Andréa Pennino Graciano e Luisa Paraguai*

# Algoritmo e tipografia: o código como parte do processo de criação de uma fonte digital

* **Andréa Pennino Graciano** é graduada em Engenharia Civil pela Unicamp e em Design Gráfico pela Universidade Anhembi Morumbi, pós graduada com MBA em Tecnologia da Informação na USP/Fia e com DESS em Gestão de Projetos pela Universidade de Grenoble (França) e é mestranda em design na Universidade Anhembi Morumbi.
*<andrea.graciano@me.com>*

**Luisa Paraguai** é pesquisadora e docente permanente do Programa de Pós-Graduação em Linguagens, Midia e Arte da Pontifícia Universidade Católica de Campinas. Consultora *Ad Hoc* da CAPES e FAPESP. Colaboradora da *Leonardo Digital Review*. Membro do Grupo de Pesquisa Produção e Pesquisa em Arte. Artista nas interlocuções entre arte, design e tecnologia. Investiga interfaces multisensoriais e espacialidades urbanas. Possui graduação em Engenharia Civil pela Universidade de São Paulo USP, mestrado e doutorado em Multimeios pela Universidade Estadual de Campinas UNICAMP, e pós-doutorado no *Planetary Collegium, Nuova Accademia di Belle Arti NABA*, Milão.
*<luisa.donati@puc-campinas.edu.br>*

**Resumo** O objetivo deste artigo é descrever o processo de criação da fonte digital TEP (Tipografia Experimental Parametrizada), cujos desenhos foram produzidos com o auxílio de algoritmos, ou seja, códigos de programação. Os processos de parametrização e de seleção randômica norteiam as soluções possíveis na formação dos caracteres, que se organizam esteticamente sem o foco da legibilidade, apesar de ser possível a identificação das letras e dos números nos mesmos. O projeto TEP organiza os tipos como módulos visuais, que repetidos e manipulados por outros algoritmos geram padrões e/ou texturas. Importa salientar a experimentação prática como ação metodológica, que incorpora o inesperado formulado pela lógica computacional.

**Palavras chave:** Algoritmo, Tipografia, Processo de Criação, Parametrização.

## Algorithm and typography: the code as part of the creation process of a digital font

**Abstract** *The purpose of this article is to describe the creative process of the digital font TEP (translation from Portuguese's Parameterized Experimental Typography), whose drawings were produced with the aid of algorithms, ie, programming codes. Once written - obeying some parameters and allowing the random selection of others by the computer – the algorithms, at every mouse click, presented a new solution for the characters' shapes. This is not a typography intended to be used in texts, since the project does not focus on readability, although the identification of letters and numbers is possible. The TEP was designed so that its characters work as visual modules which, when repeated and manipulated by other algorithms, generate patterns. As methodology, added to theoretical research, the experimentation experienced in the computer lab was critical to this work.*

**Keywords** *Algorithm, Typography, Creation Process, Parameterization.*

## Introdução

O objetivo do presente artigo é descrever o processo de criação da fonte TEP (Tipografia Experimental Parametrizada) produzida em paralelo com as pesquisas referencias realizadas nas primeiras fases de um projeto[1] cujo foco era explorar a utilização de elementos visuais, sobretudo geométricos, na construção de estruturas formais mais complexas. Cita-se Moles (apud Munari, 1998, p.38), para compreender que o contexto complexo se apresenta quando "contém um grande número de elementos reagrupáveis mas em poucas classes". Para o autor, bem como para Alexander (1964), o exercício projetual implica em decompor variáveis que articulem vínculos – dependências e independências – para a geração de entidades mais complexas. Assume-se no projeto TEP, formalmente esta proposição, pois os elementos visuais básicos (círculos) foram combinados de diversas maneiras – pela repetição e variação de sobreposições, escalas, rotações, reflexos especulares, translações e acumulados, conforme escreve Munari (2006), em camadas para gerar possibilidades de composições visuais ou padronagens[2]. Para o autor a simetria define modos de acumular as formas básicas pela repetição até constituir a forma global.

Partindo-se da compreensão que repetir, permutar e sobrepor são operações realizadas com eficiência pelo computador, passou-se a investigar a possibilidade de usar algoritmos, ou seja, códigos de programação para a manipulação dos módulos resultantes e a produção de estruturas visuais mais complexas. Segundo Reas *et al.* (2010, p.13), "este tipo de código – frequentemente chamado um algoritmo, procedimento ou programa – define um processo específico com instruções suficientemente detalhadas para serem executadas". Os algoritmos são escritos em uma linguagem de programação, e atuam como exercícios de abstração para mediar diálogos entre o programador/usuário e a máquina. Conforme os autores, para uma dada hipótese definem-se variáveis e estruturam-se escolhas em operações modulares (comportamentos) que atendem específicas funcionalidades, considerando que existem diferentes percursos possíveis. Os códigos deste projeto foram escritos em Java – linguagem de programação utilizada pelo programa *Processing*, diante das seguintes propriedades: trata-se de um *software livre*; possui uma grande base de dados on-line[3], com referências, exemplos, tutoriais, fóruns de discussão, etc., alimentada por uma extensa comunidade virtual; e atua "como ferramenta para não programadores, através da satisfação imediata proporcionada pelo retorno visual" (*PROCESSING*, 2004).

