Eduardo Omine

Design gráfico computacional

Computação aplicada no projeto e na produção
de imagens dinâmicas e interativas

São Paulo / 2014

Eduardo Omine

Design gráfico computacional:
computação aplicada no projeto e na
produção de imagens dinâmicas e
interativas

Dissertação apresentada à Faculdade de Arquitetura e Urbanismo da Universidade de São Paulo, para a obtenção de Título de Mestre em Arquitetura e Urbanismo.

Área de Concentração: Design e Arquitetura

Orientadora: Daniela Kutschat Hanns

São Paulo

2014

Autorizo a reprodução e divulgação total ou parcial deste trabalho, por qualquer meio convencional ou eletrônico, para fins de estudo e pesquisa, desde que citada a fonte.

Email do autor: eduardo.omine@gmail.com

Exemplar revisado e alterado em relação à versão original, sob responsabilidade do autor e anuência do orientador. O original se encontra disponível na sede do programa.

São Paulo, 26 de junho de 2014

O55d Omine, Eduardo.
Design gráfico computacional: computação aplicada no projeto e na produção de imagens dinâmicas e interativas / Eduardo Omine – São Paulo, 2014
148 p. : il.

Dissertação (Mestrado – Área de Concentração: Design e Arquitetura) – Fauusp.

Orientadora: Daniela Kutschat Hanns

1. Design gráfico 2. Computação aplicada 3. Design computacional 4. Arte generativa I. Título

CDU 76

Agradecimentos

À professora Daniela Kutschat Hanns, pela orientação.

Aos entrevistados Hugo Cristo Sant’anna, Gustavo Ferreira, e Dimitre Lima, pelos valiosos depoimentos.

À professora Patrícia Moran e ao professor Carlos Zibel, pelas contribuições no exame de qualificação.

Ao professor Vicente Gil, pelas importantes contribuições ao longo de minha trajetória acadêmica, na graduação e no mestrado.

À professora Priscila Farias, pelas contribuições ao longo do mestrado.

Aos professores Ricardo Nakamura e Gil Garcia de Barros, com quem tive contato durante o estágio PAE.

Aos colegas de pós-graduação da FAU, em especial a Jair Alves, Leandro Velloso, Luis Carli, Roberto Temin, Rui Alão, e Tatiana Machado.

À minha família, aos meus amigos, e à Kátia.

Resumo

OMINE, E. **Design gráfico computacional: computação aplicada no projeto e na produção de imagens dinâmicas e interativas**. Dissertação de mestrado. Faculdade de Arquitetura e Urbanismo, Universidade de São Paulo, São Paulo, 2014.

Esta pesquisa objetiva investigar o design computacional, explorando a aplicação de ferramentas conceituais da computação no projeto e na produção de imagens dinâmicas e interativas. A realização desta pesquisa foi motivada por observações feitas pelo autor ao longo de sua experiência profissional, projetando e desenvolvendo interfaces digitais. Constatou-se que muitos profissionais, intitulados “designers”, possuem pouco ou nenhum conhecimento das tecnologias envolvidas na implementação das interfaces que projetam. Por outro lado, esses mesmos profissionais se limitam a usar alguns poucos softwares comerciais (como por exemplo o Adobe Photoshop) enquanto ferramentas computacionais. Se o designer não conhece as tecnologias envolvidas no projeto (com suas possibilidades e suas limitações), pode acabar propondo soluções tecnicamente inviáveis ou inadequadas. E se o designer limita seu repertório de ferramentas, suas soluções podem ser concebidas mais em função dos recursos disponíveis nas ferramentas do que em função dos requisitos do projeto. Considerando que as ferramentas computacionais ocupam atualmente um papel central na produção, distribuição, e recepção de conteúdo, torna-se importante desenvolver uma postura crítica a respeito do uso da computação em design. E para tanto, faz-se necessário pesquisar as relações entre computação e design. Consultando autores das áreas de comunicação, cibernética, computação, artes, e design, buscou-se identificar os efeitos da computação na sociedade e no pensamento, sua aplicabilidade em outras áreas do conhecimento, e precedentes da arte e do design para uma abordagem computacional do design. Também foram consultados textos (em forma de depoimentos) de designers que trabalham com computação, nos quais relatam seus próprios processos de design. Além disso, foram realizadas entrevistas com 3 designers brasileiros, que ofereceram contribuições valiosas para aqueles que buscam compreender o design computacional de uma perspectiva brasileira. Esta dissertação pretende contribuir para uma maior compreensão das possibilidades e limitações da computação no design.

Palavras-chave: design, computação, design computacional, arte generativa

Abstract

OMINE, E. **Design gráfico computacional: computação aplicada no projeto e na produção de imagens dinâmicas e interativas.** Master's dissertation. Faculdade de Arquitetura e Urbanismo, Universidade de São Paulo, São Paulo, 2014.

This research aims to study computational design, exploring the application of computing's conceptual tools in the design and the production of dynamic and interactive images. This research was motivated by reflections made by the author throughout his professional experience, designing and developing digital interfaces. It was found that many professionals, entitled "designers", have little or no knowledge of the technologies involved in the implementation of the interfaces they design. On the other hand, these same professionals limit themselves to use a few commercial softwares (for example, Adobe Photoshop) as computational tools. If the designer doesn't know the technologies involved in the design (with its possibilities and its limitations), he may end up proposing technically unfeasible or inappropriate solutions. If the designer limits his own tool set, his solutions may be conceived more as a function of the features available in the tools than the project's requirements. Considering that computational tools currently occupy a fundamental role in production, distribution, and reception of content, it becomes important to develop a critical stance regarding the use of computing in design. Therefore, it becomes necessary to study the relationships between computing and design. Consulting authors in the fields of communication, cybernetics, computing, arts, and design, we sought to identify the effects of computing in society and thinking, its applicability in other fields, and precedents from art and from design for a computational approach to design. Statements and articles from designers who work with computing, describing their own design processes, were also consulted. In addition, 3 interviews with Brazilian designers were conducted; the interviewees offered rich contributions for those who seek to understand computational design from a Brazilian standpoint. This dissertation aims to contribute to a larger comprehension of the possibilities and limitations of computing in design.

Keywords: design, computation, computational design, generative art

Lista de Figuras

1	untitled246, 2009.	14
2	untitled246, 2009.	14
3	Golden Ages, 2011.	15
4	Golden Ages, 2011.	16
1.1	Alan Kay. Dynabook, 1972.	28
2.1	Sol LeWitt. Wall drawing #122: All combinations of two lines crossing, placed at random using arcs from corners and sides, straight, not straight and broken lines, 1972.	66
2.2	Esquema para realização de Wall Drawing #122, 1972.	67
2.3	Wall Drawing #122 na Paula Cooper Gallery, 2011.	68
2.4	Wall Drawing #122 na Paula Cooper Gallery, 2011.	68
2.5	Terry Riley. Partitura de In C, 1964.	70
2.6	Terry Riley. Instruções para a execução de In C (página 1), 1964.	71
2.7	Terry Riley. Instruções para a execução de In C (página 2), 1964.	72
2.8	Manfred Mohr. Inversion logique, 1970.	75
2.9	Frieder Nake. Series 201, 1966.	81
2.10	Karl Gerstner. Programa da Idade Média.	82
3.1	Waldemar Cordeiro. Gente - Grau 3 (1972) na capa da revista Computer Graphics and Art (v.1, n.3, ago. 1976).	93
3.2	A. Michael Noll. Computer composition with lines, 1964.	95
3.3	John Maeda. Pôster comemorativo do aniversário de 30 anos de comerciais da marca japonesa de cosméticos Shiseido, 1995.	99

3.4	Processing.	100
3.5	Letterror. Espaço das ideias versus espaço das ferramentas.	105
3.6	Letterror. Exemplos de aplicação da fonte tipográfica FF Beowolf.	107
3.7	Ed Burton. Sodaconstructor, 2000.	109
3.8	Jared Tarbell. Substrate, 2003.	111
3.9	Golan Levin. Meshy, 2006.	114
3.10	Scott Snibbe. Rework, 2012.	116
3.11	Hugo Cristo Sant'anna. Derridaw, 2012.	121
3.12	Gustavo Ferreira. Tuplefont, 2011.	126
3.13	Dimitre Lima. Vinheta MTV Brasil, 2009.	129

Sumário

Apresentação	13
Introdução	17
1 Computação como mídia e artefato mental	21
2 Design como programa	61
3 Práticas programáticas	91
Considerações finais	135
Referências bibliográficas	139

Apresentação

Para contextualizar a pesquisa realizada, considero importante apresentar brevemente minha trajetória acadêmica e profissional. Cultivo grande interesse pelo desenho e pela matemática desde a infância. Devido à influência familiar, tive os primeiros contatos com a programação de computadores durante o ginásio. Ingressei na graduação em 1998, no curso de Arquitetura e Urbanismo da Fauusp. Concentrei os estudos nas disciplinas de Programação Visual e de Desenho Industrial. Desenvolvi o trabalho final de graduação, que consistiu no desenvolvimento de uma família tipográfica sem serifa, em 2002 sob orientação do Professor Doutor Vicente Gil. Em 2006, retomei os conhecimentos em programação de computadores e passei a trabalhar com design e desenvolvimento de interfaces digitais para websites, aplicativos, e instalações interativas. Por meio da aplicação da computação em design, tive oportunidade de produzir trabalhos pessoais e comerciais que obtiveram reconhecimento nacional e internacional. Participei da exposição de arte eletrônica FILE (Festival Internacional de Linguagem Eletrônica) em 5 edições, fui convidado a produzir videoclipes para bandas internacionais, e tive uma animação exibida no festival Punto y Raya, realizado em Madri em 2012.

A seguir serão apresentados dois trabalhos que produzi com a aplicação de computação em design. O primeiro trabalho é *untitled 246* (2009)¹, exibido no festival Punto y Raya em 2012. As imagens (quadros ou *frames*) que compõem a animação foram geradas por um programa de computador, escrito na plataforma Processing. Ao longo de sua duração, é possível visualizar as alterações nos parâmetros do programa (cores, linhas, ângulos, etc) sendo executado. É importante ressaltar que não houve manipu-

¹Disponível em <http://vimeo.com/8393653>

Figura 1: untitled246, 2009.

Figura 2: untitled246, 2009.

lação direta de cada quadro da animação. Os quadros da animação são gerados por um sistema autônomo, cujos parâmetros podem ser manipulados. Fornecendo diferentes valores para os parâmetros, o mesmo sistema produz diferentes animações. É possível alterar os parâmetros e experimentar variações rapidamente.

O segundo projeto é *Golden Ages* (2011)², trabalho produzido a pedido da banda de rock Golden Ages, dos EUA. Este projeto consistiu no desenvolvimento de uma animação que reage aos sons produzidos pela performance da banda ao vivo. A animação é gerada por um programa de computador escrito em Processing. O programa é executado em um computador conectado ao sinal de áudio produzido pela banda. O sinal de áudio é recebido pelo programa, que utiliza essa informação como parâmetros para gerar a animação em tempo real. Assim como em *untitled 246*, não houve manipulação direta dos quadros das animações produzidas. Foi criado um sistema algorítmico de geração de imagens, e a performance da banda fornece os parâmetros que interferem na animação produzida computacionalmente. A interação em tempo real por meio do som é uma possibilidade aberta pela utilização da computação.

Figura 3: *Golden Ages*, 2011.

²Disponível em <http://vimeo.com/19880647>

Figura 4: *Golden Ages*, 2011.

O ponto de partida desta pesquisa de mestrado foi uma busca pessoal por reflexão sobre as relações entre design e computação, tema recorrente em minha trajetória profissional. A aplicação da computação em design permite rápida experimentação e formas inovadoras de interação. Esta pesquisa pretende contribuir para a desmistificação e a construção de uma visão crítica das relações entre design e computação.

Introdução

Esta dissertação tem como objetivo investigar o design computacional, explorando a aplicação de ferramentas conceituais da computação no projeto e na produção de imagens dinâmicas e interativas.

Com este objetivo, foi realizada pesquisa bibliográfica de livros e artigos nas áreas de meios de comunicação, cibernética, computação, artes, e design. Também foi realizado um levantamento de depoimentos de designers que trabalham com computação, relatando seus próprios processos de design. Os depoimentos de designers estrangeiros foram obtidos em obras bibliográficas, e os depoimentos de designers brasileiros foram obtidos por meio de entrevistas. Todas as traduções de textos em línguas estrangeiras são de autoria do autor da dissertação.

Em sua atividade profissional, o autor constatou que muitos profissionais intitulados “designers” possuem pouco ou nenhum conhecimento sobre as tecnologias que são utilizadas no desenvolvimento dos sites e aplicativos que projetam. Além disso, apesar de usarem diariamente ferramentas computacionais como o Photoshop, Illustrator, InDesign, etc, muitos desses designers não possuem uma posição crítica em relação a essas ferramentas. O surgimento do Photoshop e de tantos outros softwares apresentou inovações na forma de trabalhar em relação a métodos não-computacionais. No entanto, dada a atual ubiquidade e constante utilização dessas ferramentas no nosso cotidiano, tornou-se difícil ter um distanciamento, e por consequência, uma visão crítica a respeito das possibilidades e limitações delas. Mas essas ferramentas acabam moldando o próprio trabalho de design (projeto). Na prática, isso resulta em duas implicações importantes: projetos que são propostos mais pela facilidade de serem implementadas nos softwares do que pela adequação ao problema que o projeto procura

solucionar, e por outro lado, projetos que se mostram inviáveis tecnicamente porque o designer não conhece e não leva em consideração as limitações técnicas daquilo que está sendo projetado: seja um site, um aplicativo, ou qualquer outro sistema de interface digital.

Dessa forma, esta dissertação pretende contribuir para a construção de uma postura crítica a respeito dos efeitos da computação na atividade projetual. Para verificar como está tema tem sido tratado em outras teses e dissertações, foi realizada uma consulta ao sistema de teses da USP. Foram consultados teses de doutorado e dissertações de mestrado, produzidos na Fauusp, que mencionam o termo “computação” ou “computador” em seu título, resumo, ou palavras-chave. A consulta revelou diversas abordagens da computação:

- enquanto ferramenta de representação gráfica em arquitetura (ROYZEN, 2009; SANTOS, 2010), em design gráfico (MOURA, 2006), no ensino de geometria (GONÇALVES, 2009), e em audiovisual (GABBAI, 2006);
- enquanto ferramenta de representação gráfica e também síntese de novas formas em arquitetura, por meio dos paradigmas de projeto paramétrico e algorítmico (FERREIRA, 2009; NATIVIDADE, 2010);
- enquanto elemento do ambiente contemporâneo (VASSÃO, 2008; MAIA, 2013);
- enquanto elemento para estudo do webdesign (STOLFI, 2010);
- enquanto elemento que possibilita o desenvolvimento de novas linguagens de expressão (OLIVEIRA, 2006).

Constatou-se que 5 dos 11 trabalhos encontrados tratam da computação enquanto ferramenta de representação gráfica. Esta dissertação propõe que a computação contribui para o design em outras formas, que não apenas a sua utilização como ferramenta de representação gráfica.

Nesse sentido, a presente dissertação poderá contribuir à área de design digital apresentando uma pesquisa bibliográfica com referências das mídias, da computação, do design, e das artes, e apresentando uma sequência de depoimentos de designers computacionais a respeito de seus processos de trabalho.

A dissertação está estruturada em 3 capítulos que apresentam os diversos aspectos envolvidos no design computacional. O primeiro capítulo busca desmistificar a

computação como um assunto simplesmente técnico, que não deve ser estudado por designers, e discute a computação de forma abrangente, enquanto mídia que possui importantes consequências sociais. Em seguida, são apresentados alguns precedentes na história da arte e do design que exibem paralelos conceituais entre a computação, a arte, e o design. Por fim, chega-se a uma investigação do conceito específico de design computacional, apoiada em depoimentos e textos de designers contemporâneos que adotam a computação em seus trabalhos. Esta pesquisa foi motivada pelo interesse em refletir sobre a importância do conhecimento em computação para a atividade profissional do designer. No decorrer desta pesquisa, constatou-se que muitos pesquisadores e profissionais já refletiram sobre o tema. Observou-se que o conhecimento em computação é considerado cada vez mais importante para a participação das pessoas nos debates públicos da sociedade.

O primeiro capítulo da dissertação, *Computação como mídia e artefato mental*, tem a computação como tema central. Discute-se a importância da computação enquanto mídia. Essa discussão parte do princípio de que a sociedade é mais influenciada pela natureza dos meios de comunicação do que pelos conteúdos das mensagens transmitidas por esses meios de comunicação. Ou seja, meios de comunicação alteram a forma como o homem percebe e age no mundo, são extensões do homem (MCLUHAN, 1964). Em seguida, é apresentado o conceito de meta-mídia, que denota a capacidade do computador de reproduzir e expandir as outras mídias (KAY, GOLDBERG, 1977). Dessa forma, se as mídias exercem influência importante na sociedade, e computadores são um novo tipo de mídia, então computadores exercem influência importante na sociedade. O predomínio da computação como instrumento de produção, distribuição, e consumo de cultura (MANOVICH, 2013) torna ainda mais importante o estudo da computação em design, e remete à “filosofia da caixa-preta” de Vilém Flusser (2002). Se aparelhos técnicos são caixas-pretas (opacas, não-transparentes) para quem as utiliza, e se possibilidades foram pré-programadas pelos seus fabricantes, surge uma relação de poder entre o fabricante que programa o aparelho e o usuário que manipula o aparelho. Se computadores forem tratados como caixas-pretas por seus usuários, estes sempre trabalharão dentro das possibilidades pré-embutidas no programas. Ainda no primeiro capítulo, discute-se a computação enquanto artefato

mental. São discutidas as ideias dos pioneiros da computação pessoal J.C.R. Licklider (1960), Douglas Engelbart (1962), Vannevar Bush (1945), e Alan Kay (1977), que vislumbravam na computação uma forma de expandir o intelecto humano. Também é abordado o conceito de “pensamento computacional” (WING, 2006), segundo o qual a computação fornece elementos conceituais que auxiliam na resolução de problemas de qualquer tipo. Há a perspectiva de que a computação não consiste apenas na programação de computadores, mas também em um conjunto de princípios conceituais que podem ser aplicados em qualquer área do conhecimento (DENNING, 2010).

Tomando a computação nesse sentido mais amplo e conceitual, o segundo capítulo, *Design como programa*, apresenta paralelos entre design, arte, e computação. É apresentado o conceito de arte generativa, que utiliza sistemas autônomos ou semi-autônomos no processo de criação artística (GALANTER, 2003). Na arte generativa também é possível obter um número de soluções a partir de um mesmo programa, ou sistema, proposto. A possibilidade de incorporar outros inputs, como a participação da audiência, ou fatores ambientais capturados por sensores, abre o leque de soluções geradas, e traz novas questões para o artista que concebe esse tipo de arte. Em seguida é apresentado o trabalho *Designing programmes*, de Karl Gerstner (1964), que trata o design como programa: “ao invés de soluções para problemas, programas para soluções” (GERSTNER, 1964, p.12). O paralelo entre design e programa está na possibilidade do resultado (output) consistir de um conjunto de soluções, e não uma única solução.

No terceiro capítulo, *Práticas programáticas*, são apresentados depoimentos de designers que trabalham com computação, procurando identificar a aplicação dos conceitos apresentados nos capítulos anteriores.

Por fim, são apresentadas as considerações finais da pesquisa e indicações de possibilidades para trabalhos futuros.

