UFC

INSTITUTO UFC VIRTUAL

SISTEMAS E MÍDIAS DIGITAIS

ANA CLÁUDIA CHAVES FROTA

ARTE GENERATIVA PARA O ENSINO DE PROGRAMAÇÃO: UMA PROPOSTA DE ENGAJAMENTO

FORTALEZA

2022

ANA CLÁUDIA CHAVES FROTA

ARTE GENERATIVA PARA O ENSINO DE PROGRAMAÇÃO: UMA PROPOSTA DE ENGAJAMENTO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas e Mídias Digitais, do Instituto UFC Virtual da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de Bacharel em Sistemas e Mídias Digitais.
Orientador: Prof. Dr. Antônio José Melo Leite Júnior

FORTALEZA
2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

F961a Frota, Ana Cláudia Chaves.
Arte generativa para o ensino de programação : uma proposta de engajamento / Ana Cláudia Chaves Frota. – 2022.
85 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Instituto UFC Virtual, Curso de Sistemas e Mídias Digitais, Fortaleza, 2022.
Orientação: Prof. Dr. Antônio José Melo Leite Júnior.

1. Ensino de Programação. 2. Arte Generativa. 3. Engajamento. I. Título.

CDD 302.23

ANA CLÁUDIA CHAVES FROTA

ARTE GENERATIVA PARA O ENSINO DE PROGRAMAÇÃO: UMA PROPOSTA DE ENGAJAMENTO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas e Mídias Digitais do Instituto UFC Virtual da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de Bacharel em Sistemas e Mídias Digitais.

Aprovada em: 08/09/2022.

BANCA EXAMINADORA

Prof. Dr. Antônio José Melo Leite Júnior (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. George Allan Menezes Gomes
Universidade Federal do Ceará (UFC)

Prof. Dr. Rafael Augusto Ferreira do Carmo
Universidade Estadual do Ceará (UECE)

À minha mãe, que nunca mediu esforços para que eu tivesse uma educação de qualidade, quem eu muito amo.

AGRADECIMENTOS

À Universidade Federal do Ceará, por todo o conhecimento e vivências adquiridos.

Ao Prof. Dr. Antônio José Melo Leite Júnior, pela excelente orientação, disponibilidade e prontidão em sanar minhas dúvidas.

Aos professores participantes da banca examinadora George Allan Menezes Gomes e Rafael Augusto Ferreira do Carmo pelo tempo, pelas valiosas colaborações e sugestões.

Aos alunos que participaram das oficinas, pelo tempo e atenção concedidos.

Aos colegas de turma do curso, pelas reflexões, críticas e sugestões recebidas.

À minha família materna, que sempre esteve presente e me apoiou durante minha trajetória até aqui, em especial ao meu tio Paulo Wagner, que me abriu portas e me transmitiu valiosos ensinamentos pelos quais sou muito grata.

Acima de tudo, à minha mãe Ana Almerinda, sem a qual não teria conseguido.

“The artist who wants to do computer or algorithmic or interactive or net art and, therefore, wants to program the computer, must learn to think *how the machine would think if it could*.” (NAKE, 2010)

RESUMO

O conteúdo de programação nos cursos de graduação é geralmente muito denso. Isso acaba se tornando uma dificuldade para os alunos que estudam tecnologia, devido ao alto volume de informações a serem passadas e à dificuldade de abstração dos conceitos de lógica, o que muitas vezes causa um baixo engajamento. Nesse contexto, visando aumentar o engajamento dos estudantes com o ensino de programação, propõe-se um itinerário que consiste em conteúdos, abordagem e cronograma de um oficina de ensino de programação baseado em Arte Generativa – um tipo de arte que pode ser feita a partir de algoritmos utilizando da programação de forma contextualizada. A partir da aplicação de quatro oficinas e dos questionários aplicados em um total de 53 alunos do curso de Sistemas e Mídias Digitais da Universidade Federal do Ceará ao decorrer do semestre 2022.1, foi possível concluir que o ensino de programação com base em oficinas de Arte Generativa tem um impacto positivo na percepção de engajamento dos estudantes.

Palavras-chave: Ensino de Programação; Arte Generativa; Engajamento.

RESUMEN

El contenido de programación en los cursos de grado superior es generalmente muy denso. Esto termina convirtiéndose en una dificultad para los alumnos que estudian tecnología, debido al alto volumen de información presentada y a la dificultad de abstraer los conceptos de lógica, lo que muchas veces provoca un bajo engajamiento. En este contexto, con el objetivo de aumentar el engajamiento de los estudiantes con la enseñanza de la programación, se propone un itinerario que consta de contenidos, enfoque y cronograma de un taller de enseñanza de la programación basado en el Arte Generativo – un tipo de arte que se puede hacer a partir de algoritmos utilizando la programación en una manera contextualizada. A partir de la aplicación de cuatro talleres y de los cuestionarios aplicados a un total de 53 estudiantes del curso de Sistemas e Mídias Digitais de la Universidade Federal do Ceará durante el semestre 2022.1, fue posible concluir que la enseñanza de la programación basada en talleres de Arte Generativo tiene un impacto positivo en la percepción de compromiso de los estudiantes.

Palabras-clave: Enseñanza de programación; Arte Generativo; Engajamiento.

LISTA DE FIGURAS

Figura 1 – Condensation Cube……..…20
Figura 2 – 23-Ecke……..……...21
Figura 3 – Diagrama do itinerário-piloto……..31
Figura 4 – Execução do código proposto na oficina-piloto adaptado……...32
Figura 5 – Execução do código proposto na oficina-piloto…...……...33
Figura 6 – Turma durante uma das oficinas-piloto……...39
Figura 7 – Diagrama do itinerário…….41
Figura 8 – Exercício principal……...45
Figura 9 – Exercício bônus……...……46
Figura 10 – Primeiro passo de execução do código……..47
Figura 11 – Segundo passo de execução do código……..47
Figura 12 – Terceiro e quarto passos de execução do código……...48
Figura 13 – Experimento de um aluno……..49
Figura 14 – Experimento de um aluno com formas arredondadas.……..50
Figura 15 – Experimento de um aluno com círculos em tamanhos aleatórios…….50

LISTA DE GRÁFICOS

Gráfico 1 – Conhecimento de programação da turma 01……..36
Gráfico 2 – Conhecimento de programação da turma 02……..37
Gráfico 3 – UES da primeira turma da oficina-piloto…..…….37
Gráfico 4 – UES da segunda turma da oficina-piloto…..…….38
Gráfico 5 – Análise do formulário UES sobre a oficina da primeira turma…...…..…….52
Gráfico 6 – Análise do formulário UES sobre programação da primeira turma.…….52
Gráfico 7 – Análise do formulário UES sobre a oficina da segunda turma……..53
Gráfico 8 – Análise do formulário UES sobre programação da segunda turma.…….…..53
Gráfico 9 – Distribuição dos alunos por semestre da primeira turma…….…..54
Gráfico 10 – Conhecimento prévio em programação da primeira turma…….….55
Gráfico 11 – Distribuição dos alunos por semestre da segunda turma……...…….55
Gráfico 12 – Conhecimento prévio em programação da segunda turma…….….56

SUMÁRIO

1. INTRODUÇÃO **14**

2. REFERENCIAL TEÓRICO **16**

2.1. Ensino de programação 16

2.2. Engajamento 17

2.3. Arte Generativa 20

3. TRABALHOS RELACIONADOS **23**

3.1. Métodos alternativos de ensino de programação 23

3.1.1. Deconstruction/Reconstruction a pedagogic method for teaching programming to graphic designers 23

3.1.2. Um Estudo sobre a Aprendizagem de Lógica de Programação Utilizando Programação por Demonstração 24

3.2. Estudos abordando Arte Generativa 26

3.2.1. Ensino de arte generativa para estudantes de graduação 26

3.2.1.1. Teaching Generative Art to Undergraduate Students at a Technologically Advanced University: it's challenges and rewards. 26

3.2.1.2. Art for Computer Scientists: Processing as an Open-Source Art Medium for Computer Science Undergraduates 27

3.2.2. Re-coding 28

3.2.2.1. Think the Image, Don't Make It! On Algorithmic Thinking, Art Education, and ReCoding 28

3.3. Síntese 29

4. METODOLOGIA **30**

4.1. Base Metodológica 30

4.1.1. Pré-Experimento 31

4.1.2. Experimento 32

4.1.3. Pós-Experimento 34

4.1.3.1. Análise pessoal 35

4.1.3.2. Análise qualitativa 35

4.1.3.3. Análise quantitativa 36

4.1.3.4. Triangulação das análises 38

4.2. Proposição metodológica 40

4.2.1. Pré-Experimento 42

4.2.2. Experimento 42

4.2.3. Pós-Experimento 43

5. EXEMPLO DE USO DO ITINERÁRIO **45**

5.1. Experimento 45

5.2. Resultados 49

5.2.1. Análise pessoal 49

5.2.2. Análise qualitativa 51

5.2.3. Análise quantitativa 51

5.2.3.1. Primeira Turma 54

5.2.3.2. Segunda turma 55

5.2.4. Triangulação 56

6. CONSIDERAÇÕES FINAIS **59**

REFERÊNCIAS **62**

APÊNDICE A – Listagem dos códigos da oficina-piloto **65**

APÊNDICE B – Códigos de inspiração da oficina-piloto **69**

APÊNDICE C – Questionário completo aplicado **72**

APÊNDICE D – Código proposto nas oficinas finais **74**

1. INTRODUÇÃO

Nos cursos relacionados a Computação, em particular na formação de profissionais híbridos, há uma dificuldade em manter o engajamento do aluno no ensino de lógica de programação (PITEIRA; HADDAD, 2011). A carga de conceitos abstratos dos primeiros anos dos cursos da área é significativa e pode ser decisiva para a motivação dos estudantes (SANTOS *et al.*, 2008).

De acordo com Haden e Mann (2003, p. 64), os educadores devem reconhecer que os estudantes atuais têm uma relação diferente com computadores do que antigamente pois já estão habituados a utilizar interfaces modernas, sendo isso citado como uma problemática devido à rápida crescente da tecnologia já em 2003, segue ainda mais acentuado esse disparate. Como esses estudantes têm altas expectativas para o ensino de programação, os educadores devem buscar propor que os estudantes façam programas que os empolgue e os engajem (HADEN; MANN, 2003).

O conceito de engajamento é muito amplo, e Zyngier (2008) define o engajamento estudantil como uma mistura de vários componentes:

1. Comportamental: persistência e participação;
2. Emocional: interesse, valor e valência;
3. Cognitivo: motivação, esforço e estratégia.

Essa construção multifacetada do engajamento é ideal e deve ser aplicada em um nível institucional para que os estudantes realizem atividades que os mobilizem intelectualmente, sendo essa uma responsabilidade das instituições de ensino superior (CÔRTE VITÓRIA *et al.*, 2018).

Atualmente, diferentes abordagens no ensino de programação se fazem importantes devido tanto à alta demanda do mercado por profissionais que saibam programar bem quanto aos benefícios pessoais que o pensamento computacional[1] oferece. Nos Estados Unidos, oportunidades de trabalho na área de Computação e Tecnologia da Informação estão estimadas para subirem em 13% nos próximos 10 anos, comparado com um crescimento geral de 7% em todos os empregos (BUREAU OF LABOR STATISTICS, 2020).

O pensamento computacional é importante não só para a colocação profissional do indivíduo, mas para o raciocínio em geral, sendo, no contexto nacional, citado nove vezes no

[1] Um dos fundamentos da Computação, consiste em estruturar problemas e elaborar soluções práticas.

documento da Base Nacional Curricular, dentre elas defendendo que a resolução de problemas matemáticos é fundamental para o desenvolvimento do pensamento computacional (XAVIER *et al*., 2021).

O pensamento computacional serve não só para a programação em si, mas para qualquer resolução de problemas, podendo ser aplicado no cotidiano de qualquer pessoa. “Pensamento computacional é uma habilidade fundamental para todos, não só para cientistas da computação. Junto com leitura, escrita e aritmética, deveríamos incluir o pensamento computacional nas capacidades analíticas de todas as crianças” (WING, 2006).

Nesse contexto, faz-se importante o uso de técnicas para aumentar o engajamento estudantil no ensino de programação, podendo ser uma delas o estudo da Arte Generativa. Segundo o professor e artista Philip Galanter (2003), Arte Generativa se refere a qualquer prática artística na qual o artista cede o controle a um sistema com autonomia funcional que contribui para, ou resulta em, uma obra de arte completa. A Arte Generativa no meio digital pode ser entendida como um algoritmo criado proporcionando a geração de arte pelo sistema computacional de forma autônoma dentro dos parâmetros pré-definidos.

Portanto, em que aspectos a Arte Generativa pode engajar os estudantes no processo de aprendizagem de programação? O objetivo geral deste projeto é propor um itinerário com conteúdos, abordagem e cronograma de práticas de ensino de programação com base em Arte Generativa. A fim de obter uma resposta mais eficaz para esse objetivo geral, delinearam-se os seguintes objetivos específicos: avaliar experiências anteriores de programação criativa, analisar o cruzamento do ensino de programação com Arte Generativa baseado em testes e verificar os impactos do experimento a partir da análise feita com o protocolo UES (*User Engagement Scale*).

2. REFERENCIAL TEÓRICO

Na presente seção são expostos conceitos explicativos acerca dos principais tópicos que este trabalho aborda, sendo esses: ensino de programação, engajamento e Arte Generativa, de modo a formar um referencial teórico.

2.1. Ensino de programação

A lógica de programação pode ser definida como a técnica de encadear ideias para atingir determinado objetivo ou solucionar determinado problema, sendo a sequência desses comandos conhecida como algoritmo (GARCIA; LOPES, 2002). Sendo classificada como uma das bases do ensino nos cursos da área de Computação e Informática (BRASIL, 2001), o ensino da lógica de programação é muito necessário para solucionar problemas tanto acadêmicos quanto para qualificar os profissionais para o mercado de trabalho na área de Tecnologia da Informação (TI).

De acordo com um estudo da Associação das Empresas de Tecnologia da Informação e Comunicação e de Tecnologias Digitais (BRASSCOM, 2019), a evasão nos cursos da área de TI foi de 69% em todo o Brasil em 2017. Ainda, em um estudo específico, feito tendo como amostra os alunos da Universidade Tecnológica Federal do Paraná, foi constatado que 80% da evasão escolar em cursos de graduação presenciais ocorreram até o 3° período dos cursos (DE OLIVEIRA JÚNIOR *et al.*, 2017). Isso ocorre principalmente pela dificuldade de adaptação dos estudantes já nas disciplinas introdutórias, como é o caso das disciplinas de lógica de programação nos cursos de tecnologia.

Vários problemas no ensino são causadores dessa evasão, dentre eles está a falta de motivação decorrente do ensino tradicional de programação, que não permite ao aluno entender a importância prática dos conteúdos abordados (BORGES, 2000), e a falta do tempo necessário para essa adaptação aos conceitos mais abstratos e maturação do conhecimento (BEREITER; NG, 1991).

De certa forma, a programação, e os computadores em geral, são um advento muito recente na sociedade. Já o ensino de Matemática, por exemplo, perdura desde a idade média e pouco se alterou na forma de aprendizagem geral, que consiste em propor problemas, usando de analogias com objetos do cotidiano e estimular os alunos que resolvam (DIJKSTRA, 1988). Porém, a computação, como algo mais recente e com tantos conceitos abstratos, de acordo com Dijkstra (1988) requer que haja um ensino lento e gradual. E, junto a isso, é

também muito difícil administrar os diversos ritmos de aprendizagens presentes em uma turma (JÚNIOR *et al.*, 2005).

Para Montfort *et al.* (2014), assim como em um labirinto, que se desvenda ao percorrer com uma linha – como no mito do Minotauro – , ao de fato percorrer linhas de código é que se encontra o caminho da programação. Somente rodando o código, testando, vendo possibilidades do que pode acontecer ao alterar o comando, se adquire a experiência para desvendar a lógica por trás dele.

Os métodos tradicionais do ensino dessa disciplina, que contém conceitos dinâmicos, geralmente envolvem materiais estáticos, como diagramas, explicações verbais e apresentações projetadas (GOMES *et al.*, 2008). É ingenuidade esperar que os calouros jovens, que são de uma geração que já possui familiaridade com a tecnologia, tenham interesse em solucionar problemas no papel sem ter sequer interação com o computador (RAPKIEWICZ *et al.*, 2006). Nesse âmbito, torna-se fundamental a contextualização no ensino de lógica de programação de modo a engajar os alunos e proporcionar um primeiro contato com o pensamento computacional.

Para melhorar o desempenho dos estudantes em determinado assunto, uma alternativa é oferecer aos alunos micromundos verdadeiramente interessantes onde eles possam utilizar o conteúdo exposto (PAPERT, 2008). Em estudo, Gomes *et al.* (2008) chegaram à conclusão de que para potencializar o ensino de programação é necessário um ambiente lúdico e estimulante, bem como respeito ao estilo individual de aprendizado, entre outros.

Com base nisso, um exemplo prático muito impactante é o do Harvey Mudd College, uma universidade de elite estadunidense, que de 2006 para 2013 conseguiu aumentar o percentual de mulheres no curso de Ciência da Computação de 10% para 48%. Tal resultado foi obtido focando em atividades que fossem práticas e aplicáveis, dando aos estudantes escolhas de quais projetos eles gostariam de fazer e tornando o curso mais divertido em geral (TAYLOR, 2013).

2.2. Engajamento

De acordo com Papert (1983), um dos pontos centrais do ensino é o engajamento do indivíduo, pois estando o aprendizado muito mais relacionado ao amor do que à lógica, a educação tem pouco em comum com a explicação, mas muito com o ato de se engajar e se apaixonar pelo material. Engajamento é um conceito muito amplo e flexível, tendo vertentes

acadêmicas, cognitivas, intelectuais, institucionais, emocionais, comportamentais, sociais e psicológicas (TING *et al.*, 2013).

Tendo múltiplas definições disponíveis, pode-se tomar como base a definição de engajamento proposta por Jaimes *et al*. (2011). Ainda que ela esteja contextualizada no âmbito das redes sociais, esta levanta uma proposta mais duradoura de engajamento, como a relação que o estudante deve criar com o uso de programação durante a graduação e sua vida profissional:

> Engajamento define o fenômeno de ser cativado e motivado: engajamento pode ser medido em termos de uma única sessão interativa ou de uma relação mais duradoura com a plataforma social durante múltiplas interações. Portanto, engajamento com mídias sociais não se trata somente sobre como uma única interação se dá, mas sobre como e porque as pessoas desenvolvem uma relação com a plataforma ou serviço e *integram ela em suas vidas*. (JAIMES *et al.,* 2011, grifo nosso, tradução nossa).

O engajamento, como um conceito tão abstrato, é algo cuja tentativa de mensuração já foi objeto de estudo de diversos trabalhos, incluindo análises comportamentais – como quantidades de visitas a páginas da web –, rastreamento neurofisiológico – analisando movimento dos olhos e atividade eletrodermal – e auto-relatos incluindo questionários, entrevistas e arguição verbal (LALMAS *et al.*, 2014; O'BRIEN; CAIRNS, 2016, apud O'BRIEN; CAIRNS; HALL, 2018). Em particular, a partir desses estudos, O'Brien (2008, apud O'BRIEN; CAIRNS; HALL, 2018) desenvolveu o *User Engagement Scale* (UES), visando validar os atributos mais relevantes para o engajamento.