O projeto TEP propõe a construção de módulos visuais, manipulados por algoritmos, para produção de tipos digitais, que não focam a legibilidade, mas a expressividade visual. Os caracteres da TEP, em um momento posterior, podem configurar-se como padrões visuais para a elaboração de texturas e/ou superfícies, a partir de outros códigos de programação. O set tipográfico desenvolvido (conjunto total de caracteres da TEP) conta com 120 desenhos assim distribuídos: 26 maiúsculas (caixa-alta), 26 minúsculas (caixa-baixa), 10 números, 26 diacríticos (letras com acentuação) e 32 sinais.

**1** O projeto em questão refere-se ao trabalho de conclusão do curso de Design Gráfico da Universidade Anhembi Morumbi, apresentado em junho de 2015.

**2** "Uma padronagem pode ser produzida manualmente, por máquinas ou códigos, mas é sempre resultado de uma repetição. Um exército de pontos pode ser regulado por um grid geométrico rígido ou agrupado ao acaso ao longo de uma superfície" (LUPTON e PHILLIPS, 2008, p.187).

**3** www.*processing*.org

## Das métricas aos parâmetros

### Métricas do projeto

Num primeiro momento pensou-se em criar uma fonte objetivamente ornamental[4] ou *dingbat*[5] pelo fato de não se saber se o controle do designer sobre os resultados visuais criados pelos algoritmos seria suficiente para garantir a legibilidade à tipografia. Outro motivo para essa escolha foi a intenção de utilizar seus caracteres como motivos visuais que dariam origem a padronagens em uma fase posterior do projeto.

Entretanto, para ficar de acordo com as definições impostas ao projeto, a TEP deveria, necessariamente, guardar uma relação com as métricas e, minimamente, com as formas dos caracteres da "fonte base" – uma fonte legível e apropriada para leitura de textos. Como "fonte base" escolheu-se a Prometo Trial Regular[6], uma fonte *display*, sem serifas, criada em 2014, pelo Studio Dalton Maag (Figura 1).


**Figura 1** Tipografia Prometo Trial Regular, usada como "fonte base" do projeto.
**Fonte** www.daltonmaag.com/library/prometo

As métricas da "fonte base" de interesse para a criação da TEP foram: a altura e a largura de seus caracteres, tomando-se como referência desenhos proporcionais a uma "altura de x" de 50mm. Observou-se também o número de vezes (interações) que o desenho do caractere toca ou cruza as linhas principais (ascendente, versal, altura de x, linha de base e descendente) da estrutura da fonte, conforme demonstrado na Figura 2.

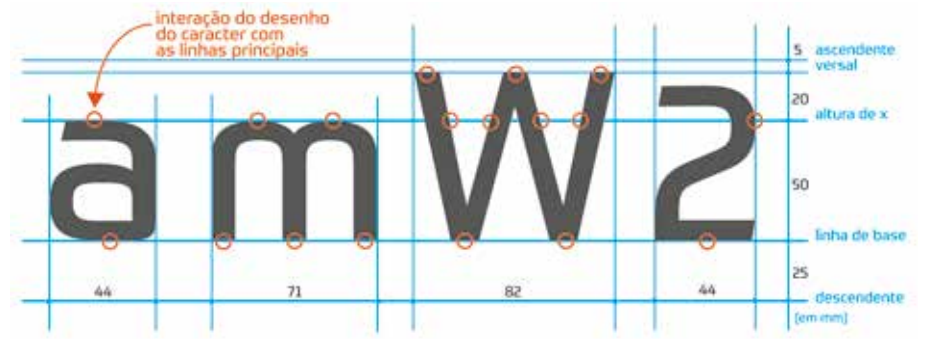


**Figura 2** Métricas da "fonte base".
**Fonte** GRACIANO, 2015, p.22

**4** Os ornamentos servem para "compor padrões gráficos (*patterns*), molduras (borders) e também podem ser utilizados isoladamente como vinhetas. Cumprem a função de estruturar e valorizar o texto" (ROCHA, 2012, p.100). São encontrados nos sets tipográficos de fontes dingbats e até mesmo em fontes de texto.

**5** Segundo Clair (2009, p. 360), dingbats são "marcas decorativas, sinais de impressão ou símbolos fornecidos da mesma maneira que uma fonte tipográfica." Portanto, dingbats não são exatamente uma fonte tipográfica mas são distribuídos de forma idêntica, ou seja, em arquivos digitais no formato de fontes.

**6** Tipografia criada pelos designers: Fernando Caro, Ron Carpenter, Fabio Haag, Bruno Mello e Rafael Saraiva na filial brasileira do Studio Dalton Maag. A Prometo Trial Regular utilizada no projeto é uma versão de avaliação, disponibilizada em http://www.daltonmaag.com/library/prometo.

