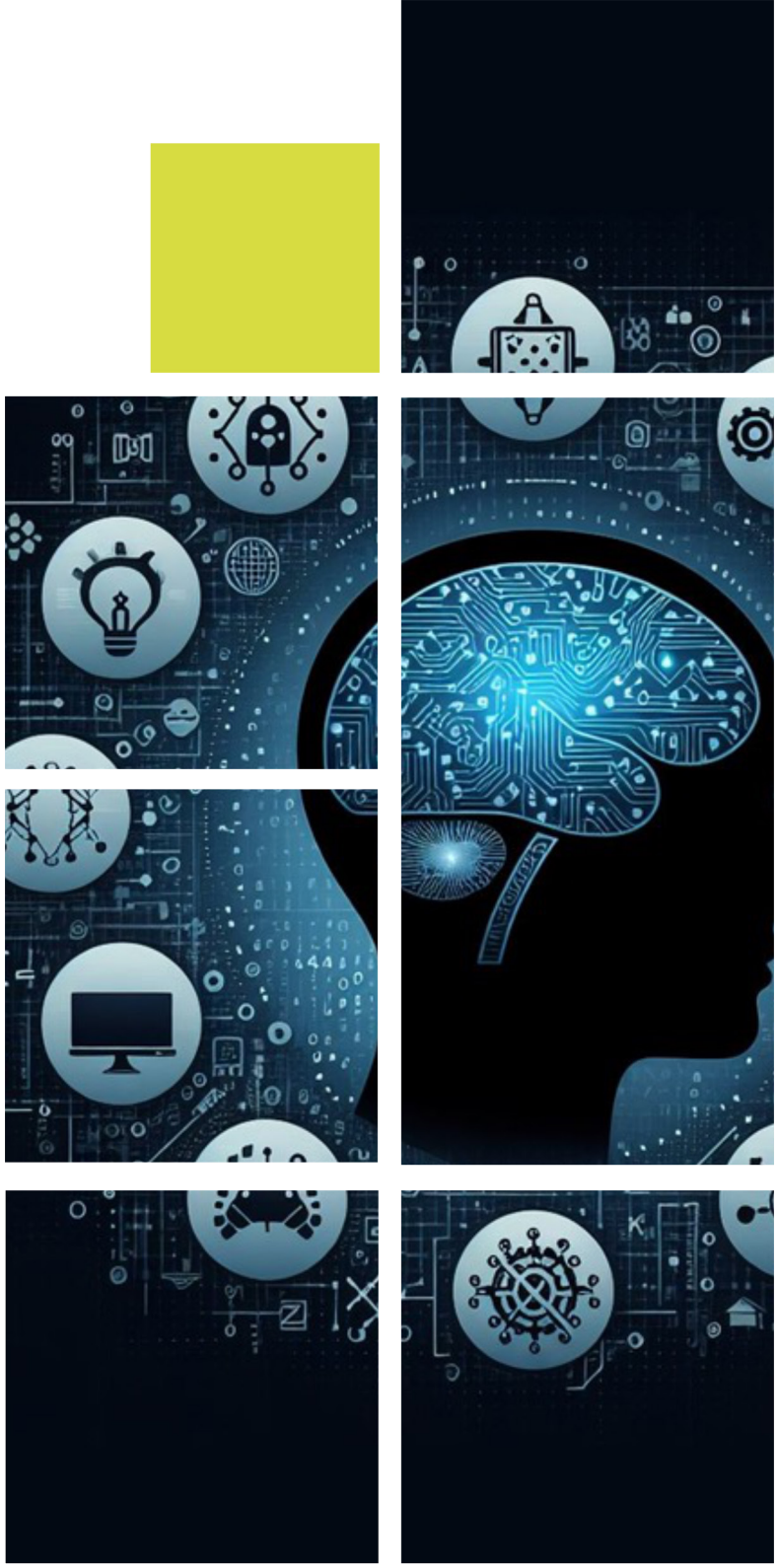


Pensamento computacional: teoria e prática

Esteic Janaina Santos Batista

EaD UFMS DIGITAL

AGEAD
Agência de Educação Digital e a Distância
UFMS

Pensamento computacional: teoria e prática

Esteic Janaina Santos Batista

EaD UFMS DIGITAL

AGEAD
Agência de Educação Digital e a Distância
UFMS

Dados Internacionais de Catalogação na Publicação (CIP)
(Diretoria de Bibliotecas – UFMS, Campo Grande, MS, Brasil)

Batista, Esteic Janaina Santos.
Pensamento computacional [recurso eletrônico] : teoria e prática. / Esteic Janaina Santos Batista. – Campo Grande, MS : Ed. UFMS, 2024.
80 p. : il. (algumas color.).

Dados de acesso: https://repositorio.ufms.br
Bibliografia: p.79-80.
ISBN: 978-85-7613-671-2
Produzido no âmbito do Programa UFMS Digital.

1. Inteligência computacional. 2. Solução de problemas. 3. Pensamento criativo. 4. Pensamento computacional. I. Batista, Esteic Janaina Santos. II. Título.

CDD (23) 006.3

Bibliotecário responsável: Tânia Regina de Brito – CRB1/ 2.395

Esteic Janaina Santos Batista

Pensamento computacional: teoria e prática

Campo Grande - MS
2024

editora UFMS

Sobre o E-book

Este e-book foi produzido no âmbito do **Programa UFMS Digital**, coordenado pela Agência de Educação Digital e a Distância da Universidade Federal de Mato Grosso do Sul.

Coordenação Geral

Hercules da Costa Sandim

Coordenação Pedagógica

Daiani Damm Tonetto Riedner

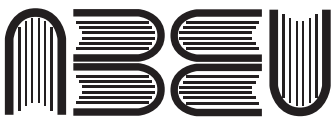
Projeto Gráfico e Diagramação

Maira Sônia Camacho

Revisão de Língua Portuguesa

Aline Cristina Maziero

Editora associada à



Associação Brasileira das Editoras Universitárias

SUMÁRIO

Prefácio 7

Apresentação 8

Capítulo 1 - Conceitos Básicos de Pensamento Computacional 9

Capítulo 2 - Pilares do Pensamento Computacional 20

Capítulo 3 - Pensamento Computacional com Jogos de Lógica 27

Capítulo 4 - Introdução a Algoritmos com *Code.org* e *GearsBot* 39

Capítulo 5 - Programação em Blocos com *Scratch* 56

Capítulo 6 - Desenvolvimento de Narrativas Digitais 66

Capítulo 7 - Análise e Planejamento de Soluções 72

Atividades Futuras 77

Palavras Finais 78

Referências 79

Prefácio

O pensamento computacional (PC) não se resume a saber programar ou a entender computadores. Trata-se de uma forma de pensar a resolução de problemas de maneira sistemática e eficiente. Essa abordagem faz uso de habilidades como a decomposição de problemas complexos em partes menores e mais gerenciáveis, o reconhecimento de padrões e a busca de similaridades a partir de experiências anteriores, a abstração de conceitos para simplificação de ideias e estratégias de solução e o desenvolvimento de algoritmos e processos com o passo-a-passo das soluções elaboradas. Em outras palavras, o pensamento computacional busca entender e criar soluções de maneira organizada, lógica e estruturada.

Neste livro, a autora explora os conceitos fundamentais do PC, começando com uma introdução à sua origem, evolução e importância no mundo moderno. Abordam-se os pilares essenciais dessa forma de pensar, proporcionando uma compreensão profunda dos elementos que a constituem, com subsídios teóricos e experimentais. Demonstra-se também como o PC pode ser desenvolvido por meio de jogos de lógica, uma maneira divertida e eficaz de internalizar estes conceitos. É introduzida a construção de algoritmos utilizando plataformas educacionais acessíveis, oferecendo ferramentas práticas para a criação de uma base sólida em lógica de programação.

Além disso, explora-se a programação em blocos com linguagens intuitivas que facilitam o aprendizado de conceitos de programação para todas as idades. A autora demonstra como desenvolver narrativas digitais, integrando criatividade e técnica para contar histórias interativas e envolventes. Discute-se, ainda, a importância da análise e do planejamento de soluções, habilidades cruciais para qualquer profissional que deseja aplicar o pensamento computacional em contextos reais, resolvendo problemas de maneira estruturada e eficiente.

Espera-se que este livro sirva como um guia, tanto para iniciantes, quanto para aqueles que já possuem algum conhecimento ou desejam aprofundar sua compreensão sobre o pensamento computacional. Acreditamos que o domínio do PC pode ajudar estudantes, professores e pesquisadores a serem habilidosos solucionadores de problemas com resultados inovadores e criativos. Convidamos você, leitor, a embarcar nesta jornada conosco, explorando como essa poderosa e divertida ferramenta pode transformar a maneira como pensamos, aprendemos e criamos soluções para problemas reais.

Boa leitura e boas-vindas ao mundo do pensamento computacional!

Amaury Antônio de Castro Junior
Hercules da Costa Sandim
Daiani Damm Tonetto Riedner

Apresentação

Em um mundo cada vez mais impulsionado pela tecnologia e pela inovação, a capacidade de entender e aplicar o Pensamento Computacional tornou-se indispensável. Este livro, “Pensamento Computacional: teoria e prática”, explora a essência desta habilidade crítica, que transcende a mera programação e se consolida como uma ferramenta fundamental na resolução de problemas em diversas áreas do conhecimento e atividades cotidianas.

O Pensamento Computacional nos permite decompor problemas complexos em partes menores e mais gerenciáveis, identificar padrões e tendências, abstrair e simplificar os elementos essenciais, e desenvolver soluções algorítmicas precisas. Ao integrar estas habilidades, indivíduos de todas as idades e profissões podem enfrentar desafios com uma nova perspectiva e eficiência notável.

Este livro foi concebido para fornecer uma compreensão profunda e prática sobre como cultivar e aplicar o Pensamento Computacional, independentemente do seu campo de atuação. O texto abrange desde a teoria por trás dessa habilidade revolucionária até aplicações práticas que ilustram seu impacto transformador em nossa maneira de pensar, aprender e trabalhar. Para te apoiar na prática das ferramentas apresentadas neste livro e outras correlatas, há uma playlist no YouTube com recursos cuidadosamente selecionados.

Espera-se que o leitor se sinta inspirado a explorar e adotar o Pensamento Computacional, não apenas como uma ferramenta educacional ou profissional, mas como um componente essencial para a inovação e a solução eficaz de problemas em sua vida diária.

Vamos descobrir como essa poderosa abordagem pode ampliar nossas capacidades e abrir novas possibilidades para o futuro.

Boa leitura e uma excelente jornada de aprendizado!

Capítulo 1
Conceitos básicos de Pensamento Computacional

Quando aprendemos matemática ou língua portuguesa nas escolas, estamos nos preparando para ser matemáticos ou escritores profissionais? A resposta, para muitos de nós, é: não. Essas disciplinas são fundamentais não porque todos utilizaremos suas técnicas e conhecimentos de forma profissional, mas porque elas nos equipam com habilidades e conceitos essenciais para nos comunicar e interagir com o mundo. Este mesmo raciocínio pode ser aplicado ao Pensamento Computacional. Mas o que exatamente é essa habilidade tão falada e por que ela é importante para todos, não apenas para os que trabalham diretamente com computadores?

Este capítulo irá explorar a origem, as transformações e a aplicação prática deste conceito crucial, destacando por que o Pensamento Computacional é fundamental para navegar e resolver os desafios do nosso tempo.

Quando chegamos aqui, era tudo *Twenty things to do with a computer*

As primeiras ideias desenvolvidas e disseminadas, que eventualmente se consolidaram sobre o conceito de Pensamento Computacional, foram apresentadas por Papert e Solomon (1972) no artigo *Twenty things to do with a computer* (Vinte coisas para fazer com um computador).

Seymour Papert foi um visionário no campo da inteligência artificial e da educação, pois introduziu a ideia de que a computação poderia ser uma ferramenta poderosa para o aprendizado e a resolução de problemas em várias áreas. Seu trabalho foi pioneiro na integração da computação com o ensino, vislumbrando uma época em que crianças usariam computadores como ferramentas para aprender, de maneira criativa e inovadora. Essa abordagem foi chamada por ele mais tarde de “construcionismo”, inspirando-se na teoria de aprendizagem de Jean Piaget, o construtivismo.

Papert co-criou a linguagem de programação Logo, desenvolvida na década de 1960 como parte de projetos de pesquisa em inteligência artificial no Massachusetts Institute of Technology - MIT, com a intenção de torná-la uma ferramenta educativa voltada para crianças. A principal característica da Logo é a “tartaruga gráfica”, uma representação visual que os usuários podem comandar para mover e desenhar, tornando-a particularmente atraente para o ensino de conceitos matemáticos e de programação para o público infantil.

Papert e Solomon exploram, nesse artigo, o uso de linguagens de programação, para controlar dispositivos físicos (como as “tartarugas”), e por meio disso, criar arte, música,

jogos e experimentos interativos. Dessa forma, abordam ideias inovadoras sobre o uso de computadores na educação, sugerindo uma variedade de atividades criativas e educativas, que incentivam habilidades tais como resolução de problemas, lógica e sequenciamento, que são essenciais para o pensamento computacional.

O artigo defende a utilização de computadores não apenas para tarefas convencionais, mas como ferramentas para promover o pensamento criativo e a aprendizagem prática entre os estudantes. Os autores finalizam o texto discutindo aspectos práticos, como a implementação de interfaces entre computadores e dispositivos externos e enfatizam a importância e a viabilidade de tornar a tecnologia acessível para a educação. A visão dos autores é inovadora para a época, antecipando muitas ideias que se tornariam centrais para estudiosos de tecnologias na educação.

Embora este artigo seja um precursor fundamental para a perspectiva de como os computadores podem ser usados na educação, não menciona explicitamente o termo "pensamento computacional". No entanto, os conceitos e atividades descritas no texto alinham-se estreitamente com os princípios fundamentais do conceito.

Papert e Solomon focam na ideia de usar computadores como ferramentas para resolver problemas, criar e expressar ideias de formas variadas e explorar conceitos científicos e matemáticos, ampliando a capacidade cognitiva dos alunos. Tal proposta é muito similar aos chamados "componentes" do pensamento computacional, que inclui a decomposição de problemas, o reconhecimento de padrões, a abstração e o desenvolvimento de algoritmos (pilares que veremos no próximo capítulo).

Depois veio o *Mindstorms: children, computers, and powerful ideas*

Segundo diversas pesquisas, a jornada do Pensamento Computacional começa com Papert (1980) no livro "*Mindstorm: children computers, and powerful ideas*" (Mindstorms: crianças, computadores e ideias poderosas), que explora o potencial transformador dos computadores na educação, propondo uma abordagem revolucionária em que as crianças aprendem programando ao invés de serem meramente instruídas por máquinas.

Papert, inspirado pelas teorias de Piaget, vê as crianças como construtoras ativas de seu próprio conhecimento, as quais utilizam os computadores como ferramentas para explorar e refletir sobre os processos do seu pensamento. O autor argumenta que esta abordagem não apenas ensina habilidades de programação, mas também fomenta um entendimento mais profundo e intuitivo de conceitos matemáticos e lógicos, transformando o aprendizado em uma experiência mais concreta e envolvente.

Papert (1980) destaca que algumas atitudes culturais e psicológicas podem inibir o aprendizado e propõe maneiras por meio das quais os computadores podem ajudar a superar essas barreiras. Ao incorporar a programação no ambiente educacional, por meio do Logo, por exemplo, as crianças podem abordar a matemática e outras áreas de conhecimento de maneiras mais significativas, cultivando um relacionamento positivo com o aprendizado. A ênfase de Papert em linguagens de programação intuitivas e acessíveis para crianças ressalta a necessidade de ferramentas que promovam a expressão criativa e o pensamento lógico.

Além disso, Papert introduz o conceito de "micromundos", pequenos ambientes de aprendizado dentro dos computadores que possibilitam às crianças experimentar e construir seu entendimento de forma autônoma e envolvente. Esses micromundos são incubadoras de conhecimento, que permitem a exploração de ideias complexas em um contexto gerenciável e motivador. Ao apresentar ideias desafiadoras em formatos acessíveis, os computadores ajudam a decompor conceitos difíceis em partes menores, facilitando a compreensão e o engajamento.

Papert reconhece os computadores como uma inovação tecnológica e também uma oportunidade de repensar radicalmente a educação, incentivando uma sociedade mais integrada com o pensamento computacional e mais adaptada às necessidades individuais de aprendizado.

Sendo assim, Papert (1980) foi pioneiro ao explorar a ideia do que hoje conhecemos como "Pensamento Computacional", embora ele não tenha formalmente definido o termo. Papert defende que aprender a se comunicar com um computador, especialmente através da programação, poderia transformar radicalmente a maneira como as crianças pensam e aprendem, destacando assim, a profunda influência que a programação e a interação com computadores pode alcançar no processo de aprendizagem e no desenvolvimento cognitivo.

No cerne desta abordagem está a crença de que aprender a programar um computador é uma forma de desenvolver o pensamento procedural, ou seja, a habilidade de estruturar e organizar processos de pensamento de maneira lógica e sequencial. Papert sugere que essa forma de pensar, semelhante à maneira como um computador processa informações, pode ser uma ferramenta intelectual poderosa. Ao aprender a "falar" com um computador, as crianças dominam um conjunto de habilidades técnicas,assim como começam a pensar de maneiras que ajudam a desenvolver a capacidade de resolução de problemas e o raciocínio lógico.

Papert não definiu explicitamente o termo "pensamento computacional", mas usou-o para descrever como a presença do computador pode alterar a cognição. A ideia é que a

interação com os computadores muda a maneira como pensamos, mesmo quando estamos longe das máquinas. Essa influência vai além do aprendizado de conceitos de programação e chega a outras áreas do conhecimento, transformando a maneira como as crianças resolvem problemas e entendem o mundo.

O pensamento computacional, conforme imaginado por Papert, ocorre quando as crianças usam a programação para expressar ideias, resolver problemas e entender conceitos que podem ser abstratos ou inacessíveis de outra forma. Essa visão pavimentou o caminho para uma compreensão mais ampla do Pensamento Computacional, que foi posteriormente popularizada por Jeannette Wing.

E então, tornou-se o Pensamento Computacional

O termo “Pensamento Computacional” ganhou grande repercussão com o artigo “Computational Thinking”, de Jeannette M. Wing, em 2006. No texto, Wing (2006) discute a importância e a natureza do pensamento computacional, defendendo-o como uma habilidade essencial para todos, não somente para cientistas da computação. A autora propõe que, além de ler, escrever e calcular, o pensamento computacional deveria integrar as habilidades analíticas fundamentais de cada pessoa.

Conforme Wing (2006), o pensamento computacional define-se pela habilidade de resolver problemas, projetar sistemas e compreender o comportamento humano, empregando conceitos básicos da ciência da computação. Isso abrange diversas ferramentas mentais que refletem a vastidão deste campo de estudos. .

O pensamento computacional destaca-se na solução de problemas por uma abordagem sistemática, que inclui decompor problemas complexos, identificar padrões e desenvolver soluções eficientes. Essa metodologia baseia-se no uso de abstrações para simplificar e compreender problemas, além de enfatizar a criação e aplicação de algoritmos lógicos e sequenciais. A abordagem também valoriza a manipulação e a interpretação estruturada de dados e informações, recorrendo a tecnologias e ferramentas computacionais como programação, simulação e modelagem.

Wing (2006) realça que o pensamento computacional é aplicável em várias áreas, incluindo biologia, medicina, economia, engenharia e ciências sociais. É utilizado na análise de dados genômicos, diagnósticos médicos, modelagem de sistemas econômicos, design de estruturas complexas e na análise de interações sociais. Essa abordagem permite solucionar problemas complexos, analisar grandes volumes de dados e tomar decisões informadas em diversos campos, fomentando a inovação e o progresso em múltiplas disciplinas. Wing (2006) argumenta que o pensamento computacional vai além da programação; trata-se de uma maneira de pensar que complementa o raciocínio matemático e de engenharia. Assim,

ela incentiva a educação em pensamento computacional desde a infância, não apenas para futuros cientistas da computação, mas para todos, enfatizando que um diploma em ciência da computação pode abrir portas em várias áreas, que não se limitam à tecnologia.

Em 2010, no artigo *Computational thinking*: what and why? (Pensamento computacional: o quê e por quê?) Wing destaca o Pensamento Computacional como uma habilidade essencial no século XXI, como uma nova literacia para todos, que pode ser aplicada na resolução de problemas do mundo real e na geração de novas ideias e soluções em diversas áreas profissionais. Wing destaca ainda como o Pensamento Computacional pode ser aplicado a partir de exemplos reais, como:

- Iniciativas para promover o Pensamento Computacional em escolas de ensino fundamental e médio, visando preparar estudantes para os desafios do século XXI.
- Projetos e programas educacionais que buscam desenvolver competências de Pensamento Computacional em estudantes e professores.
- A influência do Pensamento Computacional em diversas disciplinas científicas e de engenharia, destacando a importância da computação como o terceiro pilar da ciência, juntamente com teoria e experimentação.

Esses exemplos ilustram como o Pensamento Computacional foi integrado em diferentes contextos educacionais e profissionais, demonstrando sua relevância e impacto em diversas áreas da sociedade.

Pensamento Computacional e as habilidades do futuro

Desde a definição do conceito de Pensamento Computacional (PC) por Jeannette M. Wing em 2006, esta habilidade vem sendo reconhecida como fundamental para o século XXI, não apenas no domínio da tecnologia e informática, mas em diversas esferas da vida cotidiana. Wing (2006) enfatiza que o PC capacitaria as pessoas a enfrentar desafios complexos da era digital e aproveitar as oportunidades oferecidas pela tecnologia, contribuindo significativamente para o exercício da cidadania e para o enfrentamento dos desafios sociais contemporâneos.

O PC desenvolve habilidades essenciais para a tomada de decisões informadas e responsáveis, como análise crítica e solução de problemas. Em um mundo saturado de informações, frequentemente contraditórias ou falsas, a habilidade de decompor argumentos complexos, identificar padrões e tendências e avaliar a veracidade e a relevância das informações torna-se crucial. Essas capacidades são essenciais para exercer a cidadania de maneira responsável e informada.

Na sociedade contemporânea, questões como privacidade de dados, segurança cibernética, desinformação e automação impactam diretamente a vida dos cidadãos. A compreensão de conceitos básicos do PC permite aos indivíduos uma nova possibilidade para entender melhor essas questões, participar de discussões informadas e contribuir para a formação de políticas públicas e sociais. Além disso, ao promover o acesso e a compreensão do PC entre todos os grupos sociais, contribui-se para reduzir a desigualdade digital e garantir oportunidades iguais de participação na sociedade. Isso é especialmente importante para grupos historicamente marginalizados, que podem utilizar o PC como uma ferramenta para ampliar suas vozes e lutar por seus direitos e interesses.