Capítulo I

Computação como mídia e artefato mental

Este capítulo discute como a computação influencia o pensar do designer. Para tal, faz-se necessário em primeiro lugar definir o que se considera como computação. Nesta dissertação partiu-se de uma definição proposta pelo grupo Joint Task Force for Computing Curricula 2005, um projeto cooperativo de 3 sociedades científicas dedicadas às ciências da computação: Association for Computing Machinery (ACM), Association for Information Systems (AIS), e Computer Society (IEEE-CS). O objetivo desse grupo é apresentar um panorama dos cursos de graduação em ciências da computação oferecidos nos EUA e no Canadá. Em relatório apresentado em 2006, o grupo propõe a seguinte definição de computação:

De forma geral, podemos definir computação como qualquer atividade orientada a um objetivo que requer, se beneficia de, ou cria computadores. Assim, computação inclui projetar e construir sistemas de hardware e software para uma ampla variedade de propósitos; processando, estruturando e gerenciando vários tipos de informação; executando estudos científicos por meio de computadores; fazendo sistemas de computadores se comportarem de forma inteligente; criando e usando meios de comunicação e entretenimento; encontrando e recolhendo informações relevantes para qualquer propósito particular, e assim por diante. A lista é virtual-

mente infinita, e as possibilidades são vastas. (JOINT TASK FORCE FOR COMPUTING CURRICULA 2005, 2006, p.9).¹

Se percebe que, mesmo no universo da engenharia, a computação é um campo bastante amplo, que compreende qualquer atividade que necessite de computadores, que utilize computadores, ou que resulte em novos computadores. Dessa maneira, a computação compreende tanto a atividade de criar um programa de computador, quanto a atividade de utilizar um programa de computador. Aplicando esta definição na atividade de design, observa-se a presença da computação em diversas etapas de um mesmo projeto, por exemplo: quando um designer utiliza computadores e software para projetar um website, quando esse website é distribuído pela internet, e quando um usuário interage com esse website. É interessante observar que a computação não necessariamente envolve a manipulação direta de um computador, argumento que é defendido pela pesquisadora Jeannete Wing e que será discutido mais adiante.

Constata-se que a computação está cada vez mais presente nas atividades do homem. A computação hoje é distribuída, ubíqua, pervasiva, e conecta não só as pessoas, mas também as “coisas: objetos, veículos, produtos de consumo, alimentos, etc” (MAIA, 2013, p.12). A computação vem se espalhando e se infiltrando nas coisas, atingindo um alto grau de presença no ambiente. E assim a computação se torna quase imperceptível para as pessoas: não é mais possível perceber os processos de interação (entre os diversos atores dos sistemas interativos, sejam pessoas ou coisas) em andamento em um dado momento (VASSÃO, 2008). Diante desse cenário de crescente presença da computação no ambiente urbano contemporâneo, torna-se importante estudar e refletir sobre a computação, buscando compreender o mundo e buscando os caminhos possíveis para atuar nesse mundo.

¹In a general way, we can define computing to mean any goal-oriented activity requiring, benefiting from, or creating computers. Thus, computing includes designing and building hardware and software systems for a wide range of purposes; processing, structuring, and managing various kinds of information; doing scientific studies using computers; making computer systems behave intelligently; creating and using communications and entertainment media; finding and gathering information relevant to any particular purpose, and so on. The list is virtually endless, and the possibilities are vast (JOINT TASK FORCE FOR COMPUTING CURRICULA 2005, 2006, p.9).

1.1 Computação como mídia

A ideia de pensar a computação como mídia parte de questões colocadas por autores como McLuhan (1964), Kay (1977), Manovich (2013), e Flusser (2002). Uma das grandes contribuições de McLuhan foi sua defesa dos estudos das mídias, e não apenas dos conteúdos transmitidos por elas. McLuhan identificou que as características intrínsecas das mídias moldam as mudanças sociais e culturais em uma sociedade. Ou seja, se as mídias alteram a forma como o homem percebe e age no mundo, e a computação é mídia (na concepção de McLuhan), então a computação altera a forma como os homens percebem e agem no mundo.

1.1.1 Meio e mensagem

Uma das frases mais reproduzidas dos textos de McLuhan é “o meio é a mensagem”, que dá nome ao primeiro capítulo de seu livro *Understanding media: the extensions of man* (1964). Para compreender o que essa expressão significa, é preciso pensar o termo **mídia** no contexto de sua obra. Para McLuhan, mídia não se limita aos meios de comunicação convencionais como jornal, rádio, e televisão. Mídia pode se referir a “qualquer extensão de nós mesmos” ou ainda, “qualquer nova tecnologia” (MCLUHAN, 1964, p.15). Em *The medium is the message*² (1967), McLuhan propõe que “todas as mídias são extensões de alguma faculdade humana – psíquica ou física” (MCLUHAN; FIORE; AGEL, 1967, p.22), e lista alguns exemplos: a roda como extensão do pé, o livro como extensão do olho, o vestuário como extensão da pele, e os circuitos elétricos como extensão do sistema nervoso central. Na visão de McLuhan, rodas, livros, roupas, e circuitos elétricos são mídias.

No pensamento de McLuhan, há uma diferença entre **conteúdo** e **mensagem**. Conteúdo corresponde à utilização de uma mídia, e consiste em outra mídia. Por exemplo, o conteúdo de um livro é a palavra escrita, e o conteúdo da escrita é um discurso.

²*The medium is the message* (MCLUHAN; FIORE; AGEL, 1967) é uma adaptação de *Understanding media*, uma versão resumida com metáforas visuais que procuram representar visualmente as ideias de McLuhan sobre a natureza das mídias. Esta obra apresenta uma linguagem gráfica dinâmica e fragmentada, com sobreposições e justaposições de textos, colagens, ilustrações, e fotografias. Sua linguagem gráfica é uma tentativa de representar a natureza instantânea da eletricidade no meio impresso. Nesse sentido, pode ser considerado uma obra precursora de trabalhos em mídias digitais.

Por sua vez, o conteúdo do discurso é o próprio pensamento, que é intrinsecamente não-verbal. Mensagem é a mudança de escala ou ritmo que uma mídia introduz em processos existentes, com consequências psíquicas e sociais para o homem e a sociedade. Dessa forma, “o meio é a mensagem” significa que as mídias afetam o homem nos níveis pessoal e social em função de suas características intrínsecas. Por exemplo, a mídia máquina pode produzir milho ou carros como conteúdos. Mas a mensagem da mídia máquina está nas mudanças que a mecanização trouxe para as relações humanas de trabalho, como resultado das características intrínsecas do próprio conceito de mecanização: divisão de um processo em etapas e a execução dessas etapas em sequência linear. Essa mensagem independe do produto (conteúdo) específico produzido por uma máquina (mídia) específica.

Outros exemplos clássicos que McLuhan utiliza para ilustrar os efeitos das mídias na sociedade são as ferrovias e a eletricidade. A ferrovia acelerou deslocamentos e favoreceu o desenvolvimento de novas formas de habitação (subúrbios), trabalho, e lazer. A mensagem da ferrovia enquanto mídia resulta de sua velocidade, e independe do que é transportado por uma ferrovia específica em uma situação específica. A eletricidade trouxe uma mudança de paradigma ainda maior, pois ela não introduziu uma simples mudança de velocidade, sua velocidade é tão alta que é percebida como instantânea. Se o paradigma da mecanização era centrado na sequencialidade, como consequência da subdivisão dos processos em etapas lineares, o paradigma da eletricidade é centrado na simultaneidade, possível graças à sua velocidade instantânea. Novamente, a mensagem da eletricidade enquanto mídia independe de alguma finalidade específica para a qual é utilizada.

Os textos de McLuhan foram escritos nos anos 60, e contemplavam meios de comunicação de massa disponíveis na época, como o rádio e a televisão. Podemos buscar outros exemplos no universo de tecnologias e mídias hoje disponíveis. Tecnologias de comunicação computacionais e conectadas em rede como a telefonia celular e a internet (com suas ferramentas: emails, mensagens, videoconferência) têm favorecido o desenvolvimento de formas de trabalho colaborativo com equipes ou pessoas distribuídas em diferentes locais ao redor do mundo. O chamado “trabalho remoto”, no qual um funcionário de uma empresa trabalha em sua casa, ao invés de se deslocar

até o escritório da empresa, têm se tornado cada vez mais comum. Com a utilização de ferramentas informatizadas de gerenciamento de projetos, até mesmo os horários de trabalho podem se tornar mais flexíveis. Como consequência, os horários de lazer também se tornam flexíveis. Novas formas de lazer também surgem, como por exemplo os chamados jogos *massively multiplayer online* (MMOs), nos quais jogadores conectados em rede jogam colaborativamente e/ou competitivamente. Redes sociais facilitam relações interpessoais à distância, e pessoas podem interagir e trocar informações com pessoas com interesses similares mas que vivem em outros locais. Artistas (sejam músicos, cineastas, artistas plásticos, escritores) podem produzir, divulgar, e comercializar seus trabalhos sem depender de intermediários.³ O universo de mídias computacionais e conectadas hoje disponíveis é vasto. E apesar de recentes, já é possível observar as mudanças que essas mídias trazem para a sociedade, as relações de trabalho, as formas de lazer e de entretenimento, os relacionamentos interpessoais, a produção, distribuição, e consumo de arte. Nos capítulos seguintes, esta dissertação discute o efeito da computação enquanto mídia na atividade de design.

“O meio é a mensagem” significa que as “sociedades sempre forma moldadas mais pela natureza dos meios pelos quais homens se comunicam do que pelo conteúdo da comunicação” (MCLUHAN; FIORE; AGEL, 1967, p.8). As mídias mudam o ambiente, alterando os sentidos, e por consequência a forma como o homem percebe o mundo. Assim, alteram a forma como o homem pensa e age no mundo. No entanto, McLuhan não está propondo um determinismo tecnológico, pelo contrário. Para ele, tecnologias não são entidades autônomas, são extensões do homem, criadas pelo homem. O homem, sob efeito das tecnologias que criou, é quem age no mundo. O que McLuhan propõe é que uma consciência dos efeitos das mídias permite ao homem compreender melhor as mudanças sociais, e assim escapar de um determinismo tecnológico.

O último capítulo de *Understanding media* é dedicado à automação como mídia. McLuhan define a automação como uma “invasão do mundo mecânico pelo caráter instantâneo da eletricidade” (MCLUHAN, 1964, p.335). Nesse sentido, a automação é tanto uma “forma de pensar” quanto uma “forma de fazer” (MCLUHAN, 1964, p.335),

³No último capítulo de *Understanding media*, dedicado à automação, McLuhan prevê algumas dessas mudanças: “os padrões sociais e educacionais latentes na automação são aqueles do emprego autônomo e da autonomia artística” (MCLUHAN, 1964, p.344).

pois a possibilidade de acessar informação instantaneamente transforma tarefas sequenciais em tarefas paralelas e transforma as partes independentes dos processos industriais em novos processos interrelacionais. Se a mecanização implicava na divisão de processos em partes autônomas, a automação requer a reunificação dessas partes, pois a natureza instantânea da eletricidade exige um alto grau de relacionamento entre as partes. Se os processos mecânicos abraçam a especialização para aumentar a eficiência, os processos automáticos abraçam a complexidade, e são menos especializados. As máquinas automáticas são adaptáveis e programáveis. A automação refuta a padronização e a especialização mecanicista, e promove a diversificação e a descentralização. Se a palavra impressa propiciou um processo fragmentador, de especialização e de desapegamento, a eletricidade alimenta um processo unificador e envolvente (MCLUHAN; FIORE; AGEL, 1967).

Uma crítica que pode ser feita aos textos de McLuhan é sua tendência de fazer generalizações. Isso gera algumas aparentes dicotomias, por exemplo, quando ele contrasta as características da mecanização com as da eletricidade. No entanto, ele mesmo reconhece que qualquer tecnologia apenas “adiciona ao que já existe” (MCLUHAN, 1964, p.18). E reconhece que a instantaneidade da eletricidade não surge como negação dos princípios da mecanização, e sim como soma. Tendo isso em vista, sua obra tem grande mérito ao discutir a importância das mídias na sociedade.

McLuhan fala da automação mas faz poucas menções ao computador em seus textos, e se concentra em falar da eletricidade e de meios de comunicação mais comuns na década de 60, como o rádio e a televisão. Computadores já existiam na época, mas ainda eram máquinas de grande porte restritas às grandes empresas e universidades. Mesmo assim, quando diz que “automação certamente supõe o servomecanismo e o computador” (MCLUHAN, 1964, p.337) e que o “computador oferece um modelo que tem as características compartilhadas por todas as formas de automação” (MCLUHAN, 1964, p.342), compreende-se que a computação e a automação são conceitos intimamente relacionados. Se por um lado, a capacidade de automatizar tarefas é um dos conceitos centrais da computação, por outro lado, a automação pode ser implementada de outras formas que não a computação. Ou seja, são dois conceitos amplos que possuem pontos de intersecção. Mesmo que os dois conceitos não sejam

exatamente iguais, é possível aplicar o que McLuhan fala sobre a automação para a computação também. A computação também trabalha com modelos complexos de representação e manipulação de informações, que geram novas formas de pensar e de fazer. Embora o código-fonte de um programa seja escrito de maneira linear, a execução das instruções contidas no código não é necessariamente linear. O programa pode reagir a mudanças nos parâmetros recebidos, alterando o curso da informação que navega pelo código-fonte. Além disso, a velocidade com a qual a informação flui pelo programa pode ser tão alta que acaba sendo percebida como instantânea: em computadores pessoais atuais, o tempo de execução dos programas são geralmente medidos em milissegundos. Ou seja, quando McLuhan diz que a automação abraça a simultaneidade, a complexidade, a adaptabilidade, a programabilidade, o mesmo pode ser dito a respeito da computação. A possibilidade de programar e realizar simulações utilizando a computação, testando hipóteses, permite ao homem lidar com problemas mais complexos, diversificando sua ação no mundo.

A importância da capacidade de um computador ser programado e realizar simulações é exatamente um dos pontos centrais na obra do cientista da computação Alan Kay. Depois de uma década depois da publicação de *Understanding media*, Kay desenvolve o conceito de meta-mídia, que caracteriza a computação como uma nova forma de mídia, com importantes implicações que são discutidas a seguir. Atualmente, Kay é professor adjunto de Ciências da Computação na University of California, Los Angeles (UCLA) e presidente da Viewpoints Research Institute, organização sem fins lucrativos dedicada a pesquisas que envolvem crianças, aprendizado, e computação. Pesquisadores como Rheingold (1985) e Manovich (2013) colocam Kay como uma das figuras mais importantes na história da computação e das interfaces digitais.

1.1.2 Meta-mídia

Em seu artigo *User interface: a personal view* (1989), Kay fala de sua leitura de *Understanding media* de McLuhan. Embora considere muitos trechos do livro “obscuros e discutíveis” (KAY, 1989, p.124), Kay reconhece que o impacto do texto de McLuhan permanece após sua leitura, pois foi essa leitura que lhe permitiu enxergar a computação como mídia. Kay diz que sempre havia pensado no computador como ferramenta, mas que

a teoria de McLuhan elevava o computador a uma condição de mídia que poderia trazer “mudanças nos padrões de pensamento de uma civilização inteira” (KAY, 1989, p.125). Kay observa, a partir de sua leitura de McLuhan, que embora as ferramentas sejam modeladas pelo homem, é a natureza das ferramentas que acaba remodelando o homem. E esta observação aponta para outra marca importante em sua obra, que é a busca pela invenção. Uma das frases mais conhecidas de Kay é: “a melhor forma de prever o futuro é inventá-lo” (KAY, 1989, p.131)⁴. Aliando a consciência do poder das mídias que absorveu de McLuhan e a busca pessoal pela invenção, Kay desenvolveu o projeto Dynabook na década de 70. E é no desenvolvimento do Dynabook que Kay desenvolve um conceito que demonstra o potencial expansivo da computação enquanto mídia: o conceito de **meta-mídia**.

Figura 1.1: Alan Kay. Dynabook, 1972. Fonte: KAY, 1972, p.6.

O projeto do Dynabook (figura 1.1) é apresentado em detalhes no artigo *Personal dynamic media* (1977), escrito em parceria com Adele Goldberg. O Dynabook foi desenvolvido no Grupo de Pesquisa em Aprendizado (*Learning Research Group*) no Centro

⁴The best way to predict the future is to invent it (KAY, 1989, p.131).

de Pesquisas da Xerox em Palo Alto (Xerox Palo Alto Research Center, ou Xerox PARC), e consiste em um dispositivo pessoal, portátil, do tamanho de um caderno de anotações, com o qual seu usuário poderia resolver todas suas necessidades relacionadas com informações. O Dynabook é definido como “uma mídia dinâmica para pensamento criativo” (KAY; GOLDBERG, 1977, p.176), capaz de captar imagens e sons, armazenar grandes quantidades de documentos, registros sonoros, desenhos, animações, simulações, e qualquer outra informação que o usuário quisesse guardar. Se McLuhan dizia que o conteúdo de uma mídia é outra mídia, Kay percebeu uma particularidade da mídia computador: seu conteúdo pode ser qualquer outra mídia. Em outras palavras, o computador é uma **meta-mídia** (KAY; GOLDBERG, 1977, p.394).

Uma primeira consideração no projeto do Dynabook foi a relação do ser humano com seus meios de comunicação. Além de comunicarem ideias e emoções, meios de comunicação funcionam como registros de pensamentos de indivíduo, que por meio de feedback, podem contribuir para o surgimento de novos pensamentos. No entanto, meios de comunicação anteriores à computação, como o rádio e a televisão, não conseguem responder ativamente às interações do receptor, funcionando como vias de mão única, sem feedback. Mensagens são representações de ideias, e o que caracteriza um meio de comunicação (mídia) é a forma como as mensagens são incorporadas. Kay e Goldberg consideraram a possibilidade do computador ser visto como uma mídia capaz de representar outras mídias, e de responder às interações do usuário de forma ativa, funcionando efetivamente como vias de mão dupla de comunicação. Aqui, é importante citar a conceituação de feedback proposta pelo cientista da computação Norbert Wiener. A teoria da cibernética de Wiener tinha o objetivo de estudar as “mensagens como meios de controlar máquinas” (WIENER, 1954, p.48), e o feedback é um dos conceitos centrais dessa teoria. Feedback é um mecanismo pelo qual seres humanos, animais, e máquinas, ajustam seus próprios comportamentos, de acordo com as mensagens recebidas pelos órgãos receptores de seus sistemas sensoriais. Os homens percebem o mundo por meio de seus sentidos. As informações recebidas pelos órgãos receptores do corpo humano são processadas pelo cérebro e sistema nervoso, que por sua vez ativam os órgãos efetores do corpo. Desse modo, o homem age no mundo, emitindo novas mensagens ao seu ambiente, que são também percebidas

por seu sistema sensorial, reiniciando um ciclo de percepção e ação. De maneira similar, máquinas automáticas como “o míssil controlado, o fusível de proximidade, a porta automática, o aparato de controle de uma indústria química, e o resto do arsenal de máquinas automáticas que executam funções industriais ou militares” (WIENER, 1954, p.51) também possuem sensores que captam as mensagens do ambiente a cada instante. Esses sensores permitem que as máquinas ajustem suas próprias ações para assim cumprir suas finalidades. Por meio de feedback, as informações do ambiente são percebidas e transformadas internamente pelo receptor, que age no mundo de acordo com novas informações recebidas. O que Kay e Goldberg propõem é que um meio de comunicação como o rádio ou a televisão não possuem essa característica de comportamento auto-ajustável por meio de feedback, mas o computador sim. Além disso, a possibilidade do computador responder de acordo com as interações do usuário alimenta outro ciclo de feedback no próprio usuário. Se o computador responde ativamente à interação do homem, essa resposta é percebida pelo sistema sensorial do homem, que então ajusta suas ações de acordo com essa resposta. A importante implicação desse conceito é que o feedback do computador permite “aumentar os caminhos que o pensamento percorre” (KAY; GOLDBERG, 1977, p.393)⁵.

Durante o desenvolvimento do Dynabook, Kay testou seu protótipo com grupos de crianças em idade escolar. O Dynabook era um sistema autocontido (ao contrário de outros computadores que têm suas partes distribuídas), que continha um compartimento para inserção de um disco de memória, e que estava conectado a equipamentos como monitores de raios catódicos, diferentes tipos de teclados (teclado alfabético, teclado de acordes, teclado musical), mouses, e caixas de som.

