INSTITUTO FEDERAL DE ALAGOAS
PROGRAMA DE PÓS - GRADUAÇÃO EM DOCÊNCIA NA EDUCAÇÃO PROFISSIONAL
CURSO DE ESPECIALIZAÇÃO EM FORMAÇÃO PEDAGÓGICA PARA PROFESSORES DA EDUCAÇÃO PROFISSIONAL

GUILHERME JOSÉ DE CARVALHO CAVALCANTI

SCRATCH COMO FERRAMENTA PEDAGÓGICA PARA O ENSINO DE PROGRAMAÇÃO NO CURSO TÉCNICO INTEGRADO DE MEIO AMBIENTE NO INSTITUTO FEDERAL DE ALAGOAS CAMPUS PENEDO

PENEDO, AL
2021

GUILHERME JOSÉ DE CARVALHO CAVALCANTI

SCRATCH COMO FERRAMENTA PEDAGÓGICA PARA O ENSINO DE PROGRAMAÇÃO NO CURSO TÉCNICO INTEGRADO DE MEIO AMBIENTE NO INSTITUTO FEDERAL DE ALAGOAS CAMPUS PENEDO

Trabalho de conclusão de curso apresentado ao Programa de Pós-Graduação em Docência na Educação Profissional do Instituto Federal de Alagoas, como requisito parcial para obtenção de título de Especialista em Docência na Educação Profissional.

Orientador: Prof. Dr. Tiago de Moraes Lenz

PENEDO, AL
2021

Dados Internacionais de Catalogação na Publicação
Instituto Federal de Alagoas
Campus Penedo
Biblioteca

C376s

Cavalcanti, Guilherme José de Carvalho.

Scratch como ferramenta pedagógica para o ensino de programação no curso técnico integrado de meio ambiente no Instituto Federal de Alagoas Campus Penedo / Guilherme José de Carvalho Cavalcanti. – 2021.

35f.; il.

Orientação: Prof. Tiago de Moraes Lenz.

Trabalho de Conclusão de Curso (Especialização em Educação Profissional), Educação à Distância, Instituto Federal de Alagoas Penedo, Penedo, 2021.

1. Programação. 2. Scratch. 3. Educação Profissional e Tecnológica. I. Lenz, Tiago de Moraes. II. Título.

CDD: 005.133

Maria Luzia Alexandre de Oliveira
Bibliotecária/Documentalista
CRB-4/2159

GUILHERME JOSÉ DE CARVALHO CAVALCANTI

SCRATCH COMO FERRAMENTA PEDAGÓGICA PARA O ENSINO DE PROGRAMAÇÃO NO CURSO TÉCNICO INTEGRADO DE MEIO AMBIENTE NO INSTITUTO FEDERAL DE ALAGOAS CAMPUS PENEDO

Trabalho de conclusão de curso apresentado ao Programa de Pós-Graduação em Docência na Educação Profissional do Instituto Federal de Alagoas, como requisito parcial para obtenção de título de Especialista em Docência na Educação Profissional.

APROVADO(A) EM: 09 / 12 / 2021.

BANCA EXAMINADORA

Prof. Dr. Tiago de Moraes Lenz
Instituto Federal de Alagoas - IFAL

Fabrícia de Almeida Cortez Pereira

Profa. Dra. Fabricia De Almeida Cortez Pereira
Instituto Federal de Alagoas - IFAL

Mayra T A Santos

Profa. Me. Mayra Tais Albuquerque Santos
Instituto Federal de Alagoas – IFAL

RESUMO

O tópico "Programação de Computadores" foi introduzido na mais recente ementa da disciplina de Informática, ministrada no primeiro ano do curso técnico integrado de Meio Ambiente do Instituto Federal de Alagoas (IFAL), Campus Penedo. Esta atualização do Projeto Pedagógico do Curso (PPC) foi aprovada no final de 2019, entrando em vigor no ano letivo de 2020. No entanto, historicamente, pesquisadores têm documentado muitas dificuldades enfrentadas por alunos no aprendizado de conceitos básicos de programação. Para contornar essas dificuldades, ambientes de programação visuais surgem como uma alternativa para o ensino programação a esses novatos e provaram ter sucesso, melhorando a compreensão e o aprendizado de alunos iniciantes. Scratch é um desses ambientes visuais de programação, o qual foi utilizado nesse estudo para comparar a aprendizagem entre o ambiente visual e a linguagem de programação textual tradicional a partir de uma sequência didática. Os alunos do IFAL, Campus Penedo, foram introduzidos à programação de computadores através de aulas com Scratch e aulas com linguagem de programação textual tradicional. Posteriormente, os alunos foram instruídos a desenvolver um projeto de programação utilizando o Scratch e responder um questionário a respeito de suas experiências e percepções. Nossos resultados mostram que Scratch facilitou a compreensão de conceitos básicos de programação pelos alunos. Além disso, os alunos, em sua maioria, alegaram que se sentiram motivados para o estudo programação e afirmaram que teriam interesse em um curso completo e aprofundado sobre o tema. Finalmente, quando comparados Scratch e a linguagem textual, Scratch apresentou resultados que sugerem que é mais apto para o ensino de programação de computadores.

Palavras-chave: Ensino e aprendizagem de programação; Programação visual; Scratch.

ABSTRACT

Computer programming was introduced in the most recent syllabus of the discipline of Informatics, given in the first year of the Environment technical high school at the Federal Institute of Alagoas (IFAL), Campus Penedo. Historically, however, researchers have documented several difficulties that students face in learning basic programming concepts. To overcome these difficulties, visual programming environments emerge as an alternative for teaching programming to these novices and they have proven to be successful. Scratch is one of those visual programming environments that are widely used for this purpose. In this study, therefore, we investigate the use of Scratch in teaching programming to students at IFAL, Campus Penedo. Students took programming classes with Scratch as well as with a traditional textual programming language. Then, they had to develop a programming project using Scratch. Finally, they were invited to answer a survey about their experiences and perceptions. Our results show that Scratch made it easier for students to understand basic programming concepts. In addition, most students claimed that they felt motivated to study programming and stated that they would be interested in a complete and in-depth programming course. Finally, when comparing Scratch and textual language, Scratch presented results suggesting that it is a better choice for teaching programming.

Keywords: Teaching programming; Visual programming; Scratch.

SUMÁRIO

1 INTRODUÇÃO

O uso de computadores e produtos computacionais estão incorporados mundialmente ao cotidiano da maioria das pessoas. No entanto, as pessoas são tipicamente consumidoras de produtos prontos, mas pouco participam do processo de criação desses produtos. Cada vez mais se busca uma mudança de paradigma a respeito da utilização de computadores. Partindo-se de um paradigma onde as pessoas são meras consumidoras, para um paradigma onde passam a utilizar o potencial de criação e desenvolvimento dos computadores, por meio da programação de computadores. Aprender a programar computadores permite o desenvolvimento do chamado Pensamento Computacional (do inglês, *Computacional Thinking*), isto é, a habilidade de resolver problemas, e entender o comportamento humano baseando-se em conceitos de ciência da computação (WING, 2006).

O pensamento computacional pode ser desenvolvido através do ensino de programação, e seus benefícios não se restringem mais somente às áreas de tecnologia. O ensino de programação, outrora típico do ensino superior em cursos relacionados à informática, vem sendo inserido na grade curricular dos mais diversos cursos em diferentes níveis educacionais. Quanto mais cedo se inicia o ensino de programação de computadores, melhor a aprendizagem do pensamento computacional (KAZAKOFF *et al.*, 2013; MCCARTNEY *et al.*, 2014; MCCARTNEY *et al.*, 2015; HEINTZ *et al.*, 2016; PAPADAKIS *et al.*, 2016). No Instituto Federal de Alagoas, Campus Penedo, a programação de computadores passou a ser parte da ementa da disciplina de Informática que entrou em vigor em 2020. A disciplina é ministrada aos discentes de primeiro ano do curso técnico integrado de Meio Ambiente, público alvo do presente estudo. No entanto, aprender programação é difícil, e o seu ensino visto como uma área problemática. Observa-se que conceitos básicos de programação, como estruturas de controle e repetição, são de difícil compreensão para uma proporção significativa dos alunos, quando estes entram em contato pela primeira vez com o assunto. Nenhuma metodologia de ensino parece beneficiar todos os alunos, com alguns alunos podendo facilmente compreender um conceito descrito em uma linguagem de programação textual (ex. Java) e outros apresentarem dificuldades, exigindo outras formas de ilustração dos conceitos, estes com carga cognitiva menor ou diferente (STACHEL *et al.*, 2013).

As dificuldades no processo ensino-aprendizagem de programação são as mais diversas. Seja pela metodologia de ensino dominante, que ainda é tipicamente a expositiva com os alunos passivamente recebendo informações, estas desconexas dos interesses e valores dos alunos. Seja pelos métodos de estudo, onde os alunos estão acostumados com disciplinas onde o sucesso é possível através de leituras seguidas por memorização. E, finalmente, pela própria natureza da disciplina, que é muito dinâmica, demanda muito raciocínio lógico e treinamento. Como consequência, a desmotivação dos alunos acerca da disciplina resulta em altas taxas de reprovação e, fatalmente, evasão de alunos (DENNING; MCGETTRICK, 2005; MAY; DHILLON, 2009). No entanto, essas observações são típicas de cursos voltados à área de tecnologia (por exemplo, Ciência da Computação, ou no contexto do ensino profissional, um curso técnico em Informática), onde se espera que a maioria dos discentes tenham aptidão pela área, pois estão ali por interesse próprio. No entanto, essa não é a realidade do público-alvo desse trabalho, os quais não são alunos de um curso de tecnologia, apresentam dificuldades de acesso aos computadores e pouquíssima habilidade de manejo de um computador. São alunos que entram em contato com o assunto por imposição, já que o assunto está inserido na ementa de uma disciplina que eles são obrigados a cursar. Portanto, é de se esperar que esses problemas aconteçam e até mesmo se agravem no decorrer do curso.

