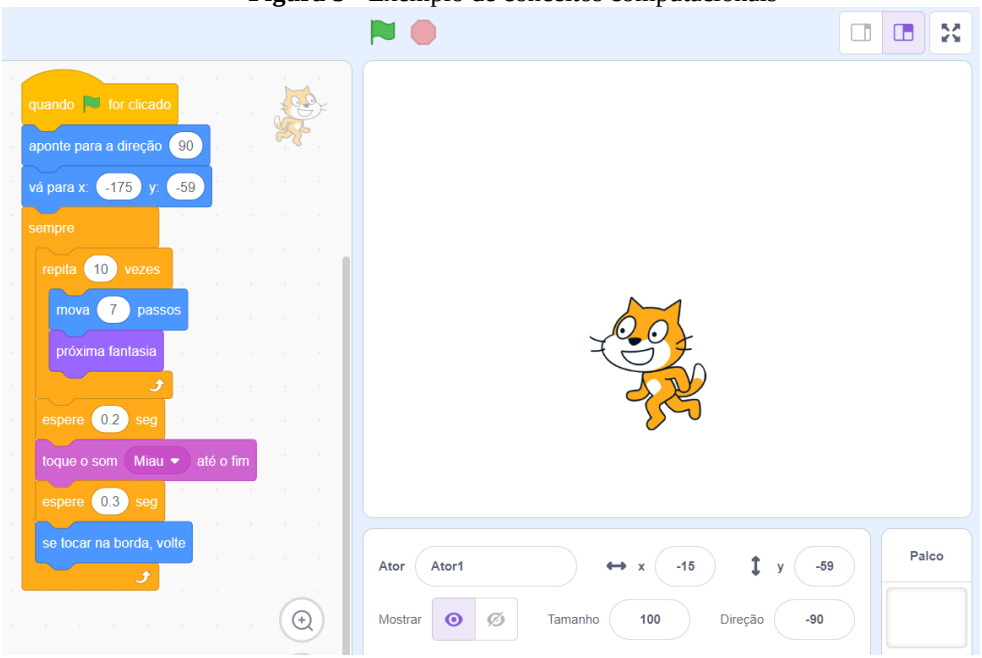
[3] *[...] by studying activity in the Scratch online community and in Scratch workshops, we have developed a definition of computational thinking that involves three key dimensions: computational concepts (the concepts designers employ as they program), computational practices (the practices designers develop as they program), and computational perspectives (the perspectives designers form about the world around them and about themselves).*

[4] *A key concept in programming is that a particular activity or task is expressed as a series of individual steps or instructions that can be executed by the computer. Like a recipe, a sequence of programming instructions specifies the behavior or action that should be produced.*

Nessa formulação, ou seja, escrita em linguagem materna sobre uma situação, já há o emprego do conceito de sequência, pois ao materializar em palavras o que deve ocorrer no Scratch, manifestam-se processos que devem ocorrer obedecendo certa ordem. Assim, o conceito *sequência* contribui para pôr em jogo a formulação de problemas, assim como formular um problema pode implicar a utilização de uma noção de sequência.

A imagem abaixo mostra blocos ordenados no Scratch em que o problema proposto é resolvido.

Figura 5 - Exemplo de conceitos computacionais



Fonte: Produzida pela autora.

Note que, no Scratch, foram construídos um ator e um conjunto de comandos que definem um estado inicial desse ator, definem um *loop* contínuo de blocos a serem repetidos e articula movimentos com trocas de aparências, lapsos temporais e emissão de sons. Esses blocos de comandos não foram posicionados em uma ordem qualquer, mas foram pensados com o objetivo de obter o que é descrito na formulação do problema.

Os dois primeiros blocos, “aponte para a direção 90 graus” e “vá para x – 175 e y – 59”, são responsáveis por posicionar o gato em um lugar de partida e se voltar para a direção que realizará sua caminhada. Na sequência, abre-se um *loop infinito* (sempre), ou seja, um bloco de comandos que repetirá ações internas enquanto o Scratch estiver em execução. Os blocos no interior do controle são responsáveis por executar o deslocamento do ator, enquanto simultaneamente ele troca de fantasias para dar a sensação visual de caminhada. Há, ainda,

blocos de lapso temporal para parar por alguns instantes a execução do programa e, em seguida, emitir o som de miado.

Um segundo processo de pensamento computacional pode ser identificado na sequência de blocos que acabamos de descrever, a decomposição. Notamos que as ações singulares do ator no palco correspondem a transladar, trocar de fantasias, paralisar e emitir som. Para cada uma delas foram dedicadas certas unidades de comandos que executam essas ações ordenadamente, o que significa que foram pensadas por quem codificou a partir de certa decomposição de uma ação composta em um conjunto de ações singulares. Em outras palavras, a decomposição ocorreu em montar um conjunto de ações singulares que, executadas em uma ordem específica, compõem a ação de miar enquanto caminha.

A caminhada consiste na combinação de dois blocos: “mova 7 passos” e “próxima fantasia”. O primeiro comando sempre é responsável por fazer o ator deslizar na direção em que está apontando (90 graus). O segundo comando, “próxima fantasia”, que ocorre imediatamente após seu deslocamento, muda a fantasia para uma próxima, em que as pernas do ator estão em posições diferentes. Essa combinação entre translação e troca de fantasia compõe uma abstração do que é o caminhar do ator. O comando “repetir” torna essa abstração executada reiteradas vezes, produzindo a sensação visual de uma caminhada.

A produção de algoritmo é manifestada via automação da abstração anterior. Assim, a combinação de comandos em blocos e seu controle de fluxo por meio de comandos de repetição produzem a solução esperada, o que compõe a codificação ou o algoritmo que produz a caminhada intercalada por miadas do ator.

3.2.2 Loops

Segundo Brennan e Resnick (2012, p. 03), “Loops são mecanismos para executar uma mesma sequência múltiplas vezes”. Em outras palavras, pode ser necessária uma repetição de uma parte de um conjunto de blocos para produzir um efeito visual desejado, como a emissão de um som, que necessita da entrada de vários dados. Nesse caso, em vez de construir um conjunto de blocos que se repetem, utiliza-se um controle de repetição.

Na animação do gato que caminha enquanto mia, foram utilizados dois blocos de repetições. O primeiro, conforme já argumentado, produz o movimento repetindo dez vezes “mova 7 passos” e “próxima fantasia”. O segundo produz a repetição contínua de um bloco maior de códigos enquanto o “programa” estiver em execução. Nesse caso, foi utilizado o controle “sempre”. Em ambos os casos há automatizações de abstrações.

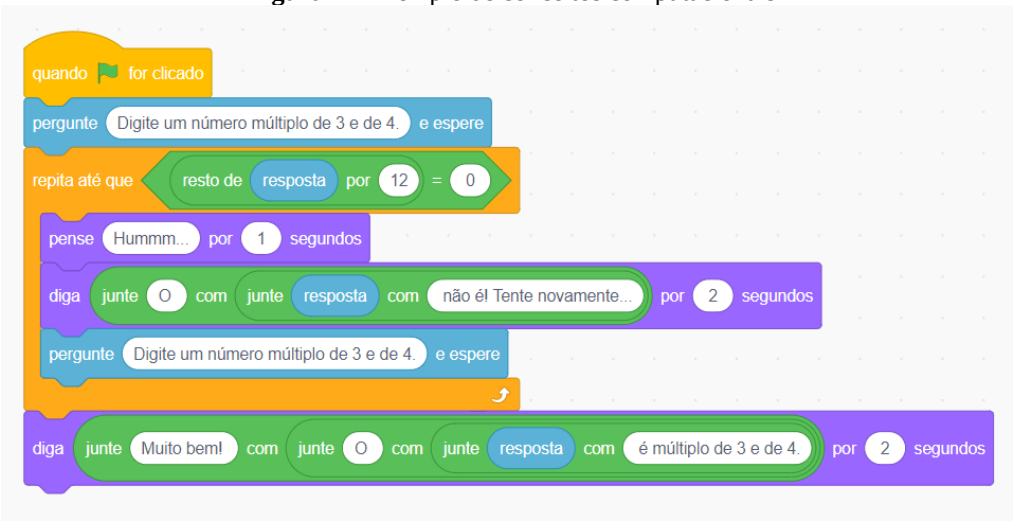
No exemplo a seguir, o ator solicita que o utilizador do “programa” digite um número que seja múltiplo simultaneamente de 3 e de 4.

Figura 6 - Exemplo de conceitos computacionais

Fonte: Produzida pela autora.

Enquanto isso não ocorrer, ele não interromperá suas reiteradas solicitações e não executará o conjunto de códigos que segue ao bloco de repetição. Trata-se de uma repetição que não possui uma quantidade fixa de execuções, mas que depende da interatividade do usuário com o objeto.

Figura 7 - Exemplo de conceitos computacionais



Fonte: Produzida pela autora.

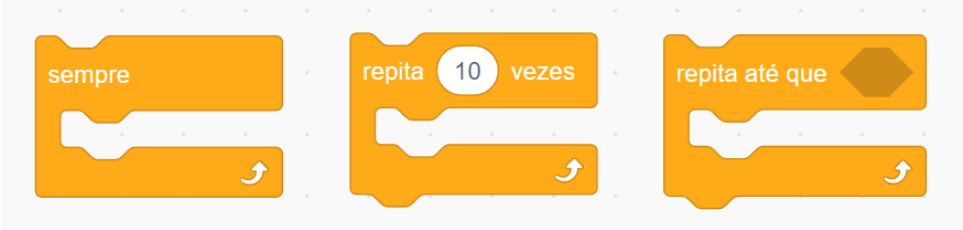
Essa interatividade prevista pelo codificador do objeto manifesta a atenção específica a um evento do usuário, a previsão de um problema e seu tratamento. Portanto, é uma compreensão possível graças à decomposição de um problema maior em suas partes menores constituintes.

Figura 8 - Exemplo de conceitos computacionais.

Fonte: Produzida pela autora.

Notamos que o Scratch apresenta três possibilidades de controles de repetições em comandos com funções específicas.

Figura 9 - Exemplo de conceitos computacionais



Fonte: Produzida pela autora.

Cada um desses comandos serve a um propósito distinto para controle de repetições de comandos ou de blocos de comandos. A escolha por um deles dependerá do resultado almejado: não interromper a repetição, repetir comandos uma quantidade limitada ou repetir até que uma condição seja satisfeita. O emprego de cada um deles dependerá da decomposição do problema e da concepção de um algoritmo com vista a certo resultado.

Os *loops* podem servir, ainda, para exploração de padrões e para depuração. No primeiro caso, o usuário do Scratch constrói uma sequência para apresentar certo padrão de comportamento, por exemplo, a produção de uma sequência numérica.

Quanto à percepção de um padrão, em alguns casos, a análise de um caso particular, o primeiro traz a certeza de que o bloco de comandos executa certa ação. Porém, não garante que a sequência de comandos produzirá um resultado esperado em série. Então, utilizar um *loop* pode auxiliar na percepção de um padrão ou na produção de um padrão.

Quanto à depuração, um *loop* pode ser útil para repetir um bloco de comandos enquanto variáveis são modificadas. Assim, pode-se perceber o efeito dos valores assumidos no que se

deseja como resultado visual. Por exemplo, em um conjunto de comandos utilizado para construir polígonos regulares dada a medida do lado e a quantidade de lados pode-se perceber que a medida dos lados deve diminuir a medida que a quantidade de lados aumenta.

3.2.3 Paralelismo

De acordo com Brennan e Resnick (2012, p. 04), "A maioria das linguagens de computador modernas suporta paralelismo – sequências de instruções acontecendo ao mesmo tempo. Scratch suporta paralelismo entre objetos".

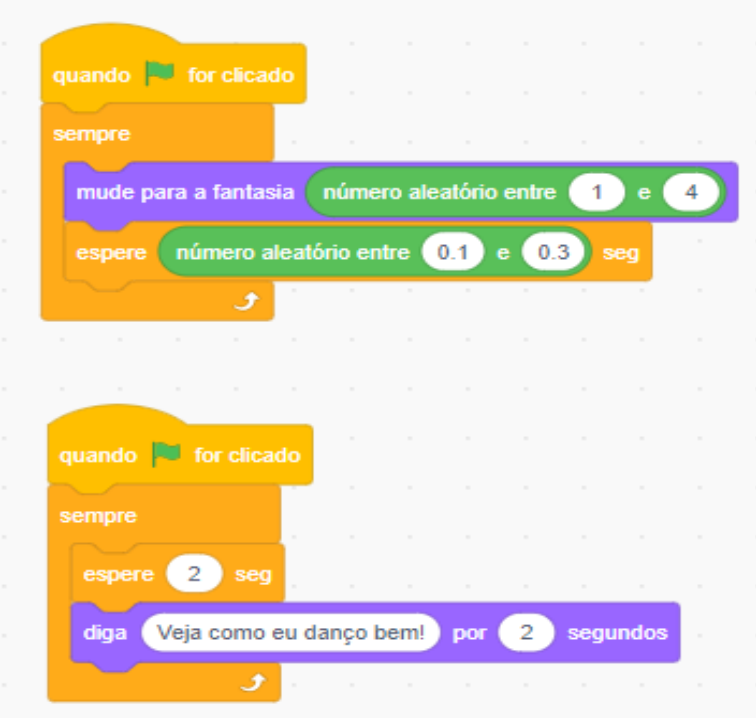
A ideia de paralelismo envolve o acontecimento de diferentes eventos simultaneamente, e um acontecimento não interfere no outro.

Segundo Silva (2019, p. 19), paralelismo se define em "organizar recursos com o fim de realizar tarefas simultaneamente com o intuito de alcançar um objetivo comum".

No pilar de formulação de um problema, determinamos que queremos que um ator, a bailarina, dance e ao mesmo tempo fale.

No exemplo abaixo, percebemos que uma bailarina dança e ao mesmo tempo fala: "Veja como eu danço bem!"

Figura 10 - Exemplo de conceitos computacionais



Fonte: Produzida pela autora.

Para criar esse evento, precisamos utilizar o paralelismo: duas ações diferentes acontecendo simultaneamente. Se colocarmos no mesmo evento e mesmo bloco, ela para e não

executa de acordo com o pilar formulado. Dessa maneira, foram criados dois comandos em que o usuário clica na bandeira verde, e os comandos terão a realização simultânea da programação, no caso, que ela dance e fale ao mesmo tempo, demonstrando o paralelismo.

Ao formularmos um problema há paralelismo, pois imaginamos/idealizamos ações de um objeto ocorrendo simultaneamente, como o caso da bailarina. Imaginamos/idealizamos, ainda, objetos interagindo uns com os outros e influenciando ou não uns aos outros mutuamente.

O pilar de decomposição vem acompanhado pela criação em dois comandos, que é a execução em partes da programação do jogo digital: no primeiro comando, a bailarina dança; no segundo, ela fala. Percebemos, aqui, o paralelismo na execução de dois comandos, que são executados paralelamente.

No projeto, aparece o pilar da abstração ao dividir os comandos em dois blocos. Caso seja necessário alterar o projeto, o usuário poderá abstrair e corrigir possíveis erros ou ampliações e realizar incrementos.

Figura 11 - Exemplo de conceitos computacionais

Fonte: Produzida pela autora.

No Pensamento Computacional, realizar a ação em dois comandos, dois eventos, mostra claramente o pilar da decomposição. O primeiro comando faz a bailarina mudar sua fantasia, e o segundo faz a bailarina falar.

Na produção de algoritmos, percebe-se uma sequência finita de passos, executáveis em um determinado tempo, com instruções ordenadas, passando pela formulação do problema,

decomposição, reconhecimento de padrões e abstração. Segundo Wing (2006), é o pilar que agrega os outros pilares.

3.2.4 Eventos

No Scratch há alguns blocos que podem iniciar um evento, assim como possibilita utilizar blocos em alguns comandos para programar um jogo, partindo da formulação do problema.

Os eventos acontecem no computador quando o usuário pensa em programar no Scratch o seu jogo, por exemplo, através da tecla espaço, se clicar na borda, se clicar em determinada letra. O usuário pode tratar um evento a partir dessas ações, e criará um jogo para depois executá-lo. Ainda pode construir a solução utilizando as estruturas conceituais que denotam a ocorrência de eventos e as ações. A ocorrência do evento mostra os blocos em ação, variando as quantidades e ações.

Abaixo, apresentamos um exemplo mostrando a utilização do conceito computacional evento.

Figura 12 - Exemplo de conceitos computacionais

Fonte: Produzida pela autora.

Apresentamos, no exemplo, uma programação de um jogo em que uma bola de beisebol começa no centro do palco. O primeiro evento, iniciar o jogo, quando disparado, faz com que a bola de beisebol vá ao centro, mova aleatoriamente pelo palco, e quando toca nas bordas, ela volta para o palco e assim sucessivamente.

O segundo evento tratado é o clique da tecla “a”, que para a bola no palco.

Na imagem abaixo são exibidos alguns “topos” de conjuntos de comandos que são úteis para tratar eventos do usuário ou de objetos de um projeto no Scratch.

Figura 13 - Exemplo de conceitos computacionais

Fonte: Produzida pela autora.

A sensibilidade para os eventos que ocorrem em um projeto no Scratch e seu tratamento com conjuntos de comandos podem produzir interatividade entre o objeto construído, um jogo, por exemplo, e o usuário. O tratamento dos eventos pode ser traduzido em ações que o programa pode executar como movimentos, emissão de sons e solicitações de ações ao usuário, o que pode levar à mobilização de processos do Pensamento Computacional.

Iniciando com a formulação do problema, percebemos que o usuário criou jogo digital com intuito que uma bola de beisebol possua movimentos aleatórios no palco, e quando for clicada a tecla “a” ela pare. Após a formulação do problema, pode-se criar outros eventos, ou programar novos comandos para um mesmo jogo digital.

Percebe-se que a decomposição está presente no evento criado, pois quando dividiu em dois eventos, observamos a decomposição em partes menores para que seja executável.

Também há o reconhecimento de padrões, mostrando que podemos utilizar o mesmo evento e trocar a tecla que queremos que seja utilizada para que a bola pare; podemos programar qualquer tecla, esse padrão nos permite utilizar o mesmo evento.

A produção do algoritmo, focando em cada parte do problema, será sua resolução, raciocínio lógico utilizado para que o que foi idealizado partindo da formulação do problema, seja executável.

A abstração acontece em cada bloco utilizado, realizando sua execução e partindo para suas correções e possíveis incrementos.

3.2.5 Condicionais

Permitem ao usuário que sejam tomadas decisões baseadas em algumas condições. As condições são de teste; dependendo de quais delas são fornecidas, podem ser feitas perguntas,

questionamentos, e de acordo com as respostas, decidir o que deve ser feito no projeto que esteja sendo idealizado. Percebemos, aqui, a formulação do problema, assim como a depuração, pois através desses processos se inicia a criação de um jogo digital ou um projeto, e a depuração permite testar se está funcionado de acordo com o que foi pensado.

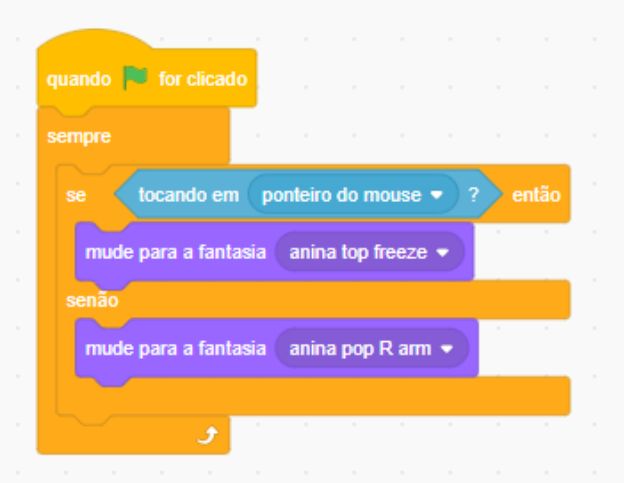
De acordo com Borges, Noronha e Backes (2022), há casos que precisam de blocos de comandos que sejam executados se a condição for verdadeira, e para isso precisamos de uma estrutura de seleção ou um comando de controle condicional que permita selecionar o conjunto de comandos a ser executado.

Por condição entende-se qualquer expressão relacional (ou seja, que use os operadores >, <, >=, <=, == ou!=) que resulte em uma resposta do tipo verdadeiro ou falso (Backer, 2022).

Os blocos de condicionais no Scratch ficam em **controle**, e apresentam-se em dois tipos: se então; se então senão.

No exemplo abaixo, da Figura 14, percebe-se que foi utilizada a tecla de controle: se tocando em ponteiro do mouse, então mude para a fantasia “anina top freeze”, se não mude para a fantasia “anina pop R arm”. Isso demonstra que existe uma condição para que mude a fantasia, e essa condição depende que o ponteiro do mouse seja tocado. Um exemplo que demonstra a utilização das teclas condicionais parte dos conceitos computacionais.