O esquema apresentado possibilitou a montagem da Tabela 1, que indica o procedimento executado para os 120 caracteres da "fonte base" necessários para a criação do set tipográfico da TEP. De posse de todas essas informações, o passo seguinte foi transformar essas métricas da "fonte base" em parâmetros para o desenho da TEP.

| CARACTERE | A | m | W | 2 |
|---|---|---|---|---|
| Altura | 50 | 50 | 70 | 65 |
| Largura | 44 | 71 | 82 | 44 |
| Interações | 2 | 5 | 9 | 2 |

**Tabela 1** Exemplo de métricas usadas no projeto (em mm).
**Fonte** GRACIANO, 2015, p.22

## Parâmetros

Enquanto a "fonte base", a Prometo Trial Regular, tem como estrutura para o desenho de seus elementos um *grid* com contornos definido e rígido, a TEP por sua vez, busca o oposto, um *grid* flexível e orgânico, construído a partir do conceito do projeto no qual módulos simples dão origem às composições visuais complexas. O *grid* escolhido como base para os desenhos da TEP referencia-se na "superposição de estruturas de repetição" sugerida por Wong (2010, p. 66) e apresentada na Figura 3.

> Uma estrutura de repetição, juntamente com as unidades de forma que comporta, pode ser superposta a uma outra estrutura de repetição. As duas estruturas e suas unidades de forma podem ser as mesmas ou diferentes uma das outras. A interação de duas estruturas pode produzir resultados inesperados.

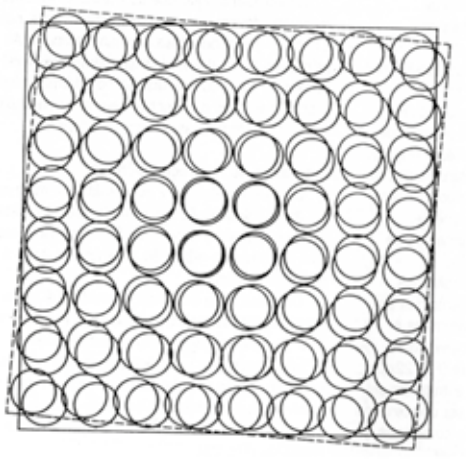

**Figura 3** *Grids* superpostos.
**Fonte** WONG, 2010, p. 64

O *grid* da TEP é composto por elementos circulares – sem preenchimento interno e somente com as bordas visíveis (*outline*) – de tamanhos variados e superpostos aleatoriamente em uma área delimitada. Os centros das circunferências são posicionados dentro da área "retângulo limite", formada pela altura e largura do caracter correspondente da "fonte base" (Figura 4).

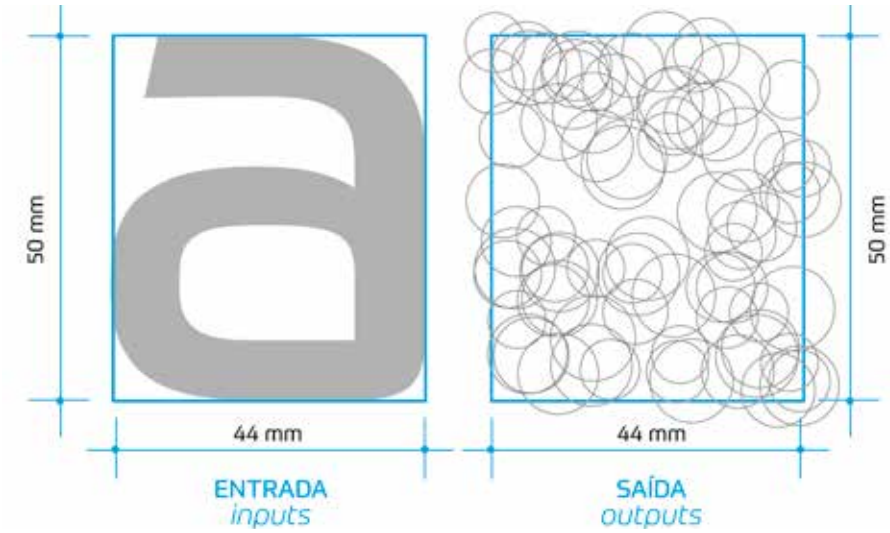


**Figura 4** Estrutura dos caracteres da TEP criada a partir dos parâmetros de entrada.
**Fonte** GRACIANO, 2015, p.23

Nesse contexto, a altura e a largura do caracter da "fonte base" são inputs para o algoritmo, ou seja, parâmetros de entrada para o sistema, enquanto o "retângulo limite" é um dos *outputs* (resultado visual) do programa. Segundo Reas et al. (2010, p. 95) "parâmetro é um valor que produz um efeito no resultado de saída de um processo", podendo ser "constantes", valores que não podem ser alterados e "variáveis", que permitem alterações. Por vezes, os valores associados aos parâmetros variáveis podem ser randômicos (aleatórios). Para os autores, o processo de parametrização define articulações entre a intenção do designer e o sistema visual projetado, na medida em que identifica e descreve os efeitos dos parâmetros na forma, suas possíveis variações e como os dados de input podem influenciar os comportamentos do mesmo. A Tabela 2 mostra a relação entre os inputs e *outputs* do algoritmo responsável pela criação da TEP.