O Dynabook poderia ser utilizado para a leitura de livros, cujos conteúdos poderiam ser pesquisados, e lidos de forma não-linear. Todos os objetos no Dynabook poderiam ser editados e agrupados. O dispositivo também poderia ser utilizado para criação de desenhos, pinturas, animações, e músicas. A linguagem de programação Smalltalk foi desenvolvida para permitir que os usuários pudessem criar suas próprias ferramentas com o Dynabook. Essa linguagem de programação foi projetada para que até mesmo as crianças pudessem utilizá-la. Em *Personal dynamic media*, Kay e Goldberg

⁵augment the actual paths the thinking follows (KAY; GOLDBERG, 1977, p.393).

apresentam algumas ferramentas para animação, desenho, e música criadas pelas próprias crianças com o Smalltalk e o Dynabook.

Kay visualizava possíveis utilizações de uma metamídia como o Dynabook por profissionais diversos como arquitetos, médicos, compositores, administradores, e educadores, que poderiam personalizar a mídia para atender às suas necessidades particulares. Uma das metas para o Dynabook é que ele fosse um produto flexível e conseguisse atender um público amplo. Para Kay, era essencial que as pessoas pudessem gerar conteúdo com o computador. Fazendo uma analogia com a palavra escrita (KAY, 1989), uma pessoa alfabetizada consegue ler textos escritos por outras pessoas e consegue também escrever textos para se comunicar com outras pessoas. Uma pessoa “alfabetizada” na mídia computador deve ser capaz de acessar conteúdos produzidos por outros, assim como produzir conteúdos para outros. Kay diz que “acreditava que usuários precisavam ser capazes de programar antes que o computador pudesse se tornar verdadeiramente deles” (KAY, 1989, p.125)⁶, e que os testes com o Dynabook corroboraram sua posição. Ao relatar os resultados dos testes do Dynabook com crianças, Kay exalta o entusiasmo das crianças ao sentirem “que estão fazendo coisas reais ao invés de brincar com brinquedos” (KAY; GOLDBERG, 1977, p.394), quando conseguiam escrever programas de computador que cumpriam objetivos que elas mesmo tinham determinado. Ou seja, a experiência de programar resultava em um grande senso de realização. Kay estudou o trabalho de Seymour Papert na linguagem de programação Logo, e foi por meio de Papert que chegou à obra de Jean Piaget e de outros psicólogos e estudiosos do aprendizado como Jerome Bruner. Kay conclui que a linguagem de programação Logo de Papert teve sucesso por ter sido desenvolvida tendo em vista a mente de seus usuários (crianças). Outra conclusão de Kay é que a utilização de uma interface está interrelacionada com seu aprendizado: usar uma interface é aprender a utilizá-la. A natureza “orientada a objetos”⁷ da linguagem de programação

⁶I had believed that end users needed to be able to program before the computer could become truly theirs (KAY, 1989, p.125).

⁷Linguagens de programação “orientadas a objetos” são baseadas na definição de entidades chamadas “objetos” que trocam “mensagens” entre si. Em geral, os objetos possuem representações genéricas que são chamadas de “classes”. Ao delimitar funcionalidades em objetos separados, o objetivo principal desse paradigma de programação é tornar o código-fonte do programa mais legível, facilitando futuras alterações e correções no programa.

Smalltalk foi a solução encontrada por Kay para tornar a linguagem mais intuitiva e sugestiva para o usuário.

Observa-se que o Dynabook proposto por Kay e Goldberg se assemelha aos tablets disponíveis no mercado atualmente, em termos de formato e dimensões físicas. No entanto, o projeto de Kay e Goldberg possui uma diferença conceitual fundamental com os artefatos contemporâneos. Enquanto o objetivo central da proposta do Dynabook era ser um instrumento de criatividade e de criação pelo seu usuário, os tablets de hoje estão mais voltados ao consumo de conteúdos do que à criação. Em entrevista fornecida à revista *Time* em 2013, Kay declara que a intenção original do Dynabook era que o dispositivo fosse utilizado na mesma medida para a autoria e para o consumo de conteúdo. No entanto, Kay observa que os iPads e os iPhones da Apple são plataformas fechadas e muitas vezes nem permitem a troca de informações entre dispositivos de diferentes usuários. Kay conclui que essa implementação “não poderia estar mais distante dos objetivos originais” das pesquisas na Xerox PARC nas décadas de 60 e 70 (GREENLISH, 2013).

A possibilidade do usuário poder criar seus próprios programas, e não apenas utilizar os programas pré-embutidos no dispositivo era central para o papel do Dynabook como dispositivo de “pensamento criativo”. O Dynabook fornecia uma linguagem de programação, projetada para ser intuitiva e fácil de usar, que permitia a qualquer usuário moldar o dispositivo para suas necessidades, podendo assim inventar novos tipos de mídia ainda não existentes. Em vários trechos do artigo *Personal dynamic media*, Kay e Goldberg utilizam o termo “dono” ao invés de “usuário”, indicando que a possibilidade de programar o dispositivo significa o domínio do dispositivo pelo usuário. A programação dos aparelhos técnicos, e a relação de domínio decorrentes, são temas centrais na obra do filósofo Vilém Flusser, que é discutida ainda neste capítulo.

Se Kay foi influenciado por McLuhan, um pesquisador que foi fortemente influenciado por Kay e McLuhan é Lev Manovich. A obra mais recente de Manovich é *Software takes command*, na qual apresenta a importância do software como mídia na sociedade contemporânea. Por software, entende-se qualquer programa de computador. Um software é uma concretização da capacidade do computador de ser programado, e dessa forma é um dos aspectos centrais da computação. Manovich defende a im-

portância dos estudos do software, assim como McLuhan defendeu a importância dos estudos das mídias. Mas Manovich argumenta que a ideia de meta-mídia proposta por Kay desafia alguns aspectos da teoria das mídias de McLuhan.

1.1.3 Software

A principal justificativa para os estudos do software é que “nós vivemos em uma cultura de software – ou seja, uma cultura em que a produção, distribuição, e recepção da maioria dos conteúdos é mediada por software” (MANOVICH, 2013, p.751)⁸. Ou seja, o software está presente em todas as etapas da vida útil de objetos culturais: um estúdio utiliza software na produção de um vídeo, um canal de vídeos utiliza software para armazenar e distribuir os vídeos, uma pessoa utiliza software para assistir o vídeo. O software alcançou uma posição de alta importância na cultura, na sociedade, na economia. Dentre as mais poderosas empresas contemporâneas, muitas trabalham com software: Google, Facebook, Apple, Microsoft, entre outras. O software se tornou parte integrante da vida cotidiana de milhões de pessoas, “o software se tornou nossa interface com o mundo, com os outros, com nossa memória e com nossa imaginação” (MANOVICH, 2013, p.155).

Tendo em vista a ubiquidade do software na vida contemporânea, e com base nas obras de McLuhan e Kay, Manovich afirma ainda que o software não é apenas mais um tipo de mídia. Para ele, a natureza meta-mídia do software supera a definição de mídia proposta por McLuhan. A pergunta-chave que Manovich coloca é: o que acontece com o conceito de “mídia” quando o software é capaz de simular e estender as mídias anteriores? Se uma mídia (software) pode simular e estender as outras mídias, então talvez já não faça mais sentido tratar as mídias separadamente, mas sim como manifestações de uma grande meta-mídia que é o software. Em artigo publicado em seu blog⁹, Manovich lança uma provocação: hoje não é mais o meio que é a mensagem, mas sim o software: “o software é a mensagem”.

Por um lado, a possibilidade do computador ser cada vez mais programado por

⁸We live in a software culture that is, a culture where the production, distribution, and reception of most content is mediated by software (MANOVICH, 2013, p.751).

⁹<http://lab.softwarestudies.com/2013/12/software-is-message-new-mini-article.html>

“leigos” (não apenas cientistas da computação em laboratórios de pesquisa), permite que indivíduos produzam suas próprias ferramentas. A produção de *plugins*, *frameworks*, e bibliotecas de código, por indivíduos ou pequenos grupos, compartilhadas publicamente pela internet¹⁰ se tornou um fenômeno cultural corrente. Por outro lado, o software exerce papel central no modelo de negócio de grandes corporações como Google e Facebook.

O software permite um alto grau de conectividade entre as pessoas, promovendo novas formas de comunicação e colaboração. Conceitos como “sociedade da informação”, “mídias sociais”, e “crowdsourcing” passam a ser temas de teses acadêmicas e livros. Mas Manovich defende que limitar os estudos aos aspectos sociais do software é confundir causa e efeito: a “sociedade da informação” é efeito, o software é a causa. Observa que o software, que permitiu o desenvolvimento desses fenômenos, ainda é pouco visto como uma “categoria teórica” (MANOVICH, 2013, p.255) em si. Dessa maneira, argumenta que o software não deve ser um tema restrito aos departamentos de ciência da computação das universidades, e que ele não deve ser visto como “algo que está lá apenas para ajudá-los a tornarem-se mais eficientes”, e sim como “a mídia na qual a criatividade humana hoje reside” (MANOVICH, 2013, p.336)¹¹.

É importante observar que o computador, enquanto hardware, é um conjunto de circuitos e componentes eletrônicos com poucas funções embutidas. É o software que confere ao computador sua flexibilidade e adaptabilidade para executar as mais diversas funções. Essa diferença entre hardware e software já era visível nas máquinas de cálculo desenvolvidas pelo inventor inglês Charles Babbage no século XIX, consideradas precursoras dos computadores modernos. Para o professor de ciência da computação Gregory Rawlins (1998), a grande inovação da máquina de Babbage foi a maneira como ele simplificou a parte física da máquina, de forma que operações complexas pudessem ser executadas como combinações das operações simples embutidas nessa máquina:

¹⁰O movimento *open-source* surgiu da distribuição de software e seus respectivos códigos-fonte. A disponibilidade do código-fonte permite que qualquer outro programador altere o software à sua maneira. Essa filosofia de compartilhamento se expandiu para outras esferas de produção cultural, como por exemplo a música.

¹¹“something which is only there to help them become more efficient, as opposed to the medium where human intellectual creativity now dwells (MANOVICH, 2013, p.336).

Por um lado, a parte importante de sua máquina não era o hardware – a pilha complexa de bronze e relojoaria que ele usou para construí-la – mas os cartões perfurados dizendo à máquina o que fazer – o software. De repente, máquinas poderiam se tornar quase insubstanciais: poucas partes físicas fazendo poucas coisas simples, com todas as outras ações programadas por software. (RAWLINS, 1998, p.9)¹²

Babbage percebeu que poderia realizar tarefas complexas a partir da combinação de poucas partes simples. Por exemplo, para realizar cálculos, um computador deve ser capaz de realizar as quatro operações matemáticas básicas: adição, subtração, multiplicação, e divisão. No entanto, nem todas essas operações precisam estar embutidas em seus circuitos eletrônicos. Como o resultado de uma multiplicação pode ser obtido pela repetição da adição, a multiplicação pode ser programada, ou seja, realizada em software ao invés de em hardware. Programas de computadores são sequências de instruções que permitem construir operações mais complexas a partir de operações simples. Ou seja, o papel central que o software ocupa na sociedade contemporânea está diretamente relacionado com a importância do software como realização da capacidade do computador de ser programado.

Uma reflexão sobre os programas embutidos nos aparelhos técnicos, e das relações de poder resultantes desses programas, pode ser encontrada na obra do filósofo Vilém Flusser. Embora Flusser não fale especificamente da computação, sua reflexão é extremamente relevante para esta pesquisa. Para Flusser, as possibilidades de um aparelho são programadas pelo fabricante, e quem manipula o aparelho trabalha dentro do universo de possibilidades pré-programadas pelo fabricante. Assim como McLuhan alertou para a importância de se desenvolver uma consciência do poder persuasivo das mídias, Flusser defende a importância da reflexão a respeito dos aparelhos técnicos, que no fundo é uma reflexão sobre liberdade, sobre “as possibilidades de se viver livremente num mundo programado por aparelhos” (FLUSSER, 2002, p.41).

¹²For one thing, the important part of his engine was not the hardware – the higgledy-piggledy pile of brass and pewter clockwork he used to build it – but the punch cards telling it what to do – the software. Suddenly, machines could become almost insubstantial: a few physical parts doing a few simple things, with all the other actions programmed by software (RAWLINS, 1998, p.9).

1.1.4 Caixas pretas

A obra de Vilém Flusser é vasta, mas é em *Filosofia da caixa preta* (2002) que ele trata da importância da programação dos aparelhos no estabelecimento de relações de poder entre quem fabrica um aparelho e quem manipula um aparelho. O teórico das artes Arlindo Machado (2007) observa que esta obra de Flusser era originalmente intitulada *Por uma filosofia da fotografia* (ou *Für eine Philosophie der Fotografie* em alemão). A versão em português, escrita pelo próprio Flusser, sofreu alterações significativas em relação ao texto original em alemão, inclusive a mudança do título. Machado aponta que a substituição da palavra “fotografia” pela palavra “caixa preta” expressa com mais clareza o tema dessa obra.

Flusser usa as imagens e as fotografias como ponto de partida de seu texto, para discutir os aparelhos e seus programas. Desse modo, identifica inicialmente que as imagens atuam como uma forma de mediação entre homem e mundo. As imagens codificam a realidade. Em uma pintura ou uma fotografia, as dimensões espaciais e temporais de qualquer evento que se pretende representar são compactadas e convertidas em uma superfície bidimensional e estática. As imagens se propõem a ser uma representação do mundo, mas acabam se tornando um obstáculo quando o homem exerce idolatria pelas imagens. Ao invés de enxergar as imagens como representações do mundo, acaba visualizando o mundo como consequência das imagens.

Imagens técnicas, como as fotografias, são aquelas produzidas por aparelhos. A natureza científica dessas imagens resulta em uma aparente fidelidade das fotografias em relação à realidade. No entanto, Flusser argumenta que essa aparente proximidade entre representação e realidade acaba escondendo ainda mais a natureza mediadora das imagens. Um observador acaba percebendo uma imagem técnica como uma janela para o mundo, mas a imagem técnica ainda é uma representação, como qualquer outra imagem.¹³ Uma câmera fotográfica é resultado de uma síntese de vá-

¹³É possível identificar nessa caracterização das imagens como representações do mundo um eco do princípio de cartografia e decalcomania de Deleuze e Guattari (1980), um dos princípios que fazem parte do conceito de rizoma apresentado por eles em *Mil platôs*. Para Deleuze e Guattari, mapas estão associados com rizomas, e decalques estão associados com árvores. Mapas e rizomas são abertos e construtivos, decalques são fechados e representativos. Decalcar um caminho dentro de um mapa (delineando uma árvore, uma hierarquia) é limitar a visão do todo (o rizoma). Aplicando esses conceitos no texto de Flusser, uma fotografia é um decalque do rizoma que é o mundo. Aquele que percebe

rios conhecimentos científicos de física (ótica, mecânica, eletrônica) e química (nos filmes fotográficos e processos de revelação). É a alta complexidade desses conhecimentos científicos que transforma os aparelhos em caixas pretas, não-transparentes e indecifráveis para quem as utiliza.

Para Flusser, o que caracteriza um aparelho é o estar programado. O produto que se espera obter da manipulação de um aparelho deve ser previamente inscrito no próprio aparelho. Uma câmera fotográfica é programada para produzir fotografias, e é isso que a define. Cada fotografia produzida por uma câmera é uma realização das potencialidades programadas no aparelho. A indecifrabilidade do aparelho-caixa preta transforma a manipulação em um jogo¹⁴, no qual o fotógrafo experimenta fornecer diferentes inputs para o aparelho em busca de diferentes outputs. O fotógrafo pode comprar, possuir, e brincar com o aparelho, mas não pode programar o aparelho. O aparelho funciona de acordo com a programação (e os interesses) da fábrica. Dessa forma, quem tem poder sobre o aparelho não é quem o manipula, e sim quem o programa.

Embora Flusser concentre seu argumento sobre a fotografia (e os aparelhos que as produzem), é possível aplicar o que ele fala para outros aparelhos. Nesta dissertação, faz-se necessário traçar um paralelo da filosofia de Flusser com a computação. Em primeiro lugar, é preciso refletir qual a programação que define o computador. Assim como uma câmera é programada para produzir fotografias, uma televisão é programada para reproduzir imagens em movimento, e um carro é programado para promover deslocamentos. Observando a organização interna do computador, composto essencialmente por uma entidade de processamento de dados e por uma memória¹⁵, pode-se dizer que o computador é programado para processar informações, manipu-

uma fotografia como janela para o mundo e não como representação, está confundindo o decalque (a fotografia) com o mapa (o mundo).

¹⁴É interessante notar que Flusser utiliza o termo *homo ludens*, mas não cita o nome de Johan Huizinga, autor do livro *Homo ludens*. Originalmente publicado em 1938, o texto de Huizinga é uma referência clássica sobre o papel do jogo na formação da cultura e da sociedade.

¹⁵O cientista da computação John Von Neumann criou a chamada “arquitetura de Von Neumann”, uma divisão do computador em componentes que ainda hoje é utilizada. No esquema de Von Neumann, o computador recebe dados de entrada, processa esses dados, e gera dados de saída. A arquitetura interna do computador é composta de uma entidade chamada “unidade de processamento central” (*Central Processing Unit*, ou CPU, em inglês) e a memória. A CPU é subdividida em “unidade de controle” e “unidade aritmética e lógica”.

ladas e armazenadas em forma de bits. O computador em si não é programado para resolver um cálculo específico. Por meio de seu processador, o computador fornece uma estrutura que permite resolver qualquer cálculo de forma automatizada. O usuário de um computador pode formular uma sequência de instruções para que o computador execute um determinado cálculo. Pode-se dizer que o computador é um exemplo de meta-programa: o computador é programado para ser programado. Flusser diz que o número de possibilidades programadas em um aparelho é muito grande, mas não infinito. Ainda não se pode dizer com certeza se essa observação se aplica também aos computadores. O problema do que é e do que não é computável é um dos maiores problemas não-resolvidos pela matemática contemporânea. De qualquer maneira, o computador empodera seu usuário, pois permite que ele programe outros aparelhos (concretizados na forma de software).

No entanto, é necessário constatar que apenas uma fração da população consiste de cientistas da computação, que tem o conhecimento necessário para manipular bits. Para aqueles que não são cientistas da computação, computadores ainda são caixas pretas. O que permite que leigos utilizem computadores são os vários tipos de software, estes desenvolvidos por programadores. Quando um indivíduo redige um texto no Word, ele manipula o computador sem necessariamente compreender seu funcionamento interno. O software oferece interfaces gráficas que ocultam os processos internos da máquina para o usuário, o software se torna uma caixa preta. Se por um lado o software permite que leigos utilizem computadores, por outro lado o software limita o computador a um aparelho pré-programado novamente. Cada software é programado com uma finalidade específica: existem programas processadores de texto, programas de desenho, programas de comunicação, etc. Ao utilizar qualquer tipo de software, o usuário trabalha dentro das limitações impostas por esse software. E, assim como na fotografia, essas limitações nem sempre ficam claras para o operador do aparelho. Essa questão é ainda mais relevante quando a finalidade da manipulação do aparelho é uma atividade criativa. Flusser diz: “quem possui aparelho fotográfico de ‘último modelo’, pode fotografar ‘bem’ sem saber o que se passa no interior do aparelho” (FLUSSER, 2002, p.30). Ao utilizar aspas em volta da palavra “bem”, Flusser indica desconfiança com essa caracterização. De maneira análoga, poderia-se dizer que,

quem possui um computador de última geração pode utilizá-lo “bem” sem saber o que se passa em seu interior. Pode-se utilizar um computador para muitas finalidades: ler, escrever, comunicar-se, e até mesmo fazer design. Mas será que basta utilizar os recursos pré-programados nos programas de criação para projetar bem? Utilizar bem o computador e o software é navegar pelas possibilidades e limitações pré-inscritas pelos programadores do software. Não há relação direta com um bom projeto.