Portanto, o UES é um questionário com 31 questões objetivas de acordo com a escala de Likert – modelo desenvolvido por Rensis Likert em 1932 com intuito de classificar questões comportamentais, no qual é exposta uma afirmação e o sujeito responde à questão em uma escala de 0 a 5, onde 0 significa "discordo totalmente" e 5 "concordo totalmente" –, que é utilizado amplamente desde sua publicação (O'BRIEN; TOMS, 2010) e aborda seis conceitos principais:

- Atenção Focada (FA, do inglês *Focused Attention*): concentrar em um estímulo somente, ignorando todos os outros (MATLIN, 1994, apud O'BRIEN; TOMS, 2010);
- Usabilidade Percebida (PU, do inglês *Perceived Usability*): as emoções vivenciadas pelo usuário e a percepção do nível de controle dele em relação à interação (O'BRIEN; TOMS, 2010). No questionário é considerado o aspecto negativo decorrido do esforço necessário e do controle do usuário (O'BRIEN; CAIRNS; HALL, 2018);

- Apelo Estético (AE, do inglês *Aesthetics*): a beleza visual ou o estudo de ambientes computacionais naturais e agradáveis (e estéticos) (JENNINGS, 2000, apud O'BRIEN; TOMS, 2010);
- Durabilidade (EN, do inglês *Endurability*): Designa a probabilidade de relembrar coisas que foram prazerosas e realizar novamente uma atividade que foi divertida (READ; MACFARLANE; CASEY, 2002, apud O'BRIEN; TOMS, 2010), implica na disposição dos usuários a recomendar a aplicação.
- Novidade (NO, do inglês *Novelty*): Variedade de mudanças súbitas e inesperadas que provocam animação, júbilo ou alarme (ABOULAFIA; BANNON, 2004, apud O'BRIEN; TOMS, 2010)
- Envolvimento Sentido (FE, do inglês *Felt Involviment*): sentir-se atraído e envolvido, em geral a percepção da experiência como "divertida" (O'BRIEN; TOMS, 2010) .

Visando criar uma alternativa válida de questionário reduzido do UES, ao perceber que diversos pesquisadores estavam utilizando o formulário de forma fragmentada, O'Brien, Cairns e Hall (2018) desenvolveram e validaram o UES reduzido. Nesse questionário constam somente 12 questões e pode ser ideal para estudos nos quais os participantes precisam realizar muitas atividades ou comparar aplicações, visto que pode ser uma tarefa fatigante, de acordo com O'Brien, Cairns e Hall (2018). As questões presentes no questionário reduzido são as seguintes (O'BRIEN; CAIRNS; HALL, 2018):

FA-S.1 Me senti perdido na Aplicação X.
FA-S.2 O tempo durante o uso da Aplicação X passou voando.
FA-S.3 Me senti imerso nessa experiência.
PU-S.1 Me senti frustrado usando a Aplicação X.
PU-S.2 Achei a Aplicação X confusa.
PU-S.3 Usar a Aplicação X foi estressante.
AE-S.1 A Aplicação X foi atrativa.
AE-S.2 A Aplicação X foi esteticamente agradável.
AE-S.3 A Aplicação X me agradou.
RW-S.1 Usar a Aplicação X valeu a pena.
RW-S.2 Minha experiência foi gratificante.
RW-S.3 Me senti interessado nessa experiência

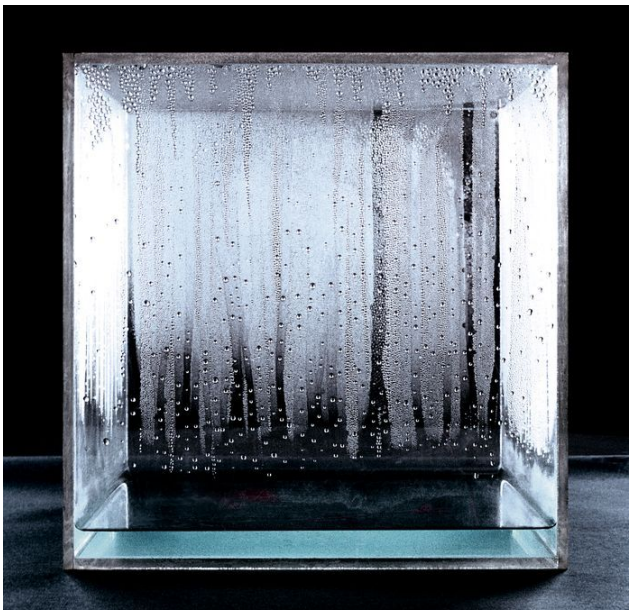
Mesmo com número de questões limitadas, o UES reduzido consta com itens abordando tanto a aplicação em si quanto a experiência do usuário, sendo um formulário completo.

2.3. Arte Generativa

Um recurso que pode ser utilizado como aliado ao ensino alternativo de programação é a Arte Generativa. Hertlein (1977), já em 1970, ensinava cursos de Arte Computacional para estudantes de Ciência da Computação, afirmando que em um só período os estudantes habituados com código já conseguiam produzir peças em nível profissional.

Arte Generativa muitas vezes é entendida como toda e qualquer Arte Computacional, mas as duas nem sempre coincidem. Arte Computacional é aquela na qual o artista instrui um agente, o computador, para que este possa sintetizar a manifestação artística (BERRUEZO, 2019). Já a Arte Generativa se caracteriza em ser uma prática na qual o artista cede parcialmente o controle a um sistema autônomo, que gera a arte completa (GALANTER, 2008). A diferença é que o sistema autônomo não necessariamente é um computador, como é o caso da peça *Condensation Cube* de Hans Haacke (1963-1965), que consiste em um cubo transparente de acrílico de 30cm³ com cerca de 1cm³ de água dentro (Figura 1).

Figura 1 - Condensation Cube



Fonte: Museum of Contemporary Art of Barcelona.[2]

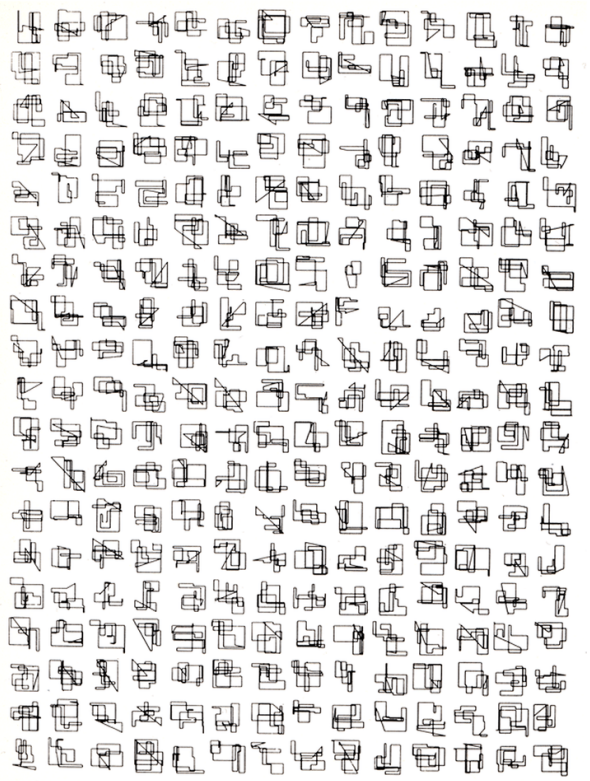
[2] Disponível em https://www.macba.cat/en/art-artists/artists/haacke-hans/condensation-cube. Acesso em 21/08/2022.

Na obra de Haacke (1963-1965), água condensava e formava padrões verticais na superfície do cubo. Assim, nesta peça, o sistema autônomo é a própria física da água, sendo portanto um exemplo de Arte Generativa que não se encaixa na definição de Arte Computacional.

> O processo de condensação não acaba. A caixa tem uma constante porém vagarosamente mutante aparência que nunca se repete. As condições são comparáveis às de um organismo vivo que reage de forma flexível aos seus arredores. (HAACKE, 2003).

No presente trabalho, foi utilizada como base a Arte Generativa coincidente com a Arte Computacional. Georg Nees foi o grande pioneiro na Arte Generativa. Sendo um matemático muito interessado nas possibilidades da aleatoriedade, ele desenvolveu algumas obras experimentais que chamaram a atenção de Max Bense, diretor da Universidade de Stuttgart, na Alemanha (NAKE; NEES, 2018).

Figura 2 - 23-Ecke



Fonte: Arte por Georg Nees, disponível em website[3]

[3] Disponível em http://dada.compart-bremen.de/item/artwork/639. Acesso em 21/08/2022.

Em 1965, Bense chama Nees para expor sua arte, e foi então que surgiu oficialmente a Arte Generativa, em sua primeira exibição na Universidade de Stuttgart, com alguns desenhos de formas aleatórias criados por Georg Ness (NAKE; NEES, 2018), como disponível na Figura 2. Quatro anos após essa exibição, Georg Nees concluiu seu PhD em Arte Computacional e o nomeou com o mesmo nome da exibição, *Generative Computergraphik*.

Desde então, a Arte Generativa vem se popularizando. Em particular, o Processing[4], uma linguagem criada por Casey Reas e Ben Fry, tem se tornado uma conhecida ferramenta para criação. O Processing tem como base a linguagem Java e, segundo seus criadores, foi feito para impulsionar o ensino dos fundamentos da programação de computadores em um contexto visual orientado à geração e processamento de imagens, podendo ser utilizado para ensino, prototipação e desenvolvimento final (REAS; FRY, 2007).

Na Arte Generativa, a imagem individual é reduzida a uma instância de uma classe de imagens categorizadas por algoritmos de programação (NAKE; GRABOWSKI, 2017). Para refazer uma arte com código é necessário pensar no processo de construção e em todos os passos e elementos necessários para reconstruir aquilo, sendo esse um exercício aplicado do pensamento computacional (NAKE; GRABOWSKI, 2017).

Assim, a Arte Generativa pode ser uma ferramenta de engajamento para estudantes ao inserir o contexto visual e artístico aplicado a problemas fundamentais de lógica. Atualmente, com ferramentas como o Processing, já existe uma ponte estabelecida entre a lógica de programação e a Arte Generativa.

[4] Disponível em processing.org

3. TRABALHOS RELACIONADOS

Para uma melhor compreensão de possibilidades, foi feito um levantamento de trabalhos relacionados que abordam temas de ensino de programação e de Arte Generativa. Desse modo, os trabalhos foram categorizados em três grupos de acordo com os temas centrais: métodos alternativos de ensino de programação em geral e específicos sobre o emprego de Arte Generativa. Tais trabalhos são expostos e discutidos a seguir.

3.1. Métodos alternativos de ensino de programação

Os trabalhos seguintes abordam o tema central da introdução ao ensino de programação.

3.1.1. Deconstruction/Reconstruction a pedagogic method for teaching programming to graphic designers

Esse estudo propõe um método pedagógico construtivista – segundo Fossile (2010), para o construtivismo pedagógico o aprendizado é um processo pessoal em que o conhecimento é resultado do que é construído pelo estudante – de desconstrução/reconstrução, tendo como público alvo os estudantes de Design Gráfico da faculdade *The Danish School of Media and Journalism* que possuíam pouco ou nenhum conhecimento prévio em programação. Para tanto, propôs-se que os participantes fizessem uso de informações prévias de como construir peças visuais como base para aprender como programar (HANSEN, 2017).

O autor aponta algumas observações feitas durante nove anos de ensino de introdução a programação para esse perfil específico de aluno, dentre elas estão:

- os estudantes precisam de uma meta fixa para que seja mais fácil que mantenham o foco;
- a matemática e a lógica devem ser ensinadas somente quando se tornar necessário;
- os estudantes precisam explorar código e funcionalidades por si sós;
- os estudantes respondem negativamente a aulas de aprendizado passivo e explicações abstratas, portanto precisando de feedback visual prático.

A metodologia proposta no trabalho foi dividida em seis passos, que podem ser agrupados em duas grandes fases de desconstrução e reconstrução.

Na desconstrução estão contidos os três primeiros passos, que utilizam como base o conhecimento prévio do aluno, sendo esses:

- seleção, que consiste em escolher uma peça de design para ser o foco da prática;
- descrição, o ato de fazer anotações sobre os componentes visuais da obra selecionada;
- análise, a reflexão sobre a lógica que conecta os componentes visuais descritos no passo anterior.

Na reconstrução, fase na qual é introduzido o código como ferramenta, estão contidos os últimos passos:

- conversão, quando as anotações da desconstrução são traduzidas em código, replicando a imagem escolhida;
- exploração, quando, a partir da base construída, são geradas variações possíveis;
- experimentação, quando, com um entendimento mais sólido, o estudante faz alterações mais radicais no código, gerando variações mais originais.

O método foi testado em duas turmas de dois cursos de introdução à programação e teve resposta positiva dos alunos em entrevistas posteriores. Ao utilizar o método tendo como base conhecimentos prévios, a ansiedade e aversão dos alunos à programação foi mitigada. Um trabalho futuro apontado pelo autor foi o de organizar a fase de desconstrução em equipes de estudantes, estimulando a colaboração entre os alunos.

3.1.2. Um Estudo sobre a Aprendizagem de Lógica de Programação Utilizando Programação por Demonstração

Tendo em vista um melhor ensino de programação, Ferreira *et al.* (2010) conduziram um estudo com alunos de ensino médio com idades entre 14 e 17 anos, que participaram de um curso de programação com base na PDB (Programação por Demonstração, do inglês Programming by Demonstration) – uma tática de ensino que visa o aprendizado de programação com interfaces gráficas, sem o aprendizado de uma linguagem de programação em si – e, posteriormente, empregando Portugol.

A ferramenta de PDB utilizada foi o StageCast Creator[5], um software educacional no qual se pode desenvolver aplicativos simples sem a necessidade de implementar código, utilizando somente a interface gráfica. Para avaliar o desempenho dos alunos, que foram divididos em grupo controle e grupo experimental, com turmas equivalentes em idade e tempo de uso de computador/jogos de video-game por semana, foi elaborado um teste com a abordagem GQM (*Goal-Question-Metric*) – abordagem que visa escolher uma meta e, a partir desta, definir os dados que devem ser monitorados de modo a metrificar a evolução dos dados em relação à meta (BASILI *et al*., 1994 apud FERREIRA *et al.*, 2010).

Primeiramente foi elaborado um estudo piloto, no qual somente o grupo experimental teve acesso a um curso de PBD de carga horária total de 6 horas divididas em três dias. Posteriormente, foi realizado um curso semanal de Portugol, juntamente com o grupo controle, no qual não foi utilizado o computador. No final, foi aplicada uma avaliação escrita, que foi inconclusiva, pois, de acordo com os autores, é muito subjetivo analisar o código sem que os alunos utilizem um ambiente de programação. No entanto, houve 68% de evasão do curso, tendo os grupos iniciado com 14 participantes, cada; e ao final somente 2 no grupo controle e 7 no grupo experimental compareceram à avaliação final.

No experimento oficial, devido à dificuldade de avaliar o desempenho no piloto, os autores optaram por realizar o curso de Portugol com o uso do compilador G-Portugol. A evasão seguiu alta, com 132 inscrições e permanecendo somente 60 já no início, sendo 40 do grupo experimental e 20 do grupo controle. Durante o curso 18 alunos desistiram, concluindo o curso 42 participantes no total; e destes, 32 eram do grupo experimental, reforçando a influência do PDB na redução da evasão.

A avaliação foi feita com um teste com quatro tarefas a serem implementadas com Portugol e compiladas com G-Portugol. Houve comparecimento de 6 participantes do grupo controle e 18 do grupo experimental. Devido à evasão, os grupos não estavam equivalentes em idade nem em uso do video-game ou computador, portanto foram feitas análises matemáticas probabilísticas de modo a comparar de forma equilibrada os grupos controle e experimental.

Como haviam 6 participantes do grupo controle somente, e 18 no experimental, foram feitas análises com todas as combinações possíveis de 6 alunos dentre os 18 do grupo experimental. A partir dos 18564 grupos possíveis, foram filtrados os que tinham perfil semelhante aos 6 alunos do grupo controle, isto é: idade média de 15 anos e horas de uso de video-games semanais entre 15,67 e 17,67. Somente 2036 dos subgrupos possíveis se

[5] O software StageCast Creator pode ser obtido em http://www.stagecast.com/creator.html.

encaixaram nesse perfil, e a partir destes foi feita a avaliação, na qual 81.5% dos subgrupos do grupo experimental equivalentes ao perfil do grupo controle obtiveram nota média acima de 7,75, que foi a nota média obtida pelos participantes do grupo controle.

Foi feita também uma análise comparativa independente do grupo experimental e controle: dos 12 alunos que utilizaram mais de 20 horas de vídeo-game e/ou computador por semana, 75% tiraram a nota máxima do teste, em comparação com 33,33% no grupo que utilizava menos de 20 horas semanais em média, demonstrando que noções de algoritmo estão presentes de forma implícita nos jogos e computadores.

Portanto, a conclusão foi que o uso do PDB para ensino, mesmo que em um minicurso ministrado antes de uma disciplina introdutória à programação, melhorou a compreensão do conteúdo de fato e reduziu significativamente a evasão.

3.2. Estudos abordando Arte Generativa

Os seguintes estudos abordam o ensino e metodologias de desenvolvimento da Arte Generativa.

3.2.1. Ensino de arte generativa para estudantes de graduação

Há trabalhos que afirmam que estudantes de computação não se sentem confortáveis com atividades artísticas ou criativas (EBER; WOLFE, 2000 apud WOOLEY; COLLINS, 2021), em contrapartida, os seguintes trabalhos são sobre o ensino de Arte Generativa para estudantes universitários de graduação, incluindo estudantes de computação.

3.2.1.1. Teaching Generative Art to Undergraduate Students at a Technologically Advanced University: it's challenges and rewards.

Ao lecionar uma disciplina sobre Arte Generativa na University of Advanced Technology, nos Estados Unidos da América, cujo perfil de alunos é muito diverso – alunos que têm experiência em programação, design gráfico, design de jogos ou animação –, Dragojlov (2009) teve como desafio engajar esses diversos perfis de aluno, visto que muitos alunos se inscrevem na disciplina devido a não possuir pré-requisitos e ser considerada uma disciplina fácil, o que seria uma forma simples de que eles cumpram mais créditos escolares.

A disciplina possui parte teórica mas é majoritariamente prática, com atividades incluindo arte com objetos físicos, de modo que os alunos precisam compreender que nem sempre Arte Generativa precisa envolver códigos computacionais. Nesse contexto, os maiores desafios encontrados pelos estudantes foram: a dificuldade de ter uma ideia original; e a dificuldade de se desprender da necessidade de que tudo signifique algo, de aceitar a arte abstrata. Mas, ao superar o desconforto inicial, muitos alunos encontraram na Arte Generativa uma forma de expressão artística.

