EXPLORANDO A TECNOLOGIA GENERATIVA NO DESIGN GRÁFICO: Do conceito à Prática com P5.js

EXPLORING GENERATIVE TECHNOLOGY IN GRAPHIC DESIGN: From concept to practice with P5.js

LIMA, João; Estudante do Curso Superior de Tecnologia em Design Gráfico; IFPE-Recife

jhcl1@discente.ifpe.edu.br

SILVA, Josinaldo; Doutor em Design gráfico pela UFPE; IFPE-Recife

josinaldobarbosa@recife.ifpe.edu.br

Resumo

O design é um campo interdisciplinar em constante evolução, que se adapta e se transforma com o surgimento de novas ferramentas, métodos e processos. Nesse contexto, o artigo estuda o campo da tecnologia generativa, reunindo conceitos, exemplos, ferramentas e práticas para avaliar seu impacto e contribuição no design gráfico. Para isso, foram conduzidos estudos teóricos e práticos com diversas ferramentas generativas a fim de avaliar suas características e potencial de uso na criação de artefatos gráficos. Entre essas, selecionamos o p5.js para uma análise mais efetiva de seu uso no processo criativo do designer gráfico. Os resultados mostraram que o p5.js possui grande potencial e eficácia para gerar alternativas complexas e dinâmicas, talvez impossíveis com métodos mais tradicionais. No entanto, a ferramenta mostrou-se limitada no que diz respeito à geração de arquivos para impressão e na gestão de blocos de texto, pois não apresenta ferramentas ágeis para lidar com espaçamentos, alinhamento e escolha tipográfica. Apesar dessas limitações, a tecnologia generativa, particularmente o p5.js, oferece um potencial significativo para o processo criativo no design gráfico.

Palavras Chave: design gráfico, tecnologia generativa, p5.js

Abstract

Design is an interdisciplinary field in constant evolution, adapting and transforming with the emergence of new tools, methods, and processes. In this context, the article examines the field of generative technology, gathering concepts, examples, tools, and practices to assess its impact and contribution to graphic design. The study involved both theoretical and practical research with various generative tools to evaluate their characteristics and potential for creating graphic artifacts. Among these, p5.js was selected for a more effective analysis of its use in the graphic designer's creative process. The results showed that p5.js has great potential and effectiveness in generating complex and dynamic alternatives, which might be impossible with more traditional methods. However, the tool proved limited in terms of generating files for print and managing text blocks, as it lacks agile tools for handling spacing, alignment, and typographic choices. Despite these limitations, generative technology, particularly p5.js, offers significant potential for enhancing the creative process in graphic design.

Keywords: *Graphic Design, Generative Technology; P5.js*

1 Introdução

A ascensão do design pós-moderno, nos anos 70, permitiu que artistas e designers tivessem mais liberdade no que concerne à criação de seus projetos. Com o desprendimento dos cânones da boa forma, surgiram novas possibilidades de expressão gráfica, processos e métodos de design que permitiram mais liberdade, exploração da aleatoriedade e, consequentemente, a experimentação (Meggs e Purvis, p. 601, 2009). Junto a isso, a integração do trabalho do designer com a informática abriu um novo leque de possibilidades, facilitando a execução de tarefas complexas e possibilitando outras abordagens na resolução de problemas (Almeida e Maranhão, 2022, p. 86).

Nesse contexto, o experimentalismo se fortaleceu, não apenas na utilização de novos materiais, mas também na adoção de novos conceitos e processos (Silva e Silva, 2011, p. 4). Com isso, surgiram novas formas de fazer design, dentre elas a prática generativa. Esta, por sua vez, de acordo com Naddeo et al. (2014), populariza-se justamente da união das novas possibilidades dentro do design com a integração do trabalho do designer junto ao computador.

Com isso, artistas pioneiros como Herbert Franke e John Whitney usaram a computação, ainda em seus primórdios, para trabalhar com o design generativo, criando elementos gráficos a partir de algoritmos que geram, de forma randômica, animações, padrões geométricos e elementos gráficos em geral (Magalhães, p. 17, 2020). A arte generativa vai muito além do uso de computadores, tendo surgido muito antes dessa tecnologia, como no caso dos trabalhos dadaístas, os quais tinham como foco a randomização de um sistema para criar artes visuais e textuais inesperadas que fugiam do controle dos poetas e artistas gráficos. Autores como Nascimento (2017) e Naddeo et al. (2014) consideram esses exemplos como arte generativa. Os autores ainda afirmam que a arte generativa está intimamente ligada à quebra dos cânones da boa forma, subvertendo alguns processos do design.

A tecnologia generativa é amplamente ligada ao uso de computadores, tendo migrado de um modo manual, ou analógico, para um processo digital, visto que estes facilitam, por meio de algoritmos, a criação de sistemas randômicos com uma vasta gama de variáveis (Magalhães, p. 23, 2020). Hoje em dia, para a criação de sistemas generativos, temos diversas ferramentas. Entre elas, podemos citar: o Processing, que usa o Java, facilita a programação visual e é amplamente usado para criação generativa (Melo, p. 75, 2015); o p5.js, que se originou do Processing e utiliza os mesmos conceitos (Santos Costa e Sigrist, p. 44, 2023); o Cavalry, que também se apresenta como uma ferramenta generativa (Cavalry, 2024); e o NodeBox. Essas últimas permitem que seja feita programação em blocos, não utilizando a programação de códigos diretamente (Sant'Anna, p.8, 2022).

Segundo Fischer e Herr (2001), o design generativo se distancia dos processos tradicionais. Porém, para Nascimento (2017), um sistema generativo também pode ser embasado por regras e variáveis. Além da forte associação com um design computacional, é preciso entender onde o eixo generativo se encontra em termos de processos do design. O design generativo teria uma abordagem mais tradicional, enviesada por métricas (Bomeny e Perrone, p.4, 2004), tal qual a definição de Nascimento (2017), ou fugiria dos processos tradicionais, como dizem Fischer e Herr (2001), indo para um design mais experimental, baseado em novas ferramentas e novos processos criativos (Silva e Silva, p. 5, 2011)? Portanto, entre computadores e experimentalismo, faz-se necessário investigar onde e como o campo generativo se enquadra no design gráfico, e quais são suas possibilidades enquanto uma ferramenta de criação gráfica.

Com isso, nossa pesquisa visou analisar a capacidade da tecnologia generativa e suas

possíveis aplicações na criação de artefatos gráficos, expandindo as possibilidades de processos de criação na área do design. Para isso, estudamos as linguagens e aplicativos utilizados como ferramentas para gerar os artefatos generativos, observando seu potencial, limitações e características. Além disso, observamos os seus recursos no desenvolvimento, manipulação, criação, organização e geração de elementos visuais, com o intuito de contribuir para o design e as artes gráficas, apresentando o potencial e as limitações da tecnologia generativa.