Para mitigar esse problema, educadores têm investigado diferentes estratégias para

tornar programação mais acessível aos alunos (JENKINS, 2002). Muito desse esforço foi direcionado para criar melhores ambientes de programação para iniciantes, resultando em muitas novas abordagens (FINCHER; PETRE, 2004; KELLEHER; PAUSCH, 2005). Um recurso comum em muitos ambientes modernos de programação é o uso de blocos de código de "arrastar e soltar", que se encaixam juntos para formar um programa, minimizando a possibilidade de erros de sintaxe e a necessidade de memorização. Nesse contexto, nós investigamos o uso do *Scratch* como ferramenta de ensino de programação para os alunos de primeiro ano do curso técnico integrado de Meio Ambiente no Instituto Federal de Alagoas, campus Penedo. Scratch foi elaborado para facilitar o aprendizado de conceitos de programação e Ciência da Computação (MEERBAUM-SALANT *et al.*, 2013). Mostrou-se ser fácil de começar, capaz de construir projetos complexos e possível abordar uma variedade de tópicos e temas (RESNICK *et al.*, 2009), especialmente no nível do ensino médio. Scratch tem sido recomendado para programadores iniciantes por sua facilidade de uso, bem como para programadores avançados por sua facilidade em construir projetos complexos (MALAN; LEITNER, 2007). Em particular, esse estudo investiga a eficácia do Scratch (uma linguagem baseada em blocos) para o ensino de programação. Em síntese, a investigação deu-se com as seguintes etapas:

- Demonstração e realização de algumas atividades simples de programação usando o Scratch com os alunos;
- Comparação entre programas gráficos construídos com Scratch e programas equivalentes construídos com linguagem de programação textual tradicional (foi escolhida a linguagem Java nesse estudo);
- Preenchimento de um breve questionário online, pelos alunos, sobre suas experiências e percepções.

Na seção 2, é apresentada a fundamentação teórica. A metodologia de pesquisa, resultados e discussão são apresentadas das seções 3 e 4, respectivamente. A seção 5 apresenta alguns trabalhos relacionados, e a seção 6 conclui o trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Pensamento Computacional não é um termo novo, havendo referencias na década de 1950 ao "pensamento algorítmico". Foi definido como uma maneira de usar *algoritmos* [1] para produzir um resultado apropriado para uma determinada entrada/estímulo (DENNING, 2009). Na prática, o objetivo não é substituir o pensamento criativo e crítico pelo pensamento computacional, mas adicionar a habilidade de usar computadores e algoritmos para resolver problemas (CUNY *et al.*, 2010), através da programação de computadores. Essas habilidades sãos enfatizadas pelo governo de diferentes nações, que afirmam a necessidade de, cada vez mais cedo, jovens tornarem-se fluentes na linguagem digital, em vez de serem meros usuários de software de computador (PEÑALVO *et al.*, 2016). Aprender a programar computadores pode induzir a mudanças na maneira de como as pessoas pensam (RESNICK, 1995). Isso se deve à natureza analítica do pensamento computacional, que é bastante semelhante ao pensamento matemático (ou seja, resolução de problemas), pensamento de engenharia (design e avaliação de processos) e pensamento científico (análise sistemática).

No entanto, aprender a programar computadores não é uma conquista fácil para iniciantes. A aprendizagem dos alunos em relação à programação são estudadas há décadas (KOOHANG, 1989) e demonstram gerar frustração e ansiedade nos alunos (RAUB, 1981). Para tentar contornar esses problemas, a literatura apresenta vários estudos sobre métodos para facilitar o ensino de programação. Uma conclusão desses estudos é que a *linguagem de programação* [2] adotada impacta na compreensão. Por exemplo, *Java* é um das linguagens mais comuns empregadas em cursos de introdução à programação, no entanto, suas peculiaridades dificultam o aprendizado (PENDERGAST, 2006). Isso ocorre porque estudantes sem qualquer exposição prévia à programação precisam simultaneamente aprender as técnicas para resolver um problema, e o uso de uma linguagem de programação como uma ferramenta para resolver o problema. Essas múltiplas demandas podem levar a uma sobrecarga cognitiva (MERRIENBOER; SWELLER, 2005) para um iniciante.

Por outro lado, há evidências de que linguagens com maior nível de abstração melhoram a retenção e compreensão dos alunos (NIKULA *et al.*, 2007; GRAY *et al.*, 2007). Como tal, ambientes visuais para programação podem ser empregados para ajudar programadores iniciantes a superar as dificuldades no aprendizado de programação (MOOR; DEEK, 2006). Esses ambientes foram projetados para evitar erros comuns de iniciantes em programação como erros de sintaxe e lógica. Portanto, em vez de digitar comandos, como em uma linguagem de programação textual tradicional como Java, a maioria desses ambientes usam linguagens de programação baseadas em blocos. Nessas linguagens, cada bloco é um elemento da linguagem de programação: uma estrutura de controle, um operador, uma variável, uma função, entre outros. Esses elementos podem ser combinados com "arrastar e soltar"de forma intuitiva de acordo com a lógica planejada para a construção de determinado programa. *Scratch* é uma linguagem com essas características.

---

[1] Sequência lógica, finita e definida de instruções/passos que devem ser seguidas para a realização de alguma tarefa.

[2] Conjunto de comandos, instruções e sintaxe usada para criar um programa. Vocabulário utilizado para escrever algoritmos.

2.1 SCRATCH

Scratch foi criado pelo Massachusetts Institute of Technology (MIT) em colaboração com a Universidade da Califórnia, mas não é o primeiro ambiente de programação e linguagem destinada a programadores iniciantes. Na verdade, pesquisas indicam um rico histórico de diferentes ambientes desenvolvidos para esse propósito (FINCHER; PETRE, 2004; KELLEHER; PAUSCH, 2005). Scratch baseia-se nas ideias de outro ambiente, o Logo (CAMPBELL *et al.*, 1986), mas substitui a digitação de código pela abordagem de "arrastar e soltar"inspirada em LogoBlocks (BEGEL, 1996) e EToys (SCHÖN *et al.*, 1998). Scratch tem como foco a manipulação da mídia e oferece suporte a atividades de programação que se alinham com os interesses dos mais jovens, como criação de histórias animadas, jogos e apresentações interativas. Scratch é destinado à educação pré-universitária, mas inclui todos os elementos básicos de uma linguagem de programação de propósito geral (como estruturas de repetição e controle, entre outros). No entanto, Scratch, embora possa ser utilizada por programadores mais avançados, não apresenta alguns recursos mais sofisticados de programação como herança e outros tipos de abstração. Scratch está disponível gratuitamente, tem uma grande base de usuários, boa documentação e vários exemplos de projetos que podem ser visualizados.

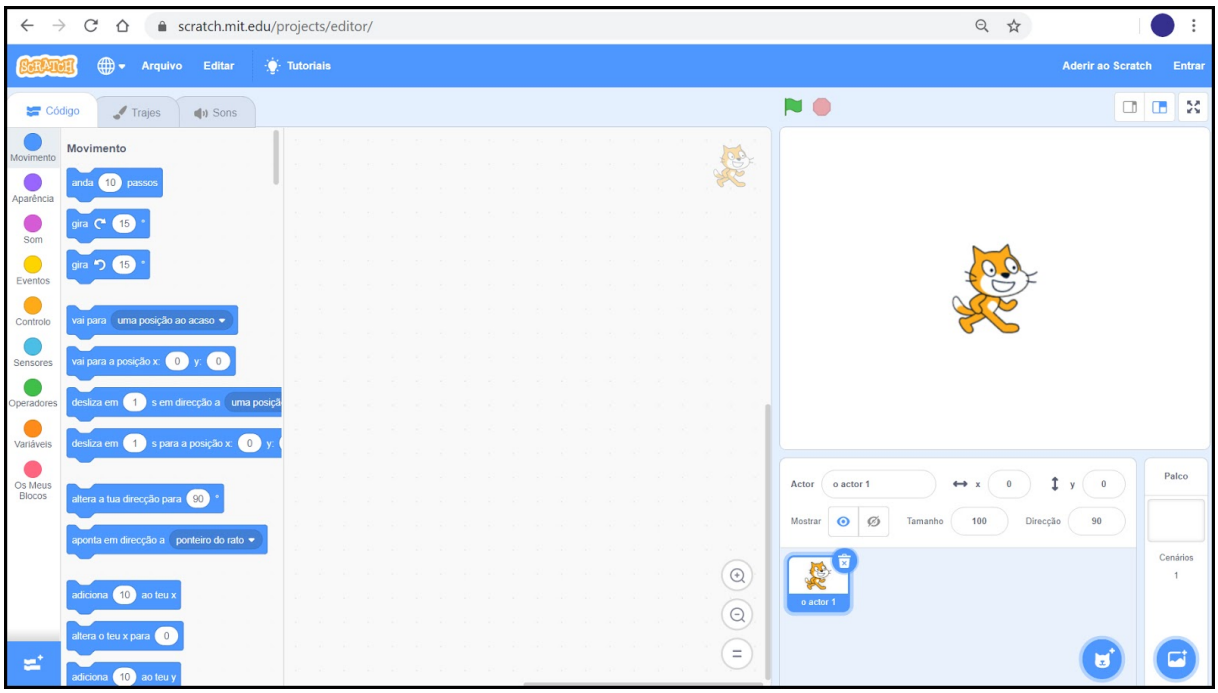
O ambiente Scratch consiste em um palco fixo (plano de fundo) e uma série de objetos móveis. Cada objeto contém seu próprio conjunto de imagens, sons, variáveis e ações. A programação é feita arrastando e juntando blocos de comando pela tela, como peças de quebra-cabeça, criando-se pilhas de blocos. Um bloco individual ou uma pilha de blocos pode ser executado clicando duas vezes nele. Os blocos também podem ser acionados como resposta a algum evento, como a inicialização do programa, uma determinada tecla do teclado sendo pressionada, ou um clique do mouse. Vários blocos podem ser executados ao mesmo tempo, e possuem determinada forma geométrica de modo que eles podem apenas ser conectados a outros blocos caso os desenhos combinem (ou se encaixem, como no jogo Tetris[3]). Na prática, em termos de programação, isso significa que o encaixe só é possível caso o bloco resultante corresponda a uma declaração sintaticamente válida em programação. Além disso, os blocos que englobam outros blocos, como blocos que representam as estruturas de repetição e condicionais, redimensionam-se dinamicamente para incluir outros blocos à medida que são adicionados ou removidos. A gama de blocos disponíveis é bastante extensa.