Um ponto-chave do texto de Flusser é quando diz: “toda crítica da imagem técnica deve visar o branqueamento dessa caixa (preta)” (FLUSSER, 2002, p.11). Enquanto não se abre a caixa preta, não é possível decifrar e compreender plenamente o objeto que essa caixa preta produz. O trabalho de Kay com o projeto Dynabook demonstra que a programação de computadores pode ser acessível até mesmo para crianças. E a pesquisa de Manovich indica que a relevância social, cultural, e econômica da produção de software tem se tornado cada vez maior. Talvez a caixa preta não se torne transparente para todos que utilizam o computador, mas é possível diminuir o grau de escuridão dessa caixa preta. Pode-se discutir que uma câmera é programável, pois o fotógrafo pode alterar sua lente ou seu filme, para obter outras cores ou ângulos. Mas, não é possível programar uma câmera fotográfica para gravar uma sequência contínua de imagens, um vídeo. Alterar a lente ou filme da câmera é alterar parâmetros do programa, não alterar o programa em si. No caso do computador, existem linguagens de programação dos mais diversos tipos e em diversos níveis de complexidade. A capacidade do computador de ser programado pelo seu usuário é a possibilidade de se abrir a caixa preta da computação.

A questão fundamental que permeia a *Filosofia da caixa preta* é: onde está o espaço para a liberdade, se aparelhos programam nossas vidas e automatizam o trabalho, transferindo o interesse do homem “do mundo objetivo para o mundo simbólico das informações”, no qual emoções ganham “caráter de jogo”, no qual a vida “passa a alimentar aparelhos e é por eles alimentado” (FLUSSER, 2002, p.40)?

Uma resposta para a questão levantada por Flusser pode ser encontrada no texto *Arte e mídia* de Arlindo Machado. Para Machado, a artemídia¹⁶ busca exatamente sub-

¹⁶“Artemídia” é um termo utilizado por Machado como tradução do inglês *media arts*, e engloba manifestações artísticas que utilizam, se apropriam, ou intervêm em tecnologias midiáticas e seus canais de difusão (MACHADO, 2007, p.7).

verter a programação de fábrica dos aparelhos. Se essa subversão é mais reconhecível nas artes, o que esta dissertação propõe é que o design pode também assumir uma posição questionadora em relação às tecnologias computacionais.

1.1.5 Por uma ruptura das caixas pretas

Machado observa que “a arte sempre foi produzida com os meios de seu tempo” (MACHADO, 2007, p.9). Enquanto Bach usou o cravo e o órgão, e Stockhausen usou o sintetizador, ambos buscavam explorar as possibilidades desses instrumentos musicais. Nos termos de Flusser, ambos artistas exploraram as possibilidades programadas nesses aparelhos. Da mesma forma, a artemídia explora as possibilidades inscritas nas mídias e tecnologias contemporâneas: fotografia, vídeo, e computador. No entanto, Machado argumenta que a artemídia busca não apenas explorar aquilo que está inscrito no aparelho, mas sim reinventar o aparelho, mencionando exemplos das obras de Nam June Paik, Frederic Fontenoy e Andrew Davidhazy, e William Gibson:

(...) não se pode mais, em nenhum desses exemplos, dizer que os artistas estão operando dentro das possibilidades programadas e previsíveis dos meios invocados. Eles estão, na verdade, ultrapassando os limites das máquinas semióticas e reinventando radicalmente os seus programas e suas finalidades. (MACHADO, 2007, p.14)

Desse modo, a artemídia se opõe a um determinismo tecnológico e propõe uma crítica das mídias e das tecnologias que subverte. A artemídia ilumina as contradições e limitações das mídias e vislumbra novas potencialidades para os aparelhos. Fazendo referência a Flusser, Machado argumenta que existe o perigo do artista se tornar apenas um operário que manipula um aparelho (MACHADO, 2007, p.48). Quando o artista apenas brinca com as possibilidades pré-programadas no aparelho, ele fica limitado a esse universo de possibilidades: o resultado é a **estereotipia**. Os artistas que utilizam os programas comerciais pré-fabricados trabalham dentro do mesmo universo de possibilidades. Dessa maneira, as obras que produzem ficam limitadas dentro desse universo comum. Quando o mesmo software é utilizado em massa por membros de uma comunidade, o resultado é que as obras produzidas por esses membros

começam a se tornar muito similares entre si. Machado observa essa “padronização” e “uniformidade generalizada” (MACHADO, 2007, p.54) em grande parte da produção artística recente produzida com o auxílio de sistemas computacionais.

Aplicando o argumento de Machado para o campo do design, é possível obter a mesma conclusão. Quando designers utilizam os mesmos programas de criação, trabalham dentro do mesmo universo de possibilidades. Que as soluções obtidas por esses designers sejam parecidas é simplesmente resultado direto da natureza caixa preta do software comercial. Uma alternativa para esse impasse é buscar outras formas de trabalho, seja no computador ou não. No entanto, se o computador já é um aparelho feito para ser programado, e a produção de software (programação) tem se tornado uma atividade cultural cada vez mais importante, o que esta dissertação propõe é que a utilização ampla dos conceitos da computação (sejam como conceitos auxiliares na resolução de problemas ou na forma de programação propriamente dita) é um caminho importante para a atividade de design nestes tempos contemporâneos.

Por fim, é interessante identificar um paralelo entre a filosofia de Flusser e a teoria de McLuhan. Para McLuhan, a aceitação inquestionada do impacto das mídias torna os homens prisioneiros dessas mídias, alheios aos efeitos pervasivos que as mídias provocam em suas vidas. O estudo de como as mídias funcionam permitem a compreensão das mudanças culturais e sociais introduzidas pelas mídias ¹⁷. De maneira similar, uma interpretação da filosofia de Flusser é a de que a aceitação inquestionada dos programas dos aparelhos torna os homens prisioneiros dos aparelhos, alheios aos limites do universo de potencialidades pré-programadas dentro do qual acabam adentrando. As caixas pretas transformam os homens em operários dos aparelhos, brincando dentro de um universo de possibilidades pré-programadas. Pode-se dizer que Flusser leva a discussão de McLuhan para um nível extremamente profundo, quando afirma que a filosofia da caixa preta traz reflexões a respeito da própria liberdade do homem. Estudar computação em design não representa ferir uma suposta “pureza” da atividade de design. Estudar computação em design é simplesmente reconhecer o

¹⁷ McLuhan cita o exemplo do pensador francês Alexis de Tocqueville, que compreendeu a Revolução Francesa como resultado da disseminação da tipografia e da palavra impressa na França. Um aristocrata altamente letrado, de Tocqueville conseguiu se distanciar dos efeitos imediatos da mídia tipografia e enxergar suas consequências sociais (MCLUHAN, 1964).

importante efeito que essa mídia tem na sociedade e nos indivíduos, e é o primeiro passo para um questionamento crítico e uma potencial libertação.

Nesta primeira etapa do primeiro capítulo, foi discutida a importância da computação como mídia na sociedade. Compreende-se que o estudo crítico dessa mídia é uma etapa importante para uma reflexão sobre a liberdade no mundo contemporâneo. A partir desse contexto geral, a segunda parte do primeiro capítulo explora um aspecto dos efeitos da computação na sociedade: os efeitos da computação no pensamento humano.

1.2 Computação como artefato mental

A consideração da computação como artefato mental se fundamenta em trabalhos de cientistas da computação como Licklider (1960), Engelbart (1962), Bush (1945), e Kay (1977), que ajudaram a construir a história da computação pessoal. Outra autora importante que relacionou computação e pensamento é Wing (2006), com sua pesquisa sobre o chamado “pensamento computacional”.

Inicialmente, a computação foi desenvolvida como uma forma de realizar cálculos, processar dados, e automatizar tarefas repetitivas. Computadores eram máquinas de grande porte, de difícil operação, restritas a empresas ou universidades. Mas ao longo de sua história, a computação deixou de ser apenas uma ferramenta matemática, e vem contribuindo para diversas áreas do conhecimento. Computadores têm se tornado cada vez menores, mais fáceis de serem utilizados, mais acessíveis. E a crescente presença da computação no ambiente têm tornado mais visíveis seus efeitos na sociedade e no homem. Essa difusão da computação se deu graças ao desenvolvimento de interfaces amigáveis que permitem que cada vez mais pessoas utilizem a computação para seus fins. Por trás do desenvolvimento dessas interfaces, está uma visão da computação como elemento participativo nos processos intelectuais do homem. Diversos cientistas da computação contribuíram para a construção dessa visão, mas nesta dissertação são discutidos os trabalhos de J.C.R. Licklider, Douglas Engelbart, Vannevar Bush, e Alan Kay, que abordaram as relações entre computação e pensamento.

Um comentarista essencial dos trabalhos destes cientistas da computação é o es-

critor Howard Rheingold, autor do livro *Tools for thought* (1984). Publicado na década de 80, em meio ao processo de popularização dos computadores pessoais, *Tools for thought* apresenta uma história da computação menos baseada em seus aspectos tecnológicos e mais voltada para as ideias e visões das pessoas que trabalharam em seu desenvolvimento. Dessa forma, pode ser considerado um precursor de livros como *Multimedia: from Wagner to multimedia* (PACKER; JORDAN, 2001) e de pesquisas como as de Lev Manovich. No livro, Rheingold narra a história da computação pessoal a partir das trajetórias pessoais de figuras notáveis da computação. Esses personagens são divididos em 3 grupos, de acordo com a época em que atuaram: **patriarcas** (que atuaram até a primeira metade do século XX)¹⁸, **pioneiros** (que começaram a atuar por volta da metade do século XX), e **infonautas** (a nova geração que despontava na década de 80). Os pesquisadores abordados nesta dissertação pertencem ao grupo dos pioneiros da computação. Assim como a importância da leitura de McLuhan é reconhecida por Kay, a importância da leitura de Rheingold é reconhecida por Manovich. Para Manovich, o livro de Rheingold lhe permitiu compreender a computação não apenas como tecnologia, mas como mídia intelectual e imaginativa (MANOVICH, 2013, p.331).

O primeiro pesquisador apresentado aqui é J.C.R. Licklider, que era originalmente pesquisador especializado em psicoacústica, mas que passou a ter maior envolvimento com a computação ao tentar construir modelos matemáticos para compreender a percepção dos sons pelo cérebro humano.

1.2.1 Simbiose homem-computador

Licklider tinha como objetivo simplificar sua pesquisa construindo modelos matemáticos, mas percebeu que seus modelos haviam se tornado tão complexos que tomavam a maior do seu tempo de pesquisa. Ao estudar o seu próprio processo de pesquisa, Licklider observou que a maior parcela do seu tempo era dedicada à tarefas automatizáveis, como cálculos, conversão de dados, e plotagem de gráficos, e que apenas uma

¹⁸Os patriarcas citados por Rheingold contribuíram com conceitos fundamentais da computação, mas não são discutidos neste capítulo porque seus trabalhos estão mais relacionados com a matemática e a lógica. Talvez a exceção seja o trabalho de Norbert Wiener, que buscou uma abordagem interdisciplinar em sua teoria da cibernética. A definição de Wiener para feedback é apresentada nesta dissertação na discussão do conceito de meta-mídia de Alan Kay.

parcela pequena do tempo era dedicada ao pensar em si. Para lidar com esse problema, Licklider propôs uma **relação simbiótica** entre homem e computador, que apresentou em seu artigo *Man Computer Symbiosis* (1960). A base dessa relação simbiótica seria uma comunicação homem-máquina em tempo-real, com feedbacks imediatos, que permitiria aos computadores participarem do processo de pensar.

Uma relação simbiótica é aquela na qual dois organismos vivos de espécies diferentes convivem em interdependência, em uma relação mutuamente benéfica. O que Licklider propôs na simbiose homem-computador é uma relação mutuamente benéfica entre homens e computadores, na qual os dois trabalham em conjunto. Computadores realizam as tarefas para as quais são mais adequadas e as tarefas que são impossíveis para o homem. Da mesma forma, os humanos realizam as tarefas para as quais estão mais aptos, e as tarefas que são impossíveis para o computador.

A proposta de simbiose homem-computador de Licklider apresentava 2 objetivos. O primeiro objetivo era fazer com que computadores participassem do processo de formulação de problemas, e não só da resolução de problemas. Ou seja, que computadores pudessem apontar erros de raciocínio ou iluminar informações inesperadas durante o processo de pensamento. Licklider não estava interessado apenas no computador como processador de dados (que realiza operações em cima de um banco de dados existente), mas também como modelador de dados, que cria e simula modelos. O segundo objetivo, que surgia em função do primeiro objetivo, era tornar a computação um fenômeno em **tempo-real**, permitindo que homens e computadores tomassem decisões complexas de maneira flexível, independentes de programas pré-determinados. Na década de 60, computadores ainda eram programados por meio de cartões perfurados, o que tornava a programação um processo lento. A partir da formulação de um dado problema, a construção de um programa para a resolução desse problema poderia tomar vários dias de trabalho de um profissional programador. Mas Licklider imaginou que, se a computação fosse um fenômeno em tempo-real, o homem poderia pensar **interativamente** com o computador, simbioticamente, como se fosse um colega com quem discute para solucionar um problema. O resultado dessa simbiose seria o avanço do próprio pensamento humano:

A esperança é que, em pouco tempo, cérebros humanos e computadores

serão fortemente acoplados, e que **a parceria resultante pensará como nenhum ser humano já pensou** e processará dados de uma forma não alcançada pelas máquinas de informação que conhecemos hoje. (LICKLIDER, 1960, p.57, tradução e grifo nosso)¹⁹

Licklider teve um papel importante na diretoria da Agência de Projetos de Pesquisa Avançada de Defesa americana ARPA (*Defense Advanced Research Projects Agency*) durante a década de 60. Seu papel na ARPA não era o de pesquisador, mas o de procurar e apoiar projetos que fossem importantes para o avanço tecnológico. A ARPA surgiu como resposta dos EUA ao lançamento do satélite artificial Sputnik pela ex-União Soviética. Acerca desse contexto, Rheingold comenta que o desenvolvimento de computadores menores, mais rápidos, e precisos havia se tornado uma necessidade. A simbiose homem-computador deixava de ser uma “proposta esotérica” para se tornar um “objetivo nacional” (RHEINGOLD, 1984). A atuação de Licklider na ARPA ainda levaria ao desenvolvimento da Arpanet, precursora da atual internet.

Licklider enxergou no computador o potencial para se tornar uma entidade codependente do homem apta a auxiliar no próprio processo de pensar do ser humano. Um dos pesquisadores que teve sua pesquisa patrocinada pela ARPA durante o período de Licklider foi Douglas Engelbart. Engelbart desenvolveu uma série de dispositivos e interfaces que hoje são elementos comuns na computação pessoal moderna: o mouse, as interfaces gráficas com janelas e menus, o email, e o processador de texto (PAC-KER; JORDAN, 2001, p.65). A demonstração de seu *oNLine System* (NLS) em 1968, na qual demonstrou protótipos de sistemas de teleconferência, é conhecida até hoje como *the mother of all demos* (MONTFORT et al, 2012, p.86). Engelbart buscava aproximar a computação do pensamento humano, o que resultaria na chamada **amplificação do intelecto humano**.

¹⁹The hope is that, in not too many years, human brains and computers will be coupled together very tightly, and that the resulting partnership will think as no human being has ever thought and process data in a way not approached by the information-handling machines we know today (LICKLIDER, 1960, p.57).

1.2.2 Amplificação do intelecto humano

Engelbart apresentou sua pesquisa sobre a “amplificação do intelecto humano” no artigo *Augmenting Human Intellect: A Conceptual Framework* (1962), no qual trata da ampliação da capacidade do ser humano de lidar com problemas complexos por meio de sistemas computacionais, com melhor e mais rápida compreensão dos problemas, com a obtenção de soluções melhores e mais rápidas, e com a possibilidade de obter soluções para problemas considerados insolúveis (ENGELBART, 1962, p.66). Quando se referia a problemas complexos, Engelbart se referia a problemas de qualquer área do conhecimento, e não a uma área específica. Essa realização de que a computação poderia contribuir para qualquer área de conhecimento encontra ecos nos trabalhos de Manovich (quando este diz que a computação não deve ser um assunto limitado aos departamentos de ciências de computação das universidades) e outros pesquisadores contemporâneos, como Wing e Denning, que serão discutidos mais adiante. Engelbart vislumbrou um modo de vida no qual a intuição humana coexistiria com as tecnologias. Percebe-se uma similaridade com o pensamento de Licklider, que tinha proposto uma relação na qual as capacidades do homem e as capacidades do computador seriam exploradas para o benefício mútuo, como numa relação simbiótica entre dois seres vivos. Para Engelbart, a capacidade do computador não se limitaria à execução de cálculos matemáticos, podendo ser utilizado também para manipular e visualizar informações, de forma significativa para processos de planejamento, organização, e estudo. Neste ponto, exhibe outro paralelo com o pensamento de Licklider, que considerava de grande importância a capacidade do computador de transformar informações e gerar visualizações gráficas dentro de um processo de pensamento e pesquisa científica.

Engelbart observou que a inteligência está distribuída em uma hierarquia de processos, e que ela depende dessa organização. A inteligência está distribuída na superfície da tela do computador, nas estruturas do programa de computador, no hardware do computador, nos circuitos e seus elementos (transistores, resistores, capacitores, e diodos), nas partículas subatômicas que se movimentam por esses circuitos. A partir dessa observação, Engelbart concluiu que a inteligência resulta da **sinergia** entre esses processos, no sentido que o todo é maior que a soma das partes. O que Engelbart

entendia como “amplificação da inteligência” não era exatamente o aumento da inteligência humana em si, mas a criação de uma superestrutura sinérgica e artificial na qual as capacidades do homem e do computador estariam organizadas para resultar em uma entidade mais inteligente.

Em *Augmenting human intellect*, Engelbart cita e comenta extensivamente o artigo *As we may think* (1945) de Vannevar Bush, que foi conselheiro científico do presidente americano Franklin Roosevelt e também inventor do “analisador diferencial”, considerado um dos primeiros computadores eletrônicos automáticos (PACKER; JORDAN, 2001, p.142). Publicado na revista *Atlantic Monthly* ao fim da Segunda Guerra Mundial, seu artigo é considerado altamente influencial na história da computação.

1.2.3 Vannevar Bush

Com o fim da guerra, a comunidade científica começou a reconsiderar os caminhos do desenvolvimento tecnológico e científico, buscando aplicar as tecnologias desenvolvidas na guerra para a paz (PACKER; JORDAN, 2001, p.142). Bush fala que a guerra afastou muitos cientistas de seus interesses acadêmicos, e que esses cientistas acabam envolvidos no desenvolvimento de tecnologias destrutivas. O fim da guerra seria uma oportunidade para que esses cientistas buscassem novos objetivos (BUSH, 1945, p.143).

Para Bush, o armazenamento e o acesso de informações seriam os problemas fundamentais para o desenvolvimento das ciências no pós-guerra. Ele constatou que a quantidade de pesquisas e descobertas científicas crescia em um ritmo que dificultava a leitura e a memorização de todas essas informações. Observou que essa dificuldade resultava da inabilidade humana de absorver a quantidade de informações produzidas, e viu nas possibilidades de automatização por máquinas uma forma de lidar com este problema. Assim, ele propôs um dispositivo computacional chamado **memex**, que funcionaria como uma extensão da memória humana, ajudando o homem a trabalhar com esse grande volume de informações.

Em seu artigo, Bush vislumbrou novas formas de armazenamento e registro de informações possibilitadas pelos avanços tecnológicos. Em relação ao armazenamento de informações, ele considerava importante a possibilidade de compressão da infor-

mação, que permitiria comprimir uma enciclopédia inteira para as dimensões de uma caixa de fósforos. Em relação ao registro de informações, ele imaginou que o registro por meio de reconhecimento de voz deixaria os pesquisadores com as mãos livres, o que lhes permitiria andar pela sala sem estar preso à frente de um teclado para digitar. Por fim, a utilização da automação de tarefas repetitivas forneceria aos pesquisadores dados que fomentariam seus pensamentos criativos. A relação entre automação da máquina e pensamento criativo encontra ecos nos artigos de Licklider, de Engelbart, e de Kay. Bush também sugeriu a necessidade de desenvolver melhores formas de consultar informação, extraindo dados, de acordo com algum critério de “seleção” (termo utilizado no sentido de “busca”). Ele observou que um sistema de busca baseado em um sistema de indexação da informação, sistemas baseados em estruturas hierárquicas nas quais cada informação possui apenas uma localização possível, não condiz com a maneira como a mente humana opera. E argumentava que a mente humana não opera de maneira linear e hierárquica, e sim por meio de “associações de pensamentos, de acordo com intrincadas redes de trilhas carregadas pelas células do cérebro” (BUSH, 1945, p.155). Mas a memória humana é transitória e não-permanente, por isso as trilhas menos utilizadas desaparecem gradualmente com o tempo. A máquina poderia ser utilizada para armazenar essas trilhas de forma permanente.