2 Metodologia

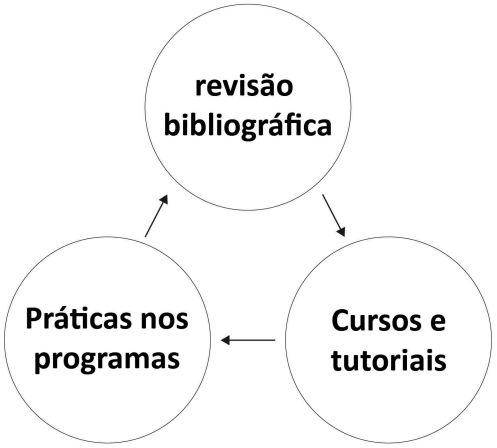
Para um melhor direcionamento para a pesquisa, buscamos, em um primeiro momento, entender melhor o processo de design tradicional e o design experimental, tendo em vista que alguns textos, como veremos adiante, tratam a tecnologia generativa como um processo criativo experimental. A compreensão dos conceitos que envolvem o processo generativo permite identificar onde e como esse campo se apresenta no design gráfico. Esses conceitos serão discutidos com mais profundidade na próxima seção.

Com base na pesquisa realizada sobre tecnologia generativa, foi possível identificar várias ferramentas comuns nessa área, tais como: o Processing, uma linguagem e ambiente de desenvolvimento voltado para a arte visual; p5.js, uma biblioteca JavaScript que faz parte do ecossistema Processing, permitindo criar arte generativa diretamente no navegador (Santos Costa e Sigrist, p. 44, 2023). Estes trabalham diretamente com a programação de códigos. Temos ainda o Cavalry, um software de animação 2D, utilizado para animar formas, personagens, textos e elementos gráficos em geral, sem necessariamente usar a programação de código (Cavalry, 2024), apresentando uma interface “what you see is what you get” (WYSIWYG). Também temos o NodeBox, uma ferramenta de programação em blocos (Sant’Anna, p.8, 2022). Em nossa pesquisa, vamos nos deter a discutir essas ferramentas, mas há outras linguagens e softwares, que não serão aqui abordadas, por serem de uso mais de programadores ou para determinadas finalidades. Dentre esses podemos citar: Python; Three.js, assim como o p5.js, é uma biblioteca JavaScript, sendo mais utilizada para gráficos em 3D; TouchDesigner, uma plataforma visual para arte generativa e design de performances, usada para criar visuais interativos e instalações; e Max/MSP, um ambiente de programação visual para música e arte multimídia interativa.

Cada uma dessas linguagens e ferramentas oferece diferentes capacidades e vantagens, dependendo do tipo de arte generativa que se deseja criar. Seguindo o conceito abordado por Nascimento (2017), a criação de sistemas generativos não precisa envolver necessariamente a utilização direta de programação, mas sim de um sistema autônomo. Nesse sentido, também analisamos softwares como o Cavalry, que tem uma interface baseada em um ambiente de desenvolvimento visual (WYSIWYG) e outros que exigem um conhecimento da linguagem de programação, por trabalharem diretamente com a digitalização dos códigos.

Considerando as ferramentas a serem analisadas, foi criado um fluxo de trabalho (Figura 1), que demonstra como a pesquisa se desenvolveu, começando com revisões bibliográficas, passando pelo estudo dos programas e observação, por meio da prática, de suas vantagens e desvantagens, em conjunto com as questões teóricas, para obter um direcionamento sobre qual software seria escolhido para nossos experimentos quanto ao uso da tecnologia generativa no design gráfico.

Figura 1 - fluxo de trabalho



Fonte: o autor (2024)

Após a escolha do programa a ser utilizado, partimos para a realização de projetos gráficos, explorando dois tipos de práticas: na primeira, trabalhamos com a criação de elementos gráficos individualmente, como texturas, animação de formas e texto; já na segunda, com a criação de artefatos gráficos mais complexos, como cartazes, que exigiam a manipulação e criação de textos e imagens segundo os princípios básicos de diagramação.

3 O Universo Generativo

3.1 O design generativo

O design generativo ganha cada vez mais notoriedade desde a chegada e crescente popularização dos computadores nos anos 90 (Naddeo et al., p. 11, 2014). Artistas pioneiros, como Ben Laposky e John Whitney, aproveitaram o acesso ao computador para trabalhar com o design generativo e criar elementos gráficos, artes abstratas e animações (Magalhães, p. 17, 2020). No Brasil, também com a popularização da computação criativa, artistas como Waldemar Cordeiro e Julio Plaza produziram obras de caráter generativo. Essa forte ligação entre a computação criativa e o universo generativo foi vital para que designers e artistas se aproximassem cada vez mais do campo (Naddeo et al., p. 11, 2014).

Por mais que seja visto apenas como uma expressão visual enviesada por códigos e algoritmos (SILVA e SILVEIRA, p. 7, 2017; Endo, p. 14, 2023), e apesar de existir uma relação muito forte entre o código e o generativo, autores como Nascimento (2017), Boden & Edmonds (2010), Endo (2023) e Naddeo et al. (2014) apontam que o campo generativo vai além do código. Este pode ser melhor definido como a montagem de um sistema autônomo, onde, dada as variáveis, abdica-se do controle dos resultados (Galanter, p. 4, 2003). Segundo o autor, esta é a definição mais citada nos estudos sobre tecnologia generativa (Galanter, p. 151, 2016).

Sendo assim, é possível dizer que o código e o computador se tornam, não o cerne da tecnologia generativa, mas sim, ferramentas facilitadoras para a criação de sistemas generativos (Naddeo et al., p. 16, 2014). Dessa forma, linguagens e programas como Processing e p5.js surgem visando facilitar o uso de código com sintaxes mais simples e funções mais diretas. Nascimento (2017), justificando esse conceito, elenca obras e artistas que, segundo ele, se enquadram como

arte generativa mesmo sem o uso da programação. Como, por exemplo, Jean Hans, fundador do movimento dadaísta, que criou um sistema manual e aleatório que consistia em jogar recortes sob a tela a fim de criar colagens únicas e inesperadas, e que, para Nascimento (2017), se tornaria um sistema de colagens generativo.

O design generativo demonstra uma ruptura dos processos tradicionais do design, como dito por Fischer e Herr (2001). No design tradicional, o designer lida diretamente com o problema e suas particularidades para criar uma solução específica. Em contraste, no design generativo, o designer não aborda o problema diretamente, mas trabalha com variáveis para desenvolver um sistema que o resolva. Esse processo permite a abdicação do controle sobre o resultado final, fazendo com que o design generativo atue como um intermediário entre o designer e o problema proposto. Diante disso, se o design generativo não se encaixa no design tradicional, qual é o seu campo de atuação?

Para essa resposta, faz-se necessário discutirmos o design tradicional ou clássico e o design Experimental. Trazemos, para isso, alguns conceitos, alguns voltados aos estudos tipográficos, mas que podem ser generalizados para o Design Gráfico. O clássico é descrito por Bomeny e Perrone (2004) como regido por métricas, grids e convenções que se moldam conforme a época e o local, definição também trazida por Meggs e Purvis (2009) ao descrever o estilo tipográfico internacional, movimento onde o design foi tratado como algo exclusivamente paramétrico, imposto por grids e malhas que não podiam ser rompidos. Além dos cânones contextuais, podemos aqui trazer Nascimento (2017) que, assim como Fischer e Herr (2001), coloca que o design tradicional tem como característica, aquele onde o designer lida diretamente com o objeto, tomando todas as decisões e respondendo a todos os problemas. Para ele, o design tradicional se caracteriza pelo processo racional na definição de problema, na tomada de decisão e solução.