O PC também encoraja o pensamento crítico e inovador, uma habilidade muito importante para enfrentar e resolver problemas sociais complexos. Cidadãos que se abrem para essa forma de pensar podem abordar questões sociais de maneiras criativas e eficazes, contribuindo para o desenvolvimento de comunidades mais resilientes e adaptáveis.

O Fórum Econômico Mundial, no relatório "*Future of Jobs Report 2023*", apresenta uma visão abrangente das habilidades que serão demandadas no mercado de trabalho nos próximos anos, em meio a transformações aceleradas pelo avanço tecnológico, mudanças econômicas e geopolíticas e pressões sociais e ambientais. De acordo o com o último relatório do fórum (2023), estas habilidades são cruciais para enfrentar os desafios e aproveitar as oportunidades desta nova era de transformações, sendo elas:

- **Pensamento analítico e inovação:** capacidade de compreender dados complexos, identificar padrões, fazer conexões e propor novas soluções e abordagens.
- **Aprendizado ativo e estratégias de aprendizagem:** habilidade para aprender de maneira contínua e adaptar-se rapidamente às novas informações.
- **Resolução de problemas complexos:** aptidão para resolver desafios novos e não familiares de maneira eficiente e inovadora.
- **Pensamento crítico e análise:** avaliar informações de fontes diversas, discernir entre fatos e opiniões, e tomar decisões informadas.
- **Criatividade, originalidade e iniciativa:** capacidade de pensar de forma criativa, propor ideias inovadoras e tomar a iniciativa em situações variadas.
- **Liderança e influência social:** habilidade para liderar, inspirar e influenciar positivamente os outros.
- **Uso, monitoramento e controle de tecnologia:** competência em utilizar e adaptar-se a novas tecnologias.

- ***Design* e programação de tecnologia:** habilidades relacionadas ao desenvolvimento e design de tecnologias.

A relação entre as habilidades destacadas pelo Fórum Econômico Mundial e o Pensamento Computacional é intrínseca e abrangente. Por exemplo, o **pensamento analítico e a inovação**, habilidades essenciais no relatório, são fundamentais no Pensamento Computacional, que envolve a decomposição de problemas em partes gerenciáveis, a análise de dados e o desenvolvimento de soluções inovadoras.

Quando se trata de **resolução de problemas complexos**, o Pensamento Computacional oferece uma abordagem estruturada e lógica, a qual permite decompor problemas, identificar padrões e elaborar soluções algorítmicas, habilidades indispensáveis na resolução de problemas complexos do mundo real.

O **pensamento crítico e a análise** também se beneficiam do Pensamento Computacional, pois este último promove uma maneira lógica e sistemática de analisar problemas e dados, que é a essência do pensamento crítico. Ao aplicar o Pensamento Computacional, os indivíduos podem avaliar informações de forma mais eficaz, distinguindo fatos de opiniões e tomando decisões mais informadas.

No que diz respeito a **criatividade, originalidade e iniciativa**, o Pensamento Computacional tem um papel significativo. A abstração, um dos seus pilares, estimula a criatividade e a originalidade ao permitir que as pessoas pensem além dos detalhes imediatos e concebam soluções inovadoras. Este aspecto é crucial em um ambiente de trabalho onde a inovação contínua é valorizada.

Em relação ao **uso, monitoramento e controle de tecnologia**, o Pensamento Computacional é essencial, uma vez que, não só facilita a compreensão de como as tecnologias funcionam, mas também como podem ser controladas e otimizadas. Este entendimento é fundamental para o uso eficiente da tecnologia em diversos contextos profissionais.

Por fim, o Pensamento Computacional é a base para o **Design e a Programação de Tecnologia**, pois fornece os conceitos fundamentais para a criação e inovação tecnológica. Ao aplicar os princípios do Pensamento Computacional, os profissionais podem desenvolver tecnologias mais eficazes e inovadoras.

O Pensamento Computacional fortalece as habilidades identificadas, não apenas preparando os profissionais para os desafios futuros, mas também capacitando-os para serem inovadores, adaptáveis e eficientes na era digital.

Portanto, o estímulo ao desenvolvimento do PC por meio de ações educacionais e profissionais é uma estratégia eficaz para preparar as sociedades para os desafios e oportuni-

dades do futuro. Além de desenvolver habilidades individuais, trata-se de uma ferramenta poderosa para nivelar o campo em termos de oportunidades econômicas e empregatícias. Ao tornar essas habilidades mais acessíveis e integradas em sistemas educacionais e profissionais, é possível combater a desigualdade, impulsionar a inovação e fomentar o crescimento mais inclusivo e sustentável.

O PC é uma competência essencial para o exercício efetivo e consciente da cidadania na era digital, contribuindo para uma sociedade mais informada, justa e democrática.

Pensamento Computacional no Brasil e no mundo

O Pensamento Computacional (PC) tem se destacado no cenário educacional global e no Brasil, refletindo a evolução notável da computação. Esta transformação impactou significativamente quase todos os aspectos da vida cotidiana, inclusive a maneira como interagimos, acessamos informações e nos entretemos. A digitalização abrangeu quase todos os setores, tornando a computação essencial em suas operações.

Nas décadas de 1980 e 1990, houve uma expansão significativa do uso de computadores, estendendo-se para a educação e o entretenimento. O advento da Internet na década de 1990 mudou radicalmente o compartilhamento e o processamento de informações. Com isso, os avanços em *hardware* e *software* possibilitaram dispositivos mais potentes e acessíveis, levando-nos à era da computação ubíqua, com dispositivos inteligentes em todos os lugares (Brackmann, 2017).

Esses avanços sublinharam a necessidade de habilidades computacionais básicas para todos, e não apenas para profissionais de TI. Consequentemente, programas educacionais em todo o mundo passaram a enfatizar essas habilidades, incluindo computação e programação na educação básica.

A integração do PC na educação tem sido uma tendência global, com países como Espanha, Reino Unido, Estados Unidos, Estônia e Finlândia liderando iniciativas governamentais e privadas. No Reino Unido, a programação e os conceitos de PC fazem parte do currículo escolar obrigatório desde a pré-escola. Nos Estados Unidos, há um foco em integrar PC através do apoio governamental e parcerias com empresas de tecnologia. A Espanha e a Estônia desenvolvem estratégias para incorporar o PC nos currículos, enquanto a Finlândia integra o PC como disciplina obrigatória desde as séries iniciais (Brackmann, 2017; Oliveira, 2022).

Essas iniciativas visam preparar os estudantes para um mercado de trabalho baseado em tecnologia e capacitá-los para compreender e moldar o mundo tecnológico. O ensino de PC visa preparar futuros profissionais em tecnologia, e desenvolver o pensamento crítico e analítico necessário ao mundo digital.

No Brasil, a integração do PC na educação básica tem se mostrado prioritária. Há uma proliferação de projetos e pesquisas, em colaboração com universidades, para explorar metodologias de ensino de PC. Essas iniciativas buscam não melhorar as habilidades computacionais e preparar os jovens estudantes para um futuro digital.

Segundo Oliveira (2022), o Brasil enfrenta desafios únicos no ensino de computação, como a escassez de profissionais de TI e a necessidade de letramento digital. A falta de representatividade no setor de TI, especialmente entre mulheres e minorias raciais, destaca a necessidade de políticas educacionais inclusivas.

Há iniciativas que visam garantir o acesso igualitário às ferramentas e conhecimentos necessários na era digital. A Política Nacional de Educação Digital (PNED), instituída pela Lei nº 14.533 de 2023, representa um marco ao estabelecer uma estrutura para a incorporação de práticas digitais na educação, contemplando diferentes eixos estruturantes, tais como: Inclusão Digital, Educação Digital Escolar, Capacitação e Especialização Digital, e Pesquisa e Desenvolvimento em Tecnologias da Informação e Comunicação (TICs). Esse arcabouço legal objetiva potencializar padrões e incrementar os resultados das políticas públicas relacionadas ao acesso a recursos, ferramentas e práticas digitais. Esses esforços indicam o compromisso do Brasil com o avanço da educação tecnológica, visando a equidade social e a preparação para um mundo globalizado.

A Base Nacional Comum Curricular (BNCC) enfatiza a familiarização dos educadores com as Tecnologias Digitais da Informação e Comunicação (TDICs), propondo uma abordagem integrada em diversas disciplinas, entendendo a importância de habilidades e competências essenciais para o século 21, as quais incluem a computação.

O ensino de Computação na Educação Básica no Brasil tem passado por uma transformação significativa, buscando alinhar a educação tecnológica às demandas contemporâneas de uma sociedade cada vez mais digitalizada. Esta mudança é evidenciada pela implementação de duas legislações-chave: a Resolução CNE/CEB nº 1, de 4 de outubro de 2022, e o Parecer CNE/CEB nº 2/2022. Ambos os documentos estabelecem normas para o ensino de Computação na Educação Básica, complementando a Base Nacional Comum Curricular (BNCC) e demarcando um avanço importante na política educacional brasileira. Eles demonstram a necessidade de preparar os estudantes para os desafios da atualidade e representam um passo importante na incorporação da computação como componente curricular.

O Parecer CNE/CEB nº 2/2022 destaca o esforço contínuo para estabelecer normas específicas sobre a computação na Educação Básica brasileira. Este documento é um marco que reflete uma longa trajetória de experimentações e desenvolvimentos no campo do ensino da computação no Brasil, que remonta à década de 1970.

A história do ensino de computação no país é marcada por várias iniciativas pioneiras. Um exemplo significativo foi o Projeto EDUCOM, um esforço colaborativo que buscou integrar a informática na educação. Este projeto se concentrou em investigar como os computadores poderiam ser utilizados como ferramentas pedagógicas, explorando diferentes abordagens educativas. O projeto não apenas abriu caminho para a utilização de tecnologias na sala de aula, mas também serviu de base para o desenvolvimento de futuras políticas educacionais relacionadas à tecnologia.

Outra iniciativa importante foi o Programa Nacional de Informática Educativa (Proninfe), que visava desenvolver a informática educativa no Brasil. O Proninfe colocou ênfase na formação de professores e na criação de materiais didáticos apropriados, com o objetivo de disseminar o uso pedagógico das tecnologias de informação nas escolas brasileiras.

Essas experiências iniciais, embora enfrentando desafios e limitações, foram fundamentais para a formação de uma base sólida na cultura de ensino de computação no Brasil. Elas pavimentaram o caminho para a implementação de políticas mais estruturadas e consolidadas no século XXI, como as delineadas no Parecer CNE/CEB Nº2/2022. Este documento não só reconhece a importância do pensamento computacional e das habilidades digitais na educação moderna, mas também estabelece diretrizes claras para a integração efetiva da computação nos currículos escolares, alinhando a educação brasileira com as demandas globais da era digital.

A Sociedade Brasileira de Computação (SBC) tem desempenhado um papel importante no desenvolvimento das diretrizes para a inserção da computação na Educação Básica, como evidenciado no anexo ao Parecer CNE/CEB nº 2/2022. Este anexo detalha competências essenciais em três eixos principais: Pensamento Computacional, Mundo Digital e Cultura Digital, abrangendo desde a Educação Infantil até o Ensino Médio.

1. Pensamento Computacional: Este eixo enfatiza o desenvolvimento de habilidades relacionadas à lógica, à resolução de problemas e ao pensamento crítico através de conceitos de computação. Envolve ensinar aos alunos a analisar e decompor problemas complexos, desenvolver algoritmos e soluções e entender os fundamentos da programação e da automação. Este tipo de pensamento auxilia os estudantes não apenas no entendimento da tecnologia, mas também em diversas áreas do conhecimento, estimulando a criatividade e a inovação.

2. Mundo Digital: Focado no entendimento e na utilização de dispositivos digitais e suas aplicações, este eixo cobre conhecimentos sobre hardware, como computadores, tablets e celulares, e software, incluindo sistemas operacionais, aplicativos e a internet. Ele visa capacitar os estudantes a navegar com eficácia no ambiente

digital, entendendo sua estrutura e funcionalidades. Além disso, busca promover a consciência sobre questões de segurança digital e ética no uso da tecnologia.

3. Cultura Digital: Este eixo lida com a compreensão do impacto da revolução digital na sociedade. Visa desenvolver uma postura crítica e reflexiva sobre os conteúdos e as práticas digitais. Além disso, enfatiza a importância de uma participação consciente e ética nas redes sociais e outras plataformas online, promovendo uma compreensão sobre direitos digitais, privacidade e proteção de dados.

Esses três eixos, propostos pela SBC e incorporados ao Parecer CNE/CEB nº 2/2022, representam uma abordagem abrangente para integrar a computação na educação, garantindo que os estudantes estejam bem equipados para enfrentar os desafios de um mundo cada vez mais tecnológico e interconectado.

Capítulo 2
Pilares do Pensamento Computacional

O Pensamento Computacional é uma abordagem sistemática para solucionar problemas, estruturada em quatro pilares principais, definidos por Wing (2006), que são: Decomposição, Reconhecimento de Padrões, Abstração e Algoritmos. De acordo com Brackmann (2017), esses pilares são interdependentes e desempenham um papel vital no processo de resolução de problemas complexos, na formulação de soluções computacionalmente viáveis em uma vasta gama de disciplinas e contextos profissionais, orientando na criação de soluções eficientes e inovadoras.

Abstração

A abstração é a capacidade de concentrar-se nos aspectos essenciais de um problema, ignorando detalhes desnecessários. No contexto do pensamento computacional, a abstração envolve identificar padrões e conceitos-chave, simplificando a representação de um problema para facilitar sua compreensão e resolução. A abstração permite lidar com a complexidade ao ignorar detalhes irrelevantes para se concentrar no que é verdadeiramente importante. Winf (2006) explica que “abstração dá-nos o poder de escala e lidar com a complexidade”.

Vejamos como esse conceito é aplicado em algumas áreas:

- **Arquitetura:** na arquitetura, a abstração é fundamental para transformar uma visão ou conceito em um design funcional e esteticamente agradável. Aqui, a abstração começa com a identificação do propósito principal do edifício: sua funcionalidade, o público-alvo, o contexto ambiental e urbano. Por exemplo, ao projetar uma biblioteca, o arquiteto concentra-se em aspectos como acessibilidade, espaço para coleções de livros, áreas de leitura e estudo, ignorando, nesta fase, detalhes menores como o design de interiores ou o tipo de mobiliário. Esta abordagem permite que o arquiteto criem estruturas que não apenas atendem às necessidades práticas, mas também enfatizem valores estéticos e culturais, sem se perderem em pormenores iniciais

- **Direito:** no campo do direito, a abstração é igualmente crucial. Advogados e juristas usam a abstração para identificar os principais elementos legais em um caso, como a legislação aplicável, os fatos relevantes e os argumentos jurídicos centrais. Por exemplo, em um caso de disputa contratual, o foco estará nos termos do contrato, nas ações das partes envolvidas e na legislação pertinente. Detalhes como as interações informais anteriores entre as partes ou questões secundárias, que não im-

pactam diretamente o resultado legal, são abstraídos para criar uma argumentação clara e focada. Essa habilidade de abstrair permite aos profissionais do direito formular argumentos mais persuasivos e encontrar soluções eficazes para disputas legais.

No dia a dia, a abstração é uma ferramenta mental que usamos frequentemente, muitas vezes sem sequer perceber. Alguns exemplos:

- **Planejamento de uma viagem:** começamos por identificar os elementos mais críticos, como o destino, a duração da viagem e o orçamento. Inicialmente, detalhes menos importantes, como a escolha de restaurantes específicos ou a seleção de roupas, são deixados de lado. O foco está em questões mais amplas e fundamentais.

- **Preparo de um jantar para amigos:** o processo de abstração é usado para simplificar e tornar mais eficientes as decisões. Inicialmente, selecionamos o *menu* focando nas preferências alimentares gerais, deixando de lado detalhes como marcas específicas de ingredientes.

Wing (2006) considera a abstração como o pilar mais importante do Pensamento Computacional, pois além do que já foi mencionado anteriormente, a abstração ainda permite:

- **Foco nos conceitos-chave:** ao abstrair um problema, os indivíduos podem identificar e focar nos conceitos-chave e padrões subjacentes, destacando as informações mais relevantes e significativas. Isso ajuda a extrair insights importantes e a tomar decisões informadas com base em dados essenciais. Exemplo: um professor pode usar a abstração para ensinar um conceito matemático como a fração. Em vez de se prender a exemplos numéricos específicos, aborda a ideia geral de divisão de um todo em partes, facilitando a compreensão dos alunos e a aplicação desse conceito em diferentes contextos.

- **Generalização e reutilização:** a abstração permite generalizar soluções para problemas específicos, tornando-as aplicáveis a uma variedade de contextos e cenários. Além disso, a capacidade de abstrair conceitos e padrões facilita a reutilização de soluções em diferentes situações, promovendo a eficiência e a produtividade. Exemplo: no desenvolvimento de software, a abstração é usada para criar bibliotecas de códigos e funções que podem ser reutilizadas em diferentes projetos como uma função para calcular a média de uma lista de números pode ser abstraída e reutilizada em diversos programas, economizando tempo e esforço.

- **Facilitação da comunicação:** a abstração também desempenha um papel relevante para a comunicação, pois permite expressar ideias complexas de forma clara e concisa. Ao abstrair informações e conceitos, os comunicadores podem transmitir mensagens de maneira eficaz e compreensível para diferentes públicos. Exemplo:

um gerente pode abstrair um plano de negócios complexo em metas e objetivos claros, facilitando a compreensão e o alinhamento da equipe.

- **Promoção da criatividade e inovação:** a capacidade de abstrair e pensar de forma conceitual estimula a criatividade e a inovação, permitindo aos indivíduos explorar novas ideias, desenvolver soluções originais e abordar desafios de maneira inventiva e eficaz. Exemplo: ao criar um novo produto, um *designer* pode abstrair o conceito de "conforto" e explorar diversas formas e materiais, levando a inovações que atendam a essa necessidade de maneira única.

A abstração é central no pensamento computacional porque permite simplificar a complexidade, identificar padrões essenciais, generalizar soluções, facilitar a comunicação e promover a criatividade. Ao abstrair conceitos e informações, os indivíduos podem enfrentar desafios complexos de forma estruturada, eficiente e inovadora.

Decomposição

A decomposição é o processo de dividir um problema complexo em componentes menores e mais gerenciáveis. Como Wing (2006) observa, a decomposição é um passo crucial no processo de resolução de problemas, pois permite que nos concentremos em partes menores de um problema maior. A decomposição ajuda a simplificar a complexidade e facilita a resolução de problemas de maneira estruturada.

Seguem alguns exemplos de como podemos utilizar a decomposição no dia a dia seja no trabalho, na ciência ou em áreas técnicas:

- **Planejamento de uma festa:** ao organizar uma festa, pode-se dividir o processo em várias etapas, como a definição do tema, a preparação da lista de convidados, a organização do menu, a decoração e o entretenimento. Cada um desses elementos representa um subproblema menor, tornando o processo de planejamento mais gerenciável e menos sobrecarregado.

- **Gerenciamento de finanças pessoais:** ao administrar um orçamento doméstico, pode-se decompor o problema em categorias menores como aluguel ou hipoteca, contas de serviços públicos, gastos com alimentação, lazer e poupança. Isso facilita a compreensão de como o dinheiro está sendo gasto e ajuda na identificação de áreas para potenciais economias.

- **Pesquisa em biologia:** ao estudar um ecossistema complexo, um biólogo pode decompor o sistema em componentes menores como populações de diferentes espécies, interações entre elas, e fatores abióticos. Isso permite um entendimento

mais profundo de cada elemento e como eles interagem no ecossistema como um todo.

- **Diagnóstico médico:** quando um médico enfrenta um caso clínico complexo, ele utiliza a decomposição para analisar os sintomas do paciente. Em vez de tentar diagnosticar uma doença com base em uma visão geral vaga, o médico divide o problema em sintomas específicos, histórico médico, resultados de exames e fatores de risco. Por exemplo, no diagnóstico de uma doença autoimune, o médico pode considerar separadamente os sintomas como fadiga, dor articular e resultados de exames de sangue. Cada um desses elementos fornece pistas que, quando combinadas, levam a um diagnóstico mais preciso.

- **Análise de fenômenos sociais:** sociólogos frequentemente enfrentam a tarefa de entender fenômenos sociais complexos, como desigualdade ou mudanças culturais. Para isso, eles podem decompor um fenômeno em várias dimensões menores, como fatores econômicos, históricos, culturais e políticos. Por exemplo, na análise da desigualdade social, um sociólogo pode examinar separadamente questões de acesso à educação, diferenças de renda, políticas governamentais e dinâmicas familiares. Ao estudar cada aspecto individualmente, torna-se mais fácil compreender o fenômeno como um todo.

- **Preparo de alimentos:** ao preparar um prato complicado, um cozinheiro pode dividir a receita em várias etapas menores, como preparação dos ingredientes, cozimento e apresentação. isso ajuda a gerenciar o tempo e os recursos de maneira mais eficiente.

- **Corte de cabelo:** ao realizar um corte de cabelo complexo, o cabeleireiro pode dividir o processo em etapas como lavar, cortar, tingir e finalizar. Isso permite focar em cada parte individualmente, assegurando um resultado final de alta qualidade.

- **Desenvolvimento de *software*:** no processo de desenvolvimento de software, a decomposição é usada para dividir um projeto grande em módulos ou componentes menores, como interface de usuário, processamento de dados e gestão de banco de dados. Isso facilita a gestão do projeto, permitindo que diferentes equipes trabalhem em componentes distintos simultaneamente.

- **Solução de *bugs*:** ao enfrentar um *bug* complicado em um código, um programador pode decompor o problema analisando separadamente as diferentes partes do código para localizar a origem do erro.

Em todas essas áreas, a decomposição permite abordar problemas de maneira estruturada, reduzindo a complexidade e tornando tarefas abrangentes mais gerenciáveis e menos

intimidadoras. Ao dividir um problema grande em partes menores, as soluções tornam-se mais claras e o processo de resolução mais eficiente.

Reconhecimento de Padrões

O Reconhecimento de Padrões desempenha um papel fundamental no Pensamento Computacional pois é essencial para identificar regularidades e estruturas tanto em conjuntos de dados quanto em problemas variados. Esta habilidade permite a extração de informações significativas, a previsão de comportamentos futuros e a tomada de decisões informadas, cruciais para a resolução eficiente de problemas.

Quando realizamos a decomposição de um problema complexo, frequentemente encontramos padrões entre os subproblemas gerados. Estes padrões, que são similaridades ou características comuns a alguns problemas, podem ser explorados para alcançar soluções mais eficientes.

O Reconhecimento de Padrões, conforme explicado por Brackmann (2017), consiste em identificar similaridades e padrões com o objetivo de solucionar problemas complexos de forma mais eficiente. Esse processo envolve a busca por elementos iguais ou muito similares em cada problema, o que na literatura também pode estar associado ao termo “generalização”.