3.2.1.2. Art for Computer Scientists: Processing as an Open-Source Art Medium for ComputerScience Undergraduates

Em artigo, Wooley e Collins (2021) discorrem sobre um módulo do primeiro ano na graduação em Ciência da Computação pela Keele University chamado CSC-10026: Animation and Multimedia, que incentiva a criatividade dos alunos enquanto aprendem programação, computação gráfica e multimídia interativa. Mesmo pertencendo à graduação em Ciência da Computação, alguns alunos também cursam a disciplina como eletiva; portanto, o grupo tem perfis muito diversos, de pessoas com pouca ou com muita experiência em programação.

A principal ferramenta utilizada nessa disciplina é o Processing – que mesmo sendo uma ferramenta não tão sofisticada, é considerada um bom meio para implementar e explorar conceitos de computação gráfica e animação fundamentais – juntamente com outras ferramentas open-source como GIMP[6], Blender[7], Audacity[8] etc. O módulo ministrado aborda muitos conceitos – trigonometria, geometria, estrutura de dados, processamento de imagem, composição, design, entre outros – e, dentre eles, a Arte Generativa.

Um ponto interessante foi que os autores decidiram utilizar o Processing para realizar o protótipo de um projeto de Realidade Aumentada, visto que foi muito mais simples do que o desenvolvimento com Android Studio[9]– ambiente de desenvolvimento nativo para Android muito utilizado –, já que tinha poucos requisitos. O Android Studio, mesmo sendo uma ferramenta mais completa, pode ser muito complexa de utilizar, e com poucos requisitos necessários para o projeto desenvolvido de um aplicativo de Realidade Aumentada para um museu, as funcionalidades do Processing foram suficientes.

---

[6] Ferramenta de edição de imagens, disponível em https://www.gimp.org.
[7] Ferramenta de processamento gráfico, disponível em https://www.blender.org
[8] Ferramenta de edição de aúdio, disponível em https://www.audacityteam.org
[9] Disponível em https://developer.android.com/studio

Como conclusão, os estudantes responderam positivamente aos conteúdos, à matéria de Animação Computacional e Multimídia e à oportunidade de serem criativos. Também, o Processing pode ser uma ferramenta útil para desenvolvimento de MVPs (do inglês, mínimo produto viável, um projeto que serve como prova de conceito para o desenvolvimento de algo mais robusto) e de aplicações simples.

3.2.2. Re-coding

O *re-coding* consiste no processo de analisar e codificar novamente peças de Arte Computacional mais antigas cujo código não está mais disponível (NAKE; GRABROWSKI, 2017). A prática se dá ao observar uma ou mais imagens provenientes de um mesmo algoritmo e tentar codificar de forma a criar um programa que gere imagens similares à observada. Nos seguintes trabalhos, é discorrido sobre métodos de ensino com base no *re-coding*.

3.2.2.1. Think the Image, Don't Make It! On Algorithmic Thinking, Art Education, and ReCoding

Nesse artigo, Nake e Grabowski (2017) utilizam o método do *re-coding* com o propósito de fazer uma reflexão sobre o processo de criação da Arte Algorítmica – termo similar a Arte Generativa, porém restrito à Arte Computacional – , no qual entender que a imagem faz parte de uma classe de imagens é um passo importante do processo.

Deve-se categorizar imagens em classes, pois pensar na estrutura de uma imagem implica em abstrair tal estrutura em uma mais geral sem se prender a detalhes minuciosos. O intuito não é criar um algoritmo que retorne uma cópia da imagem observada, visto que desenvolver tal algoritmo que só produz uma única imagem, para os autores, seria um exercício sem sentido. Pensar na imagem como uma classe de imagens também levanta alguns questionamentos, como "todas as instâncias de uma classe de imagens são de alto interesse?" e "quais parâmetros podem definir que o algoritmo vá gerar obras de alta qualidade?", que são muito importantes na fase de curadoria da Arte Generativa.

Para re-codificar uma imagem, deve ser feita uma análise artística minuciosa, traduzindo a semiótica da peça em um algoritmo. Sendo algoritmos descrições sequenciais de passos a serem executados, ao observar uma imagem é necessário refletir que passos foram precisos para que ela chegasse ao resultado observado.

O algoritmo deve ter como saída uma instância única de uma classe de imagens que seguem o modelo da peça de exemplo, de forma que esta então seja também uma instância da classe. Esse processo pode ser um exercício de pensamento computacional instigante, por estimular a quebra de um processo complexo em passos lógicos e sequenciais de forma a gerar um algoritmo, exercício muito necessário na computação.

Os autores defendem que com experimentos com *re-coding* são promissoras as chances de introduzir Arte Generativa para os mais diversos grupos, como crianças, estudantes e adultos. Também defendem que para o ensino, não são interessantes imagens-base muito triviais, que podem ser entediantes, nem imagens muito complexas e de difícil codificação e compreensão, visto que o mais interessante é um exemplo desafiador mas alcançável. Isso se dá pela exemplificação de exercícios muito difíceis, que seriam quase impossíveis de re-codificar, mas não foram feitos experimentos com alunos.

3.3. Síntese

Ainda que, na presente seção de trabalhos relacionados, tenham sido citados diversos trabalhos acerca de Arte Generativa, dificuldades com ensino de programação e propostas de ensino de programação alternativos, não foi encontrado nenhum trabalho que tenha avaliação de engajamento, ensino de programação e Arte Generativa simultaneamente. Por isso, se faz necessário o presente trabalho, de modo a validar esse cenário.

4. METODOLOGIA

A partir do problema inicialmente estabelecido de ensinar programação de modo a proporcionar um maior engajamento por meio da Arte Generativa e dos conceitos apresentados anteriormente e levando em consideração que não foram encontrados trabalhos que tenham avaliação engajamento, ensino de programação e Arte Generativa, pode-se propor o presente trabalho. Este trabalho é, portanto, uma pesquisa aplicada situada na grande área de Ciências Exatas e da Terra, de cunho exploratório, caráter quantitativo, realizada em ambiente de campo e que utilizou como metodologia a pesquisa experimental.

Isso se justifica pois a pesquisa experimental é uma metodologia de variáveis controláveis, na qual o pesquisador age ativamente para analisar os efeitos das variáveis sobre o objeto de pesquisa (LAKATOS;MARCONI, 2011). Foi realizada em ambiente de campo e é de caráter quantitativo e exploratório pois foi testada a hipótese do impacto da Arte Generativa no ensino de programação no ambiente de sala de aula, e foi validado com o formulário UES quantitativo.

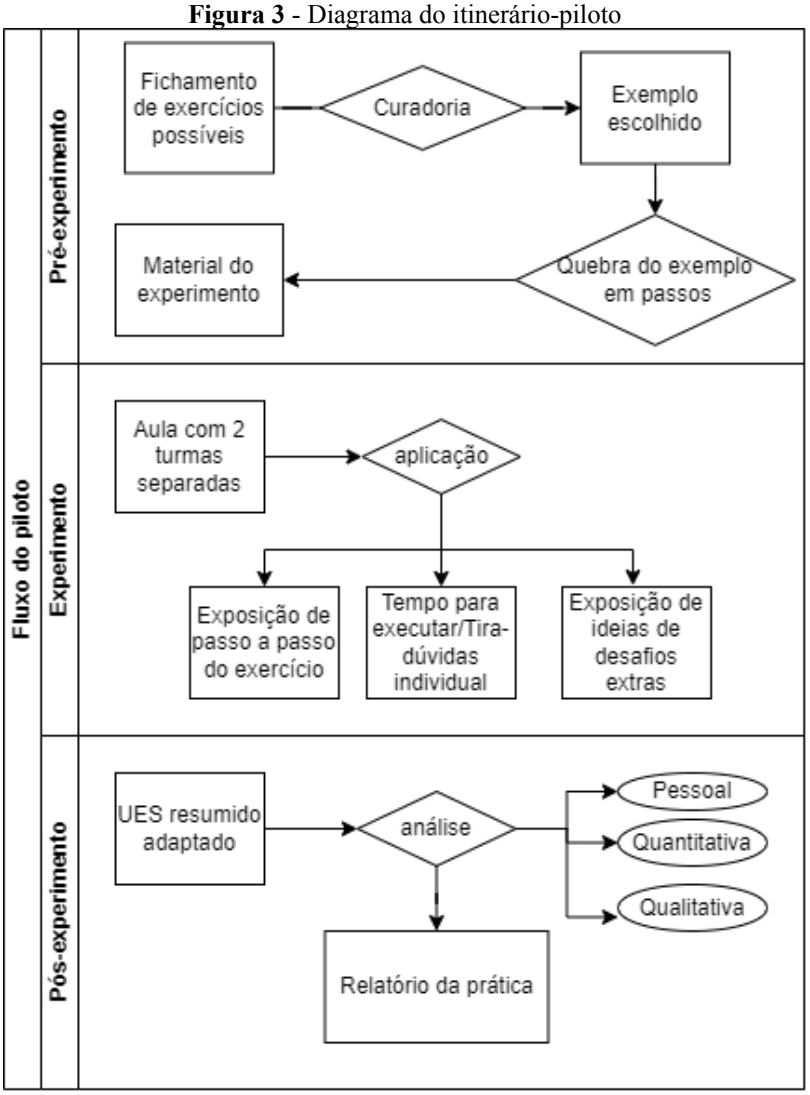
Para uma melhor compreensão da metodologia empregada no desenvolvimento do trabalho realizado, a presente seção se divide em duas subseções, sendo elas a base metodológica e a proposição metodológica.

4.1. Base Metodológica

O presente trabalho tem como objetivo de propor um itinerário com conteúdos, abordagem e cronograma de práticas com Arte Generativa, tendo como objetivos específicos de avaliar experiências de programação criativa, analisar o cruzamento do ensino de programação com Arte Generativa e verificar os impactos do experimento a partir de análise feita com o questionário UES.

Portanto, tendo como objetivo adquirir experiência prática para a realização da oficina final e criar uma base válida para a metodologia, foi elaborado um itinerário para a realização de oficinas-piloto. Todas as oficinas foram realizadas integralmente pela autora do trabalho e a base inicial considerou os resultados dos trabalhos apresentados na seção de trabalhos relacionados. As oficinas-piloto foram aplicadas a alunos calouros do Curso de Bacharelado em Sistemas e Mídias Digitais da Universidade Federal do Ceará, no evento chamado Semana Zero, de introdução ao curso. Todo o itinerário proposto consistiu em três grandes fases: Pré-Experimento, Experimento e Pós-Experimento, como exposto na Figura 3.

Figura 3 - Diagrama do itinerário-piloto



Fonte: Compilação da autora[10].

4.1.1. Pré-Experimento

Durante a primeira fase, de Pré-Experimento, foi feita uma pesquisa de exercícios com Arte Generativa para iniciantes, dentre elas foi escolhido um exemplo de aplicação de aleatoriedade com *noise* (ruído)[11] .Esse exemplo foi escolhido a partir da leitura do livro "O código transcendente" (BERRUEZO, 2019), no qual há diversos exemplos de Arte Generativa em Processing, com explicações teóricas e códigos de fácil execução.

No exemplo escolhido, o uso de ruído para tornar a aleatoriedade aparentemente mais orgânica consistia na geração de círculos coloridos que percorriam a tela em um caminho

[10] Diagrama feito com *draw.io* esquematizando o itinerário-piloto.
[11] Ruído Perlin é um algoritmo que gera uma sequência de números pseudo-aleatórios, inventado pelo Dr. Ken Perlin no ano de 1983 (BERRUEZO, 2019).

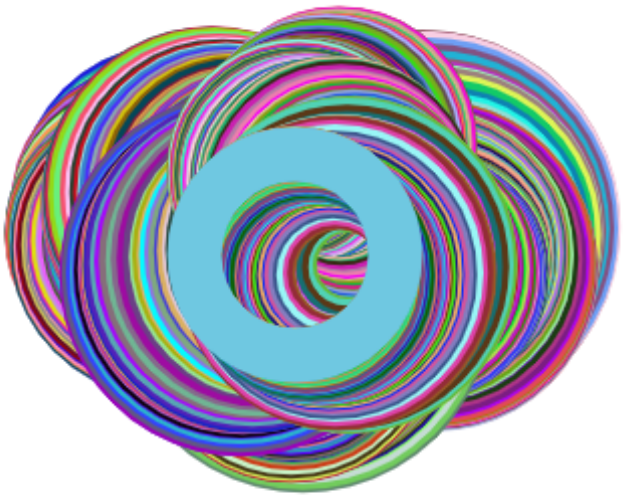
pré-determinado pelo sistema, de acordo com a curva de ruído, como na Figura 5. A partir do exemplo escolhido foi feita a elaboração do material de aula, que foi dividido em quatro momentos: explicação geral; exemplo; usando a criatividade, e responder o questionário.

4.1.2. Experimento

Na segunda fase, a de Experimento, foram realizadas duas aulas com turmas de 22 e 15 alunos, com uma hora de duração, cada. As aulas seguiram o roteiro pré-estabelecido na formação do material de aula, no qual constavam os momentos de explicação geral; exemplo; usando a criatividade e resposta ao questionário, como mencionado anteriormente.

O momento de explicação geral consistiu em uma explicação breve sobre Arte Generativa e Processing, expondo sites como o *OpenProcessing*[12] – no qual há uma exposição de vários experimentos ligados à Arte Generativa. Após a explicação geral, foi iniciado o momento do exemplo prático, no qual o código foi exposto gradualmente, em 8 passos, como visto no apêndice A. Os principais conceitos vistos foram de utilizar as funções *draw* e *setup*[13]; fazer elipses na tela; mudar cores; declarar e usar variáveis, e usar as funções *noise*[14] e *random*[15].

Figura 4 - Execução do código proposto na oficina-piloto adaptado.



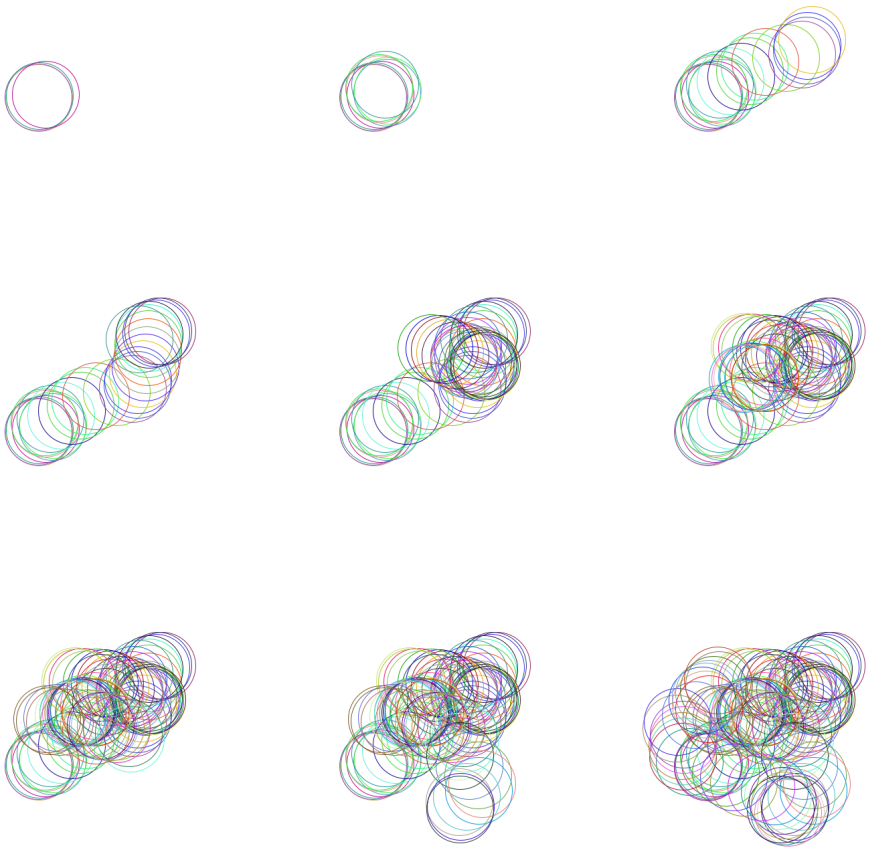
Fonte: Compilação da autora[16]

[12] Disponível em https://openprocessing.org.
[13] Funções do Processing que são responsáveis por toda a exibição do código.
[14] Função nativa do Processing que busca um valor específico na curva de ruído de Perlin gerada.
[15] Função nativa do Processing que gera um número aleatório.
[16] Exemplo de execução do código final proposto executado no *Processing*.

O código do exemplo iniciava com o desenhar de uma elipse na tela, em seguida foi introduzido o conceito de variáveis para armazenar o valor do raio, sendo depois explicado como escolher cores diferentes e alterar a posição para uma específica de modo *hardcoded*[17] e, posteriormente de modo aleatório com a função *random*.

Após esses primeiros passos, foi dada uma breve explicação, que tomou cerca de 5 minutos, acerca do ruído Perlin e da função noise nativa do Processing que gera esse ruído. Depois foi exemplificado como implementar o *noise* no código existente, finalizando o momento de exposição do exemplo (ver figura 5).

Figura 5 - Execução do código proposto na oficina-piloto



Fonte: Compilação da autora[18]

No momento posterior, de usar a criatividade, foi estimulado que os alunos tentassem experimentar com as cores, com a função noise e com diferentes formas geométricas. Foram

[17] Termo do inglês, que simboliza uma prática na qual são utilizados valores fixos dentro do código fonte.
[18] Exemplo de execução do código final proposto executado no *Processing*.

deixados *links* de códigos de autoria próprios (apêndice C), exemplificando implementação dos desafios sugeridos para eventuais alunos mais avançados, sendo estes: modificar a grossura do contorno da elipse de forma dinâmica (ver figura 5); modificar a forma geométrica para retângulo; e utilizar diferentes proporções e modificar as cores de acordo com a posição da forma geométrica.

A prática recorrente da aula funcionou de forma a mostrar um pequeno passo de código, executar, explicar e então observar se os alunos conseguiam fazer o mesmo, tirando dúvidas pontuais individualmente. Esse ciclo se repetiu por volta de oito vezes, sendo essa a quantidade de passos que existiam da fragmentação do exemplo, como explicado previamente e disponibilizado no apêndice A. Durante a explicação, um dos passos consistiu em uma pausa de por volta de cinco minutos da codificação para explicar o que é o ruído e como funciona essa geração aleatória de uma curva.

Mesmo sendo um tanto simples o exemplo escolhido, foi necessário explicar desde o início como utilizar o *Processing* e conteúdos básicos de programação, visto que as turmas de calouros necessitavam de uma aula bem elementar. Por isso, as dúvidas excederam o tempo esperado e foi preciso discutir melhor os fundamentos, de forma que não houve tempo de sobra para que os alunos testassem muitas alternativas ao final.

4.1.3. Pós-Experimento

A etapa de Pós-Experimento foi a fase na qual foi desenvolvida a análise de engajamento da oficina. Ao término das duas oficinas foi aplicado um questionário reduzido UES (apêndice D), através de um formulário eletrônico usando a ferramenta *Google forms,* juntamente com algumas questões discursivas para fazer uma análise cruzada, sendo estas:

- O que você considera mais proveitoso e interessante no emprego de Arte Generativa como base para o ensino de programação?
- Em sua opinião, quais os principais problemas dessa oficina?
- De uma forma geral, como foi sua experiência em relação a aprender a programar empregando recursos de Arte Generativa?
- Quais seriam suas sugestões para as próximas ofertas dessa oficina?