Com o design e tipografia experimental, autores como David Carson (Carson citado por Bil'ak, 2005), Rocha (2003) e Triggs (2003) trazem definições que consideramos contribuir pouco para o assunto: tratando-o apenas como algo novo, fora do comum, meramente estético ou até mesmo uma simples brincadeira. Essas definições não ajudam no entendimento do que seria o experimentalismo no design, pois tratam o assunto de forma rasa e pouco abrangente. Já para Silva e Silva (2011), o experimental não está relacionado apenas ao resultado, mas se relaciona muito mais ao processo criador, diferenciando-se do método tradicional de projetar. Ele divide o processo experimental em três eixos: Conceitual, onde se experimentam conceitos e os seus limites na percepção gráfica; Material, onde é utilizada a experimentação com materiais não usuais; e Processual, onde as técnicas e ações realizadas são o que importa, abdicando-se do controle do resultado. Na mesma linha, temos Guerrero (2014) e Espinoza (2018), que, ao falarem sobre a tipografia experimental, tratam como algo referente ao processo, e não ao resultado. É esse conceito que assumimos em nosso trabalho, exposto por esses últimos autores, que compreende o processo experimental como um processo criador e não apenas como amadorismo, brincadeira ou relacionado necessariamente a um resultado inovador.

Com isso, é possível observar que tanto as definições propostas por Galanter (2003) para generativo, quanto as definições trazidas por Silva e Silva (2011) para experimental processual, são bastante semelhantes, principalmente no que concerne ao desprendimento quanto ao controle do resultado, tornando o design generativo parte do design experimental processual. Podemos considerar que o design generativo se distingue do design tradicional, já que, por mais que ainda seja enviesado na criação de um sistema de regras e variáveis, o processo de criação é volátil e muitas vezes randômico, permitindo a abdicação do controle sobre os resultados, aproximando-o do design experimental.

3.2 As possibilidades do campo generativo

Entendidos os conceitos de design generativo, é importante delinear sobre as possibilidades de uso do campo generativo na criação gráfica e expressões visuais. Como apresentado por Endo (2023), com um sistema generativo, é possível a geração de uma infinidade de alternativas para artefatos gráficos de maneira prática, pois, como um sistema autônomo, a cada produção há a possibilidade de obtermos um resultado diferente do anterior, gerando uma diversidade de opções.

Com a tecnologia generativa, designers e artistas geram desde animações, como John Whitney, padrões gráficos, como Vladimir Bonacic (Magalhães, p. 17, 2020), até mesmo tipografias, como as criadas por Guerrero (2017), e as mais variadas expressões gráficas explorando elementos como variáveis e ações randômicas. Um exemplo de como a randomização se dá no campo generativo pode ser encontrado em Gross e Bohnacker (2018), onde são apresentadas ao leitor diversas criações generativas, feitas com código utilizando o p5.js. Eles mostram uma série de tipografias randômicas, onde, cada vez que um algarismo é solicitado pelo teclado, o desenho do tipo sofre variações, sendo modificado a cada uso, porém ainda mantendo certas características visuais. Os tipos permanecem seguindo uma mesma lógica visual, mesmo apresentando formas diferentes, lembrando o conceito proposto por Galanter (2003), onde as variáveis e dados para a criação do sistema generativo permanecem os mesmos, logo, uma constância visual é mantida mesmo que o resultado seja aleatório.

Essa forma de gerar elementos gráficos com infinitas possibilidades de variações, porém mantendo um padrão visual, pode se tornar bastante útil, por exemplo, na criação de sistemas de identidade visual. Em Vieira e Bruscato (2023), temos marcas que utilizam a tecnologia generativa que mudam segundo o artefato criado, obtendo novos padrões gráficos com facilidade, porém mantendo-os na identidade visual proposta. Por exemplo, temos o caso da marca de vinhos Brute, que usa um sistema com base nas medições climáticas de onde as vinícolas se localizam, sendo os dados das medições extraídos para criar variáveis para um sistema generativo que gera diferentes padrões a cada garrafa produzida. Outro exemplo é o citado por Blasi (2018), que demonstra uma marca que gera formas geométricas em diferentes tamanhos, rotações e posições, mantendo o padrão cromático da identidade visual estabelecida pela empresa. Nesse exemplo, a cada artefato criado, as formas se diferenciam, podendo gerar crachás com os elementos da marca em disposições diferentes para cada funcionário, mas no mesmo sistema visual.

Além disso, a tecnologia generativa se apresenta como promissora para gerar elementos gráficos e composições também no campo da abstração visual. Este é o caso do artista italiano Julien Gachadoat, que trabalha com arte generativa há cerca de 20 anos e produz uma série de quadros e composições abstratas e geométricas, seguindo padrões visuais matemáticos e randômicos, tanto como peças gráficas individuais, quanto como elementos para compor outros artefatos (Gachadoat, 2024).

Com isso, é possível perceber que a tecnologia generativa apresenta um ótimo potencial dentro do campo gráfico, sendo ela útil para automatizar processos e criar, de forma ágil e eficiente, uma infinidade de opções para artefatos gráficos distintos. Faz-se importante agora delimitar onde e como produzir sistemas generativos, bem como demonstrar, de modo prático, como as criações generativas podem ser utilizadas no campo do design gráfico.

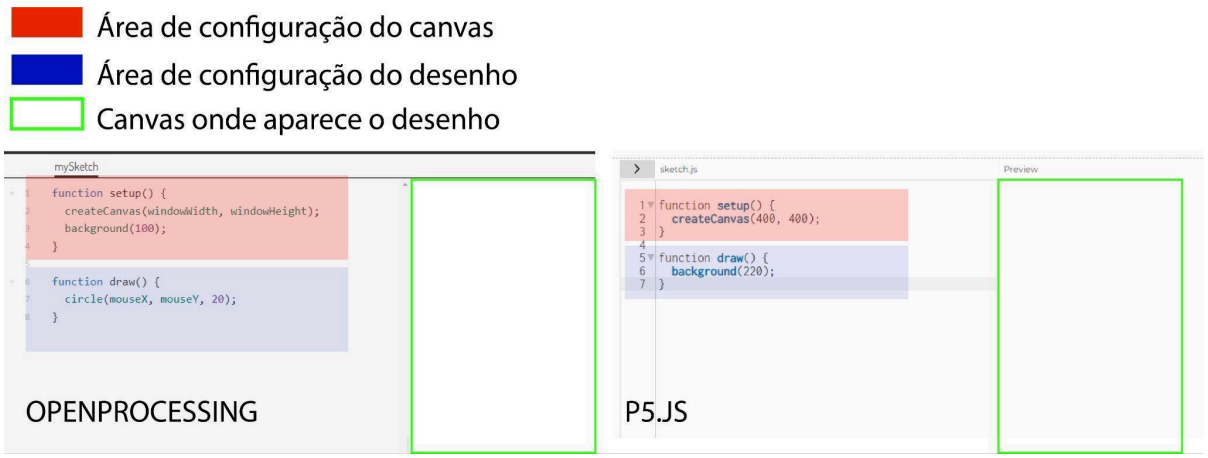
4 Levantamento de programas e ferramentas generativas

4.1 Processing e p5.js

Ferramentas de programação criativa, p5.js e Processing têm suas histórias interligadas: o Processing surge em 2001, para facilitar a integração entre o código e a representação visual. Ele usa a linguagem Java, porém a simplifica. Enquanto no Java, para criar um simples quadrado, se utiliza uma dezena de linhas de código, no Processing, a mesma ação pode ser feita em 2 linhas (Oakim, p.78, 2015). Já o p5.js, surge anos depois, visando ser uma releitura do Processing, porém agora em JavaScript (Santos Costa e Sigrist, p. 44, 2023), linguagem da programação que se integra muito bem com o HTML (Voicu, 2017).