| *INPUT* | *OUTPUT* |
|---|---|
| Caractere | nome do arquivo |
| Altura e Largura | limite externo do caractere (retângulo) |
| Interações | num. de módulos (círculos) do grid |

**Tabela 2** Relação entre os parâmetros de entrada (*inputs*) e de saída (*outputs*) do algoritmo.
**Fonte** GRACIANO, 2015, p.22

A parametrização do projeto e a codificação dessas relações, na forma de regras que constituem o algoritmo constituem o processo de criação do designer. Esse apresenta-se cíclico, ou seja, o designer define um conjunto de parâmetros e regras, executa o algoritmo e analisa os resultados visuais obtidos. A macroestrutura do ambiente projetual com feed-backs permite alterar as regras e/ou os parâmetros de entrada ao buscar formas visuais resultantes que não constituam redundância quando observados os caracteres da fonte base. Neste projeto, após vários testes, chegou-se ao seguinte conjunto de regras que estão codificadas sob a forma de algoritmo:

- A largura e a altura do caracter da "fonte base" são utilizadas para desenhar o "retângulo limite" da TEP;
- A quantidade de módulos que irão formar o *grid* está entre 16 e 18 vezes o número de interações, definida por uma escolha randômica;
- As posições de todos os centros das circunferências são escolhidas de forma randômica mas, necessariamente, devem estar dentro do "retângulo limite";
- Ao se clicar o botão esquerdo do mouse, o programa é reprocessado e nova opção de imagem surgirá na tela; e
- Ao se clicar o botão direito, antes de rodar o programa e apresentar novo resultado, a imagem que estiver sendo exibida na tela será salva em um arquivo (.pdf) cujo nome tem relação com o caracter que se está trabalhando.

Até este ponto do trabalho, os resultados visuais obtidos a partir da execução do algoritmo eram os *grids* formado pelas circunferências e somente isso. A definição da forma final do caracter a partir dessas estruturas ficaram a princípio por conta do designer, que preenchendo a mão algumas circunferências ia criando relações e definindo associações próximas ao contorno do caracter da "fonte base" (Figura 5).

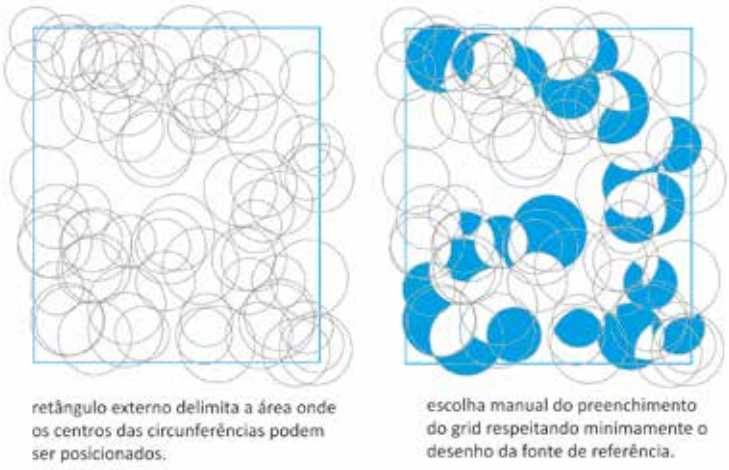


**Figura 5** Preenchimento manual do *grid*.

**Fonte** GRACIANO, 2015, p.23

## Meta-criação

Preencher o *grid* (baseado em círculos) a mão para desenhar os caracteres de modo que ficassem reconhecíveis, provou que utilizar o algoritmo na criação da estrutura da fonte funcionou. Mas essa operação era bastante trabalhosa e o teste de várias possibilidades para a posterior seleção de um desenho para o caracter seria muito demorado.

Entretanto, o computador poderia fazer mais pelo projeto. O código poderia ser ampliado para preencher alguns dos círculos automaticamente. Mas não seriam quaisquer círculos e sim os círculos certos, ou melhor, os que possibilitassem a identificação dos caracteres. Para que isso acontecesse de forma automática adotou-se a seguinte estratégia: em primeiro lugar, as circunferências "vazadas" deveriam ser substituídas por círculos, sem linhas de contorno, preenchidos pela cor branca (para representar o fundo, ou seja, onde não fosse caracter) e outros preenchidos por uma outra cor qualquer, por exemplo o azul (Figura 6). Os círculos coloridos sobrepostos e unificados vão dar origem à forma do caracter.


**Figura 6** Preenchimento automático do grid.
**Fonte** GRACIANO, 2015, p.24

A outra parte da estratégia consistia em decidir quais círculos deveriam ser brancos e quais deveriam ser coloridos. Para solucionar este problema, colocou-se no canto superior esquerdo da tela de saída do *Processing* o caracter da "fonte base" num tamanho menor, embora proporcional, ao desenho da TEP a ser construído (Figura 6).