A tela Scratch (Figura 1) é dividida em quatro áreas. À direita encontra-se o palco, um botão na barra acima do palco permite a exibição em tela inteira. Abaixo do palco encontra-se uma área que mostra miniaturas de todos os objetos do projeto. Clicar em uma dessas miniaturas seleciona o objeto correspondente. O painel do meio permite ao usuário visualizar e alterar os blocos de comandos, trajes (imagens) ou sons do objeto selecionado. No painel mais à esquerda encontram-se os blocos que podem ser arrastados para a área de blocos de comandos. Este design de interface com o usuário do Scratch surgiu do desejo de tornar os o conceitos chaves do Scratch o mais acessível possível. Tendo a área de blocos de comandos sempre visível estimula a exploração e curiosidade. Quando um usuário encontra um comando de seu interesse, basta clicar para ver o que ele faz. Conforme uma ação se desenrola no palco, o usuário pode assistir a como seus blocos de comandos são executados. Esse ideal de exploração se dá graças à presença do painel de seleção de blocos de comandos, área de blocos escolhidos, e palco simultaneamente visíveis, passando uma ideia de como as ações são interpretadas pelo computador.

O Scratch apresenta cerca de noventa comandos, incluindo comandos para movimento de objetos, posicionamento cartesiano, transformações de imagem (rotação, redimensionamento, etc), animações, reprodução de som, entre outros. Como muitos desses comandos são baseados

[3] https://pt.wikipedia.org/wiki/Tetris

Figura 1 – Captura de tela do Scratch.



Fonte: online, disponível em https://bit.ly/3jFCisy.

em operações aritméticas, Scratch permite ao usuário um desenvolver sua prática matemática. Por exemplo, usar um número negativo no comando que se refere à movimentação dos objetos faz com que ele ande para trás. Atualmente, são suportadas operações aritméticas, comparativas e lógicas. Há também blocos de comandos que permitem detectar quando um objeto toca a borda da tela, ou então quando toca em outro objeto, e, até mesmo, uma cor específica. Há também blocos de detecção que relatam a localização do mouse, a tecla pressionada no teclado, e muitos outros.

3 METODOLOGIA

Nesse estudo, nós investigamos o uso do *Scratch*, em comparação à tradicional linguagem de programação textual *Java*, para o ensino de programação para alunos de primeiro ano do curso técnico integrado em Meio Ambiente do Instituto Federal de Alagoas, campus Penedo. São alunos, em sua grande maioria, com 15-16 anos de idade, ingressantes do curso no ano letivo de 2020. Esse estudo ocorreu durante o quarto bimestre da disciplina de Informática, componente curricular obrigatório para os alunos desse curso. Em suma, o estudo consistiu de:

- Aulas online (em virtude da pandemia de COVID-19) mostrando conceitos de comandos, variáveis, estruturas de controle, e estruturas de repetições em Scratch e também na linguagem de programação textual Java;
- Projeto de programação realizada pelos alunos com Scratch;
- Preenchimento voluntário de questionário acerca das experiências e percepções pelos alunos.

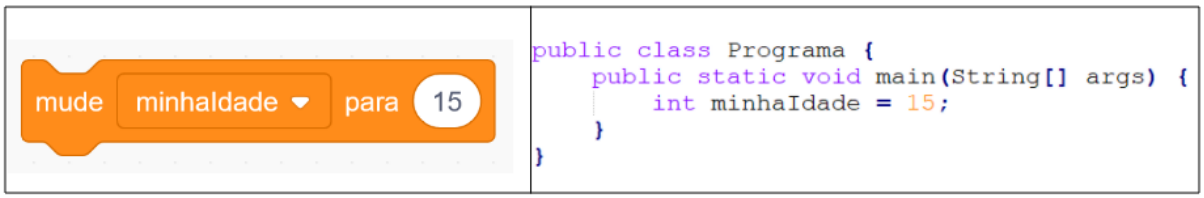
3.1 AULAS

Como possivelmente este foi o primeiro contato dos alunos com o mundo da programação de computadores, foi necessário fornecer-lhes uma base teórica na qual puderam basear sua aprendizagem e percepções a respeito da programação e tecnologias empregadas, como o próprio Scratch. Essa base teórica foi estabelecida através de aulas e ilustrações de conceitos básicos de programação.

Para investigar a comunicabilidade de conceitos básicos de programação através de ambas as linguagens de programação baseada em blocos (Scratch) e linguagem textual (Java), foram ministradas aulas cobrindo quatro elementos básicos de programação, que podem ser encontrados nas mais importantes e populares linguagens de programação. Para que os alunos fossem expostos a duas tecnologias empregadas no ensino de programação, e, consequentemente, pudessem desenvolver critérios de comparação, os conteúdos foram apresentados em Scratch e na linguagem de programação textual Java. Java foi escolhida devido à expertise do docente, que possui mais de uma década de experiência na linguagem. A seguir, os conceitos de programação explorados nas aulas em detalhes:

1. **Sequências**: demonstrar como um programa consiste em um conjunto de etapas ou instruções que são executadas uma de cada vez na ordem em que são escritas. Na programação, a sequência é um algoritmo básico: um conjunto de etapas lógicas realizadas em ordem. Os computadores precisam de instruções na forma de um algoritmo para concluir uma tarefa desejada, e esse algoritmo deve seguir uma ordem correta de etapas ou sequência.
2. **Variáveis**: demonstrar como variáveis são declaradas, nomeadas e utilizadas para armazenar valores como números, texto etc. que são usados em um programa. Uma variável é uma forma de se referir a uma área de armazenamento em um programa de computador, e esse local da memória contém valores - números, texto ou outros tipos mais complexos de dados (ver Figura 2).

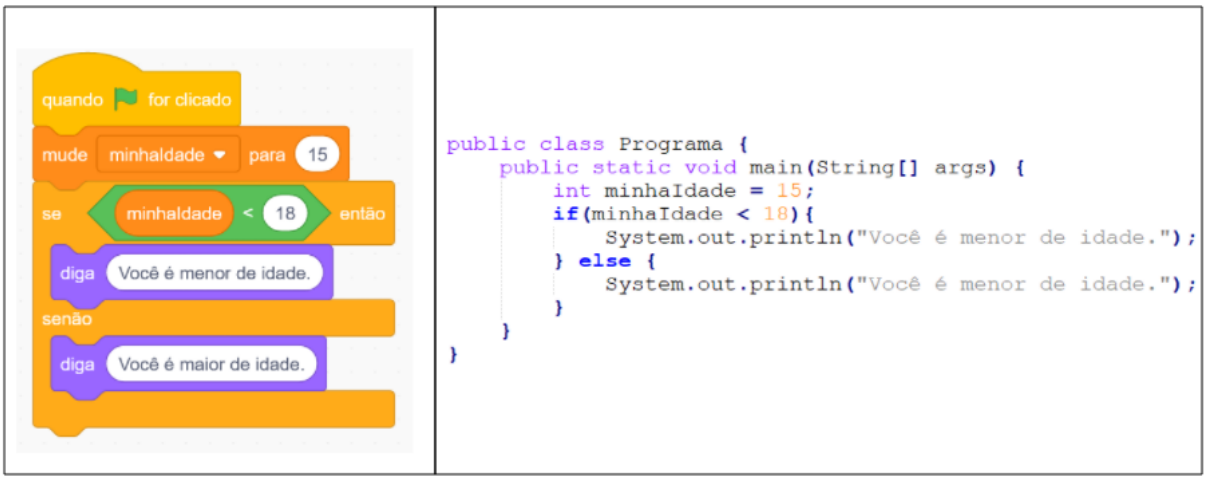
Figura 2 – Variável em Scratch e Java.



Fonte: autoria própria.

3. **Estruturas de Controle**: demonstrar como construir estruturas que permitem um ponto de escolha entre diferentes conjuntos de instruções subsequentes a serem executados em um programa. Frequentemente, um programa de computador deve fazer escolhas sobre como proceder, por exemplo, "se a senha digitada estiver correta, faça uma coisa, senão, faça algo diferente."As estruturas condicionais são os elementos mais básicos que permitem esse processo de tomada de decisão na programação (ver Figura 3).

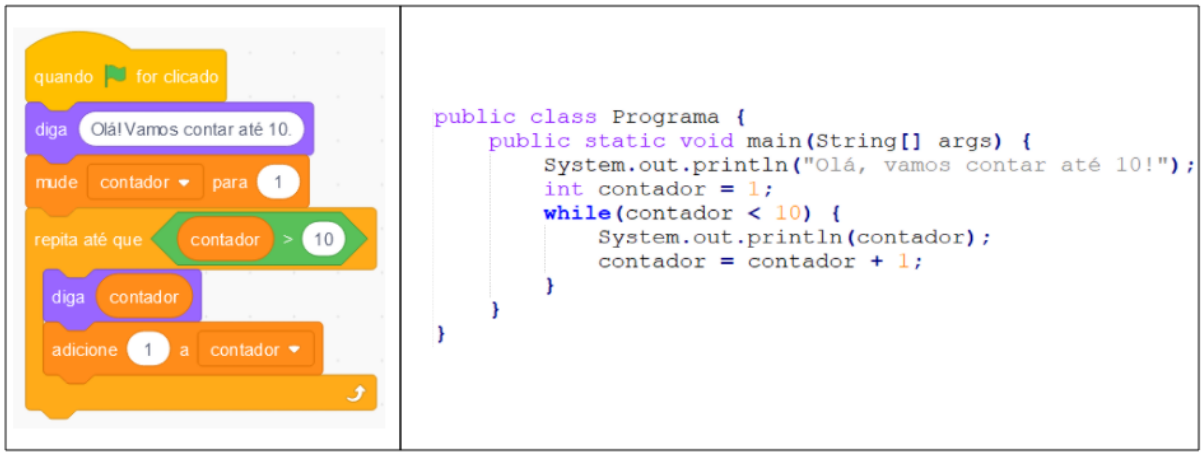
Figura 3 – Controle em Scratch e Java.



Fonte: autoria própria.

4. **Estruturas de repetição**: demonstrar como permitir que um conjunto de instruções seja executado várias vezes, repetindo-as quantas vezes forem necessário. As estruturas de repetição repetem uma parte do código fonte do programa um determinado número de vezes até que o processo desejado seja concluído. Tarefas repetitivas são comuns em programação, e as estruturas de repetição são essenciais para economizar tempo e minimizar erros (ver Figura 4).