Esse reconhecimento permite resolver problemas rapidamente, aplicando soluções previamente definidas e baseadas em experiências anteriores. Questões como “Esse problema é similar a outro que já resolvi?” ou “Como ele é diferente?” são fundamentais nessa etapa, pois ajudam a definir os dados, processos e estratégias a serem utilizados na solução do problema. Algoritmos projetados para resolver um problema específico podem ser adaptados em uma série de de situações similares, aplicando uma solução generalizada.

O reconhecimento de padrões pode ser aplicado em diversos cenários, tais como:

- **Economia:** para prever tendências de mercado e comportamentos de consumidores, o que é crucial para o planejamento estratégico e políticas econômicas. Na meteorologia, padrões climáticos são analisados para elaborar previsões do tempo mais precisas, beneficiando áreas como agricultura, segurança pública e outras que dependem das condições climáticas.

- **Biologia:** para entender complexos processos biológicos, como padrões de migração de espécies ou a disseminação de doenças. Esta aplicação é essencial para avanços em pesquisas e para o desenvolvimento de estratégias de conservação e saúde pública.

- **Medicina:** no diagnóstico de doenças para identificar doenças com base em sintomas comuns. Por exemplo, padrões específicos nos sintomas podem indicar uma gripe ou uma infecção bacteriana, ajudando no diagnóstico e tratamento adequado.

- **Educação:** na avaliação os professores podem reconhecer padrões no desempenho dos alunos, como dificuldades em certos tipos de questões ou tópicos, e usar essas informações para adaptar métodos de ensino ou oferecer suporte adicional.

- **Trânsito:** no gerenciamento de fluxo de tráfego, os padrões são analisados para otimizar os sinais de trânsito e reduzir engarrafamentos, especialmente em horas de pico.

- **Cozinha:** na preparação de doces como beijinho e brigadeiro, observamos um padrão comum no processo de cozimento, apesar de diferenças nos ingredientes.

O reconhecimento de padrões permite simplificar a solução de problemas e replicar esta solução em cada subproblema que apresente semelhanças. Quanto mais padrões são identificados, mais dinâmica e rápida se torna a solução do problema maior.

Algoritmos

Os algoritmos são sequências de passos e/ou regras bem definidos e ordenados que descrevem como resolver um problema ou executar uma tarefa. No contexto do pensamento computacional, os algoritmos são essenciais para a resolução sistemática de problemas, a automação de processos e a criação de soluções eficientes. Compreender e desenvolver algoritmos é fundamental para a aplicação prática do pensamento computacional em diversas áreas.

Algoritmos estão presentes em muitas situações do dia a dia, muitas vezes de maneiras que nem percebemos. Eles nos ajudam a organizar tarefas e a tomar decisões com base em uma série de etapas lógicas. Aqui estão alguns exemplos comuns:

- **Preparação de receitas na cozinha:** Ao seguir uma receita, estamos seguindo um algoritmo. A receita fornece uma lista de ingredientes (dados de entrada) e uma série de etapas específicas (processo) para produzir um prato (resultado). A ordem e o método de mistura, cozimento e tempero são exemplos de algoritmos culinários.

- **Rotinas matinais:** Nossas rotinas matinais podem ser vistas como algoritmos. Por exemplo, você pode ter uma sequência de ações como acordar, escovar os dentes, tomar banho, se vestir, tomar café e sair para o trabalho. Cada passo segue uma ordem lógica e leva ao objetivo de estar pronto para o dia.

- **Navegação com GPS:** Quando usamos um GPS para chegar a um destino, o sistema utiliza algoritmos para calcular a melhor rota, considerando distância,

tráfego e outros fatores. O algoritmo analisa os dados e oferece uma rota eficiente.

- **Decisões de compras:** Muitas pessoas seguem um algoritmo informal ao fazer compras, como verificar primeiro itens essenciais, comparar preços ou procurar promoções, e então tomar uma decisão de compra com base nessa análise.

- **Jogos e quebra-cabeças:** Resolver um quebra-cabeça ou jogar certos jogos implica seguir algoritmos. Por exemplo, em jogos de xadrez, os jogadores utilizam estratégias (algoritmos) para prever movimentos e reagir de acordo.

- **Exercícios físicos:** programas de treino físico seguem algoritmos, com exercícios ordenados em uma sequência específica para maximizar os benefícios do treino.

- **Organização doméstica:** ao limpar ou organizar uma casa, muitas pessoas seguem um conjunto de regras ou etapas - primeiro arrumar as áreas comuns, depois os quartos, seguido pela cozinha, e assim por diante.

No âmbito do trabalho, algoritmos também estão presentes em diversas áreas:

- **Enfermagem:** na triagem de pacientes, os algoritmos são utilizados para determinar a urgência dos cuidados médicos necessários. Baseando-se em sinais vitais, sintomas e informações do paciente, a enfermagem pode priorizar o atendimento de acordo com a gravidade do caso.

- **Agricultura:** os algoritmos são empregados na gestão de cultivos para aumentar a eficiência e produtividade agrícola. Eles utilizam dados climáticos para prever necessidades de irrigação e melhores momentos para plantio e colheita. Além disso, monitoram a saúde do solo e das plantas, recomendando a quantidade ideal de fertilizantes e nutrientes. Esses algoritmos também otimizam a irrigação, reduzindo o desperdício de água e possibilitando uma agricultura mais sustentável e eficiente.

- **Automobilismo:** os sistemas de navegação utilizam algoritmos sofisticados para fornecer rotas otimizadas aos motoristas. Eles consideram fatores como tráfego em tempo real, condições das estradas, condições climáticas e preferências do usuário. Esses algoritmos permitem aos motoristas evitar congestionamentos, encontrar rotas mais seguras e personalizar suas viagens de acordo com suas preferências, resultando em uma experiência de direção mais eficiente e segura.

Em todas essas situações, os algoritmos nos ajudam a estruturar atividades, tomar decisões informadas e realizar tarefas de maneira eficiente. Eles são ferramentas fundamentais para a organização e a execução lógica de ações no nosso cotidiano.

Capítulo 3
Pensamento Computacional com jogos de lógica

Nos capítulos anteriores, entendemos que o pensamento computacional é uma habilidade fundamental não apenas para profissionais da área de tecnologia, mas para todos que desejam resolver problemas de forma eficiente e criativa. Uma maneira divertida e eficaz de desenvolver essa habilidade é através de jogos e quebra-cabeças que desafiam nossa lógica e capacidade de encontrar soluções.

Neste capítulo, apresentaremos diversos jogos de lógica e quebra-cabeças que promovem o desenvolvimento do pensamento computacional. Esses jogos são divertidos e educativos, além de auxiliar tanto os leitores em geral quanto os educadores a introduzir conceitos de pensamento computacional de forma prática e envolvente em suas vidas ou salas de aula.

Physics Drop

Physics Drop é um jogo bastante popular que explora elementos de física e de lógica. É um quebra-cabeça que desafia os jogadores a desenhar linhas e formas com o objetivo de guiar uma pequena bola vermelha até um determinado ponto, que geralmente é um balde. O jogo utiliza princípios básicos da física, como gravidade e impulso, e requer que os jogadores pensem de maneira criativa e estratégica para superar obstáculos e completar cada nível.

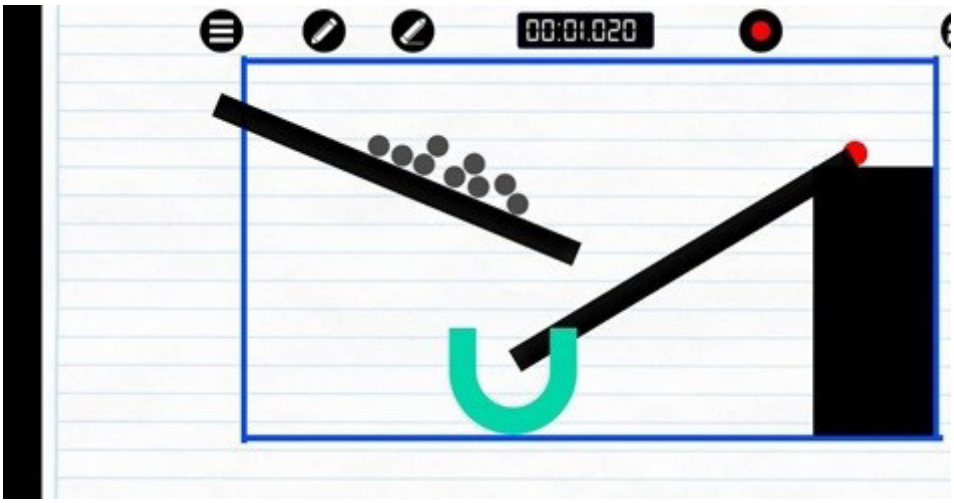
No *Physics Drop* os jogadores precisam levar em conta a direção e a força que aplicam ao desenhar as linhas, além de prever essas linhas podem interagir com o ambiente do jogo para mover a bola de forma eficaz. Isso envolve a aplicação prática de conceitos de física de uma maneira interativa e visual.

Por ser um jogo que estimula o pensamento crítico, a resolução de problemas e a criatividade, *Physics Drop* é uma ferramenta útil para o desenvolvimento do pensamento computacional, especialmente no que se refere à lógica e ao planejamento estratégico.

Ao jogar *Physics Drop*, os quatro pilares do Pensamento Computacional - Decomposição, Abstração, Reconhecimento de Padrões e Algoritmos - podem ser aplicados, pois cada desafio exige uma combinação de análise, estratégia e criatividade. Para exemplificar, vamos utilizar o desafio 15 do jogo na sua versão disponível na Play Store no endereço:

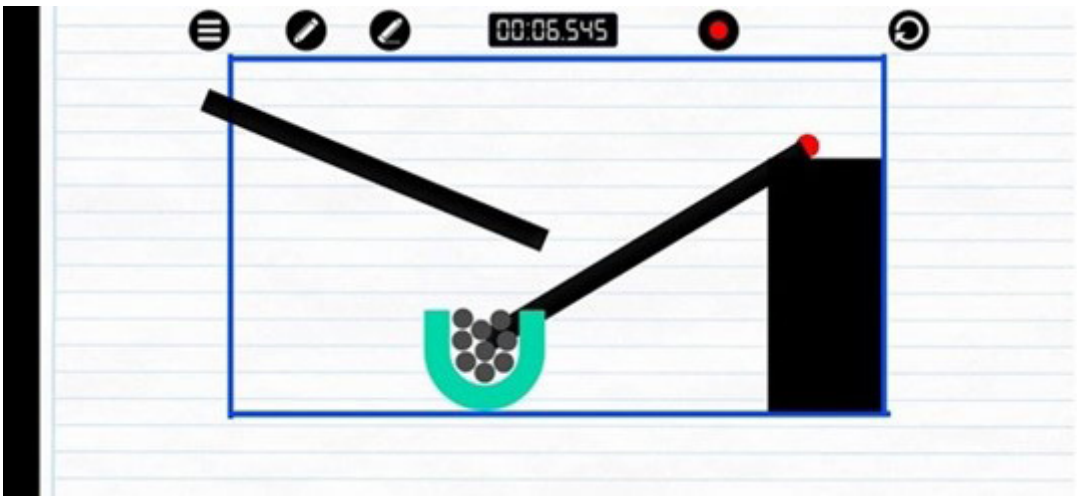
https://play.google.com/store/apps/details?id=com.dreamed.physicsdrop.

Figura 1. Visão inicial do Desafio 15: a bola vermelha deve alcançar o balde verde



Fonte: Captura de tela *Physics Drop*

Figura 2. Momento crítico onde as bolas cinzas começam a cair, potencialmente complicando o desafio



Fonte: Captura de tela do *Physics Drop*

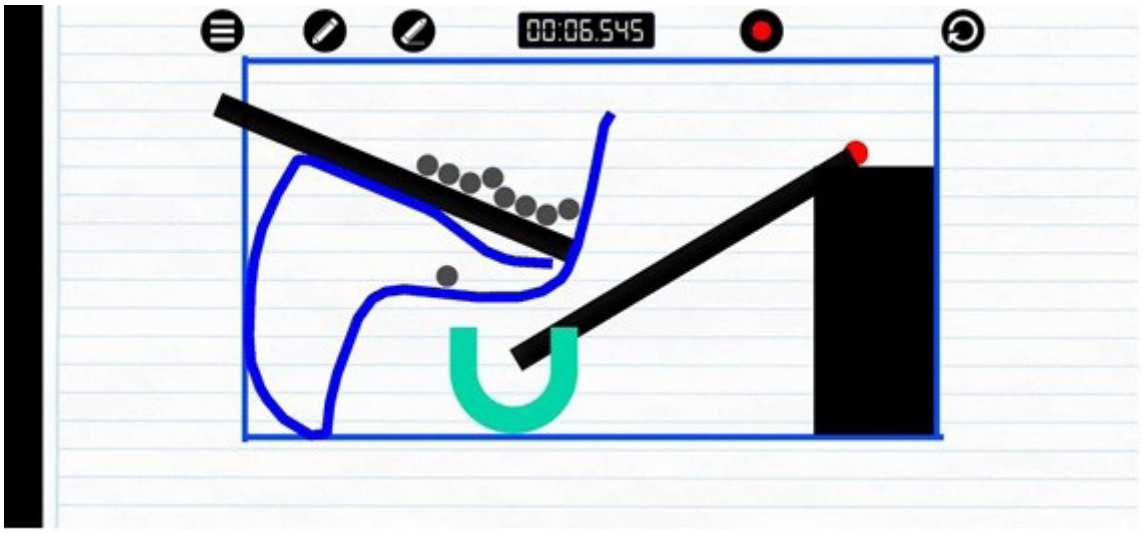
Decomposição: a decomposição no contexto deste jogo específico envolve quebrar o desafio em etapas sequenciais e gerenciáveis, permitindo uma abordagem focada e minimizando tentativas infrutíferas. Antes de qualquer ação, observa-se o layout inicial do nível e identifica-se onde estão as bolas cinzas e a bola vermelha em relação ao balde verde (Figura 1). Decompomos então em dois subproblemas:

O primeiro subproblema é evitar que as bolas cinzas caiam no caminho da bola vermelha (Figura 2). Nesse estágio, o jogador precisa desenvolver uma estratégia para bloquear ou desviar as bolas cinzas. Isso pode incluir desenhar linhas ou estruturas que atuem como barreiras físicas (Figura 3).

Após conter as bolas cinzas, o foco muda para a bola vermelha. Aqui, o jogador precisa planejar como movimentar a bola vermelha para que ela ganhe impulso e direção suficientes para alcançar o balde verde (Figura 4). Isso pode exigir a criação de rampas, declives ou até mecanismos de balanço.

Com as estruturas básicas em lugar, o jogador pode precisar fazer ajustes finos, adicionando ou modificando linhas para refinar a trajetória da bola vermelha e garantir que ela chegue ao destino.

Figura 3. Solução para a primeira parte do desafio 15 - as bolas cinzas são contidas, permitindo que a bola vermelha prossiga

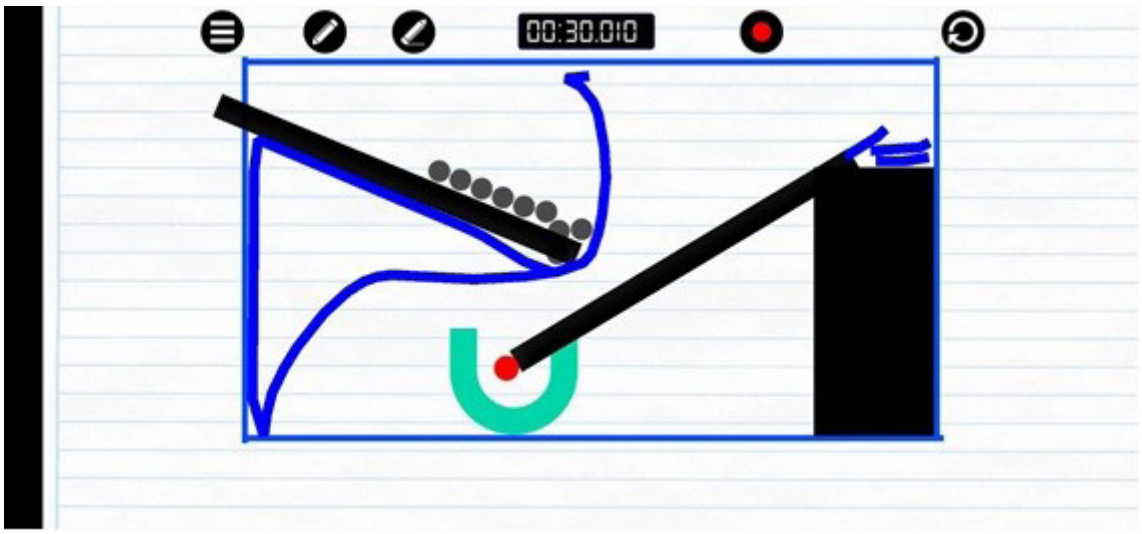


Fonte: Captura de tela *Physics Drop*

Abstração: concentramo-nos em elementos que influenciam diretamente a solução, como as forças físicas atuantes na bola, ignorando aspectos estéticos como a aparência das linhas desenhadas.

Reconhecimento de Padrões: jogadores observam quais tipos de linhas e formas resultam na movimentação desejada da bola. Por exemplo, eles podem notar que rampas inclinadas em um ângulo específico são eficazes em dar à bola a velocidade e direção corretas para alcançar o objetivo. Com a experiência adquirida em níveis anteriores, os jogadores podem aplicar soluções similares em novos desafios. Por exemplo, se uma rampa suave ajudou a bola a pular sobre um obstáculo anteriormente, uma solução similar pode ser tentada em um novo cenário com um obstáculo semelhante.

Figura 4. Solução para a segunda parte do desafio 15 - a bola vermelha é direcionada com sucesso para o balde verde



Fonte: Captura de tela *Physics Drop*

Algoritmos: a criação de um algoritmo pode ser vista como a definição de uma série de ações ou desenhos que o jogador faz para guiar a bola até o objetivo. Cada tentativa de resolver um nível é, essencialmente, um algoritmo, onde o jogador aplica uma sequência específica de passos (desenhos e ajustes) para alcançar o resultado desejado. Cada solução bem-sucedida é um algoritmo que pode ser ajustado ou reaplicado conforme necessário.

Ao aplicar estes pilares do Pensamento Computacional, os jogadores podem abordar cada nível do *Physics Drop* de maneira mais estratégica e eficaz, melhorando suas habilidades de resolução de problemas e raciocínio lógico enquanto se divertem.

Jogos clássicos

Jogos clássicos como xadrez, sudoku e quebra-cabeças de palavras cruzadas também são ótimas ferramentas para desenvolver o pensamento computacional. Eles exigem estratégia, análise e tomada de decisões, estimulando habilidades como a resolução de problemas, o raciocínio lógico e a previsão de resultados. Nesta seção, o Sudoku vai ser utilizado para exemplificar como o Pensamento Computacional pode ser desenvolvido em sala de aula. No entanto, caso você não atue como docente, ainda assim poderá seguir essas mesmas estratégias de forma individual.

O Sudoku é um quebra-cabeça lógico de números muito popular. Consiste em uma grade de 9x9, subdividida em nove quadrados menores de 3x3. O objetivo é preencher a grade com números de 1 a 9, de forma que cada número apareça apenas uma vez em cada linha, coluna e quadrado menor. Cada puzzle de Sudoku começa com alguns números já

colocados na grade, e o jogador deve usar esses números como pistas para completar os espaços vazios, aplicando lógica e raciocínio dedutivo.

Figura 5. Tela do Jogo Sudoku

Fonte: Sudoku Online

Integrar o Sudoku na sala de aula começa com uma discussão sobre os princípios fundamentais do jogo, enfatizando a importância de habilidades como lógica e reconhecimento de padrões. Este enfoque inicial ajuda os alunos a compreender como abordar problemas complexos de uma maneira mais holística, ou seja, vendo o todo e não apenas as partes. Ao aprenderem a identificar padrões e aplicar lógica para resolver os quebra-cabeças do Sudoku, os alunos desenvolvem uma base sólida para pensar criticamente e encontrar soluções eficazes para diversos tipos de desafios.

Para promover a decomposição, os alunos são incentivados a dividir o puzzle em regiões menores, facilitando a gestão da complexidade do jogo. Isso é análogo ao processo de dividir um problema de programação em partes menores e mais manejáveis.

Ao abordar o Sudoku, os estudantes praticam abstração ao filtrar as informações desnecessárias (ex.: ordem que os os números são preenchidos) e se concentrar nas regras e técnicas (ex.: espaços vazios e números presentes na grade) que conduzem à solução. Eles aprendem a discernir entre as informações úteis e as distratoras, uma habilidade crucial no desenvolvimento de soluções eficazes em tecnologia.

Os alunos também podem ser desafiados a criar seus próprios puzzles de Sudoku. Esse processo criativo os obriga a entender profundamente as regras e os padrões do jogo, uma prática que desenvolve seu entendimento de algoritmos e regras lógicas.

Os alunos também podem ser desafiados a criar seus próprios puzzles de Sudoku, mesmo que de forma desplugada criando apenas a lógica do jogo. Esse processo criativo os ajuda a entender profundamente as regras e os padrões do jogo, uma prática que desenvolve seu entendimento de algoritmos e regras lógicas.

Ao usar jogos como Sudoku em sala de aula, que em versão física ou digital, é essencial que os professores acompanhem como os alunos resolvem os problemas, e os incentivem a expressar o raciocínio por trás de suas escolhas. Isso envolve questionar os alunos sobre o processo de pensamento que estão aplicando, o que ajuda a integrar e a concretizar os pilares do Pensamento Computacional. Para que esta abordagem seja eficaz, é importante que os alunos já tenham um conhecimento prévio desses pilares, permitindo que eles identifiquem e apliquem conscientemente essas habilidades durante o jogo.

Pensamento Computacional desplugado

Como aprendemos nos capítulos anteriores, o Pensamento Computacional é um dos eixos da Computação no Brasil. Considerando um país com diferentes realidades educacionais, estratégias foram criadas para que a Computação pudesse ser ensinada em escolas com infraestrutura tecnológica limitada. Uma dessas estratégias é a Computação Desplugada, que é uma abordagem educacional que ensina conceitos fundamentais de computação sem a necessidade de dispositivos eletrônicos. Esta metodologia é particularmente útil para introduzir noções básicas de computação e lógica de forma acessível e envolvente, especialmente em ambientes com recursos limitados ou para complementar o aprendizado digital.

Computação Desplugada refere-se a um conjunto de atividades pedagógicas que utiliza jogos, quebra-cabeças (que vimos na seção anterior), e outras ferramentas físicas para ensinar conceitos computacionais. Por exemplo, as atividades podem incluir o uso de cartões para simular algoritmos ou jogos de tabuleiro para ensinar lógica de programação sem necessidade de tecnologia digital.