Em questão de interface e logística, ambos apresentam as mesmas características, apenas mudando a linguagem. Até mesmo as interfaces de ambos são muito parecidas, como visto na figura 2. Ambas possuem como princípios básicos uma área do código destinada à configuração do canvas e uma área do código destinada ao desenho e criação das formas, que em sua maioria se dá por coordenadas e valores geométricos. São relativamente fáceis de se usar, a princípio basta apenas entender os conceitos de criação das formas com coordenadas em um plano cartesiano, além disso, contam com extensas bibliotecas de códigos alimentadas constantemente por usuários.

Figura 2 - Comparação entre Processing e p5.js



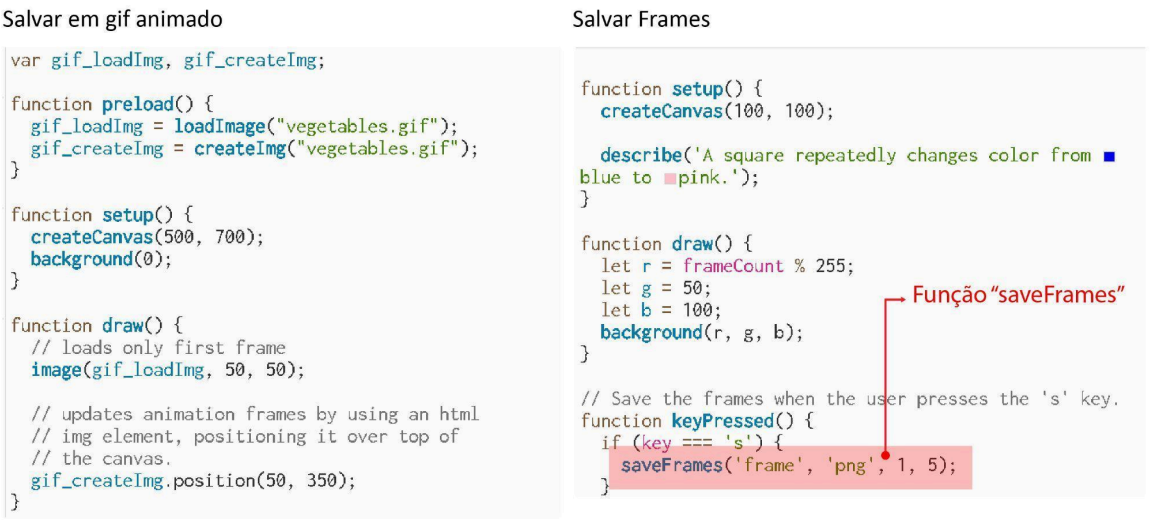
Fonte: o autor(2024)

Quanto às possibilidades de criação nos softwares, podemos dizer que são inúmeras. É possível criar animações, tipografias, objetos 3D e uma série de artefatos gráficos, sendo ainda possível criar sistemas de partículas e clonagens (p5.js, 2024). O p5.js, por exemplo, exporta em .png; .jpg e .svg. No entanto, na análise dos códigos disponíveis para a exportação, a alteração da resolução fica limitada ao tipo de tela que você está usando, ficando dependente do PPI do seu monitor, que, caso seja baixo, irá interferir na exportação de imagens em jpg e png. Essa característica dificulta a impressão das criações , sendo necessário o uso de outros softwares para ajustes e finalização do arquivo. No entanto, isso pode ser parcialmente contornado usando a função pixelDensity, que se propõe a alterar a densidade de pixels por polegada no canvas (p5.js, 2024).

No caso das animações e vídeos feitos com p5.js, é possível incorporar o código diretamente em sites, permitindo interatividade sem necessidade de arquivos de vídeo. Porém,

para exportar em vídeo, as opções são menos abrangentes. Um método é adicionar um código que salva a animação como um GIF animado (Figura 3). Outra opção é a função saveFrames do p5.js, que exporta uma sequência de frames em .png ou .jpg, os quais podem ser compilados em um programa externo para criar a animação (p5.js, 2024). Assim, fica claro que as ferramentas se afastam das possibilidades de impressão, sendo mais adequadas para uso web, devido às limitações de exportação e à facilidade de integração com HTML (Voicu, 2017).

Figura 3 - Maneiras de exportar animações e vídeos no p5.js



Fonte: o autor (2024)

Pelo apresentado, p5.js e Processing mostram ser promissoras ferramentas no campo generativo. Ambos contam com uma linguagem de programação fácil, tornando-a mais atrativas para quem não domina ou tem dificuldade com programação. O p5.js tem maior facilidade para se encontrar referências e bibliotecas de código, contando com uma grande listagem de auxílio em seu próprio site, que contém diversos códigos, funções e propriedades (p5.js, 2024).

4.2 **Cavalry**

Cavalry é um software para criação de composições e motion graphics, podendo ser comparado ao famoso e consolidado nesta área, o After Effects. Segundo a própria equipe do software, o Cavalry tem muitas funções, como design para jogos, publicidade, animação 2D, transmissão de informação e, o que mais nos interessa, arte generativa (Cavalry, 2024).

Diferentemente do p5.js ou do Processing, no Cavalry você não precisará lidar diretamente com código, mas sim alterar parâmetros em um painel, criando formas vetoriais diretamente na tela e aplicando modificadores e conexões, sendo o código mascarado pela interface do programa. Porém, também é possível utilizar JavaScript em um campo de código para a criação, mas em sua forma comum e não a simplificada que Processing e p5.js utilizam (Cavalry, 2024).

A interface do Cavalry, sem o uso direto do código, limita algumas funções quanto a randomização, como comenta Moreira (2018), ao apresentar que, com o uso de algoritmos e programação computacional, a arte pode se tornar randômica de uma forma mais abrangente, gerando diversos resultados inesperados. Quanto a isso, o Cavalry apresenta uma função de

randomização (Cavalry, 2024), porém, pelo menos até onde foi possível observar, essa função pode aleatorizar somente algumas variáveis. Enquanto, com p5.js, que permite lidar diretamente com a programação, e de forma mais simplificada, é possível adicionar funções de aleatoriedade com cálculos matemáticos e variáveis mais complexas (p5.js, 2024). Mesmo assim, o Cavalry compensa esse fato, pela vasta gama de modificadores que oferece, diferenciando-se de um sistema pré-definido. A maior dificuldade para lidar com o programa foi quanto à sua aprendizagem, pois, como é relativamente novo, há pouco material disponível sobre o mesmo.

4.3 **NodeBox**

O NodeBox se apresenta como uma ferramenta de código aberto, que visa facilitar a computação criativa, permitindo que quem use tal ferramenta possa visualizar uma grande quantidade de dados de forma simples, sem necessariamente se perder em extensas linhas de código (NodeBox, 2024). Diferentemente dos outros programas observados, que usam Java ou JavaScript, o NodeBox usa como base a linguagem Python, acrescida de novos recursos (Sant'Anna, p.8, 2022).