Figura 14 - Exemplo de conceitos computacionais



Fonte: Produzida pela autora.

No Scratch existem alguns blocos que funcionam como condicionais, como demonstrado pela figura 15.

Figura 15 - Exemplo de conceitos computacionais

Fonte: Produzida pela autora.

Na Figura 15, no primeiro bloco de condicionais há *se, então*: com espaço vazio para ser adicionado um questionamento, pergunta. Ele tem um espaço vazio com formato hexagonal, que é chamado pela ciência da computação de expressão booleana, sendo uma maneira de questionar algo para que a resposta seja executada como verdadeira; é um desvio condicional simples.

No segundo bloco da figura está o bloco *se, então, senão*, também com mesmas características do primeiro bloco. A diferença está no caso de a resposta ao questionamento ser falsa, tendo como resposta aquilo que estará programado para ser executado; é um desvio condicional composto.

Percebe-se que, na formulação do problema, sua idealização partiu de criar um projeto em que, tocando com o mouse, o ator mude sua fantasia. A criação demonstra que foi utilizada a abstração partindo da relevância em criar o bloco, focando nas partes principais, no caso aqui demonstrado pelo bloco, com condição de desvio condicional composto. A produção de algoritmos é o processo que demonstra, nesse projeto, os passos que foram utilizados para que o projeto seja executável, cada bloco e sua depuração para que funcionasse corretamente de acordo com a formulação do problema.

O reconhecimento do padrão está no bloco da fantasia, sendo reutilizado e mudando somente a fantasia a ser utilizada.

3.2.6 Operadores e dados

Segundo Brennan e Resnick (2012), os operadores podem ser utilizados para uma série de operações matemáticas (incluindo adição, subtração, multiplicação, divisão, bem como funções, como seno e expoentes, assim como operações com *strings* [concatenação]).

Os dados envolvem armazenamento, recuperação e atualização de valores. Atualmente, o Scratch oferece dois contêineres para dados: variáveis (que podem manter um único número ou *string*) e listas (que podem manter uma coleção de números ou *strings*).

De acordo com Marji (2014), o gerenciamento de dados por meio do Scratch pode ser feito por *variáveis* ou *listas*. Suporta três tipos de dados: booleanos, números e scripts.

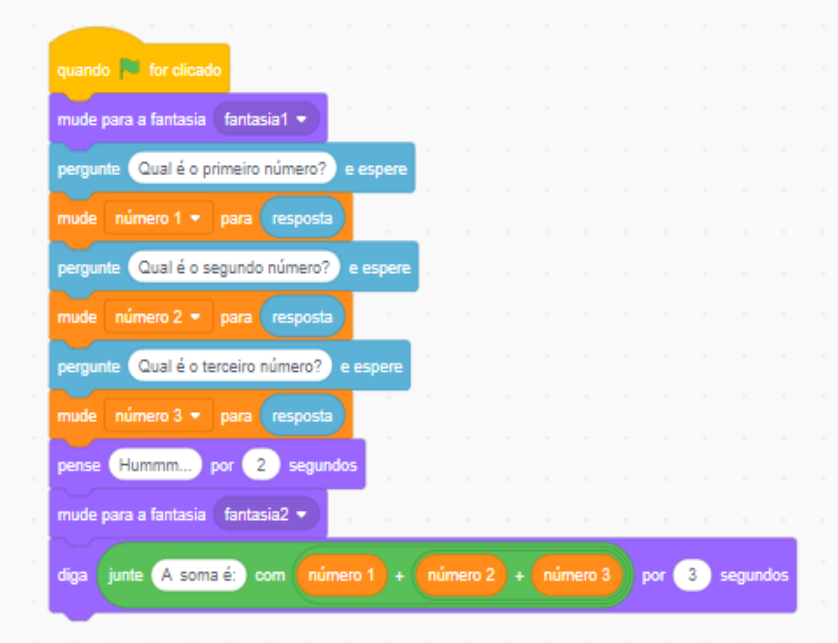
Um booleano apresenta dois valores, *verdadeiro* e *falso*, usados para testar uma ou mais condições, de acordo com o resultado, selecionando um caminho diferente.

Na variável numérica, podem-se utilizar números inteiros ou decimais, através dos operadores, com arredondamento de números.

String é uma sequência de caracteres que podem ser utilizados na programação, como letras, números, símbolos, incluindo espaços.

No exemplo abaixo há idealização de um problema que o usuário pode ter pensado na soma de três numerais.

Figura 16 - Exemplo de conceitos computacionais

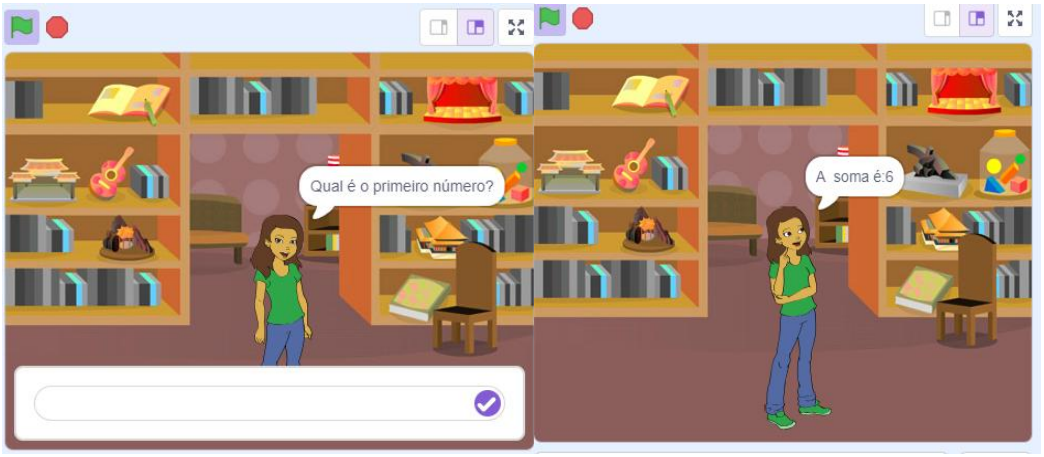


Fonte: Produzida pela autora.

Nesse jogo digital, na formulação do problema, o jogador programou para que digite um numeral; depois, o segundo numeral; e por fim, mais um. No final será dada a soma total dos três numerais digitados.

A Figura 16 mostra que primeiro o ator pensa, aparece a palavra HUMMMM, como forma de pausar e dar um suspense na brincadeira, aparecendo a soma dos três numerais no final.

Figura 17 - Exemplo de conceitos computacionais



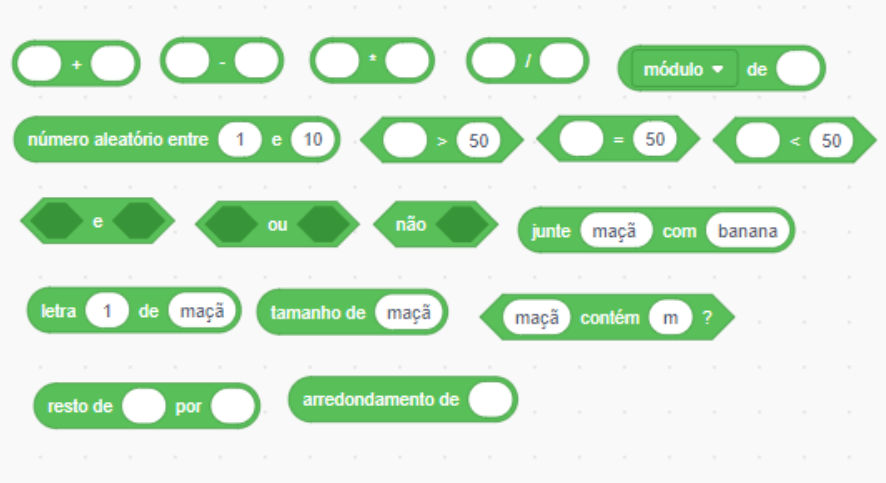
Fonte: Produzida pela autora.

Percebemos, através da programação dos blocos, o reconhecimento de padrões, que vem a partir do bloco de perguntas, mostrando que o padrão se repete, mudando apenas a pergunta feita.

A decomposição também aparece na utilização dos blocos separados; depois, fazendo a junção com a mudança da pergunta, utilizando a produção de algoritmos para que a programação produza o resultado esperado. Nessa fase, a abstração já foi utilizada na verificação, testagem do jogo digital para verificar sua execução. De acordo com Brennan e Resnick (2012), os operadores permitem realizar operações numéricas, lógicas, *strings* e os dados envolvem armazenamento, recuperação e atualização. O Scratch oferece as variáveis e as listas como armazenamento de dados, que em computação, segundo Backer (2022), uma variável é uma posição de memória em que podemos guardar determinado dado ou valor e modificá-lo ao longo da execução do programa.

No Scratch existe a possibilidade de utilização de vários operadores. A figura 18, na página seguinte, mostra alguns operadores que o Scratch possui.

Figura 18 - Exemplo de conceitos computacionais



Fonte: Produzida pela autora.

Os blocos relacionais são utilizados com dados, que podem, em um programa, ser denominados valores ou quaisquer outros dados, e podem ser retornados valores verdadeiros ou falsos, utilizados juntamente com as estruturas condicionais.

4 UM CURSO SOBRE PENSAMENTO COMPUTACIONAL

O curso *Pensamento Computacional na construção de Jogos com o Scratch* foi idealizado a partir de leituras e estudos realizados pelo grupo Autômato, sobre o livro intitulado *Computação na Educação Básica, fundamentos e experiências* (Raabe; Zorzo; Blikstein, 2020).

Um dos capítulos desse livro apresenta a realização de uma oficina de produção de jogos digitais que foi criada e aplicada em 2013 no Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP) e na Universidade do Chile.

Barcelos *et al.* (2020) argumentam que, após a realização da oficina, a primeira constatação de sua equipe foi a motivação dos alunos em aprender sobre o Pensamento Computacional, em particular sobre programação de computadores. Para atender a essa demanda, a pesquisadora e seus colegas optaram por jogos digitais, e propuseram algumas diretrizes para as atividades da oficina.

A primeira diretriz sustenta que a construção de jogos deve motivar o desenvolvimento de todas as atividades da oficina. É apresentada uma visão segundo Basawapatna *et al.* (2013), mostrando a importância de manter o aprendiz o mais próximo possível do estado de fluxo.

Para os autores, apresentar uma quantidade considerável de conceitos antes de sua aplicação seria o equivalente a introduzir novas habilidades sem um desafio à altura do uso dessas habilidades.

Como segunda diretriz as atividades devem levar progressivamente à construção de um jogo completo. A proposta é que os estudantes tenham acesso a pequenos programas prontos, com os quais possam interagir a fim de compreender seu funcionamento. Após essa etapa, são convidados a introduzir modificações no funcionamento e na aparência dos programas. Depois de adquirirem mais confiança, passam a criar seus próprios jogos, utilizando os conhecimentos propiciados pelas etapas anteriores.

A terceira diretriz afirma que a produção deve demandar que novos conceitos sejam explorados pelos alunos e, ao mesmo tempo, solicitar o uso de conceitos explorados anteriormente.

Lee *et al.* (2011) propõem o arcabouço *usar-modificar-criar*, com objetivo que os estudantes tenham novos desafios que os levem a buscar outros conhecimentos. Espera-se manter os alunos em um estado de fluxo nos seus trabalhos na construção de jogos. Assim, o professor atua como um facilitador nos momentos em que os estudantes apresentem dificuldades, como a falta de um conceito específico.

A quarta diretriz sustenta que a mecânica dos jogos deve trazer referências ao universo dos jogos *reais*, que sejam significativos para os alunos. Não basta a construção de um jogo qualquer, é necessário que esteja no cotidiano do aluno, ou se for mais famoso, melhor. Essa estratégia foi utilizada no projeto Code.org, que tiveram base em jogos como *Angry Birds*. Os jogos mais casuais são naturalmente mais fáceis de aplicar essa diretriz, pois será mais aplicável ao ambiente Scratch.

Segundo Barcelos (2020), a oficina tem sido oferecida desde 2013 para diferentes idades e variadas necessidades, com resultados promissores. Assim, encontramos nessa leitura a motivação para nossa pesquisa, e buscamos desenvolver jogos digitais no Scratch para nortear, aprofundar e aplicar o Pensamento Computacional com os jogos desenvolvidos.

Partindo disso, em nossas discussões no grupo Autômato, surgiu a oportunidade de oferecer uma oficina para professores do Núcleo Regional de Apucarana. Após as reuniões com o Núcleo de Educação, decidiu-se que a oficina seria na plataforma Moodle, e a primeira reunião seria presencial. Foram inscritos sessenta professores do referido Núcleo Regional, e contou-se com a colaboração de dezesseis professores tutores para acompanhamentos de quatro cursistas por módulo. O trabalho do professor tutor foi o de cooperar e auxiliar o cursista na realização do módulo e de sanar possíveis dúvidas na construção dos jogos digitais.

Cada módulo era apresentado em uma seção (aba) do ambiente virtual, com os professores que acompanhariam cada cursista. Os cursistas poderiam enviar mensagens nos fóruns ou diretamente para seus professores (em particular) para tirar dúvidas e buscar esclarecimentos sobre a oficina. Os professores tutores que fizeram parte da equipe, são professores da rede estadual de Educação do Paraná e Santa Catarina, também ex-alunos (*alumni*) da UNESPAR (Universidade Estadual do Paraná).

Os objetivos do curso foram assim descritos:

- Capacitar professores quanto aos conceitos e aplicações do Pensamento Computacional;
- Capacitar professores acerca dos aspectos tecnológicos do programa Scratch; e
- Capacitar professores na utilização do Scratch para construção de jogos digitais utilizando o Pensamento Computacional.

As atividades da oficina foram distribuídas ao longo de aproximadamente sete semanas, e cada módulo contou com a construção de um jogo digital no Scratch a partir de um conjunto de vídeos e de um material escrito.

Os vídeos foram realizados com os passos para construção dos jogos no Scratch, junto deles, também houve explicações e material para estudo dos processos do Pensamento Computacional que são mobilizados na construção dos jogos propostos durante a realização do curso. Além disso, havia um conjunto de materiais com imagens e sons para que os estudantes pudessem replicar os jogos produzidos pelo professor durante as aulas em vídeos.

As tarefas dos cursistas eram apresentadas em duas partes, em uma dimensão individual e outra coletiva.

4.1 As partes das tarefas

4.1.1 Parte 1: trabalho individual

Na primeira parte da tarefa, a individual, o enunciado propunha que o cursista construísse o jogo tal como apresentado nos vídeos do módulo em questão. Nesses vídeos, exploravam-se os processos do pensamento computacional aplicados ao planejamento e à realização de um jogo no Scratch. Em seguida, o cursista devia realizar algum acréscimo no jogo, utilizando as noções de pensamento computacional exploradas nos vídeos. Por fim, o cursista precisava postar o jogo construído em um fórum no curso com uma descrição da alteração que realizou em seu constructo.

4.1.2 Parte 2: trabalho coletivo

Na segunda parte da tarefa, cada cursista precisava interagir com, no mínimo, dois colegas de curso, a partir dos jogos que eles apresentaram em suas postagens individuais. Eles eram orientados a formular perguntas, sugerir acréscimos, indicar possíveis incorreções, ou mesmo contribuir para que uma construção inacabada fosse concluída com êxito.

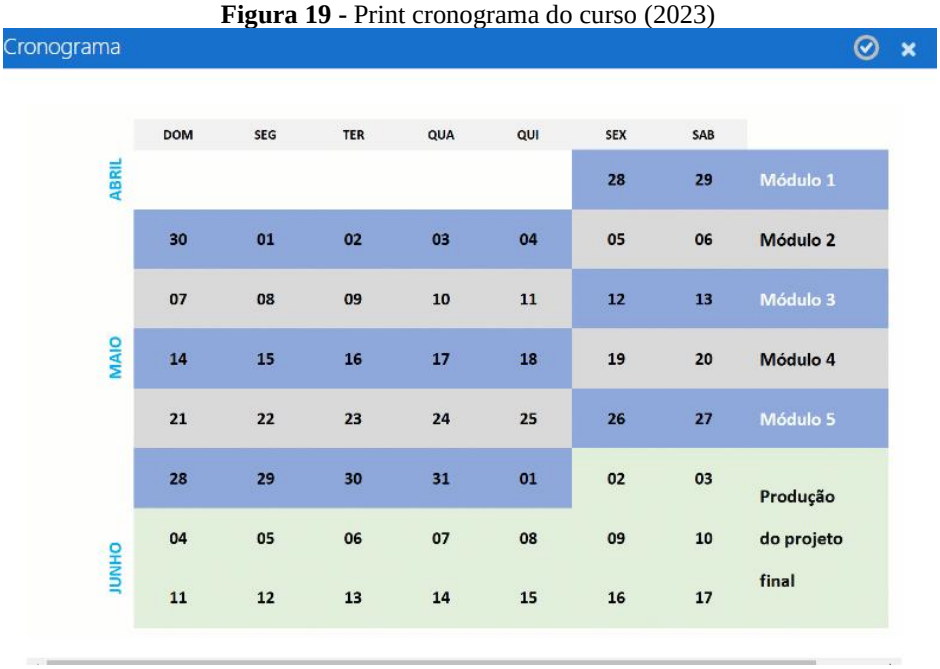
Os professores que acompanhavam cada grupo de cursistas forneciam indicativos para a coordenação atribuir carga-horária a esses participantes de acordo com os seguintes critérios:

- 50% para a postagem do jogo construído no Scratch, acompanhada da descrição;
- 25% para o comentário da postagem de um cursista; e
- 25% para o comentário da postagem de um cursista distinto do primeiro.

Na plataforma Moodle, primeiramente foi feito um vídeo de apresentação do curso, intitulado *Introdução, você precisa saber*, e seus objetivos, assim como os professores que atuariam junto aos cursistas, auxiliando na construção e sanando dúvidas que poderiam surgir.

Foi apresentado aos cursistas, na aula inaugural, o cronograma do curso.

Figura 19 - Print cronograma do curso (2023)



Fonte: produzida pela autora.

Cada módulo do curso era realizado durante uma semana de estudos: da zero hora de sexta-feira até às vinte três horas e cinquenta e nove minutos da quinta-feira da semana seguinte.

Após cinco semanas de estudos, em que os cursistas puderam construir cinco jogos diferentes fazendo uso de diferentes abordagens sobre o Pensamento Computacional e sobre diferentes técnicas de utilização do Scratch, foi dedicado um período de dezesseis dias para que pudessem realizar a construção de um jogo. Essa fase do trabalho foi chamada *produção do projeto final*.

O projeto consistia na construção de um jogo digital no Scratch, diferente daqueles abordados no curso, com os recursos aprendidos durante os cinco módulos, que foram as noções de Pensamento Computacional e recursos/ferramentas do Scratch. Essa produção do cursista deveria ser postada em um fórum, acompanhada de uma descrição do jogo e do emprego das noções de Pensamento Computacional.

4.2 Jogos do Curso e suas relações com o Pensamento Computacional

Conforme já mencionado, em cada módulo, de 1 a 5, foi apresentada a construção de um jogo, os quais foram elaborados pela coordenação do curso de maneira a evidenciar aplicações do Pensamento Computacional na resolução de problemas. Cada um desses jogos é apresentado a seguir, com uma breve descrição de seu funcionamento.

4.2.1 Jogo das Bexigas

No módulo 1 do Curso foi apresentada a construção do Jogo das Bexigas.

Figura 20 - Tela do Jogo das Bexigas em execução

Fonte: produzida pela autora.

Na dinâmica do jogo, 100 bexigas saem da parte inferior da tela e se movimentam para a parte superior. O jogador deve estourar, com o ponteiro do mouse (ou com as mãos), cada uma das bexigas, evitando que elas toquem na parte superior da tela. Porém, o jogador não deve estourar as bexigas vermelhas, pois nesse caso, ele perde pontos, assim como perde pontos quando as bexigas de qualquer cor estouram ao tocar na parte superior da tela. A cada bexiga estourada, que não seja vermelha, o jogador marca um ponto, que é contabilizado na caixa de texto na parte superior esquerda da tela.

Apresentamos, a seguir como foi demonstrada a construção do Jogo das Bexigas, explorando os seis processos do Pensamento Computacional, conforme discutidos no capítulo 2.

4.2.1.1 Formulação do problema

Construir um jogo em que bexigas surjam na parte inferior da tela e se desloquem para a parte superior. O jogador deve utilizar o mouse (ou as mãos) para estourar as bexigas e marcar pontos. O estouro das bexigas também evita que elas toquem na parte superior e o jogador perca pontos. A única bexiga que não deve ser estourada pelo jogador é a de cor vermelha, pois isso causa perda de pontos.

4.2.1.2 Decomposição

O problema descrito no processo anterior pode ser decomposto em partes menores para operacionalizar sua construção. Com isso, é possível concentrar a atenção na resolução de partes específicas do problema, obtendo-as ao fim o jogo em si.