A partir dessas ideias, Bush propôs um dispositivo chamado **memex**, que descreveu como um tipo de “arquivo pessoal e biblioteca mecanizado” (BUSH, 1945, p.155), em que uma pessoa armazenaria seus livros, anotações, e comunicações. Com o memex, essas informações poderiam ser consultadas de forma rápida e eficiente, por meio de associações mecanizadas. O memex seria uma extensão da memória do usuário. Os registros seriam armazenados em microfilmes, e seus conteúdos seriam projetados em uma tela translúcida. Por meio de teclados e alavancas, o usuário desse sistema poderia navegar entre os documentos, tanto por meio do sistema de indexação, quanto por meio de um sistema de associações construídas pelo próprio usuário. Ao contrário da memória humana, essas trilhas não desapareceriam com o tempo, de forma que o usuário do memex poderia retomar associações de pensamento após muitos anos do registro inicial. Dessa maneira, as associações entre diferentes textos se tornariam tão importantes quanto os textos em si.

Com seu artigo, Bush tratou da importância dos cientistas trabalharem no problema da forma de acessar, registrar, e interagir com informações, abrindo caminho para o desenvolvimento dos conceitos de hipermídia e hipertexto, desenvolvidos posteriormente por Ted Nelson (em suas obras *Dream Machines* e *Computer Lib*, e em sua proposta do sistema *Xanadu*) e por Tim Berners-Lee (em sua proposta de *World Wide Web*). A ideia de que uma interface computacional poderia servir como extensão da memória humana e alimentar pensamentos criativos foi a grande contribuição de Bush com seu memex. Além de ter influenciado Engelbart, ecos do memex de Bush podem ser encontrados no Dynabook de Kay, que não só pensou no conceito de metamídia, mas também na importância dessa meta-mídia como “amplificadora de fantasias” (RHEINGOLD, 1984).

1.2.4 Amplificação de fantasia

Como já foi mencionado anteriormente neste capítulo, o Dynabook era o projeto de Kay para um computador pessoal que combinaria a possibilidade de simular outras mídias e possibilidade de se inventar novas mídias. Mas é importante notar que Kay não desenvolveu o conceito de meta-mídia apenas como uma classificação teórica. A conceituação de meta-mídia foi um passo de Kay no processo de desenvolvimento de uma visão da computação pessoal como amplificadora do pensamento humano, especialmente em seus aspectos criativos.

E para Rheingold, o Dynabook de Kay era não apenas um amplificador intelectual, mas também um amplificador de fantasias. A meta-mídia se caracteriza pela possibilidade de reproduzir outras mídias (as existentes e as ainda não existentes) e pela capacidade de responder ativamente às interações de seu usuário. Dessa forma, a meta-mídia pode ser programada para simular outras mídias e ideias. Enquanto Kay dizia que “simulação é a noção central do Dynabook” (KAY; GOLDBERG, 1977, p.399), Rheingold fala que simular é “o poder de ver o que você imagina” (RHEINGOLD, 1984). Portanto, a meta-mídia seria um meio de criar (ou simular) universos imaginários. E ao responder ativamente ao usuário, a meta-mídia fornece novas informações ao usuário que o programou, “amplificando” suas fantasias. A importância da simulação no trabalho de Kay está relacionada com sua leitura das teorias do aprendizado de Piaget.

Segundo Rheingold, Piaget havia concluído que “crianças são cientistas – elas realizam experimentos, formulam teorias, e testam suas teorias com mais experimentos” (RHEINGOLD, 1984)²⁰. Ou seja, para as crianças, brincar é uma forma de pesquisar. Da mesma maneira, para Kay a simulação seria uma forma de amplificação do poder de aprendizado.

Para compreender a contribuição de Kay para o entendimento da computação como artefato mental, é interessante retomar seu comentário a respeito da leitura de *Understanding media* de McLuhan:

Li *Understanding media* (1964) e entendi que a coisa mais importante em qualquer meio de comunicação é que recepção da mensagem é na verdade recuperação da mensagem: qualquer um que deseja receber uma mensagem embutida em uma mídia precisa primeiro ter internalizado a mídia para que possa “subtraí-la” e ficar com a mensagem. Quando ele disse que “o meio é a mensagem”, ele quis dizer que você precisa se tornar a mídia se você a utiliza. (KAY, 1989, p.124)²¹

Ou seja, para Kay, quem utiliza uma mídia precisa incorporá-la. Embora não se possa tomar essa frase de forma literal, o que pode se compreender é que a utilização de uma mídia implica em uma internalização mental da mídia²². Essa constatação fez

²⁰children are scientists – they perform experiments, formulate theories, and test their theories with more experiments (RHEINGOLD, 1984).

²¹I read McLuhan’s *Understanding Media* (1964) and understood that the most important thing about any communications medium is that message receipt is really message recovery: anyone who wishes to receive a message embedded in a medium must first have internalized the medium so it can be “subtracted” out to leave the message behind. When he said “the medium is the message” he meant that you have to become the medium if you use it (KAY, 1989, p.124).

²²Aqui é interessante mencionar a pesquisa da psicóloga Sherry Turkle a respeito dos aspectos psicológicos da utilização do computador pessoal. Se Kay dizia que é preciso incorporar uma mídia para utilizá-la, Turkle constatou em suas pesquisas na década de 80 que alguns usuários de computadores identificavam suas próprias mentes com computadores, utilizando termos de informática como analogias para seus estados mentais. Turkle observou que “pessoas estão pensando sobre si mesmas em termos computacionais” (TURKLE, 2005, p. 22), usando termos como *hardwired* (quando uma funcionalidade está incorporada permanentemente nos circuitos do computador, e não pode ser alterada), *buffer* (espaço de memória em que a informação é armazenada temporariamente e depois esvaziada), e *debugging* (o ato de vasculhar um programa em busca das origens de um erro) para se referirem às próprias memórias ou aos próprios comportamentos. Para Turkle, essas analogias não são apenas figuras de linguagem, e carregam uma percepção implícita de que os processos mentais internos são equivalentes

Kay estudar os trabalhos de diversos psicólogos (como Jean Piaget e Jerome Bruner). Antes do desenvolvimento do Dynabook, Kay estava envolvido no projeto de um computador chamado FLEX. Rheingold relata que foi nesse projeto que Kay percebeu que era necessário não apenas projetar a máquina, mas também considerar a linguagem que o usuário utilizaria para fornecer os comandos para a máquina. Kay constatou que o projeto da máquina deveria ser feito de acordo com seu futuro usuário, e essa constatação foi um dos aspectos fundamentais no desenvolvimento da linguagem de programação Smalltalk para o Dynabook, além de ser uma ideia que permeia o próprio projeto do Dynabook e os textos escritos por Kay.

Ao relacionar computação e psicologia, Kay tornou clara a relação entre computação e pensamento que já havia sido explorada por Licklider, Engelbart, e Bush. Os trabalhos desses pesquisadores foram centrais para o progresso da computação pessoal. A influência de seus trabalhos é visível nos computadores, smartphones, e tablets disponíveis atualmente. Seus trabalhos foram guiados por um entendimento de que a computação poderia ser muito mais que uma máquina automatizadora de tarefas ou uma máquina simuladora de mídias analógicas. Rheingold observa que esses pesquisadores estavam preocupados em tirar proveito das capacidades dos computadores para “ampliar o poder de seus próprios cérebros”, utilizando o computador para auxiliar nos “aspectos mais mecânicos do pensamento” (RHEINGOLD, 1984). Para eles, o objetivo do computador não era apenas automatizar tarefas, e sim “melhorar os mais criativos aspectos da inteligência humana – para todos, não apenas para uma elite tecnológica” (RHEINGOLD, 1984)²³. Ou seja, o computador poderia auxiliar as mentes humanas em problemas de diversas naturezas, e não apenas os problemas matemáticos. Para eles, a computação poderia se tornar um elemento essencial para avançar o pensamento humano, em qualquer área do conhecimento. Uma contribuição mais contemporânea para esta ideia é o trabalho da cientista da computação Jeanette Wing, que cunhou o termo **pensamento computacional** em 2006.

aos processos computacionais. Como consequência, essas pessoas lidam com computadores como se estivessem lidando com uma mente simplificada.

²³enhance the most creative aspects of human intelligence—for everybody, not just the technocognoscenti (RHEINGOLD, 1984).

1.2.5 Pensamento computacional

Wing define pensamento computacional como a utilização de conceitos fundamentais da computação para “resolver problemas, projetar sistemas, e compreender comportamentos humanos” (WING, 2006, p.33). Wing acrescenta que o pensamento computacional se baseia em processos computacionais, que podem ser executados tanto por um computador quanto por uma pessoa. Se os pioneiros da computação já haviam reconhecido os efeitos da computação no pensamento humano, Wing constata que não é necessário utilizar um computador para se pensar computacionalmente. Os pioneiros da computação trabalharam em um contexto no qual a computação pessoal estava começando a tomar forma. Suas pesquisas buscavam exatamente contribuir para a construção de uma computação pessoal, e portanto envolviam propostas de computadores e interfaces que ajudariam o homem em seus processos mentais. Para eles, a amplificação do intelecto humano passava pela utilização de computadores (ou sistemas computacionais). Atualmente, a computação pessoal está estabelecida, e se difunde cada vez mais na sociedade e no ambiente. Neste novo contexto de crescente ubiquidade de dispositivos computacionais, Wing percebe que a computação pode ser aplicada no pensamento humano, independente da utilização ou não de computadores.

A constatação de Wing implica em uma distinção entre computador (tecnologia) e computação (princípio). Essa distinção é clara na pesquisa do cientista da computação Peter Denning (2010). Para Denning, a computação é mais do que o estudo de linguagens de programação, sistemas operacionais, redes, banco de dados, etc. Essas tecnologias envolvem a computação, mas não são suficientes para definir a computação. Denning buscou uma definição menos tecnológica da computação, e para isso identificou os princípios fundamentais na computação, aqueles que caracterizam as tecnologias computacionais em suas possibilidades e suas limitações. Em seguida, agrupou os princípios que identificou em 7 grandes categorias. Por exemplo, a categoria da **comunicação** trata da “movimentação confiável de informação entre locais”, e a categoria da **avaliação** lida com a “previsão da performance de sistemas complexos” (DENNING, 2010, p.370). Esses princípios independem de implementações específicas de alguma tecnologia. Outro fator que motivou a pesquisa de Denning foi o

reconhecimento das interações entre a computação e outras áreas do conhecimento. A computação é influenciada e influencia fenômenos físicos, sociais, e biológicos. A construção de um quadro conceitual da computação que independe das tecnologias computacionais facilitaria a identificação de paralelos entre a computação e as outras áreas. Desse modo, o quadro de Denning permite compreender a computação como “um campo amplo que estuda processos de informação, naturais e artificiais” (DENNING, 2010, p.372). A pesquisa de Denning contribui para o pensamento computacional de Wing ao reconhecer que a computação não está apenas no computador, e que a computação interage ativamente com outras áreas do conhecimento²⁴.

Para Wing, “a essência do pensamento computacional é a abstração” (WING, 2008, p.3717). Abstração pode ser qualquer modelo ou representação que destaque os aspectos mais importantes de um problema, facilitando sua resolução. Por exemplo, números são abstrações que permitem realizar operações matemáticas com mais facilidade do que contando objetos. Outros exemplos de abstrações utilizados pela computação para lidar com informações são as pilhas (elementos empilhados um em cima do outro), as filas (elementos enfileirados um atrás do outro), e as árvores (hierarquias nas quais um nó dá origem a vários galhos). E as abstrações podem ser ainda mais complexas. Uma linguagem de programação é uma abstração de uma linguagem natural, utilizada para fornecer comandos a um computador. Sendo assim, Wing considera as abstrações como “as ferramentas ‘mentais’ da computação” (WING, 2008, p.3718). Escolher as abstrações mais adequadas para o problema em questão é uma das etapas essenciais do pensamento computacional.

Problemas complexos exigem vários níveis de abstrações, ou seja, abstrações em cima de abstrações. Abstrações de “baixo nível” são aquelas que lidam com os aspectos mais específicos de um problema, enquanto abstrações de “alto nível” são aquelas mais próximas da visão geral do problema. Dessa maneira, pensar computacionalmente envolve definir abstrações, identificar os níveis de abstrações, e entender as re-

²⁴Interessantemente, Denning publicou um artigo intitulado *Beyond computational thinking* (2009), no qual observa que o termo “pensamento computacional” poderia representar uma simplificação exagerada da ciência da computação. É uma discussão mais voltada para os cientistas da computação, já que trata da própria definição do que é a “ciência da computação”. Denning entende que o pensamento computacional não é bem um princípio geral que explica a computação, e sim uma das várias práticas fundamentais na ciência da computação.

lações existentes entre esses níveis. Como observa Wing, pensar computacionalmente “significa mais do que ser capaz de programar um computador. Requer pensar em múltiplos níveis de abstração” (WING, 2006, p.35) ²⁵. Ou seja, o pensamento computacional está mais relacionado com a utilização de abstrações do que à capacidade de um indivíduo de programar computadores. Afinal, é possível programar computadores sem utilizar abstrações e padrões. Linguagens de programação simples, como Basic, não possuem recursos que permitem a aplicação de abstrações em seus programas, ou possuem esses recursos de forma limitada. Essas linguagens simplificadas são conhecidas como linguagens de scripting, sendo que scripts consistem em programas curtos e simples. Geralmente os scripts são utilizados para automatizar tarefas simples. Devido à sua própria brevidade e simplicidade, os scripts não exigem a aplicação de abstrações complexas.

No entanto, o pensamento computacional compreende outros recursos além das abstrações, como por exemplo:

- decomposição (subdividir um problema em partes menores),
- coleta, análise, e representação de dados (obter e interpretar dados, identificando padrões e obtendo conclusões),
- algoritmos (sequências de instruções para resolver um problema),
- automação (utilização de máquinas para tarefas repetitivas),
- simulação (experimentação por meio de modelos),
- paralelização (organização de recursos para a execução de tarefas em paralelo com um objetivo em comum).

Esta lista é um resumo do “vocabulário do pensamento computacional” (COMPUTER SCIENCE TEACHERS ASSOCIATION; INTERNATIONAL SOCIETY FOR TECHNOLOGY IN EDUCATION, 2011, p.8-9) proposto pela *Computer Science Teachers Association* (CSTA) e pela *International Society for Technology in Education* (ISTE) em documento que foi publicado em 2011 e que tem como objetivo introduzir o pensamento computacional para professores de ensino fundamental e médio. Essa parceria entre CSTA

²⁵Thinking like a computer scientist means more than being able to program a computer. It requires thinking at multiple levels of abstraction (WING, 2006, p.35).

e ISTE reflete uma crescente valorização do pensamento computacional como habilidade importante no ensino fundamental e no ensino médio nos EUA. O Google também mantém um site dedicado à promoção do pensamento computacional no ensino fundamental e médio. O documento da CSTA e da ISTE, e o site do Google apresentam o pensamento computacional de forma mais simplificada, e trazem exemplos e ideias para sua aplicação em atividades escolares. Alguns exemplos de pensamento computacional, aplicados fora do próprio campo da computação, incluem (GOOGLE, 2011):

- quando se fornece instruções para alguém chegar em algum lugar, o problema do deslocamento é **decomposto** em etapas,
- o ato de comprar e vender ações na bolsa de valores envolve a **análise de dados** e o **reconhecimento de padrões**,
- uma agenda é uma **abstração** dos dias da semana, que se utiliza para planejar as atividades ao longo do tempo,
- quando um cozinheiro cria uma receita para um prato, ele está criando um **algoritmo** que permite a outras pessoas reproduzir o mesmo prato,
- quando um técnico de futebol treina jogadas com seu time, ele está criando **algoritmos** que os jogadores devem cumprir durante o jogo.

Mas é necessário fazer alguns esclarecimentos em relação ao pensamento computacional. Em primeiro lugar, a utilização da ferramenta computacional (computador, máquina) não necessariamente implica em compreensão dos conceitos computacionais. Como diz Wing, “nós não queremos que as pessoas pensem que elas entendem os conceitos porque elas são competentes no uso da ferramenta” (WING, 2008, p.3721)²⁶. Ou seja, o ato de utilizar uma ferramenta computacional não necessariamente exige o ato de pensar computacionalmente. É possível dominar um software e mesmo assim não pensar computacionalmente. Por outro lado, também é possível pensar computacionalmente sem possuir domínio de software. Essa diferenciação é

²⁶(...) we do not want people to come away thinking they understand the concepts because they are adept at using the tool (WING, 2008, p.3721).

essencial para que não se confunda domínio técnico com compreensão conceitual.²⁷ Em segundo lugar, o estudo dos princípios computacionais não representa uma tentativa de fazer com que humanos pensem como máquinas, mas sim de fazer com que o potencial computacional disponível seja melhor compreendido e utilizado pela criatividade humana. Segundo Wing, “pensamento computacional é uma forma como humanos resolvem problemas; não é tentar fazer com que humanos pensem como computadores” (WING, 2006, p.35)²⁸. A computação fornece conceitos e elementos para resolução de problemas, mas é a criatividade humana que permite utilizar e explorar tais conceitos na prática. Em conjunto com a capacidade de processamento numérico de computadores, a criatividade humana e os conceitos computacionais têm permitido a exploração de problemas de novas espécies e em escalas que não eram viáveis antes do computador.

Ainda para discutir o pensamento computacional de Wing, é interessante fazer referência à distinção entre os termos **ferramenta** e **instrumento** proposta pelo pesquisador, arquiteto, e designer Caio Vassão. Para Vassão, instrumento é um “aparato distinto e explicitamente desacoplável” (VASSÃO, 2008, p.284). Ou seja, um instrumento não se envolve com o processo; é uma entidade claramente separada e substituível. Já a ferramenta “se entranha em nós, e nós nela” (VASSÃO, 2008, p.284). Uma ferramenta se envolve completamente com o processo, e se confunde com quem a utiliza. Antes de uma tecnologia se tornar “ferramenta”, ela é “instrumento”. A distinção entre “ferramenta” e “instrumento” não está inscrita nas tecnologias, ela resulta da postura do homem em relação às tecnologias: “tudo depende de como nos permitimos acoplar” (VASSÃO, 2008, p.284). Um martelo é “instrumento” quando utilizado para bater um prego em uma parede, e é “ferramenta” quando utilizado por um carpinteiro para construir um móvel. O carpinteiro utiliza diariamente o martelo, e

²⁷Em anúncios de vagas de emprego, é comum que se exija o domínio técnico de determinado software comercial. Essa exigência reflete uma certa confusão que ainda existe a respeito da computação. O domínio da computação não deveria passar pelo domínio de algum software específico, mas no domínio geral da ferramenta. Aquele que domina a computação tem condições de determinar o software mais adequado para a tarefa que precisa executar, e não busca resolver todos os seus problemas usando sempre o mesmo software. Retomando as reflexões de Flusser, o usuário pensa que domina o software, mas é o software que domina o usuário.

²⁸Computational thinking is a way humans solve problems; it is not trying to get humans to think like computers (WING, 2006, p.35).

conhece a fundo suas características, conhecendo diversas técnicas de utilização para obter diversos efeitos. O martelo é ferramenta essencial no trabalho do carpinteiro. Da mesma forma, a computação é “instrumento” quando utilizado como simulador de alguma outra mídia. Por exemplo, se o trabalho realizado pela computação pode ser substituído por uma máquina de escrever, ela é “instrumento”. Por outro lado, a computação é “ferramenta” quando não se pode substituí-la facilmente por outra mídia. Aquele que conhece os conceitos computacionais e desenvolve diversas formas de utilizar a computação para resolver seus problemas, utiliza a computação como “ferramenta”. O pensamento computacional propõe que a computação utilizada como “ferramenta” não deve ser uma habilidade exclusiva de cientistas da computação.