Um fator que chamou a atenção para o possível uso do NodeBox, assim como o Cavalry, foi a possibilidade de não se envolver diretamente com a programação. O NodeBox pode ser dito como uma ferramenta da programação blocada, a qual pode ser explicitada como uma forma praticamente gráfica de se programar. Com ela, o código é mascarado por formas visuais (ou blocos) que são ligadas uns aos outros para que suas funções se conectem e funcionem juntas (Lima Sousa, Farias e Carvalho, p. 2, 2020). Esta maneira de programar é útil para quem tem pouco contato com a programação direta. Segundo Kelleher e Pausch (2005), isso torna o ato de programar mais objetivo, fazendo com que o programador não precise decorar uma série de sintaxes, mas apenas aprenda e aplique a lógica por trás das funções.

Porém, assim como o Calvary, há pouco material de estudo sobre o NodeBox. A maioria encontra-se no próprio site do aplicativo, que disponibiliza uma quantidade razoável de tutoriais básicos sobre diversos temas, incluindo, princípios generativos (NodeBox, 2024). Dessa forma, a prática no software ficou bastante limitada aos poucos tutoriais disponíveis. Ainda que haja a vantagem de poder desenhar diretamente no próprio canvas, sem se preocupar com dados matemáticos, algumas funções, como a modificação de tipografias, a criação de sistemas de partículas e a randomização, nos pareceram um tanto mais complicadas, talvez devido ao pouco conhecimento disponível sobre o software. É possível observar que o Nodebox apresenta vantagens quanto ao seu uso e nos parece que é uma forma mais simplificada de programa, apresentando ferramentas para a criação de sistemas generativos, como sistemas de partículas e randomização de variáveis.

5 **Ferramenta escolhida e análise da prática**

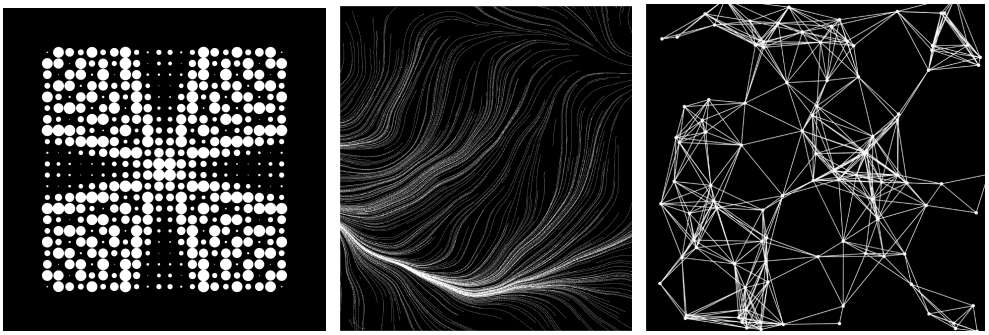
O p5.js foi escolhido para os testes práticos de criação generativa devido aos seus amplos recursos e à facilidade de aprendizagem, apoiada por extensas bibliotecas e uma comunidade ativa. A pesquisa teórica e a análise de materiais gráficos criados com o p5.js mostraram a versatilidade do programa, que permite a criação de animações, tipografias, interfaces, modelos 3D, etc. (p5.js, 2024). Nessa etapa, experimentos práticos foram realizados para explorar como a criação generativa pode ser integrada ao processo criativo do designer, considerando nossas limitações enquanto não-programadores e a pouca experiência com o software. Foram gerados

diversos elementos gráficos, incluindo texturas, ilustrações, tipografias, artefatos como cartazes e posts para redes sociais.

As primeiras criações dos autores foram baseadas em tutoriais e cursos, servindo apenas como exercícios de aprendizagem e mimetismos. Depois, foram analisados projetos generativos, estudando seus códigos, bem como tutoriais, para a realização de interferências objetivas em determinadas programações, gerando novos projetos. Após essa fase, houve dois tipos de prática essenciais para entender o design e a arte generativa no p5.js: a criação de elementos gráficos, como texturas, animações e tipografias; e a integração desses elementos em um único código para gerar artefatos completos, como cartazes e posts para redes sociais.

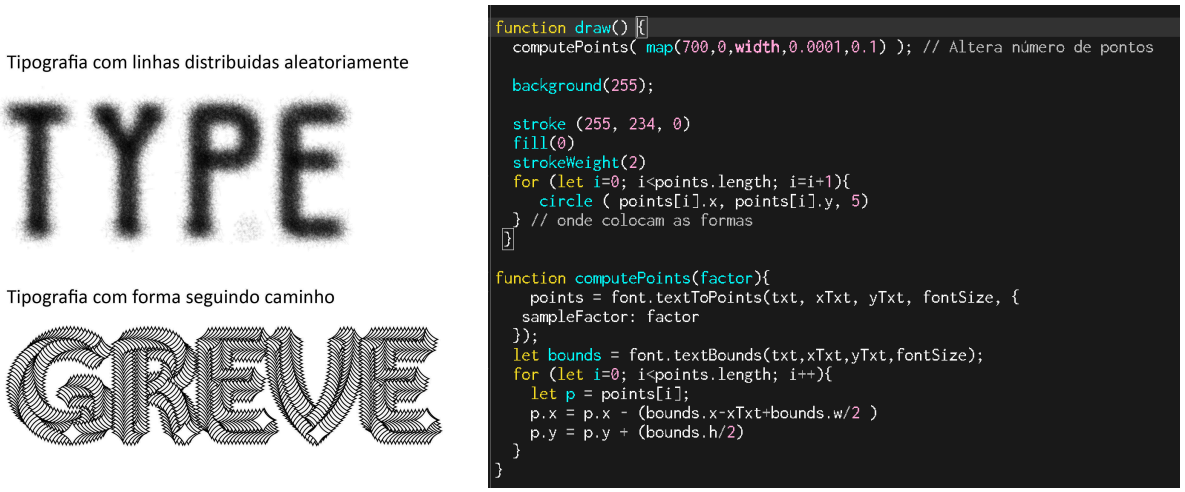
Na primeira prática, não houve muitos impedimentos, sendo desenvolvidos padrões, texturas e desenhos randômicos (Figura 4). Também foram criadas tipografias, tendo como base uma fonte tipográfica já existente, da qual eram extraídos os seus pontos para servirem de coordenadas para a criação de formas diferentes. No exemplo apresentado (Figura 5), temos uma tipografia com linhas distribuídas aleatoriamente e outra com uma forma geométrica distribuída de maneira uniforme. A comunidade do p5.js, juntamente com o uso da inteligência artificial, mostrou-se bastante útil para resolver esse código, pois não tínhamos ideia de como executá-lo.