- Bexigas se deslocam da parte inferior da tela para a parte superior de forma aleatória;
- Bexigas estouram ao tocar na parte superior da tela;
- Bexigas estouram no contato com o ponteiro do mouse (ou com imagens das mãos);
- Indicador de pontuação soma um ponto para cada bexiga que é estourada;
- Indicador de pontuação subtrai um ponto para cada bexiga vermelha estourada; e
- O jogo acontece durante a emissão de 100 bexigas.

4.2.1.3 Reconhecimento de padrões

O processo de reconhecimento de padrões permite encontrar similaridades ou padrões entre pequenos problemas decompostos. Por exemplo, não é necessário construir um objeto único para cada comportamento que se deseja das bexigas, mas um único objeto que se clona e que realiza comportamentos diferentes, simulando uma aleatoriedade controlada. Em outras palavras, como todas as bexigas possuem o mesmo padrão de comportamento, elas podem ser apenas um objeto que é clonado e cujos clones realizam movimentos aleatórios. Além desse padrão, há outros:

- bexigas não devem estourar ao terem contato umas com as outras;
- zerar a pontuação a cada início de jogo e somar números positivos e negativos no cálculo da pontuação; e
- como serão emitidas no máximo 100 bexigas, a pontuação máxima é 100 e a mínima é -100.

4.2.1.4 Abstração

Essa etapa consiste na seleção e classificação dos dados, criando maneiras que ajudam a separar apenas os elementos essenciais em determinado problema, descartando detalhes irrelevantes. No jogo das bexigas, a abstração foi pensada da seguinte forma:

- as cores das bexigas são aparências (fantasias) que um objeto matriz troca aleatoriamente no funcionamento do jogo; e
- sorteia-se uma posição aleatória na parte inferior da tela e uma posição aleatória na parte superior da tela e uma bexiga se desloca em velocidade moderada de um ponto a outro. O processo se repete a cada nova bexiga criada.

4.2.1.5 Produção de algoritmos

A produção de algoritmos é o processo em que se utiliza decomposição, reconhecimento de padrões e abstração para produção de modelos abstratos que garantem o funcionamento do jogo. No jogo das bexigas, a produção de algoritmos se manifesta ao:

- construir blocos separados para tratar o comportamento do objeto principal bexiga;
- controlar o comportamento de um clone;
- controlar as ações necessárias para o jogo iniciar; e
- terminar o jogo ao ter realizado um ciclo de ações.

4.2.1.6 Depuração

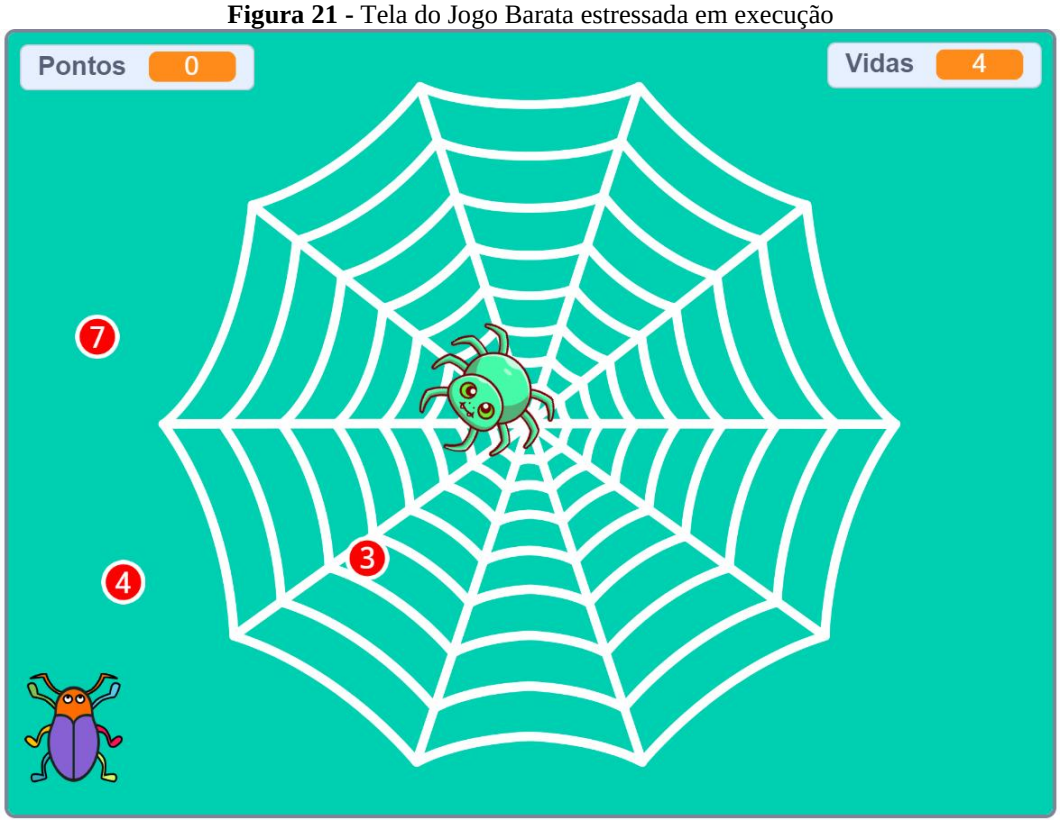
Durante a construção do jogo das bexigas, a depuração acontece a cada trecho de código que é construído e testado. Questionamentos na construção aparecem como contribuição para aprimoramento na produção do jogo digital. Partindo dos questionamentos, pode-se fazer a verificação do jogo e perceber falhas e, através delas, testar novos códigos e formular outros questionamentos e depurar, até que o jogo esteja executável da maneira inicial em que foi proposto na etapa de sua formulação.

4.2.2 Barata estressada

No módulo 2 do Curso foi apresentada a construção do jogo intitulado *Barata estressada*. Trata-se de um jogo de perseguição em que uma barata caminha sobre a teia de uma

aranha, tentando capturar círculos numerados de 1 a 7 que aparecem em pontos aleatórios, ao mesmo tempo que foge do contato com uma aranha que se movimenta por toda a tela.

Figura 21 - Tela do Jogo Barata estressada em execução



Fonte: produzida pela autora.

4.2.2.1 Formulação do problema

Construir um jogo em que uma barata corra pelas teias de uma aranha tentando capturar pontuações e fugir do contato com a aranha. Com as setas do mouse, o jogador deve mover a barata no sentido vertical (para baixo ou para cima) e no sentido horizontal (para direita ou para a esquerda).

A pontuação de 1 a 7 aparece de forma aleatória na tela. Os círculos, ao serem tocados pela barata, marcam-se pontos e desaparecem. Ao serem tocados pela aranha, apenas desaparecem. A barata deve evitar o contato com a aranha, pois a cada contato perde uma de suas cinco vidas.

4.2.2.2 Decomposição

O problema formulado no processo anterior, quando pensado em sua efetivação, pode conduzir à elaboração de algumas perguntas, cujas respostas são problemas menores (decomposição) a serem resolvidos durante a construção do jogo digital:

- Como construir a parte visual do jogo?
- Como a barata se move?
- A barata emite sons? Em quais momentos?
- Quais são os movimentos da aranha?
- Como colocar números que aparecem aleatoriamente na tela?
- Como zerar o contador de pontos no início do jogo e somar a pontuação do número tocado pela barata?
- O que acontece no contato da barata com a aranha?

4.2.2.3 Reconhecimento de padrões

A ação no jogo da Barata estressada consiste na realização de movimentos por dois atores principais: a barata, que é controlada pelas setas para cima, para baixo, para direita e para esquerda do teclado; e a aranha, que realiza movimentos aleatórios e que dão a impressão ao jogador de estar perseguindo a barata.

Assim, esses movimentos podem ser descritos de acordo com padrões:

- O movimento da barata tem a mesma velocidade nas quatro direções e sentidos;
- A aranha caminha pela tela de forma aleatória e retorna ao toque nas bordas;
- A cada contato com a barata, a aranha retorna ao centro e a barata perde uma vida; e
- A pontuação é zerada a cada início do jogo e somam-se números positivos para o cálculo da pontuação.

Há, ainda, um objeto *número* que aparece em intervalos aleatórios de tempo, entre 0 e 1 segundo.

Esses objetos também permanecem um tempo aleatório e, em seguida, desaparecem da tela.

4.2.2.4 Abstração

Nesse processo, percebe-se que:

- os movimentos dos atores são deslocamentos intercalados por trocas de duas fantasias; e
- testes de contato entre os atores controlam pontuações e modificações no jogo.

4.2.2.5 Produção de algoritmos

Construir blocos separados para tratar os comportamentos dos atores barata, aranha e números:

- deslocamentos;
- vidas;
- pontuação; e
- testes de contato.

4.2.2.6 Depuração

O processo de depuração acontece entre uma ação e outra na construção do jogo, nos momentos em que são realizados testes para verificação de seu funcionamento. Nesses casos, compara-se o projeto mental do jogo com sua realização efetiva no Scratch.

4.1.3 Continue a voar

O jogo *Continue a voar* foi apresentado no Módulo 3 do Curso. Na tela principal há um pássaro que voa da esquerda para direita, e é sujeito a um efeito de gravidade, sendo atraído para baixo, e sua altura de voo deve ser controlada pelo teclar da barra de espaço. Além de controlar a altura do voo do pássaro, o jogador deve evitar seu contato com obstáculos que surgem à sua frente, tais como nuvens, árvores e rochas.

Figura 22 - Tela do Jogo Continue a voar em execução

Fonte: produzida pela autora.

4.1.3.1 Formulação do problema

Construir um jogo em que um pássaro deve voar à esquerda da tela, na horizontal. A força da gravidade atrai o pássaro para baixo, e o jogador deve controlar sua altura para que não caia e não atinja montanhas, árvores ou nuvens. Em dois displays são exibidos dados do jogo. No primeiro, a distância percorrida de 10 em 10. No segundo, registra-se o recorde do jogador em partidas sucessivas.

4.1.3.2 Decomposição

O jogo *Continue a voar* corresponde à interação de um objeto principal, o pássaro, controlado pelo jogador; e um objeto que troca fantasias à medida que se desloca da direita para a esquerda. Essas interações determinam a dinâmica do jogo, que em sua decomposição, podem ser respondidas as seguintes questões:

- Como construir a parte visual do jogo?
- Como construir o ator pássaro e quais os seus movimentos?

- Como conectar sons, movimentos e aparências?
- Como verificar o contato do pássaro com elementos do jogo e tratar esses eventos?
- Como contabilizar a distância percorrida?
- Como registrar recordes em partidas sucessivas?

4.1.3.3 Reconhecimento de padrões

Um dos principais padrões da dinâmica do jogo é imaginar um objeto parado, que parece estar em movimento por conta de seu plano de fundo, e deslocar-se em direção oposta a ele:

- o pássaro fica parado em relação ao seu deslocamento horizontal;
- o cenário que é obtido por um ator se desloca da direita para a esquerda, provocando a impressão de voo do pássaro;
- o jogo é encerrado em um único contato do pássaro com elementos da tela; e
- a pontuação é zerada a cada início do jogo e o recorde permanece em partidas sucessivas do mesmo jogador.

4.1.3.4 Abstração

Dada as características da ilusão de movimento do ator principal, a sensação de movimento é reforçada pelo seu movimento vertical controlado pela tecla espaço. A cada toque do jogador nessa tecla, o pássaro é impulsionado uma medida y em direção à parte superior da tela.

- O movimento do pássaro acontece somente na vertical e pelas suas modificações de direções; e
- Testes de contato entre os atores devem ser evitados, pois finalizam o jogo.

4.1.3.5 Produção de algoritmos

Construir blocos separados para tratar os comportamentos dos atores:

- Quando o jogo inicia para produzir movimentos, testar contatos e marcar pontuação;
- Quais as definições iniciais?
- Quais as mudanças de fantasias?
- Como movimentar os obstáculos?

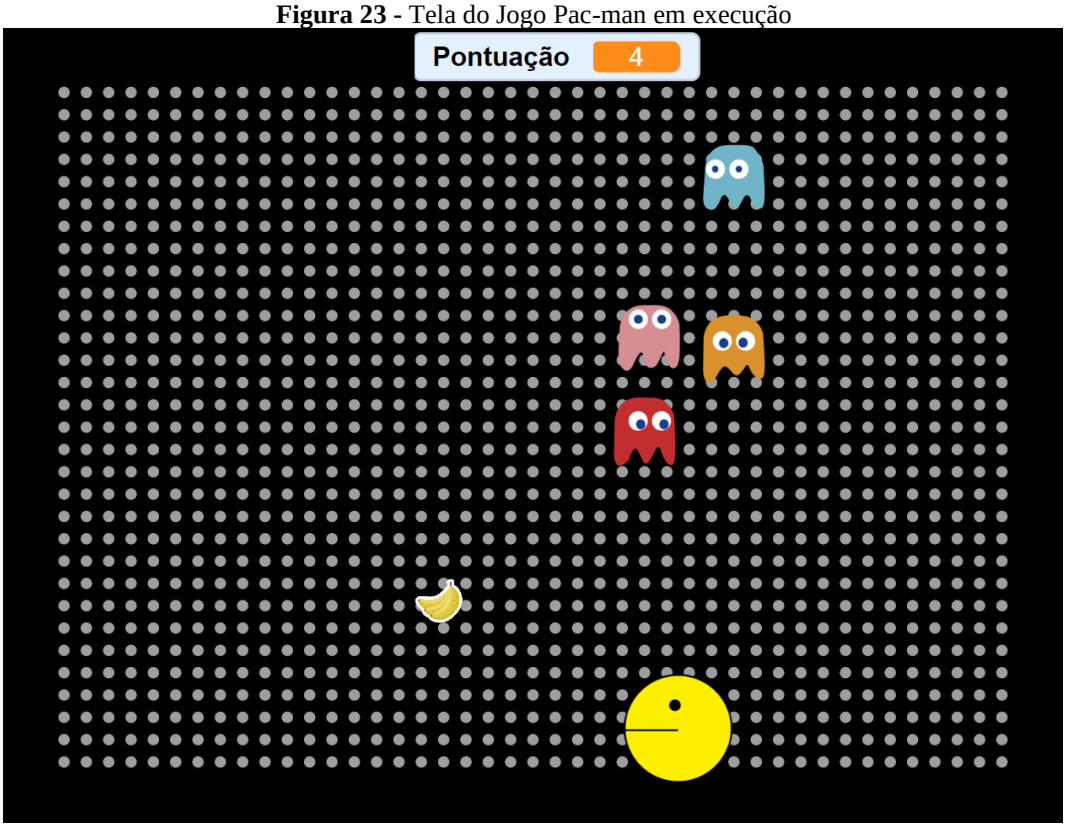
- Como tocar uma música de fundo?

4.1.3.6 Depuração

O processo de depuração acontece entre uma ação e outra na construção do jogo, nos momentos em que são realizados testes para verificar seu funcionamento. Nesses casos, compara-se o projeto mental do jogo com sua realização efetiva no Scratch.

4.1.4 Pac-man

Figura 23 - Tela do Jogo Pac-man em execução



Fonte: produzida pela autora.

O jogo Pac-man, explorado no Módulo 4, consiste em um jogo inspirado no jogo original de mesmo título, criado no Japão em 1980. Trata-se de um personagem amarelo que se movimenta pela tela capturando as frutas que surgem de forma aleatória. Em tempos aleatórios também surgem, nos cantos da tela, fantasmas que se deslocam em zigue-zague. O personagem Pac-man deve evitar o contato com os fantasmas para não perder sua única vida.

4.1.4.1 Formulação do problema

Construir um jogo em que um ator se desloca pela tela na horizontal e na vertical em busca de capturar frutas e fugir da perseguição de quatro fantasmas.

4.1.4.2 Decomposição

A decomposição do jogo Pac-man foi realizada a partir das seguintes questões norteadoras:

- Como construir a parte visual do jogo?
- Como construir os movimentos do ator principal que se desloca ao toque das setas do teclado?
- Como construir o movimento de quatro monstros pela tela?
- Como fazer as frutas aparecerem aleatoriamente na tela?
- Como marcar pontos?
- Como realizar os testes de contato que fazem o jogo finalizar?

4.1.4.3 Reconhecimento de padrões

- Os quatro monstros possuem movimentos semelhantes, mas em sentidos diferentes; e
- O surgimento das frutas é tratado por um simulacro de um evento aleatório.

4.1.4.4 Abstração

- O Pac-man se movimenta na vertical e na horizontal ao toque das setas do teclado; e
- Testes de contato entre os atores devem ser evitados, pois finalizam o jogo.

4.1.4.5 Produção de algoritmos

Construir blocos separados para tratar os comportamentos dos atores Pac-man, monstros e frutas com as:

- Definições iniciais;

- Mudanças de fantasias;
- Pontuação; e
- Testes de contato.

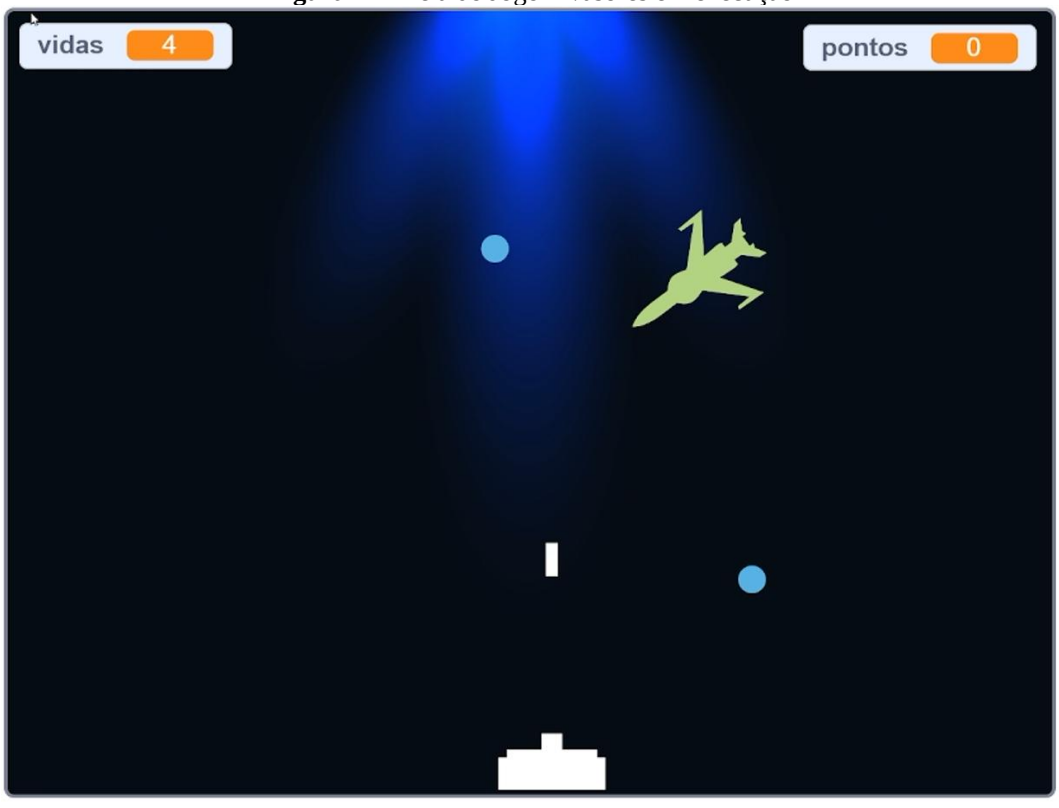
4.1.4.6 Depuração

O processo de depuração acontece entre uma ação e outra na construção do jogo, nos momentos em que são realizados testes para verificação de seu funcionamento. Nesses casos, compara-se o projeto mental do jogo com sua realização efetiva no Scratch.

4.1.5 Invasores

No Módulo 5 foi proposta a construção do jogo intitulado *Invasores*. Esse jogo foi inspirado no jogo *Asteroids* da Atari, de 1979. O jogo é composto por um canhão controlado pelo jogador, aviões e uma nave fixa na parte superior central da tela. O canhão que se desloca na horizontal, na parte inferior da tela, pode disparar tiros contra aviões que cruzam o céu. Além disso, o canhão deve fugir das bombas lançadas pela nave inimiga, que consegue mirar em sua posição atual.

Figura 24 - Tela do Jogo Invasores em execução



Fonte: produzida pela autora.

4.1.5.1 Formulação do problema

Construir um jogo que produza os movimentos de um canhão a partir do uso das teclas direita e esquerda do teclado e esquivar-se dos ataques de bombas, ao mesmo tempo que tenta coletar vidas extras.