Para desenvolver o Pensamento Computacional de forma desplugada, ou seja, sem a necessidade de dispositivos eletrônicos, podem ser adotadas diversas estratégias e atividades. Aqui estão algumas maneiras de promover o Pensamento Computacional de forma desplugada:

- **Uso de metáforas físicas:** utilize objetos tangíveis, como cartões, blocos, cordas ou papel colorido, para representar conceitos abstratos da computação, como algoritmos, estruturas de dados e operações lógicas. Isso ajuda os alunos a visualizarem e compreenderem melhor os conceitos.

- **Atividades de Simulação:** crie atividades que simulem processos computacionais do mundo real. Por exemplo, simular uma rede de comunicação usando pessoas como "nós" e passando mensagens para demonstrar conceitos de redes e comunicação de dados.

- ***Storytelling* e *Role-Playing*:** incentive os alunos a criar histórias ou jogos de representação onde devem "programar" o comportamento de personagens ou resolver problemas por meio de sequências lógicas de eventos. Isso ajuda a desenvolver habilidades de pensamento algorítmico e lógico.

Essas abordagens desplugadas não apenas tornam o aprendizado de computação mais acessível, mas também promovem a colaboração entre os alunos, o desenvolvimento de habilidades de comunicação e a compreensão mais profunda dos conceitos computacionais.

Existem diversos materiais, como livros e artigos, que detalham várias atividades. Alguns deles, já tive a oportunidade de aplicar a alunos das séries iniciais quando lecionava no Ensino Fundamental, nas aulas de robótica, e outras usei mais tarde, quando lecionava para alunos do ensino técnico em disciplinas técnicas de programação, para introduzir conceitos.

O livro "CS Unplugged: an enrichment and extension programme for primary-aged students" de Tim Bell, Ian H. Witten e Mike Fellows, apresenta uma série de atividades educativas destinadas a ensinar conceitos fundamentais de ciência da computação sem o uso de computadores. As atividades, projetadas principalmente para estudantes do ensino fundamental, abordam temas como algoritmos, codificação de erro e compressão de dados através de jogos e quebra-cabeças interativos.

O livro de Bell, Witten e Fellows (2015) serve como um recurso valioso para educadores que desejam integrar o ensino de ciência da computação em suas aulas de forma prática e envolvente. Cada capítulo do livro é detalhado com instruções passo a passo, objetivos de aprendizado e sugestões para variações das atividades, permitindo que os professores adaptem o material a diferentes idades e contextos educacionais. "CS Unplugged" não apenas facilita a introdução de princípios computacionais no currículo escolar, mas também promove uma abordagem de ensino inclusiva e diversificada, preparando os estudantes para os futuros desafios tecnológicos de diversas áreas

Você pode acessar e baixar o livro completo gratuitamente pelos seguintes links, que oferecem detalhes mais específicos e permitem a utilização das atividades propostas:

· CS Unplugged em inglês.
·CS Unplugged em português, com tradução adaptada por Luciana Porto Barreto, realizada no ano de 2011.

A série “Computação Fundamental” é composta por livros didáticos destinados ao ensino de computação para estudantes do Ensino Fundamental II. Os livros têm a versão para o professor com comentários e dicas nas atividades propostas, sendo uma grande parte delas, atividades desplugadas. Os livros podem ser acessados e utilizados de forma gratuita a partir do endereço:

https://sites.google.com/view/computacaofundamental/.

Ao integrar a computação desplugada nas salas de aula, os educadores têm uma ferramenta poderosa para cultivar o pensamento computacional entre os estudantes de uma forma inclusiva e engajadora. Essas práticas não apenas simplificam a introdução dos conceitos fundamentais da ciência da computação, mas também reforçam habilidades críticas como solução de problemas, pensamento lógico e colaboração. As atividades propostas nos livros da série “Computação Fundamental” e em “CS Unplugged” demonstram a viabilidade e o impacto positivo do ensino desplugado, preparando os alunos para enfrentar desafios tecnológicos e sociais em um mundo cada vez mais digitalizado.

Pensamento Computacional e questões de lógica

O Pensamento Computacional (PC) é uma metodologia que transcende a área de TI, aplicando-se a diversos contextos e problemas e inclui questões de lógica complexas. Esta habilidade de pensar computacionalmente é construída sobre quatro pilares: decomposição, reconhecimento de padrões, abstração e algoritmos. Cada um desses pilares oferece uma ferramenta poderosa para desmembrar e entender desafios, não apenas em computação, mas em qualquer área que exija análise sistemática e resolução de problemas. Ao ensinar e aplicar esses pilares em questões lógicas, promovemos uma forma de raciocínio que é essencial para abordagens analíticas e para o desenvolvimento de soluções inovadoras e eficientes.

No Brasil, as Olimpíadas de Robótica (OBR) e de Informática (OBI) são eventos anuais significativos que desempenham um papel crucial no estímulo à educação em ciências exatas e tecnologia entre jovens estudantes. Ambas as competições incentivam o desenvolvimento de habilidades em áreas como lógica, programação e resolução de problemas.

A OBR apresenta duas modalidades principais: prática e teórica. A modalidade prática envolve a construção e programação de robôs que devem realizar tarefas específicas ou superar obstáculos em um ambiente controlado, simulando situações reais em contextos industriais ou de resgate.

A modalidade teórica da OBR é projetada para testar os conhecimentos dos alunos em conteúdos curriculares vinculados à Base Nacional Comum Curricular (BNCC), aplicados no contexto da robótica. Esta modalidade apresenta provas de múltipla escolha que introduzem exemplos reais de robótica e suas potenciais aplicações para a sociedade. A modalidade é dividida em seis níveis, variando conforme o ano escolar dos participantes, com o número de questões e a duração da prova ajustados por nível. Os níveis vão desde o 1º ano do Ensino Fundamental até o final do Ensino Médio.

A Olimpíada Brasileira de Informática (OBI) é uma competição nacional que envolve duas principais modalidades: Iniciação e Programação, cada uma dividida em vários níveis para acomodar diferentes faixas etárias e habilidades dos participantes.

A modalidade programação da OBI é destinada a alunos com conhecimento em linguagens de programação. Essa modalidade desafia os participantes com problemas que devem ser resolvidos por meio da codificação em diversas linguagens de programação aceitas pela competição, como Python, Java, e C++, entre outras.

A modalidade iniciação da OBI é focada em estudantes mais jovens: essa modalidade propõe problemas de lógica e raciocínio computacional sem a necessidade de conhecimento prévio em programação. Os problemas são apresentados de forma que os alunos utilizem o raciocínio lógico para solucionar questões que simulam situações práticas e teóricas relacionadas à computação.

O Pensamento Computacional pode ser aplicado nas resoluções destas provas por meio de:

- **Decomposição:** quebrar as questões complexas em partes menores para facilitar a compreensão e solução.
- **Reconhecimento de Padrões:** identificar semelhanças nas questões ou conceitos recorrentes que podem guiar para a solução.
- **Abstração:** focar nas informações essenciais das questões, ignorando detalhes que não contribuem para a solução.
- **Algoritmos:** desenvolver uma sequência lógica de passos para resolver cada questão, aplicando conceitos teóricos de forma estruturada.

Vamos exemplificar a partir da resolução da questão 13 aplicada para o nível 4 da OBR em 2022 (Figura 6).

- A decomposição no contexto do código apresentado na questão envolve entender cada componente e sua função dentro do algoritmo.

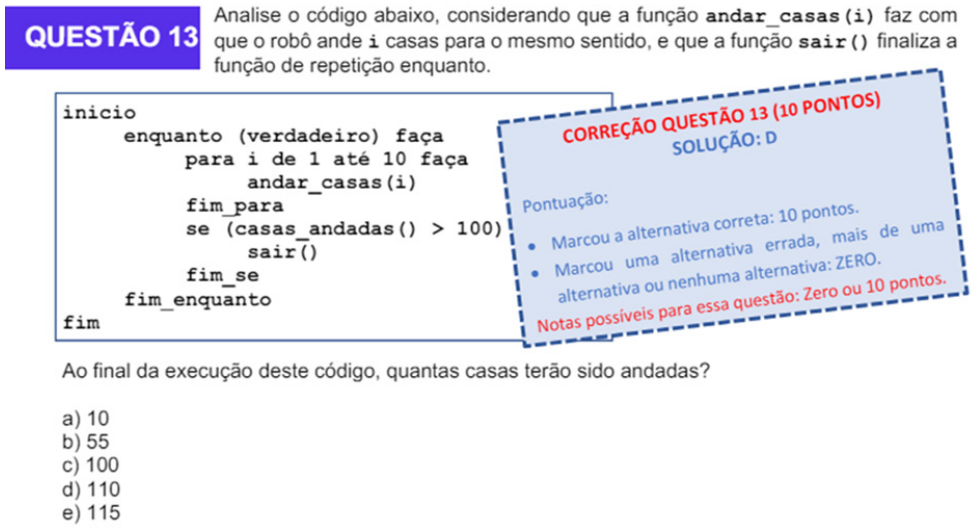
- **Loop “enquanto (verdadeiro)”:** Este loop cria um ciclo contínuo, fazendo com que o código dentro dele seja repetido indefinidamente até que uma condição específica seja atendida para interrompê-lo.

- **Loop “para i de 1 até 10”:** Este loop controla o movimento do robô, fazendo-o andar um número crescente de casas, de 1 a 10, em cada iteração. Ele é executado repetidamente dentro do loop “enquanto”.

- **Função casas_andadas():** Esta função é chamada para verificar o total de casas que o robô andou até o momento. É crucial para determinar se a condição para sair do loop infinito foi atingida.

- **Função sair():** Esta função é chamada para terminar o loop “enquanto” quando o robô atinge ou excede 100 casas andadas.

Figura 6. Questão 15 do nível 4 da prova da OBR modalidade iniciação – edição de 2022.



QUESTÃO 13

Analise o código abaixo, considerando que a função `andar_casas(i)` faz com que o robô ande `i` casas para o mesmo sentido, e que a função `sair()` finaliza a função de repetição enquanto.

```
inicio
     enquanto (verdadeiro) faça
          para i de 1 até 10 faça
               andar_casas(i)
          fim_para
          se (casas_andadas() > 100)
               sair()
          fim_se
     fim_enquanto
fim
```

CORREÇÃO QUESTÃO 13 (10 PONTOS)
SOLUÇÃO: D

Pontuação:
- Marcou a alternativa correta: 10 pontos.
- Marcou uma alternativa errada, mais de uma alternativa ou nenhuma alternativa: ZERO.

Notas possíveis para essa questão: Zero ou 10 pontos.

Ao final da execução deste código, quantas casas terão sido andadas?

a) 10
b) 55
c) 100
d) 110
e) 115

Fonte: OBR (2022).

Na aplicação do **reconhecimento de padrões** para a questão do robô, observa-se o comportamento repetitivo do loop que faz o robô andar de 1 a 10 casas. Identifica-se que a operação realizada pelo loop é a soma progressiva de números de 1 até 10. Reconhecer este padrão ajuda a entender que, em cada ciclo completo do loop, o robô andará um número fixo de casas, permitindo prever o comportamento do código sem cálculos detalhados. Esta percepção é crucial para simplificar a resolução do problema e entender quando o robô deve parar.

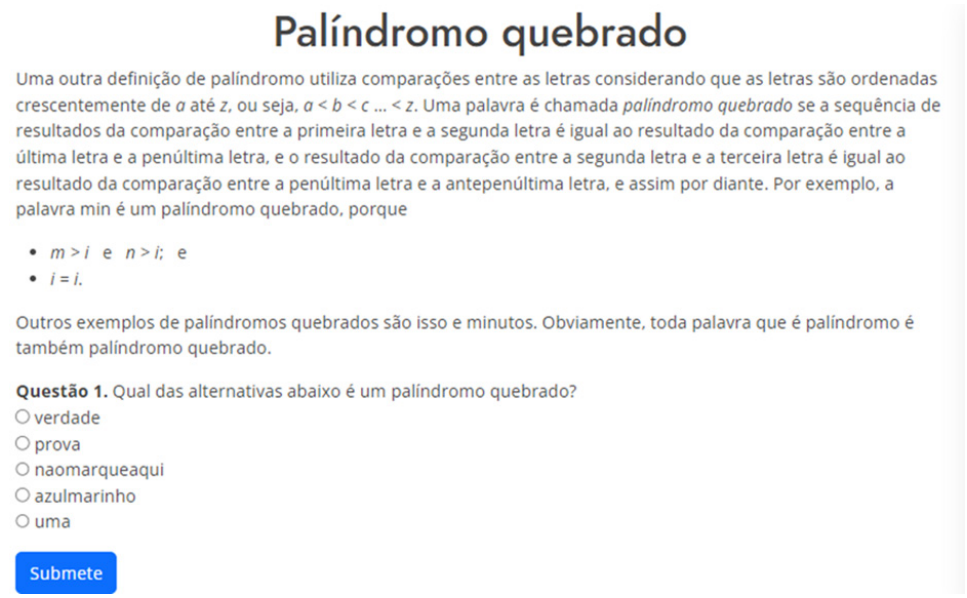
Na aplicação da **abstração**, concentramo-nos na lógica essencial do código, ignorando detalhes técnicos menores. Focamos em como o número de casas andadas pelo robô é

acumulado e como isso influencia a condição de parada do loop. Detalhes como a implementação específica das funções, características do hardware do robô e o ambiente de operação são abstraídos. Assim, o foco recai sobre o fluxo lógico principal: o robô anda uma sequência crescente de casas até atingir um limite, acionando a função de saída. Esse processo enfatiza a importância da lógica de contagem e da condição de término, sem aprofundar-se na implementação técnica de cada parte.

Na questão proposta, o **algoritmo** no código é uma ferramenta essencial para controlar as ações do robô. A aplicação do algoritmo começa com o loop “para”, que direciona o robô a andar um número crescente de casas, de 1 a 10, somando um total de 55 casas na primeira iteração. Ao reiniciar o loop, o robô repete a ação, andando por outras 55 casas. Assim que a soma total atinge 110 casas, ultrapassando o limite estabelecido de 100, a condição dentro do loop “enquanto” (verificação através da função `casas_andadas() > 100`) é ativada, levando à execução da função `sair()`, que termina o loop e o programa. Este processo ilustra como os algoritmos organizam sequencialmente as operações para resolver problemas e atingir resultados específicos com base em condições pré-definidas.

Agora, vamos analisar uma questão da OBI e entender como o pensamento computacional pode ser aplicado na sua resolução.

Figura 7. Questão da modalidade Iniciação Nível 2 da OBI da edição 2021.



Palíndromo quebrado

Uma outra definição de palíndromo utiliza comparações entre as letras considerando que as letras são ordenadas crescentemente de *a* até *z*, ou seja, $a < b < c \ldots < z$. Uma palavra é chamada *palíndromo quebrado* se a sequência de resultados da comparação entre a primeira letra e a segunda letra é igual ao resultado da comparação entre a última letra e a penúltima letra, e o resultado da comparação entre a segunda letra e a terceira letra é igual ao resultado da comparação entre a penúltima letra e a antepenúltima letra, e assim por diante. Por exemplo, a palavra min é um palíndromo quebrado, porque

- $m > i$ e $n > i$; e
- $i = i$.

Outros exemplos de palíndromos quebrados são isso e minutos. Obviamente, toda palavra que é palíndromo é também palíndromo quebrado.

Questão 1. Qual das alternativas abaixo é um palíndromo quebrado?

- ○ verdade
- ○ prova
- ○ naomarqueaqui
- ○ azulmarinho
- ○ uma

Submete

Fonte: Unicamp

Ao abordar a resolução de palíndromos quebrados utilizando o Pensamento Computacional, iniciamos com a decomposição da palavra em pares de letras adjacentes. Este passo simplifica o problema ao reduzir a análise para comparações binárias entre as letras. Prosseguimos com o reconhecimento de padrões, observando as relações de com-

paração (maior, menor ou igual) entre essas duplas. O foco se mantém estritamente nas relações entre letras, desconsiderando outros atributos da palavra, o que é um processo de abstração que destaca apenas os elementos necessários para a análise. Finalmente, aplicamos um algoritmo para verificar se esses padrões de comparação são simétricos em relação ao centro da palavra, comparando pares do início e do fim e movendo-se progressivamente para o centro. Esta abordagem sistemática não apenas facilita a verificação de cada par de letras, mas assegura uma análise abrangente e eficaz para identificar palíndromos quebrados.

Entender e aplicar os pilares do Pensamento Computacional é crucial não só para o sucesso em olimpíadas, mas também para resolver problemas em diversas disciplinas. Essa abordagem não apenas aprimora habilidades lógicas e analíticas dos alunos, mas também os prepara para enfrentar desafios complexos de maneira sistemática e eficiente, transcendendo os limites das salas de aula e aplicando-se a situações reais e variadas do cotidiano.

Capítulo 4
Introdução a algoritmos com *Code.org* e *GearsBot*

Neste capítulo, exploraremos os fundamentos dos algoritmos, essenciais para a computação e para a resolução de problemas de forma lógica e estruturada. Algoritmos são conjuntos de instruções passo a passo que definem como realizar uma tarefa, sendo a base para o desenvolvimento do pensamento computacional. Esse conceito central não apenas direciona a maneira como interagimos com as tecnologias modernas, mas também molda nosso modo de pensar e resolver problemas em diversas áreas, mostrando a universalidade e a importância dos algoritmos no dia a dia.

Hoje, existem diversas ferramentas e iniciativas destinadas a ensinar algoritmos para pessoas com diferentes níveis de conhecimento, áreas de interesse e faixas etárias. Essas plataformas variam desde cursos online que introduzem conceitos básicos de programação até ambientes de simulação avançados que permitem aos usuários testar e refinar algoritmos em cenários complexos. Tais recursos são essenciais para democratizar o acesso ao aprendizado de habilidades cruciais em tecnologia, preparando uma geração diversificada para os desafios do futuro digital.

Para enriquecer a compreensão prática dos conceitos discutidos neste capítulo, exploraremos as plataformas *Code.org* e *GearsBot*. Estas são ferramentas exemplares para a aprendizagem interativa de programação, permitindo aos usuários não apenas aprender, mas também aplicar os conceitos de algoritmos de maneira engajadora. *Code.org* é conhecida por sua interface amigável e cursos variados que cativam desde jovens estudantes a adultos, enquanto *GearsBot* oferece uma experiência mais focada na simulação de robótica, ideal para quem deseja uma abordagem mais técnica e prática.

Ao explorar essas plataformas, você terá a oportunidade de realizar diversos projetos que estimulam o pensamento crítico e algorítmico. Interagindo com essas ferramentas, desenvolveremos habilidades valiosas no mundo tecnológico, como raciocínio lógico e resolução sistemática de problemas, essenciais em diversas disciplinas e carreiras.

Conceitos essenciais de algoritmos

Algoritmos são instruções passo a passo que definem como executar tarefas e resolver problemas. Eles são a base de todo o processamento computacional, influenciando diretamente a eficiência e eficácia de softwares e sistemas.

Todo algoritmo consiste em entradas (dados iniciais), processamento (instruções que transformam as entradas) e saídas (resultados do processamento). Essa estrutura permi-

te que algoritmos manipulem dados e produzam resultados específicos, seguindo lógicas predeterminadas que garantem consistência e previsibilidade.

A estrutura de um algoritmo pode ser comparada à de uma receita de cozinha. Nas receitas, as entradas são os ingredientes necessários, o processamento é o conjunto de passos culinários que transformam os ingredientes (como misturar, cozinhar ou assar), e as saídas são o prato finalizado. Assim como em um algoritmo, a receita segue uma lógica específica para garantir que, se os passos forem seguidos corretamente, o resultado será previsível e consistente, como um bolo ou uma refeição completa.

Algoritmos podem ser representados de várias formas, como fluxogramas ou sequências de passos, e são implementados em softwares através de linguagens de programação. Estas funcionam como ferramentas que permitem aos desenvolvedores escrever "receitas" que os computadores podem entender e executar, cada uma com suas regras próprias de sintaxe (estrutura e forma do código) e semântica (significado das instruções).

Essas linguagens de programação são usadas para instruir computadores a realizar desde tarefas simples até operações complexas de forma precisa e eficiente, sendo que a sintaxe se refere à estrutura e forma do código, enquanto a semântica aborda o significado das instruções dentro do contexto do programa.

Por exemplo, na linguagem de programação Python, a sintaxe para um loop simples que imprime números de 1 a 5 seria:

```
for i in range(1, 6):
    print(i)
```

Neste código, a sintaxe se refere ao uso correto de palavras-chave como *for* e *range*, e a função *print()*. Erros de sintaxe, como uma pontuação incorreta ou palavras-chave mal digitadas, impedirão que o programa seja executado. Por outro lado, a semântica está relacionada ao que o código realiza: repetir um bloco de código cinco vezes, incrementando o número a cada iteração. Erros semânticos ocorrem quando a lógica do algoritmo não está alinhada com o objetivo pretendido, apesar do código estar sintaticamente correto.

Neste capítulo, utilizaremos a programação visual em blocos no *Code.org* e *GearsBot*, por se tratarem de comando que são encaixados para criar a lógica do programa, não precisaremos nos preocupar com a sintaxe, mas teremos que nos preocupar com a semântica.

Neste capítulo, exploraremos a programação em blocos usando as plataformas *Code.org* e *GearsBot*. Essas ferramentas empregam uma abordagem visual onde os comandos

são representados por blocos que se encaixam para formar a lógica do programa. Essa metodologia elimina a preocupação com erros de sintaxe, pois os blocos garantem que a estrutura do código esteja correta. No entanto, ainda precisamos focar na semântica, ou seja, garantir que a sequência de blocos e a lógica implementada correspondam aos objetivos desejados e produzam os resultados esperados.

É importante entendermos qual é o fluxo de execução de um algoritmo, que diz respeito à ordem na qual as instruções em um código de programação são executadas pelo computador. Isso é fundamental para programar efetivamente, pois permite aos desenvolvedores controlar como seus programas respondem a diferentes condições e como processam dados. A execução de um programa envolve vários conceitos chave. entre os quais pode-se citar:

- **Execução Sequencial:** As instruções num programa são normalmente executadas de cima para baixo, uma após a outra. Este é o fluxo mais básico de execução, semelhante a seguir passos numa receita. Por exemplo, em um programa que lê uma entrada, realiza um cálculo e depois imprime um resultado, cada uma dessas operações ocorrerá exatamente nessa ordem:

Leia número1
Leia número2
Soma = número1 + número2
Mostre Soma

- **Decisões (condicionais):** permitindo que o programa execute diferentes blocos de código dependendo de condições específicas. Isso é crucial para criar programas que possam responder de maneira diferente a diferentes entradas ou situações. Imagine um programa de software para uma máquina de vendas que precisa decidir que tipo de troco fornecer baseado no valor pago pelo cliente. O programa utilizaria uma estrutura condicional para determinar o troco:

Se o valor_pago > preço_do_produto então
  troco = valor_pago - preço_do_produto
  Mostre "Seu troco é", troco
Senão Se valor_pago = preço_do_produto então
  Mostre "Sem troco"
Senão
  Mostre "Valor insuficiente"

- **Loops (repetições):** Os loops, ou laços de repetição, são estruturas de programação usadas para repetir um conjunto de instruções até que uma condição específica seja atendida, tornando o programa mais eficiente ao eliminar a necessidade de escrever o mesmo código várias vezes. Existem dois tipos principais de loops: o laço de repetição para (for), que é utilizado para iterar sobre uma sequência de valores, e o laço de enquanto (while), que executa repetidamente enquanto uma condição for verdadeira.