Antes de inserir os círculos no interior do retângulo que representa os limites do caracter, como explicado no item anterior, era verificado se o ponto correspondente ao centro do círculo, na imagem de base era branco ou colorido. Se fosse branco significava que o centro do círculo se posicionaria fora do caracter - ou em alguma parte interna deste que não tivesse preenchimento, como por exemplo, no miolo[7] da letra "a" – portanto, necessariamente o círculo deveria ser branco. Agora, caso o ponto correspondente fosse colorido, o que indicaria se tratar de uma posição

7 Segundo Rocha (2005, p.39) "miolo ou counter é o espaço interno de algumas letras do alfabeto. Pode ser fechado, como nas letras a, d, o e b, ou aberto, como nas letras h, n, u, e e".

sobre o desenho do caracter, o círculo a ser colocado deveria ser colorido ou branco, numa escolha randômica feita pelo computador numa proporção de 1:4, na grande maioria dos casos, seguindo as regras do algoritmo. O fato de permitir que a cor de preenchimento do círculo fosse escolhida aleatoriamente, multiplicou o número de resultados possíveis uma vez que inseriu mais uma variável ao processo. "Quanto mais parâmetros forem adicionados ao processo, mais opções de saída serão possíveis" (REAS et al., 2010, p.95). Percebe-se um movimento de confronto no processo de criação: o designer deixa de realizar escolhas plásticas baseadas na visualidade da forma, para articular e propor os domínios de atuação das variáveis – o campo do possível. Neste sentido, o projeto "pode atuar como linguagem de propósito, sobrepondo-se ao mesmo tempo à representação e à mediação. Ele não serve nem para revelar nem para esconder o significado, mas em última instância para (re)inventá-lo" (BECCARI, 2013).

Projetado dessa maneira, o algoritmo permitia que a cada "clique" do mouse, ou seja, cada vez que o programa fosse executado, uma nova solução de desenho do caracter fosse apresentada na tela. No código foi inserida, também, a opção de salvar as soluções que mais agradassem num arquivo de formato PDF. A seleção do desenho final do caracter que faria parte da fonte digital foi feita com base nessas imagens (em pdf) obtidas durante a execução do algoritmo. A Figura 7 mostra as várias opções pré-selecionadas para a letra "g" minúscula, sendo que apenas a indicada integrou o alfabeto da TEP. O processo repetiu-se para a definição de todos os caracteres do set tipográfico.

Neste ponto do projeto, a execução do algoritmo e a seleção das formas não precisava mais, necessariamente, ficar a cargo do designer criador do código, podendo ser realizada por outra pessoa (outro designer ou até mesmo um cliente, se fosse esse o caso). De acordo com essa perspectiva, o algoritmo funciona como meta-criação, gerando produtos finais distintos – todos oriundos da mesma formulação idealizada pelo designer criador do código. Pode-se compreender estas articulações projetuais entre requisitos e a forma a partir de Alexander (1964, p.90-92) ao definir "*pattern*" enquanto um "diagram construtivo" – uma entidade abstrata que se constitui enquanto estrutura e formulação. Corroborando esta afirmação, Vassão (2010, p.64) escreve que o foco do designer deve estar no processo e não somente no produto, quando se exploram as múltiplas variantes de um mesmo "meta-objeto", para alterar suas condicionantes pré-programadas. O projeto TEP visa a criação do programa, que contém virtualmente todas as possibilidades de configurações visuais. Conforme Paraguai (2014, p.2227) afirma "exercita-se na prática e significação da linguagem o reconhecimento de *pattern*s e o potencial articulador dos mesmos como dinâmica de invenção de objetos visuais".

O usuário, por assim dizer, não precisa ter nenhum conhecimento sobre técnicas de programação ou sobre as estratégias adotadas pelo designer criador para construir a sua tipografia. Retoma-se o conceito de "caixa-preta" (FLUSSER, 1985), na medida em que o usuário ignora o que acontece especificamente no interior do algoritmo - enquanto se vale de suas