4.1.5.2 Decomposição

- Escolher o cenário;
- Inserir os personagens que farão parte e interagir no jogo;
- Programar o movimento dos objetos a serem lançados aleatoriamente;
- Construir o movimento do canhão e o lançamento dos projéteis;
- Construir os comandos de ganho e perda de pontos;
- Construir os movimentos da bomba;
- Construir variáveis para marcar vidas e pontuação;
- Criar uma condição de parada; e
- Testar o funcionamento do jogo.

4.1.5.3 Reconhecimento de padrões

O movimento do canhão possui semelhança, alterando apenas o sentido; o surgimento dos aviões possui programação de comando aleatório, alternando entre personagens amigos e inimigos.

4.5.1.4 Abstração

O canhão deve evitar atingir o personagem inimigo, pois perde pontos, e ao mesmo tempo atingir os aviões e as bombas, porque ele ganhará pontos e vidas. O canhão deve evitar ser atingido pelas bombas, uma vez que elas tiram vidas. O canhão possui movimento para esquerda e direita com as setas do teclado e atira projétil ao se movimentar sem o uso de teclas adicionais.

4.5.1.5 Produção de algoritmos

- Construção de blocos para direcionar o movimento dos personagens;
- criação de variáveis;
- criação de novo bloco deslocamento; e
- inserção de sons para os personagens.

4.5.1.6 Depuração

Durante a construção de cada um dos blocos de comando, testar para validar a utilização de cada bloco, se estão executando o planejado sem afetar nenhum outro bloco.

5 METODOLOGIA

Neste capítulo, apresentamos o percurso metodológico que foi utilizado na seleção e análise dos jogos digitais, objeto desta pesquisa.

Primeiramente, concentramo-nos no objetivo e na área de pesquisa e natureza, determinando-a como pesquisa de cunho qualitativo. A pesquisadora analisou os dados obtidos referentes à oficina online, no ambiente Moodle; interpretou as interações produzidas pelos cursistas/professores de forma a investigar quais processos do Pensamento Computacional, de acordo com grupo de pesquisa Autômato, foram mobilizados em suas construções.

Posteriormente, delimitamos os jogos digitais que seriam utilizados em nossa pesquisa, os procedimentos que seriam empregados na obtenção das informações, e como ocorreria a análise.

Portanto, este capítulo traz, de maneira detalhada, a metodologia empregada neste estudo, providenciando uma visão pormenorizada das estratégias, técnicas e procedimentos que foram utilizados durante a pesquisa.

Baseados nas ideias de Barcelos *et al.* (2014), a oficina de jogos digitais no ambiente Scratch tem por objetivo responder ao seguinte questionamento: quais processos do Pensamento Computacional são mobilizados por um grupo de professores de matemática na construção de jogos no Scratch?

5.1 Delimitação do grupo estudado

No início de nossa pesquisa, traçamos que o nosso estudo seria realizado com estudantes do Ensino Fundamental I, pois trabalho em uma escola municipal com alunos de 2° ano. A partir das orientações, surgiu a oportunidade de oferecer uma oficina para professores do Núcleo de Apucarana. Mudamos os sujeitos da pesquisa por entendermos que no estado do Paraná são ofertadas aulas de Pensamento Computacional. Com isso, acreditamos ser pertinente, diante de uma disciplina nova para professores e estudantes, a pesquisa ser desenvolvida com professores da rede pública paranaense, em específico junto ao Núcleo de Apucarana.

5.2 Procedimentos para obtenção das informações

A seleção dos jogos digitais partiu do princípio de construções que podem ser utilizadas em qualquer faixa etária, também atinjam público tanto de alunos como professores da educação no geral.

No capítulo 4, apresentamos a descrição de cada jogo digital trabalhado no curso. Os vídeos foram produzidos e apresentados aos cursistas em cinco módulos. Cabia ao cursista realizar a construção do jogo de cada módulo, assim como realizar alterações, fazendo com que a cada módulo fosse aperfeiçoado seu aprendizado.

A cada módulo e cada construção, os cursistas interagiam no fórum nas construções dos colegas, referentes ao módulo do jogo apresentado. Poderiam dar sugestões, acréscimos, prints com questionamentos, auxiliar e criar um ambiente provocador de novas ideias nas construções.

As informações para análise do projeto final foram retiradas do fórum do ambiente Moodle, e os cursistas deveriam escrever como pensaram na construção partindo dos processos constituintes do Pensamento Computacional.

Cada módulo tinha um professor da equipe como tutor dos cursistas, auxiliando nas construções, dúvidas, como também analisando os projetos criados por eles, acessando o ambiente diariamente. Cada professor/tutor foi responsável por acompanhar quatro cursistas durante cada módulo da oficina.

Durante o acompanhamento dos cursistas no ambiente Moodle, os professores/tutores deixavam suas impressões sobre as postagens e participações de cada cursista, o que é conhecido como *Sistema de Acompanhamento*. Esse sistema funcionava no decorrer da oficina, e o professor/tutor deveria fazer anotações em uma *ficha de acompanhamento* do cursista, registrando seu histórico de desenvolvimento durante um módulo da oficina. A cada módulo, mudavam os professores/tutores que acompanhavam cada cursista.

5.3 Enfoque da análise

Primeiramente, selecionamos os jogos digitais do Projeto Final que atenderam a proposição do curso. Foram selecionados trinta projetos finais, nos quais foi verificado se aquilo que eles tinham compunha um jogo, sua funcionalidade, e se o cursista registrou no fórum a descrição dos processos do Pensamento Computacional.

Para seleção dos projetos, constituímos uma planilha com critérios pelos quais os projetos foram analisados:

1) Objetivo do jogo;
2) Funcionamento do jogo;
3) Interação entre objetos do jogo e cenários;
4) Elementos constituintes do jogo; e
5) Processos do Pensamento Computacional aplicados na construção do jogo.

Classificamos os projetos finais que atenderam a esses critérios em ordem crescente, e para esta dissertação, foram analisados os três primeiros jogos que tiveram os critérios mais pontuados em nossa planilha. Os títulos dos jogos analisados são os seguintes:

1. *Arco e flecha*
2. *Batalha Estelar*
3. *Tiro ao pato*

A análise que realizamos apresenta-se em seções divididas da seguinte forma:

- **Descrição geral do jogo**

  Apresentar uma descrição geral do jogo envolvendo o cenário, personagens, elementos, sistema de pontuação, vidas, ações do jogador, cores, sonoridade, história, textos, mensagens, entre outros elementos.

- **Objetivo do jogo**
  a) O que o jogador deve fazer? Qual objetivo deve atingir?
  b) Qual é a mecânica do jogo? Como ele se comporta? Como responde às ações do jogador?
  c) Quais desafios e dificuldades foram programadas pelo professor-programador?

- **Elementos visuais e sonoros do jogo**
  a) Quais são os atores que interagem no jogo?
  b) Há algum personagem/avatar comandado pelo jogador?
  c) O cenário do jogo é fixo ou muda?
  d) Há elementos do cenário que possuem funções específicas?
  e) Há sons? Para que servem?
  f) Há músicas? Trilha sonora?
  g) Há vozes? Algum personagem ou elemento fala?

- **Funcionamento do jogo**
  a) Como é a jogabilidade?
  b) Há elementos que não funcionam? Há elementos que funcionam de acordo com estados?

c) É para apenas um jogador?
d) Há vidas?
e) Há contagem de tempo?
f) Há contagem de pontuação? Como isso ocorre?
g) As informações para o jogador são claras? Como elas são oferecidas?

Após a realização dessa análise estritamente técnica, procedemos com uma análise do jogo quanto à aplicação dos processos do Pensamento Computacional.

5.4 O método em ação

Os princípios que utilizamos na revisão bibliográfica e no referencial teórico norteiam as ideias sobre Pensamento Computacional, permitindo um olhar criterioso para o objeto construído pelo cursista. O cursista teve acesso às noções de Pensamento Computacional, a cada módulo do curso, e é legítimo intuir que ele internalizava essas noções na medida em que construía jogos digitais no ambiente Scratch. Dessa forma, acreditamos que o cursista utilizou dessas noções na construção de seu projeto final.

Para analisar esses jogos digitais, primeiramente realizamos a leitura das descrições sobre processos do Pensamento Computacional que foram mobilizados nas construções, que os cursistas escreveram no fórum do ambiente. Após as leituras, buscamos o objetivo do jogo digital construído, os elementos visuais e sonoros, e como a jogabilidade funciona em cada jogo.

Visitando cada projeto no ambiente Scratch, percebemos se o cursista acrescentou uma breve descrição de como funciona o jogo, quais as teclas o jogador deve acionar para que o critério de jogabilidade possa acontecer, e quais seriam as mecânicas do jogo.

Em relação a como o jogo funciona, requer de nossa parte acionar e jogar o projeto várias vezes, vivenciando, relacionando ao que o cursista colocou em sua descrição do jogo, tentando entender o processo de construção.

No interior de cada Projeto Final analisado, buscamos elencar quais foram os comandos utilizados e quais suas funções junto ao projeto. Além disso, procuramos quais percepções se destacaram em relação aos comandos utilizados: atores, cenários, sons, variáveis, e quais contribuições eles trazem para a construção do Projeto Final.

De acordo com cada Projeto Final, foram feitos prints da tela para que a parte visual do jogo pudesse ilustrar este trabalho, trazendo a ideia do cursista para mais próximo do leitor.

Concluindo os apontamentos dos elementos que foram analisados em cada Projeto Final, partimos para a análise dos processos que foram mobilizados nas construções do jogo pelo cursista, de acordo com sua descrição no fórum.

Quanto à formulação do problema, procuramos analisar se o cursista deixou claro como seria seu jogo e quais objetivos ele idealizou com seu Projeto Final.

Na decomposição, buscamos verificar, na descrição do cursista, se ele procurou dividir o problema idealizado em subproblemas, e para isso foi realizando a construção em blocos, fracionando seu problema e fazendo a testagem para verificar sua funcionalidade. Também observamos quais comandos foram definidos para o funcionamento do jogo, como cenários, atores, sons, variáveis, bem como a programação de cada um deles. Procuramos analisamos quais seriam os processos mentais que o cursista pode ter mobilizado em sua construção e comparando com seu projeto, baseados na teoria sobre Pensamento Computacional.

Percebemos, também, que ao fazer a programação do projeto, o cursista pode não ter discorrido sobre mecânicas, e assim concluímos que ele não compreendeu esse pilar na totalidade, ou simplesmente programou, mas não fez a descrição.

No que tange ao reconhecimento de padrões nos jogos construídos, percebemos que os cursistas podem ter feito busca de outros modelos de projetos para fazerem sua construção.

Aparecem nas descrições, também, o que o cursista configura como padrão. Não podemos inferir, em suas mecânicas explicadas, que ele se apropriou dos conceitos desse processo. O cursista poderia ter explicado como o processo de identificar as similaridades e relações que compõem as partes do problema o auxiliou na simplificação dele, ou se foi possível utilizar uma mesma solução em outros subproblemas, se precisasse fazê-lo.

Durante a análise dos jogos, verificamos quais padrões foram utilizados, que objetos existem, suas cores, bem como se o cursista realizou a utilização do padrão em suas previsões.

Na abstração, verificamos se os cursistas realizaram a verificação do que era importante em seus projetos e focaram a atenção na resolução, colocando o projeto em prática. A partir da abstração, procuramos perceber se o cursista tentou simplificar seu projeto, dando ênfase à parte por ele considerada uma informação importante e o resultado que ele desejava alcançar. Percebemos que o cursista, partindo da abstração, em seus processos mentais, tenta interpretar, selecionar, rever, classificar e ordenar os modelos que surgem, partindo para a solução do resultado que ele quer alcançar.

Concluímos que, se o cursista conhece as mecânicas empregadas em seu projeto e comandos que utilizou, ele conseguirá definir qual a informação importante para chegar ao resultado, caracterizando que ocorreu a abstração.

Na análise da produção de algoritmos, identificamos que os cursistas descrevem quais algoritmos foram utilizados, mas entrando em suas construções, percebemos que outros foram utilizados. Como exemplo, um cursista apresentou que utilizou três algoritmos, mas não descreveu outros que aparecem em seus comandos, que por descuido ou falta de conhecimento, não aparecem em sua descrição.

Acreditamos que a depuração perpassa cada processo do Pensamento Computacional, pois para dar jogabilidade ao Projeto Final, os cursistas realizaram testes e possíveis ajustes para que funcionasse de acordo com sua ideia inicial, partindo da formulação de seu problema e colocando em prática.

6 ANÁLISE DOS JOGOS

No texto que segue são apresentadas as análises de três jogos construídos 60 (sessenta) por professores de Matemática participantes do Curso de Pensamento Computacional na Construção de Jogos com o Scratch.

6.1 Análise do jogo Arco e flecha

Esta subseção tem por objetivo efetuar uma análise do jogo desenvolvido pelo cursista José, o qual foi intitulado por ele como *Arco e Flecha*. Realizamos uma descrição geral do projeto desenvolvido pelo cursista, apresentando o objetivo do jogo, seus principais elementos visuais e sonoros, além do quesito jogabilidade.

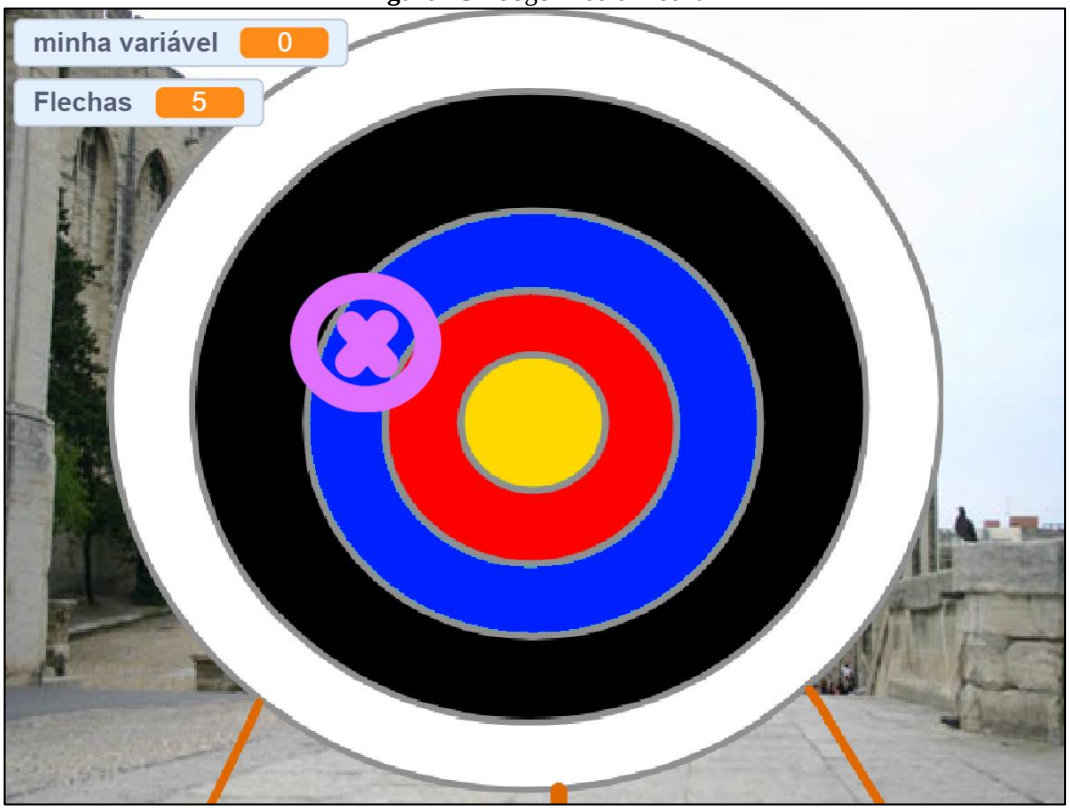
Na sequência, com base em sua postagem no fórum do curso, analisamos como o cursista mobilizou os processos constituintes do Pensamento Computacional no desenvolvimento de seu jogo.

Iniciamos a análise afirmando que o cursista, na página de seu projeto no Scratch[5], não realizou uma descrição de seu jogo, seja no campo das instruções ou no campo das notas e créditos. Sendo assim, ao lidar com o jogo, um jogador pode, inicialmente, não saber o que deve fazer, compreender o objetivo a ser atingido ou quais são as mecânicas do jogo.

O jogo possui dois atores, que serão chamados de *objetos* desse ponto do texto em diante. O primeiro foi denominado *flecha,* e trata de uma das mecânicas do jogo, a mira. O segundo objeto, por sua vez, refere-se à tela de *menu*, que aparece para o jogador antes do jogo ser iniciado. Além desses elementos, o jogo possui o cenário de um alvo com várias cores, conforme apresentado na Figura 25.

[5] O jogo pode ser acessado em: https://scratch.mit.edu/projects/873256685.

Figura 25 – Jogo *Arco e Flecha*



Fonte: Jogo *Arco e Flecha*.

Como é possível observar na Figura 25, o jogador possui, inicialmente, cinco flechas. Ao começar o jogo, o objeto da flecha, que corresponde à mira de cor rosa presente na figura, desliza aleatoriamente pelo cenário. O jogador, então, deve utilizar a tecla espaço do teclado para disparar a flecha, travando a mira em algum local sobre o alvo.

O jogador pontua de acordo com a posição em que a flecha é acertada, de maneira que, quanto mais próximo do centro, maior será a pontuação. Ao acertar a região em branco, o jogador marca 5 pontos; na região em preto, 10 pontos; na região em azul, 25 pontos; na região vermelha, 75 pontos; e por fim, caso o jogador acerte a flecha na região amarela, são contabilizados 150 pontos.

Os pontos são contabilizados e armazenados na variável *minha variável*, e são acumulados conforme o jogador acertar as flechas no alvo. Acredita-se, no entanto, que o cursista tenha esquecido de renomear essa variável (seja para pontuação, pontos etc.) a fim de explicitar para o jogador que sua função é contabilizar a pontuação.

Em síntese, o objetivo do jogo é que o jogador pontue o máximo possível, acertando a maior quantidade de flechas no centro ou próximo da região central, tendo um total de cinco flechas, o que corresponde a cinco chances. Após os cinco disparos, o jogo é finalizado e o jogador pode reiniciá-lo a fim de tentar atingir ou superar o seu recorde.

É possível ouvir um único som durante o jogo, que é disparado quando o jogador aperta o botão de *iniciar*, presente no menu inicial.

Realizados esses apontamentos a respeito dos elementos que constituem o jogo e discutimos, na sequência, sobre como os processos do Pensamento Computacional foram mobilizados pelo cursista. Evidenciamos, contudo, que o cursista foi orientado, após a conclusão de seu projeto, a redigir um texto explicitando como ele identificou os processos do Pensamento Computacional durante e após a construção do jogo.

No que tange ao primeiro processo do Pensamento Computacional, a **formulação do problema,** o cursista apresentou o seguinte texto em sua postagem no fórum:

> *Decidi criar um jogo de arco, precisei pensar em como abordar esse desafio. O objetivo é desenvolver uma experiência interativa e divertida, onde os jogadores possam testar suas habilidades de mira e pontaria enquanto se divertem.*

A partir dessa descrição, é possível inferir que o cursista teve, desde o início do desenvolvimento do jogo, uma ideia clara sobre o problema que deveria ser resolvido, que era criar um jogo em que o jogador controlasse um arco com o objetivo de acertar um alvo. Entretanto, acreditamos que o cursista, ao colocar em prática o que havia pensado, deparou-se com entraves em relação ao conceito computacional de sequências, visto que

> Projetar um projeto não é um processo limpo e sequencial de primeiro identificar um conceito para um projeto, depois desenvolver um plano para o design e, então, implementar o design em código. É um processo adaptativo, no qual o plano pode mudar em resposta à abordagem de uma solução em pequenos passos (Brennan; Resnick, 2012, p. 12).

Dessa forma, processos do Pensamento Computacional, como a formulação do problema, produção de algoritmos e depuração, ocorreram de forma concomitante com os conceitos computacionais de sequências, *loops* e paralelismo, conforme apontamos mais adiante.

Considerando que o jogador deveria ter uma experiência interativa e divertida, a **formulação do problema** pode ter auxiliado o cursista antes do início do desenvolvimento de seu jogo, por meio de questões norteadoras e estabelecimento de objetivos concretos que deveriam ser atingidos ao final da construção.

Partindo da formulação do problema, percebe-se o conceito computacional de produção de algoritmos, pois através dele o cursista inicia a criação de seu jogo digital.

Percebe-se que o cursista, no que tange aos conceitos computacionais, ao formular o que iria construir e partindo de sua idealização, teve uma sequência de passos para que chegasse

ao resultado almejado, uma série de etapas que foi pensada antes de sua construção, que se assemelha à produção de algoritmos.

Na Figura 26, apresentamos um trecho do código do jogo, no qual evidenciamos a ação a ser produzida pela sequência de instruções: o objeto *flecha* é posicionado na tela, tendo um tamanho pré-definido e movimentando-se pela tela por um espaço também estabelecido pelas coordenadas aleatórias de x e y.