Enquanto o pensamento computacional busca aplicar conceitos da computação no pensamento humano, a inteligência artificial procura o inverso, que é aplicar conhecimentos sobre o funcionamento do cérebro humano na computação. O pensamento computacional busca mostrar que o pensamento humano pode emprestar conceitos da computação para criar novas soluções para problemas. Já as iniciativas de inteligência artificial procuram fazer com que computadores atuem como mentes humanas. Embora a inteligência artificial seja um campo relevante para a computação, ela geralmente é utilizada para resolver problemas pontuais e específicos (como por exemplo no reconhecimento de padrões em imagens, ou em filtros anti-spam), ao invés de contribuir em processos dinâmicos como os processos de design. Dessa maneira, a inteligência artificial (incluindo temas relacionados como vida artificial e redes neurais) não é discutida com maior profundidade neste trabalho.²⁹

Wing identifica que princípios computacionais já são utilizados e contribuem para outras áreas do conhecimento, como a estatística, a biologia, e a economia. A computação auxilia no desenvolvimento de novas abstrações e modelos³⁰, contribuindo para a obtenção de resultados para problemas complexos. Por exemplo, a utilização

²⁹Um dos pioneiros da inteligência artificial, o cientista cognitivo Marvin Minsky se questionou: “como pode uma inteligência emergir a partir de uma não-inteligência?” (MINSKY, 1988, p.17). Para Minsky, uma mente é constituída de uma partes simples, chamadas agentes, que não possuem capacidade de pensar por si próprios, mas que quando interagem de formas específicas, resultam em inteligência. Minsky identificou características de máquinas no cérebro humano em seu livro *The society of mind* (1988).

³⁰Essa aplicação da computação remete à trajetória de Licklider, que se interessou pela computação para desenvolver e testar modelos matemáticos em sua pesquisa sobre percepção do som.

de modelos computacionais em estatística tem permitido a identificação de padrões e anomalias em grandes conjuntos de dados, com aplicações práticas em campos diversos como astronomia, medicina e sistemas bancários. O pensamento computacional na biologia tem acelerado o sequenciamento do genoma humano, além de permitir a compreensão de outros fenômenos da natureza (WING, 2008, p.3719).

1.2.6 Para além da computação

Na segunda etapa deste primeiro capítulo, foi discutida a importância da computação como artefato mental, que participa ativamente dos processos do pensamento humano. As ideias de Licklider, Engelbart, Bush, e Kay contribuíram para a construção de uma visão da computação como amplificadora do intelecto humano. A grande contribuição dessas figuras centrais na história da computação não está nas soluções de sistemas, dispositivos, e interfaces que propuseram. A maior contribuição deles é a realização de que essas interfaces e dispositivos teriam uma importância fundamental na forma como o homem lida com seus processos intelectuais. A ideia de pensamento computacional vai ainda além e sugere que não é nem mesmo necessário utilizar computadores para se pensar computacionalmente ³¹. O que se apreende a partir dos trabalhos desses pensadores da computação é que a computação pode contribuir fortemente para todas as áreas do conhecimento, e não apenas para a própria ciência da computação. Assim como é possível encontrar paralelos entre a computação e atividades cotidianas (como aquelas encontradas nos exemplos de pensamento com-

³¹Um autor que deve ser mencionado neste desfecho de capítulo é o filósofo francês Gilbert Simondon. Em *El modo de existencia de los objetos técnicos*, Simondon fala da falsa oposição entre homem e máquina: “a oposição que se ergueu entre a cultura e a técnica, entre o homem e a máquina, é falsa e sem fundamentos; apenas disfarça ignorância ou ressentimento” (SIMONDON, 2007, p.31). Para ele, esse falso antagonismo não é causado pela máquina, e sim pela ignorância das características intrínsecas da máquina: “a maior causa de alienação no mundo contemporâneo reside no desconhecimento da máquina, que não é uma alienação causada pela máquina, e sim pelo não-conhecimento de sua natureza e sua essência” (SIMONDON, 2007, p.31). Para Simondon, a cultura trata os objetos técnicos meramente em seus aspectos funcionais, ignorando seus outros aspectos, como os expressivos e estéticos. Ao criar uma dicotomia entre homem e máquina, a cultura acaba negando a realidade humana que existe dentro dos objetos técnicos. Mas os objetos técnicos não são subordinados ao homem, não estão em um nível hierárquico inferior. E Simondon não propõe uma visão tecnocêntrica, muito pelo contrário: ele critica fortemente a idolatria da máquina. O que Simondon defende é uma conscientização dos modos de existência próprios dos objetos técnicos. É um texto importante para desfazer a dicotomia entre homem e máquina.

putacional), é possível encontrar paralelos entre a computação e a arte e o design. O próximo capítulo explora algumas ideias advindas da arte e do design que exibem paralelos com a computação.

Capítulo 2

Design como programa

Este capítulo tem como objetivo discutir o design como programa, mostrando que a computação pode contribuir para o design não apenas como simulação de outras ferramentas, mas principalmente com uma abordagem sistêmica e dinâmica dos processos de projeto. Para tanto, este capítulo discorre a respeito da arte generativa e dos programas de Karl Gerstner.

Como foi visto no capítulo anterior, uma das maiores contribuições da computação para a sociedade é o software, pois é por meio do software que as pessoas interagem com os computadores. E, como consequência da natureza meta-mídia da computação, o software frequentemente funciona como simulações de ferramentas analógicas. Por exemplo, programas de desenho simulam ferramentas como papel, lápis, caneta, e pincel. E além de simular essas tecnologias analógicas, o software expande suas funcionalidades, facilitando seus usos. No caso de um software de desenho por exemplo, esse software permite a obtenção de diversas cores instantaneamente, sem a necessidade de comprar ou misturar tintas, e permite também a correção de erros sem que o suporte do desenho seja danificado. Dessa forma, ferramentas computacionais colaboram para um aumento da produtividade no trabalho em diversas atividades profissionais, inclusive no design¹.

¹Talvez alguns designers não queiram associar o termo “produtividade” com seu trabalho, mas há muitas tarefas que podem ser realizadas de maneira mais eficiente com o auxílio de computadores. O artista computacional Mark Wilson diz: “computadores podem fazer certos tipos de imagens muito mais rápido do que a mão do artista. Essas máquinas não só podem fazer imagens mais rápido, mas elas reduzem ou eliminam o trabalho braçal de fazer as imagens” (WILSON, 1985, p.16).

Tornar o trabalho de desenhar mais eficiente (por meio de simulações de outras ferramentas) já é uma grande contribuição, mas a computação pode contribuir ainda mais para o design: a computação pode participar de forma ativa dos processos intelectuais envolvidos no processo de design. Quando Kay falava em meta-mídia, imaginava que a computação permitiria não só a simulação de outras mídias, mas também a invenção de novas mídias. Licklider, Engelbart, e Bush imaginavam que a computação participaria ativamente dos processos intelectuais do homem, quase como se fosse um segundo cérebro. Wing propõe que os conceitos da computação podem ser aplicados mesmo sem a utilização do computador. No entanto, observando a prática profissional do design hoje, constata-se que a computação é extensivamente utilizada enquanto ferramenta de simulação de outras ferramentas, e pouco utilizada como amplificador intelectual. As ferramentas são computacionais, mas os processos ainda são pautados pelas ferramentas simuladas: o papel, o lápis, a caneta. Em geral, o design vem explorando muito mais o aspecto “simulador” da computação do que o aspecto “amplificador”. A questão que se coloca é, como proceder para que a computação possa contribuir mais para o design, participando ativamente de seus processos mentais? A seguir são abordados dois temas que indicam caminhos possíveis para essa relação: a arte generativa e o trabalho de Karl Gerstner com seu livro *Designing programmes* (1964).

A arte generativa é uma prática artística que joga luz nos **processos** envolvidos na criação e na produção de uma obra de arte. Ou seja, a arte generativa não está preocupada apenas com a técnica e a composição da obra de arte. A arte generativa enxerga a obra de arte como uma entidade dinâmica, resultante de processos complexos nos quais o artista é apenas um dos atores envolvidos. Se a computação trata de “processos de informação, naturais e artificiais” (DENNING, 2010, p.372), a arte generativa trata da arte processual.

2.1 Arte generativa

Em primeiro lugar, é necessário compreender o que define essa “arte generativa”. Uma definição é dada pelo pesquisador e artista Philip Galanter, em seu artigo *What is generative art?* (2003):

Arte generativa se refere a qualquer prática artística na qual o artista usa um sistema, como um conjunto de regras em linguagem natural, um programa de computador, uma máquina, ou outra invenção procedimental, que é colocada em movimento com algum grau de autonomia, contribuindo para ou resultando em um trabalho completo de arte. (GALANTER, 2003, p.4)²

Em outras palavras, na arte generativa, o artista coloca entre ele mesmo e sua obra algum sistema que possui autonomia para interferir ou gerar essa obra. Esta definição emprega o termo “sistema” de forma ampla, compreendendo desde um conjunto de regras escritas em linguagem natural até uma máquina. Ou seja, esses “sistemas” podem ser entendidos como qualquer conjunto de partes ou coisas que formam uma unidade. No entanto, nem qualquer tipo de sistema está relacionado com a arte generativa. No artigo *Generative art and rules-based art*, de 2006, Galanter discorre sobre os sistemas de regras na arte e aponta algumas diferenças entre regras não-generativas e regras generativas, observando que essa diferenciação é imprecisa, com muitos casos que apresentam características ambíguas. Em termos gerais, regras não-generativas limitam a autonomia do executor (instruções para fabricação de um objeto industrial, por exemplo), ou são abertas à livre interpretação do executor (instruções baseadas em frases inspiracionais, por exemplo). Por outro lado, regras generativas estão associadas com algoritmos, permutações, sequências numéricas, linhas, mosaicos, simetrias, aleatoriedade, mapeamentos, e ciclos (GALANTER, 2006). Desse modo, os sistemas envolvidos na arte generativas são aqueles que trabalham esses tipos de regras. Galanter ainda observa que sistemas podem ser simples ou complexos, de acordo com o grau de ordem e de desordem que possuem. Sistemas altamente ordenados e altamente desordenados são considerados simples. Sistemas que apresentam ordem e desordem em medidas iguais são considerados complexos (GALANTER, 2003). Para Galanter, a arte generativa é tão antiga quanto a própria arte, pois recursos de organização espacial como a simetria e o mosaico são sistemas, ainda que simples. A defi-

²Generative art refers to any art practice where the artist uses a system, such as a set of natural language rules, a computer program, a machine, or other procedural invention, which is set into motion with some degree of autonomy contributing to or resulting in a completed work of art (GALANTER, 2003, p.4).

nição de arte generativa de Galanter é uma das mais difundidas atualmente, por sintetizar o conceito em poucas palavras e, ao mesmo tempo, apresentar o conceito sem limitá-lo a tecnologias específicas.

Outra definição pode ser encontrada na apresentação da conferência internacional *Generative Art* (realizada desde 1998), escrita pelo pesquisador Celestino Soddu. Para Soddu, “arte generativa é a ideia realizada (...) como construção de sistemas dinâmicos e complexos capazes de gerar variações sem fim” (SODDU, 2008). Tanto para Galanter quanto para Soddu, a utilização de sistemas na arte é o aspecto central da arte generativa. Soddu ainda complementa que a arte generativa produz “eventos únicos e não-repetíveis” (SODDU, 2008). Ou seja, mesmo que um artista utilize sistemas autônomos ou semi-autônomos, se esses sistemas repetem sempre os mesmos resultados, a obra resultante não pode ser considerada generativa.

Portanto, a arte generativa pode ser caracterizada por alguns elementos: o emprego de sistemas, a autonomia desses sistemas em relação ao artista, e a variabilidade dos resultados obtidos, sendo que os sistemas da arte generativa são caracterizados por trabalharem com recursos como algoritmos, permutações, e aleatoriedade. Mas é importante esclarecer que algoritmos são caracterizados pela previsibilidade de seus resultados, e não pela variabilidade deles. Um algoritmo consiste de uma sequência de instruções que, repetida várias vezes com os mesmos parâmetros, produz os mesmos resultados. Para se obter resultados variáveis de um algoritmo, é necessário executá-lo várias vezes com diferentes parâmetros. Parâmetros aleatórios podem ser obtidos computacionalmente, mas não são considerados rigorosamente aleatórios (pois possuem padrões estatísticos), sendo denominados números pseudo-aleatórios. Números verdadeiramente aleatórios devem ser obtidos a partir de fenômenos naturais imprevisíveis ou de interações humanas. Além disso, no caso de algoritmos que são executados por pessoas (e não por computadores), a interpretação subjetiva das instruções do algoritmo é outro elemento que possibilita o surgimento de resultados aleatórios e variáveis.

Galanter ressalta que a arte generativa é uma prática artística, e não um movimento artístico (GALANTER, 2003, p.16). Portanto, a prática generativa pode ser reconhecida em obras de diferentes épocas e diferentes movimentos artísticos. A seguir são

apresentados 3 obras de arte que exemplificam a prática da arte generativa: *Wall painting #122* de Sol LeWitt (1972), *In C* de Terry Riley (1964), e *Inversion logique* de Manfred Mohr (1970). Foram selecionadas obras de diferentes gêneros, para demonstrar que a arte generativa pode se manifestar em diversos gêneros, linguagens, e tecnologias.

O primeiro exemplo aqui discutido é um trabalho do artista plástico Sol LeWitt (1928-2007). Embora LeWitt seja mais reconhecido como artista conceitual e minimalista, Galanter defende que a maior parte da produção de LeWitt é generativa, já que LeWitt explorava os valores intrínsecos dos sistemas (GALANTER, 2003, p.18). Na série de trabalhos conhecida como *Wall paintings* (murais), LeWitt fornecia instruções com textos e diagramas para a execução de pinturas em muros. LeWitt não pintava esses murais nem determinava a configuração dos desenhos a serem pintados. Os desenhos deveriam ser determinados por sua equipe de assistentes, a partir das instruções deixadas por LeWitt. Essas instruções eram abertas, ou seja, permitiam a obtenção de resultados diferentes a cada execução. Em *Wall drawing #122* (1972), o seguinte enunciado constitui a instrução para sua pintura: “todas as combinações de duas linhas cruzando, posicionadas aleatoriamente, usando arcos dos cantos e lados, e linhas retas, tortas, e tracejadas” (figura 2.1). Logo abaixo do título da obra e da instrução, LeWitt desenhou os elementos gráficos (arcos e linhas) que deveriam ser combinados no desenho final. O mural resultante deveria exibir todas as combinações possíveis de 2 elementos desse conjunto.

A figura 2.2 apresenta um esquema para a realização desse mural. O plano apresenta os 20 elementos a serem combinados, e atribui um número de 1 a 20 a cada um dos elementos. Na planta do espaço de intervenção, foi atribuída uma letra de A a D para cada uma das paredes a serem pintadas. Cada uma das 4 paredes foi dividida em um grid de 3 linhas por 16 colunas (resultando em 192 unidades no total, com 2 unidades em branco³). O plano ainda inclui 2 esquemas para cada parede: um que mostra os códigos dos elementos a serem combinados em cada unidade do grid, e outro que mostra o desenho final a ser obtido.

³Duas unidades do grid ficaram em branco porque o total de combinações dos 20 elementos dados por LeWitt é 190, e o grid possui 192 unidades. Para cada elemento X do conjunto de 20 elementos possíveis, há 19 pares possíveis. Ou seja, o número de combinações possíveis é $20 \times 19 = 380$. No entanto, é necessário ignorar as combinações repetidas, como por exemplo os pares XY e YX. Assim, é preciso dividir o resultado obtido por 2, resultando em 190 combinações únicas $380 \div 2 = 190$.

Figura 2.1: Sol LeWitt. Wall drawing #122: All combinations of two lines crossing, placed at random using arcs from corners and sides, straight, not straight and broken lines, 1972. Fonte: site da revista Mousse Magazine. Disponível em: <http://moussemagazine.it/sol-lewitt-arcs-and-lines-at-paula-cooper-gallery-new-york/>.

Figura 2.2: Esquema para realização de Wall Drawing #122, 1972. Fonte: site de Alex Miller. Disponível em: <http://blog.alexmilller.info/post/5568758820>.

Em 2011, Wall painting #122 foi executada para uma exposição de Sol LeWitt realizada na Paula Cooper Gallery. As fotos 2.3 e 2.4 mostram a obra exposta na galeria. Nas fotos, percebe-se que a configuração do desenho na parede é diferente do desenho apresentado na figura 2.2, embora os elementos e as regras utilizadas sejam os mesmos. A instrução fornecida pelo artista é a mesma, mas a obra resultante é diferente. Esta possibilidade de obter resultados diferentes a partir de um mesmo conjunto de instruções é uma das características da arte generativa.

Neste exemplo, o sistema utilizado pelo artista é uma instrução escrita, acompanhada de um desenho. Essa instrução escrita pode ser interpretada de várias formas. A cada vez que a mesma instrução é executada, o resultado obtido pode ser completamente diferente dos resultados obtidos anteriormente. Cada equipe que executa uma obra de LeWitt interpreta suas instruções de forma diferente. Cada situação oferece espaços e dimensões diferentes. E são essas variáveis que informam a pintura resultante. Pode-se dizer que o trabalho de LeWitt se baseia na **abstração** do mural, na criação de um sistema de regras que é utilizado pelos pintores para **gerar** instâncias **concretas** da

Figura 2.3: Wall Drawing #122 na Paula Cooper Gallery, 2011. Fonte: site da Paula Cooper Gallery. Disponível em: <http://www.paulacoopergallery.com/artists/SL/works/305>.

Figura 2.4: Wall Drawing #122 na Paula Cooper Gallery, 2011. Fonte: site da Paula Cooper Gallery. Disponível em: <http://www.paulacoopergallery.com/artists/SL/works/305>.

obra. É um sistema primariamente combinatório, no qual os elementos fornecidos por LeWitt são combinados pela sua equipe de pintores.

Outra obra que exemplifica a prática da arte generativa é a peça musical *In C* (1964), do compositor Terry Riley. A peça consiste em uma partitura contendo 53 frases musicais curtas compostas por Riley (figura 2.5) e uma série de instruções escritas (figuras 2.6 e 2.7) para a execução dessa peça. A regra para a execução de *In C* é que as 53 frases musicais devem ser tocadas em sequência, mas cada músico da orquestra determina o número de repetições para cada frase. Por exemplo: todos os músicos iniciam a peça tocando a frase 1. No entanto, enquanto um músico repete essa frase por 5 vezes e passa para a frase 2, outro músico pode repetir a mesma frase por 8 vezes até passar para a frase 2. Dessa forma, Riley queria que padrões musicais inesperados emergissem das interações entre as diferentes frases sendo executadas pelos músicos a cada momento. Para tanto, Riley deixou alguns parâmetros da composição em aberto: a duração da peça fica a cargo dos músicos que executam a peça, assim como o próprio número de músicos na orquestra (embora Riley recomende uma formação de aproximadamente 35 membros). Tal como na obra de LeWitt, as instruções de *In C* podem ser interpretadas de diversas formas. Desse modo, *In C* é uma peça que pode **gerar** inúmeras performances diferentes baseadas no mesmo conjunto de instruções. O sistema utilizado por Riley é algorítmico, pois define uma sequência de operações para os músicos da orquestra, e trabalha com combinação, sequenciamento, e aleatoriedade na execução das 53 frases musicais pré-compostas.

Outros compositores do século XX, como John Cage, Karlheinz Stockhausen, Steve Reich, e Brian Eno, também utilizaram processos generativos em suas composições musicais. E é possível identificar um importante precedente histórico para essa abordagem generativa de composição musical nos “jogos musicais de dados”, ou *Musikalisches Würfelspiel* (DIAZ-JEREZ, 2000, p.21), que surgiram no século 18 (NIERHAUS, 2009, p.36). Assim como em *In C* de Riley, esses jogos musicais fornecem um conjunto de frases musicais pré-compostas, que devem ser selecionadas aleatoriamente de acordo com o resultado da rolagem de 2 dados. Existem jogos musicais de dados cujas autorias são atribuídas à Mozart e Haydn, mas a autoria desses compositores não é comprovada (NIERHAUS, 2009, p.38).