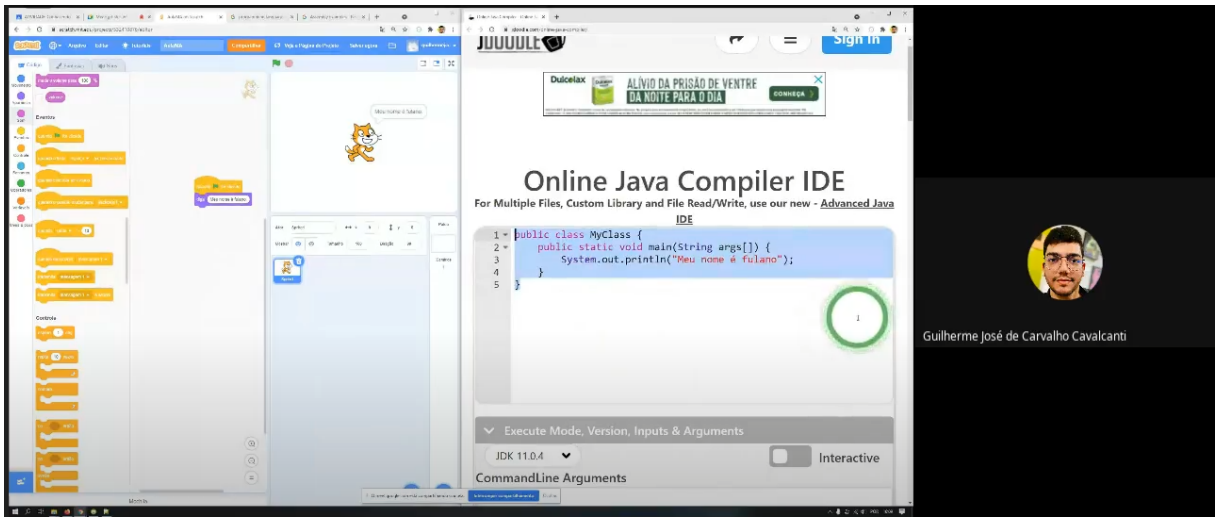
Figura 4 – Repetição em Scratch e Java.



Fonte: autoria própria.

Foram realizadas um total de cinco aulas, com duração de 1H cada. As aulas foram conduzidas de forma online (síncrona) pelo docente da disciplina, através da ferramenta *Google Meet*[1] (ver Figura 5), para todos os alunos matriculados na disciplina de Informática, do primeiro ano do curso técnico integrado de Meio Ambiente, que ainda não haviam desistido do curso ou da disciplina. Essas restrições deveram-se à pandemia de COVID-19 e consequente regimento interno vigente na instituição, chamado de Ensino Remoto Emergencial.

Figura 5 – Captura de tela de aula realizada via Google Meet.



Fonte: autoria própria.

3.2 PROJETO DE PROGRAMAÇÃO

Na última aula, os alunos foram instruídos a trabalhar em um Projeto Scratch, que contou como nota do quarto bimestre da disciplina de Informática. Eles deveriam exercitar sua criatividade e também demonstrar seu aprendizado, escrevendo jogos usando os blocos disponíveis no Scratch. Os alunos trabalharam em seus projetos por um período de 10 dias.

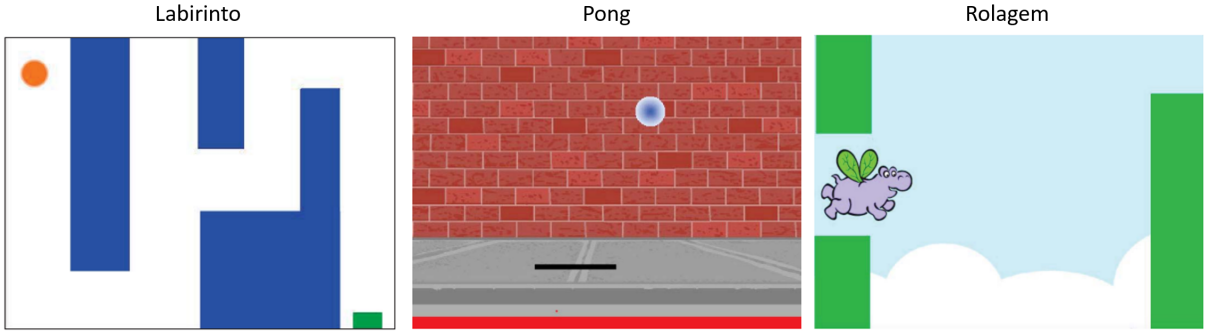
[1] https://meet.google.com/

Como a instituição estava em regime de ensino remoto, os alunos tiveram toda a liberdade para fazer os seus projetos de casa, em seu próprio ritmo. Em caso de dúvidas sobre o assunto e na condução do projeto, os alunos utilizavam a sala de aula virtual da disciplina, via *Google Classroom*, onde eram assistidos pelo docente e pelos próprios colegas.

Em particular, os alunos tiveram três opções de jogos para desenvolver (ver Figura 6):

- **Labirinto**: jogo no qual se deve conduzir um objeto pelo cenário até a saída do labirinto.
- **Pong**: um clássico jogo onde o jogador controla uma raquete para impedir que uma bola caia no chão.
- **Rolagem lateral**: nesse jogo, deve-se conduzir um objeto entre obstáculos, impedindo que ele entre em contato com o chão e com os obstáculos.

Figura 6 – Jogos a serem desenvolvidos pelos alunos.



Fonte: autoria própria.

Não era esperado que os alunos seguissem precisamente a descrição dos projetos. Acreditava-se que os alunos iriam querer construir seus projetos de forma diferente, e isso é visto como algo positivo, pois incentiva a criatividade. Assim, a liberdade criativa foi incentivada aos alunos. Por exemplo, tentar formas de aumentar o nível de dificuldade do jogo, criar níveis diferentes, usar um cronômetro e marcar pontuação, entre outras ideias.

3.3 QUESTIONÁRIO

Os alunos foram convidados a preencher voluntariamente um formulário de pesquisa (ver Apêndice A) após o término de seus projetos para não interromper os estudos caso tivessem que responder durante o projeto. O questionário foi criado usando a ferramenta *Google Formulários*[2], e apresentava um total de dez questões, entre questões fechadas e abertas; questões obrigatórias e não-obrigatórias. Em resumo, o questionário investiga os alunos em termos de:

- Motivação dos alunos acerca do estudo de programação dada à experiência que tiveram no curso;
- Percepção quanto às linguagens de programação apresentadas no que diz respeito à facilidade de compreensão e preferências em relação às linguagens;
- Interesse dos alunos em programação para o futuro.

[2] https://www.google.com/intl/pt-BR/forms/about/

- Recursos do Scratch;

Uma característica central da pesquisa é a percepção pelos alunos de que as habilidades de programação que aprenderam dentro do ambiente visual podem ser transferidas para uma linguagem textual. Essa percepção é importante, pois aprender a programar na linguagem de blocos, pedagogicamente, não é um fim e sim um meio. A linguagem visual não é tão flexível ou poderosa quanto uma linguagem convencional e não é adequada para, ou utilizada, no mercado de trabalho. Usar o Scratch pode simplificar o aprendizado de programação e é uma abordagem divertida, mas, o mais importante, para ter valor real, as habilidades aprendidas devem ser transferíveis para outras linguagens.

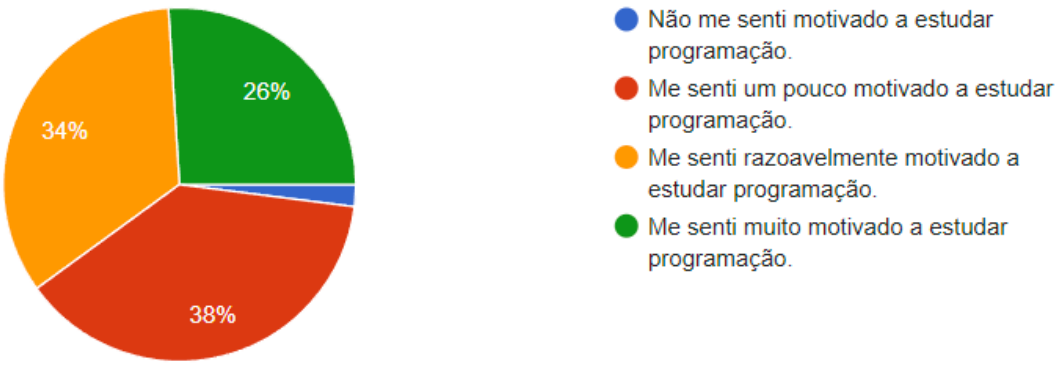
Por fim, o questionário é limitado em seu escopo, de modo que não foi possível investigar aspectos como "viés positivo"(YAACOUB *et al.*, 2004), onde os alunos podem responder de modo a mostrar suporte ao Scratch por acharem que este é o resultado desejado. O potencial de viés positivo foi reduzido pelo momento em que o questionário foi realizado, uma vez que os alunos foram convidados a participar após a conclusão de seus trabalhos e ficou claro que a participação era voluntária e não estava associada à nota daquele bimestre. A nota, por sua vez, foi dada com base na execução e resultado do projeto. Além disso, o questionário foi planejado para ser breve, de modo a facilitar participação e evitar a chamada "fadiga da pesquisa"(KAMPEN, 2007), embora seja provável que outros fatores influenciaram as respostas, como evitar a seleção de respostas que estão nas extremidades.

4 RESULTADOS E DISCUSSÃO

No total, 50 (cinquenta) estudantes concluíram o projeto de programação e participaram do questionário de pesquisa. Ao fazer o cruzamento com a participação dos alunos em bimestres anteriores da disciplina (por exemplo, quantos alunos tiveram notas no terceiro bimestre mas não no quarto), observa-se que todos os alunos com nota no terceiro bimestre do curso também conquistaram notas no quarto. Lembrando que programação (e, consequentemente, esse estudo) realizou-se no quarto bimestre da disciplina de Informática, e que a nota para esse bimestre foi um projeto de programação em Scratch (os jogos apresentados na seção anterior). Isso significa que **nenhum** aluno desistiu de cursar o quarto bimestre da disciplina, isto é, ninguém desistiu de cursar programação. Esse resultado é bastante positivo porque o ano letivo no qual se realizou o curso foi um ano bem conturbado por conta da pandemia de Covid-19, com várias desistências ao longo do ano, mas programação não foi um fator de desistência para os alunos. Além disso, a maioria dos alunos já conseguiram aprovação na disciplina no terceiro bimestre, conferindo ao quarto um bimestre um caráter opcional, sem o desespero típico de conseguir nota para aprovação como acontece em outras disciplinas. No que diz respeito ao desempenho dos alunos, apenas três alunos obtiveram um rendimento abaixo de 7.0, o que significa que não executaram o projeto de programação adequadamente. Quanto ao questionário, por ele ter tido uma natureza breve, com poucas e rápidas questões a serem respondidas, a participação em massa dos alunos não é algo que surpreende. Havia apenas três questões abertas, e elas não eram obrigatórias.