Figura 4 - Criações no p5.js



Fonte: o autor(2024)

Figura 5 - Tipografias desenvolvidas no p5.js e código base para criação



Fonte: o autor (2024)

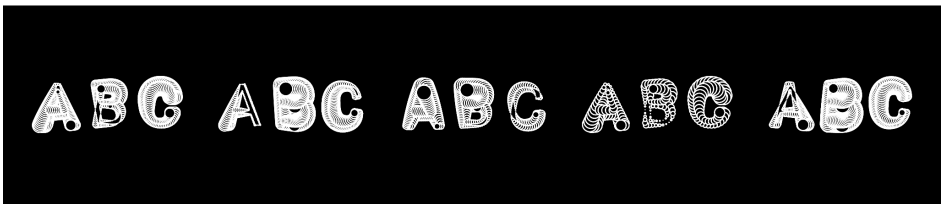
Ainda nas primeiras práticas, descobrimos o potencial do p5.js para criar sistemas generativos, pois oferece funções que facilitam a randomização de praticamente todas as variáveis no código com a função "*random*". Essa característica automatiza facilmente os resultados e, devido à sua aleatoriedade, cria um sistema aonde parte do controle dos resultados é perdida, tal qual a definição de Galanter (2003). Além disso, ele possibilita produzir uma ampla variedade de resultados, como propõe Endo (2023).

Também foram descobertas funções como as *Particle* e *Array,* que possibilitam, por meio de códigos e cálculos, fazer com que os elementos variem inesperadamente e tornem seus resultados aleatórios. O entendimento desses conceitos e funções, atrelados ao código citado anteriormente, que extrai os pontos de tipografias e transforma-os em coordenadas, nos permitiu gerar tipografias modificadas e aleatórias, que variam a cada execução, como mostram as figuras 6 e 7. Além de tipografias, foram gerados outros elementos randômicos, como as ilustrações feitas com Bézier, (Figura 8), que forma um desenho que surge com o passar dos *frames* de maneira aleatória, obtendo assim resultado diferente a cada vez que o código é executado.

Figura 6 - Tipografia randômica que muda ao passar do tempo

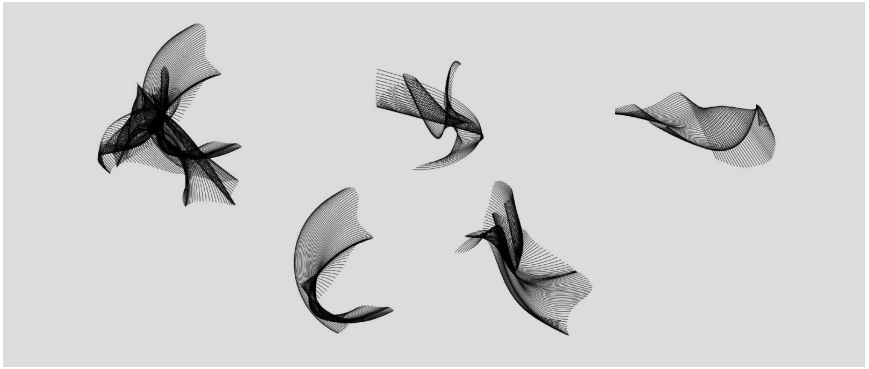
Fonte: o autor(2024)

Figura 7 - Tipografia que se altera segundo as coordenadas do mouse



Fonte: o autor (2024)

Figura 8 - Ilustrações randômicas de formas de Bézier



Fonte: o autor (2024)

Na segunda fase da prática, a criação de artefatos completos no p5.js revelou-se mais desafiadora devido às limitações das ferramentas. Por exemplo, temos que um simples alinhamento de elementos exige uma série de cálculos para determinar corretamente as suas coordenadas. Isso ocorre porque o eixo de alinhamento dos objetos está localizado no seu centro, exigindo uma série de cálculos e um pouco de sorte para o seu posicionamento correto. Isso é especialmente problemático com textos, cujas larguras variam e não podem ser facilmente medidas, dificultando o alinhamento de palavras ou blocos de texto, especialmente para aqueles com menos experiência em programação.

Ainda em relação à diagramação, há outros entraves: como exemplos, temos o fato de simplesmente não haver uma função específica e direta para modificar os espaços entre letras. Esses problemas ocorrem na manipulação de textos simples ou mais extensos. Outro ponto é o fato de ser desgastante trabalhar com diversas fontes tipográficas, por ser necessário fazer o upload de cada arquivo da fonte. Isso torna o processo cansativo e gera um código extenso. Ou seja, apesar de possível, o trabalho com diagramação de texto no p5.js é complexo e cansativo, pois suas ferramentas não são direcionadas aos ajustes da microtipografia nem tão pouco correspondem a exigência do designer enquanto diagramador.

Além de problemas na diagramação e no refinamento do texto, há outros fatores que tornam pouco atrativa a criação de artefatos complexos diretamente no p5.js. Primeiro, a pequena tela de visualização do p5.js requer barras de rolagem para *canvas* maiores que 500×500 pixels, o que é trabalhoso e não permite uma visualização completa do projeto. Outro ponto é a complexidade e extensão do código na criação de cartazes. Temos, por exemplo, os cartazes da figura 9, criados pelos autores totalmente dentro do p5.js. Com base nessas experiências, observamos que o desenvolvimento desses cartazes, ambos construídos por meio de códigos, demandou um tempo consideravelmente mais longo do que o necessário se desenvolvidos por meio de um software de diagramação. E, mesmo sendo cartazes com pouca informação, foi necessário mais de 200 linhas de código para sua construção. Diante do exposto, podemos dizer que a criação de artefatos inteiramente feitos no p5.js não se mostra muito atrativa.

Figura 9 - Cartazes criados e diagramados com p5.js

Fonte: O autor (2024)

Outra forma avaliada para o uso do p5.js no design gráfico, mais especificamente na

diagramação de cartazes, foi gerar elementos gráficos separadamente, depois exportá-los em SVG, e usá-los em programas que permitissem maior facilidade na diagramação e composição. Foram geradas texturas, títulos e ilustrações, que foram importadas e trabalhadas em outro software, resultando nos cartazes da Figura 10. Esse método mostrou-se mais eficaz, por permitir uma melhor disposição dos elementos, ganho de tempo e melhores opções para exportação do tipo de arquivo final.

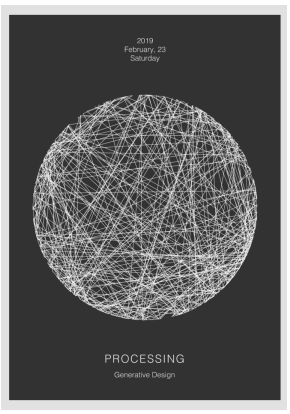
É importante ressaltar que os autores não têm muita experiência com programação, o que pode ter amplificado as dificuldades de criar artefatos completos e complexos no p5.js, uma questão que pode não se aplicar a programadores mais experientes.

Figura 10 - Cartazes Feitos em p5.js e software externo

Fonte: o autor(2024)

Baseadas em nossas experiências, o p5.js mostrou-se uma ferramenta poderosa para trabalhar com programação criativa e sistemas generativos. No entanto, o ideal seria trabalhar o p5.js juntamente com outros softwares que permitam maior liberdade na diagramação. Inclusive, essa abordagem é utilizada pelo artista Oleg Frolov, que cria elementos no p5.js e os finaliza em outro software, como ilustrado na Figura 11.