Exemplo do laço para. Este código irá imprimir os números de 1 a 5:

Para i de 1 até 5 faça
  Mostre i

Exemplo do laço enquanto enquanto. Este código continuará a executar a "ação" enquanto a "condição" for verdadeira.

Enquanto condição for verdadeira faça
  Execute ação

- **Funções e subrotinas:** funções e subrotinas são blocos de código que executam tarefas específicas dentro de um programa. Estes blocos permitem encapsular lógicas complexas de forma a simplificar o código principal, tornando-o mais organizado e fácil de manter. Além disso, funções permitem a reutilização de código, já que uma mesma função pode ser chamada em diferentes partes do programa ou até em diferentes programas. Exemplo:

Função somar(a, b)
  resultado = a + b
  Retorne resultado

Neste exemplo, a função `somar` recebe dois parâmetros, `a` e `b`, e retorna a soma deles. Ela pode ser chamada em qualquer parte do programa que necessite realizar uma soma, promovendo a reutilização e a modularidade do código.

Algoritmos na prática no *Code.org*

Code.org é uma organização sem fins lucrativos dedicada a expandir o acesso ao ensino de ciência da computação nas escolas e a aumentar a participação de mulheres e minorias sub representadas nesse campo. Fundada em 2013 por Hadi e Ali Partovi, a organização ganhou notoriedade com o lançamento da campanha "Hour of Code", que promove

uma introdução de uma hora à programação, incentivando alunos e professores a começarem a aprender sobre computação.

Os principais objetivos do *Code.org* incluem:

- **Promover a educação em ciência da computação:** *Code.org* trabalha para que a ciência da computação seja reconhecida como uma disciplina fundamental nas escolas, assim como matemática e ciências naturais. Eles defendem políticas educacionais que suportem o ensino de computação e oferecem currículos e materiais didáticos para facilitar esse ensino.

- **Desenvolver recursos educacionais gratuitos:** a organização oferece uma variedade de recursos educacionais gratuitos que são acessíveis online. isso inclui cursos, tutoriais e atividades interativas que ajudam os alunos a aprender a programar, pensar criticamente e resolver problemas complexos.

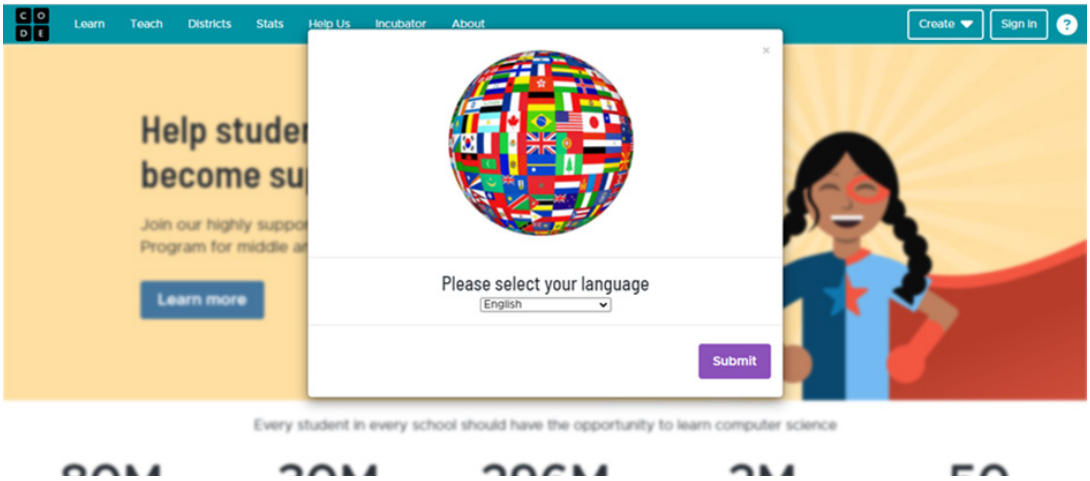
- **Promoção da diversidade:** um dos focos centrais do *Code.org* é aumentar a diversidade no campo da tecnologia. Eles realizam isso incentivando a participação de estudantes de grupos sub-representados, como mulheres e minorias étnicas, em suas iniciativas e programas educacionais.

- **Capacitação de educadores:** *Code.org* também oferece treinamento e recursos para professores que desejam integrar ciência da computação em suas salas de aula. Isso inclui oficinas, seminários online e acesso contínuo a uma comunidade de educadores.

O *Code.org* tem um impacto significativo na educação em ciência da computação, com milhões de estudantes e professores participando de suas atividades e utilizando seus recursos todos os anos. Ao tornar a ciência da computação acessível a todos, eles visam preparar uma geração mais diversificada e bem equipada para enfrentar os desafios futuros em um mundo cada vez mais dominado pela tecnologia.

Para conhecer a plataforma do *Code.org*, entre no site: https://*Code.org*/ e selecione o seu idioma (Figura 8).

Figura 8. Seleção de idioma no *Code.org*



Fonte: https://*Code.org*/

Ao entrarmos no *Code.org* podemos seguir para um dos seguintes caminhos apresentados (Figura 9):

- **Hora do Código:** esta seção convida os usuários a experimentarem uma "Hora do Código", que é uma introdução de uma hora à ciência da computação, projetada para desmistificar a codificação e mostrar que qualquer pessoa pode aprender os fundamentos básicos.

- **Alunos:** oferece recursos para estudantes de diferentes faixas etárias explorarem o "Code Studio" e conhecerem os cursos disponíveis, incentivando-os a aprender e praticar programação.

- **Educadores:** fornece materiais e suporte para professores do ensino fundamental e médio, ajudando-os a integrar a ciência da computação em suas salas de aula e a inspirar seus alunos.

- **Participe:** encoraja a comunidade a apoiar a campanha de promoção da ciência da computação e a ver as estatísticas de impacto, destacando a importância da participação coletiva no avanço da educação em tecnologia.

Figura 9. Página inicial do *Code.org*

Fonte: https://*Code.org*/

O site do *Code.org* oferece a opção de impressão de certificados ao término de qualquer curso da plataforma por meio do perfil do usuário para cursos realizados individualmente a partir de uma conta ou pelo perfil do professor que tenha criado a turma que o aluno está inserido.

Na plataforma *Code.org*, os educadores têm a opção de criar salas de aula virtuais, onde podem gerenciar o aprendizado dos alunos de maneira organizada e estruturada (Figura 10).

Para configurar uma sala de aula, os professores devem criar uma nova seção, atribuir cursos e estabelecer contas de login para os estudantes. Embora essa configuração de login seja opcional para jovens e adultos que já possuem contas individuais na plataforma, é essencial para as crianças pequenas, pois o site oferece métodos simplificados de acesso, como login por meio de imagens e palavras.

Figura 10 Painel de controle do usuário com perfil de professor no *Code.org*

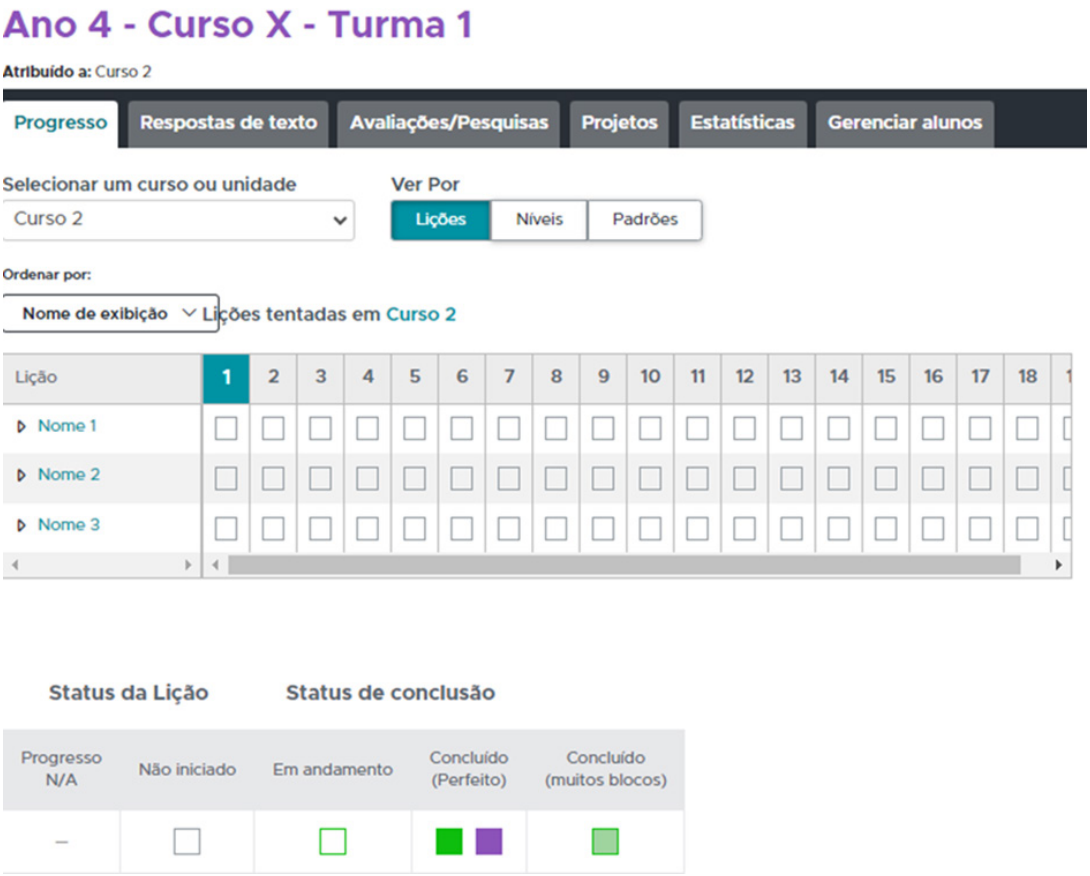
Fonte: https://*Code.org*/

Uma vez criada a seção, o progresso dos alunos pode ser monitorado, permitindo um acompanhamento detalhado do desenvolvimento de cada estudante (Figura 11). Ademais, ao concluir qualquer curso na plataforma, há a possibilidade de imprimir certificados, tanto através do perfil do usuário individual quanto pelo perfil do educador responsável pela turma.

Usando estas ferramentas, os educadores podem efetivamente orientar e apoiar os alunos através de seus estudos de ciência da computação, monitorando seu progresso, fornecendo feedback direto e adaptando as lições conforme necessário para atender às necessidades individuais dos alunos.

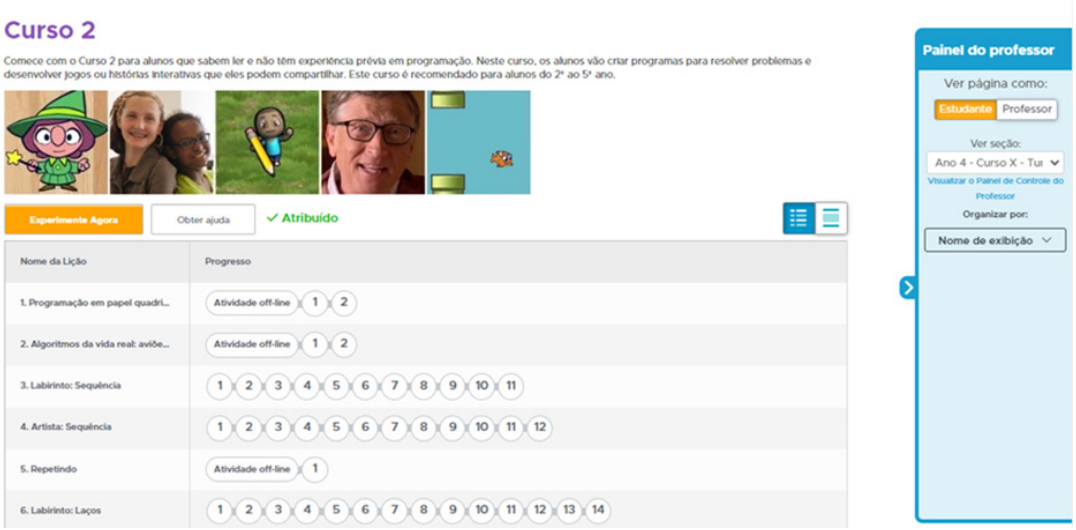
Para experimentar a prática de algoritmos, comece por criar uma conta no *Code.org*. Caso não atue como docente, você pode criar uma conta de aluno para se familiarizar com a plataforma. Se for um professor, você pode visualizar os cursos do ponto de vista de um aluno simplesmente alternando seu perfil na barra lateral (Figura 12).

Figura 11. Acompanhamento de progresso de alunos em determinada turma/seção no perfil de docente no *Code.org*



Fonte: https://*Code.org*/

Figura 12. Barra lateral no Painel do professor para alternar a visualização da página entre professor e estudante.



Fonte: https://Code.org/

Para começar a se familiarizar com algoritmos, acesse a página da Hora do Código. Caso não tenha conhecimento prévio em programação, é recomendado iniciar por uma das opções do curso com tema Festa Dançante. Para exemplificar, escolhi a Festa Dançante: versão IA.

O curso se inicia com um vídeo introdutório (Figura 13). Apesar de estar em inglês, é possível ativar legendas em português brasileiro ou em outro idioma de sua escolha

Figura 13. Vídeo de Introdução ao curso Festa Dançante, Edição para IA

Fonte: https://Code.org/

Ao iniciar as atividades do curso “Festa Dançante: edição IA” no *Code.org*, temos vários componentes interativos na interface que visam facilitar e tornar o aprendizado de programação mais acessível e divertido (Figura 14).

- **Seleção de música:** No topo, você pode escolher a trilha sonora para personalizar a experiência da dança.

- **Instruções:** Esta área contém as instruções passo a passo para a atividade. Além disso, possui recursos de acessibilidade, como um botão de áudio para ouvir as instruções e opções para ajustar o tamanho do texto e a configuração da voz.

- **Faixa de progresso da atividade:** Geralmente localizada no topo ou na lateral, mostra o seu avanço através das lições ou níveis do curso

- **Área de blocos:** À esquerda, encontram-se os blocos de programação disponíveis. Nas lições introdutórias, são fornecidos apenas os blocos necessários para completar a atividade, evitando a sobrecarga de informações e facilitando o foco no aprendizado.

- **Área de trabalho:** No centro, você arrasta e solta os blocos de programação para construir seu algoritmo para realizar o desafio. Aqui, também é possível alternar para ver o código correspondente em JavaScript, que é uma linguagem de programação amplamente utilizada para desenvolvimento web.

- **Palco:** É o espaço onde a animação ou a ação programada acontece. Quando você clica em “Executar”, pode ver o personagem (neste caso, um gato) realizar os movimentos de dança que foram codificados na área de trabalho.

- **Botão Executar:** Localizado na parte inferior, este botão permite iniciar a animação ou ação programada para testar o código criado. Um feedback é fornecido ao término de cada lição.

Figura 14. Primeiro desafio do curso Festa Dançante, Edição para IA

Fonte: https://*Code.org*/

Esses elementos juntos proporcionam uma experiência de aprendizado interativa e engajadora, onde os conceitos fundamentais da programação são ensinados de maneira prática e visual.

Ao concluir um curso no *Code.org*, você tem a oportunidade de obter um certificado de conclusão, um reconhecimento formal de seu empenho e aprendizado (Figura 15). Este certificado pode ser impresso para ser guardado como uma lembrança física de sua conquista ou compartilhado nas redes sociais para celebrar e incentivar outros a também embarcarem na jornada de aprendizado da ciência da computação. É uma excelente maneira de documentar o progresso pessoal e pode servir de incentivo para continuar explorando e aprofundando conhecimentos na área de tecnologia.

Figura 15. Certificado de conclusão do curso

Fonte: https://*Code.org*/

Após concluir o curso introdutório e ganhar experiência prática com programação, você está convidado a embarcar no curso de Ciência da Computação Expresso, que aprofunda elementos e conceitos fundamentais de programação. Durante este curso, você explorará:

- **Algoritmos e sequenciamento:** a base para construir instruções passo a passo que formam um programa.
- **Depuração:** a arte de encontrar e corrigir erros (bugs) em um código.
- **Programação de eventos:** aprender como fazer programas que respondem a ações do usuário ou a outros eventos.
- **Loops (Laços):** entender como usar repetições para economizar tempo e esforço na codificação.
- **Condicionais (Instruções If/Else):** tomar decisões dentro de um programa com base em condições diferentes.
- **Variáveis:** armazenar e manipular dados que podem mudar enquanto o programa é executado.
- **Funções:** reutilizar e organizar o código de maneira eficiente com subprogramas.
- ***Sprites* e animações:** criar e controlar personagens e objetos para desenvolver jogos e histórias interativas.

- **Arte digital e *design*:** utilizar o código para criar arte e designs complexos.

- **Criação de jogos e histórias interativas:** Desenvolver a criatividade e a lógica para construir jogos e narrativas.

- **Desenvolvimento de habilidades de resolução de problemas:** refinar a habilidade de resolver problemas de maneira estruturada e lógica.

- **Conceitos de Computação Baseada em Blocos:** construir programas arrastando e soltando blocos de comandos, uma introdução intuitiva à lógica de programação.

- **Conceitos de Computação Textual (JavaScript):** avançar para linguagens de programação baseadas em texto, começando com JavaScript.

Cada lição é acompanhada por vídeos explicativos e atividades práticas, tornando o aprendizado interativo e engajador. O curso é enriquecido com elementos de jogos e projetos criativos para manter os alunos motivados. Os professores têm acesso a recursos como planos de aula e opções de impressão para auxiliar no ensino e na avaliação do progresso dos alunos.

Este curso é projetado para ser interativo e auto-guiado, permitindo que os alunos apliquem o que aprenderam em uma série de projetos práticos e desafios de codificação. Ao final, os alunos terão um entendimento sólido dos fundamentos da ciência da computação e estarão preparados para abordar problemas mais complexos e continuar sua jornada de aprendizado na programação.

Robótica Educacional Virtual com *GearsBot*

GearsBot é uma ferramenta de simulação de robótica desenvolvida em Singapura durante a pandemia de COVID-19 entre 2020 e 2021. O projeto é completamente open-source e tem contribuído por muitos desenvolvedores nos últimos anos.

O *GearsBot* permite o aprendizado de robótica e programação na abordagem STEAM (Ciência, Tecnologia, Engenharia, Artes e Matemática). Ele oferece um ambiente virtual onde os alunos podem construir, programar e testar robôs, proporcionando uma experiência prática e envolvente. Isso facilita a compreensão dos conceitos de robótica e a aplicação prática da programação.

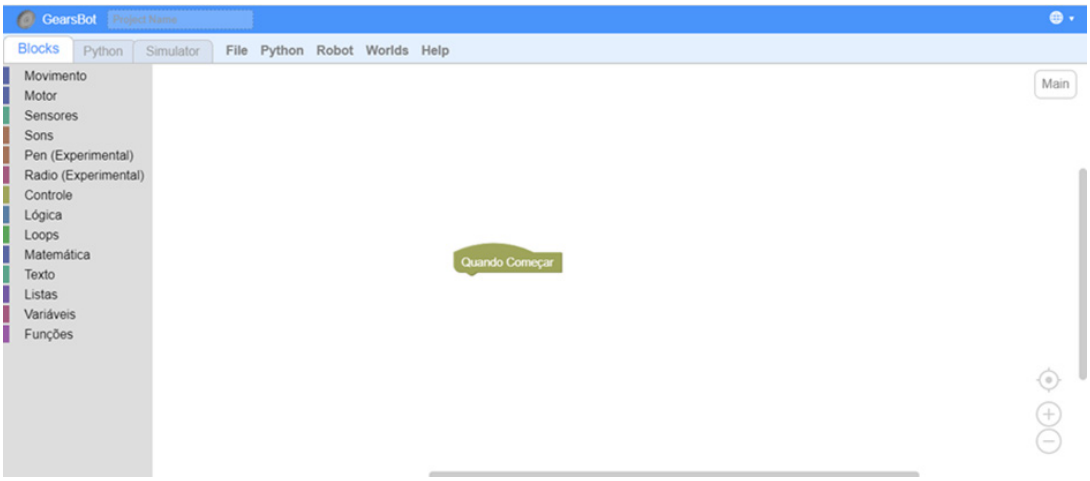
O *GearsBot* ensina robótica e programação, assim como prepara os alunos para o futuro digital, promovendo o desenvolvimento de várias habilidades, incluindo as que envolvem o pensamento computacional.

Além disso, o *GearsBot* é uma alternativa acessível para escolas que não têm recursos para investir em kits físicos de robótica. Como é uma plataforma online, não requer nenhum hardware especializado além de um computador ou tablet com acesso à internet. Isso torna a robótica educacional acessível a um público muito mais amplo, permitindo que mais alunos se beneficiem do aprendizado prático que a robótica pode oferecer.

O *GearsBot* também pode auxiliar na preparação dos alunos para competições de robótica do mundo real, proporcionando uma plataforma para prática e aprendizado contínuos. Isso torna o *GearsBot* uma ferramenta pedagógica poderosa para o ensino de robótica educacional.

Para acessar a ferramenta basta digitar https://gears.aposteriori.com.sg/. A interface do *GearsBot* é intuitiva e fácil de usar, mesmo para quem é leigo em robótica e programação. Aqui está uma explicação detalhada de suas principais áreas e funcionalidades apresentadas na tela inicial (Figura 16) da ferramenta.