Tal análise foi dividida em três partes: pessoal, quantitativa e qualitativa. Com as análises feitas, foi feito um cruzamento das três análises, resultando na triangulação. As quatro seções seguintes explicam detalhadamente esses passos.

4.1.3.1. Análise pessoal

A análise pessoal, realizada com base na observação da autora deste trabalho, que realizou as oficinas, foi de que na primeira turma os alunos pareciam ter mais conhecimento inicial com programação e que estavam mais interessados em geral, visto que expressaram querer que a oficina durasse mais e exploraram outras possibilidades dentro do escopo mostrado. Devido a imprevistos, como atraso por parte da oficineira e dificuldades com o equipamento de projeção do material de aula, a oficina na primeira turma teve duração de 45 minutos, e não 60 como planejado inicialmente, o que fez com que o conteúdo fosse passado mais rapidamente e de forma mais dinâmica, o que pode ter influenciado na impressão de que os alunos estavam mais interessados.

Em contraponto, a segunda turma, mesmo dispondo de mais tempo de oficina, pareceu mais dispersa, com alguns estudantes com mais experiência e sem interesse e outros ainda pesquisando assuntos de oficinas relacionadas, mas não tão interessados na prática ali proposta.

4.1.3.2. Análise qualitativa

Na análise qualitativa, baseada na avaliação das respostas às questões dissertativas presentes no questionário, os participantes consideraram a oficina uma experiência que "ajuda a aprender a lógica de uma forma mais simplificada e divertida, já que se trata de poder fazer desenhos", e que "aprender programação por meio de arte com estética torna o processo mais interessante".

Na primeira turma, a imersão foi citada por dois participantes diferentes, um alegando que o uso da Arte Generativa "é uma ótima maneira de ensinar os que têm menos conhecimento de uma forma bastante imersiva" e outro que a oficina precisava ter mais tempo "porque precisamos imergir nessa linguagem para saber mais". Em geral foi um consenso entre a turma de que o tempo foi insuficiente, tendo um comentário em específico pedindo "oficina de mais tempo, e com mais exemplos prontos para desbloquear mais ideias de como usar as ferramentas do *processing* de forma criativa".

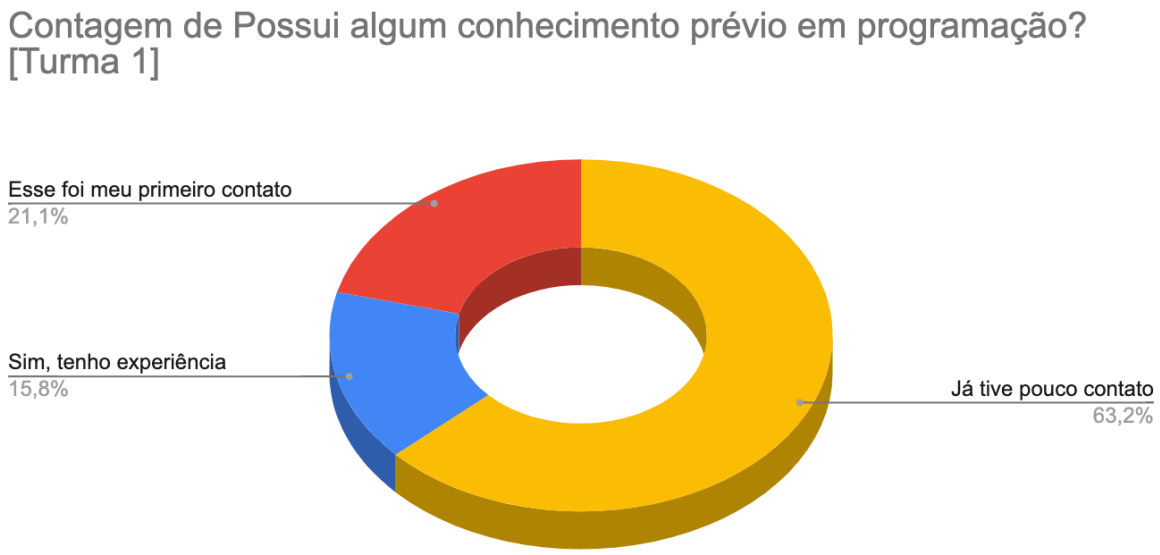
Na segunda turma os participantes consideraram que o ensino de programação dessa forma foi um facilitador para aqueles que tinham o primeiro contato. Foi destacado que tal prática "torna o aprendizado mais atrativo do que simplesmente fazer cálculos repetitivos e

densos - que foi a forma que eu ‘aprendi’ no ensino médio.", pois proporciona "a obtenção de resultados satisfatórios com pouco trabalho para iniciantes em programação". Menos alunos apontaram o tempo como problema, e houve alguns que disseram observar nenhum problema na oficina. A maioria dos alunos alegou ter sido uma experiência muito interessante, ainda que um tenha dito que foi "ótima, apesar de [eu] ter absorvido pouco". Os pontos a melhorar tiveram como consenso geral entre as turmas de que seria ideal "talvez mais tempo para ficar menos corrido" e "mais exemplos" e "incentivar a produzir exemplos".

4.1.3.3. Análise quantitativa

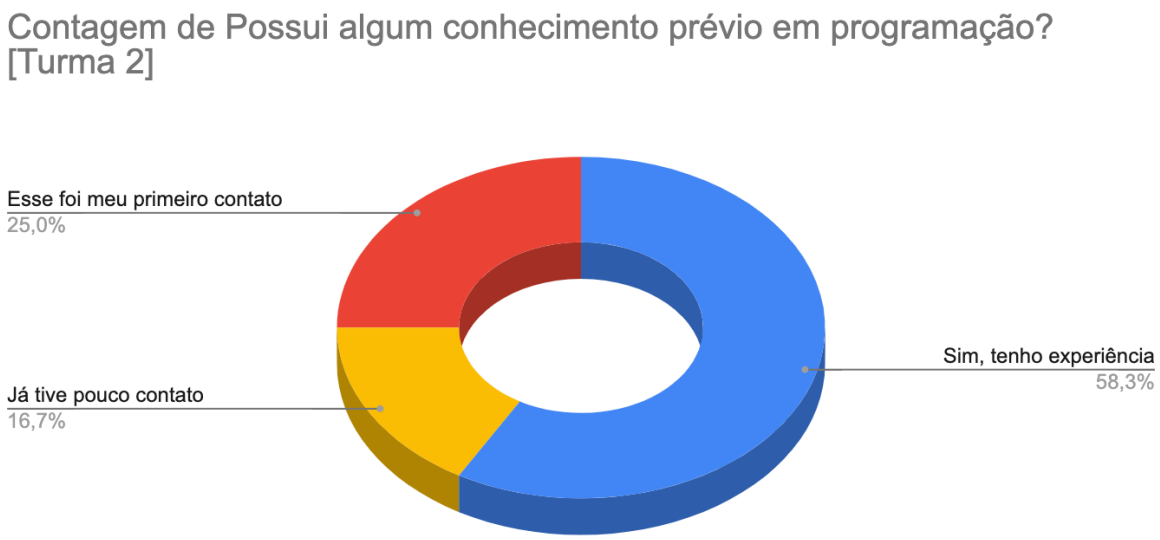
Após a análise qualitativa, foi feita a partir do questionário aplicado uma análise quantitativa. Diferentemente da hipótese levantada análise pessoal de que a primeira turma parecia ter mais experiência prévia com programação, a primeira turma de total de 19 alunos tinha maioria (63%) que haviam tido um primeiro contato com programação, mas somente 3 alunos (16%) de fato tinham experiência, pouco menos que o tanto que teve o primeiro contato com a oficina (21%), ver Gráfico 1. Já na segunda turma, de um total de 12 alunos, a maioria (58%) alegou já ter experiência com programação, seguidos de 25% que estavam tendo o primeiro contato e 16% que tinham pouco contato, ver Gráfico 2.

Gráfico 1 - Conhecimento de programação da turma 01.



Fonte: Questionário aplicado.

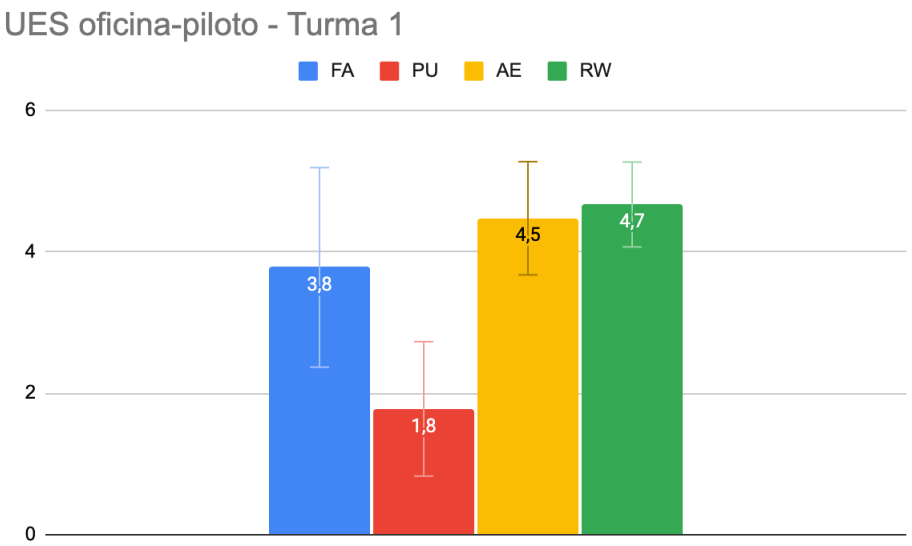
Gráfico 2 - Conhecimento de programação da turma 02



Fonte: Questionário aplicado.

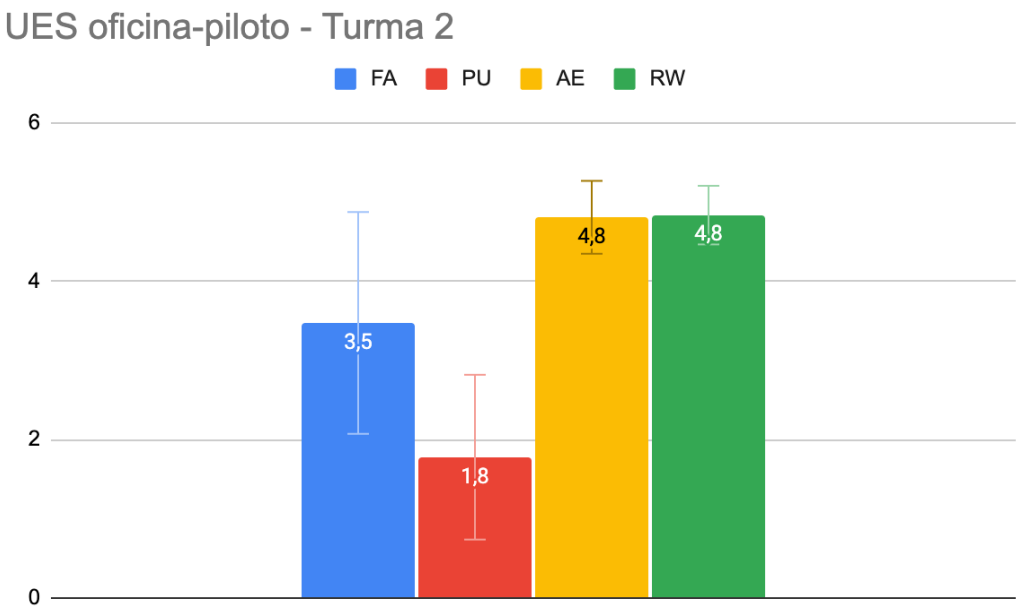
Considerando o UES, para a turma 2 o Fator de Recompensa (RW) foi mais alto, mas não por muito (4.8, ao passo que na turma 1 foi de 4.7). O Apelo Estético (AE) para a segunda turma teve 3 pontos decimais a mais que na primeira (4.8, na primeira 4.5). A Usabilidade Percebida (PU) foi a mesma e a Atenção Focada (FA) na primeira turma foi mais alta por 3 pontos decimais (3.8 na primeira turma e 3.5 na segunda). Portanto, houve médias bem similares, tornando essa parte da análise inconclusiva (ver gráficos 3 e 4).

Gráfico 3 - UES da primeira turma da oficina-piloto



Fonte: Questionário aplicado.

Gráfico 4 - UES da segunda turma da oficina-piloto



Fonte: Questionário aplicado.

O desvio padrão nas respostas da primeira turma foi bem maior do que na segunda (desvio médio de 1 ponto na primeira e de 0.8 na segunda, pode ser observado nos gráficos 3 e 4), muito provavelmente devido à segunda turma ter mais pessoas com experiência, que tendem a ter uma percepção mais parecida, diferentemente da primeira turma, cuja maioria era de pessoas que alegaram ter tido pouco contato com programação, o que é uma faixa mais subjetiva, podendo ter pessoas com níveis de conhecimento mais disparates, o que pode ter causado percepções mais distintas e resultados com desvio padrão maiores por consequência.

4.1.3.4. Triangulação das análises

Após as análises isoladas, foi feito um estudo comparativo com todos os fatores considerados, buscando discrepâncias e correlações.

Uma discrepância encontrada em particular foi a de um dos alunos da segunda turma que já tinha experiência com programação e avaliou o material e a oficina muito bem em geral, mas considerou de 1 a 5 a oficina em 4, como densa; e nas questões abertas declarou que a oficina não teve nenhum problema, e que somente adicionaria mais exemplos distintos: "Buscar outros modelos de arte generativa além dos círculos aleatórios.". Porém, esse aluno declarou neutralidade na questão "me senti imerso nessa oficina".

Uma hipótese para tal ocorrência é que, para ele, talvez, a oficina estava densa em conteúdos e poderia ter sido espalhada com mais exemplos diferentes visuais, aumentando a imersão e a dinâmica. Um outro aluno com o mesmo perfil também declarou que uma sugestão seria "incentivar a produzir exemplos", o que possivelmente traria um maior engajamento com alunos que já têm uma certa experiência em programação.

Em geral, o tempo de atraso na primeira turma influenciou bastante, pois a oficina necessitava todo seu tempo. Na segunda turma, que teve o tempo de uma hora completa, houve sugestão de deixar exemplos a mais para que os alunos se aventurassem. Havia alguns exemplos no material mas eram todos relacionados ao exemplo de classe. Assim, talvez pudessem ser utilizados outros exemplos distintos para serem vistos posteriormente pelos alunos ainda na oficina ou em outros momentos. Em geral a oficina funcionou melhor na primeira turma, tendo tido respostas mais positivas sobre a experiência como "quero estudar mais sobre", que era a meta da experiência, alunos com vontade de aprender mais e aplicar conceitos de programação inclusive em seu tempo livre.

É possível que, por haver mais alunos já com experiência em programação, a segunda oficina obteve um retorno nas questões abertas mais positivo. Ainda, a primeira turma tinha maioria de pessoas com pouca experiência prévia, e aparentou ter mais interesse e interagir mais. Uma hipótese para isso é que, por não ser uma turma de maioria sem nenhum contexto de programação, que poderia ter mais dificuldade, tampouco não tinha maioria de pessoas com muita experiência que pudessem ficar entediados, a matéria foi considerada novidade e empolgante.

Figura 6 - Turma durante uma das oficinas-piloto.

Fonte: Registro da autora.

4.2. Proposição metodológica

A partir da base metodológica, na qual foram realizadas as oficinas-piloto, foram estabelecidos três grupos de ações específicas para reelaborar o itinerário: manter, manter em partes, e descartar. As ações escolhidas foram de manter a introdução à Arte Generativa e a execução individual do código; manter em partes a divisão bem delimitada de passos e o tira-dúvidas individual; e, descartar o público alvo de pessoas com experiência muito contrastante, a não exposição da meta final do exercício e a escolha de um exercício com muitos conceitos novos.

Foi decidido manter a introdução à Arte Generativa pois é muito importante contextualizar bem o que está sendo feito. Pelo mesmo motivo, foi decidido mostrar já no início da oficina como deve ficar o experimento, o código evoluído, ao final. Foi decidido também manter a execução prática do código, pois esse é um fator que promove autonomia aos alunos, já que, como visto no referencial teórico, os alunos tendem a aprender melhor ao praticar, da mesma forma que foi decidido manter em partes a divisão em passos muito bem delimitados. É importante que haja um caminho base a ser percorrido, mas é necessário que o público consiga realizar aquilo sem precisar recorrer a copiar e colar passivamente os códigos fornecidos no material disponibilizado durante a oficina.

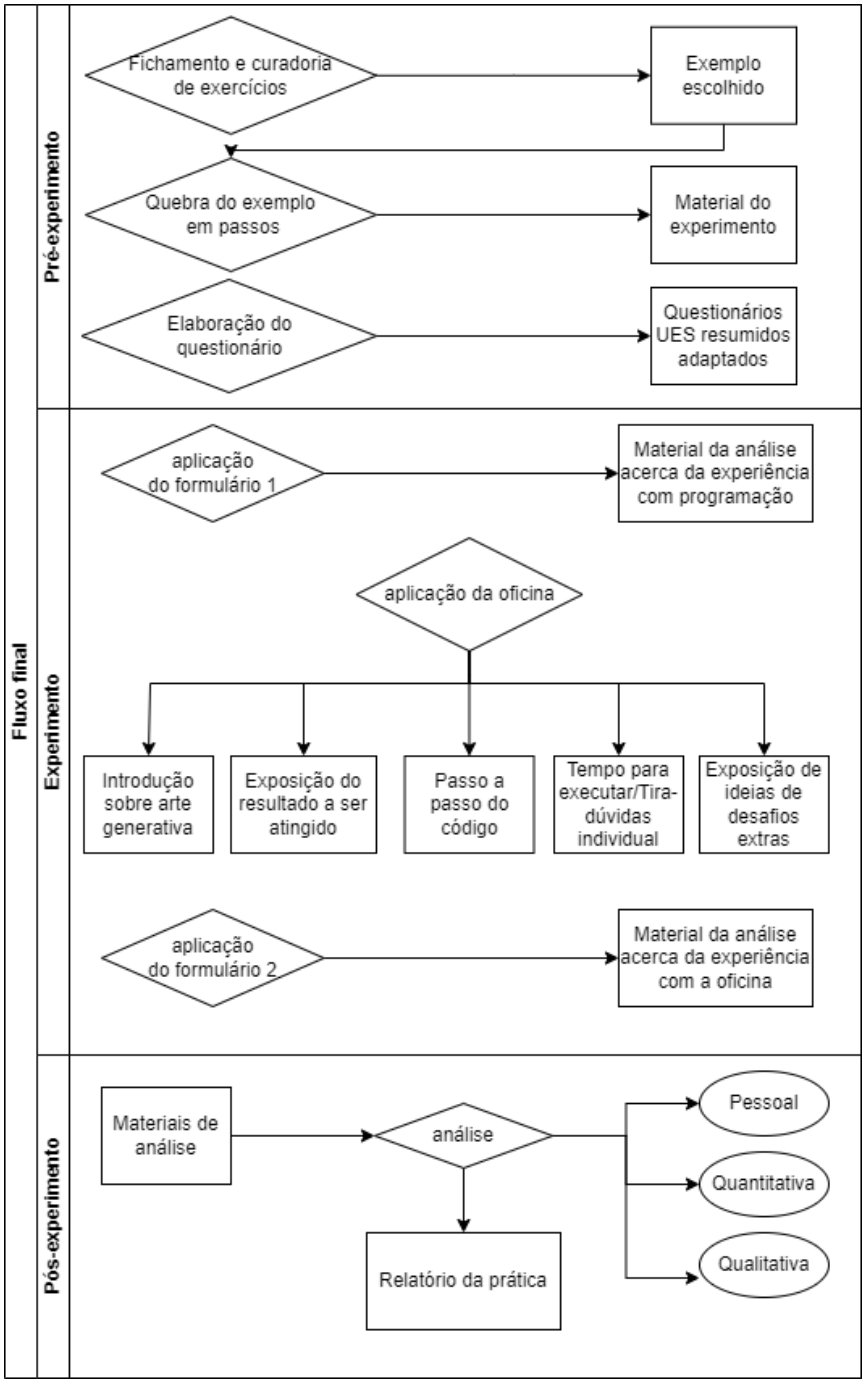
O tira-dúvidas individual foi classificado como manter em partes pois é importante que dúvidas recorrentes sejam explicadas à turma como um todo, para evitar repetição de explicações. Por último, ao observar que, como a turma era muito iniciante, mesmo o exemplo simples introduziu muitos conteúdos novos, percebeu-se que é interessante escolher um exemplo mais familiar, introduzindo somente algumas poucas ideias novas, para facilitar a compreensão geral.