Na primeira questão do questionário, avalia-se o grau de motivação dos alunos para o estudo de programação após a experiência no curso. O curso teve uma duração muito breve, e com restrições importantes de infraestrutura por conta da pandemia. Como consequência, as aulas foram curtas e acompanhadas, em sua grande maioria, através de dispositivos móveis. Assim, o curso teve muito mais um caráter demonstrativo, de mostrar aos alunos uma nova perspectiva de uso dos computadores, do que tornar os alunos de fato programadores (o que seria um objetivo um tanto ingênuo ou otimista demais). Então, julgou-se importante investigar a motivação dos alunos para o estudo de programação. Na Figura 7 abaixo, considerando os resultados estritamente positivos (em laranja e verde na figura), observa-se que 60% dos alunos responderam que ficaram motivados para o estudo de programação, enquanto apenas 2% responderam que não se sentiram motivados.

Figura 7 – Como você considera seu grau de motivação para o estudo de programação após a experiência no curso?



Fonte: autoria própria.

Dado que a disciplina foi ministrada a alunos que não necessariamente possuem vocação para a área de tecnologia, com sérias limitações de recursos, num curso que não é sobre tecnologia, acredita-se que esses números são bastante positivos. E foi justamente essa não identificação com a área de tecnologia que pôde ser observado nos 2% que não se sentiram motivados pelo estudo de programação, conforme relatado numa questão aberta não-obrigatória do questionário que perguntou a justificativa pela motivação ou falta de motivação dos alunos:

> "Sempre é meio complicado fazer atividades de informática, inclusive essa, não sou muito apto para informática, além de não ter quase nenhum acesso a aparelhos eletrônicos."

E o que motivou os alunos? As respostas dos alunos apontam para o deslumbre que os alunos tiveram de que programar permite-lhes criar produtos com os o quais eles se identificam, como os jogos, como pode ser observado nos relatos a seguir:

> "A plataforma se demonstrou bastante interativa, desenvolver meu jogo e jogá-lo depois, foi imensamente prazeroso."

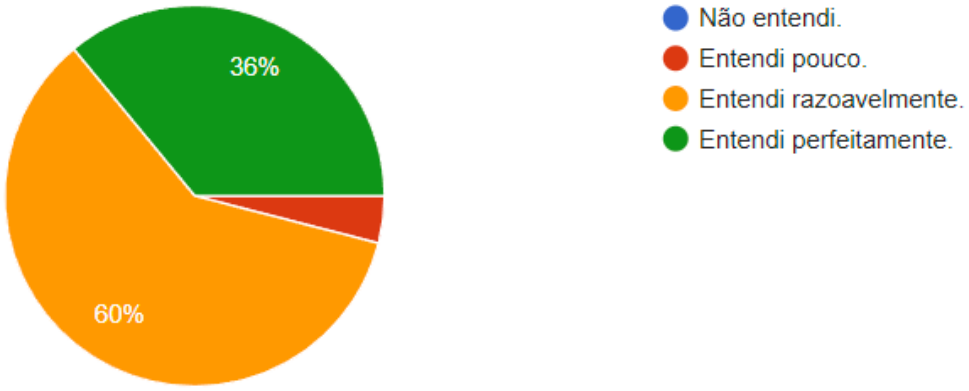
> "Eu estudaria [programação], achei bem legal uma criação de um jogo. A criação do meu jogo me deixou bastante animada, então eu estudaria sim!"

As duas questões seguintes avaliaram se os alunos foram capazes de compreender os conceitos de programação apresentados em aula quando demonstrados na linguagem visual Scratch, como também na linguagem textual Java. Os resultados nas Figuras 8 e 9 abaixo mostram que alunos afirmaram que foram capazes de entender os conceitos apresentados em ambas as linguagens: percentuais somados das cores verde e laranja nas duas figuras. Além disso, como era de se esperar, ao se comparar os percentuais das cores vermelhas nas figuras, observa-se que os alunos demonstraram maior dificuldade de compreensão quando os conceitos de programação foram apresentados na linguagem textual Java.

Um aspecto que chamou a atenção é o percentual de alunos que responderam que compreenderam razoavelmente os conceitos apresentados ser praticamente o mesmo (58% e 60% em laranja nos gráficos) entre as duas linguagens. Pode ter sido mera coincidência, mas também há outras causas possíveis. Por exemplo, alguns conceitos de programação são mais abstratos que outros. Enquanto há conceitos que podem ser expressos no Scratch com uma única instrução/bloco, conceitos mais abstratos, como o de variáveis, e o relacionamento entre esses conceitos, demandam várias instruções/blocos no Scratch para que faça algum sentido, assim também ocorre na linguagem textual, não resultando em diferença prática entre as duas abordagens. O ensino apropriado desses conceitos têm o potencial de poder melhorar sua compreensão. No caso desse estudo, o processo de ensino estava aquém do ideal. Apesar de muito convidativo, Scratch é um software sofisticado e leva tempo para aprender seus aspectos técnicos e pedagógicos.

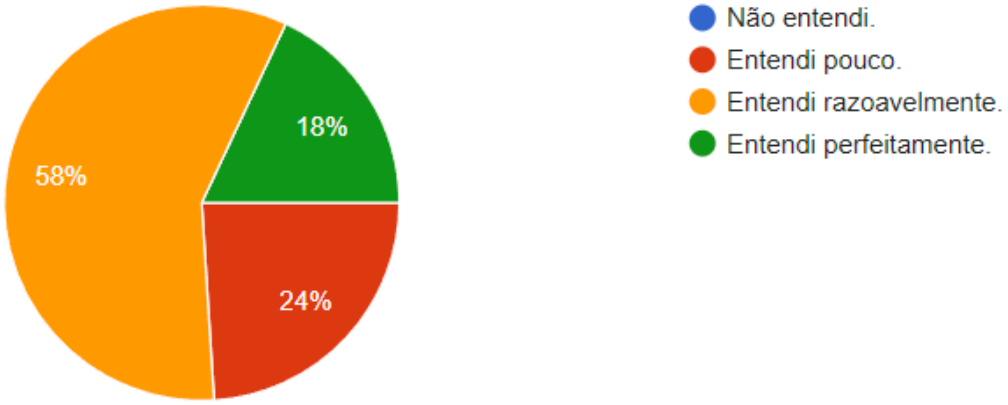
Na questão seguinte, avalia-se qual linguagem os alunos escolheriam para aprender programação. No gráfico da Figura 10 a seguir, observa-se que 64% dos alunos escolheriam apenas Scratch, 30% escolheria ambas as linguagens, e apenas 6% escolheria a linguagem textual. O resultado acima pode ser justificado pela necessidade de se ter que definir um determinado estilo de programação quando se adota uma linguagem textual, de forma a estruturar as soluções dos problemas já voltados para esse estilo de programação. Recomenda-se que o estudo de algoritmos deva ser capaz de distinguir o que é a natureza do problema e é o que é uma técnica de solução para o problema (CURRICULA, 2001). Isto é, o problema deve ser tratado independentemente da forma de implementar a solução para ele (HENDERSON, 1986; KOLIVER *et al.*, 2004). Por exemplo, as linguagens textuais apresentam grande número de detalhes

Figura 8 – Você foi capaz de entender os conceitos de programação apresentados (variáveis, estruturas de controle, estruturas de repetição, etc.) por meio do Scratch?



Fonte: autoria própria.

Figura 9 – Você foi capaz de entender os conceitos de programação apresentados (variáveis, estruturas de controle, estruturas de repetição, etc.) por meio da linguagem de programação textual (Java) apresentada em aula?

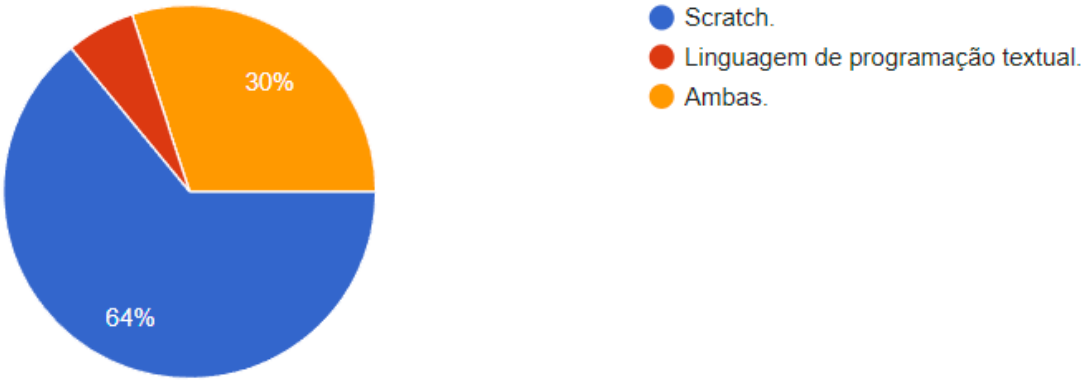


Fonte: autoria própria.

sintático-semânticos, que nem sempre permitem a abstração dos conceitos de programação no momento da codificação da solução para um problema, além de, em certa forma, prender o aluno ao estilo de programação inerente à aquela linguagem, o que não ocorre com a linguagem visual. Quando questionados sobre a justificativa de suas respostas, os comentários dos alunos apontam de fato para simplicidade do Scratch em relação à linguagem textual:

> "Pelo fato de que eu achei bem mais fácil usar o Scratch."
>
> "Acho que o Scratch é mais simplificado que a linguagem de programação textual."
>
> "Pois acho que a linguagem de programação do Scratch é mais didática e achei que facilita o uso e entendimento."

Figura 10 – Se você tivesse que escolher entre Scratch e uma linguagem de programação textual (como a mostrada nas aulas) para aprender conceitos de programação, qual das duas você escolheria?



Fonte: autoria própria.

Dado o caráter demonstrativo do curso como mencionado anteriormente, julgou-se pertinente questionar o interesse futuro dos alunos em programação. Isto é, será que os alunos fariam um curso completo e aprofundado de programação usando a linguagem visual Scratch e a linguagem textual Java. Os gráficos nas Figuras 11 e 12 mostram que, considerando as respostas estritamente positivas (em verde e roxo nos gráficos), 36% dos alunos aceitariam continuar seus estudos através da linguagem textual, e 44% através do Scratch. Considerando agora as respostas estritamente negativas (em azul e vermelho nos gráficos), 32% dos alunos dificilmente ou não fariam um curso completo na linguagem textual comparado a 20% em Scratch. Já as repostas neutras (em laranja) correspondem a 32% na linguagem textual e 36% no Scratch.