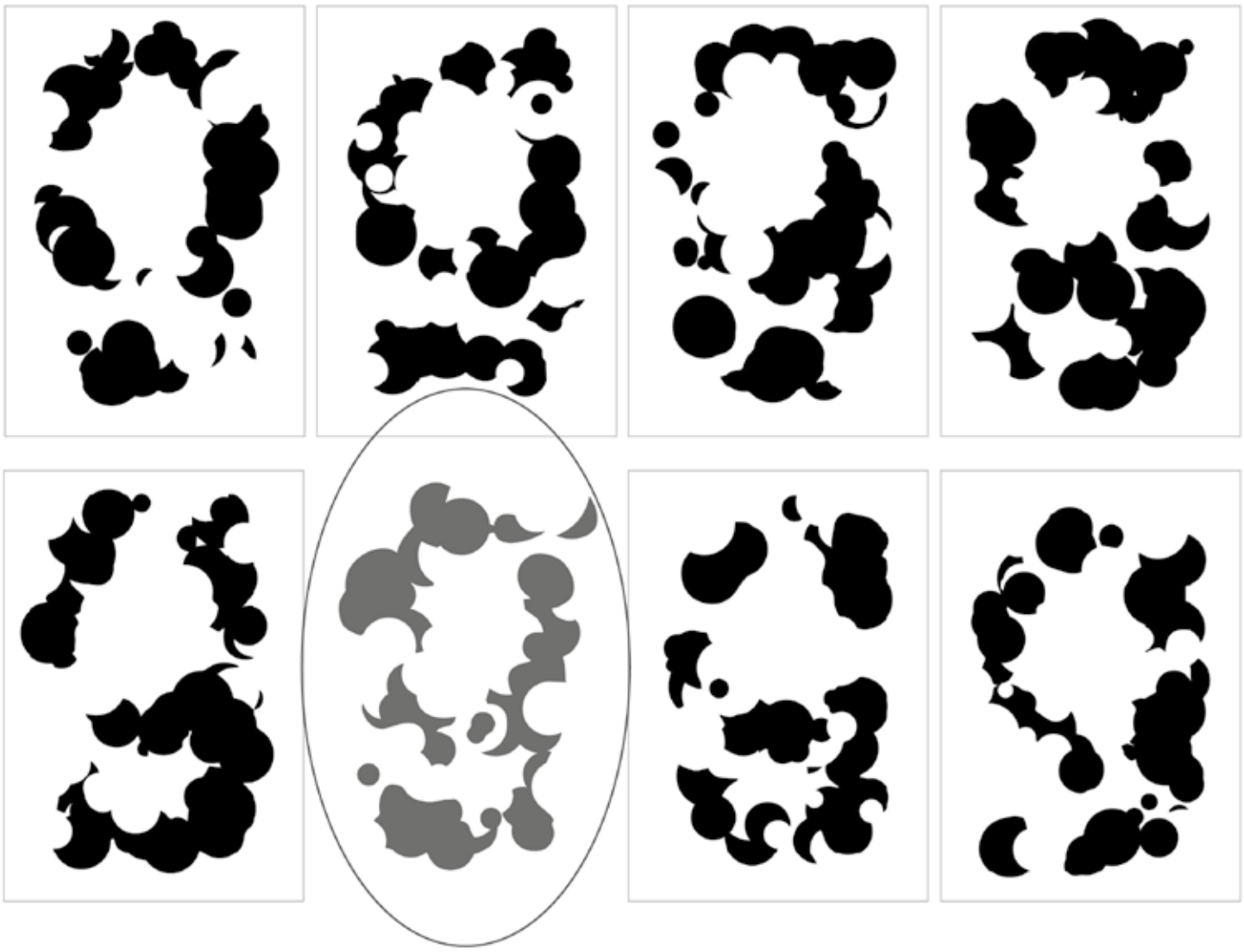

**Figura 7** Seleção da letra "g " dentre as várias opção geradas pelo algoritmo.
**Fonte** GRACIANO, 2015, p.25

escolhas e intenção para estruturar formas visuais a partir dos inputs. Ou seja, uma vez que se conheça os inputs (o caracter, a largura, altura e o que se denominou como sendo as interações desse caracter) ele terá acesso aos outputs (desenhos) gerados por esse programa.

Note que, o objetivo deste projeto não contemplava a criação de um algoritmo para ser usado por outras pessoas. Mas se assim o fosse, uma tabela que relacionasse todos os caracteres com sua altura, largura e número de interações com as linhas principais, semelhante a Tabela 1 mostrada anteriormente, poderia ter sido incluída dentro do código, reduzindo o input para apenas uma variável, o caracter. A criação de uma inteface para a utilização do *Processing* reduziria, por sua vez, a complexidade inerente ao processo.

## Criando e usando a fonte digital

### TEP.otf

Para criar uma fonte digital utilizam-se frequentemente programas como o *FontLab Studio* e o *Glyphs*. Neste projeto, o *Glyphs Mini*, apesar de um número reduzido de funções em comparação com o programa *Glyphs*, atendeu perfeitamente às demandas do mesmo. Existem três formatos para arquivos de fonte: o *PostScript*, o *TrueType* e o *OpenType*[8], sendo este último o formato escolhido para a fonte TEP. Todo *setting* tem seu corpo constituído de armações e elementos que organizam a sua arquitetura, sendo formado por caixa ou gaveta de tipos – o recipiente maior com subdivisões associadas a uma tecla ou uma combinação de teclas do computador – que atua como espaço e índice de alocação dos caracteres: neste projeto os desenhos selecionados no processo anterior com o *Processing*. Depois de inseridas as 120 imagens correspondentes ao set tipográfico, gerou-se o arquivo TEP.otf. Na Figura 8 são apresentadas as letras minúsculas que fazem parte dessa tipografia.

**8** Open type é um formato de fonte universal, desenvolvido em conjunto pela Adobe e pela Microsoft, para impressão e visualização em tela. Uma fonte Opentype pode incluir mais de 65 mil glifos (diferentes desenhos de caracteres, sinais, pontuação, ornamentos e dingbats)" (ROCHA, 2005, p.23).