É interessante que o público-alvo tenha um conhecimento mais homogêneo, de forma que o exemplo escolhido seja instigante para a maior parte dos alunos. E é mais desejável propor um exercício simples e com tempo de sobra para que os estudantes possam experimentar por mais tempo do que um exemplo mais elaborado que a turma não conseguirá acompanhar devidamente.

A partir de tais decisões, foram estabelecidas novamente três etapas para continuar os trabalhos de pesquisa: pré-experimento, experimento e pós experimento. No entanto, tais etapas foram refinadas. Assim, a primeira etapa foi dividida em três fases: curadoria de possíveis exercícios, quebra em passos do exercício de forma a elaborar o material de aula e geração do formulário de análise, com base no questionário UES. A segunda etapa consistiu

em três fases: aplicação do formulário acerca da experiência com programação, aplicação da oficina e aplicação do formulário sobre a prática da oficina. E, ao fim, a terceira etapa consistiu em duas fases: análise dos resultados e relatório da prática (ver figura 7).

Figura 7 - Diagrama do itinerário.



Fonte: Compilação da autora[19]

O único instrumento foi, então, o questionário UES reduzido de escala de engajamento de usuários que consiste em 12 (doze) perguntas. Os equipamentos utilizados para a coleta de

[19] Diagrama feito com *draw.io* esquematizando o itinerário.

dados foram os computadores e dispositivos móveis, que foram utilizados na obtenção de dados, visto que os sujeitos precisavam destes equipamentos para responder às perguntas. Ainda, os pesquisadores se reuniram 2 (duas) vezes por mês, durante 15 (quinze) minutos, para discutirem o andamento do projeto.

4.2.1. Pré-Experimento

A primeira etapa, de Pré-Experimento, foi dividida em três fases. Na primeira fase foi feita uma busca e fichamento no livro "O código transcendente" (BERRUEZO, 2019) e em tutoriais simples no *YouTube* por possíveis exercícios a serem aplicados que estimulam o aprendizado de conceitos de programação de forma alternativa e acessível ao público-alvo.

Foram então escolhidos dois exercícios, sendo o primeiro o exercício principal e o segundo um exercício extra, a ser demonstrado caso houvesse tempo que possibilitasse. Isso foi feito tendo em vista que, caso o ritmo da turma fosse mais acelerado, haveria mais possibilidades a serem exploradas; e, caso não, o exercício extra poderia ficar como atividade para eventuais alunos mais adiantados.

Na segunda fase foram estudados os exercícios escolhidos (principal e extra) e estes foram então divididos e organizados em passos mais simples, possibilitando a explicação e o desenvolvimento progressivo no momento da oficina, sendo assim elaborado o material de aula, assim como foi feito nas oficinas-piloto.

Na terceira e última fase foi elaborado o questionário de análise, que possuía as questões objetivas do questionário UES reduzido, algumas questões com intuito de classificar o público da oficina e questões dissertativas, bem como foi feito na oficina-piloto. Um ponto diferente é o de que foi decidido aplicar o mesmo formulário ao início e ao final da prática, sendo a primeira aplicação referente à experiência geral do aluno com ensino de programação até então, para ter uma base de comparação para os resultados do experimento.

4.2.2. Experimento

A segunda etapa, de experimento, foi dividida em três fases: aplicação do formulário acerca da experiência com programação, aplicação da oficina e aplicação do formulário sobre a prática da oficina. A primeira fase consistiu em aplicar o primeiro questionário de análise desenvolvido na etapa de Pré-Experimento acerca da experiência do aluno com o ensino de

programação em geral. Essa fase foi crucial para conseguir perfilar o público-alvo e, em seguida, poder comparar com os resultados da oficina.

Em seguida, na segunda fase foi aplicada a aula tendo como base o exercício escolhido relacionado a Arte Generativa. Com as informações obtidas no experimento-piloto algumas mudanças foram elaboradas na aplicação da oficina, de modo que a aplicação foi dividida em seis passos.

No primeiro passo da fase de experimento foi mostrado inicialmente o resultado que deve ser almejado, em seguida no segundo passo foi fomentada uma reflexão sobre como e o que deveria ser utilizado para produzir aquele efeito, estimulando o pensamento computacional. Esses dois passos foram acrescentados à oficina para contextualizar melhor os alunos.

Após esse momento inicial, no terceiro passo foi demonstrado como desenvolver o código final, tal qual foi feito na oficina-piloto, de modo que no quarto passo foi dado um tempo para execução e tira-dúvidas individual. Por fim, no quinto passo foram mostrados exemplos mais desafiadores e ideias de evolução da prática como foi sugerido pelos alunos das oficinas-piloto.

Na terceira e última fase dessa etapa, foi aplicado o mesmo questionário elaborado com base no questionário UES (*User Engagement Scale*) do início do experimento, sendo agora sobre a percepção dos alunos acerca da oficina, com fim de verificar de quais formas experimentos com Arte Generativa podem influenciar no engajamento dos estudantes universitários do curso de Sistemas e Mídias Digitais na Universidade Federal do Ceará.

4.2.3. Pós-Experimento

A terceira etapa, de Pós-Experimento, se dividiu em 2 (duas) fases: análise e relatório da prática. Nessa etapa foi elaborada a conclusão geral de toda a prática para entender as formas com que oficinas com metodologias alternativas para o ensino de programação, mais especificamente as ligadas a Arte Generativa, podem influenciar no engajamento de estudantes universitários do curso de Sistemas e Mídias Digitais nas disciplinas ligadas à programação.

Na primeira fase da terceira e última etapa, foi feita uma análise pessoal e arbitrária no momento em que a oficina terminou, seguida de uma análise quantitativa dos dados coletados pelo questionário UES e de uma análise qualitativa das questões discursivas consideradas pertinentes para entender melhor o contexto dos estudantes.

Após as três análises feitas, na segunda fase da terceira etapa, foi feita uma triangulação das análises e foi gerado o relatório geral da prática.

5. EXEMPLO DE USO DO ITINERÁRIO

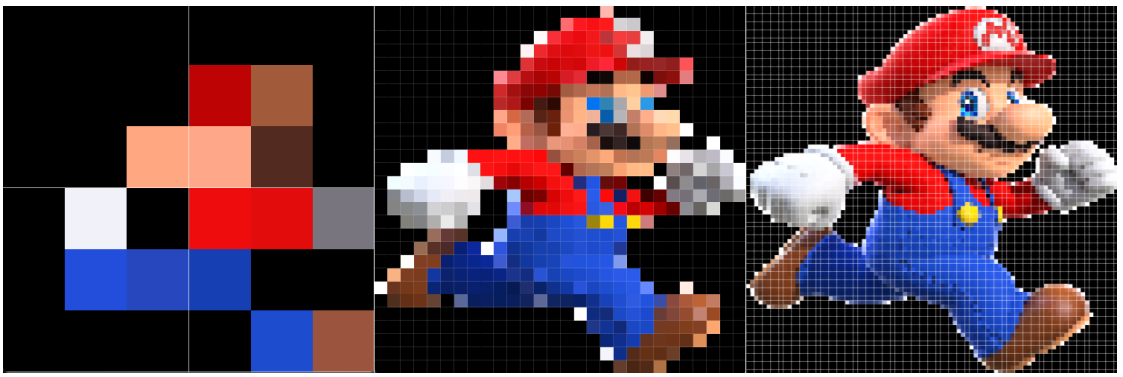
Por fins de organização, o presente trabalho foi dividido de modo que a presente seção consiste no relatório do exemplo de uso do itinerário proposto na seção passada. Essa seção, por sua vez, consiste em detalhar o exemplo de uso do itinerário proposto, dividindo-se em duas subseções, sendo elas o experimento propriamente dito e a análise dos resultados obtidos.

5.1.Experimento

A fim de avaliar o itinerário proposto, foram realizadas duas oficinas no curso de Sistemas e Mídias Digitais. Houve uma dificuldade de encontrar turmas para aplicar a oficina, porém, com o apoio dos professores Clemilson Santos e Melo Júnior, que ministram a disciplina de Matemática Aplicada à Multimídia no Curso de Bacharelado em Sistemas e Mídias Digitais da Universidade Federal do Ceará, foram apresentadas duas oficinas nos dias 23 e 30 de junho de 2022 às 20 e 14 horas, respectivamente, utilizando os horários de aula cedidos pelos professores.

Como o público-alvo da oficina consistia em alunos da disciplina de Matemática Aplicada a Multimídia, que já possui como pré-requisito a disciplina de Programação I; na etapa de Pré-Experimento foi escolhido um experimento um pouco mais avançado que o escolhido nas oficinas-piloto, que eram voltadas aos entrantes no Curso.

Figura 8 - Exercício principal.



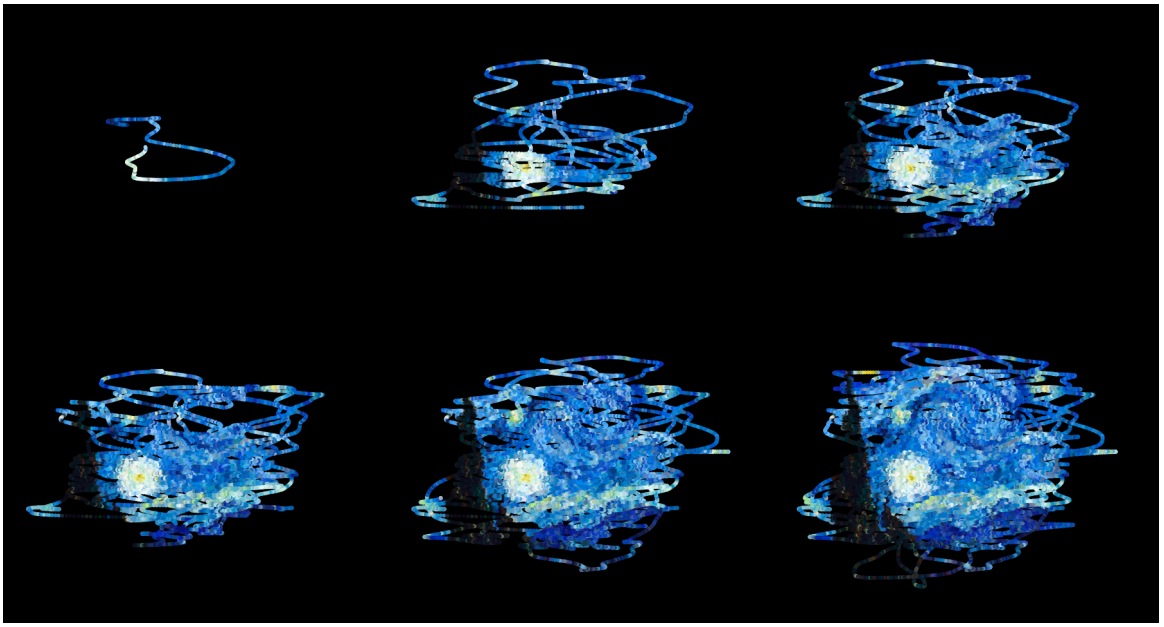
Fonte: Compilação da autora.[20]

O exercício principal escolhido para essas oficinas foi o de preencher uma imagem com um *grid* de quadrados cujas dimensões variavam de acordo com a interação com o

[20] Imagem do código do exercício principal executado no *Processing* com diferentes posicionamentos do *mouse*.

mouse. O exercício extra consistia em preencher uma imagem percorrendo um caminho aleatório novamente determinado pela função *noise*.

Figura 9 - Exercício bônus.



Fonte: Compilação da autora.[21]

Como na oficina-piloto não houve base comparativa para os dados resultados da oficina, foi decidido aplicar um formulário no início acerca da percepção geral do aprendizado de programação até então. Desse modo, em ambas as turmas foi seguido o protocolo de inicialmente pedir aos alunos que respondessem o formulário acerca de sua experiência com aprendizado de programação em geral, em seguida aplicar a oficina e ao final aplicar o mesmo formulário, agora acerca da experiência dos alunos com a oficina, na etapa de Experimento.

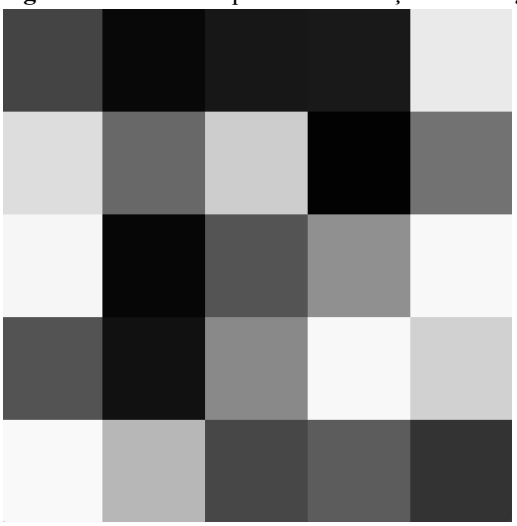
Para a aplicação inicial do formulário foram necessários em média 15 minutos, visto que era necessário esperar que chegassem novos alunos e, a cada aluno novo que chegava, iniciava tardiamente o preenchimento do formulário.

Durante a aplicação da oficina de fato, isto é, na fase de aplicação da oficina, os cinco primeiros minutos foram reservados para explicar um pouco sobre o que é a Arte Generativa e sobre o propósito do experimento. Então, foi iniciada a exposição acerca do exemplo: a atividade principal de formar a imagem do personagem Mario pixelada foi dividida em 4 (quatro) passos, sendo eles a explicação do *grid*, a obtenção das cores baseado na imagem

[21] Imagem do código do exercício bônus compilado no decorrer do tempo.

original, a alteração do tamanho das células do *grid* e a interatividade com o *mouse*. O código propriamente dito está disponível no Apêndice D.

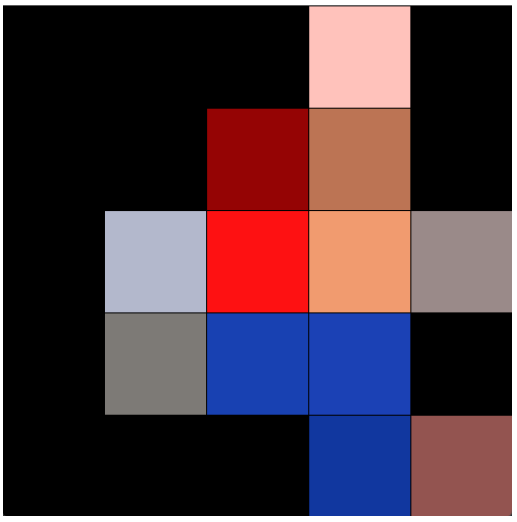
Figura 10 - Primeiro passo de execução do código.



Fonte: Compilação da autora.[22]

No primeiro passo, de explicação acerca do *grid,* foi mostrado de forma incremental como funciona a estrutura de laço *for*[23], gerando aos poucos uma fileira com 5 quadrados e depois realmente uma matriz 5x5 de quadrados em tons de cinza, gerando o *grid* monocromático inicial (ver Figura 10).

Figura 11 - Segundo passo de execução do código.



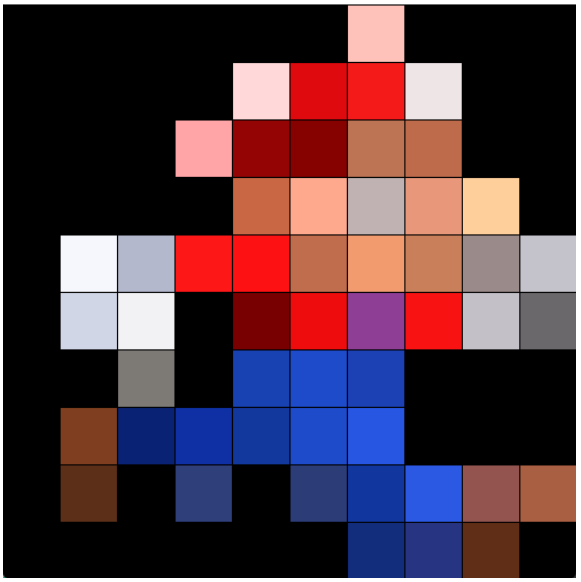
Fonte: Compilação da autora.[24]

[22] Imagem do primeiro passo do código do exercício compilado.
[23] Estrutura de repetição amplamente utilizada no desenvolvimento de sistemas.
[24] Imagem do segundo passo do código do exercício compilado.

Posteriormente, no segundo passo foi explicado como utilizar a classe *Image* nativa da ferramenta *Processing,* de modo a obter a cor correspondente de uma coordenada específica. Nesse estágio os alunos conseguiram que seu *grid*, até então composto de quadrados cinzas, tivesse semelhança à silhueta do personagem Mario da imagem base devido as cores, como na Figura 11.

Por fim, nos terceiro e quarto passos foi demonstrado como alterar a proporção do *grid* e introduzida a variável de acompanhamento do *mouse*, que foi utilizada como parâmetro de modo a promover a interatividade, como na figura 12.

Figura 12 - Terceiro e quarto passos de execução do código.

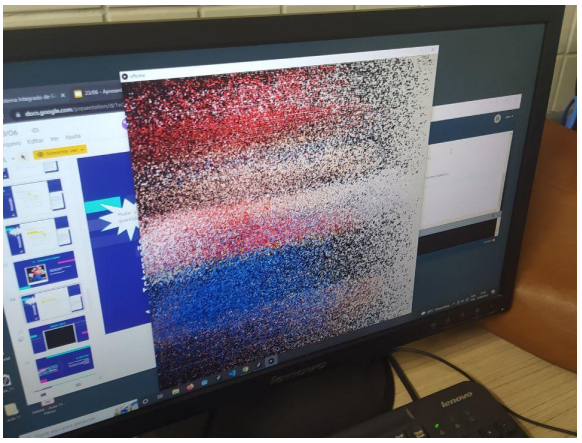


Fonte: Compilação da autora.