Os percentuais positivos e neutros próximos entre as duas linguagens talvez signifique que os alunos foram capazes de perceber a importância e despertar o interesse em programação, independentemente da tecnologia adotada, seja ela a visual ou textual. Isto é, a percepção de que a tecnologia adotada no ensino é apenas um meio para um fim maior (aprender a programar). Por exemplo, a natureza dos projetos desenvolvidos pelos alunos permitiu que eles progredissem até atingir um aspecto desafiador da atividade, e ai é onde eles naturalmente gastariam a maior parte do tempo. Mesmo que a linguagem baseada em blocos proteja os alunos da confusão sobre sintaxe de linguagem, como ocorre na linguagem textual, talvez eles simplesmente tenham passado para outros aspectos da programação igualmente desafiadores.

Alguns alunos também foram capazes de perceber a linguagem visual e textual como complementares:

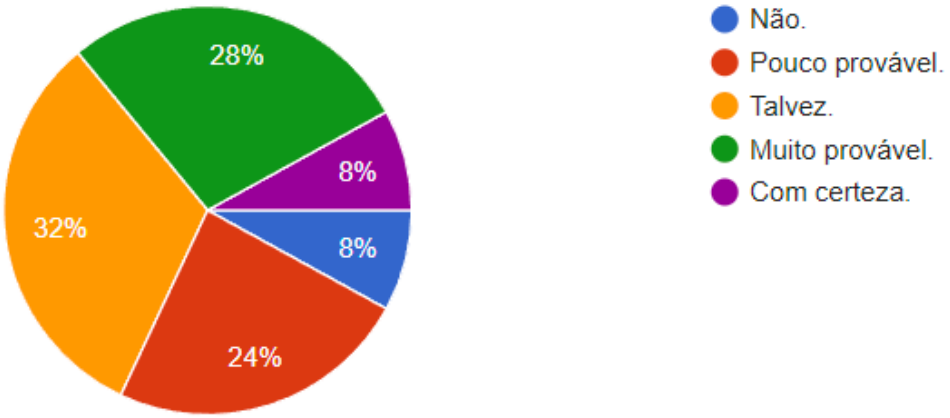
> "No Scratch aprenderia até chegar o ponto que eu tivesse a manha de programar e depois partiria para linguagens textuais."

> "A linguagem de programação textual é mais ampla, abrange muito mais conceitos e dá pra fazer muito mais coisas. Além de não se limitar a criar jogos/animações, como o Scratch."

De fato, Scratch pode servir como uma introdução a conceitos e práticas de programação básicos, seguidos pelas linguagens de programação baseadas em texto mais tradicionais em cursos de tecnologia. Por exemplo, o curso CS50[1] da Universidade de Harvard usa o Scratch como introdução à programação antes de apresentar a linguagem de programação textual.

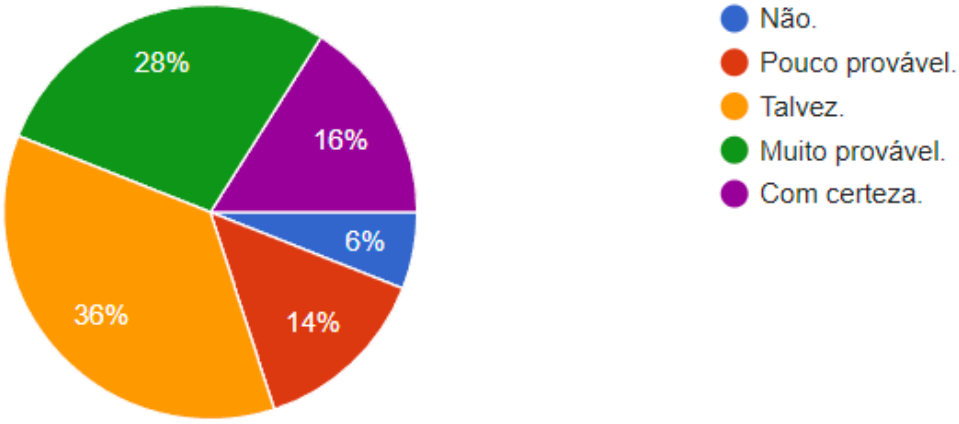
[1] https://cs50.harvard.edu

Figura 11 – Você faria um curso completo de programação utilizando a linguagem de programação baseada em texto (Java) que foi mostrada nas aulas?



Fonte: autoria própria.

Figura 12 – Você faria um curso completo de programação utilizando o ambiente Scratch?



Fonte: autoria própria.

Uma vez que o Scratch é proposto como uma alternativa pedagógica ao ensino de programação, foi pertinente questionar quais características do Scratch foram mais atrativas para os alunos. No gráfico da Figura 13 abaixo, observa-se que a possibilidade de criação de jogos, animações e histórias animadas facilitou a sua aceitação. Em seguida, os fatos de a programação se dar por meio de encaixes de blocos coloridos, e o ambiente possuir tradução para o português facilitou a utilização por parte dos alunos. Esse top 3 coincide com as expectativas geradas em torno do uso Scratch: o Scratch traz a programação para um contexto tipicamente de interesse dos alunos, por exemplo, criação de jogos; e facilita a programação ao evitar detalhes específicos de uma linguagem de programação textual, por exemplo, regras sintáticas e o fato de as linguagens de programação serem tradicionalmente no idioma inglês.

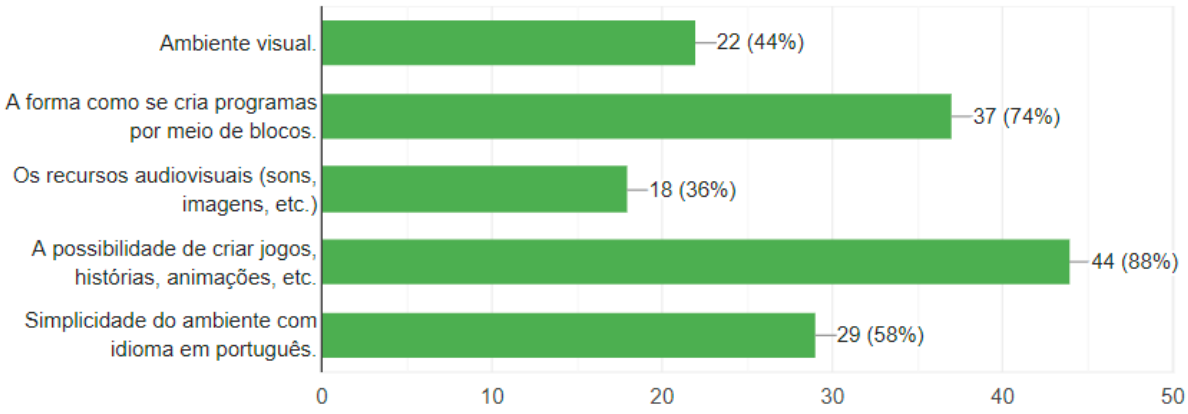
Finalmente, o questionário foi concluído pedindo que os alunos comentassem características das quais não gostaram, tiveram dificuldade, ou poderia ser melhor no Scratch. Os comentários apontam para três principais aspectos:

- Visual: o Scratch possui um aspecto visual muito infantil, o que pode desestimular e deixar os alunos com a faixa etária estudada (15-16 anos) céticos quanto a sua eficácia.
- Baixo desempenho com muitos recursos: colocar muitos recursos (imagens, sons, músicas)

deixa o programa pesado e dependendo da capacidade da máquina utilizada, os comandos ficam lentos e acabam travando.

- Utilizar muitos blocos de comando para realizar algo: no Scratch em alguns casos, dependendo do que se deseja fazer, necessita da junção de muitos blocos de comando, tornando a programação trabalhosa para os alunos.

Figura 13 – Quais características do Scratch mais atraiu você?



Fonte: autoria própria.

5 TRABALHOS RELACIONADOS

Scratch já foi avaliado em vários contextos. Por exemplo, Meerbaum-Salant *et al.* (2013) elaborou uma oficina de Scratch de duas horas de duração e observou sua eficácia em alunos de 15 anos em média. Uma análise das pontuações dos alunos em pré e pós testes de conceitos de programação mostrou uma melhoria significativa após o uso do Scratch, embora os alunos ainda apresentassem bastante dificuldades com conceitos mais abstratos de programação como inicialização, variáveis e concorrência. Maloney *et al.* (2008) descreveu sua experiência usando Scratch e sua análise dos programas criados pelos alunos. Eles perceberam que Scratch já era popular entre os alunos, com alunos usando-o voluntariamente e com mais frequência do que qualquer outra linguagem visual. Enquanto cerca de 20% dos programas criados pelos alunos incluíam apenas manipulação de mídia sem código, cerca de metade dos programas restantes empregaram conceitos como repetição e interação com o usuário do programa, com 25% usando conceitos de condicionais e instruções de sincronização. Eles também observaram que os alunos produziram programas cada mais complexos com o passar do tempo.

Alguns estudos comparam a programação em blocos e textual entre si. Por exemplo, Lewis (2010) comparou dois grupos de alunos de 12 anos participando de um curso de verão de programação, ao longo de 6 dias. Um grupo foi ensinado usando Scratch, e o outro recebeu aulas semelhantes usando Logo, uma linguagem textual. O curso foi elaborado para ensinar a fazer música, filmes e jogos usando computadores e, como consequência, as aulas eram ricas em conteúdo multimídia. Ao contrário da hipótese do autor de que a falta de erros de sintaxe do Scratch faria o aprendizado de programação mais fácil, os alunos acharam os exercícios igualmente difíceis em ambos os grupos. Alunos que ficaram com a linguagem textual também expressaram mais confiança em sua capacidade de programação após as atividades. Além disso, ambas as linguagens de programação pareciam mais adequadas para ensinar construções específicas, com os alunos da linguagem textual mostrando uma melhor compreensão de repetições, e alunos do Scratch mostrando uma melhor compreensão de condicionais. McKay e Kölling (2013) usaram modelagem preditiva para comparar uma variedade de ambientes de programação textual e em bloco, incluindo Scratch. Usando uma ferramenta de prototipagem, eles modelaram o tempo de execução de uma variedade de tarefas de programação em cada ambiente. Seus resultados sugerem que as linguagens textuais são mais adequadas para alguns tipos tarefas, enquanto as linguagens visuais são mais adequadas para outras.