Figura 26 – Trecho do código do jogo *Arco e Flecha*

Fonte: Jogo *Arco e Flecha*.

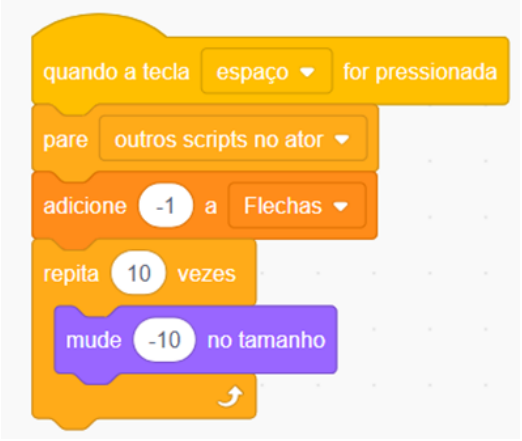
Um conceito computacional essencial para a mídia interativa é o de eventos, ou seja, quando há um gatilho para que uma sequência seja executada. Na imagem abaixo, o cursista organizou os blocos de modo que, *quando a bandeira verde for clicada*, seja exibida ao jogador uma tela inicial, *escondendo* o cenário e o objeto flecha, além de limpar o contador de pontos, que permaneceu nomeado como *minha variável*, e restabelecendo o número de flechas.

Figura 27 – Trecho do código do jogo *Arco e Flecha*

Fonte: Jogo *Arco e Flecha*.

Outro exemplo de eventos no trecho de código é apresentado abaixo, em que a tecla espaço, quando pressionada, inicia a sequência de fixar a flecha no alvo, diminuindo em um o número de flechas a serem lançadas, também diminuindo seu tamanho.

Figura 28 – Trecho do código do jogo *Arco e Flecha*



Fonte: Jogo *Arco e Flecha*.

A fim de atingir os objetivos estabelecidos inicialmente, de construir o jogo *Arco e Flecha*, o cursista expõe que o processo de **decomposição** desse problema em subproblemas menores ocorreu da seguinte maneira:

> *Para construir o jogo de arco e flecha, foi necessário dividir o processo em etapas menores. Primeiro, criei o cenário e defini o fundo do jogo [...] Em seguida, projetei um personagem para simular o lançamento da flecha. Depois [o] adicionei alvo. Por fim considerei o sistema de pontuação e a lógica para determinar se a flecha atingiu o alvo.*

Nota-se que o cursista, no início de sua descrição sobre o processo de decomposição, apresenta a definição desse processo utilizando suas próprias palavras. Ele evidencia que, para atingir seu objetivo, construir o jogo, foi necessário dividir esse problema em etapas menores.

Segundo o cursista, o trabalho de criação de seu jogo foi dividido em quatro subproblemas menores, a saber:

1) *Definir a imagem que iria compor o cenário de seu jogo* - o cursista utilizou-se de uma imagem da biblioteca do próprio Scratch para compor seu cenário.
2) *Projetar o ator que iria simular o lançamento da flecha* - a ideia da flecha idealizada pelo cursista, no Scratch, foi concretizada por meio da criação de um objeto no formato de uma mira com a cor rosa, conforme ilustrado anteriormente, na Figura 26.
3) *Desenhar o alvo* - o alvo foi desenhado sobre a paisagem inicial e, sendo assim, o cursista deve ter utilizado o próprio Scratch com as ferramentas apropriadas para desenhá-lo. Cabe salientar que o cursista poderia ter criado um ator para desenhar esse

objeto, deixando-o independente do cenário. Isso poderia ser útil caso o cursista criasse alguma mecânica em que o alvo precisasse se movimentar de alguma maneira pelo cenário. No entanto, pode-se inferir que o cursista, visto que em suas ideias iniciais não vislumbravam mecânicas referentes ao movimento do alvo, optou por construí-lo anexado ao cenário.

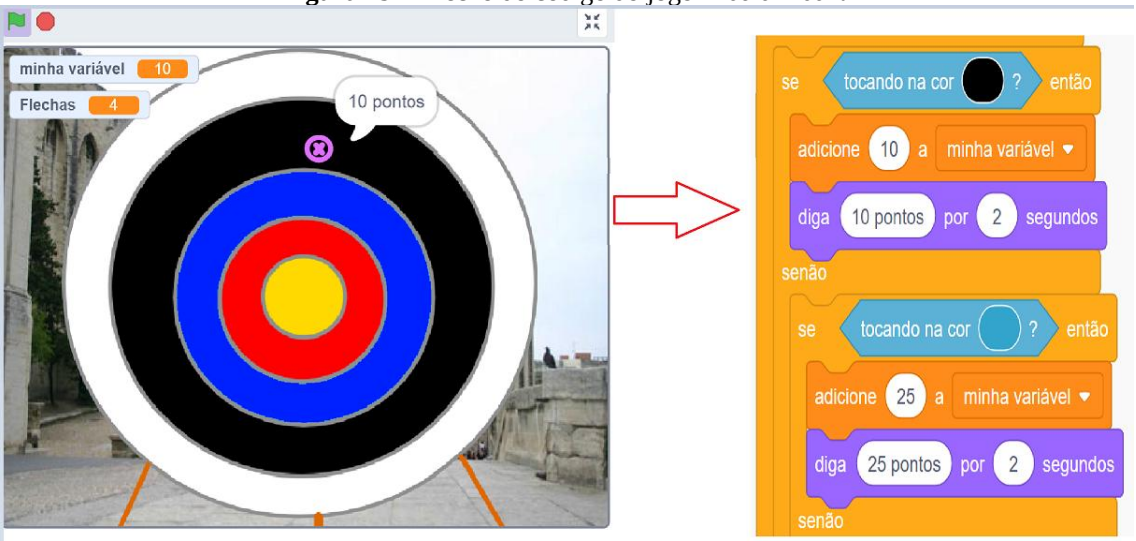
4) *Criar um sistema de pontuação e verificação de acertos* - esse pode ter sido o subproblema que mais exigiu do cursista, uma vez que foi necessário programar os blocos do Scratch a fim de atingir o objetivo esperado. O cursista criou uma variável para contabilizar a quantidade inicial de flechas e, para verificar em qual lugar o jogador acertou, utilizou-se dos blocos de sensores, verificando se a mira estava tocando em uma determinada cor no alvo.

Apesar de possuir um menu antes de iniciar e a mira se movimentar aleatoriamente pelo cenário durante o jogo, essas mecânicas não foram identificadas pelo cursista ao mencionar o processo de decomposição.

Certamente o cursista teve que tratar desses subproblemas no desenvolvimento de seu jogo, podendo ter sido considerado inicialmente. No entanto, ao descrever seu projeto, o cursista não discorreu sobre essas mecânicas. Sendo assim, em nosso entendimento, é possível afirmar que o cursista compreendeu parcialmente o que consiste a decomposição do problema e como ela pode ser útil na resolução do problema original, nesse caso, construir o jogo.

Ao analisar a maneira como o cursista descreveu o processo de decomposição e examinar sua construção no Scratch, percebemos, em seu projeto, blocos de programação semelhantes e relacionados ao conceito computacional de condicionais, pois de acordo com a capacidade de tomar decisões e com base em certas condições é que teremos suporte à expressão de múltiplos resultados, visto que as funções desses blocos diferem de acordo com a cor em que o objeto flecha toca no alvo. Em outras palavras, conforme a cor tocada pela *flecha*, será adicionado à *minha variável* o valor sugerido pelo cursista, transmitindo-lhe uma mensagem com a pontuação alcançada nesse disparo. Tais condições são expressas pelos blocos *se* e *senão* para escolher entre as alternativas. A Figura 29 ilustra o uso de condicionais e demonstra a dinâmica estabelecida pelo cursista para efetivar o sistema de pontuação planejado.

Figura 29 – Trecho do código do jogo *Arco e Flecha*



Fonte: Jogo *Arco e Flecha*.

Em relação ao **reconhecimento de padrões**, o cursista mencionou que:

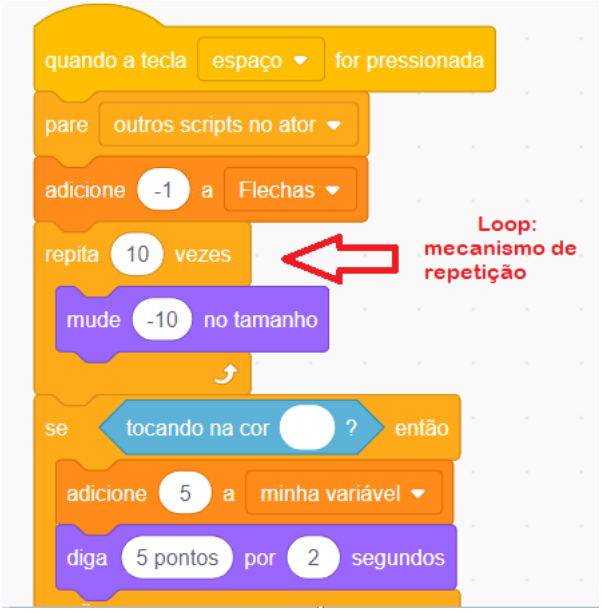
> *Ao criar um jogo de arco e flecha, precisei observar jogos semelhantes existentes e analisar os padrões comuns. Isso inclui a mecânica de mira e pontaria, a trajetória da flecha, o cálculo da distância e da velocidade do lançamento.*

A partir desse texto do cursista, é possível inferir que ele buscou outros jogos semelhantes antes e/ou durante o trabalho de desenvolvimento de seu jogo. Ele afirma, ainda, que esse processo foi realizado com a finalidade de reconhecer possíveis padrões nesses tipos de jogos.

No entanto, apesar de o cursista afirmar que reconheceu alguns padrões durante o desenvolvimento de seu jogo, como a mecânica de mira e pontaria, a trajetória da flecha, o cálculo da distância e da velocidade do lançamento, não é possível inferir em que medida as mecânicas explicitadas configuram-se como um padrão.

Percebe-se que o *loop* está presente como conceito computacional, pois quando o cursista verifica que existe um padrão e reutiliza o mesmo comando/bloco, está se apropriando de repetições (*loops*), executando a mesma sequência várias vezes. Na figura da página seguinte, percebe-se a utilização do bloco *repita 10 vezes*, demonstrando que, em seu pensamento, o cursista percebeu que poderia diminuir os blocos utilizados colocando o mecanismo de repetição.

Figura 30 – Trecho do código do jogo *Arco e Flecha*



Fonte: Jogo *Arco e Flecha*.

Seria apropriado se o cursista detalhasse como cada uma das mecânicas citadas poderiam ser consideradas padrões. Por exemplo, o cursista poderia citar que o algoritmo que contabiliza a pontuação segue um padrão. Independentemente do local que o jogador acerte o alvo, a ideia por trás do algoritmo que contabiliza a pontuação é a mesma: caso a mira encoste em uma determinada cor, será adicionado um determinado valor à variável da pontuação.

O cursista poderia, portanto, explicitar como a identificação das similaridades e relações que compõem as partes do problema lhe auxiliou na simplificação dele, ou ainda, se foi possível replicar uma determinada solução em outros subproblemas, caso fosse necessário.

No que tange ao exercício da **abstração**, o cursista afirma que:

> *Para simplificar o desenvolvimento do jogo, eu me concentrei em suas partes essenciais. Isso inclui a criação de um personagem jogável e a definição de uma pontuação com base na proximidade do acerto do alvo. Abstrai esses elementos principais e me permitiu criar um jogo funcional e divertido, sem me perder em detalhes.*

O cursista iniciou apresentando a definição de abstração, que consiste em se concentrar nas partes essenciais de um determinado objeto, com a finalidade de simplificar o problema. Na sequência, o cursista argumentou que, em seu projeto, o exercício da abstração ocorreu no momento da criação do ator da mira e da definição do sistema de pontuação, com base no acerto do jogador.

De fato, é possível inferir que a ideia original da flecha passou por um processo de abstração, uma vez que, no jogo, não existe nenhum arco ou flecha, mas uma versão

simplificada dessas ideias, levando em consideração aspectos essenciais em sua criação, como o movimento e o formato da mira.

A implementação do sistema de pontuação, por sua vez, também passou pelo processo de abstração. Durante a construção da mecânica que contabiliza os pontos, o cursista levou em consideração apenas a propriedade das cores do alvo, sem se atentar à forma, posição ou quaisquer outras características desse objeto. Apenas o atributo das cores do alvo foi suficiente para construir essa mecânica com êxito.

Desse modo, concluímos que o cursista, durante o desenvolvimento de seu projeto, utilizou-se da abstração com a finalidade de colocar suas ideias em prática. Sinalizamos, no entanto, que podem ter ocorrido outros processos de abstração na criação de seu jogo que não foram explicitados em seu comentário.

Em relação à **produção de algoritmos**, o cursista escreveu:

> *Agora é a hora de escrever os algoritmos que darão vida ao jogo. Vou começar definindo os comandos para mover o personagem e criar a lógica para o lançamento da flecha com base na direção selecionada pelo jogador. Em seguida, implementei bem a pontuação com base onde o tiro acertou.*

Nota-se que o cursista menciona três algoritmos:

Movimento da flecha - para a movimentação da flecha, que deve acontecer 5 vezes (relembramos que o jogador tem cinco chances para acertar o alvo), o cursista utilizou um algoritmo que define a posição e o tamanho inicial do ator, além do bloco *deslize*, permitindo que o ator se movimente aleatoriamente pelo cenário. O algoritmo é exibido na sequência, na Figura 31.

Figura 31 – Algoritmo que movimenta a flecha no jogo *Arco e Flecha*

Fonte: Jogo *Arco e Flecha*.

Salientamos que, apesar do movimento ser aleatório, o cursista não utilizou o bloco *deslize por [x] segs. até posição aleatória*. Acreditamos que isso se deve ao fato de que, apesar

de a flecha se movimentar aleatoriamente pelo cenário, o movimento deve ficar restrito à área sobre o alvo.

Lançamento da flecha - o algoritmo responsável pelo *lançamento* da flecha foi construído junto ao que contabiliza a pontuação. No entanto, eles foram analisados separadamente, pois o cursista realizou essa diferenciação. Uma vez que a flecha se movimenta indefinidamente, para o *lançamento* da flecha, o cursista criou um algoritmo que aguarda até que o jogador pressione a tecla espaço do teclado. Quando essa ação é executada, o algoritmo para os outros *scripts* no ator, consequentemente parando sua movimentação. Então, a quantidade de flechas diminui em uma unidade e uma pequena animação do ator diminuindo é realizada. A Figura 30 ilustra a construção dessas funções.

Os operadores são conceitos computacionais que fornecem suporte para expressões matemáticas, lógicas e de *string*, permitindo que o programador execute manipulações numéricas e de *string*. Esses operadores estão explícitos na construção de *arco e flecha*, inclusive já exposto na Figura 31, que trata da movimentação da flecha. Portanto, podemos discorrer que esse conceito computacional está aliado aos seguintes processos do pensamento computacional: **formulação do problema**- exibir a flecha no ponto determinado, fazendo-a movimentar-se no espaço delimitado; **decomposição de problemas**- essa é apenas uma das mecânicas do jogo; **abstração** - preocupar-se, nesse momento, com a movimentação da flecha. O cursista elaborou seu projeto de modo que, ao ser pressionada a tecla espaço, o ator flecha fosse parado, dando a impressão de ser fixado no alvo, ao mesmo tempo em que o contador de flechas era diminuído, pois uma flecha foi lançada.

Na figura 32, observa-se a utilização, pelo cursista, de dados e operadores, demonstrando um conceito computacional responsável pelo movimento aleatório da flecha pelo cenário. Percebe-se, também, na programação do jogo, a repetição dos blocos, assim definindo o padrão e a utilização do comando *sempre*.

Figura 32 – Algoritmo que "lança" a flecha no jogo *Arco e Flecha*

Fonte: Jogo *Arco e Flecha*.

Pontuação - o algoritmo que contabiliza a pontuação, cuja exibição parcial é realizada na Figura 27, também salientado nos pilares *reconhecimento de padrões* e *abstração*. O cursista utiliza blocos de sensores para verificar se o ator flecha está encostando em uma cor específica do alvo e, dependendo da cor, uma pontuação diferente é contabilizada na variável correspondente.

Apesar de o cursista fazer menção apenas a esses três algoritmos, outros também estão presentes nos blocos de comandos. Por exemplo, para a execução do menu, o cursista construiu um algoritmo que inicialmente faz com que ele apareça e, quando o jogador clica no botão para iniciar, esse ator é ocultado e o jogo começa. Outro algoritmo que também foi construído pelo cursista é o que reseta a pontuação e a quantidade de flechas que o jogador deve possuir no início do jogo.

Muito embora o cursista tenha se preocupado em organizar suas ideias em algoritmos distintos, cabe salientar que seria mais adequado que os algoritmos do *lançamento* das flechas e o que contabiliza a pontuação fossem alocados em lugares distintos, já que possuem finalidades bastante marcantes.

Por fim, no que tange ao processo de **depuração**, o cursista mencionou que:

> *Após a criação do jogo de arco e flecha no Scratch, é importante testar e depurar todas as funcionalidades, verifiquei se o personagem se move corretamente, o lançamento da flecha está funcionando adequadamente. Além disso, verifiquei se a pontuação é calculada corretamente.*

O cursista afirmou que o processo de depuração ocorreu após a finalização do jogo, verificando se funcionalidades desejadas foram implementadas com êxito, como por exemplo, o lançamento da flecha ou o cálculo da pontuação. Contudo, o cursista não especificou que tipos de exercícios foram realizados durante o processo de depuração para verificar se todas as funcionalidades estavam operando adequadamente.

Levantamos uma hipótese de que o processo de depuração possa ter ocorrido em outros momentos, principalmente durante o desenvolvimento do jogo. Apesar de o cursista não afirmar que a depuração foi um exercício constante, podem ter ocorrido erros, ou ainda algoritmos que não funcionaram da maneira esperada, obrigando o cursista a reformular suas ideias, refinando as resoluções e algoritmos iniciais.

Realizada essa análise sobre o jogo *Arco e Flecha*, é possível concluir que o cursista pôde mobilizar processos do Pensamento Computacional durante e após o desenvolvimento de seu projeto.

Em alguns processos, como o reconhecimento de padrões e a depuração, o cursista não explicita como auxiliaram na construção do jogo. Cabe salientar, porém, que isso não significa que ele não mobilizou esses processos mentais durante o desenvolvimento do jogo.

Em contrapartida, em outros casos, o cursista preocupou-se em evidenciar, para os leitores, como esses processos foram mobilizados antes, durante e após a construção do jogo. Desse modo, pode-se inferir que os processos do Pensamento Computacional sistematizaram parte significativa do desenvolvimento de seu jogo, podendo ter auxiliado o cursista nessa construção.

No projeto do cursista, percebem-se sequências de instruções acontecendo ao mesmo tempo. No momento da depuração, por vezes devemos analisar se o que está sendo executado/processado está de acordo com o que planejamos, de acordo com a formulação do problema. Do jogo *arco e flecha*, o cursista elaborou seu projeto de modo que, ao ser pressionada a tecla espaço, o ator flecha fosse parado, dando a impressão de ser fixado no alvo, ao mesmo tempo em que o contador flechas era diminuído, pois uma flecha foi lançada. Isso demonstra que ele utilizou o conceito computacional Paralelismo (figura 31).

6.2 Análise do jogo Batalha Estelar

O jogo construído pelo cursista Gieverson, intitulado por ele como Batalha Estelar, é o objeto de nossa análise nesta subseção do texto. Realizamos uma descrição geral do projeto desenvolvido pelo cursista, e seguimos os mesmos critérios de análise do jogo digital Arco e flecha.

Na sequência, com base em sua postagem no fórum do curso, analisamos como o cursista mobilizou processos do Pensamento Computacional no desenvolvimento de seu jogo.

Iniciamos a análise afirmando que o cursista, na página de seu projeto no Scratch[6], realizou uma pequena descrição de seu jogo no campo das instruções. O cursista convidou e incentivou o acesso ao seu jogo.