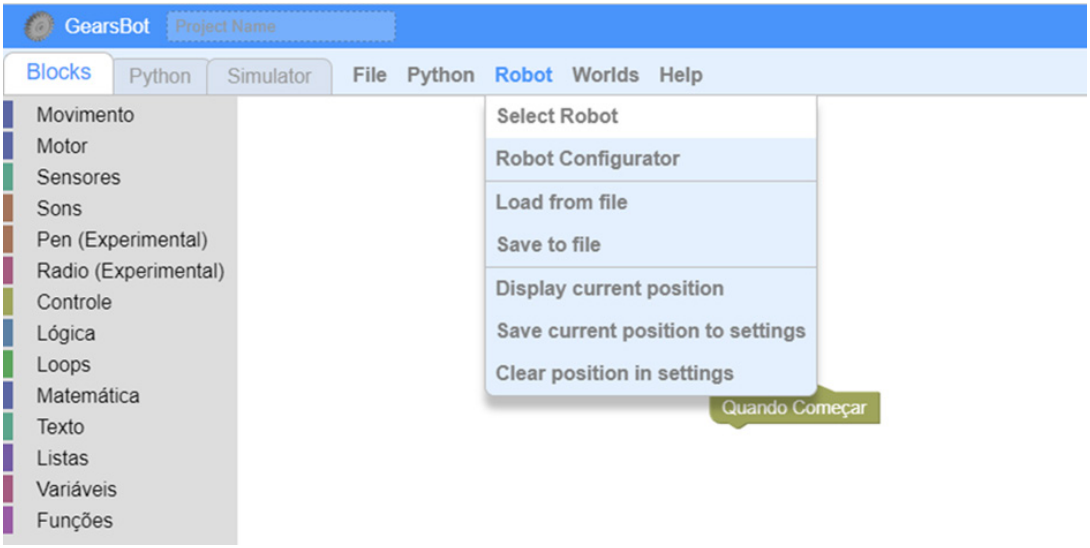
Figura 16. Interface Inicial do *GearsBot*



Fonte: Gears

Primeiro é necessário selecionar o robô que será utilizado no menu “Robot à Select Robot” (Figura 17). Uma das características do *GearsBot* é a capacidade de configurar os robôs equipando-os com uma variedade de sensores e atuadores disponíveis. Para isso, basta acessar o menu “Robot à Robot Configurator” e na tela seguinte “Go to Configurator”.

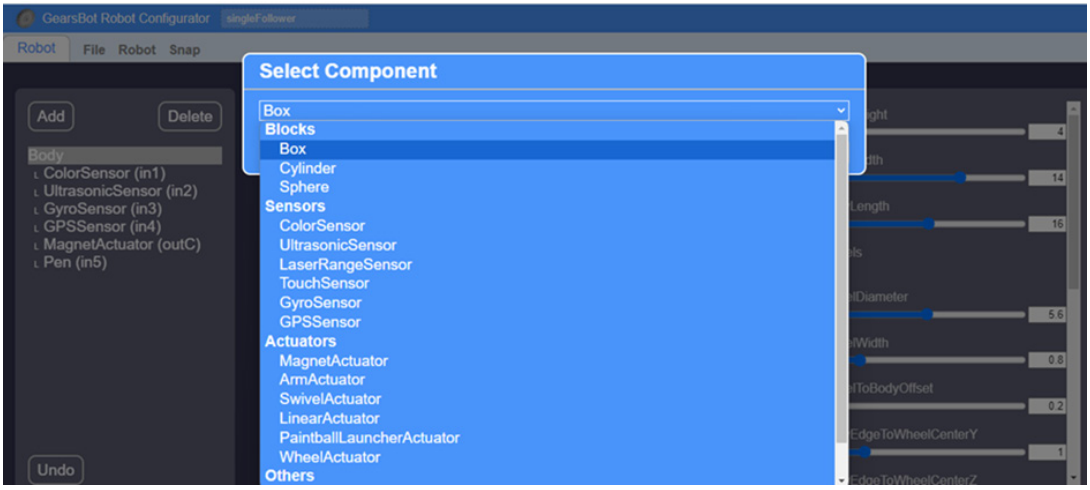
Figura 17. Seleção de um robô no *GearsBot*



Fonte: Gears

Os alunos podem equipar seus robôs com diferentes sensores, como sensores de Cor, Ultrassônico, Giroscópio, Toque, Laser Range e GPS. Cada sensor tem suas próprias funcionalidades e pode ser usado para coletar informações sobre o ambiente ao redor do robô. Por exemplo, um sensor ultrassônico pode ser usado para detectar obstáculos, enquanto um sensor de cor pode ser usado para seguir uma linha no chão.

Figura 18. Configuração de um robô no *GearsBot*



Fonte: Gears

Da mesma forma, os alunos podem equipar seus robôs com diferentes atuadores, como Braço, Giratório, Linear, Lançador de Bolas de Tinta, Ímã, Roda. Cada atuador permite que o robô interaja com o ambiente de uma maneira específica. Por exemplo, um braço pode ser usado para pegar e mover objetos, enquanto uma roda pode ser usada para mover o robô ao redor.

Essa capacidade de personalizar o hardware do robô dá aos alunos a liberdade de experimentar e aprender sobre diferentes aspectos da robótica. Eles podem ver como diferentes configurações de hardware afetam o desempenho do robô e podem ajustar o design do robô para atender a diferentes desafios e objetivos. Isso também incentiva a criatividade e o pensamento crítico, pois os alunos precisam pensar cuidadosamente sobre como projetar e programar seus robôs para obter o melhor desempenho.

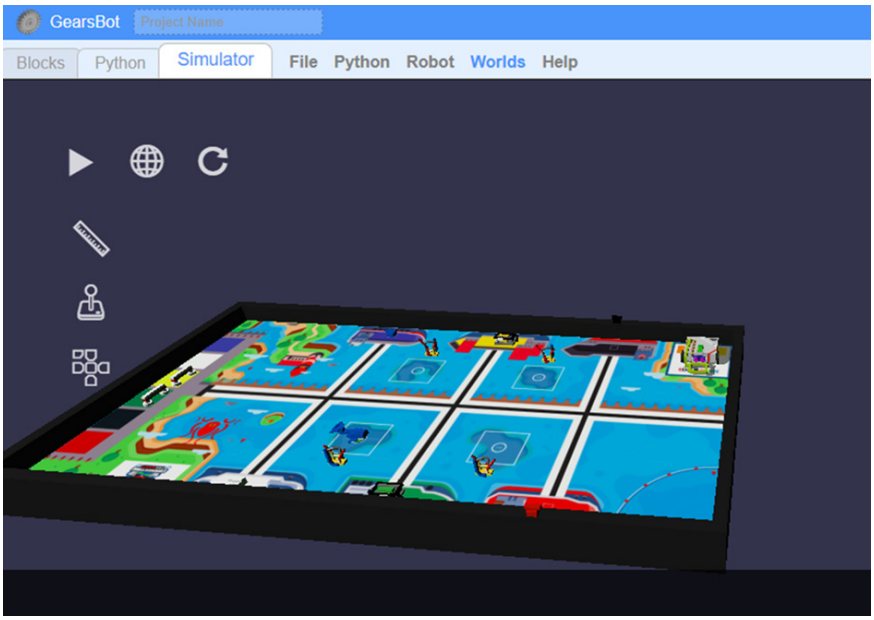
O simulador *GearsBot* é baseado em física, o que significa que ele tenta replicar as leis da física do mundo real. Isso dá aos alunos uma compreensão mais profunda de como os robôs interagem com seu ambiente e como eles podem ser programados para navegar de forma eficaz.

O *GearsBot* também oferece mundos virtuais onde os alunos podem testar seus robôs. Estes mundos podem ser configurados para simular uma variedade de ambientes e desafios, permitindo que os alunos testem suas habilidades em uma variedade de cenários. Para selecionar um mundo você deve acessar o menu “Worlds à Select World”.

O *GearsBot* propõe desafios aos alunos em mundos virtuais que simulam arenas de competições de robótica (Figura 19) como a Olimpíada Brasileira de Robótica (OBR), FIRST LEGO League (FLL) e outras. Essas simulações podem ser adaptadas para replicar os desafios e obstáculos encontrados nessas competições reais.

Os alunos podem construir e programar seus robôs para navegar por essas arenas virtuais, resolver problemas e completar tarefas, assim como fariam em uma competição real. Isso permite que eles apliquem e testem suas habilidades de robótica e programação em um ambiente desafiador e competitivo.

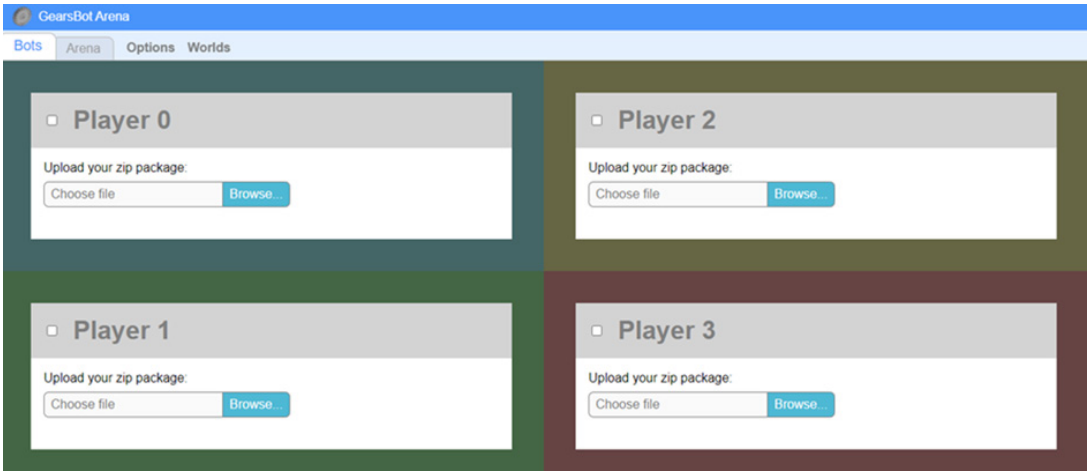
Figura 19. Mundo Virtual do *GearsBot* simulando arena de prova da FLL 2013.



Fonte: Gears

O *GearsBot* oferece a opção de Arena dentro dos Mundos Virtuais acessando o menu "Worlds à Arena". É um recurso especial que oferece ação multiplayer única. Ela permite que até quatro robôs participem simultaneamente em desafios (Figura 20). Isso pode ser particularmente útil para simular competições de robótica, onde os alunos podem testar suas habilidades contra outros robôs em um ambiente competitivo

Figura 20. Arena do *GearsBot*



Fonte: Gears

Além disso, o *GearsBot* suporta programação baseada em blocos e Python, permitindo que os alunos desenvolvam habilidades de programação práticas. Eles podem ver imediatamente como seu código afeta o comportamento do robô no simulador, proporcionando feedback instantâneo e a oportunidade de aprender com a experimentação. Para programar o robô em blocos, basta arrastar os blocos para a área principal.

A Posteriori, a organização idealizadora do *GearsBot*, oferece tutoriais detalhados em seu site. Esses tutoriais abrangem uma variedade de tópicos, desde a introdução ao aplicativo *GearsBot* até a programação de robôs usando blocos de programação e Python. Também há explicação da ferramenta na playlist de vídeo aulas que dá apoio a esse livro.

Em resumo, o *GearsBot* é uma ferramenta pedagógica poderosa que torna a robótica e a programação acessíveis a todos os alunos, independentemente de seus antecedentes ou recursos. Ele oferece uma maneira prática e envolvente de aprender sobre robótica, bem como aplicar e desenvolver habilidades envolvidas no pensamento computacional

Capítulo 5
Programação em Blocos com *Scratch*

A programação em blocos é uma introdução amigável à lógica da programação, permitindo que os iniciantes compreendam os conceitos fundamentais sem se preocupar com a sintaxe específica de uma linguagem de programação. Utilizando o *Scratch*, uma plataforma visual de programação desenvolvida pelo MIT, os alunos podem arrastar e soltar blocos que representam comandos, loops, variáveis e outros elementos essenciais de programação.

Neste capítulo, mergulharemos nos fundamentos dos blocos de programação, ilustrando como combiná-los para construir scripts eficazes que dão vida a histórias, animações e jogos.

Conhecendo o *Scratch*

Na abordagem construcionista de Seymour Papert, o uso do computador é visto como uma ferramenta valiosa para a educação, pois permite que os alunos visualizem e conectem suas construções. Esta abordagem é baseada na ideia de que as crianças aprendem melhor quando descobrem o conhecimento por si mesmas, explorando e interagindo com o conteúdo de maneira significativa.

Conforme vimos no Capítulo 1, para facilitar esse tipo de aprendizagem, Papert criou a linguagem de programação Logo, projetada para ser acessível e intuitiva. Logo foi especialmente pensada para o público jovem e para aqueles sem experiência prévia em computação ou conhecimento avançado em matemática, tornando a programação uma atividade inclusiva e educativa.

O LOGO é uma linguagem de programação que proporciona aos alunos a capacidade de criar micromundos, conforme definido por Papert (1994). Estes são espaços educativos interativos nos quais é possível investigar, experimentar e emular situações do mundo real. Como esclarece Rezende (2000), os micromundos requerem uma linguagem de programação que atue como um intermediário, permitindo que o estudante interaja com o ambiente simulado. Os alunos precisam articular suas ações através de uma sequência de comandos que, de maneira indireta, refletem seu pensamento. A resposta do micromundo às ações implementadas serve como um feedback. Além disso, a possibilidade de acessar e modificar os comandos permite que os estudantes ajustem seu raciocínio inicial baseados nos resultados das ações no micromundo.

Em 2007, o *Scratch* foi introduzido como uma nova ferramenta de programação direcionada a crianças. Desenvolvida com inspiração na linguagem Logo, esta plataforma é fruto do esforço colaborativo de Mitchel Resnick e sua equipe.

Mitchel Resnick (2014) propôs uma metodologia de aprendizagem criativa que ele chama de “os 4Ps”. Esta metodologia é amplamente fomentada por plataformas de programação em blocos como *Scratch*, assim como por outras ferramentas semelhantes que emergiram posteriormente, permitindo que as crianças desenvolvam seus próprios projetos criativos. Os 4Ps são:

- **Projetos:** a aprendizagem é mais eficaz quando as pessoas se engajam ativamente em projetos que consideram significativos, onde podem gerar novas ideias, projetar protótipos e aperfeiçoá-los através de um processo iterativo.

- **Pares:** a aprendizagem se potencializa em um ambiente social, onde indivíduos podem compartilhar ideias, colaborar em projetos e construir juntos.

- **Paixão:** o envolvimento profundo com projetos que despertam o interesse pessoal leva as pessoas a dedicar mais esforço, persistir diante de desafios e, consequentemente, aprender mais.

- ***Play* (Brincar):** a aprendizagem é reforçada através de brincadeiras experimentais, tentativas de novas ideias, manipulação de materiais, teste de limites e riscos, além de repetidas iterações e ajustes.

Estes princípios estão intrinsecamente alinhados e são inspirados pelo construcionismo educacional, que valoriza o processo de aprendizado dos estudantes ao criar projetos que são significativos para eles, em colaboração com seus colegas e dentro de um contexto lúdico.

O *Scratch* é uma iniciativa do grupo Lifelong Kindergarten, parte do Media Lab do MIT, nos Estados Unidos. Este ambiente de programação visual em blocos foi originalmente desenvolvido para crianças e adolescentes, mas sua simplicidade e flexibilidade têm atraído usuários de todas as idades. Ele permite a realização de uma variedade de projetos educativos, incentivando o aprendizado precoce de programação.

O *Scratch* utiliza uma interface de “arrastar e soltar” para manipular blocos de código que se encaixam como peças de Lego, facilitando a programação e o controle de personagens e ações dentro de uma aplicação (Batista, 2015). Este método simplifica o processo de aprendizado, além de tornar a experiência intuitiva e acessível, promovendo uma introdução engajadora ao mundo da computação.

O *Scratch* proporciona um ambiente lúdico e intuitivo onde crianças podem criar jogos, animações e histórias, estabelecendo assim micromundos interativos. Este ambiente facilita a personalização, permitindo que os usuários modifiquem cenários, personagens, incluam sons e utilizem a câmera, tornando-o aplicável a diversas disciplinas educativas.

Com o sucesso do *Scratch*, surgiram outras ferramentas de programação visual em blocos inspiradas nele. Entre estas, destacam-se o *ScratchJr*, voltado para crianças pré-leitoras; o AppInventor, que possibilita a criação de aplicativos para Android; e as ferramentas do *Code.org*, que oferecem cursos e atividades de programação para todas as idades.

Interface do *Scratch*

O *Scratch*, atualmente na sua terceira edição, é acessível tanto em sua versão on-line quanto offline para computadores. Através do site oficial https://*Scratch*.mit.edu/, os usuários podem mergulhar em diversas funcionalidades que estimulam a criatividade e o aprendizado da programação.

A página inicial do site do *Scratch* (Figura 20) apresenta os seguintes menus e recursos:

- **Criar:** esta aba é um convite aberto para iniciar novos projetos. Ao selecioná-la, você é levado para a área de desenvolvimento do *Scratch*, um espaço interativo onde a programação é feita através do encaixe de blocos.

- **Explorar:** essa área serve como um repositório inspirador, cheio de projetos criados pela comunidade global do *Scratch*. Neste espaço colaborativo, os projetos podem ser visualizados, remixados e compartilhados.

- **Ideias:** além de ser um repositório de projetos, o *Scratch* oferece um conjunto robusto de recursos educacionais, como guias de atividades, cartões para codificação e tutoriais interativos. Esses materiais são projetados para orientar os usuários através dos fundamentos da criação de projetos e incentivá-los a superar novos desafios, tornando a aprendizagem da programação acessível e agradável.

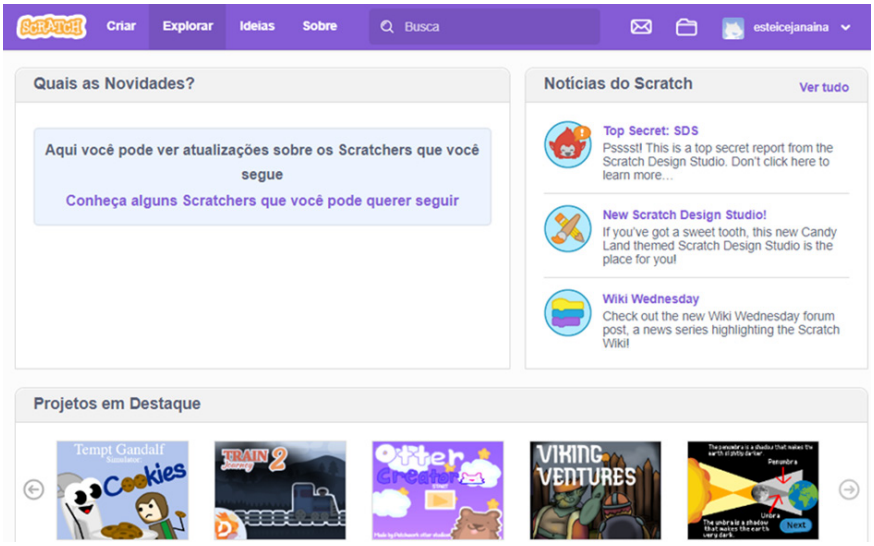
- **Sobre:** nesta seção, você encontrará informações sobre a missão e o impacto do *Scratch*. Além de histórias de sucesso, há também recursos dedicados para pais e educadores, bem como informações sobre o *Scratch* Day - um evento global que celebra e compartilha criações feitas dentro da comunidade do *Scratch*.

- **Busca:** uma ferramenta para filtrar e encontrar projetos específicos, outros *Scratch*ers ou até mesmo recursos didáticos. Isso facilita a navegação no vasto universo do *Scratch*.

- **Perfil do Usuário:** localizado no canto superior direito, o perfil do usuário é seu espaço pessoal no *Scratch*. Ao fazer login, você pode gerenciar seus projetos, interagir com mensagens e personalizar sua experiência no *Scratch*. Cada perfil possui uma área de armazenamento em nuvem, assegurando que seus projetos sejam salvos e sincronizados automaticamente enquanto você cria e explora.

Estes recursos, juntos, compõem o ambiente do *Scratch*, tornando-o não apenas uma plataforma de programação, mas também uma comunidade onde se pode aprender, compartilhar e crescer criativamente.

Figura 21. Página Inicial do *Scratch*.



Fonte: Scratch

Para começar a utilizar o *Scratch*, o primeiro passo é se registrar no site. Um detalhe é que o *Scratch* oferece a possibilidade de associar várias contas a um único endereço de e-mail, o que é particularmente útil para pais ou educadores que pretendem configurar contas para crianças ou estudantes que ainda não possuem um e-mail próprio. Depois de estabelecer a conta, selecione a opção “Criar” no menu principal para ingressar na versão online da ferramenta e dar início ao desenvolvimento de projetos interativos e educativos.

A interface do *Scratch* (Figura 22) é intuitiva e projetada para facilitar o aprendizado de programação de forma visual e interativa. Aqui está um guia detalhado:

- **Área de Menus (no topo, à esquerda)**: Aqui você encontra opções para configurar o seu projeto. Os menus “Configurações” e “Arquivo” permitem que você salve, carregue ou comece um novo projeto, enquanto “Editar” oferece ferramentas para desfazer/refazer ações.

- **Abas de Categoria (centro à esquerda):** Estas abas, como “Código”, “Fantasias” e “Sons”, permitem que você alterne entre escrever o código do projeto, criar ou editar os visuais dos personagens (chamados de sprites) e adicionar efeitos sonoros.

- **Blocos de Código (centro à esquerda, abaixo das abas):** Aqui estão os blocos de código que você pode arrastar e soltar para formar scripts. Eles são categori-

zados por cor e função, como “Movimento”, “Aparência”, “Som”, “Eventos”, entre outros.

- **Área de *Scripts* (centro):** Este é o espaço onde você encaixa os blocos de código para programar os comportamentos dos sprites. Você pode arrastar os blocos da paleta de blocos e encaixá-los aqui para construir seu código.

- **Área de Visualização (centro à direita):** É o palco onde você pode ver suas criações ganharem vida. O sprite selecionado executará as ações programadas aqui.

- **Área dos *Sprites* (canto inferior direito):** Mostra todos os sprites que você tem em seu projeto. Você pode adicionar novos sprites, mudar entre eles para programar comportamentos específicos, ou alterar seus visuais.

- **Controles de Projeto (canto superior ao centro):** Incluem botões para iniciar ou parar a execução do projeto, possibilitando testar o que você programou.

- **Informações do Sprite (direita):** Aqui você pode ver e alterar as propriedades do sprite selecionado, como sua posição no palco, tamanho e direção.

- **Mochila (inferior):** Um espaço de armazenamento pessoal onde você pode guardar blocos de código, sprites e sons favoritos ou frequentemente usados.

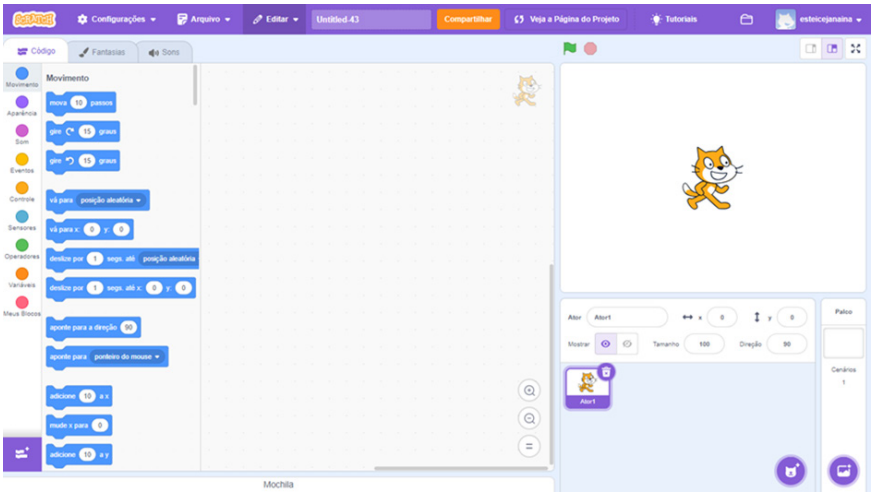
- **Botão Compartilhar (No centro superior, à direita):** Permite que você compartilhe seu projeto com a comunidade *Scratch*, tornando-o público para outros usuários verem e interagirem.