Outros trabalhos investigaram a transição de linguagens visuais para as textuais. Wagner *et al.* (2013) descobriram que, repetindo exercícios primeiro usando linguagem visual e, em seguida, a textual, os alunos foram cada vez mais capazes de mapear mentalmente os conceitos de programação de uma linguagem para outra. Dann *et al.* (2012) comparou turmas treinadas puramente com a linguagem textual, e turmas treinadas com as duas linguagem, e descobriram que as turmas que passaram pelos dois tipos de linguagens foram superiores às treinadas exclusivamente com a linguagem textual. Esses estudos são importantes porque mostram que habilidades aprendidas em um ambiente visual como o Scratch podem ser transferidas para um ambiente textual.

6 CONSIDERAÇÕES FINAIS

Computadores permeiam a infraestrutura técnica da sociedade moderna: *smart* TVs e pagamento eletrônico são apenas alguns exemplos. Além disso, a Internet conecta toda a humanidade atualmente. Uma boa educação tecnológica é necessária para acompanhar o desafio de entender e também controlar essas tecnologias que mudam rapidamente, e tornar as pessoas capazes de lidar com o sua crescente complexidade. Nesse contexto, as instituições de ensino têm buscado cada vez mais, e cada vez mais cedo, promover uma compreensão de conceitos fundamentais de ciência da computação. Um desses aspectos mais fundamentais é que todos os computadores são controlados por programas, isto é, por algoritmos definidos rigorosamente e expressos em uma notação formal. Não há dúvidas de que aprender programação em uma idade jovem é útil para todos os alunos, pelo menos em sua vida cotidiana. No mundo digital de hoje, programação é uma habilidade fundamental junto com matemática e leitura. Aprender a programar ajuda os alunos a obter vantagens em raciocínio lógico, resolução de problemas e comunicação, entre outros. Como consequência, o tópico "Programação de Computadores"foi introduzido na mais recente ementa da disciplina de Informática, ministrada no primeiro ano do curso técnico integrado de Meio Ambiente do Instituto Federal de Alagoas, Campus Penedo, numa atualização do PPC do curso aprovada no final de 2019, que entrou em vigor no ano letivo de 2020.

No entanto, aprender programação é uma atividade complexa. Por exemplo, quando jovens são iniciados diretamente em linguagens de programação tradicionais, que são tipicamente textuais, eles ficam entediados e desanimados, porque um dos maiores obstáculos no aprendizado de uma linguagem de programação é aprender a sintaxe da linguagem. Uma alternativa é o uso de ambientes visuais de programação, dentre os quais, Scratch, que permitem abstrair esses detalhes sintáticos de outras linguagens, de modo que os alunos possam se concentrar no desenvolvimento dos algoritmos.

Nesse estudo, nós avaliamos o uso do Scratch para o ensino de programação às turmas ingressantes de Meio Ambiente. Os resultados mostraram que a maioria dos alunos afirmaram terem sido capazes de compreender os conceitos de programação, apoiando assim as evidências de que Scratch é uma plataforma viável para o ensino de programação. Outro resultado importante é que o ensino de programação não foi um fator de desistência da disciplina pelo alunos. Além disso, os alunos afirmaram-se motivados pelo estudo de programação, e dispostos a seguir adiante num curso mais aprofundado sobre o tema. Finalmente, algumas comparações foram feitas com uma linguagem de programação textual (Java), e como esperado, os alunos afirmaram maior dificuldades de compreensão quando se tratava da linguagem textual. No entanto, os alunos mostraram um atitude positiva em relação à linguagem textual, afirmando que também se interessariam em um estudo aprofundado de programação com a linguagem textual.

REFERÊNCIAS

BEGEL, Andrew. Logoblocks: A graphical programming language for interacting with the world. **Electrical Engineering and Computer Science Department, MIT, Boston, MA**, p. 62–64, 1996.

CAMPBELL, Patricia F; FEIN, Greta G; SCHOLNICK, Ellin K; SCHWARTZ, Shirley S; FRANK, Rita E. Initial mastery of the syntax and semantics of logo positioning commands. **Journal of educational computing research**, SAGE Publications Sage CA: Los Angeles, CA, v. 2, n. 3, p. 357–378, 1986.

CUNY, Jan; SNYDER, Larry; WING, Jeannette M. Demystifying computational thinking for non-computer scientists. **Unpublished manuscript in progress, referenced in http://www. cs. cmu. edu/˜ CompThink/resources/TheLinkWing. pdf**, 2010.

CURRICULA, CORPORATE The Joint Task Force on Computing. Computing curricula 2001. **Journal on Educational Resources in Computing (JERIC)**, ACM New York, NY, USA, v. 1, n. 3es, p. 1–es, 2001.

DANN, Wanda; COSGROVE, Dennis; SLATER, Don; CULYBA, Dave; COOPER, Steve. Mediated transfer: Alice 3 to java. *In*: **Proceedings of the 43rd ACM technical symposium on Computer Science Education**. *[S.l.: s.n.]*, 2012. p. 141–146.

DENNING, Peter J. The profession of it beyond computational thinking. **Communications of the ACM**, ACM New York, NY, USA, v. 52, n. 6, p. 28–30, 2009.

DENNING, Peter J; MCGETTRICK, Andrew. Recentering computer science. **Communications of the ACM**, ACM New York, NY, USA, v. 48, n. 11, p. 15–19, 2005.

FINCHER, Sally; PETRE, Marian. **Computer science education research**. *[S.l.]*: CRC Press, 2004.

GRAY, Simon; CLAIR, Caroline St.; JAMES, Richard; MEAD, Jerry. Suggestions for graduated exposure to programming concepts using fading worked examples. *In*: **Proceedings of the third international workshop on Computing education research**. *[S.l.: s.n.]*, 2007. p. 99–110.

HEINTZ, Fredrik; MANNILA, Linda; FäRNQVIST, Tommy. A review of models for introducing computational thinking, computer science and computing in k-12 education. *In*: **2016 IEEE Frontiers in Education Conference (FIE)**. *[S.l.]*: IEEE Press, 2016. p. 1–9.

HENDERSON, Peter B. Anatomy of an introductory computer science course. **ACM SIGCSE Bulletin**, ACM New York, NY, USA, v. 18, n. 1, p. 257–264, 1986.

JENKINS, Tony. On the difficulty of learning to program. *In*: CITESEER. **Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences**. *[S.l.]*, 2002. v. 4, n. 2002, p. 53–58.

KAMPEN, Jarl K. The impact of survey methodology and context on central tendency, nonresponse and associations of subjective indicators of government performance. **Quality & quantity**, Springer, v. 41, n. 6, p. 793–813, 2007.

KAZAKOFF, Elizabeth R; SULLIVAN, Amanda; BERS, Marina U. The effect of a classroom-based intensive robotics and programming workshop on sequencing ability in early childhood. **Early Childhood Education Journal**, Springer, v. 41, n. 4, p. 245–255, 2013.

KELLEHER, Caitlin; PAUSCH, Randy. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 37, n. 2, p. 83–137, 2005.

KOLIVER, Cristian; DORNELES, Ricardo Vargas; CASA, Marcos Eduardo. Das (muitas) dúvidas e (poucas) certezas do ensino de algoritmos. *In*: SN. **XII Workshop de Educação em Computação**. *[S.l.]*, 2004.

KOOHANG, Alex A. A study of attitudes toward computers: Anxiety, confidence, liking, and perception of usefulness. **Journal of research on computing in education**, Taylor & Francis, v. 22, n. 2, p. 137–150, 1989.

LEWIS, Colleen M. How programming environment shapes perception, learning and goals: logo vs. scratch. *In*: **Proceedings of the 41st ACM technical symposium on Computer science education**. *[S.l.: s.n.]*, 2010. p. 346–350.

MALAN, David J; LEITNER, Henry H. Scratch for budding computer scientists. **ACM Sigcse Bulletin**, ACM New York, NY, USA, v. 39, n. 1, p. 223–227, 2007.

MALONEY, John H; PEPPLER, Kylie; KAFAI, Yasmin; RESNICK, Mitchel; RUSK, Natalie. Programming by choice: urban youth learning programming with scratch. *In*: **Proceedings of the 39th SIGCSE technical symposium on Computer science education**. *[S.l.: s.n.]*, 2008. p. 367–371.

MAY, Jeffrey; DHILLON, Gurpreet. Interpreting beyond syntactics: A semiotic learning model for computer programming languages. **Journal of Information Systems Education**, v. 20, n. 4, p. 431, 2009.

MCCARTNEY, Robert; TENENBERG, Josh; HUBWIESER, Peter; ARMONI, Michal; GIANNAKOS, Michail N; MITTERMEIR, Roland T. Special issue on computing education in (k-12) schools. **Trans. Comput. Educ.**, 2014.

MCCARTNEY, R; TENENBERG, J; HUBWIESER, P; ARMONI, M; GIANNAKOS, M. Special issue ii on computer science education in k-12 schools. **Trans. Comput. Educ.**, v. 15, n. 2, 2015.

MCKAY, Fraser; KÖLLING, Michael. Predictive modelling for hci problems in novice program editors. **Proceedings of BCS HCI 2013-The Internet of Things XXVII**, British Computer Society, 2013.

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

MERRIENBOER, Jeroen JG Van; SWELLER, John. Cognitive load theory and complex learning: Recent developments and future directions. **Educational psychology review**, Springer, v. 17, n. 2, p. 147–177, 2005.

MOOR, Brian D; DEEK, Fadi P. On the design and development of a uml-based visual environment for novice programmers. **Journal of Information Technology Education: Research**, Informing Science Institute, v. 5, n. 1, p. 53–76, 2006.

NIKULA, Uolevi; SAJANIEMI, Jorma; TEDRE, Matti; WRAY, Stuart. Python and roles of variables in introductory programming: experiences from three educational institutions. **Journal of Information Technology Education: Research**, Informing Science Institute, v. 6, n. 1, p. 199–214, 2007.

PAPADAKIS, Stamatios; KALOGIANNAKIS, Michail; ZARANIS, Nicholas. Developing fundamental programming concepts and computational thinking with scratchjr in preschool education: a case study. **International Journal of Mobile Learning and Organisation**, Inderscience Publishers (IEL), v. 10, n. 3, p. 187–202, 2016.

PEÑALVO, Francisco José García; REIMANN, Daniela; TUUL, Maire; REES, Angela; JORMANAINEN, Ilkka *et al.* An overview of the most relevant literature on coding and computational thinking with emphasis on the relevant issues for teachers. 2016.