Cada um dos passos foi explicado de modo que a realizadora da oficina, novamente a autora do presente trabalho, mostrou o código inicialmente, depois o executou e mostrou o resultado aos participantes, empregando um projetor multimídia. Após isso, os alunos foram estimulados a repetir os mesmos códigos. Feito isso, foi incentivado que eles fizessem suas próprias alterações, tentando gerar resultados inusitados e únicos (como o exemplo da Figura 13).

Esse ciclo durava em média 10 minutos para cada passo, e em sendo 4 passos, durou 40 minutos no total, que somados aos 15 minutos iniciais totalizaram 55 minutos. Sendo a primeira turma uma aula no último período da noite, e tendo em vista que os alunos ainda teriam que responder o questionário final, foi decidido não aplicar o exemplo extra, totalizando o tempo de 65 minutos do experimento, visto que o preenchimento do questionário final implicou em 10 minutos adicionais.

Figura 13 - Experimento de um aluno.



Fonte: Compilação da autora.

A segunda turma, em contraponto, por ser no período da tarde e ter menos integrantes (somente 4), possibilitou que houvesse a exposição do último exercício, que precisou da explicação sobre *noise*, que durou cinco minutos, e mais dois passos de execução – o preenchimento aleatório da imagem e a exemplificação do uso do noise – somando 25 minutos ao tempo de aula, totalizando 90 minutos.

5.2. Resultados

Bem como nas oficinas-piloto, os resultados foram divididos em três análises: pessoal, qualitativa e quantitativa. Após as análises, foi feita uma triangulação, portanto a presente seção possui quatro subseções.

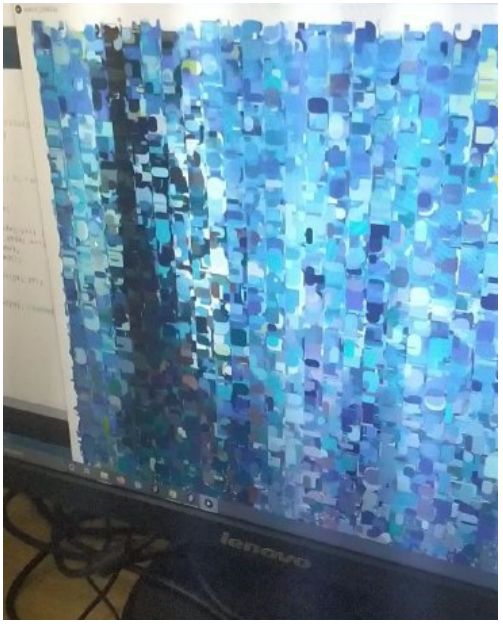
5.2.1. Análise pessoal

A análise pessoal, que consistiu nas impressões da oficineira, ou seja, a autora do trabalho, foi de que nas duas turmas houve real interação com o experimento, na primeira turma, alunos experimentaram com formas diferentes e com as imagens do Mario e da Noite Estrelada, de Van Gogh, como pode ser observado nas Figuras 14 e 15. A primeira turma tinha mais alunos que eram de semestres mais avançados e que, talvez por isso, interagiram mais e demonstraram mais interesse, ainda que ambas turmas tenham sido participativas.

Ainda assim, na segunda turma também houveram experimentos interessantes, um aluno baixou uma imagem de um jogo, editou no photoshop e aplicou no código, outro aluno fez experimentos com canais de cor, algo que não tinha sido inicialmente foco da oficina.

Sendo assim, a oficina despertou interesse e curiosidade para que os alunos buscassem além do que estava sendo explicado em aula.

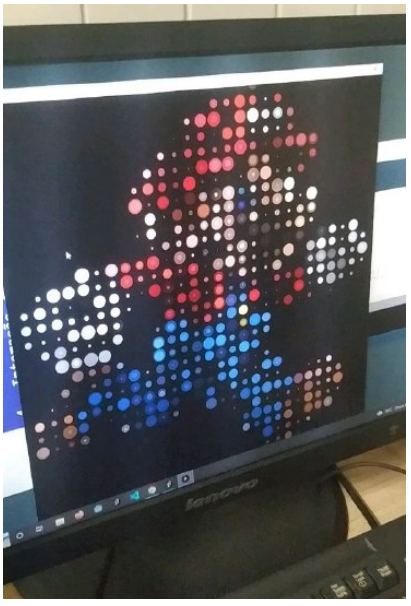
Figura 14 – Experimento de um aluno com formas arredondadas.



Fonte: Compilação da autora.

Ainda, a primeira turma, por ser maior, tinha um grupo de alunos mais focados e um grupo que parecia ter mais dificuldades com a prática, enquanto que a segunda turma, por ser menor, pareceu estar mais nivelada em relação tanto a experiência com programação em geral quanto ao interesse na prática em si.

Figura 15 – Experimento de um aluno com círculos em tamanhos aleatórios.



Fonte: Compilação da autora.

5.2.2. Análise qualitativa

Considerando as respostas às questões dissertativas presentes no formulário, no geral, os alunos apontaram como principais problemas com os seguintes:

- aprendizado de programação o excesso de teoria com poucas práticas;
- a falta de interesse dos próprios alunos;
- a dificuldade em acompanhar, visto que algumas turmas são “niveladas por cima” e é suposto que os alunos têm conhecimentos prévios;
- a falta de flexibilidade dos métodos avaliativos.

Os alunos alegaram frustração com o aprendizado de programação, mas que uma vez que se aprende pode se tornar muito prazeroso. Porém, como o processo pode ser desgastante, apontaram que seria ideal uma forma de flexibilização do ensino, com mais práticas contextualizadas, estímulo ao aprendizado e respeito ao ritmo individual.

Com relação à avaliação da oficina, a segunda turma, o dia 30/06, cuja oficina foi apresentada na íntegra, não apresentou críticas relevantes, e como principais *feedbacks* positivos disseram se sentir imersos, gostar bastante, ter várias ideias e "vontade de gerar algo bonito", o que indica que o quesito de Apelo Estético foi muito provavelmente atingido.

Já na primeira turma, o ponto mais recorrente foi que os participantes gostariam que houvesse mais tempo para que pudessem ter testado o exemplo extra. Uma pessoa alegou que o material estava um pouco confuso também.

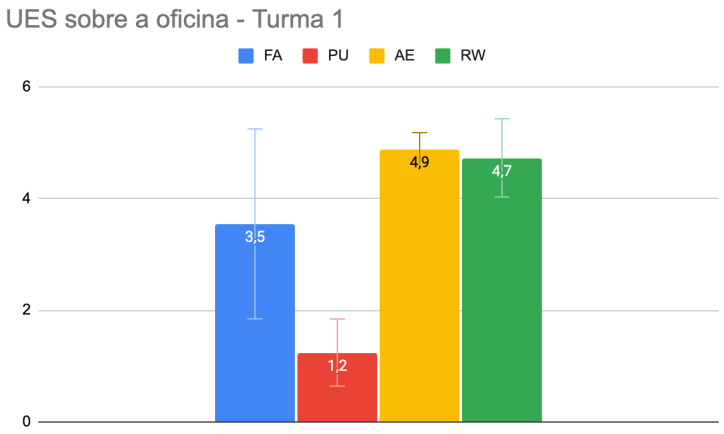
Mas, em geral, o resultado foi muito positivo, com *feedbacks* de que foi “incrível, acaba sendo mais interessante que o modelo tradicional de ensino”, que ficaram “experimentando mil coisas diferentes” e até um aluno citou que durante a oficina se sentiu “muito mais imerso e tentado a tentar novas alternativas de geração da arte por meio das mudanças no código pelo fato do visual atrair bastante.”.

5.2.3. Análise quantitativa

Considerando o UES reduzido, analisando resultados do formulário respondido pelos participantes, em que FA, PU, AE e RW se referem respectivamente a Atenção Focada – se sentir imerso na interação e perder a noção do tempo –, Usabilidade Percebida – afeto negativo vivenciado como resultado da interação e do nível de controle e de esforço gastos –, Apelo Estético – a atratividade e o apelo visual da interface – e Fator de Recompensa –

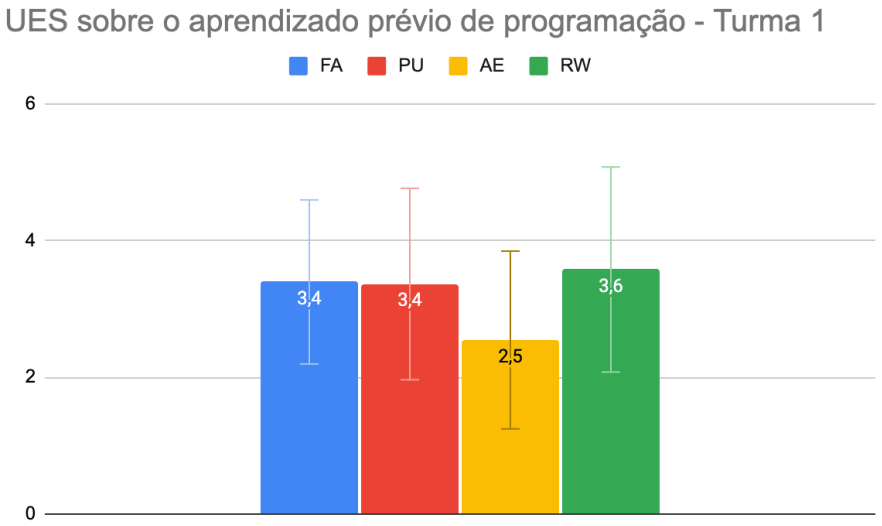
incluindo o senso de novidade, curiosidade e interesse, o envolvimento percebido e a durabilidade do engajamento – pode-se perceber que em geral as oficinas tiveram resultados mais positivos que a experiência inicial com programação (ver Gráficos 5, 6, 7 e 8).

Gráfico 5 - Análise do formulário UES sobre a oficina da primeira turma.



Fonte: Material de análise.

Gráfico 6 - Análise do formulário UES sobre programação da primeira turma.

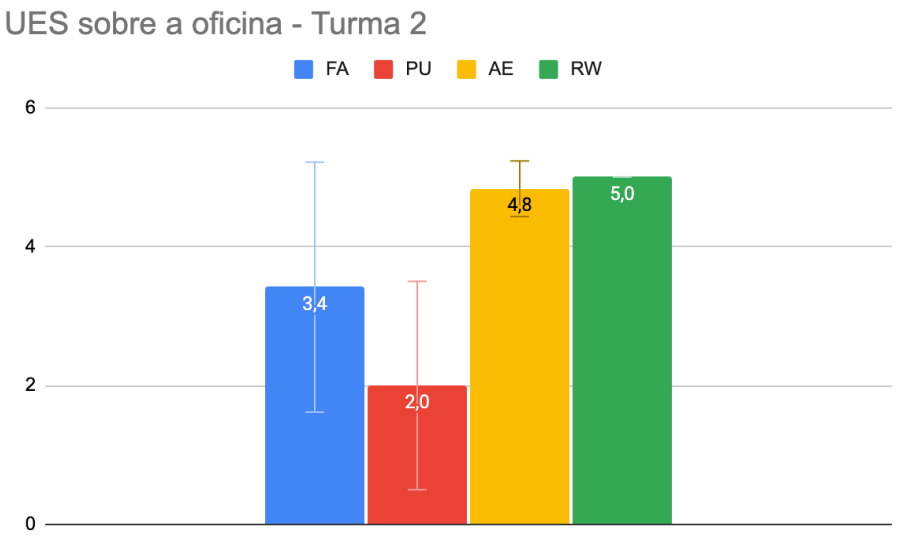


Fonte: Material de análise.

Considerando que a Atenção Focada (FA) nas oficinas-piloto foi de média de 3.65, e os valores permaneceram na média de 3.45 na oficina final, houve uma queda de 0.2. Porém, no fato de estética (AE), nas oficinas-piloto a média foi de 4.65, enquanto nas oficinas finais foi de média de 4.85, ou seja, um ganho de 0.2. O Fator de Recompensa (RW) aumentou em média 0.1 e a Usabilidade Percebida (PU), que é uma métrica de impacto negativo, caiu em

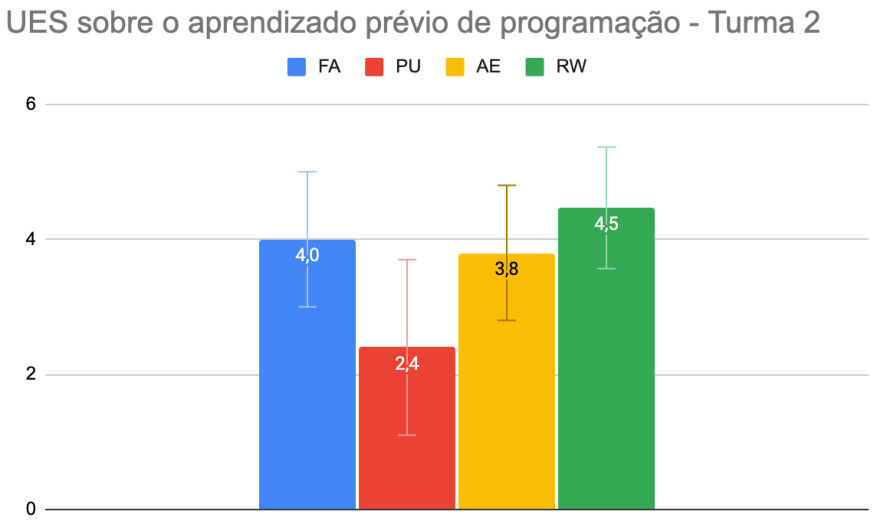
0.2. Ou seja, todos os fatores tiveram melhora de 0.2 ou 0.1 pontos em comparação com as oficinas-piloto (ver Gráficos 5, 6, 7 e 8), em exceção da Atenção Focada.

Gráfico 7 - Análise do formulário UES sobre a oficina da segunda turma.



Fonte: Material de análise.

Gráfico 8 - Análise do formulário UES sobre programação da segunda turma.



Fonte: Material de análise.

Portanto, em geral as oficinas finais tiveram impacto mais positivo em comparação com as oficinas-piloto, indicando que a aplicação de oficinas base de fato foi um procedimento válido para consolidar um modelo de itinerário mais refinado.

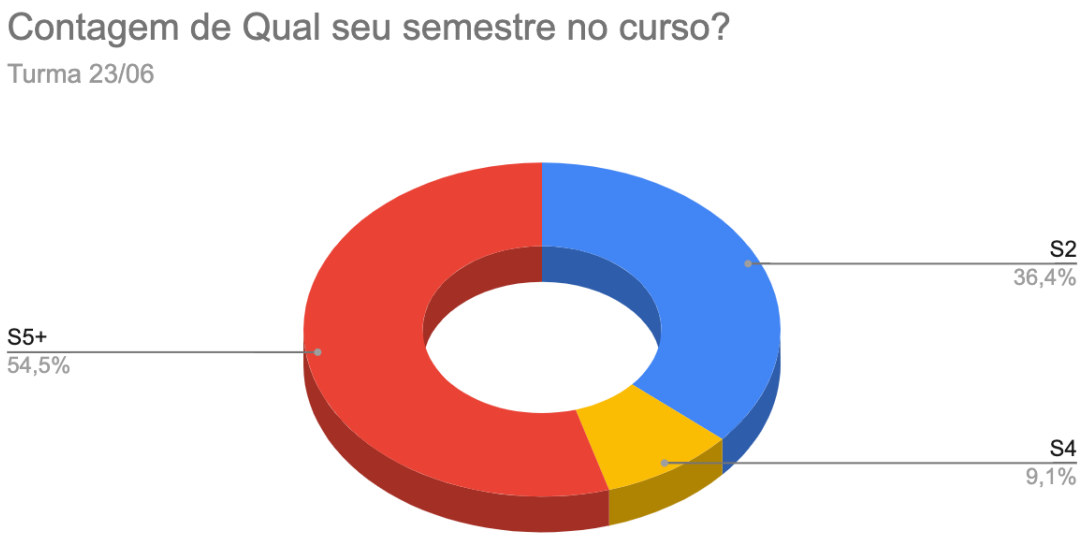
Essa subseção está dividida de modo a discorrer individualmente sobre o resultado de cada uma das turmas, para melhor organização.

5.2.3.1. Primeira Turma

A primeira turma, cuja oficina foi aplicada no dia 23/06/2022, e teve o total de 11 alunos, possuía maioria de alunos (55%) do quinto semestre em diante, seguido de 36,4% do segundo semestre, e 9% (1 aluno) do quarto semestre (ver Gráfico 9). Mesmo assim, 81,8% da turma considerou ter pouco contato apenas com programação, indicando que não é uma turma experiente em programação, mesmo que tenham muito tempo de curso, isso é possível devido ao caráter multidisciplinar do curso de Sistemas e Mídias Digitais.

Analisando os resultados do formulário UES da primeira turma, percebe-se que as respostas acerca da experiência com a oficina tiveram desvio padrão de no máximo 1,3 pontos e mínimo de 0, sendo uma questão em que todos concordaram, e média de 0,6 pontos de desvio padrão.

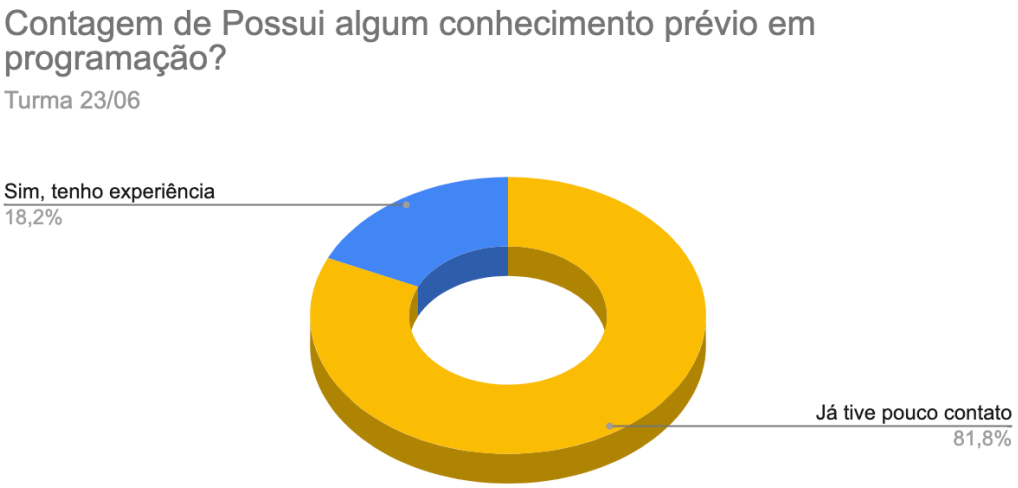
Gráfico 9 – Distribuição dos alunos por semestre da primeira turma.



Fonte: Material de análise.

Em contraponto, na mesma turma as respostas relacionadas à experiência com programação tiveram mínima de 1.1 e máxima de 1.5 de desvio padrão, com média de 1,4 de desvio. Isso muito provavelmente ocorreu pois enquanto a oficina é a mesma sendo avaliada por alunos diferentes, portanto tendo variações só pelo referencial de cada um, a experiência com ensino de programação em geral pode ter sido muito mais distinta, visto que nem todos os alunos aprenderam com os mesmos professores ou com as mesmas técnicas.