Figura 11 - cartaz Oleg Frolov



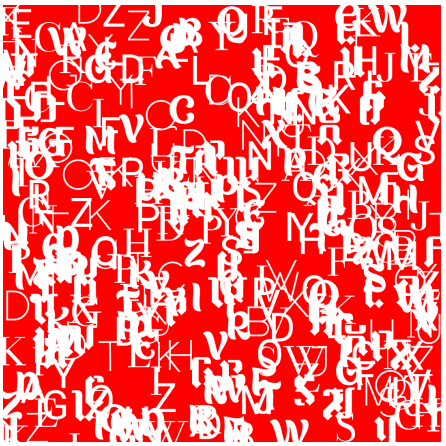
Fonte: https://www.behance.net/Volorf (2019)
(autorizado pelo artista)

6 Considerações finais

Esta pesquisa teve como objetivo analisar e entender a tecnologia generativa no campo do design gráfico, buscando compreender suas possibilidades e como ela pode contribuir para a área. Como vimos, ela não é uma ferramenta nova; sua popularização está relacionada ao próprio desenvolvimento dos computadores em meados dos anos 60. Além disso, encontramos trabalhos na arte e no design, datados de muito antes, que utilizam os mesmos mecanismos do experimentalismo generativo, focados nos processos e que apresentam resultados inesperados e diversos. O que ela traz para o cenário é um imenso potencial criador que possibilita a geração de imagens complexas com relativa facilidade, interação com obras de forma mais efetiva e surpreendente, fácil geração de alternativas e a integração cada vez mais propícia do uso de algoritmos na geração de imagens.

A tecnologia generativa demonstra ser um campo interessante para designers ao trazer formas eficientes e até não convencionais para soluções de velhos e novos problemas. Pode-se dizer que o campo generativo é vasto e proporciona tanto praticidade quanto inovação ao processo criativo do designer. Por exemplo, apesar da criação com código parecer difícil, ao adquirir certo conhecimento, é possível automatizar tarefas complexas. Observando a figura 12, que apresenta uma composição com diversas fontes de tamanhos variados e dispostas de forma aleatória, vemos um projeto que seria desgastante se realizado manualmente, mas, com o uso da tecnologia generativa, foi feita em segundos apenas com algumas linhas de código. Podemos, também, explorar características randômicas e variáveis, gerando infinitas possibilidades de resolução para um determinado problema ou até mesmo fazendo uso dessas características para criar sistemas de identidade abertos e randômicos.

Figura 12 - composição com tipos

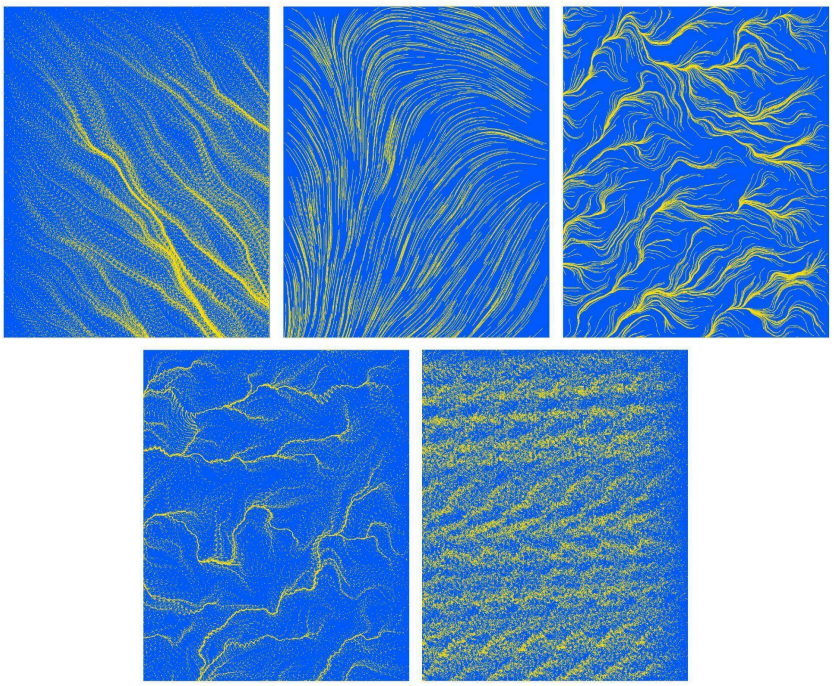


Fonte: o autor (2024)

É possível afirmar que as criações generativas trazem ao design gráfico novas possibilidades e permitem impulsionar a originalidade nos projetos. A tecnologia facilita a geração de imagens, texturas e efeitos que são ou parecem inviáveis de serem produzidos manualmente, ou nem sempre de fácil criação nos softwares que usam *plugins* para gerar efeitos. No sistema generativo, mudando apenas algumas variáveis, é possível obter uma série de recursos gráficos diferentes, como nas texturas mostradas na figura 13, que foram criadas com o mesmo sistema generativo,

apenas alterando alguns valores.

Figura 13 - Diferentes texturas com mesmo sistema



Fonte: o autor (2024)

A tecnologia generativa se apresenta e se firma como uma excelente ferramenta para o design, acrescentando agilidade e potencial de possibilidades formais criativas às já comumente utilizadas. Pode ir desde uma criação tipográfica, uma simples peça gráfica até a criação de identidades visuais inteiras, como já relatado por Vieira e Bruscato (2023) e por Nascimento (2017). Porém, como dito por Omine (2014), infelizmente, designers acabam por limitar-se a utilizar apenas programas e técnicas mais convencionais, tornando as possibilidades de criação limitadas àqueles programas e processos, perdendo então oportunidades de inovar e experimentar novas formas de criar, como o uso da tecnologia generativa, por exemplo.

Podemos dizer que se é necessária uma maior difusão das ferramentas generativas, seus conceitos e processos como práticas reais no aprendizado e trabalho do designer. Vimos que a tecnologia generativa traz vantagens para o processo criativo, potencializando suas capacidades, gerando uma vasta gama de possibilidades e uma forma diferente do designer encarar o problema. As novas ferramentas, com o Cavalry e o NodeBox, estão, de certa forma, tornando o ambiente de desenvolvimento por códigos mais amigáveis a um público não acostumado a essas linguagens. Além disso, como vimos, mesmo as ferramentas mais direcionadas ao uso dos códigos oferecem uma gama de bibliotecas e tutoriais que possibilitam desenvolver projetos sem necessariamente precisar ter domínio das linguagens de programação.

Neste artigo, vimos o potencial, as características e as limitações da tecnologia generativa, principalmente através do p5.js. O programa foi analisado no desenvolvimento de artefatos generativos, e chegamos à conclusão de que, por limitações na exportação e diagramação, o p5.js se torna melhor utilizado em conjunto com outros softwares e processos, devido ao seu viés

voltado para a web. Salientamos que o estudo foi produzido por designers que não tinham experiência prévia em linguagens de programação, mostrando assim que a ferramenta é acessível a todos que se disponham a dedicar um pouco de tempo e dedicação para o seu conhecimento. A ferramenta generativa se demonstrou promissora, tornando a integração dessa prática com os processos existentes e os novos, como a Inteligência Artificial, bastante benéfica à atuação do designer. Portanto, não podemos ficar passivos e restritos a métodos e processos tradicionais, mas devemos ampliar o campo interdisciplinar, não só da atuação, mas também da criação.

7 Referências

BIL'AK, P. **Experimental Typography: Whatever that means.** [S.l.]. 2005. Disponível em: <https://www.typotheque.com/articles/experimental_typography_whatever_that_means.>

BLASI, Lorenzo. **Identità generativa - Nuovo metodo di progettazione visiva**. Orientado por: pier Luigi Capucci. 2018. Trabalho de Conclusão de Curso em Design Gráfico – LABA, Libera Accademia Di Belle Arti, Rimini, 2018.