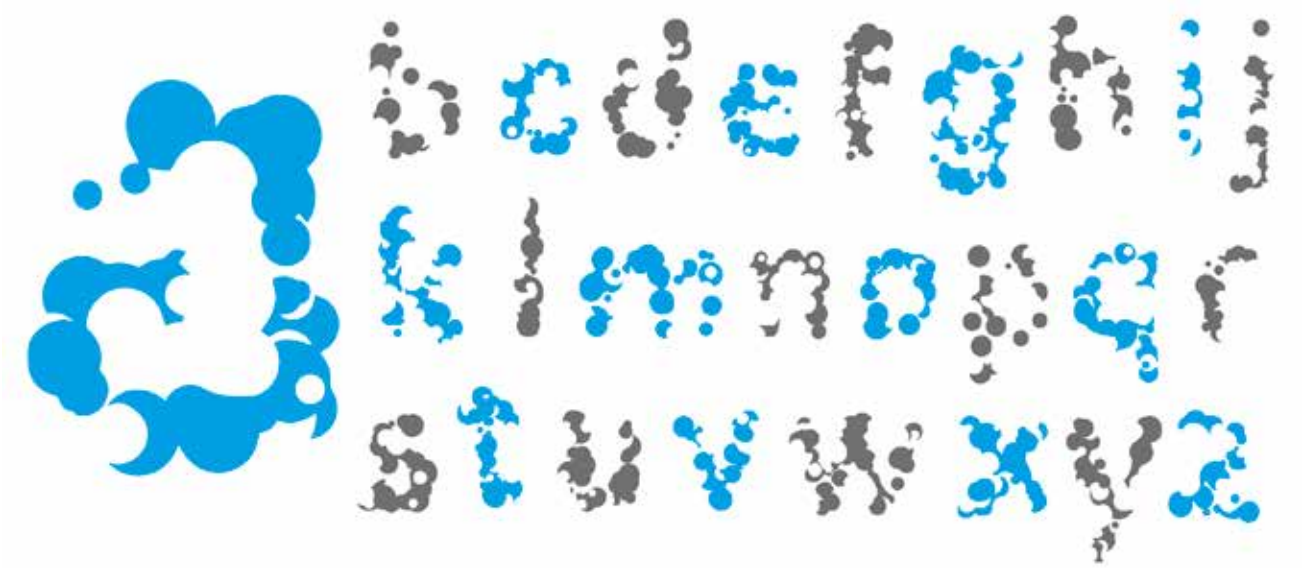

**Figura 8** Tipografia TEP – minúsculas (caixa-baixa).
**Fonte** GRACIANO, 2015, p.27

### Tipografia e padronagem

Uma vez instalada no computador, a TEP.otf passou a se comportar como qualquer outra fonte digital, podendo ser utilizada por diversos *softwares* – desde o Bloco de Notas ou *Excel* até um *Illustrator* ou *Photoshop*, por exemplo –, inclusive pelo próprio *Processing* novamente, só que agora inserida em outros algoritmos, escritos com o propósito de gerar padronagens, por meio da repetição, manipulação e acumulação, conforme Munari (2006) e Wong (2010), dos caracteres da TEP, que passam a ser módulos de composições visuais - padronagens, conforme exemplos nas figuras 9 e 10.