> *Olá, caro jogador, se prepare para o jogo do século, onde você conduzirá uma nave e terá que atingir seus inimigos, atirando a tecla espaço e se movimenta com as teclas A, W, S, D. Desvie das naves e das bombas que vêm em sua direção. Boa sorte!*

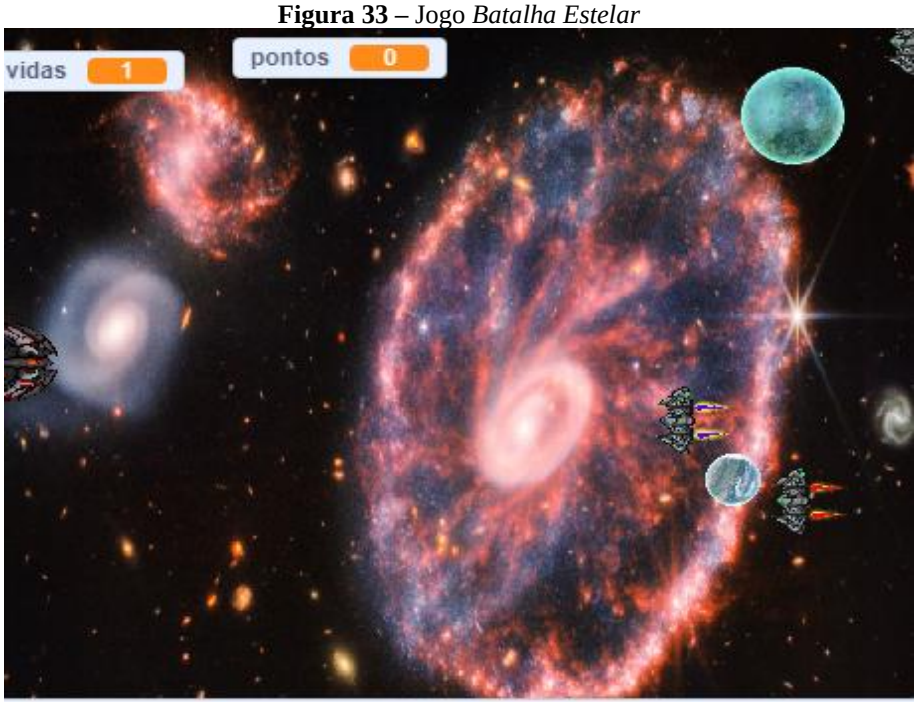
[6] O jogo pode ser acessado em https://scratch.mit.edu/projects/872492441.

Analisando a descrição do jogo, percebemos que não houve detalhes sobre a jogabilidade. O cursista apenas mencionou que o objetivo era atirar, conduzindo uma nave e movimentando-se com a utilização das seguintes teclas: A, W, S, D. Por último, deseja-se boa sorte a quem está disputando uma partida no jogo.

O jogo possui nove objetos. O primeiro foi denominado *nave-jogador*, e trata-se da nave principal do jogador, pois ela que defenderá o espaço aéreo e tentará eliminar as naves inimigas. O segundo e o terceiro objetos são planetas que aparecem no cenário e se movem conforme a nave vai se locomovendo em ataque. O quarto e o oitavo objetos são naves inimigas, programadas para atacar a *nave-jogador*. O quinto objeto corresponde a uma mensagem de fim de jogo, aparecendo para o jogador a palavra *game over*. O sexto objeto é um projétil que sai da *nave-jogador* para que ataque com o objetivo de atingir as naves inimigas e eliminá-las. O sétimo objeto é um cenário que muda durante o jogo. O nono objeto é uma *explosão*, que aparece quando a *nave-jogador* ou as naves inimigas são atingidas.

Além desses elementos, o jogo possui três cenários que marcam o início do jogo, avisando *se prepare para Batalha Estelar*, que entendemos como a abertura do jogo. O segundo cenário foi denominado *Terra*, local que aparece a *nave-jogador* no espaço em volta do planeta Terra, com um X em vermelho, também uma nave maior aparecendo ao lado direito. O terceiro cenário mostra uma galáxia, representando o palco principal do jogo, conforme apresentado na Figura 33.

Figura 33 – Jogo *Batalha Estelar*



Fonte: Jogo Batalha Estelar.

Como é possível observar na Figura 33, o jogador possui, inicialmente, uma nave. Ao começar o jogo, o ator *nave-jogador*, que corresponde ao ator principal, deve deslizar pelo cenário, tentando defender seu espaço aéreo. O jogador, então, deve utilizar as teclas espaço do teclado para disparar tiros na tentativa de atingir e derrubar as naves inimigas, intituladas de *aliens*1, 2, 3, 4 e 5.

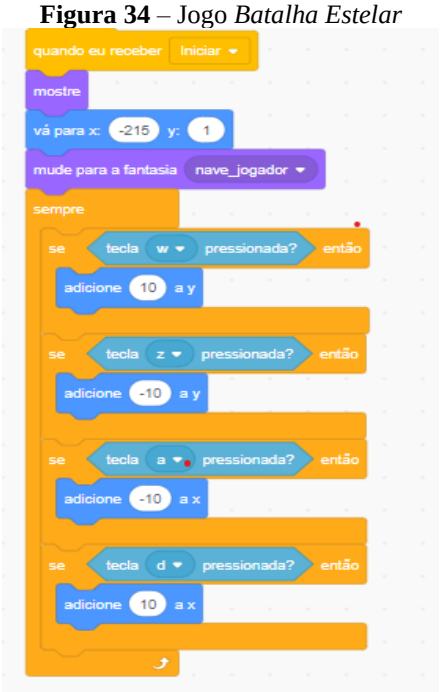
Para mover o objeto principal, *nave-jogador*, o jogador deve utilizar as teclas *A W, S e D*, movimentando a nave principal para fugir dos *aliens.* Verificando as teclas de letras, a letra *A* move a *nave-jogador* para trás, a tecla *W* move a *nave-jogador* para cima do cenário, a tecla *D* movimenta a *nave-jogador* para frente, e não foi verificada a funcionalidade da tecla *S*. Uma hipótese para essa tecla seria movimentar a *nave-jogador* para a parte inferior do cenário.

Os pontos são contabilizados pela variável *minha variável*, e são acumulados conforme o jogador acertar os tiros nas naves inimigas.

Cada nave inimiga abatida por meio do tiro da tecla espaço acrescenta dois pontos à variável *pontos*, e caso essa quantidade chegue a vinte pontos, a *nave-jogador* é contemplada com uma vida extra.

Acredita-se, no entanto, que o cursista tenha digitado erroneamente a tecla *S*, pois a funcionalidade está para a tecla *Z*, como citado anteriormente, e verificando a figura 34, na próxima página, a hipótese se confirma.

Figura 34 – Jogo *Batalha Estelar*



Fonte: Jogo Batalha Estelar[7].

[7] Disponível em https://scratch.mit.edu/projects/872492441. Acesso em 26 nov. 2023.

Executando o jogo, identificam-se três sons diferentes: o primeiro é executado quando o jogador clica o botão de iniciar presente no menu inicial; o segundo é acionado quando a *nave-jogador* é atingida por um *alien;* e o terceiro sinaliza que o jogo acabou, conjuntamente com a tela *game over*.

No que tange à **formulação do problema,** um dos processos do Pensamento Computacional, o cursista realizou o seguinte comentário em sua postagem no fórum:

> *Problema: Desenvolver um jogo que simule uma batalha no espaço, com naves que se movimentam para direita e esquerda, tendo que atingir a nave do jogador, o qual tem que desviar das naves e das bombas.*

Percebe-se que o cursista teve uma ideia inicial e colocou-a como desafio de criar seu jogo digital no Scratch.

Quanto à **decomposição**, partindo dos objetivos que deseja atingir na construção de seu jogo digital, destacamos a postagem no fórum:

> *Determinar os elementos que fazem parte do jogo, através de atores que simulem essa batalha, como figuras que representam as naves inimigas, projéteis para fazer compor o jogo e programá-los a partir de blocos da plataforma Scratch.*

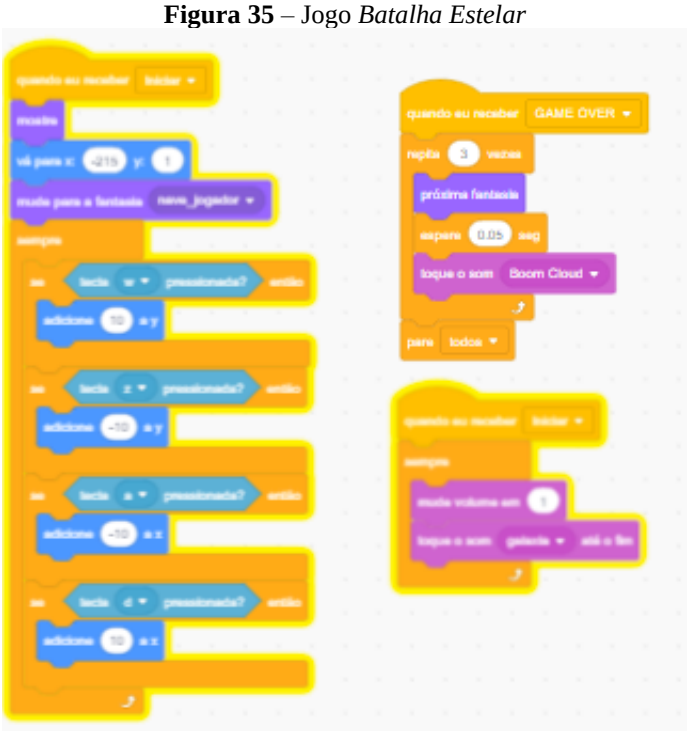
De acordo com a postagem do cursista, o processo de criação de seu jogo foi dividido em quatro subproblemas menores:

1) *Determinar os elementos que fazem parte do jogo* - o cursista imaginou os elementos que poderiam fazer parte de seu jogo, idealizou quais seriam, mas não deu detalhes de quais utilizaria;
2) *Objetos que simulem essa batalha* - o cursista utilizou imagens para caracterizar o ator principal, a *nave-jogador*. Programou a nave principal para atirar e usou um segundo ator como *tiros*, programando a tecla espaço como disparadora de tiros a cada vez que é pressionada;
3) *Como figuras que representam as naves inimigas* - percebemos que o cursista já deixou claras as imagens que utilizaria. Pesquisou algumas na internet, mas não relatou na sua postagem claramente quais seriam, mas na construção aparecem como *aliens* 1, 2, 3, 4, e 5, programou como naves inimigas; e
4) *Projéteis para fazer compor o jogo e programá-los a partir de blocos da plataforma Scratch* - percebemos, na construção, que o cursista pesquisou na internet imagens de projéteis para simular a explosão quando as naves são atingidas, nomeando como *Explosion.*

Certamente o cursista teve que tratar desses subproblemas no processo de desenvolvimento de seu jogo, podendo ter sido considerado inicialmente. No entanto, ao descrever seu projeto, o cursista não discorreu sobre essas mecânicas. Sendo assim, em nosso entendimento, é possível afirmar que o cursista compreendeu parcialmente em que consiste a decomposição e como ela pode ser útil no processo de resolver o problema original, nesse caso, construir o jogo.

Na Figura 35, verifica-se o conceito computacional paralelismo, pois o cursista organizou a execução simultânea de tarefas traçadas em sua formulação do problema. No conceito computacional evento, *quando eu receber iniciar*, o cursista programou dois blocos que apresentam paralelismo, sequências acontecendo ao mesmo tempo, mudando de fantasia, de posições no cenário e tocando som galáxia até o final.

Figura 35 – Jogo *Batalha Estelar*



Fonte: Jogo Batalha Estelar.

No que diz respeito ao **reconhecimento de padrões**, o cursista relata que:

Os movimentos dos atores e os comandos utilizados pelo jogador.

Em seu relato, o cursista não explicita quais reconhecimentos de padrões foram úteis na construção da jogabilidade de seu constructo. Porém, ao analisar sua construção como um todo e os elementos que a compõem, identifica-se o reconhecimentos de padrões, como as naves *aliens*, que eram duplicadas e mudavam a fantasia, modificando o cenário conforme programação dos eixos x, e na mudança de cenário também percebemos que o cursista utilizou

um mesmo padrão, repetindo os blocos, pois somente mudando em relação aos eixos x e y, as teclas utilizadas seguem o mesmo padrão, mudando somente as letras utilizadas e sua lateralidade em relação ao cenário. O cursista repetiu os blocos e mudou a posição dos eixos. Na Figura 36, apresentamos os blocos utilizados nesse tópico específico.

Figura 36 – Jogo *Batalha Estelar*

Fonte: Jogo Batalha Estelar.

No que diz respeito aos conceitos computacionais de Resnick e Brennan (2012), podemos perceber que o cursista utilizou l*oops* (repetições), assemelhando-se ao pilar reconhecimento de padrões.

Percebe-se, na figura 36, que ele utilizou o bloco de condicionais simples, que depois de feito o teste lógico, seria a pergunta *tecla W pressionada*; se a resposta for verdadeira, ele executa o código associado ao bloco adicionado, no caso aqui programado, nessa tecla seria adicionar *10 ao eixo y*.

Quanto à **abstração,** segue o relato do cursista no fórum:

> *Fazer com que os projéteis da nave do jogador atinjam os alvos e consigam se movimentar e não sejam atingidos pelas bombas e nem pelas naves inimigas.*

Verificando o projeto do cursista, percebemos que os projéteis da *nave-jogador*, nomeados como *tiros*, realizam sua função ao ser acionada a tecla espaço. O cursista não relata claramente a jogabilidade, mas acessando sua construção, verifica-se essa funcionalidade. A abstração apresenta-se quando o cursista afirma o objetivo, que seria fugir das bombas das naves inimigas.

Em relação à **produção de algoritmos**, o cursista, em sua postagem no fórum do curso, escreveu que:

> *Organizar os blocos para que os atores se movimentem pelo cenário e não sejam atingidos.*

De acordo com o cursista, sua **produção de algoritmos** é tentar organizar os blocos para não ser atingido. Percebe-se que realmente realiza a programação dos blocos fazendo uso das teclas A, W, Z[8] e D, as quais têm funções de locomoção da *nave-jogador*, que já foram apresentadas anteriormente, através da programação dos blocos em linguagem que a máquina possa entender a sequência de passos (vire à direita, a esquerda), comandos executáveis.

Ao cursista escolher esse ou aquele modo de operacionalizar a solução de um problema, as condicionais estão presentes em seu projeto. Notamos o conceito computacional condicionais, os quais são utilizados na construção do jogo digital os blocos *sempre* e os blocos *se, então* na figura 36. O bloco em que o cursista utilizou *sempre* demonstra que executará o comando indefinidamente, ou seja, pressionado as teclas W, A, Z, D, serão adicionados valores para x e y, modificando a locomoção da nave. Percebe-se que a condição para que haja locomoção depende das teclas pressionadas.

No último processo de Pensamento Computacional analisado, o cursista relatou sobre a **depuração** em seu projeto:

> *Conferir se todos os atores funcionam perfeitamente para que o jogo possa ser conduzido por seus jogadores.*

Percebe-se que o cursista relatou que a **depuração** aconteceu em todo o seu processo de construção. Na postagem, ele escreveu que os jogadores podem conduzir e jogar com objetivo de fugir das naves inimigas. Contudo, nas construções, percebe-se que pode haver pequenos ajustes que não foram verificados pelo cursista, como exemplo citamos a troca de letra no comando para navegar no cenário, em que o cursista escreveu uma letra e programou outra. Contudo, isso não impediu que o jogador pudesse jogar, testando outras teclas.

---

[8] Adotamos a tecla com a letra Z, que foi a intenção do cursista.

Realizada essa análise sobre o jogo *Batalha Estelar*, é possível concluir que o cursista mobilizou os seis pilares do Pensamento Computacional durante e após o processo de desenvolvimento de seu projeto.

No momento da depuração, por vezes devemos analisar se o que está sendo executado/processado está de acordo com o que foi planejado na formulação do problema, demonstrando que o que foi idealizado pelo cursista chegou próximo ao que pensou.

Analisando o projeto do cursista no interior da plataforma Scratch, verifica-se que o cursista não utilizou *operadores* em sua construção.

6.3 Análise do jogo Tiro ao Pato

O terceiro jogo analisado é intitulado *Tiro ao Pato*, idealizado e realizado pela cursista Josiani. Realizamos uma descrição geral do projeto desenvolvido pela cursista, seguindo os mesmos critérios de análise do jogo digital *Arco e flecha e Batalha Estelar*.

No ambiente Scratch[9], a cursista postou uma breve descrição no campo de instruções para os jogadores:

> *Ao iniciar o jogo, use o mouse para levar o alvo até o pato, pressione o botão do mouse para atirar no pato e somar pontos. O jogo tem duração de 60 segundos.*

A descrição da cursista a respeito de seu projeto fornece instruções claras de como o jogador deve proceder ao iniciar o jogo, explicitando suas mecânicas.

O jogo possui quatro objetos ao todo. O primeiro, denominado pela cursista de *pato1*, é responsável por uma das mecânicas do jogo, o alvo. O segundo objeto, por sua vez, refere-se à *mira*, que aparece para o jogador quando o jogo é iniciado. O botão *play* tem a função de dar início ao jogo ao ser clicado. O quarto objeto, por fim, é o *pato2*, que se movimenta no palco em movimentos aleatórios, que a *mira* precisa atingir para pontuar.

Além desses elementos, há três cenários que aparecem no decorrer do jogo. O primeiro é utilizado no início do jogo, mostrando seu título, *Tiro ao pato*, e o botão *play* para ser clicado e iniciar o jogo. O segundo cenário (Figura 37) aparece ao iniciar o jogo, mostrando o movimento do ator *pato1* e o movimento da *mira*, que deve ser conduzida pelo jogador utilizando o mouse, conforme instruções da cursista.

[9] O jogo pode ser acessado em: https://scratch.mit.edu/projects/872603400.

Percebe-se que a cursista utilizou alguns conceitos computacionais de Resnick e Brennan (2012) no projeto desse jogo.

De acordo com a programação de seu projeto, é possível perceber a utilização de blocos que remetem uso dos conceitos computacionais Variáveis, Eventos, Paralelismo, *Loops*, Dados e Condicionais, tratados no decorrer do texto.

Figura 37 – Jogo Tiro ao pato - início

Fonte: Jogo Tiro ao pato.

Ao acionar o botão *play*, iniciando o jogo, pode-se observar que o objeto pato*1* movimenta-se aleatoriamente pelo cenário, e com o botão do mouse, o jogador deve mirar nesse objeto, o qual possui um alvo desenhado em vermelho e branco (Figura 37).

Desse modo, o jogador deve tentar atingir os patos para marcar pontos. Os pontos foram programados sendo armazenados por meio da variável *pontos*, como podemos verificar na Figura 38.

Figura 38 – Jogo Tiro ao pato-pontos

Fonte: Jogo Tiro ao pato.

Para isso, posiciona-se a *mira* (em preto) sobre o alvo em vermelho e, então, o jogador deve pressionar o botão esquerdo do mouse para *atirar*, marcando um ponto. Após quinze segundos do início do jogo, o segundo pato aparece, também com movimentos aleatórios no cenário.

Em suma, o objetivo do jogo é que o jogador pontue o máximo possível, acertando a maior quantidade de vezes os atores principais *pato1* e *pato2* no centro ou próximo da região central de seus corpos, contabilizando um ponto de cada vez. Após sessenta segundos de cliques nos atores, o jogo é finalizado e o jogador pode reiniciar o jogo.

Na análise da programação do jogo digital, foi identificado um único som que é reproduzido no jogo. Ele é disparado quando o jogador aperta o botão do mouse e acerta a *mira* no corpo do pato (alvo), aumentando simultaneamente a pontuação. Em outras palavras, esse som é um *feedback* do jogo de que o pato foi atingido pelo jogador e ele marcou um ponto.

As variáveis têm como objetivo, nesse caso, armazenar dados para serem acessados à medida em que for necessário (Marji, 2014).

Percebe-se a apropriação do conceito computacional variáveis na figura 39, cujo objetivo no projeto *Tiro ao pato* é armazenar uma quantidade de memória suficiente, como uma caixa. Aparecem as variáveis cronômetro (contagem do tempo ao começar o jogo de 60 segundos) e pontos (mostra quantos pontos o jogador fez por partida).

No primeiro pilar do Pensamento Computacional, a **formulação do problema,** a cursista escreveu o seguinte texto em sua postagem no fórum:

> *Construir um jogo em que 2 patinhos surgem e se deslocam aleatoriamente pelo cenário e com o mouse o jogador deve tocar e atirar no pato para marcar pontos.*

É possível perceber, a partir do comentário acima, que a cursista teve, desde o começo do processo de programação do jogo, uma ideia clara sobre o problema que deveria ser resolvido, que destacamos: criar um jogo em que o jogador controlasse uma mira com o objetivo de acertar um alvo (pato).

Quanto ao processo de **decomposição**, a cursista faz a seguinte descrição:

> *- Construir um cenário de abertura com o botão iniciar que quando clicado mudaria para outro cenário;*
> *- Fazer o ator pato surgir e deslizar aleatoriamente no cenário, da direita para à esquerda;*
> *- Após 15 segundos de jogo, um segundo ator pato deve surgir e deslizar aleatoriamente no cenário, da esquerda para a direita;*
> *- Fazer o ator mira acompanhar o mouse e quando clicado corretamente sobre os atores "pato" o mesmo desapareça, emita um som e some pontos;*
> *- Cada pato abatido vale 1 ponto;*

- Finalizar o jogo após 60 segundos, mudar o cenário para fim de jogo, parar e esconder todos os atores.
- Os pontos e o cronômetro são zerados no início do jogo.