PENDERGAST, Mark O. Teaching introductory programming to is students: Java problems and pitfalls. **Journal of Information Technology Education: Research**, Informing Science Institute, v. 5, n. 1, p. 491–515, 2006.

RAUB, Annalyse Callahan. **Correlates of computer anxiety in college students**. *[S.l.]*: University of Pennsylvania, 1981.

RESNICK, Mitchel. New paradigms for computing, new paradigms for thinking. *In*: **Computers and exploratory learning**. *[S.l.]*: Springer, 1995. p. 31–43.

RESNICK, Mitchel; MALONEY, John; MONROY-HERNÁNDEZ, Andrés; RUSK, Natalie; EASTMOND, Evelyn; BRENNAN, Karen; MILLNER, Amon; ROSENBAUM, Eric; SILVER, Jay; SILVERMAN, Brian *et al.* Scratch: programming for all. **Communications of the ACM**, ACM New York, NY, USA, v. 52, n. 11, p. 60–67, 2009.

SCHÖN, Donald A; SANYAL, Bishwapriya; MITCHELL, William J. The computer clubhouse: Technological fluency in the inner city. MIT Press, 1998.

STACHEL, John; MARGHITU, Daniela; BRAHIM, Taha Ben; SIMS, Roderick; REYNOLDS, Larry; CZELUSNIAK, Vernon. Managing cognitive load in introductory programming courses: A cognitive aware scaffolding tool. **Journal of Integrated Design and Process Science**, IOS Press, v. 17, n. 1, p. 37–54, 2013.

WAGNER, Amber; GRAY, Jeff; CORLEY, Jonathan; WOLBER, David. Using app inventor in a k-12 summer camp. *In*: **Proceeding of the 44th ACM technical symposium on Computer science education**. *[S.l.: s.n.]*, 2013. p. 621–626.

WING, Jeannette M. Computational thinking. **Communications of the ACM**, ACM New York, NY, USA, v. 49, n. 3, p. 33–35, 2006.

YAACOUB, Elias E; GROVES, Robert M; DAWY, Zaher; JR, Floyd J Fowler; COUPER, Mick P; LEPKOWSKI, James M; SINGER, Eleanor; TOURANGEAU, Roger. **Survey Methodology**. *[S.l.]*: John Wiley & Sons, 2004. v. 337.

APÊNDICES

APÊNDICE A – QUESTIONÁRIO DE PESQUISA

Sua Experiência no Scratch!

Perguntas meramente com fins de pesquisa científica, em nada influenciará a nota da disciplina. Por favor, seja o mais sincero possível nas perguntas abaixo.

Faça login no Google para salvar o que você já preencheu. Saiba mais

*Obrigatório

Seu nome: *

Sua resposta

Seu e-mail: *

Sua resposta

Sua turma: *

- ◯ A
- ◯ B

Como você considera seu grau de motivação para o estudo de programação após a experiência com o Scratch? *

- ◯ Não me senti motivado a estudar programação.
- ◯ Me senti um pouco motivado a estudar programação.
- ◯ Me senti razoavelmente motivado a estudar programação.
- ◯ Me senti muito motivado a estudar programação.

Por que você se sentiu ou não se sentiu motivado a estudar programação?

Sua resposta

Você foi capaz de entender os conceitos de programação apresentados (variáveis, estruturas de controle, estruturas de repetição, etc.) por meio da linguagem de programação textual (Java) apresentada em aula? *

- ◯ Não entendi.
- ◯ Entendi pouco.
- ◯ Entendi razoavelmente.
- ◯ Entendi perfeitamente.

Você foi capaz de entender os conceitos de programação apresentados (variáveis, estruturas de controle, estruturas de repetição, etc.) por meio do Scratch? *

- ◯ Não entendi.
- ◯ Entendi pouco.
- ◯ Entendi razoavelmente.
- ◯ Entendi perfeitamente.

Se você tivesse que escolher entre Scratch e uma linguagem de programação textual (como a mostrada nas aulas) para aprender conceitos de programação, qual das duas você escolheria? *

- ◯ Scratch.
- ◯ Linguagem de programação textual.
- ◯ Ambas.

Justifique a escolha da pergunta anterior.

Sua resposta

Você faria um curso completo de programação utilizando o ambiente Scratch? *

- ◯ Não.
- ◯ Pouco provável.
- ◯ Talvez.
- ◯ Muito provável.
- ◯ Com certeza.

Você faria um curso completo de programação utilizando a linguagem de programação baseada em texto (Java) que foi mostrada nas aulas? *

- ◯ Não.
- ◯ Pouco provável.
- ◯ Talvez.
- ◯ Muito provável.
- ◯ Com certeza.

Sua Experiência no Scratch!

Quais características do Scratch mais atraiu você? (Pode marcar mais de uma opção.) *

- [ ] Ambiente visual.
- [ ] A forma como se cria programas por meio de blocos.
- [ ] Os recursos audiovisuais (sons, imagens, etc.)
- [ ] A possibilidade de criar jogos, histórias, animações, etc.
- [ ] Simplicidade do ambiente com idioma em português.

Existe(m) característica(s) das quais você não gostou, teve dificuldade, ou poderia ser melhor no Scratch? Descreva abaixo.

Sua resposta

Enviar Limpar formulário

APÊNDICE B – SEQUÊNCIA DIDÁTICA

B.1 JUSTIFICATIVA

O Pensamento Computacional pode ser utilizado em diversas áreas do conhecimento, uma vez que a tecnologia está fortemente inserida na sociedade atual e a cada dia surge uma nova tecnologia nas mais diversas áreas e contextos da sociedade. Contudo, é impossível ao ser humano dominar todas as tecnologias, mas este deve estar preparado para as constantes mudanças. Todavia, desenvolver tais habilidades não é uma tarefa imediata, mas sim um processo de construção de conhecimento que pode ser iniciado na Educação Básica.

B.2 OBJETIVO

Desenvolver o Pensamento Computacional no aluno por meio da programação de computadores utilizando a plataforma computacional Scratch. Espera-se que o aluno desenvolva competências e habilidades para resolver problemas do seu cotidiano, desenvolva habilidade de pensar de forma organizada ou estruturada antes de sair tentando resolver problemas sem antes pensar sobre, desenvolva sua criatividade, seu raciocino lógico matemático e o seu pensamento sistêmico.

B.3 CARACTERÍSTICAS

- **Proponente**: prof. Guilherme José de Carvalho Cavalcanti;
- **Público alvo**: alunos do primeiro ano do ensino médio;
- **Tempo de aplicação**: ensino remoto, com cinco encontros síncronos de duração de 60 minutos cada;
- **Recursos utilizados**: dispositivo computacional com acesso à internet, microfone e câmera;
- **Requisitos mínimos**: conhecimento de Informática Básica.

B.4 AULAS

Conteúdo:	1. Algoritmos e Programação.
Objetivo:	Desenvolver o conceito de algoritmos e programação. Dialogar com os alunos sobre como eles resolveriam os problemas propostos utilizando seu pré-conceito adquirido ao longo da vida. Aplicar a definição de algoritmos por meio da resolução de problemas e fazer com que o aluno aplique o referido conceito no seu cotidiano.
Avaliação de Aprendizagem:	Pedir aos alunos que escrevam algum algoritmo de alguma atividade de suas atividades cotidianas, um algoritmo de algo que eles saibam fazer. A sugestão é pedir para que eles imaginem que vão construir um robô que fará aquela atividade pra eles.

Conteúdo:	2. Sequências
Objetivo:	Demonstrar como um programa consiste em um conjunto de etapas ou instruções que são executadas uma de cada vez na ordem em que são escritas. Na programação, a sequência é um algoritmo básico: um conjunto de etapas lógicas realizadas em ordem. Os computadores precisam de instruções na forma de um algoritmo para concluir uma tarefa desejada, e esse algoritmo deve seguir uma ordem correta de etapas ou sequência. Fazer demonstrações de sequências usando o Scratch e também a linguagem de programação textual Java.
Avaliação de Aprendizagem:	Pedir aos alunos que façam seu cadastro na plataforma Scratch e siga qualquer um dos tutoriais sugeridos pela plataforma. Eles estarão naturalmente exercitando o conceito de sequências.

Conteúdo:	3. Variáveis
Objetivo:	Demonstrar como variáveis são declaradas, nomeadas e utilizadas para armazenar valores como números, texto etc. que são usados em um programa. Uma variável é uma forma de se referir a uma área de armazenamento em um programa de computador, e esse local da memória contém valores - números, texto ou outros tipos mais complexos de dados. Fazer demonstrações de variáveis usando o Scratch e também a linguagem de programação textual Java.
Avaliação de Aprendizagem:	Pedir aos alunos que façam uma calculadora simples (adição, subtração, multiplicação ou divisão) no Scratch, utilizando o conceito de variáveis para armazenar os valores que serão operados.

Conteúdo:	4. Estruturas de Controle
Objetivo:	Demonstrar como construir estruturas que permitem um ponto de escolha entre diferentes conjuntos de instruções subsequentes a serem executados em um programa. Frequentemente, um programa de computador deve fazer escolhas sobre como proceder, por exemplo, "se a senha digitada estiver correta, faça uma coisa, senão, faça algo diferente." As estruturas condicionais são os elementos mais básicos que permitem esse processo de tomada de decisão na programação. Fazer demonstrações de estruturas de controle usando o Scratch e também a linguagem de programação textual Java.
Avaliação de Aprendizagem:	Pedir aos alunos que façam um programa que verifica a situação de um aluno de numa disciplina, utilizando o conceito de estruturas de controle para testar o valor de uma nota e dizer a situação do aluno: aprovado ou reprovado.

Conteúdo:	5. Estruturas de Repetição
Objetivo:	Demonstrar como permitir que um conjunto de instruções seja executado várias vezes, repetindo-as quantas vezes forem necessários. As estruturas de repetição repetem uma parte do código fonte do programa um determinado número de vezes até que o processo desejado seja concluído. Tarefas repetitivas são comuns em programação, e as estruturas de repetição são essenciais para economizar tempo e minimizar erros. Fazer demonstrações de estruturas de repetição usando o Scratch e também a linguagem de programação textual Java.
Avaliação de Aprendizagem:	Pedir aos alunos que façam um programa que adivinha o que o usuário digitou no teclado, utilizando o conceito de estruturas de repetição para testar o valor do que foi digitado até que o valor digitado seja o esperado.