**Figura 8** Exemplo de padronagem com o "X" da TEP.

**Fonte** GRACIANO, 2015, p.27

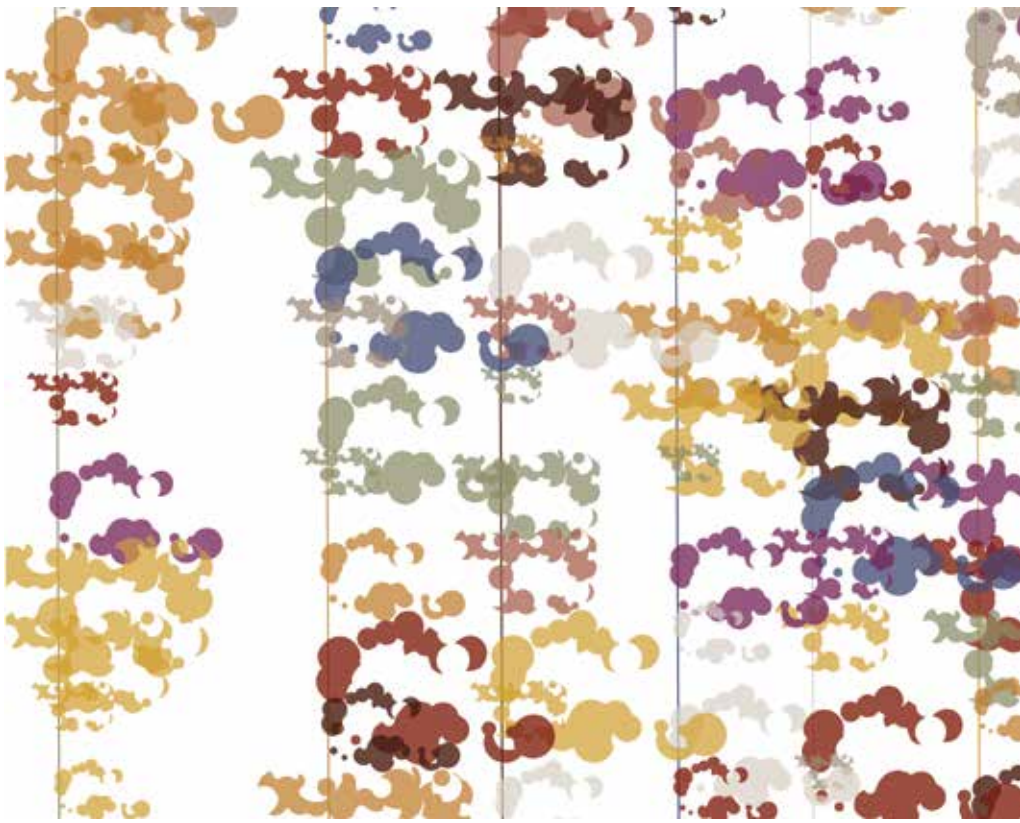

**Figura10** Exemplo de padronagem com os caracteres d" e "p" da TEP.

**Fonte** GRACIANO, 2015, p.29

Para compreensão destas composições visuais, retoma-se Alexander (1964) e o conceito de *"standard"*, que homologa propriedades abstratas e formas de uso do módulo, formulados no *"pattern"*. Vassão (2008, p.267), referenciando o autor, explica que o "standard opera uma apropriação e transpõe o *pattern* como um stencil, carimbando-o de local em local".

## Considerações finais

Os resultados obtidos com este projeto vão muito além da tipografia criada. Foi a experiência prática que possibilitou o aprendizado, consolidações e a aplicação dos conceitos teóricos e, sobretudo, a experimentação do processo de criação levando em conta as particularidades do uso dos algoritmos. Considerando as pouquíssimas funções empregadas do código *Processing*, frente aos inúmeros recursos e possibilidades do programa - a simplicidade do algoritmo gerador da fonte TEP foi suficiente para evidenciar contribuições sobre a aleatoriedade (procedimentos randômicos) na definição de parâmetros, multiplicando o campo de soluções possíveis.

Partindo dessa perspectiva, observa-se que contextos de meta-criação implicam em operações de abstração, mais do que projetar soluções visando resultados aplicados. O exercício da formulação contribui decisivamente para a criação de um modelo complexo, no qual "os níveis de abstração são representações de um sistema em que cada nível de abstração encompassa os detalhes de níveis inferiores, e os oculta" (VASSÃO, 2008, p.135). Como afirma Paraguai (2014, p.2228), "as práticas como formulações pragmáticas que hibridizam tecnologias e linguagens, [...] definem uma estrutura imanente da relação entre indivíduo e ambiente, conectando modos de viver, e assim, potencializam experiências".

As funções de: operação de programa; seleção dos resultados; e até mesmo algum grau de personalização do produto de design – resultado final desse processo – não precisam, necessariamente, ficar a cargo do designer criador podendo ser executadas por outros atores, como outros designers e clientes.

## Referências

ALEXANDER, Christopher. **Notes on the synthesis of form**. Cambridge, Massachusetts; London, England: Harvard University Press, 1964.

BECCARI, Marcos. **Dilemas do design VII**: metalinguagem. 2013. Disponível em <http://filosofiadodesign.com/dilemas-do-design-vii-metalinguagem/>. Acessado em agosto/2016.

FLUSSER, Vilem. **Filosofia da caixa preta**: ensaios para uma future filosofia da fotografia. São Paulo: Annablume Editora, 2011.

GRACIANO, Andrea Pennino. **Projeto Trama.Tipos**: composições visuais com tipografias ornamentais (Trabalho de Conclusão de Curso). Caderno: Set Tipográfico. São Paulo, SP: Universidade Anhembi Morumbi, 2015.

LUPTON, Ellen; PHILLIPS, Jennifer Cole. **Novos fundamentos do design**. São Paulo: Cosac Naify, 2008.

MUNARI, Bruno. **Das coisas nascem coisas**. São Paulo: Martins Fontes, 1998.

_____. **Design e Comunicação Visual**. Lisboa, Portugal: Edições 70 Lda, 2006.

PARAGUAI, Luisa. **Mapeamento da mobilidade**: traços de percursos. In Anais do XXIII Encontro Nacional de Pesquisadores em Artes Plásticas. Afonso Mederiso, Lucia Gouvêa Pimentel, Idanise Hamoy, Yacy-Ara Froner (orgs.). Belo Horizonte: ANPAP; Programa de Pós-Graduação em Artes, UFMG, 2014.

*PROCESSING*. ***Processing***, 2004. Disponível em: http://www.*processing*.org. Acessado em jun. de 2016.

REAS, Casey; MCWILLIANS, Chandler; LUST. **Form + Code**: In design, art and architecture, New York: Princeton Architectural Press, 2010.

ROCHA, Claudio. **Projeto tipografico**: Análise e Produção de Fontes, São Paulo: Editora Rosari, 2005.

_____, **Novo projeto tipográfico**, São Paulo: Editora Rosari, 2012.

VASSÃO, Caio Adorno. **Arquitetura livre**: complexidade, metadesign e ciência nômade. (Tese de Doutorado em Arquitetura). São Paulo, SP: FAUUSP, 2008.

_____, **Metadesign**: Ferramentas, estratégias e ética para a complexidade. São Paulo: Blucher, 2010.

WONG, Wucius. **Princípios de forma e desenho**. São Paulo: Editora WMF Martins Fontes, 2010.

**Recebido:** 29 de Junho de 2016

**Aprovado:** 30 de Agosto de 2016