Percebe-se que a cursista apresenta o que deve ser desenvolvido em cada fase, evidenciando quais passos seriam necessários para atingir o seu objetivo, dividindo o problema inicial em problemas menores.

Com a finalidade de aprofundar nossa análise a respeito da **decomposição,** na sequência, discorremos sobre os sete subproblemas apresentados pela cursista.

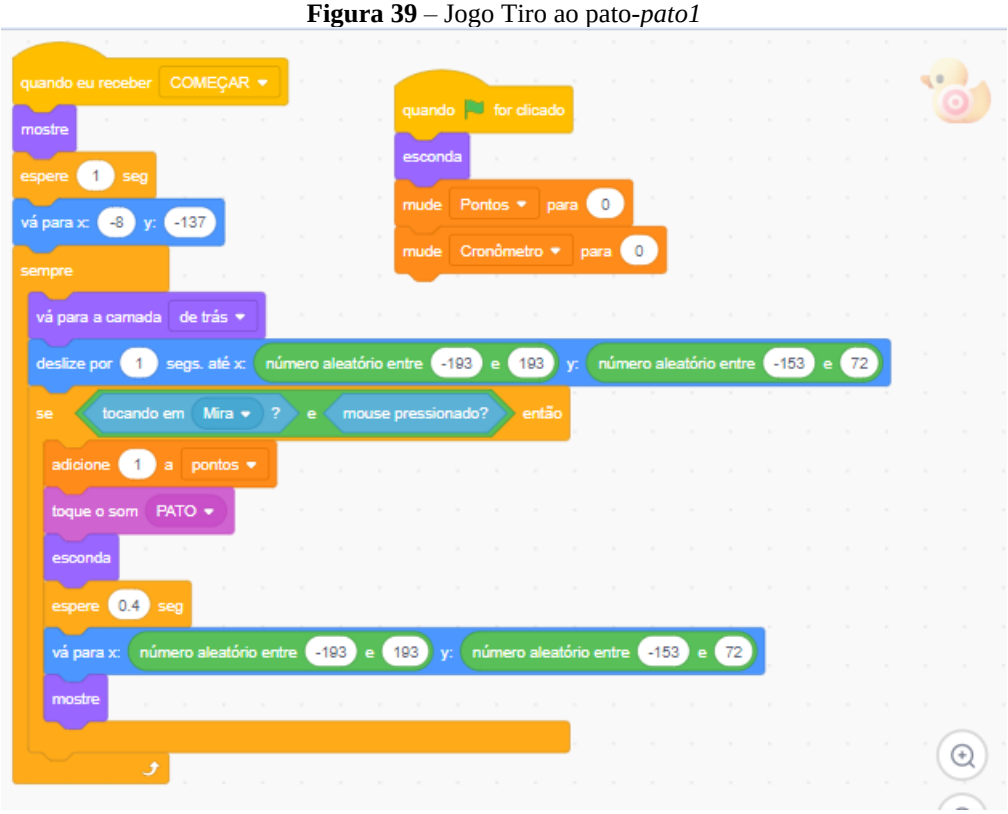
Construir um cenário de abertura com o botão iniciar que quando clicado mudaria para outro cenário.

A cursista, ao começar seu projeto, escolheu o cenário principal, intitulado *mar1*, mas apesar de ter relacionado como 1, ele somente aparece após clicar no *play*, que está no segundo cenário, denominado *mar2*.

Fazer o ator pato surgir e deslizar aleatoriamente no cenário, da direita para à esquerda.

Percebe-se que a cursista idealizou que o pato seria o alvo e que precisaria programar para que o alvo atingisse o primeiro pato que desliza aleatoriamente no cenário. Na Figura 39, pode-se verificar o código do primeiro pato em sua construção.

Figura 39 – Jogo Tiro ao pato-*pato1*

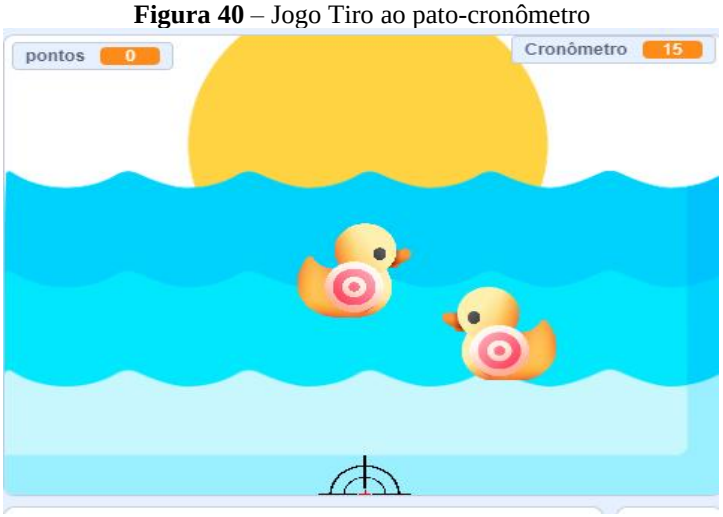


Fonte: Jogo Tiro ao pato.

Nota-se que, nos blocos, a cursista programa para que o *pato1* deslize aleatoriamente pelo cenário.

> *Após 15 segundos de jogo, um segundo ator pato deve surgir e deslizar aleatoriamente no cenário, da esquerda para a direita.*

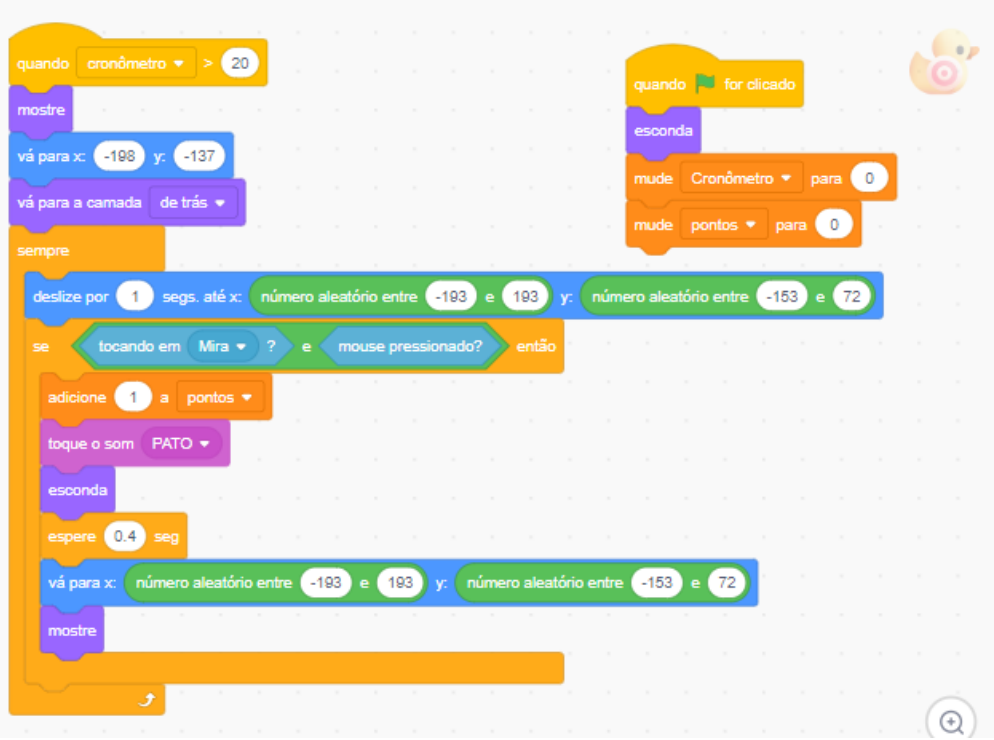
Figura 40 – Jogo Tiro ao pato-cronômetro



Fonte: Jogo Tiro ao pato.

Percebe-se que o segundo ator, *pato2*, aparece quando o cronômetro atinge 15 segundos, de acordo com a decomposição idealizada pela cursista.

Figura 41 – Jogo Tiro ao pato-*pato2*



Fonte: Jogo Tiro ao pato.

Assim, ela dividiu seu problema em sete partes, apresentadas por ela em sua descrição sobre decomposição, e foi construindo seu projeto. Na programação do *pato2* (Figura 42), podemos perceber que, quando a variável cronômetro atingir vinte segundos, o ator deslizará pelo cenário.

> *Fazer o ator mira acompanhar o mouse e quando clicado corretamente sobre os atores "pato" o mesmo desapareça, emita um som e some pontos;*
> *Cada pato abatido vale 1 ponto.*

Figura 42 – Jogo Tiro ao pato-mira

Fonte: Jogo Tiro ao pato.

Identificamos, na Figura 43, que a cursista programa a *mira* para ser o ponteiro do mouse, e quando alinhado ao alvo no corpo do *pato1* e *pato2*, emita um som (Figura 43), desapareça e some um ponto.

O conceito computacional eventos, que tem objetivo de produzir, iniciar uma ação, aparece no projeto, quando a cursista utilizou (1) *quando eu receber*, começa o jogo; (2) *quando a bandeira for clicada*, com objetivo de esconder o pato; (3) *quando eu receber*, com objetivo de terminar o jogo, aparece a mensagem FIM DE JOGO (Figura 44).

Figura 43 – Jogo Tiro ao pato-som

Fonte: Jogo Tiro ao pato.

> *Finalizar o jogo após 60 segundos, mudar o cenário para fim de jogo, parar e esconder todos os atores.*

Os pontos e o cronômetro são zerados no início do jogo.

Figura 44 – Jogo Tiro ao pato-fim de jogo

Fonte: Jogo Tiro ao pato.

O cenário três, *mar3*, foi programado para aparecer quando o jogo chega aos sessenta segundos (Figura 44), mostrando que o jogo acabou. Identificamos que a cursista programou que, ao clicar na bandeira verde, o cronômetro zera e começa novamente até que o jogador chegue a sessenta segundos.

Partindo de suas colocações no fórum, verificamos que a cursista, ao mobilizar o processo de **decomposição,** pode ter realizado os seguintes questionamentos:

> - É possível decompor as partes em partes menores?
> - Como a decomposição do problema pode servir para resolvê-lo ou compreendê-lo? (Espadeiro, 2021, p. 6).

Nota-se que a cursista percebeu que é possível resolver as partes do problema destacadas por ela, obtendo a solução ao final de cada parte/etapa.

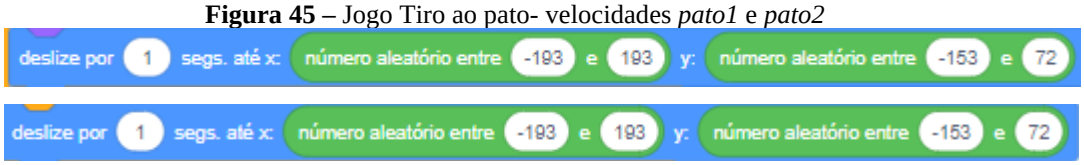
Em relação ao **reconhecimento de padrões**, a cursista explicou que:

> *- O movimento dos atores "pato" tem a mesma velocidade;*
> *- Os atores "pato" (res)surgem e se movimentam de forma aleatória, desaparecem quando são abatidos pela mira e ressurgem;*
> *- Soma-se 1 ponto a cada pato abatido;*
> *- O jogo se encerra após 60 segundos do início;*
> *- A pontuação e o cronômetro são zerados a cada início do jogo.*

Analisando cada frase apresentada pela cursista como reconhecimento de padrões, temos:

- O movimento dos atores "pato" tem a mesma velocidade.

Figura 45 – Jogo Tiro ao pato- velocidades *pato1* e *pato2*



Fonte: Jogo Tiro ao pato.

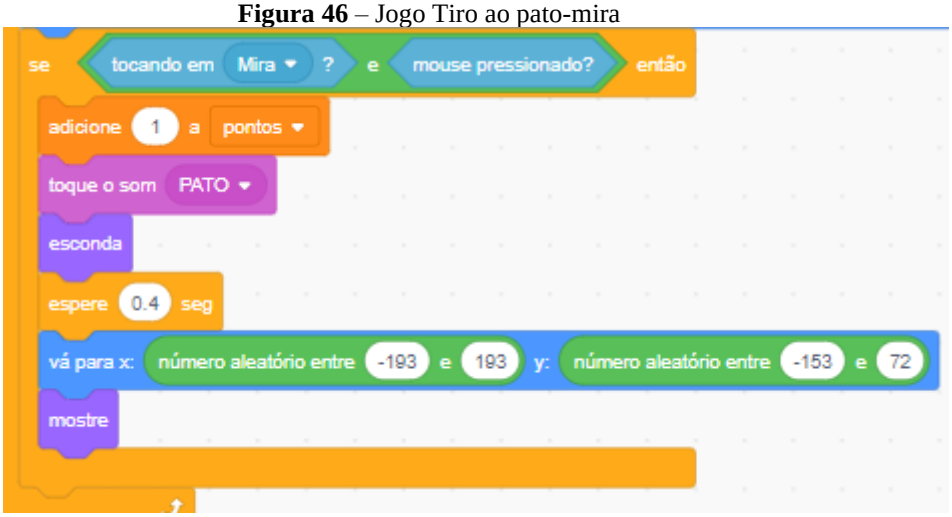
A cursista programou os atores *pato1* e *pato2* com a mesma velocidade. Podemos verificar, na Figura 46, que os sensores e operadores utilizados foram os mesmos.

O conceito computacional *loop* (laços), que demonstra a repetição dos atores *pato1* e *pato2* com a mesma velocidade, assemelha-se ao pilar reconhecimento de padrões. Podemos verificar que os sensores e operadores utilizados foram os mesmos.

Utilizando *loops,* não é necessário repetir várias vezes o mesmo comando, e segundo Marji (2014), é chamado de laço definido, e as iterações aparecem como repetições na velocidade dos patos, na soma de pontos, na pontuação zerada a cada início de jogo, demonstrando o objetivo dos *loops*, que quando a condição for verdadeira, o comando será repetido.

> *Os atores "pato" (res)surgem e se movimentam de forma aleatória, desaparecem quando são abatidos pela mira e ressurgem;*
> *Soma-se 1 ponto a cada pato abatido.*

Figura 46 – Jogo Tiro ao pato-mira



Fonte: Jogo Tiro ao pato.

Na Figura 46, percebe-se que, na programação da *mira*, que ao tocar nos atores, eles somam um ponto na variável, tocam o som pato, escondem e ressurgem após quatro segundos, determinando sua descrição sobre reconhecimento de padrão, que é o mesmo para os dois atores.

> *O jogo se encerra após 60 segundos do início;*
> *A pontuação e o cronômetro são zerados a cada início do jogo.*

Identifica-se, quanto aos conceitos computacionais, na Figura 47, que a cursista possivelmente pensou nos blocos que utilizaria para possível programação da mira em relação aos patos 1 e 2. Assim que ela reconheceu o padrão, demonstrou ter pensado em repetir a programação para *pato1* e *pato2*, demonstrando o paralelismo no movimento dos patos pelo cenário.

Figura 47 – Jogo Tiro ao pato-início/fim de jogo

Fonte: Jogo Tiro ao pato.

Para início e fim do jogo (Figura 47), também foi utilizado **o reconhecimento de padrões** para determinar a repetição dos mesmos comandos.

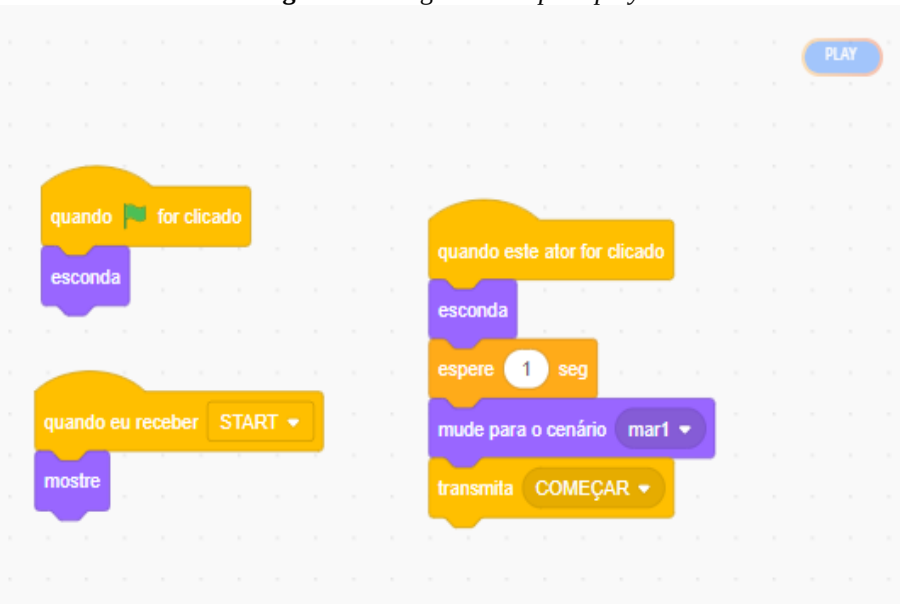
Quanto à **abstração**, verificamos que a cursista postou no fórum:

> *- Sorteia-se uma posição aleatória no cenário, respeitando um limite pré-determinado, onde os objetos "pato" irão surgir e deslizar por 1 segundo ou até que seja atingido pela mira. O processo se repete a cada 1 segundo.*
> *- O contato entre os atores "pato" e o ator mira modifica a pontuação do jogo;*
> *- A mira se movimenta conforme o jogador movimenta o mouse.*

Percebe-se que a cursista programou partindo do início do jogo com o botão *play* (Figura 49), e logo após descreveu que o contato entre os objetos *pato* e *mira* modificaram a pontuação do jogo. Então existe a hipótese de que ela pode ter realizado questionamentos de qual seria a informação importante para solução da questão. Os questionamentos que a cursista

supostamente tenha feito levaram-na ao processo de abstração, que consistiu em programar para que os objetos programados, ao serem atingidos, acionariam a variável pontos.

Figura 48 – Jogo *Tiro ao pato-play*



Fonte: Jogo Tiro ao pato.

Em relação a **produção de algoritmos**, a cursista salientou no fórum:

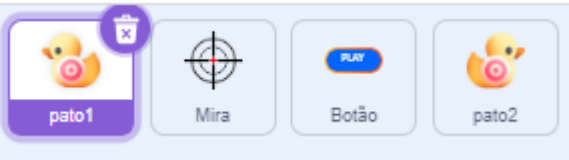
> *- Construir blocos separados para tratar os comportamentos dos atores pato1, pato2, mira e botão:*
> *- Quando o jogo inicia;*
> *- Deslocamentos;*
> *- Pontuação;*
> *- Testes de contatos.*

Percebe-se que a cursista explicou como pensou na produção de algoritmos, pois cada bloco construído por ela tem funções que verificamos que podem ser também relacionadas com a abstração. Cada bloco programado mostra que a cursista idealizou e construiu os comandos dos patos, mira e os botões dos deslocamentos aleatórios, pontuação e testes de contatos.

Destacamos, partindo de cada frase da cursista e de acordo com sua construção, como supostamente ocorreu a produção de algoritmos.

> *- Construir blocos separados para tratar os comportamentos dos atores pato1, pato2, mira e botão.*

Figura 49 – Jogo *Tiro ao pato-atores*



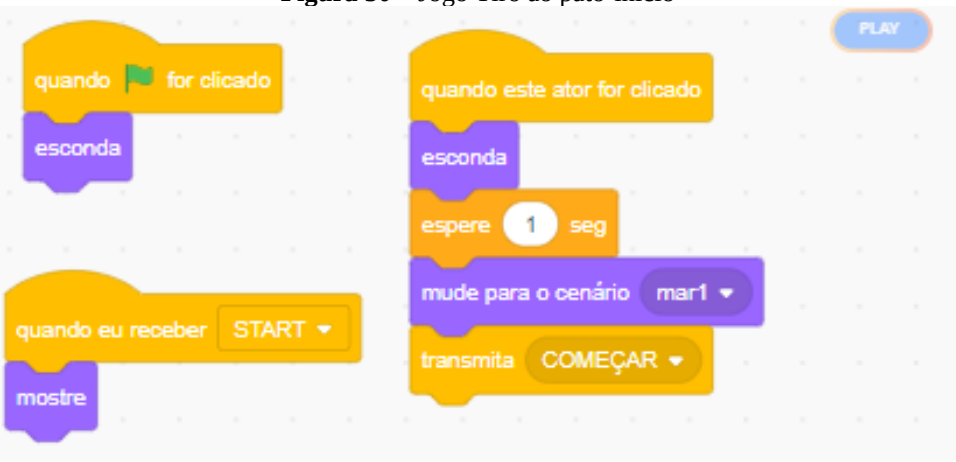
Fonte: Jogo Tiro ao pato.

Nota-se que a cursista programou os quatro atores e, assim, produziu algoritmos para cada ator (Figura 50).