- **Botão “Veja a Página do Projeto” (No centro superior, à direita):** Este botão redireciona para a página web do seu projeto, onde outras pessoas podem ver e interagir com ele.

- **Tutoriais (canto superior direito):** Acesso rápido a uma série de tutoriais que podem ajudar a entender melhor como usar a ferramenta e aprender conceitos de programação.

O *Scratch* é desenhado para ser um ambiente onde a experimentação é incentivada, assim, explorar e interagir com cada um desses elementos é a melhor maneira de aprender e se familiarizar com a ferramenta. Ele está disponível em mais de 80 idiomas e você pode alterar nas Configurações.

Figura 22. Interface da versão on-line do *Scratch*.



Fonte: *Scratch*

A interface do *Scratch* é semelhante à estrutura de uma peça de teatro, o que a torna uma analogia perfeita para explicar seus conceitos a crianças que estão começando a aprender programação:

- **Palco:** assim como em uma peça de teatro, onde a ação ocorre em um palco, no *Scratch* o palco é a área de visualização onde os sprites (atores) se movem e interagem.

- **Sprites:** correspondem aos atores em uma peça. No *Scratch*, os sprites são os personagens ou objetos que realizam ações, movem-se pelo palco e reagem a eventos, muito como atores seguindo um roteiro.

- **Fantasias:** da mesma forma que os atores podem mudar de figurino, os sprites podem mudar suas "fantasias" para aparecer diferentes em várias cenas ou para mostrar diferentes estados emocionais ou ações.

- **Sons:** enquanto em uma peça os sons de ambiente ou música de fundo vêm de fora do palco, no *Scratch* os sons de fundo devem ser programados para serem tocados pelo próprio palco, e não pelos sprites, para representar sons ambientais.

- **Roteiro de blocos:** os blocos de código no *Scratch* são como o script de uma peça. Eles direcionam o que os sprites fazem e como interagem com o ambiente e outros sprites.

- **Direção e movimento:** da mesma maneira que um diretor indica onde um ator deve se mover no palco, os blocos de movimento no *Scratch* direcionam os sprites para onde devem ir.

- **Cenas:** assim como uma peça de teatro tem diferentes cenas, o *Scratch* permite que você mude o pano de fundo para criar diferentes cenas ou ambientes para seus sprites.

Ao utilizar essa analogia, crianças podem transferir sua compreensão de uma atividade familiar, como assistir ou participar de uma peça de teatro, para entender e criar projetos interativos no *Scratch*. Isso facilita o aprendizado, permitindo que elas façam conexões diretas entre os elementos visuais da ferramenta e os conceitos de programação subjacentes.

Possibilidades no *Scratch*

O *Scratch* é uma ferramenta multifacetada com várias aplicações. A ferramenta possui inúmeros recursos que facilitam o aprendizado individual de programação. Portanto, detalhar cada bloco de programação se torna desnecessário neste capítulo, pois os próprios usuários podem explorar e aprender de forma autônoma.

Professores podem utilizar o *Scratch* para criar jogos, animações e histórias interativas para uso didático. Seguindo as recomendações de design de Interação Criança-Computador, como descrito em um dos meus artigos, é possível desenvolver recursos que são não apenas instrutivos, mas também atraentes para crianças. A incorporação de elementos como *affordance*, *feedback* adequado, fácil navegação, gestos intuitivos, uso de sons, cores vibrantes, tipografia legível e personagens envolventes pode tornar o aprendizado mais prazeroso e efetivo. As interfaces que aderem a esses princípios de design são mais atraentes para o público infantil e contribuem para uma experiência educacional mais rica e engajadora.

Além de promover o aprendizado individualizado e apoiar os educadores no desenvolvimento de conteúdos didáticos, o *Scratch* pode ser empregado em uma variedade de contextos educacionais para fomentar habilidades relacionadas ao pensamento computacional. Por exemplo, ao enfrentar desafios de programação e solucionar problemas dentro da ferramenta, os estudantes podem desenvolver habilidades essenciais como abstração, análise lógica, decomposição de problemas e reconhecimento de padrões, todos elementos centrais do pensamento computacional.

Aqui estão algumas experiências em que implementei o *Scratch* ao longo dos últimos 10 anos:

- **Curso de programação para Pedagogia:** conduzi um curso voltado para estudantes de Pedagogia, onde o foco foi o desenvolvimento de jogos educativos para crianças do Ensino Fundamental I. Os estudantes foram desafiados a aplicar os conceitos de programação para criar jogos que não apenas entretêm, mas também promovem o aprendizado interativo. Esta experiência foi particularmente valiosa para mostrar como a programação pode ser integrada de maneira efetiva em práticas pedagógicas

- **Maratona de programação com *Scratch*:** realizei uma maratona de programação para alunos envolvidos em um curso extracurricular de robótica. A dinâmica da maratona incentivou os alunos a pensar criticamente e a resolver problemas de maneira colaborativa, aplicando a lógica de programação de forma criativa e engajada.

- **Competição de jogos e animações:** Organizei um evento durante a Semana de Ciência e Tecnologia onde os alunos do Ensino Fundamental I criaram jogos e animações relacionados ao tema do evento. Esses projetos foram submetidos a uma competição online, permitindo que toda a comunidade escolar participasse na votação, criando um ambiente de engajamento e aprendizado compartilhado.

- **Narrativas digitais com *ScratchJr*:** Utilizei o *ScratchJr* para ajudar alunos do primeiro ano do Ensino Fundamental a contar histórias. O *ScratchJr*, adaptado para crianças que ainda não desenvolveram plenamente habilidades de leitura e escrita, provou ser uma ferramenta que auxilia a estimular a criatividade e expressão individual.

- **Oficinas de jogos durante Semanas de Ciência e Tecnologia:** Liderei oficinas para alunos de cursos técnicos em Agricultura e Informática, com ênfase no uso do *Scratch* para criar jogos que exploram conceitos de ciência e tecnologia. Essas oficinas proporcionaram uma plataforma para os estudantes aplicarem o conhecimento teórico de maneira prática e interativa.

- **Introdução à programação no Ensino Médio:** Integrei o *Scratch* nas disciplinas de programação do ensino médio para apresentar conceitos fundamentais de uma forma acessível, incentivando o interesse e a confiança dos alunos em suas habilidades de codificação.

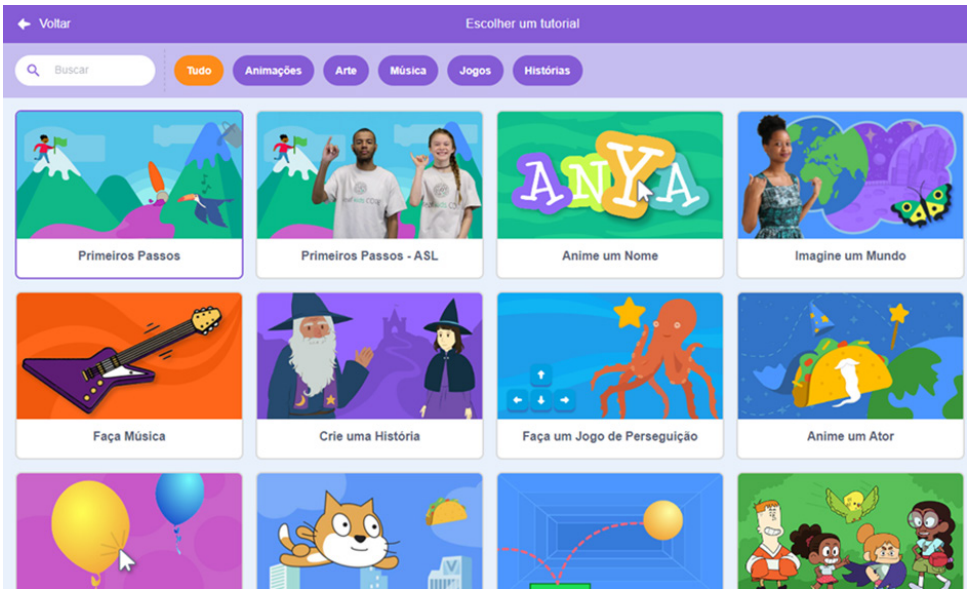
- ***Scratch* e Eletrônica com S4A e *TinkerCad*:** Integrei o *Scratch* com o mundo da eletrônica, utilizando ferramentas como *Scratch* para Arduino (S4A) e, mais recentemente, *TinkerCad*. Isso abriu um novo espectro de possibilidades para os alunos, que puderam ver suas criações ganharem vida no mundo físico, através da programação e simulação de circuitos.

Cada uma dessas experiências destacou a flexibilidade do *Scratch* como ferramenta de ensino e aprendizado, oferecendo aos alunos a oportunidade de desenvolver habilidades digitais cruciais de uma maneira envolvente e significativa.

Introdução Prática à Programação com *Scratch*

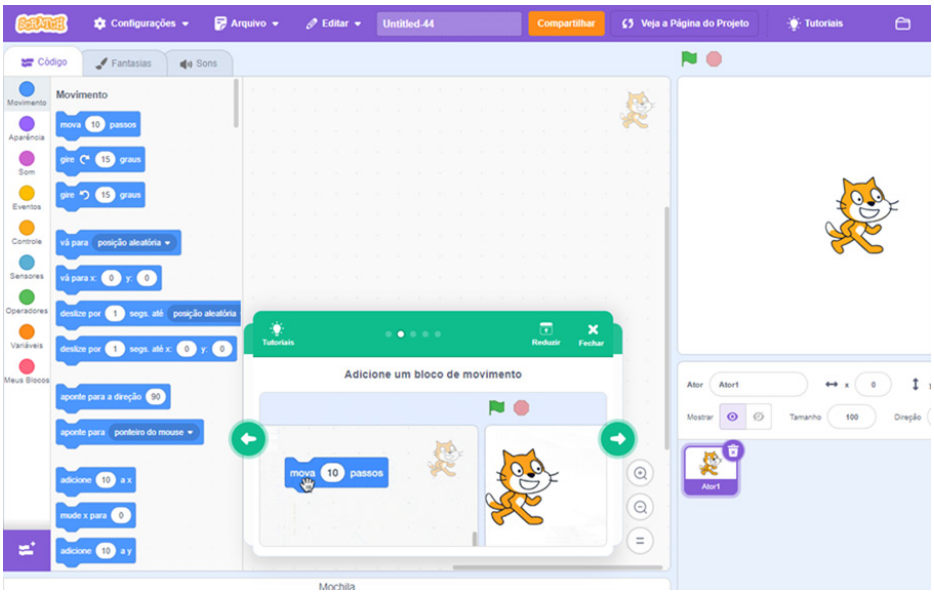
Para uma iniciação eficaz no mundo da programação através do *Scratch*, aconselho a utilizar os tutoriais interativos incorporados na plataforma (Figura 23). Com mais de 20 tutoriais disponíveis, você pode escolher um que desperte seu interesse e seguir um guia detalhado, passo a passo, diretamente no ambiente de programação do *Scratch* (Figura 24). Esses tutoriais são projetados para fornecer uma compreensão prática e imersiva dos diversos blocos de código e suas funcionalidades.

Figura 23. Tutoriais interativos do *Scratch*.



Fonte: *Scratch*

Figura 24. Execução de um dos tutoriais do *Scratch*



Fonte: *Scratch*

Além disso, recomendo explorar a playlist no YouTube associada a este livro, onde você encontrará uma série de vídeos explicativos. Cada vídeo é dedicado a desvendar os conceitos fundamentais por trás dos blocos de *Scratch*, proporcionando um aprofundamento do seu conhecimento em programação e permitindo que você explore as possibilidades criativas da ferramenta com maior confiança e compreensão. Esses recursos visuais e interativos são especialmente úteis para reforçar o aprendizado e incentivar a experimentação autônoma na criação de projetos pessoais e educativos.

Possibilidades com o *ScratchJr*

Para iniciar crianças pré-letradas no mundo da programação, o *ScratchJr* é uma opção especialmente desenhada para este público. Essa versão simplificada do *Scratch* foi criada para se adaptar às habilidades cognitivas e motoras de crianças entre 5 e 7 anos, que ainda estão no início do seu processo de alfabetização.

O *ScratchJr* facilita a programação com uma interface intuitiva, onde os blocos de comando são representados por ícones facilmente reconhecíveis, eliminando a necessidade de leitura de texto. As crianças podem arrastar e soltar blocos gráficos para construir suas próprias histórias e jogos, estimulando a criatividade e o pensamento lógico. Ao manipular os blocos, elas aprendem os fundamentos da programação, como sequência e controle de fluxo, de uma forma lúdica e acessível.

O aplicativo incentiva a aprendizagem por meio da exploração e do jogo, alinhando-se com os princípios pedagógicos do construcionismo. Além disso, o *ScratchJr* oferece um ambiente seguro para as crianças experimentarem e construírem projetos sem a necessidade de navegar na internet ou gerenciar contas online.

Ao utilizar o *ScratchJr*, educadores e pais podem introduzir conceitos computacionais básicos e habilidades de resolução de problemas de uma maneira que é ao mesmo tempo educativa e divertida, proporcionando uma base sólida para futura aprendizagem em STEM (ciência, tecnologia, engenharia e matemática).

Capítulo 6
Desenvolvimento de narrativas digitais

Narrativas são a essência da comunicação humana e desempenham um papel central na forma como entendemos e nos conectamos com o mundo ao nosso redor. Com a evolução da tecnologia, a narrativa também se transformou, adaptando-se ao ambiente digital para se tornar mais interativa e imersiva. Este capítulo explora como as ferramentas de programação em blocos, especialmente através de plataformas como o *Scratch*, podem ser empregadas para desenvolver narrativas digitais complexas e envolventes.

Abordaremos técnicas avançadas de narrativa em ambientes digitais, focando na maneira como os elementos narrativos podem ser manipulados para criar histórias, jogos e animações interativas. Exploraremos como diferentes componentes de uma narrativa — tais como enredo, personagens, cenário, e a sequência temporal dos eventos — podem ser habilmente entrelaçados para produzir experiências narrativas ricas e memoráveis.

Ao longo deste capítulo, discutiremos exemplos práticos e ofereceremos orientações detalhadas sobre como usar as ferramentas disponíveis no *Scratch* para que educadores e estudantes possam construir suas próprias narrativas digitais, desde a concepção inicial até a execução final. Vamos mergulhar nas possibilidades criativas que a programação em blocos oferece para transformar ideias em histórias digitais cativantes que podem ser compartilhadas e experienciadas por uma audiência global.

Elementos de uma narrativa digital no *Scratch*

Na programação em blocos, como no *Scratch*, a criação de narrativas digitais para jogos, histórias e animações envolve a manipulação de vários componentes essenciais. Esses componentes ajudam a estruturar uma narrativa de maneira que ela seja envolvente e interativa. Vamos explorar cada um desses elementos e como eles se aplicam ao contexto de programação em blocos:

- **Personagens:** são os protagonistas da narrativa. No *Scratch*, os personagens podem ser representados por sprites (figuras animadas), que os usuários podem programar para se mover, falar e interagir. Cada sprite pode ter múltiplas fantasias, permitindo mudanças de aparência que podem simbolizar diferentes estados emocionais ou ações.

- **Cenários:** correspondem ao fundo onde a ação ocorre. No *Scratch*, os cenários podem ser alterados para refletir diferentes locais ou momentos da história. Essa

mudança de cenário pode ser programada para acontecer em resposta a ações dos personagens ou a interações do usuário, ajudando a avançar a narrativa.

- **Enredos:** são a espinha dorsal da narrativa, consistindo na sequência de eventos que contam uma história. No *Scratch*, o enredo é construído através da sequência lógica dos blocos de código que determinam como os personagens e cenários interagem e respondem às ações do usuário. Os eventos podem incluir diálogos, conflitos, resoluções e outros elementos narrativos chave.

- **Diálogos e textos:** essenciais para o desenvolvimento do enredo e dos personagens, os diálogos no *Scratch* podem ser criados usando blocos que fazem os personagens "falar" ou "pensar". Textos podem aparecer na tela para fornecer contexto adicional ou instruções ao usuário.

- **Interações:** no *Scratch*, as interações são programadas para responder a entradas do usuário, como cliques do mouse ou pressionamento de teclas, que podem fazer a narrativa avançar ou mudar de direção. Essas interações são cruciais para jogos e animações interativas, onde o usuário tem um papel ativo na história.

- **Áudio:** sons e músicas são utilizados para enriquecer a atmosfera e dar vida à narrativa. No *Scratch*, os usuários podem adicionar efeitos sonoros e músicas de fundo que podem ser ativados por ações específicas dos personagens ou mudanças no cenário.

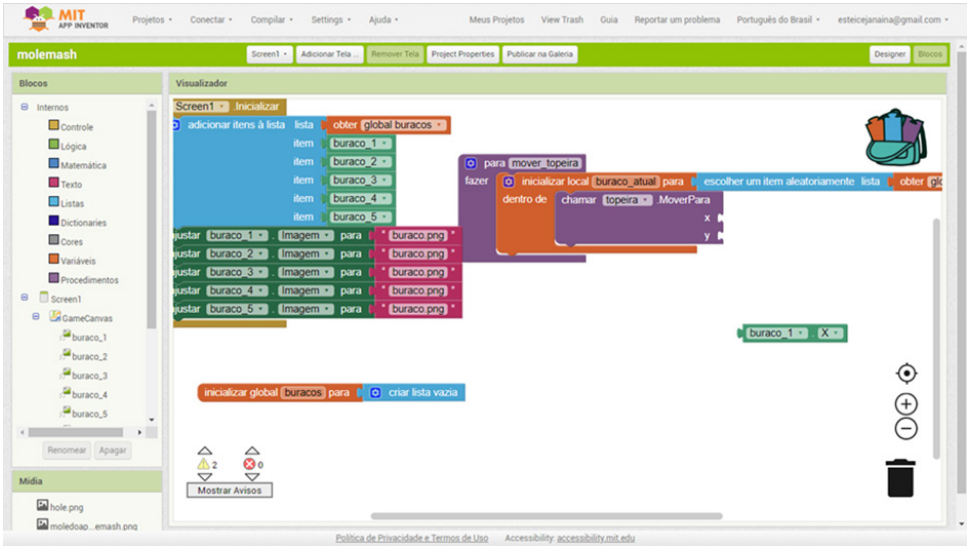
- **Transições e efeitos visuais:** efeitos como fades, cortes e mudanças visuais podem ser programados para indicar mudanças de cena ou destacar momentos importantes dentro da história. Esses efeitos ajudam a criar um ritmo para a narrativa e a manter o interesse visual do espectador.

Estes componentes combinados permitem que educadores e estudantes criem experiências narrativas ricas e envolventes que vão além do texto tradicional, explorando a interatividade e a multimídia para contar histórias de maneiras novas e emocionantes.

Elaborando narrativas digitais no *AppInventor*

O *App Inventor* é uma plataforma de desenvolvimento de aplicativos móveis visual e intuitiva criada pelo MIT. Ela permite que usuários de todos os níveis de habilidade, especialmente iniciantes e estudantes, criem aplicativos para dispositivos Android sem a necessidade de escrever código em linguagens de programação tradicionais. Em vez disso, o App Inventor usa uma abordagem de programação em blocos, semelhante ao *Scratch*, que facilita a visualização da lógica e das funções do aplicativo (Figura 25).

Figura 25. Interface do *AppInventor* para a programação do aplicativo

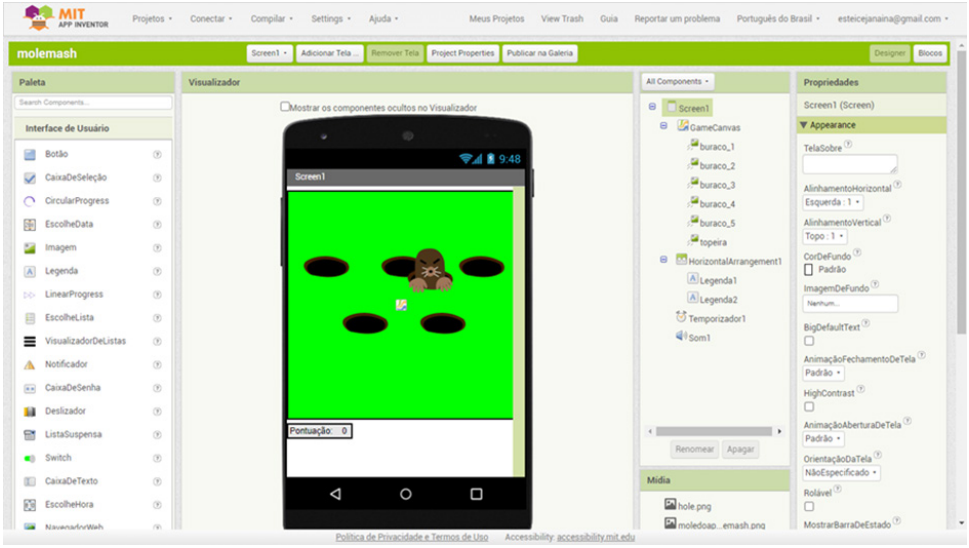


Fonte: Appinventor

Dessa forma, os usuários constroem a lógica dos aplicativos arrastando blocos visuais de programação para criar o fluxo e as operações do aplicativo. Cada bloco representa diferentes funções, como manipulação de dados, controle de entrada do usuário ou integração de serviços de hardware do dispositivo como GPS e câmera.

A plataforma possui uma seção específica para o design da interface do usuário (Figura 26), onde os desenvolvedores podem adicionar e organizar componentes como botões, textos, imagens e outros elementos interativos.