Figura 2.5: Terry Riley. Partitura de *In C*, 1964. Fonte: blog Mechanical Forest Sound. Disponível em: <http://mechanicalforestsound.blogspot.com.br/2011/01/festival-toronto-new-music-marathon.html>.

In C

Performing Directions

All performers play from the same page of 53 melodic patterns played in sequence.

Any number of any kind of instruments can play. A group of about 35 is desired if possible but smaller or larger groups will work. If vocalist(s) join in they can use any vowel and consonant sounds they like.

Patterns are to be played consecutively with each performer having the freedom to determine how many times he or she will repeat each pattern before moving on to the next. There is no fixed rule as to the number of repetitions a pattern may have, however, since performances normally average between 45 minutes and an hour and a half, it can be assumed that one would repeat each pattern from somewhere between 45 seconds and a minute and a half or longer.

It is very important that performers listen very carefully to one another and this means occasionally to drop out and listen. As an ensemble, it is very desirable to play very softly as well as very loudly and to try to diminuendo and crescendo together.

Each pattern can be played in unison or canonically in any alignment with itself or with its neighboring patterns. One of the joys of IN C is the interaction of the players in polyrhythmic combinations that spontaneously arise between patterns. Some quite fantastic shapes will arise and disintegrate as the group moves through the piece when it is properly played.

It is important not to hurry from pattern to pattern but to stay on a pattern long enough to interlock with other patterns being played. As the performance progresses, performers should stay within 2 or 3 patterns of each other. It is important not to race too far ahead or to lag too far behind.

The ensemble can be aided by the means of an eighth note pulse played on the high c's of the piano or on a mallet instrument. It is also possible to use improvised percussion in strict rhythm (drum set, cymbals, bells, etc.), if it is carefully done and doesn't overpower the ensemble. All performers must play strictly in rhythm and it is essential that everyone play each pattern carefully. It is advised to rehearse patterns in unison before attempting to play the piece, to determine that everyone is playing correctly.

Figura 2.6: Terry Riley. Instruções para a execução de In C (página 1), 1964. Fonte: site do Museu de Arte Contemporânea de Houston. Disponível em: http://camh.org/calendar/2013/8/musica-loft-concert#.Uuqs_HddVcQ.

The tempo is left to the discretion of the performers, obviously not too slow, but not faster than performers can comfortably play.

It is important to think of patterns periodically so that when you are resting you are conscious of the larger periodic composite accents that are sounding, and when you re-enter you are aware of what effect your entrance will have on the music's flow.

The group should aim to merge into a unison at least once or twice during the performance. At the same time, if the players seem to be consistently too much in the same alignment of a pattern, they should try shifting their alignment by an eighth note or quarter note with what's going on in the rest of the ensemble.

It is OK to transpose patterns by an octave, especially to transpose up. Transposing down by octaves works best on the patterns containing notes of long durations. Augmentation of rhythmic values can also be effective.

If for some reason a pattern can't be played, the performer should omit it and go on.

Instruments can be amplified if desired. Electronic keyboards are welcome also.

IN C is ended in this way: when each performer arrives at figure #53, he or she stays on it until the entire ensemble has arrived there. The group then makes a large crescendo and diminuendo a few times and each player drops out as he or she wishes.

Figura 2.7: Terry Riley. Instruções para a execução de In C (página 2), 1964. Fonte: site do Museu de Arte Contemporânea de Houston. Disponível em: http://camh.org/calendar/2013/8/musica-loft-concert#.Uuqs_HddVcQ.

Por fim, a terceira obra selecionada para exemplificar a prática generativa é do pioneiro da arte computacional Manfred Mohr⁴. Mohr produziu seus primeiros desenhos computacionais em 1969, e foi influenciado pelos textos do filósofo Max Bense sobre a estética generativa⁵. Mohr também foi influenciado por McLuhan, e enxergava na computação uma extensão do homem, “em corpo e em mente” (MOHR, 1976, p.14). Assim como Licklider, Engelbart, Bush, e Kay, Mohr também enxergava o potencial de amplificação intelectual da computação, e dizia que “o pensamento humano pode ser ‘amplificado’ por máquinas, elevando nossa consciência a um nível mais alto de compreensão” (MOHR, 1976, p.14). Em 1971, o Museu de Arte Moderna de Paris realizou uma exposição com seus trabalhos. No catálogo dessa exposição, Mohr reconhece “o computador como **parceiro**” em seu trabalho (MOHR, 1971), numa postura semelhante à de Licklider com seu conceito de simbiose. Para Mohr, a computação permite a exploração das diversas possibilidades de um programa, que por sua vez permite a obtenção de resultados não-imaginados pelo autor, resultados “diferentes e talvez mais interessantes” (MOHR, 1971).

A figura 2.8 exibe um desenho de uma série intitulada *Inversion logique*, que fez parte da exposição de Mohr em 1971. Esta série é baseada em linhas horizontais, verticais, e inclinadas em 45 graus, em que os comprimentos e as espessuras das linhas são variáveis. Os desenhos da série são compostos por 63 símbolos, dispostos em uma matriz de 7 colunas por 9 linhas. E esses símbolos são gerados por um algoritmo, que preenche a matriz em uma sequência linear, da esquerda para a direita, e de cima para baixo. Na primeira metade da matriz, os símbolos são construídos por adição. Ou seja, um símbolo em uma determinada posição da matriz é igual ao símbolo na posição anterior, com a adição de uma linha. A rotação, o comprimento, e a espessura da linha adicionada são determinadas aleatoriamente pelo algoritmo. Na segunda metade da matriz, os símbolos são obtidos por subtração. Um símbolo em uma determinada po-

⁴Esta dissertação não se aprofunda na história da arte computacional, pois são muitos os artistas que contribuíram para essa história. Alguns nomes merecem destaque, como A. Michael Noll, Frieder Nake, Georg Nees, Harold Cohen, e Roman Verotsko. A exposição *Cybernetic serendipity*, realizada no Institute of Contemporary Arts em Londres em 1968, com organização por Jasia Reichardt, também teve papel importante no desenvolvimento da arte computacional, por ter sido uma das primeiras grandes exposições de arte que exibia trabalhos produzidos com tecnologias computacionais.

⁵Disponível em: <http://www.emohr.com/biography.html>.

sição da matriz é igual ao símbolo na posição anterior, menos uma linha. As linhas são removidas na ordem inversa em que foram adicionadas. Desse modo, o último elemento adicionado é sempre o elemento que sobra na última posição da matriz. Assim como em *In C*, o sistema empregado nesta obra é essencialmente algorítmico, pois resulta de uma sequência de operações, e trabalha com permutações e aleatoriedade.

Assim, foram apresentados 3 exemplos de práticas artísticas generativas, com tecnologias e linguagens diferentes. O que esses trabalhos têm em comum é abordagem sistêmica do processo artístico. Os artistas criaram sistemas de regras que lhes permitiam gerar inúmeras variações de suas obras. Ou seja, a arte generativa não introduz uma técnica ou linguagem específica na arte, mas evidencia o papel dos processos da arte na busca por variações em seus resultados. Há muitas motivações para a arte generativa, e Galanter lembra que “o que artistas generativos têm em comum é como eles realizam seu trabalho, mas não porque o realizam, ou mesmo porque eles escolhem usar sistemas generativos em suas práticas artísticas” (GALANTER, 2003, p.14)⁶. A arte generativa não é um movimento artístico; ela abrange um número de diversas práticas artísticas que exploram o processo de criação como matéria-prima para a própria criação artística.

Nesse sentido, é importante ressaltar que, embora haja uma proximidade entre a arte generativa e a arte computacional, a arte generativa não é necessariamente computacional. Galanter diz: “como a arte generativa contemporânea é frequentemente baseada em computadores, muitos assumem que ela é um subconjunto da arte computacional” (GALANTER, 2003, p.15)⁷. Por outro lado, Soddu supõe que a arte generativa é computacional, quando diz que “cada projeto generativo é um software-conceito” (SODDU, 2008). Embora esta dissertação trate do design computacional, a postura adotada aqui é a de Galanter, e não a de Soddu. A definição de Galanter permite uma compreensão mais abrangente da arte generativa, na qual o importante não é a utilização de uma tecnologia específica em si, mas os processos nos quais essa tecnologia está envolvida. Dessa forma, a computação é apenas uma das ferramentas que pode

⁶what generative artists have in common is how they make their work, but not why they make their work, or even why they choose to use generative systems in their art practice (GALANTER, 2003, p.14).

⁷Because contemporary generative art is so very often computer based many assume it is a subset of computer art (GALANTER, 2003, p.15).

Figura 2.8: Manfred Mohr. *Inversion logique*, 1970. Fonte: site do artista. Disponível em: http://www.emohr.com/sc69-73/vfile_26.html.

ser utilizada na prática da arte generativa.

Tampouco é possível classificar todos os trabalhos de arte computacional como arte generativa. Se um artista utiliza o computador apenas como simulador de um pincel, a obra resultante não deve ser considerada generativa. Mesmo que o pincel seja programado, se seu programa apenas reproduz repetidamente o mesmo resultado, sem influência de fatores externos, ou sem variação nos seus parâmetros internos, a obra resultante ainda não pode ser considerada generativa. A obra só pode ser considerada generativa quando esse sistema computacional, o programa, contemplar a influência de fatores externos ou variações em seus parâmetros internos, gerando resultados variáveis e não-repetidos, ou seja, quando tiver algum grau de autonomia em relação ao artista que o criou.

Outra prática artística relacionada com a arte generativa e a arte computacional é a arte evolutiva, também conhecida como arte genética em alguns casos⁸. A arte evolutiva trabalha com sistemas “baseados na simulação de processos de seleção natural e reprodução em um computador” (MCCORMACK; DORIN; INNOCENT, 2004, p.5). Na arte evolutiva, um número de elementos (sejam desenhos, imagens, objetos, notas musicais) são inicialmente gerados com parâmetros aleatórios. O artista (ou um algoritmo programado pelo artista) então escolhe aqueles elementos que considera mais interessantes, e faz com que eles “procriem” entre si, resultando em uma nova geração de elementos que herda características dos seus “pais”. Esse processo pode ser repetido inúmeras vezes, de acordo com critérios determinados pelo artista. Pode-se considerar a arte evolutiva como uma subcategoria da arte generativa e da arte computacional. Mas esta dissertação não tem interesse em tratar das diversas subcategorias da arte generativa, e sim em explorar seus paralelos com a computação.

Sendo assim, pode-se observar algumas aproximações entre arte generativa e computação: ambas lidam com sistemas, processos, e abstrações. Se o pensamento computacional propõe a abstração de um problema, com a identificação de padrões e camadas dentro desse problema, a arte generativa propõe a abstração da própria obra

⁸O termo “arte genética” também pode se referir a obras que trabalham com o DNA de organismos vivos (GESSERT, 2004). Esse tipo de arte pode até ser relacionada com a arte generativa, por lidar com processos biológicos. No entanto, essa arte “biogenética” acaba questionando muito mais as relações entre tecnologia e natureza, entre artificial e natural. Dessa forma, não é do interesse desta dissertação tratar dessa prática artística.

de arte, por meio da utilização de sistemas autônomos ou semi-autônomos. A partir de uma abstração de uma obra, é possível **gerar** (daí “generativo”) várias instâncias **concretas** (não-abstratas). A arte generativa engloba não apenas o objeto final (pintura, escultura, música) da arte em si, mas também uma abstração desse objeto, e os processos envolvidos na concretização dessa abstração.

Outra aproximação entre arte generativa e computação é o uso de instruções. Para Bense, autor de *The projects of generative aesthetics* (texto que influenciou muitos artistas generativos como Mohr), “a estética generativa é uma ‘estética de produção’, que torna possível a produção metódica de estados estéticos, ao dividir este processo em um número finito de etapas distintas e separadas que são capazes de formulação” (BENSE, 1971, p.5)⁹. Ou seja, uma estética generativa implica em uma abordagem sistemática do processo de produção dessa mesma estética. A subdivisão desse processo em etapas constitui uma sequência de instruções, um algoritmo. Essas instruções funcionam como formas de descrever abstrações e padrões, fazendo ponte entre algo abstrato e algo concreto. O algoritmo permite a obtenção de um conjunto de diferentes resultados concretos a partir de uma abstração.

Pode-se dizer que o que caracteriza a arte generativa é, mais do que a utilização de alguma máquina ou computador, a importância do processo pelo qual essa obra de arte foi criada. Em uma obra de arte não-generativa, o artista determina exatamente as cores, dimensões, e texturas de uma obra, tendo o maior controle possível sobre seu resultado final. Já na arte generativa, o artista cria um sistema, que por sua vez determina as cores, dimensões, e texturas do trabalho final. Esse sistema gera resultados de acordo com instruções programadas pelo artista, mas o artista não controla exatamente a execução dessas instruções. O sistema funciona de acordo com os inputs recebidos, inputs que são utilizados como parâmetros para as instruções programadas. Quanto mais variáveis envolvidas nesse processo, maior é o número de possibilidades que podem ser geradas, por meio de combinações entre essas variáveis. Os inputs recebidos pelo sistema podem ser de diversas fontes: pessoas que interagem com o sistema, dados da natureza providos por sensores, outros sistemas externos, ou até

⁹generative aesthetics is an ‘aesthetics of production’, which makes possible the methodical production of aesthetic states, by dividing this process into a finite number of distinct and separate steps which are capable of formulation (BENSE, 1971, p.5).

mesmo geradores de números (pseudo)aleatórios (como dados). Visto que o sistema recebe inputs que não são controlados pelo artista, este não tem controle total sobre o resultado desse processo. No entanto, é interessante lembrar que o funcionamento desse sistema é programado pelo artista, portanto não é que o artista não tem controle nenhum sobre a obra. Na arte generativa, o artista tem controle parcial sobre a obra, sendo que o grau de controle que o artista possui é consequência do tipo de sistema que esse artista propõe.

A arte generativa implica em menor controle do autor sobre a obra. Em outras palavras, o autor de uma obra de arte generativa transfere algumas das tomadas de decisões envolvidas no processo de criação da obra para um sistema autônomo, cedendo controle sobre a obra para esse sistema. Por outro lado, o modernismo tende a apontar para a direção contrária, de maior controle do autor sobre a obra. Ou seja, um autor modernista procura estar envolvido em todos os aspectos envolvidos na criação de sua obra. Interessantemente, é justamente no design modernista suíço que se encontra um designer cujo trabalho exhibe paralelos com a arte generativa e a computação. Esse designer é Karl Gerstner, que falava em “projetar programas”.

2.2 Projetando programas

Karl Gerstner é um dos expoentes da escola suíça de design, ao lado de nomes como Josef Müller-Brockmann, Emil Ruder, e Armin Hofmann. A importância de Gerstner compreende tanto a prática quanto a teoria do design gráfico. Para o pesquisador de design gráfico Richard Hollis, Gerstner é um dos mais coerentes teóricos do design gráfico (HOLLIS, 2002). Entre suas maiores contribuições como profissional designer, está seu trabalho com grids, a utilização de texto alinhado à esquerda e não-justificado (desalinhado à direita), e a ideia de **tipografia integral** (que consiste na unidade entre linguagem e tipografia; ou seja, a tipografia e a ideia representada pela tipografia devem formar uma unidade coesa, uma integralidade). De sua produção teórica, esta dissertação concentra as discussões na obra *Designing programmes* (1964), considerado o livro mais influencial de Gerstner por Hollis (2002). Neste livro, Gerstner apresenta uma abordagem de design baseada em permutações. Identifica-se nesta

abordagem uma proximidade com a arte generativa e o pensamento computacional.

O texto introdutório escrito por Paul Gredinger (sócio de Gerstner na agência de publicidade GGK¹⁰) é esclarecedor para compreender o conceito de “programa” que permeia a obra de Gerstner. Segundo Gredinger, “projetar programas pode significar inventar regras de composição” (GREDINGER, 1964, p.4), sendo que programa é “um tipo de fórmula” que cria “um grupo de formas”. Um programa deve compreender um conjunto de elementos e um conjunto de regras para a combinação desses elementos. Um dos exemplos dados por Gredinger é a receita culinária. Considerando a receita como um programa, seus ingredientes são os elementos do programa, e a preparação é o conjunto de regras do programa; essas regras devem ser seguidas para a obtenção do prato desejado. Outro exemplo cotidiano de programa pode ser encontrado no tricô, em que o programa consiste em uma sequência de instruções que resulta em um desenho ou padrão ¹¹. E outro exemplo são as partituras musicais, especialmente as de compositores de vanguarda como John Cage e Karlheinz Stockhausen. Nessas partituras, os sons não são completamente predeterminados pelo compositor; o acaso e a aleatoriedade fazem parte das peças musicais resultantes. Em suma, projetar programas é definir elementos e suas regras de composição. Aos olhos racionalistas de Gredinger (e Gerstner), o “prazer criativo” do design está na invenção de regras de composição:

Não é importante que o resultado seja isto ou aquilo; o que é importante é que a forma pode e deve adquirir forma em obediência a uma ordem ou fórmula. É no design da fórmula (imagem: uma flor de tulipa) e não no design da forma (imagem: tulipa) que o prazer criativo reside. (GREDINGER, 1964, p.5)¹²

¹⁰GGK é a sigla composta pelas iniciais de seus 3 sócios: Gerstner, Gredinger, e Kutter.

¹¹A tecelagem ofereceu uma importante contribuição para a computação, com a máquina de tear automatizada de Joseph-Marie Jacquard. O tear de Jacquard automatizava as tarefas repetitivas envolvidas no processo de tricotar, e era programável por meio de programas contidos em cartões perfurados. A ideia de armazenar programas em cartões perfurados influenciou Charles Babbage no desenvolvimento de suas máquinas de calcular, programadas por Ada Lovelace (RHEINGOLD, 1984, cap.1).

¹²Result: a whole series of solutions. It is not important that the result should be this or that; what is important is that the form should and must take its shape in obedience to an order or formula. It is in the design of the formula (image: a tulip bulb) and not in the design of the form (image: tulip) that the creative pleasure resides (GREDINGER, 1964, p.5).

Embora Gredinger não mencione a computação em seu texto, pode-se observar nele paralelos diretos com a computação, especialmente quando dá exemplos. A propósito, o exemplo da receita culinária já foi visto no capítulo anterior, na discussão sobre o pensamento computacional. Gerstner aborda a computação em seu texto, mas limita o assunto a apenas uma página. Gerstner reproduz um depoimento do autor do pioneiro da arte computacional Frieder Nake¹³ e também algumas imagens da obra *Series 201* (1966) de Nake. Esse trabalho consiste de uma série de desenhos produzidos por computador, e são variações obtidas a partir de um mesmo programa (figura 2.9).

A teoria de Gerstner sobre os “programas” em design não se limitava aos programas de computador. Os programas de Gerstner podem ser compreendidos mais como sistemas que permitem permutações. Um exemplo apresentado por Gerstner e que evidencia o aspecto combinatório dos programas é o conjunto de janelas de uma catedral gótica avistada por Gerstner em seu caminho para o trabalho (figura 2.10). Cada janela dessa catedral possui um ornamento de desenho único, diferente dos ornamentos nas outras janelas. No entanto, o desenho de todos esses ornamentos segue um mesmo princípio, um mesmo programa: suas linhas são curvas, e se adaptam organicamente entre si, com a borda da janela, e com a divisão tripartite da janela. A variedade de soluções que pode ser obtida a partir de um mesmo sistema de elementos e princípios é que caracteriza um programa para Gerstner.

Tendo isso em vista, os programas de Gerstner constituem uma prática de design próxima da prática da arte generativa, pois seus programas são sistemas, que geram resultados variados, e que não se repetem. Considerando que um programa consiste de um conjunto de elementos e de um conjunto de regras (para obter combinações entre esses elementos), constata-se que os trabalhos de LeWitt e Riley aqui apresentados também operam dessa maneira. Tanto *Wall drawing #122* quanto *In C* apresentam um conjunto de elementos e instruções para a combinação desses elementos; são “fórmulas” para gerar “formas” gráficas e musicais.