- *Quando o jogo inicia.*

Na figura 50, especificamente nos atores *pato1* e *pato2*, visitando o projeto em seu interior, percebe-se que foi utilizado o conceito computacional paralelismo, em que os dois atores são acionados após 15 segundos do início do jogo, acontecendo ao mesmo tempo uma sequência de instruções, pois o Scratch suporta paralelismo. O paralelismo também ocorre quando o ator *mira* é acionado, utilizando o mouse como controle, ao mesmo tempo em que os atores deslizam pelo cenário.

Figura 50 – Jogo Tiro ao pato-início

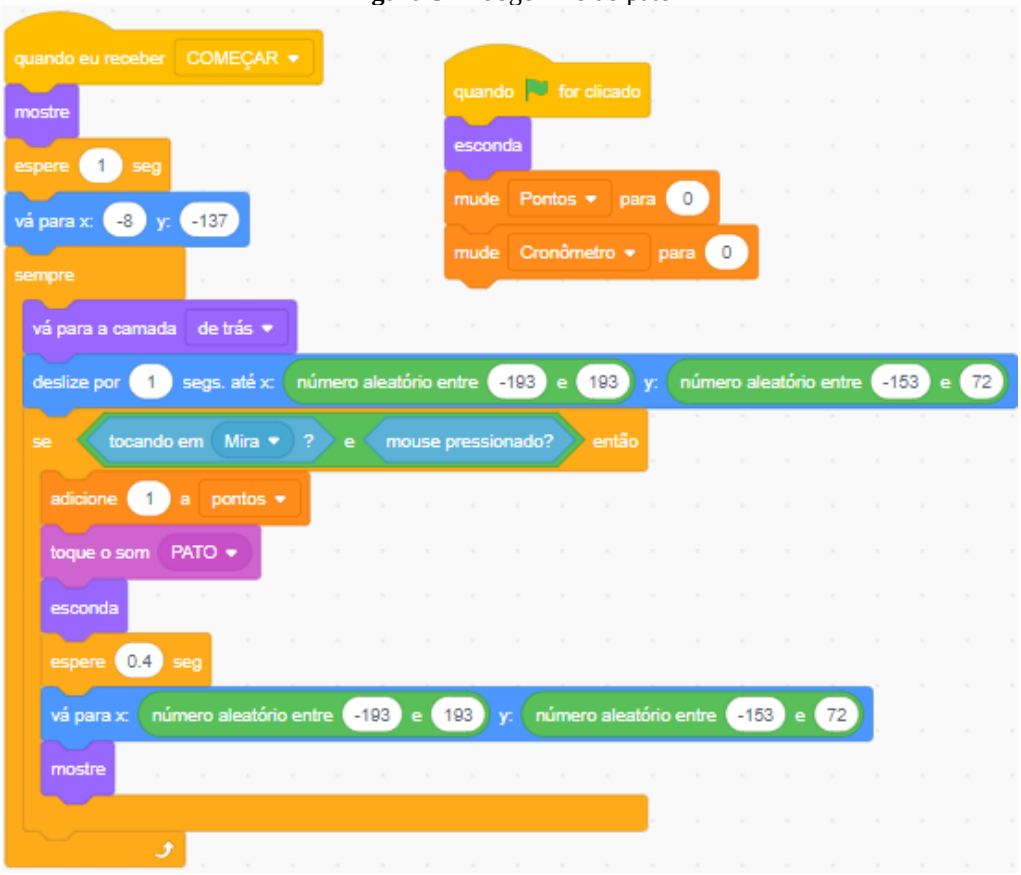


Fonte: Jogo Tiro ao pato.

Percebe-se (Figura 50) que esse comando produziu o algoritmo para iniciar o jogo, mudando o cenário após apertar o botão *play*.

- *Deslocamentos;*
- *Pontuação;*
- *Testes de contatos.*

Figura 51 – Jogo Tiro ao pato



Fonte: Jogo Tiro ao pato.

Na Figura 51, observa-se que os comandos dos blocos, evento, operadores, sensores e variáveis realizam a **produção do algoritmo** como citado pela cursista, acionando a pontuação, deslocamentos e testes de contato entre os atores.

Na figura 52, verifica-se o uso do conceito computacional operadores para programar a posição aleatória dos patos no cenário, deslizando por um segundo e mudando de posição, permitindo à cursista realizar manipulações numéricas e de *string*s.

Por fim, no que diz respeito ao processo da **depuração**, a cursista afirmou que:

> *O processo de depuração acontece a cada bloco construído entre uma ação e outra, nos momentos em que são realizados testes para verificar se o seu funcionamento está de acordo com o que se pretendia realizar no Scratch.*

A cursista salientou que a depuração acontece nos momentos de testes da funcionalidade do jogo. Demonstrou claramente a mobilização desse pilar em sua postagem, pois de acordo com Espadeiro (2021), a depuração infere questionamentos que acontecem durante todo o processo, para garantir que o projeto funcionou ou não, se conseguiu corrigir os erros ou não, e verificando, ao final, se o resultado funcionou corretamente. Analisando a postagem e o projeto

final da cursista, percebemos que ela fez a depuração e resolveu o problema, pois partindo de sua formulação de problema, concluímos que ela conseguiu com que o jogo funcionasse.

7 CONSIDERAÇÕES FINAIS

A construção desta dissertação baseou-se, de maneira principal, na temática do papel relevante do Pensamento Computacional no desenvolvimento de práticas pedagógicas para um processo de ensino e de aprendizagem mais motivador e qualitativo. Além disso, reconhecemos o Pensamento Computacional como fator determinante do aprendizado amplo, bem como para soluções de problemas diversos, especificamente no contexto educacional, ou mesmo na vida cotidiana.

A fundamentação teórica apontou para um franco desenvolvimento sobre a temática do Pensamento Computacional como fruto de uma sociedade cada vez mais tecnológica, que necessita compreender o processo de pensamento como similar ao funcionamento de um computador, para buscar meios de solução para problemáticas diversas.

Foram abordados, na análise dos jogos digitais, os conceitos computacionais de Brennan e Resnick (2012), com intuito de aproximar as técnicas utilizadas durante a codificação e os processos do Pensamento Computacional.

Nesse sentido, não há uma relação direta entre Pensamento Computacional e as tecnologias digitais atuais, sendo importante considerar que essa prática ainda que não fosse nomeada dessa maneira, pois é anterior à criação da ciência da computação, já existindo desde quando se utilizavam análises numéricas e tabulação. Considerando que após a criação da computação essas práticas foram aprimoradas, elas ampliaram possibilidades.

A parte prática do estudo foi construída com base na apresentação dos processos do Pensamento Computacional e na realização de um curso para professores do Núcleo Regional de Educação de Apucarana-PR, com o intuito de conceituar o Pensamento Computacional como recurso no processo de ensino-aprendizagem, ampliando possibilidades de ensino através da utilização da plataforma Scratch.

Os professores que participaram do curso desenvolveram atividades diversas na plataforma. Alguns deles, aprenderam a utilizar a referida plataforma durante o curso, sem contar com nenhum tipo de conhecimento prévio sobre esse recurso. Os docentes desenvolveram jogos que visando aos processos do Pensamento Computacional, que também lhes foram apresentados na parte teórica do curso ofertado.

No decorrer dessas atividades, os docentes relacionavam o Pensamento Computacional aos jogos desenvolvidos por eles. É interessante pontuar que cada professor idealizou a realidade de abordagem nas próprias turmas e como os jogos que construíram poderiam ser

úteis para o processo de aprendizagem de seus alunos na disciplina de Pensamento Computacional.

No contexto trabalhado com os professores durante a oficina de Scratch, alguns docentes vislumbraram oportunidades de atividades construídas por eles nessa plataforma para que pudessem ser aplicadas em sua rotina de aulas, contribuindo para o aprendizado dos alunos e trazendo novas abordagens metodológicas.

Após a realização do curso, foram analisados os jogos criados pelos docentes, observando qual a relação com os processos constituintes do Pensamento Computacional. Foi considerado um recorte das produções dos professores que participaram dessa prática, com destaque para aqueles processos que foram mobilizados pelos participantes durante a realização do curso.

O curso foi proposto com três objetivos, atingindo-os satisfatoriamente, uma vez que capacitou docentes em diversos aspectos: conceitos, processos e aplicações do Pensamento Computacional; funcionamento e tecnologia do programa Scratch; e utilização do Scratch na elaboração de jogos digitais, com base no Pensamento Computacional. Também foram abordados conceitos computacionais de Brennan e Resnick (2012) na análise dos jogos, mostrando suas semelhanças com os processos analisados nos jogos digitais selecionados.

A experiência conquistada durante esse período foi ampla e enriquecedora, possibilitando uma visão direta sobre a prática docente e reconhecendo que há falta de oportunidade de uma formação diversa e atual para os professores, que contribua para uma prática docente que possa encantar e motivar seus alunos, já imersos em recursos tecnológicos. Atender a essas demandas traz possibilidades para que o professor tenha conhecimento e recursos para diversificar sua atuação.

Partindo da análise dos jogos digitais criados pelos cursistas, chegamos à conclusão da efetiva mobilização dos processos e conceitos computacionais, pois todos foram citados, analisados, utilizados, enfim, mobilizados pelos cursistas. Nos três jogos digitais escolhidos para serem analisados e minuciosamente recortados e descritos, percebe-se a mobilização de cada processo. Assim, nossa pesquisa responde à pergunta, foram mobilizados os processos e conceitos computacionais em suas construções. Deixamos, aqui, um estudo partindo dos processos do Pensamento Computacional para que outros pesquisadores possam dar continuidade com estudos e pesquisas que enriqueçam ainda mais o assunto.

REFERÊNCIAS

ALMEIDA, A. V.; MIRANDA, G. M. O ensino dos pilares do Pensamento Computacional para professores da Educação Básica. *In*: Workshop sobre Educação em Computação (WEI), 31., 2023, João Pessoa/PB. **Anais** [...]. Porto Alegre: Sociedade Brasileira de Computação, 2023.

BARCELOS, T.S. **Relações entre o pensamento computacional e a matemática em atividades didáticas de construção de jogos digitais**. Tese (Doutorado em Ensino de Ciências e Matemática). Universidade Cruzeiro do Sul, São Paulo, 2014.

BARCELOS, T.; BORTOLETTO, R.; ANDRIOLI, M. Formação online para o desenvolvimento do Pensamento Computacional em professores de Matemática. In: **Anais** dos Workshops do Congresso Brasileiro de Informática na Educação. [S.l.: s.n.], 2016. v. 5, n. 1, p. 1228. Citado nas páginas 14, 22 e 52.

BARCELOS, T. S.; MUÑOZ, R.; VILLARROEL, R.; SILVEIRA, I. F. Jogar para aprender, construir para jogar: oficinas de construção de jogos digitais para o desenvolvimento do pensamento computacional. *In* RAABE, A.; ZORZO, A. F.; BLIKSTEIN, P. (Orgs.). **Computação na Educação básica, fundamentos e experiências.** Porto Alegre: Penso, 2020, p. 152-174.

BARCELOS, T. S.; SILVEIRA, I. F. Pensamento Computacional e Educação Matemática: Relações para o Ensino de Computação na Educação Básica. *In* **Workshop sobre Educação em Computação**, Dourados, 2012.

BARICHELLO, L. **Introdução ao Pensamento Computacional**. Editora Instituto Nacional de Matemática Pura e Aplicada, 2021.

BARROS, T. T. T.; REATEGUI, E. B.; MEIRA, R. R.; TEIXEIRA, A. C.. Avaliando a formação de professores no contexto do pensamento computacional. **Renote**: revista novas tecnologias na educação. Vol. 16, n.2 (dez. 2018), p. 556-565.

BASAWAPATNA, A. R. *et al*. The zones of proximal flow: guiding students through a space of computacional thinking skills and challenges. *In* **Internacional ACM Conference on Internacional computing education research**, 9, San Diego, 2013.

BELETI JÚNIOR, C. R.; SFORNI, M. S. F. Possibilidades do Pensamento Computacional: um novo olhar teórico. **Anais** do XXXII Simpósio Brasileiro de Informática na Educação (SBIE 2021).

BORDINI, A. **Pensamento Computacional nos Ensinos Fundamental e Médio**: uma revisão sistemática. Artigo de qualificação, maio de 2017.

BORGES, K. S.; NORONHA, F. P. T.; BACKES, L. Pensamento computacional desplugado: análise da experiência com o projeto Pi. **Tecnologias, Sociedade e Conhecimento**, Campinas, SP, v. 7, n.º 1, p. 141-155, 2020. Disponível em: https://econtents.bc.unicamp.br/inpec/index.php/tsc/article/view/14705/9694. Acesso em: 02 jun. 2022.

BRACKMANN, C. P. **Desenvolvimento do Pensamento Computacional através de atividades desplugadas na Educação Básica**. Tese (Doutorado em Informática na Educação). Universidade Federal do Rio Grande do Sul - UFRGS, 2017.

BRACKMANN, C. P.; AUGUSTO, D. B.; CASALI, A. C.; HERNÁNDEZ, S. Pensamento computacional: Panorama nas américas. *In* **XVIII Simpósio Internacional de Informática Educativa**, SIIE, 2016.

BRASIL. Ministério da Educação. **Base Nacional Comum Curricular**. Brasília, 2018.

BRENNAN, K.; RESNICK, M. **New frameworks for studying and assessing the development of computational thinking**. MIT Media Lab, Vancouver, Canada, 2012.

BRENNAN, K; RESNICK, M. New framework for studying and assessing the development of computational thinking. *In* Annual meeting of the American Educational Research Association. **Proceedings** [...]. Vancouver. AERA, 2012. Disponível em: https//web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf. Acesso em: 30 out. 2023.

CONFORTO, D.; CAVEDINI, P.; MIRANDA, R.; CAETANO, S. **Pensamento Computacional na Educação Básica**: Interface tecnológica na construção de competências do Século XXI. 5° SENID, Cultura Digital na Educação, 2018.

DANTAS, S. C. Pensando e resolvendo problemas com o GeoGebra. **Revista do Instituto GeoGebra Internacional de São Paulo**, [s. l.], v. 12, n. 2, p. 133-164, 2023. Disponível em: https://revistas.pucsp.br/index.php/IGISP/article/view/63719. Acesso em: 29 out. 2023.

DENNING, P. J. **Computing is a Natural Science**. Communications of the ACM, july, 2007, vol. 50, n. 7.

ESPADEIRO, R. G. O Pensamento Computacional no currículo de Matemática. **Educação e Matemática**: Revista da Associação de Professores de Matemática, [s. l.], n. 162, p. 5-10, 2021. Disponível em: https://em.apm.pt/index.php/em/article/view/2737. Acesso em: 30 out. 2023.

GONÇALVES, L. M.; PORTELLA, A. C. F. ; LUZ, M. S. L. Softwares livres e equipamentos manufaturados: possíveis recursos para a integração curricular das TDIC. **Revista Observatório**, v. 5, n. 1, p. 455-477, 2019.

LEE, I. *et al*. Computational thinking for Youth in practice. **ACM Inroads**, v. 2, n.1, p 32-37, 2011.

LIRIO, J. R.; PRADO, S. P. Contradições da BNCC acerca do desenvolvimento e uso das TDICs e do pensamento computacional. **Educação Matemática Sem Fronteiras**: Pesquisas em Educação Matemática, v. 5, n. 1, p. 59-75, 2023. Disponível em: Acesso em 14 nov. 2023.

LOPES, H. M. B. **Do desplugado ao plugado**: uma proposta para o desenvolvimento do pensamento computacional e do pensamento matemático avançado em aulas do Ensino Médio. Instituto Federal do Espírito Santo, Vitória-ES, 2022.

MARCOMINI, A. A teoria do flow e você. **RTS –** Solução em Gestão de Pessoas ltda. [s.d]. Disponível em: http://www.rtsconsultoria.com.br/a-teoria-do-flow-e-voce/. Acesso em: 30 out. 2023.

MARJI, M. **Learn to program with scratch**. Scratch Press. Portuguese - language, by Nonatec Editora Ltda. All rights reserved, 2014.

MARJI, M.; PRATES, R (Trad.). **Aprenda a programar com scratch:** uma introdução visual à programação com jogos, arte, ciência e matemática. São Paulo: Novatec, 2014.

NARDELLI, H. Viewpoint Do we really need Computational Thinking? **Communications of the ACM**, February 2019, vol. 62, n. 2.

NAVARRO, E. R. **O desenvolvimento do conceito de Pensamento Computacional na Educação Matemática segundo contribuições da Teoria Histórico-Cultural**. Universidade Federal de São Carlos, 2021.

PLEWKA, V. G.; DANTAS, S. C. Concepções acerca do pensamento computacional presentes na BNCC e do referencial curricular do Paraná no ensino médio. In: Encontro Paranaense de Tecnologia na Educação Matemática, 3., 2023, Apucarana. **Anais** [...]. Apucarana: EPTEM, 2023. p. 1-15.

RAABE, A.; ZORZO, A.; BLIKSTEIN, P. (org). **Computação na educação básica**: fundamentos e experiências. Porto Alegre: Penso, 2020.

RAABE, A.; VIEIRA, M. V.; SANTANA, A. L. M.; GONÇALVES, F.; BATHKE, J. **Recomendações para introdução do pensamento computacional na Educação Básica**. 2015. Disponível em: https://sol.sbc.org.br/index.php/desafie/article/view/10049/9931. Acesso em: 18 fev. 2023.

REIS, S. R.; BARICHELLO, L.; MATHIAS, C. V. Novos conteúdos e novas habilidades para a área de Matemática e suas tecnologias. **RIPEM**, v. 11, n.1, 2021 pp. 37-58.

RESNICK, M. **Jardim de infância para a vida toda**: por uma aprendizagem criativa, mão na massa e relevante para todos. Porto Alegre: Penso, 2020.

RESNICK, M. **Jardim de infância para a vida toda.** Trechos do capítulo 1. 2017. Disponível em: https://lcl.media.mit.edu/resources/readings/chapter1-excerpt.pt.pdf?pdf=ch1-pt. Acesso em: 30 out. 2023.

RIBEIRO, L.; FOSS, L.; CAVALHEIRO, S. A. C. Entendendo o Pensamento Computacional. RAABE, A.; ZORZO, A. F.; BLIKSTEIN, P. (Orgs.). **Computação na Educação básica, fundamentos e experiências.** Porto Alegre: Penso, 2020, p. 16-30.

SANTOS, L. P.; PRADO, S. P.; ALMEIDA, N. T.; SOUZA, S. M.; DANTAS, S. C. Pensamento computacional: a construção do jogo pac-man no Scratch. In: Encontro Paranaense de Tecnologia na Educação Matemática, 3., Apucarana. **Anais** [...]. Apucarana: EPTEM, 2023. p. 1-15.

SILVA, L. C. L. **A relação do pensamento computacional com o ensino de matemática na educação básica**. Dissertação (Mestrado Profissional em Matemática) - Faculdade de Ciências e Tecnologia, Universidade Estadual Paulista, Presidente Prudente, 2019. Disponível em: https://repositorio.unesp.br/handle/11449/191251 Acesso em: 21 dez. 2024.

SOUSA, R. P.; MIOTA, F. M. C. S. C.; CARVALHO, A. B. G. (orgs.). **Tecnologias digitais na educação** [online]. Campina Grande: EDUEPB, 2011. 276 p. ISBN 978-85-7879-124-7.
TEDRE, M.; DENNING, P. J. The Long Quest for Computational Thinking. **Proceedings** of the 16th Koli Calling Conference on Computing Education Research, November 24-27, 2016, Koli, Finland: pp. 120-129.

TORRES, J.; FIGUEIREDO, M. Programação e Pensamento Computacional. **Educação e Matemática**, 2021, 162, Editorial. Disponível em: https://em.apm.pt/index.php/em/issue/view/162. Acesso em: 30 out. 2023.

TORRES, J.; FIGUEIREDO, M. Aprender Matemática para programar ou programar para aprender Matemática? **Educação e Matemática**, 2021, 162, 11-14. Disponível em: https://em.apm.pt/index.php/em/issue/view/162. Acesso em: 30 out. 2023.

VALENTE, J. A. Integração do pensamento computacional no currículo da Educação Básica: diferentes estratégias usadas e questões de formação de professores e avaliação do aluno. Revista e-Currículo [enlinea]. 2016, 14(3), 864-897. Disponível em: https://www.redalyc.org/articulo.oa?id=76647706006. Acesso em: 30 set. 2023.

VEE, A. Ideologies of a New Mass Literacy. **Conference on College Composition and Communication Convention**. The Riviera, Las Vegas. 14 Mar, 2013.

VICARI, R. M.; MOREIRA, A.; MENEZES, P. B. **Pensamento Computacional**: revisão bibliográfica. Projeto UFRGS, 2018.

WING, J. Computational thinking. **Communications of the ACM**, v. 49, n. 3, 2006. Disponível em: https://www.educacao.gov.br/arquivos/pdf/organizacoes-parceiras-bncc.pdf. Acesso em: 05 fev. 2023.