BOMENY, Maria Helena Werneck; PERRONE, Rafael Antonio Cunha. **Os Manuais de Desenho da Escrita**. São Paulo, Ateliê Editorial, 2004.

CAVALRY. **Cavalry.** Disponível em: https://cavalry.scenegroup.co. Acesso em: 19 jun. 2024

SILVA, Michelly Pessoa da; SILVA, Josinaldo Barbosa da. **GERAÇÃO DE TIPOGRAFIAS ATRAVÉS DE PROCESSOS EXPERIMENTAIS**. IFPE-Recife, 2011. Disponível em: <https://www.academia.edu/1704488/GERAÇÃO_DE_TIPOGRAFIAS_ATRAVÉS_DE_PROCESSOS_EXPERIMENTAIS>

LIMA SOUSA, Lucas de; FARIAS, Eder Jacques; CARVALHO, Windson Viana de. **Programação em blocos aplicada no ensino do pensamento computacional: um mapeamento sistemático.** In: Anais do XXXI Simpósio Brasileiro de Informática na Educação. SBC, 2020. p. 1513-1522.

SANTOS COSTA, Daniele dos; SIGRIST, Vanina Carrara. **WEB ARTE:: a biblioteca p5. js e o estudo de caso do projeto cápsulas sonoras**. Revista Extensão, v. 23, n. 1, p. 41-50, 2023. Disponível em: <https://periodicos.ufrb.edu.br/index.php/revistaextensao/article/view/3031> Acesso em 20 de abril 2024.

ENDO, Gabriel. **Design generativo para sistemas visuais: uma abordagem sistematizada e sistêmica.** 2023, UNB. Disponível em <https://bdm.unb.br/handle/10483/36991>

ESPINOZA, José Luis. **Estudio iconográfico de las manifestaciones textiles de la Parroquia Cacha para la creación de una tipografía experimental.** 2018. Riobamba. Disponível em: <http://dspace.unach.edu.ec/handle/51000/4767>

FROLOV, Oleg. **Processing Posters**. In: Behance, 2019. Disponível em: https://www.behance.net/Volorf. Acesso em: 15 abr. 2024.

GACHADOAT, Julien. **Generative Artis**. V3GA, s.d. Disponível em: <https://www.v3ga.net>. Acesso em: 29 jul. 2024.

GALANTER, Philip. Generative Art Theory. In: PAUL, Christiane (Ed.). A Companion to Digital Art. [S.l.]: Wiley Blackwell, 2016. Disponível em: https://doi.org/10.1002/9781118475249.ch5. Acesso em: 29 jul. 2024

GALANTER, Philip. What Is Generative Art? Complexity Theory as a Context for Art Theory. 2003.

Disponível em: https://philipgalanter.com/downloads/ga2003_what_is_genart.pdf. Acesso em: 29 janeiro de. 2024.

GROSS, Benedikt; BOHNACKER, Hartmut. **Generative Design: Visualize, Program, and Create with JavaScript in p5.js**. New York: Princeton Architectural Press, 2018.

GUERRERO, Manolo. **Type code: uma introdução à tipografia generativa.** In: O Tipo da Fonte, 2017. Disponível em: <https://otipodafonte.com.br/2017/04/type-code-uma-introducao-tipografia-generativa/. Acesso em: 15 abr. 2024.>

GUERRERO, Manolo. **Diseño tipográfico experimental**. Recuperado el, v. 6, 2014. Disponível em: <https://www.academia.edu/download/48800462/Diseno_tipografico.pdf>

FISCHER, Thomas; HERR, Christiane.. **Teaching Generative Design**. International Conference on Generative Art, jan. 2001.

KELLEHER, Caitlin; PAUSCH, Randy. **Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers**. ACM computing surveys (CSUR), v. 37, n. 2, p. 83-137, 2005.

MAGALHÃES, Miguel. **Design generativo: experiência sinestésica na música eletrónica**. ESAD, 2020. Disponível em: http://hdl.handle.net/10400.26/33234

MEGGS, Philip B.; PURVIS, Alston W. **História do design gráfico**. Cosac Naify, 2009.

NADDEO, Diogo Navarro Loureiro de Barros et al. **Arte Generativa: Uma Análise Conceitual, Processual e Referencial.** Vol 1, UFRB, 2014.

NASCIMENTO, Ruben David Ferreira Amaral do. **Tipografia experimental em sistemas generativos**. Design Gráfico - Instituto de Arte, Design e Empresa - Universitário, 2017.

NODEBOX. **NodeBox**. Disponível em: http://nodebox.net. Acesso em: 19 jun. 2024.

MELLO, Patricia Oakim Bandeira de. **Arte e programação na linguagem Processing**. 2015. 137 f. Dissertação (Mestrado em Tecnologias da Inteligência e Design Digital) – Programa de Estudos Pós-Graduados em Tecnologias da Inteligência e Design Digital, Pontifícia Universidade Católica de São Paulo, São Paulo, 2015.

OMINE, Eduardo Hiroshi. **Design gráfico computacional: computação aplicada no projeto e na produção de imagens dinâmicas e interativas**. 2014. Dissertação (Mestrado em Design e Arquitetura) - Faculdade de Arquitetura e Urbanismo, Universidade de São Paulo, São Paulo, 2014. doi:10.11606/D.16.2014.tde-12092014-122450. Acesso em: 2024-07-29

OPENPROCESSING. **OpenProcessing**. Disponível em: https://www.openprocessing.org. Acesso em: 15 mar. 2024.

P5.JS. **P5.js**. Disponível em: https://p5js.org. Acesso em: 19 jun. 2024.

ROCHA, Claudio. **Projeto tipográfico: análise e produção de fontes digitais**. Rosari, 2003.

SANT'ANNA, Hugo Cristo. **Entre foguetes, estrelas e canetas: relato de experiência de ensino de princípios da Computação para estudantes de graduação em Design**. Projetica, [S. l.], v. 13, n. 3, p. 166–183, 2022. DOI: 10.5433/2236-2207.2022v13n3p166. Disponível em: https://ojs.uel.br/revistas/uel/index.php/projetica/article/view/46940. Acesso em: 29 jul. 2024

SILVA, Caio Pereira; SILVEIRA, Lucas Corá. **Sound system: sistema tipográfico generativo baseado em algoritmos musicais**. 2017. 70 f., il. Trabalho de Conclusão de Curso (Bacharelado em Desenho

Industrial)—Universidade de Brasília, Brasília, 2017.

TRIGGS, Teal et al. **The typographic experiment**: radical innovation in contemporary type design. Thames & Hudson, 2003.

VIEIRA, Bruna Luz; BRUSCATO, Léia Miotto. **Parâmetros para a criação de sistemas generativos de identidade visual.** Estudos em Design, Rio de Janeiro, v. 31, n. 3, 2023. Disponível em: https://estudosemdesign.emnuvens.com.br/design/article/view/1816. Acesso em: 29 maio de 2024

VOICU, D. **Which to Choose: Processing or p5.js**. 2017. Disponível em: <https://medium.com/tab-space/which-to-choose-processing-or-p5js-7cbba6fb6e11>. Acesso em: 9 mar. 2024.