Gráfico 10 – Conhecimento prévio em programação da primeira turma.

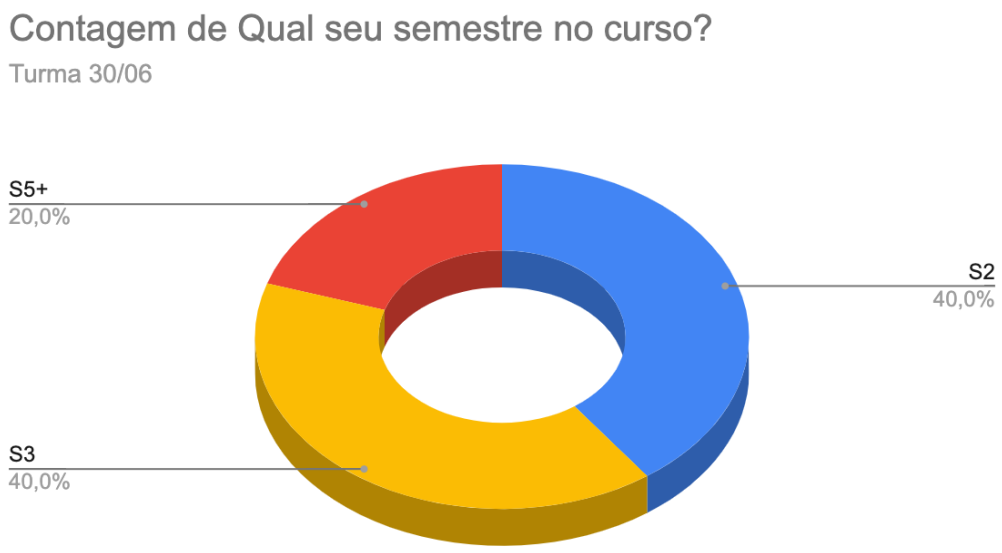


Fonte: Material de análise.

5.2.3.2. Segunda turma

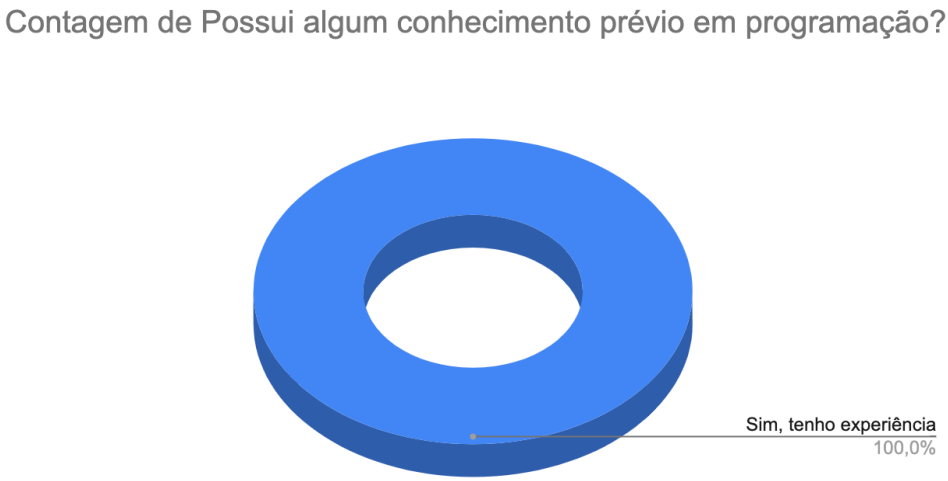
Essa oficina, aplicada no dia 30/06/2022, iniciou com 5 alunos, sendo 2 do segundo semestre, 2 do terceiro e um de um semestre superior ao quinto, no entanto tendo esse aluno abandonado a prática logo no início (Gráfico 11). Quanto ao conhecimento em programação, 100% da turma alegou ter experiência prévia, como pode ser visto no Gráfico 12.

Gráfico 11 – Distribuição dos alunos por semestre da segunda turma.



Fonte: Material de análise.

Gráfico 12 – Conhecimento prévio em programação da segunda turma.



Fonte: Material de análise.

Com relação ao desvio padrão, a segunda turma, tal qual a primeira, apresentou um valor muito mais significativo no questionário acerca da experiência com programação, com média de 1 ponto e máxima de 1,3, enquanto que no questionário sobre a oficina o desvio foi de 0,3 em média, com máxima de 0,6. A hipótese, conforme já discutido, é que a experiência prévia com programação é muito individual, visto que os alunos podem ter vindo de contextos muito distintos, podendo ter tido experiências muito mais heterogêneas com relação ao aprendizado em geral em comparação com a oficina que foi aplicada para a turma simultaneamente.

5.2.4. Triangulação

A triangulação consiste numa análise geral a partir das outras três análises apresentadas anteriormente, visando evidenciar padrões e disparidades de modo a levantar hipóteses e constatações finais acerca do experimento.

Desse modo, uma das disparidades encontradas foi a da primeira questão do formulário, que foi "me senti perdido nessa oficina", essa questão é muito aberta para interpretação, pois pode ser considerada como estar perdido na vertente de se sentir confuso (que seria algo negativo) ou estar perdido no sentido de estar muito imerso (que seria algo positivo). Na avaliação do UES essa questão é considerada positiva, e a grande maioria dos alunos respondeu que discorda totalmente da afirmação, ainda que as demais respostas do

formulário tenham sido positivas. A tradução do formulário e interpretação são pontos cruciais e que podem ter afetado os resultados.

Outro ponto que chamou atenção foi uma aluna da primeira turma avaliou nas questões dissertativas que a experiência foi melhor que a forma tradicional, mas que o material estava um pouco confuso. Porém, a mesma aluna nas questões do formulário UES, concordou totalmente com as afirmações de que o material da oficina teria a agradado e sido esteticamente agradável. Em geral, as respostas dessa usuária foram muito controversas, declarando não ter se sentido nenhum pouco imersa, mas tampouco frustrada, por exemplo.

A aluna em questão tinha pouco contato com programação e era de um semestre superior ao quinto do curso, o que pode indicar ser uma pessoa que já teve dificuldades prévias com programação em geral, por isso considerou que foi uma forma melhor que a tradicional, mas que ainda tem uma certa dificuldade. Mesmo assim, ela declarou que um ponto positivo foi "ver algo mais visual sendo construído a partir do código que você mesmo digitou", ou seja, utilizar um contexto diferente para expor os conceitos foi algo positivo.

Algo curioso foi também o aluno de semestre mais avançado que estava presente ao início da oficina na segunda turma e que foi o único desistente da oficina como um todo. Foi levantada a hipótese de que, para pessoas com mais experiência, a oficina não seja tão instigante, e por isso o aluno desistiu, o que foi intrigante pois a primeira turma tinha 55% de sua composição de alunos do quinto semestre em diante, e teve *feedbacks* muito positivos, parecendo inclusive estar mais empolgada do que a turma com alunos de semestres mais iniciais, conforme visto nas seções anteriores.

Porém, ao observar a pergunta acerca do conhecimento prévio em programação, na segunda turma 100% declarou ter experiência, o que indica ser uma turma com mais experiência efetiva em programação, ainda que não tão avançada no curso. Os resultados dessa turma foram um pouco abaixo dos da primeira turma em geral (em exceção do Fator de Recompensa), e, mesmo que por poucos décimos, essa pode ser uma validação da hipótese levantada. Portanto, faria sentido que pessoas com mais experiência com programação não se sentissem tão instigadas pelo fato de não ter a sensação de novidade que pessoas com menos contato tem.

O aluno em questão, nas questões dissertativas sobre a experiência com aprendizado de programação, sugeriu que seria mais proveitoso um ensino com ênfase na matemática, lógica e física. Uma hipótese adicional é que esse aluno não tenha tanto interesse em aspectos visuais, o que é esperado, nem todos os alunos aprendem de forma similar e se instigam pelos mesmos temas.

Em geral, os resultados foram muito positivos, já nas oficinas-piloto os resultados tinham sido muito satisfatórios, e, como visto anteriormente, o resultado das questões do formulário UES indicaram que o resultado das oficinas finais foram ainda melhores que os das oficinas-piloto.

Mesmo com mais tempo e mais conteúdo sendo explicado na segunda turma, a primeira turma das oficinas finais teve um aproveitamento maior que ela, devido a oficina ter sido elaborada focando mais em pessoas que não tenham tanta experiência em programação, o que indica que o público alvo deve ser bem estudado ao escolher a prática. O desafio deve sempre ser alcançável porém nunca simples a ponto de ser entediante.

6. CONSIDERAÇÕES FINAIS

O presente trabalho propôs um itinerário com conteúdos, abordagem e cronograma de práticas de ensino de programação com base em Arte Generativa com o objetivo de impactar no engajamento dos estudantes do curso de Sistemas e Mídias Digitais da Universidade Federal do Ceará (UFC).

Para tanto, inicialmente foram analisados experimentos apresentados em outros trabalhos passados que lidaram com Arte Generativa e ensino de programação. Feito isso, foram realizados experimentos que consistiram em uma série de oficinas com conteúdos de Arte Generativa visando aumentar o engajamento dos alunos de disciplinas de programação. Foram no total quatro turmas, com média de uma hora de oficina cada, obtendo a participação 53 alunos.

Para avaliar os impactos dos experimentos, foi utilizado um formulário que empregou o protocolo UES reduzido. Além disso, o formulário também contou com questões dissertativas para entender melhor o contexto de cada grupo de estudantes e conseguir interpretar melhor e de forma mais assertiva as respostas do formulário UES reduzido (ver apêndice D).

Foi relatada pelos participantes das oficinas uma dificuldade inicial de perseverar com o estudo de programação, mas os mesmos alunos se sentiram instigados a pesquisar mais quando expostos a uma prática em um contexto em que estão mais curiosos e se sentem mais instigados para experimentar com, isto é, a arte.

Como resultados dos experimentos, das observações pessoais da autora do presente trabalho, que ministrou as oficinas, e da análise das respostas dos formulários, pode-se verificar que o ensino com experiências práticas guiadas com base em Arte Generativa pode ser muito positivo. Isso se dá pois os resultados tanto quantitativos, por meio do questionário UES reduzido, quanto qualitativos, por meio de questões dissertativas no formulário de *feedback* tiveram resultados significativamente impactantes, como o exemplo do depoimento de um aluno de que se sentiu “tentado a tentar novas alternativas de geração da arte por meio das mudanças no código pelo fato do visual atrair bastante”. Foi observado também que uma boa parcela dos alunos participantes nas oficinas interagiu bastante com o experimento, buscando criar versões alternativas próprias.

No entanto, dada a pandemia de COVID-19 ainda no período atual, foi um pouco incerta a comunicação para conseguir alocação e espaço para ministrar as oficinas, e um dos principais problemas encontrados foi o do curto tempo disponível. Nas quatro turmas, o

tempo disponível era de em média uma hora e, com as aplicações de questionários e eventuais imprevistos, este foi um empecilho encontrado.

Uma outra questão crucial foi a de selecionar exercícios que fossem de simples compreensão para o público em geral, sendo ao mesmo tempo instigantes mas de fácil execução. Poderia ser um problema de mais fácil resolução se o perfil dos alunos fosse bem delimitado; mas, com a dificuldade de conseguir turmas para o experimento, veio o fato de que as turmas das oficinas acabaram apresentando perfis muito diversos, o que dificultou o processo de entender qual exercício seria ideal para a maior parte do grupo e de estimar precisamente o tempo que levaria para explicar e tirar as dúvidas de todos. Com isso, foi constatado que, quanto mais homogêneas forem as turmas e mais tempo estiver disponível para a realização da oficina, provavelmente mais proveitosa será a experiência.

Uma outra dificuldade encontrada foi a de mensurar bem os impactos das oficinas realizadas, sendo necessário empregar questões específicas nos questionários sobre a experiência prévia com o aprendizado de programação para se ter uma linha de base. Porém, essa é uma experiência muito individual pois os alunos vêm de contextos muito diferentes antes da faculdade, o que ficou evidente pelo desvio padrão elevado nos questionários aplicados ao início das oficinas sobre a experiência prévia com ensino de programação.

Apesar disso, os resultados obtidos não se tornam inválidos. Pelo contrário, eles demonstram que, independente do perfil, os participantes em geral tiveram uma recepção positiva à prática e em geral foram bastante engajados por ela.

Sendo assim, é importante dar continuidade aos trabalhos aqui relatados de modo a acompanhar o desempenho de uma mesma turma ao longo de um período, realizando múltiplas oficinas e tendo assim uma base melhor de comparação dos impactos das práticas propostas. Seria interessante também poder realizar testes com um grupo controle e um grupo experimental, ou seja, separar a turma em dois grupos equivalentes e apresentar uma prática tradicional de ensino para o grupo controle e uma prática alternativa com base em Arte Generativa para o grupo experimental, comparando ao fim o engajamento dos dois grupos.

Uma prática futura interessante seria a de realizar uma atividade de *re-coding* com uma imagem escolhida pela turma, dando tempo para que os alunos desenvolvessem individualmente seu código, sem necessariamente seguir um passo a passo apresentado, e sim criando seu próprio algoritmo desde o início.

Com este trabalho, espera-se contribuir com a comunidade acadêmica, e em geral, de modo que as aulas introdutórias de programação sigam um modelo um pouco mais flexível,

possivelmente com práticas relacionadas a temas que engajem mais os estudantes, como é o caso da Arte Generativa.

REFERÊNCIAS

BEREITER, C.; NG, E. Three levels of goal orientation in learning. **Journal of the Learning Sciences**, p. 243-271, 1991. Disponível em: <https://doi.org/10.1080/10508406.1991.9671972.>. Acesso em 31 ago. 2022.

BERRUEZO, M. P. **O código transcendente**: Uma introdução prática à programação e arte gerativa. Belo Horizonte: MIT Press, 2019. 268 p.

BORGES, M. A. F. Avaliação de uma metodologia alternativa para a aprendizagem de programação. **Workshop de Educação em Computação–WEI**, Curitiba, v. 8, p. 15, 2000.

BRASIL, MEC. **Diretrizes Curriculares de Cursos da área de Computação e Informática**. MEC–SESU (Secretaria de Educação Superior). CEEInf–Comissão de Especialistas de Ensino de Computação e Informática. Brasília, 2001.

BRASSCOM - Associação Brasileira de Empresas de Tecnologia de Informação e Comunicação. **Formação Educacional e Empregabilidade em TIC:** Achados e Recomendações. 2002. Disponível em: <https://brasscom.org.br/wp-content/uploads/2019/09/BRI2-2019-010-P02-Formacao-Educacional-e-Empregabilidade-em-TIC-v83.pdf>. Acesso em 21 nov. 2021.

BUREAU OF LABOR STATISTICS. **Computer and information technology**. Estados Unidos, 2020. Disponível em:<https://www.bls.gov/ooh/computer-and-information-technology/home.htm>. Acesso em 21 de nov. 2021.

CÔRTE VITÓRIA, M. I.; CASARTELLI, A.; RIGO, R. M.; COSTA, P. T. Engajamento acadêmico: desafios para a permanência do estudante na Educação Superior. **Educação**, v. 41, n. 2, p. 262-269, 2018.

DE OLIVEIRA JÚNIOR, J. G.; NORONHA, R. V.; KAESTNER, C. A. A. Método de seleção de atributos aplicados na previsão da evasão de cursos de graduação. **Revista de Informática Aplicada**, v. 13, n. 2, 2017.

DIJKSTRA, EDSGER W. On the Cruelty of Really Teaching Computing Science. **Communications of ACM**, v. 32, n. 12, p. 1398-1404, 1988.

DRAGOJLOV, V. Teaching Generative Art to Undergraduate Students at a Technologically Advanced University: Its challenges and rewards. **Generative Art Conference**, Milão, nº 12, p. 392-397, 2009.

FERREIRA, C.; GONZAGA, F.; SANTOS, R. Um estudo sobre a aprendizagem de lógica de programação utilizando programação por demonstração. **Anais do Workshop sobre Educação em Computação**, v. 18, XXX CSBC, Belo Horizonte, 2010.

FOSSILE, D. K. Construtivismo versus socio-interacionismo: uma introdução às teorias cognitivas. **Revista Alpha**, Patos de Minas, UNIPAM, 2010.

GALANTER, P. What is generative art? Complexity theory as a context for art theory. **Generative Art Conference,** Milão, nº 6, 2003, Milão.

GALANTER, P. . Complexism and the role of evolutionary art. **The Art of Artificial Evolution,** p. 311-332, Springer, Berlin, Heidelberg, 2008.

GARCIA, G.; LOPES, A. **Introdução à Programação: 500 algoritmos resolvidos.** Rio de Janeiro: Elsevier, 2002.

GOMES, A., HENRIQUES, J., MENDES, A. J. Uma proposta para ajudar alunos com dificuldades na aprendizagem inicial de programação de computadores. **Educação, Formação & Tecnologias**; v.1, n.1, p. 93-103, 2008. Disponível em http://eft.educom.pt

HAACKE, H. **Condensation Cube**, acrylic, plastic, water, climate in area of display, 12 × 12 × 12 in, 1963-1965.

HAACKE, H. Condensation Cube, **Leonardo**, v. 36, n. 4, p. 265-265, 2003.

HADEN, P.; MANN, S. The Trouble with Teaching Programming, **Annual NACCQ**, nº 16, 2003, Palmerston North, 2003. Proceedings.

HANSEN, S. M. Deconstruction/Reconstruction: A pedagogic method for teaching programming to graphic designers, **Generative Arts Conference**, p. 419-431, 2017.

HERTLEIN, G. C. Computer art for computer people-a syllabus. **Annual conference on Computer Graphics and Interactive Techniques**, San Jose, nº 4, p. 249-254, 1977. Proceedings of SIGGRAPH '77. San Jose, CA: 1977.

JAIMES A., LALMAS M., e VOLKOVICH Y. First international workshop on social media engagement (SoME 2011). **International Conference companion on World Wide Web (WWW'11)**, nº 20, 2011, New York. Proceeding. New York: ACM Press, 2011, p. 309.

JÚNIOR, J. C. P. *et al* (2005). Ensino de algoritmos e programação: Uma experiência no nível médio. **Workshop de Educação em Computação (WEI'2005)**, São Leopoldo, v. 8, p. 1–12, 2005

LAKATOS, E. M.; MARCONI, M. A. **Metodologia científica**. 6. ed. São Paulo: Atlas, 2011.

MONTFORT, N.; BAUDOIN, P., BELL, J.; BOGOST, I.; DOUGLASS, J. **10 PRINT CHR $(205.5+ RND (1));: GOTO 10**. MIT Press, 2014.

NAKE, F. Paragraphs on Computer Art, Past and Present. **CAT 2010: Ideas before their time : Connecting the past and present in computer art** (CAT), 2010.

NAKE, F; NEES, E. The Pioneer of Generative Art: Georg Nees. **Leonardo**, v. 51, n. 03, p. 277-279, 2018.

NAKE, F.; GRABOWSKI, S. Think the image, don't make it! On algorithmic thinking, art education, and re-coding. **Journal of Science and Technology of the Arts**, v. 9, n. 3, p. 21-31, 2017.