Figura 26. Interface do *AppInventor* para o design do aplicativo



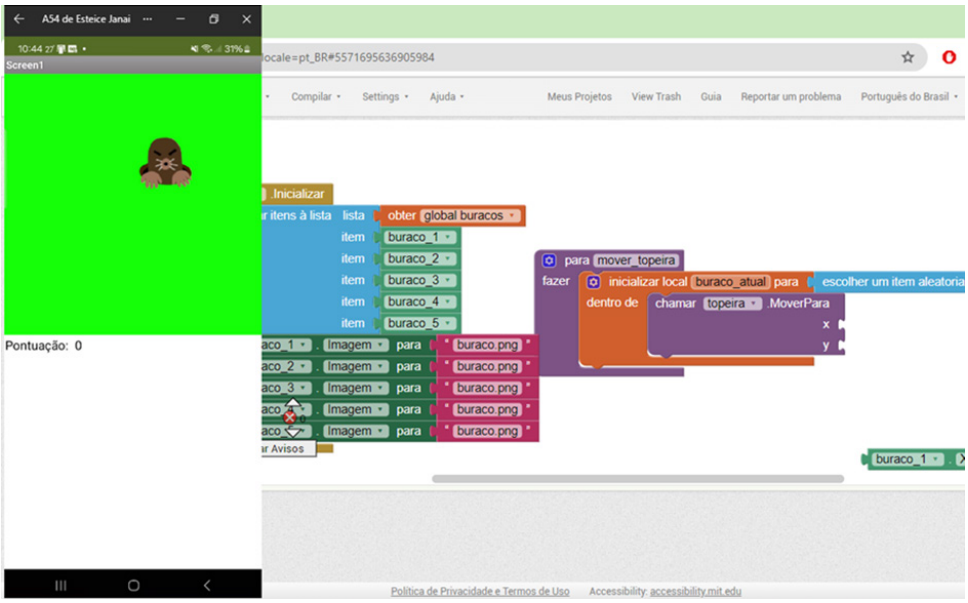
Fonte: Appinventor

O App Inventor oferece a possibilidade de testar aplicativos em tempo real usando um emulador incorporado ou diretamente em um dispositivo Android conectado (Figura 27). Isso permite uma iteração rápida durante o processo de desenvolvimento.

Na playlist do YouTube, que serve como uma extensão deste livro, há uma série de vídeos dedicados a explicar o uso do App Inventor. Esses vídeos são projetados para complementar o conteúdo escrito, proporcionando uma experiência de aprendizado mais rica e interativa. Eles cobrem desde os fundamentos básicos até aspectos mais avançados do desenvolvimento de aplicativos com o App Inventor, incluindo tutoriais passo a passo, dicas de design de interface do usuário e estratégias para otimizar a funcionalidade dos aplicativos. Essa abordagem visual e prática é ideal para aqueles que preferem aprender de forma dinâmica e aplicada.

Ao criar uma aplicação que incorpora elementos de narrativas digitais no AppInventor, você pode construir histórias interativas, jogos e animações que aproveitam os recursos do dispositivo móvel. Vamos explorar como os componentes de uma narrativa digital podem ser implementados no AppInventor:

Figura 27. Emulador para testar as aplicações em tempo real



Fonte: Appinventor

Personagens: No AppInventor, personagens podem ser representados através de imagens ou botões personalizados que respondem a interações do usuário. Você pode programar esses elementos para executar ações ou mudar de estado conforme a narrativa avança. Por exemplo, um personagem pode falar ou mudar de expressão quando o usuário toca na tela.

Cenários: Os cenários são configurados como telas diferentes dentro do aplicativo. Cada tela pode representar um local diferente da história. O AppInventor permite a transição entre telas, o que pode ser usado para avançar a narrativa, mudando o cenário conforme os personagens se movem através da história.

Enredos: O enredo em um aplicativo AppInventor é controlado por eventos e ações que são definidos pelos blocos de programação. Estes eventos podem ser desencadeados por interações do usuário, como toques ou movimentos, ou podem ocorrer automaticamente como parte da lógica do aplicativo. O fluxo do enredo pode incluir diálogos, pontos de decisão que alteram o curso da narrativa, e outros elementos dinâmicos.

Diálogos e textos: Os diálogos podem ser implementados usando caixas de texto ou diálogos pop-up que aparecem em resposta a ações do usuário. O AppInventor permite a fácil implementação de entrada e saída de texto, o que facilita a inclusão de diálogos interativos na narrativa.

Interações: As interações no AppInventor são principalmente baseadas em toques e gestos. Essas interações podem ser programadas para influenciar a narrativa, como escolhas feitas pelo usuário que afetam o desenrolar da história ou o desempenho em um jogo.

Áudio: O AppInventor suporta a inclusão de arquivos de áudio, que podem ser usados para adicionar música de fundo, efeitos sonoros ou diálogos falados. O áudio pode ser programado para tocar em momentos específicos, enriquecendo a experiência narrativa e aumentando a imersão.

Transições e efeitos visuais: Embora o AppInventor não ofereça tantas opções de efeitos visuais quanto plataformas mais avançadas, ele permite algumas transições básicas entre telas e mudanças visuais que podem ser usadas para enfatizar aspectos da narrativa.

Para finalizar

A capacidade de contar histórias evoluiu com o advento da tecnologia digital, proporcionando novos meios para expressar criatividade e compartilhar experiências. Utilizando ferramentas de programação em blocos, como *Scratch* e App Inventor, educadores e alunos podem explorar o vasto potencial das narrativas digitais. Estas plataformas permitem uma abordagem acessível e interativa para a criação de histórias, jogos e animações, permitindo aos usuários não apenas consumir conteúdo, mas também se tornarem criadores ativos.

O desenvolvimento de narrativas digitais através dessas ferramentas oferece uma oportunidade inestimável para integrar habilidades de pensamento crítico, criatividade e

competências técnicas, essenciais no século XXI. Ao praticar a arte de contar histórias em um formato digital, os estudantes aprendem a articular suas ideias, trabalhar em colaboração e pensar de maneira inovadora, enfrentando desafios e resolvendo problemas de forma criativa.

Além disso, as narrativas criadas podem servir como pontes culturais, permitindo que histórias de diversas origens e perspectivas sejam compartilhadas e apreciadas por uma audiência global. Isso não só enriquece o aprendizado dentro da sala de aula, mas também fora dela, preparando os alunos para serem cidadãos informados e empáticos.

Encorajamos educadores a utilizar as técnicas e ferramentas discutidas neste capítulo para desbloquear o potencial criativo dos alunos e mergulhar no mundo enriquecedor das narrativas digitais. À medida que continuam a explorar e a se adaptar às novas tecnologias, a habilidade de contar histórias de forma eficaz e envolvente será uma ferramenta valiosa em muitos aspectos de suas vidas, acadêmicas, profissionais e pessoais.

Assim, convidamos você, leitor, a iniciar sua jornada no desenvolvimento de narrativas digitais, explorando o poder das histórias para conectar, educar e inspirar. Mãos à obra, e que cada projeto seja uma nova história a ser descoberta e contada.

Capítulo 7
Análise e Planejamento de Soluções

Nos capítulos anteriores, detalhamos os pilares fundamentais do Pensamento Computacional (PC) e como eles auxiliam indivíduos para enfrentar e resolver problemas complexos de forma eficaz. Com uma compreensão profunda de conceitos como decomposição, reconhecimento de padrões, abstração e desenvolvimento de algoritmos, estamos prontos para aplicar essas habilidades em situações práticas.

Neste capítulo, vamos avançar além da teoria e nos concentrar em como o Pensamento Computacional pode ser diretamente aplicado no planejamento de soluções para desafios do mundo real. Inicialmente, abordaremos as estratégias gerais que podem ser usadas para analisar problemas. A seguir, vamos propor o desenvolvimento de um plano sistemático para solucioná-los, utilizando os quatro pilares do PC de maneira integrada e prática.

A partir de agora, nossa jornada nos levará através de um processo interativo de análise e planejamento, onde cada passo do Pensamento Computacional será demonstrado e utilizado para fornecer soluções eficazes.

Entendendo o Problema

Na Escola Futuro Brilhante, localizada em uma região urbana com uma diversidade socioeconômica significativa, observou-se um padrão preocupante de baixo desempenho em matemática entre os alunos dos anos finais do Ensino Fundamental e do Ensino Médio. As avaliações internas e os resultados de exames nacionais indicaram que uma grande proporção desses alunos não está atingindo os níveis de proficiência esperados em matemática. Professores relatam que muitos estudantes demonstram desinteresse pela matéria, além de ansiedade significativa quando confrontados com problemas matemáticos complexos.

Os educadores e a administração da escola notaram que os métodos tradicionais de ensino não estão sendo suficientemente eficazes para engajar esses alunos. Isso inclui uma abordagem que muitas vezes é muito abstrata e não relaciona os conceitos matemáticos com situações do mundo real que sejam relevantes para os estudantes. Além disso, identificou-se que a falta de recursos didáticos interativos e tecnológicos contribui para o desinteresse e a dificuldade em compreender conceitos matemáticos fundamentais.

Esse cenário causou preocupação entre os educadores e a administração da escola, pois reconhece-se que a competência em matemática é essencial para o sucesso acadêmico

dos alunos e para sua capacidade de aproveitar futuras oportunidades educacionais e profissionais, especialmente nas áreas de STEM. A escola está, portanto, em busca de soluções inovadoras que possam melhorar o ensino de matemática e, consequentemente, o desempenho dos alunos nesta disciplina essencial.

Propondo uma Solução com Pensamento Computacional

Ao enfrentar o problema do baixo engajamento dos alunos do Ensino Médio nas aulas de Matemática no Colégio Educacional Futuro Brilhante, podemos aplicar os quatro pilares do Pensamento Computacional (PC) para desenvolver uma solução eficaz, como um aplicativo ou jogo educacional.

1. Decomposição

Para abordar o problema de baixo desempenho em matemática na Escola Futuro Brilhante usando a decomposição, vamos detalhar o processo em várias etapas claras e gerenciáveis.

Primeiro vamos aprofundar o conhecimento sobre as preferências pessoais dos alunos e as áreas específicas de dificuldade em matemática, a partir das seguintes atividades:

- Conduzir pesquisas e grupos focais com os alunos para identificar seus interesses, tipos de jogos que preferem, e como gostam de aprender.
- Analisar os resultados acadêmicos para detectar quais tópicos matemáticos apresentam os maiores desafios.
- Realizar avaliações diagnósticas para identificar lacunas específicas no conhecimento matemático de cada aluno.

Posteriormente, precisamos determinar o formato de jogo mais adequado para engajar os alunos, com base em suas preferências e necessidades educacionais. As seguintes atividades são necessárias:

- Revisar diferentes gêneros de jogos educativos (como aventura, puzzle, simulação) e avaliar qual estilo seria mais atraente para a faixa etária em questão.
- Consultar especialistas em design de jogos educativos para compreender quais mecânicas de jogo são mais eficazes no ensino de matemática.
- Escolher um formato de jogo que não apenas interesse os alunos, mas também se alinhe com os objetivos educacionais do currículo de matemática.

Para garantir um jogo engajador e relevante, é necessário envolver os alunos no processo de desenvolvimento do jogo para garantir que o produto final seja de qualidade. Abaixo há uma lista de atividades que podem ser realizadas para essa finalidade.

- Organizar oficinas de co-criação com alunos e professores para esboçar elementos do jogo.
- Utilizar prototipagem rápida para testar conceitos e iterar baseados no feedback dos alunos.
- Desenvolver um protótipo inicial do jogo, incorporando os conceitos matemáticos que necessitam de reforço.

As últimas atividades envolverão avaliar a eficácia do jogo desenvolvido e fazer ajustes conforme necessário:

- Implementar uma fase piloto onde uma amostra de alunos utiliza o jogo em um ambiente controlado.
- Coletar dados sobre o engajamento dos alunos e sua performance em matemática antes e depois de usar o jogo.
- Ajustar o jogo com base no feedback dos alunos e na observação de sua performance, otimizando os aspectos educacionais e de entretenimento.

Este processo detalhado não só descomplica o desafio inicial, mas também garante que a solução desenvolvida seja motivadora para os alunos, abordando diretamente suas necessidades e preferências.

2. Reconhecimento de Padrões

Ao aplicar o reconhecimento de padrões nas atividades abaixo, podemos maximizar a eficiência do desenvolvimento e da revisão do jogo educativo, garantindo que seja tanto engajador quanto pedagogicamente sólido, otimizando recursos e tempo nos processos de desenvolvimento e avaliação.

- Identificar quais áreas de matemática são universalmente desafiadoras para os alunos. Para isso, podemos compilar e analisar os dados de desempenho dos alunos em avaliações e testes diagnósticos para identificar tendências comuns de erros ou conceitos mal compreendidos. Reconhecer esses padrões ajuda a priorizar esses conteúdos no design do jogo.
- Avaliar a eficácia do jogo em melhorar o desempenho matemático dos alunos. Nesse passo, é importante coletar feedback quantitativo e qualitativo dos alunos

sobre sua experiência com o jogo, e posteriormente analisar os dados de pré e pós-teste dos alunos para detectar melhorias significativas ou falta de progresso em suas habilidades matemáticas.

- Reconhecer padrões nas preferências de jogo e engajamento dos alunos pode ajudar a personalizar elementos gamificados que mais motivam cada aluno. Se determinados tipos de recompensas ou desafios são mais eficazes para certos alunos, o jogo pode adaptar esses elementos para manter cada aluno motivado e engajado.

- Implementar sistemas de feedback baseados no reconhecimento de padrões de respostas e erros dos alunos. Se o jogo percebe que um aluno continua cometendo um tipo específico de erro, pode oferecer dicas personalizadas ou ajustar a dificuldade dos desafios para melhor atender às necessidades desse aluno.

- Na programação do jogo, podemos identificar repetições de código no desenvolvimento do jogo. Ao reconhecer esses padrões, podemos escrever partes do código de forma modular e reutilizável, economizando tempo de desenvolvimento e tornando o código mais limpo e fácil de manter. Isso também nos permite aplicar soluções comprovadas em novos contextos, contribuindo para a consistência e eficiência do jogo como um todo.

3. Abstração

A aplicação da abstração será útil em diversas atividades no desenvolvimento da solução, entre elas:

- Abstrair os conceitos matemáticos em desafios ou quebra-cabeças dentro do jogo, tornando-os mais acessíveis e atraentes para os alunos.

- No projeto do jogo, pode-se usar abstrações para ocultar detalhes de implementação complicados e focar na funcionalidade geral. Isso permite concentrar-se nos aspectos educacionais e de entretenimento do jogo, em vez de preocupar-se com todos os detalhes técnicos subjacentes.

- Identificar componentes comuns dentro do jogo que podem ser abstraídos em módulos reutilizáveis. Por exemplo, pode-se criar uma abstração para representar desafios matemáticos, permitindo que esses desafios sejam facilmente adicionados e modificados ao longo do desenvolvimento do jogo.

4. Desenvolvimento de Algoritmos

Ao incorporar algoritmos no desenvolvimento do jogo educacional, podemos criar uma experiência mais personalizada, adaptável e eficaz para os alunos, ajudando-os a melho-

rar suas habilidades matemáticas de forma significativa e motivadora. Aqui estão algumas maneiras de como podemos aplicar algoritmos:

- Definir a lógica do jogo, incluindo como os desafios matemáticos são apresentados aos jogadores, como as pontuações são calculadas, e como o progresso é rastreado. Por exemplo, podemos desenvolver algoritmos que gerem aleatoriamente problemas matemáticos com base no nível de dificuldade do jogador.

- Personalizar a experiência de cada aluno com base em seu desempenho e preferências. Por exemplo, podemos desenvolver algoritmos de adaptação que ajustam a dificuldade dos desafios com base no progresso do aluno, garantindo que eles sejam desafiados de forma adequada.

- Analisar as respostas dos alunos e fornecer feedbacks personalizados. Isso pode incluir a detecção de padrões de erros recorrentes e a oferta de dicas ou explicações relevantes para ajudar os alunos a superar suas dificuldades.

- Analisar os dados coletados durante o jogo, como o desempenho dos alunos e os padrões de uso. Isso pode nos fornecer insights valiosos sobre a eficácia do jogo e identificar áreas que precisam de melhorias.

Atividades futuras

Compreendemos de maneira prática e detalhada como o Pensamento Computacional pode ser aplicado para desenvolver soluções inovadoras para problemas complexos no mundo real. Através do exemplo da Escola Futuro Brilhante, demonstramos como desdobrar um desafio educacional em uma oportunidade de inovação pedagógica, utilizando os quatro pilares do Pensamento Computacional: decomposição, reconhecimento de padrões, abstração e desenvolvimento de algoritmos.

Ao avançar, encorajamos você a aplicar estes princípios ao desenvolver sua própria solução para o problema apresentado, utilizando ferramentas de programação em blocos. Este exercício não só solidificará seu entendimento do material, mas também lhe proporcionará a oportunidade de criar uma ferramenta educacional que poderá ter um impacto positivo significativo.

Além disso, desafiamos você a identificar outros problemas dentro de seu domínio de conhecimento ou comunidade em que a aplicação do Pensamento Computacional seja benéfica. Pense em como você pode utilizar as técnicas aprendidas para abordar questões em diversos campos, seja na gestão de negócios, na saúde, no meio ambiente ou em qualquer outra área que requeira soluções criativas e eficientes.

Importante lembrar que o desenvolvimento de uma solução é apenas o começo. A fase de testes e o refinamento contínuo com base no feedback são essenciais para garantir que a solução não apenas resolva o problema de forma eficaz, mas também se aprimore e se adapte às mudanças nas necessidades e condições. Este processo iterativo de melhoria contínua é fundamental para o sucesso a longo prazo de qualquer projeto.

Finalmente, enquanto você avança na implementação da solução escolhida e na exploração de novos desafios, lembre-se de manter uma postura de aprendizado contínuo e de abertura para o feedback.

Palavras finais

À medida que chegamos ao final desta obra, espero que as páginas anteriores tenham servido não apenas como um guia para o Pensamento Computacional, mas também como um estímulo para sua aplicação prática. A capacidade de pensar como um computador — decompondo problemas, reconhecendo padrões, abstraindo e criando algoritmos — é mais do que uma habilidade técnica; é uma forma de entender e moldar o mundo ao nosso redor.

Este livro visou desmistificar o Pensamento Computacional, apresentando-o de forma acessível e aplicável. Seja você um educador procurando integrar novas técnicas em sala de aula, um profissional buscando eficiência em seus projetos, ou simplesmente um curioso sobre como a tecnologia pode melhorar a vida das pessoas, espero que as discussões aqui propostas tenham sido enriquecedoras.

Encorajo cada um de vocês a levar adiante o conhecimento adquirido aqui, aplicando-o de maneiras novas e criativas. O mundo está em constante mudança e as ferramentas do Pensamento Computacional são essenciais para nos adaptarmos e prosperarmos neste cenário dinâmico.

Para continuar explorando ideias e discussões sobre o Pensamento Computacional e muito mais, convido você a me seguir no Instagram em https://www.instagram.com/esteicejanaina/. Vamos juntos inspirar novas descobertas e inovações.

Obrigada por embarcar nesta jornada comigo. Que as ideias compartilhadas neste livro inspirem novas descobertas e inovações.

Referências

BATISTA, Esteic Janaina S. et al. Utilizando o *Scratch* como ferramenta de apoio para desenvolver o raciocínio lógico das crianças do ensino básico de uma forma multidisciplinar. In: **Anais do XXI Workshop de Informática na Escola**. SBC: 2015. p. 350-359.

BATISTA, Esteic Janaina Santos. **Uma análise de ambientes de programação em blocos com base em recomendações de interação criança-computador**. Trabalho de Conclusão de Curso. Universidade Federal de Mato Grosso do Sul. Campo Grande: 2017. Disponível em: https://link.ufms.br/1eRNA. Acesso em 01 abr. 2024.

BATISTA, Esteic Janaina Santos; SILVA, Camila; LIMA, Anderson. Abordagem de recomendações de design da interação criança-computador no curso de formação de professores em uma linguagem de programação visual em blocos. **In: Anais do XXIII Workshop de Informática na Escola**. SBC: 2017. **p. 835-844**.

BELL, Tim; WITTEN, Ian H.; FELLOWS, Michael. CS **Unplugged:** an enrichment and extension programme for primary-aged students. [S.l.]: CS Unplugged, 2015. Disponível em: https://link.ufms.br/EIuVN. Acesso em: 01 abr. 2024.

BRACKMANN, Christian Puhlmann. **Desenvolvimento do pensamento computacional através de atividades desplugadas na educação básica**. Tese. Universidade Federal do Rio Grande do Sul, Porto Alegre: 2017.

BRASIL. **Lei nº 14.533, de 11 de janeiro de 2023. Política Nacional de Educação Digital**. Diário Oficial da União: Edição extra, Brasília: 22 dez. 2023. Disponível em: https://link.ufms.br/fkGhc. Acesso em: 01 abr. 2024.

BRASIL. Ministério da Educação. **Parecer CNE/CEB nº 2/2022. Normas sobre Computação na Educação Básica – Complemento à Base Nacional Comum Curricular (BNCC)**. Brasília: 17 fev. 2022. Disponível em: https://link.ufms.br/CHlWx. Acesso em: 01 abr. 2024.

BRASIL. Ministério da Educação. **Anexo ao Parecer CNE/CEB nº 2/2022 - BNCC Computação**. Brasília: 17 fev. 2022. Disponível em: https://link.ufms.br/YWQn4. Acesso em: 01 abr. 2024.

BRASIL. Ministério da Educação. **Resolução CNE/CEB nº 1, de 4 de outubro de 2022. Normas sobre Computação na Educação Básica – Complemento à BNCC**. Diário Oficial da União, Brasília, DF, 6 out. 2022, Seção 1, p. 33. Disponível em: https://link.ufms.br/DgR0G. Acesso em: 01 abr. 2024.

OLIMPÍADA BRASILEIRA DE ROBÓTICA. **Gabarito:** nível 4, fase 1. 2022. Disponível em: https://link.ufms.br/L0jDw. Acesso em: 01 abr. 2024.

OLIVEIRA, Pedro Wachsmann Schanzer de. **Ensino da computação na educação básica**. Trabalho de Conclusão de Curso. Porto Alegre: 2022. Disponível em: https://link.ufms.br/NMkWg. Acesso em: 01 abr. 2024.

PAPERT, Seymour A. ***Mindstorms*:** children, computers, and powerful ideas. Basic Books, 1980.

PAPERT, Seymour. **A máquina das crianças:** repensando a escola na era da informática. 1994.

PAPERT, Seymour; SOLOMON, Cynthia. Twenty things to do with a computer. **Educational Technology Magazine**, 1972. Disponível em: https://link.ufms.br/rQxyh. Acesso em: 01 abr. 2024.

RESNICK, Mitchel. Give P'sa chance: Projects, peers, passion, play. In: **Constructionism and creativity:** proceedings of the third international constructionism conference. Austrian computer society, Viena: 2014. p. 13-20.

REZENDE, Flavia. As novas tecnologias na prática pedagógica sob a perspectiva construtivista. **Ensaio:** Pesquisa em Educação em Ciências. Belo Horizonte:, v. 2, p. 70-87, 2000.

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

WING, J. M. ***Computational Thinking*:** what and why? 17 out. 2010. Disponível em: https://link.ufms.br/s20k6. Acesso em: 01 abr. 2024.

WORLD ECONOMIC FORUM. **Future of Jobs 2023**. Davos: 2023. Disponível em: https://link.ufms.br/hGSK9. Acesso em: 01 abr. 2024.

AGEAD
Agência de Educação
Digital e a Distância