O'BRIEN, H. L.; TOMS E. G. The development and evaluation of a survey to measure user engagement. **Journal of the American Society for Information Science and Technology**, v. 61, n. 1, p. 50-69, 2010. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.21229. Acesso em: 05 ago. 2022.

O'BRIEN, H. L.; CAIRNS, P.; HALL, M. A practical approach to measuring user engagement with the refined user engagement scale (UES) and new UES short form. **International Journal of Human-Computer Studies**, v. 112, p. 28-39, 2018. Disponível em: https://www.sciencedirect.com/science/article/pii/S1071581918300041. Acesso em: 31 ago. 2022.

PAPERT, S. **A máquina das crianças**: repensando a escola na era informática. Porto Alegre: Artes Médicas, 2008.

PAPERT, S. **Documentário**, 1983.
Disponível em: <https://www.youtube.com/watch?v=bOf4EMN6-XA&feature=youtu.be&t=1m27s>. Acesso em 21 nov. 2021.

PITEIRA, M.; HADDAD, S. R. Innovate in your program computer class: An approach based on a serious game. **ACM International Conference Proceeding Series**, p. 49–54, 2011.

RAPKIEWICZ, C. E.; FALKEMBACH, G. A. M.; SEIXAS, L. M. J. D.; SANTOS, N. D. S. R. S. D.; CUNHA, V. V. D.; KLEMANN, M. Estratégias pedagógicas no ensino de algoritmos e programação associadas ao uso de jogos educacionais. **RENOTE: revista novas tecnologias na educação [recurso eletrônico]**, Porto Alegre, 2007.

REAS, C.; FRY, B. **Processing**: a programming handbook for visual designers and artists. Cambridge, MA: Mit Press, 2007.

SANTOS, R. P.; COSTA, A.X.; RESENDE, A. M. P.; SOUZA J. M. O Uso de Ambientes Gráficos para Ensino e Aprendizagem de Estruturas de Dados e de Algoritmos em Grafos. **Congresso da Sociedade Brasileira de Computação**, Belém, nº 28, 2008. Anais do XVI Workshop sobre Educação em Computação. Belém: Sociedade Brasileira de Computação – SBC, 2008, p. 157-166.

TAYLOR, C. **How Harvey Mudd Transformed Its Computer Science Program - And Nearly Closed Its Gender Gap**.
Disponível em: <https://techcrunch.com/2013/10/10/how-harvey-mudd-transformed-its-computer-science-program-and-nearly-closed-its-gender-gap/>. Acesso em 21 nov. 2021.

TING C.; CHEAH W.; HO C. C. Student engagement modeling using bayesian networks. **IEEE International Conference on Systems, Man, and Cybernetics**, Manchester, nº 13, p. 2939-2944, 2013. Proceedings. Manchester: IEEE, 2013, p. 2939–2944.

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

WOOLEY, S.; COLLINS, T. Art for Computer Scientists: Processing as an open-source art medium for computer science undergraduates. **Proceedings of EVA London 2021**, London, p.177-182, 2021.

XAVIER, E. A.; FOSS, L.; CAVALHEIRO, S. A. da C.; SOARES, M. A. da S.; ROMIO, L. C.. Pensamento Computacional integrado à Matemática na BNCC: proposta para o primeiro ano do Ensino Fundamental. **SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO**, nº 32 , 2021, Online. Anais. Porto Alegre: Sociedade Brasileira de Computação, 2021, p. 989-1001.

ZYNGIER, D. (Re)conceptualising student engagement: Doing education not doing time, **Teaching and Teacher Education**, v. 24, n. 7, p. 1765-1776, 2008.

APÊNDICE A – Listagem dos códigos da oficina-piloto

O código da oficina foi proposto em passos, os itens a seguir seguem a ordem dos códigos que foram apresentados.

1.

```
void setup () {
        print("Olá");
}
void draw () {
        //print("mundo");
        ellipse(0, 0, 10, 10);
}
```

2.

```
void setup () {
        size(500, 500);
        background(255, 255, 255);
}
void draw () {
        ellipse(0, 0, 100, 100);
}
```

3.

```
void setup () {
        size(500, 500);
        background(255, 255, 255);
        noFill();
}

void draw () {
        int raio = 100;
        stroke( ?, ?, ? );
        ellipse(200, 200, raio, raio);
}
```

4.

```
void setup () {
        size(500, 500);
        background(255, 255, 255);
        noFill();
}

void draw () {
        int raio = 100;
        stroke(random(255), random(255), random(255));
        ellipse(200, 200, raio, raio);
}
```

5.

```
int raio;
void setup () {
        size(500, 500);
        background(255, 255, 255);
        noFill();
        raio = 100;
}

int posX = 200;
int posY = 200;

void draw () {
        posX = posX + 2;
        stroke(random(255), random(255), random(255));
        ellipse(200, 200, raio, raio);
}
```

6.

```
int raio;
```

```
int posX;
int posY;

void setup () {
        size(500, 500);
        background(255, 255, 255);
        noFill();
        raio = 100;
}

void draw () {
        posX = random(width);
        posY = random(height);
        stroke(random(255), random(255), random(255));
        ellipse(200, 200, raio, raio);
}
```

7.

```
int raio, posX, posY, noiseX, noiseY;

void setup () {
        size(500, 500);
        background(255, 255, 255);
        noFill();
        raio = 100;
        noiseX = 27;
        noiseY = 73;
}

void draw () {
        posX = int(noise(noiseX)*width);
        posY = int(noise(noiseY)*height);
        stroke(random(255), random(255), random(255));
        ellipse(200, 200, raio, raio);
```

```
}
```

8.

```
int raio, posX, posY;
float noiseX, noiseY;

void setup () {
    size(500, 500);
    background(255, 255, 255);
    noFill();
    raio = 100;
    noiseX = 27;
    noiseY = 73;
}

void draw () {
    posX = int(noise(noiseX)*width);
    posY = int(noise(noiseY)*height);
    stroke(random(255), random(255), random(255));
    ellipse(200, 200, raio, raio);
    noiseX = noiseX + 0.06;
    noiseY = noiseY + 0.02;
}
```

APÊNDICE B – Códigos de inspiração da oficina-piloto

Primeira alternativa[25]:

```
int raio, posX, posY;
float noiseX, noiseY, noiseWeight;
void setup () {
    size(500, 500);
    background(255, 255, 255);
    noFill();
    raio = 100;
    noiseX = 27;
    noiseY = 73;
    noiseWeight = 0;
}

void draw () {
    posX = int(noise(noiseX)*width);
    posY = int(noise(noiseY)*height);
    stroke(random(255), random(255), random(255));
    strokeWeight(pow(2, noise(noiseWeight)*10));
    ellipse(posX, posY, raio, raio);
    noiseX = noiseX +0.003;
    noiseY = noiseY + 0.002;

    noiseWeight = noiseWeight + 0.02;
}
```

Segunda alternativa[26]:

```
int raio, posX, posY;
float noiseX, noiseY, noiseSize;
void setup () {
    size(500, 500);
```

[25] Acesso ao código compilado disponível em https://openprocessing.org/sketch/1519689.
[26] Acesso ao código compilado disponível em https://openprocessing.org/sketch/1519695.

```
        background(25, 25, 25);
        noFill();
        raio = 20;
        noiseX = 27;
        noiseY = 63;
        noiseSize = 0;
    }

    void draw () {
        posX = int(noise(noiseX)*width);
        posY = int(noise(noiseY)*height/2);
        stroke(random(255), random(255), random(255), 100);
        rect(posX, posY, raio/noiseSize, raio*noiseSize);
        noiseX = noiseX +0.003;
        noiseY = noiseY + 0.002;

        noiseSize = noiseSize + 0.02;
    }
```

Terceira alternativa[27]:

```
    int raio, posX, posY;
    float noiseX, noiseY, noiseSize;
    void setup () {
        size(500, 500);
        background(25, 25, 25);
        noFill();
        raio = 20;
        noiseX = 27;
        noiseY = 63;
        noiseSize = 0;
    }

    void draw () {
```

[27] Acesso ao código compilado disponível em https://openprocessing.org/sketch/1519712.

```
        posX = int(noise(noiseX)*width);
        posY = int(noise(noiseY)*height/3);
        stroke(noise(noiseX)*255, random(100), 255 -noise(noiseX)*255, 100);
        rect(posX, posY, raio/noiseSize, raio*noiseSize);
        noiseX = noiseX +0.006;
        noiseY = noiseY + 0.002;

        noiseSize = noiseSize + 0.02;
}
```

APÊNDICE C – Questionário completo aplicado

O questionário foi dividido em quatro setores: dados pessoais, questões objetivas e questões dissertativas.

1. Dados pessoais:
    1. Qual seu semestre no curso?
        a. S1
        b. S2
        c. S3
        d. S4
        e. S5+
    2. Qual sua identidade de gênero?
        a. Mulher
        b. Homem
        c. Não binário
        d. Prefere não informar
    3. Qual sua faixa etária?
        a. Menor de 18 anos
        b. 19 a 24 anos
        c. 25 a 35 anos
        d. Mais de 35 anos
    4. Possui algum conhecimento prévio em programação?
        a. Sim, tenho experiência
        b. Já tive pouco contato
        c. Esse foi meu primeiro contato

2. Questões objetivas do formulário UES reduzido:
    1. Senti-me perdido nessa oficina
    2. O tempo durante essa oficina passou voando
    3. Senti-me imerso nessa oficina
    4. Senti-me frustrado nessa oficina
    5. Achei a oficina confusa
    6. Achei a oficina densa

7. Achei o material da oficina atrativo
8. Achei o material da oficina esteticamente agradável
9. O material da oficina me agradou
10. Participar da oficina valeu a pena
11. A experiência foi gratificante
12. Senti-me interessado na oficina como um todo

3. Questões dissertativas:
    1. O que você considera mais proveitoso e interessante no emprego de arte generativa como base para o ensino de programação?
    2. Em sua opinião, quais os principais problemas dessa oficina?
    3. De uma forma geral, como foi sua experiência em relação a aprender a programar empregando recursos de arte generativa?
    4. Quais seriam suas sugestões para as próximas ofertas dessa oficina?

APÊNDICE D – Código proposto nas oficinas finais

Foram propostos dois exemplos: o exemplo principal e o exemplo bônus.

Exemplo Principal – Pixelização do personagem Mário:

O Exemplo Principal foi dividido em quatro etapas: formação do Grid, aplicação das Cores, variação de tamanho do Grid e aplicação da interatividade. Cada etapa foi dividida em múltiplos passos, como exposto a seguir.

1 – Formação do Grid

1.1.

```
void setup () {
    size(500, 500);
    background(255, 255, 255);
    noStroke();
}

void draw () {
    background(255, 255, 255);

    int posicaoX = 0;
    int posicaoY = 0;
    int lado = 100;

    fill(random(255));
    rect(posicaoX, posicaoY, lado, lado);

    fill(random(255));
    rect(posicaoX + lado, posicaoY, lado, lado);

    fill(random(255));
    rect(posicaoX + lado*2, posicaoY, lado, lado);
```

```
}
```

1.2.

```
void setup () {
    size(500, 500);
    background(255, 255, 255);
    noStroke();
}

void draw () {
    background(255, 255, 255);

    int posicaoX = 0;
    int posicaoY = 0;
    int lado = 100;

    fill(random(255));
    rect(posicaoX + lado*0, posicaoY, lado, lado);

    fill(random(255));
    rect(posicaoX + lado*1, posicaoY, lado, lado);

    fill(random(255));
    rect(posicaoX + lado*2, posicaoY, lado, lado);
}
```

1.3.

```
void setup () {
    size(500, 500);
    background(255, 255, 255);
    noStroke();
}

void draw () {
```

```
        background(255, 255, 255);

        int posicaoX = 0;
        int posicaoY = 0;
        int lado = 100;

        int x = 0;
        fill(random(255));
        rect(posicaoX + lado*x, posicaoY, lado, lado);

        x++;
        fill(random(255));
        rect(posicaoX + lado*x, posicaoY, lado, lado);

        x++;
        fill(random(255));
        rect(posicaoX + lado*x, posicaoY, lado, lado);
    }
```

1.4.

```
    void setup () {
        size(500, 500);
        background(255, 255, 255);
        noStroke();
    }

    void draw () {
        background(255, 255, 255);

        int posicaoX = 0;
        int posicaoY = 0;
        int lado = 100;
```

```
        for(int x = 0; x < 5; x++) {
                fill(random(255));
                rect(posicaoX + lado*x, posicaoY, lado, lado);
        }
}
```

1.5.

```
void setup () {
        size(500, 500);
        background(255, 255, 255);
        noStroke();
}

void draw () {
        background(255, 255, 255);

        int posicaoX = 0;
        int posicaoY = 0;
        int lado = 100;

        for(int x = 0; x < 5; x++) {
                fill(random(255));
                rect(posicaoX + lado*x, posicaoY + lado*0, lado, lado);
        }

        for(int x = 0; x < 5; x++) {
                fill(random(255));
                rect(posicaoX + lado*x, posicaoY + lado*1, lado, lado);
        }
}
```

1.6.

```
void setup () {
        size(500, 500);
        background(255, 255, 255);
        noStroke();
}

void draw () {
        background(255, 255, 255);

        int posicaoX = 0;
        int posicaoY = 0;
        int lado = 100;

        for(int y= 0; y < 5; y++) {
                for(int x = 0; x < 5; x++) {
                        fill(random(255));
                        rect(posicaoX + lado*x, posicaoY + lado*y, lado, lado);
                }
        }
}
```

2 – Aplicação das Cores

2.1.

```
PImage img;

void setup() {
        size(500, 500);
        img = loadImage("mario.png");
        img.resize(width, height);
}

void draw(){
        //image(imagem, 0, 0);
```

```
        float x = mouseX;
        float y = mouseY;
        color c = img.get(int(x), int(y));
        fill(c);
        rect(x,y,30,30);
}
```

2.2.

```
PImage img;

void setup() {
        size(500, 500);
        img = loadImage("mario.png");
        img.resize(width, height);
}

void draw(){
        background(255, 255, 255);

        int lado = 100;

        for(int y = 0; y < 5; y++) {
                for(int x = 0; x < 5; x++) {
                        int pX = int(x*lado);
                        int pY = int(y*lado);

                        color c = img.get(pX, pY);

                        fill(c);
                        rect(pX, pY, lado, lado);
                }
        }
}
```

3 – Variação de tamanho do Grid

3.1.

```
PImage img;

void setup() {
    size(500, 500);
    img = loadImage("mario.png");
    img.resize(width, height);
}

void draw(){
    background(255, 255, 255);

    int lado = 100/2;

    for(int y = 0; y < 5*2; y++) {
        for(int x = 0; x < 5*2; x++) {
            int pX = int(x*lado);
            int pY = int(y*lado);

            color c = img.get(pX, pY);

            fill(c);
            rect(pX, pY, lado, lado);
        }
    }
}
```

3.2.

```
PImage img;
```

```
void setup() {
    size(500, 500);
    img = loadImage("mario.png");
    img.resize(width, height);
}

void draw(){
    background(255, 255, 255);

    int qtde = 10;
    int lado = width/qtde;

    for(int y = 0; y < qtde; y++) {
        for(int x = 0; x < qtde; x++) {
            int pX = int(x*lado);
            int pY = int(y*lado);

            color c = img.get(pX, pY);

            fill(c);
            rect(pX, pY, lado, lado);
        }
    }
}
```

4 – Aplicação da interatividade

4.1.

```
PImage img;

void setup() {
    size(500, 500);
    img = loadImage("mario.png");
```

```
        img.resize(width, height);
    }

    void draw(){
        background(255, 255, 255);

        float qtde = mouseX/5;
        float lado = width/qtde;

        for(int y = 0; y < qtde; y++) {
            for(int x = 0; x < qtde; x++) {
                int pX = int(x*lado);
                int pY = int(y*lado);

                color c = img.get(pX, pY);

                fill(c);
                rect(pX, pY, lado, lado);
            }
        }
    }
```

Exemplo bônus – Preenchimento da imagem com *noise*

O exemplo bônus também foi dividido em passos incrementais, listados a seguir.

1.

```
    void setup(){
        size(600, 480);
        background(0);
        noStroke();
    }
```

```
void draw(){
	for(int i = 0; i < 500; i++){
		float x = random(width);
		float y = random(height);
		fill(255, 255, 255);
		ellipse(x,y,6,6);
	}
}
```

2.

```
PImage imagem;

void setup(){
	imagem = loadImage("noite-estrelada.jpg");
	size(600, 480);
	background(0);
	noStroke();
}

void draw(){
	//image(imagem, 0, 0);
	for(int i = 0; i < 500; i++){
		float x = random(width);
		float y = random(height);
		color c = imagem.get(int(x), int(y));
		fill(c);
		ellipse(x,y,6,6);
	}
}
```

3.

```
PImage imagem;

void setup(){
```

```
        imagem = loadImage("noite-estrelada.jpg");
        size(600, 480);
        background(0);
        noStroke();
}

void draw(){
        float x = mouseX;
        for(int i = 0; i < 100; i++){
                int y = int(random(height));
                color c = imagem.get(int(x), int(y));
                fill(c);
                ellipse(x,y,6,6);
        }
}
```

4.

```
PImage imagem;

void setup(){
        imagem = loadImage("noite-estrelada.jpg");
        size(600, 480);
        background(0);
        noStroke();
}

void draw(){
        for(int i = 0; i < 500; i++){
                float x = random(width);
                float y = random(height);
                color c = imagem.get(int(x), int(y));
                fill(c);
                ellipse(x,y,6,6);
```

```
        }
}
```

5.

```
PImage imagem;

void setup(){
        imagem = loadImage("noite-estrelada.jpg");
        size(600, 480);
        background(0);
        noStroke();
}

void draw(){
        for(int i = 0; i < 500; i++){
                float x = mouseX;
                float y = mouseY;
                color c = imagem.get(int(x), int(y));
                fill(c);
                ellipse(x,y,6,6);
        }
}
```

6.

```
float noiseX, noiseY;

void setup () {
        noiseX=10;
        noiseY=27;
}

void draw () {
```

```
        ellipse(noise(noiseX)*width, noise(noiseY)*height, 1, 1);
        noiseX +=0.006;
        noiseY +=0.008;
}
```

7.

```
float noiseX, noiseY;
PImage imagem;

void setup(){
        imagem = loadImage("noite-estrelada.jpg");
        size(600, 480);
        background(0);
        noStroke();
        image(imagem, 0, 0);
}

void draw () {
        ellipse(noise(noiseX)*width, noise(noiseY)*height, 5, 5);
        noiseX +=0.006;
        noiseY +=0.008;
}
```

8.

```
PImage imagem;
float noiseX, noiseY;

void setup(){
        imagem = loadImage("noite-estrelada.jpg");
        size(600, 480);
        background(0);
        noiseX = 27;
```

```
        noiseY = 73;
        noStroke();
}

void draw(){
        for(int i = 0; i < 100; i++){
                float x = (noise(noiseX)*width);
                float y = (noise(noiseY)*height);
                color c = imagem.get(int(x), int(y));
                fill(c);
                ellipse(x,y, 6, 6);
                noiseX = noiseX + 0.006;
                noiseY = noiseY + 0.002;
        }
}
```