¹³Nake publicou um artigo intitulado *There should not be computer art* (1971), no qual se opõe à apropriação da computação pelo mercado da arte. Para Nake, a arte computacional havia se tornado um modismo da arte em busca de lucros. Para ele, a computação não deveria ser utilizada apenas para produzir imagens bonitas, e sim para produzir um filme sobre, por exemplo, a distribuição desigual de renda (NAKE, 1971, p.19). Essa atitude política em relação à computação e às artes tecnológicas encontra ecos nos textos de Machado (2007).

Figura 2.9: Frieder Nake. *Series 201*, 1966. Fonte: GERSTNER, 1964, p.21.

Figura 2.10: Karl Gerstner. Programa da Idade Média. Fonte: GERSTNER, 1964, p.11.

Para Gerstner, até o grid tipográfico pode ser considerado um programa, ao funcionar como sistema que permite permutações no espaço: “se o grid é considerado como um regulador de proporções, um sistema, então é um programa por excelência” (GERSTNER, 1964, p.16). A utilização de grids era um dos elementos em comum entre os trabalhos de Gerstner e de seus colegas suíços. Outros elementos em comum eram o uso de formas geométricas e da tipografia sem-serifa. Mas o trabalho de Gerstner se destacava dos outros designers suíços por sua tendência matemática. Em texto introdutório da edição de 2007 de *Designing programmes*, Hollis identifica que o “interesse em repetição e permutação era evidente no início da carreira de Gerstner” (HOLLIS, 2007, p.7). Gerstner explorava composições matemáticas e geométricas e a ideia de programas desde a década de 50. Em *Design gráfico: uma história concisa*, Hollis ainda observa que o trabalho de Gerstner é caracterizado pela importância do conceito de sistema, “no qual um efeito estético emerge da escolha lúcida de meios gráficos limitados” (HOLLIS, 2001, p.140). A preocupação de Gerstner com a lógica e a permutação se torna evidente na seguinte passagem, em que defende a primazia da razão (“critérios intelectuais”) sobre a emoção (“sentimento”):

Descrever o problema é parte da solução. Isto implica em: não tomar decisões criativas de acordo com sentimento mas de acordo com critérios intelectuais. Quanto mais exatos e completos forem esses critérios, mais criativo o trabalho se torna. O processo criativo deve ser reduzido a um ato de seleção. Projetar significa: selecionar elementos determinantes e combiná-los. Visto nesses termos, projetar clama por método. (GERSTNER, 1964, p.12)¹⁴

Quando Gerstner propõe que “descrever um problema é parte da solução”, é possível identificar alguns paralelos com o pensamento computacional. Em primeiro lugar, a afirmação de Gerstner reproduz o princípio da **decomposição** utilizado na computação. Decomposição significa subdividir um problema em partes menores, ou seja,

¹⁴To describe the problem is part of the solution. This implies: not to make creative decisions as prompted by feeling but by intellectual criteria. The more exact and complete these criteria are, the more creative the work becomes. The creative process is to be reduced to an act of selection. Designing means: to pick out determining elements and combine them. Seen in these terms, designing calls for method (GERSTNER, 1964, p.12).

descrever um problema em termos de suas menores partes. E a decomposição é “parte da solução” porque é uma etapa essencial na construção de um **algoritmo** para a resolução do problema. Em seguida, quando fala em “critérios intelectuais”, é possível entender esses “critérios” como **parâmetros** ou **variáveis** de um algoritmo. A seleção e combinação de elementos pode funcionar como uma **simulação**, ou uma forma de experimentação. Se essas permutações seguirem mesmo um algoritmo, podem ainda ser **automatizadas**. O método de design proposto por Gerstner incorpora conceitos análogos a conceitos da computação, mesmo que esse não tenha sido seu objetivo. Ainda assim, seu método é voltado à criação, e assume o fator humano na descrição do problema, na determinação dos critérios, e na seleção dos resultados permutados. Quando se fala em fórmula e forma, trata-se da busca pelo design de “uma série de soluções” (GREDINGER, 1964, p.5). A grande contribuição dessa abordagem combinatoria de design é a possibilidade de se obter “uma série de soluções” a partir de um mesmo programa.

Até aqui, este capítulo discutiu a arte generativa e a teoria de Gerstner para iluminar algumas aproximações e intersecções existentes entre arte, design, e computação. Embora essas relações não sejam necessariamente diretas, elas apontam para uma compreensão do design computacional.

2.3 Design computacional

Embora nem LeWitt, nem Riley, nem Gerstner tivessem envolvimento direto com a computação, seus trabalhos exibem conceitos muito próximos aos utilizados na computação¹⁵. Wing fala que o pensamento computacional pode existir sem a manipulação direta de um computador, e os trabalhos de LeWitt, Riley, e Gerstner demonstram isso na prática. Seus trabalhos apresentam aspectos de natureza computacional mesmo sem a utilização de computadores. O que esta dissertação propõe é justa-

¹⁵Pode-se questionar se esses conceitos da computação (abstrações, algoritmos, permutações) não são anteriores à própria computação. No entanto, essa é uma grande questão que não cabe nesta dissertação. Essa é uma discussão mais adequada para os especialistas da matemática, da lógica, e da computação. Independente desses conceitos serem anteriores à computação ou não, o que se pode afirmar é que a computação é construída em cima desses conceitos.

mente que a mera utilização do computador não caracteriza um design computacional. Tendo em vista que a maior parte dos trabalhos criativos atualmente são mediados pela computação (MANOVICH, 2013), observa-se que a maior parte dos profissionais já trabalha com a computação, por meio de pacotes de software comerciais. No entanto, se o computador é utilizado apenas como simulador de outras mídias, seu potencial enquanto máquina computacional é pouco aproveitado. Dessa forma, propõe-se que o que caracteriza o design computacional é uma utilização **programática** da computação nos processos de design.

No design computacional, o processo de design se torna indireto, pois ao invés do designer trabalhar diretamente no produto final (seja um cartaz, uma cadeira, ou um carro) do projeto, é elaborado um sistema de regras (algoritmo) que, ao ser colocado em prática, gera várias soluções. Uma destas soluções pode ser escolhida como resultado final do projeto. Caso a solução selecionada não seja satisfatória, é possível alterar o próprio sistema de regras, ou os parâmetros alimentados nesse sistema, de modo a obter outras soluções. Em outras palavras, o design computacional adota uma prática generativa no design, acarretando em uma transformação não só tecnológica, mas também conceitual da atividade de projeto. O design computacional possui uma visão abrangente que não foca apenas no objeto de projeto, mas que compreende também os processos (inclusive os computacionais) empregados no processo de projeto. Esta observação encontra ecos no artigo *Generative design: a paradigm for design research*, dos pesquisadores de arte generativa Jon McCormack, Alan Dorin e Troy Innocent:

[Sistemas generativos] envolvem a reconsideração do artefato estático e as ações que o manipulam. A conceitualização migra da primazia dos objetos para o vislumbre de componentes, sistemas e processos interagentes, que por sua vez geram novos artefatos, com propriedades especiais (...). (MCCORMACK, DORIN, INNOCENT, 2004, p.1)¹⁶

Ou seja, por meio da prática generativa, o design computacional valoriza as relações dinâmicas dos elementos, sistemas, e processos que envolvem aquilo que é pro-

¹⁶For designers, it involves a reconsideration of the static artefact and the actions that manipulate it. Conceptualisation shifts from the primacy of objects to envisaging interacting components, systems and processes, which in turn generate new artefacts, with special properties (...) (MCCORMACK, DORIN, INNOCENT, 2004, p.1).

jetado, e não apenas o resultado desse projeto como ente estático. Além disso, essa abordagem pode resultar em novos artefatos, com novas características. A abordagem sistêmica do processo criativo não significa uma redução do papel do designer no processo de projeto. Significa que designers passam a manipular (mas não necessariamente controlar) uma série de fatores adicionais. Nesse sentido, o design computacional adota uma postura semelhante à do meta-design, proposto pelos pesquisadores Gerhard Fischer e Elisa Giaccardi (2004). Para Fischer e Giaccardi, meta-design significa uma transformação do próprio processo de projeto em objeto de projeto:

Meta-design é uma perspectiva útil para projetos nos quais “projetar o processo de projeto” é uma atividade de primeira classe (...). Isto quer dizer que criar as condições técnicas e sociais para ampla participação em atividades de design é tão importante quanto criar o próprio artefato. (FISCHER, GIACCARDI, 2004, p.6)¹⁷

E como apontam McCormack, Dorin, e Innocent (2004), o grande desafio do trabalho de projeto usando métodos generativos é a necessidade de lidar com relações entre diversos elementos como o contexto, o processo, e o próprio objeto de projeto. Se na abordagem tradicional, o designer explora um espaço de soluções, numa abordagem generativa ele explora um espaço de problemas. E ao lembrarmos que o pensamento computacional consiste em um conjunto de conceitos para resolução de problemas, concluímos que pensamento computacional e métodos generativos atuam no mesmo espaço conceitual, o espaço de problemas. Em design tradicional, o designer busca uma solução; existe uma relação direta entre designer e objeto. Em design com métodos generativos, o designer manipula regras e sistemas; ou seja, manipula problemas. A relação entre designer e objeto se torna indireta. Dessa forma:

O processo de design se torna um de meta-design no qual um design final é resultado das propriedades emergentes de um sistema interagente (McCormack, Dorin, 2001). A “arte” de projetar desta maneira está em dominar a relação entre especificação de processo, ambiente e artefato gerado.

¹⁷Meta-design is a useful perspective for projects for which “designing the design process” is a first-class activity (...). This means that creating the technical and social conditions for broad participation in design activities is as important as creating the artifact itself (FISCHER, GIACCARDI, 2004, p.6).

(...) O papel do designer humano permanece, assim como no design convencional, central para o processo de design. (MCCORMACK, DORIN, INNOCENT, 2004, p.7)¹⁸

Desta maneira, o design computacional não aponta para uma redução do papel do designer no projeto, e sim para uma transformação desse papel. No entanto, tendo isso em vista, é interessante observar que existe um certo receio por parte de alguns artistas e designers em relação à computação. A preocupação é que a computação se torne uma ameaça à profissão de designer, por automatizar seu trabalho, eliminando ou descaracterizando a profissão. Sobre esse sentimento, o pesquisador do MIT Henry Lieberman diz:

Para muitos designers gráficos e artistas, a ideia de computadores entrando no processo de design era vista como ameaçadora. A exploração de formas visuais e o senso de comunicação em design visual eram vistos como essencialmente e unicamente humanos, e a incursão do computadores no processo era vista por muitos como um rebaixamento do papel do profissional criativo. (LIEBERMAN, 1996, p.39)¹⁹

Mesmo com a crescente presença da computação, essa é uma postura que ainda se pode encontrar nos dias de hoje. No entanto, como o próprio Lieberman observa, o potencial do computador é reconhecido por tantos outros designers. No seguinte trecho, Lieberman fala do computador como “parceiro inteligente” do designer, remetendo à simbiose de Licklider, e ao depoimento de Mohr tratando o computador como “parceiro”:

Eles acolheram ferramentas que poderiam auxiliar um designer em relacionar forma e significado, em explorar e avaliar alternativas. Eles viram

¹⁸The design process becomes one of meta-design where a finished design is the result of the emergent properties of the interacting system (McCormack & Dorin 2001). The ‘art’ of designing in this mode is in mastering the relation between process specification, environment, and generated artefact. (...) The role of the human designer remains, as with conventional design, central to the design process (MCCORMACK, DORIN, INNOCENT, 2004, p.7).

¹⁹For many graphic designers and artists, the idea of computers entering the design process was viewed as threatening. The exploration of visual forms and sense of communication in visual design was viewed as essentially and uniquely human, and the incursion of the computer into the process was viewed by many as demeaning the role of the creative professional (LIEBERMAN, 1996, p.39).

a possibilidade do computador se tornar um **parceiro inteligente** no processo de design, para ampliar o alcance e o poder do designer humano. (LIEBERMAN, 1996, p.39, grifo nosso)²⁰

Ou seja, uma importante contribuição da computação para o design é a exploração de variações, que pode levar à obtenção de resultados inesperados. Com o imprevisto fazendo parte do processo de design, o designer pode expandir seu repertório de soluções. Uma maneira de um artista ou designer encontrar originalidade e invenção²¹ em seu trabalho é abdicando do controle total sobre esse trabalho. A relação entre criação e controle revela uma aparente contradição do processo criativo: para criar algo original, o autor abre mão do controle total sobre sua obra. Um artista busca inspiração em coisas que estão fora de seu domínio, busca transcender suas próprias limitações, para assim criar uma obra que supere seus próprios pré-conceitos. Como pode um artista ou um designer criar uma trabalho que supere suas próprias especificações?

No sentido de design, é possível criar sistemas criativos que exibam propriedades emergentes além das intenções conscientes do designer, desse modo criando um artefato, processo ou sistema que é “mais” do que foi concebido pelo designer. (MCCORMACK, DORIN, 2001, p.9)²²

A ampliação do espaço de soluções representa uma certa perda de controle do designer sobre o objeto de projeto. Para Fischer e Giaccardi, essa não é apenas uma preocupação supérflua, e sim uma etapa necessária para que o design possa agir em um mundo cada vez mais dinâmico e em constante mudança²³. Os fenômenos emergentes e os imprevistos não devem ser reprimidos no processo de projeto, e sim incorporados

²⁰They welcomed tools that could assist a designer in relating form and meaning, in exploring and evaluating alternatives. They saw the possibility for the computer to become an intelligent partner in the design process, to extend the reach and grasp of the human designer (LIEBERMAN, 1996, p.39).

²¹Não se pretende pressupor que originalidade e a invenção são objetivos de qualquer artista. Há infinitas motivações para a arte, que não necessariamente contemplam a originalidade e a invenção.

²²In a design sense, it is possible to make creative systems that exhibit emergent properties beyond the designer's conscious intentions, hence creating an artefact, process, or system that is “more” than was conceived by the designer (MCCORMACK, DORIN, 2001, p.9).

²³Discussões filosóficas a respeito desse crescente dinamismo das mudanças sociais pode ser encontrada nas obras *Tempos hipermodernos* de Gilles Lipovetsky e *Modernidade líquida* de Zygmund Bauman.

como ferramentas para que o design possa propor soluções dinâmicas para problemas dinâmicos.

Em um mundo que não é previsível, improvisação, evolução, e inovação são mais do que artigos de luxo: são necessidades. O desafio do design não é uma questão de se livrar do emergente, mas sim de inclui-lo e fazer dele uma oportunidade para soluções mais criativas e adequadas para problemas. (FISCHER, GIACCARDI, 2004, p.5)²⁴

Neste capítulo, tratou-se da arte generativa, da teoria dos programas em design de Gerstner, e do design computacional propriamente dito. Se a questão do controle do artista sobre a obra levanta um debate sobre a autoria na prática generativa em arte, no design a diminuição do controle do designer sobre o projeto levanta ainda outra questão. Como a prática generativa permite a emergência de soluções não pensadas anteriormente, ela tem grande utilidade em design. A prática generativa, em sua essência, não é tecnológica, muito menos formalista, embora muitas vezes seja utilizada como tal. Um processo generativo de design (projeto) é um processo de meta-design, no qual as regras e os sistemas envolvidos nesse processo são tão importantes quanto o objeto de projeto. Dessa forma, abraça a falta de controle e amplia o horizonte de soluções possíveis. Tal método de design permite o surgimento de soluções que vão além do que foi especificado pelo designer. Por meio do princípio multiplicativo da matemática combinatória, é possível verificar que quanto maior o número de interações entre as partes de um sistema generativo, maior o número de resultados esse sistema pode gerar. Ou seja, o espaço de soluções é potencialmente maior que o espaço de conhecimento do designer. Em contraste, no método rígido e controlado, o espaço de soluções é necessariamente menor que o espaço de conhecimento do designer. Design computacional implica explorar um espaço de problemas ao invés de um espaço de soluções. Em conjunto com o poder de processamento de informação dos computadores, a utilização de práticas generativas permite a exploração de um número muito

²⁴In a world that is not predictable, improvisation, evolution, and innovation are more than luxuries: they are necessities. The challenge of design is not a matter of getting rid of the emergent, but rather of including it and making it an opportunity for more creative and more adequate solutions to problems (FISCHER, GIACCARDI, 2004, p.5).

grande de possibilidades. A busca por menos controle reflete a busca pela ampliação do número de soluções possíveis.

O próximo capítulo traz depoimentos de designers computacionais, que trabalham programaticamente com a computação. Entre os depoimentos, estão entrevistas com 3 designers brasileiros que exploram as relações entre computação e design em seus trabalhos.

Capítulo 3

Práticas programáticas

Este capítulo tem como objetivo verificar como a inclusão da computação nos processos de design se reflete nas práticas dos designers, especificamente aqueles que trabalham programaticamente com a computação. Para isso, foram coletados depoimentos de designers (e artistas) que trabalham dessa maneira. Buscou-se compreender o que motivou esses designers a adotarem a computação em seus processos, e como essa abordagem computacional afeta seus trabalhos. Depoimentos de designers estrangeiros foram obtidos na pesquisa bibliográfica, especialmente nos livros *Creative code* (MAEDA, 2004) e *Processing: a programming handbook for visual designers and artists* (FRY; REAS, 2007). Outra fonte importante consultada nesta pesquisa é a coleção de edições da revista *Computer graphics and arts*, publicadas no site *CompArt Dada*¹, da Universidade de Bremen (Alemanha). Por outro lado, não foram encontrados depoimentos de designers brasileiros contemporâneos sobre este tema. As referências nacionais encontradas são textos dos artistas pioneiros da artemídia brasileira Waldemar Cordeiro e Julio Plaza, que discutem questões como as telecomunicações e a interatividade.

Waldemar Cordeiro organizou a exposição de arte eletrônica brasileira *Arteônica* em 1971. Seu trabalho também teve alcance internacional, com sua participação na seminal exposição internacional de arte eletrônica *Cybernetic Serendipity* em 1968, e com a contribuição (póstuma) de uma imagem de capa para a revista *Computer Graphics and Art*, em sua edição de agosto de 1976. A figura 3.1 apresenta uma imagem dessa capa.

¹http://dada.compart-bremen.de/browse/publication?filter_type=item_type&filter_value=magazine.

A imagem foi criada como parte da série *Gente*, produzida por Cordeiro em conjunto com Raul Fernando Dada e J. Soares Sobrinho, no Centro de Computação do Instituto de Arte da Unicamp. As outras imagens da mesma série foram exibidas na exposição *Arteônica*, organizada pelo próprio Cordeiro. No manifesto que escreveu para a exposição, Cordeiro enfatiza a importância dos meios de comunicação eletrônicos para a difusão da cultura e da arte no Brasil. Diz Cordeiro: “através do espaço continental do país, a informação artística poderia atingir os lugares mais remotos e isolados do território muito antes do que os equipamentos materiais da civilização” (CORDEIRO, 1971, p.23). Ele fala que a ocupação territorial do país é caracterizada por aglomerações de alta densidade separadas por áreas de baixíssimas densidades populacionais. Esse contraste, e os problemas resultantes dele, poderia ser aliviado pelas telecomunicações. Cordeiro viu nas mídias eletrônicas um meio de fortalecer uma cultura artística brasileira. E destaca a particularidade do movimento da arte concreta brasileira, que teria sido “a única que utilizou métodos digitais para a criação” (CORDEIRO, 1971, p.23).

No texto *Arte e interatividade: artista - obra - recepção* (2000), o artista e professor Julio Plaza apresenta um panorama de autores que discutem a questão da abertura da obra. Para Plaza, existem 3 graus de abertura da obra de arte: o primeiro grau (aquele apresentado por Umberto Eco em seu conceito de obra aberta) é o da ambiguidade de significado embutida na obra pelo artista. No segundo grau de abertura, a obra se abre para a participação ativa do público, que é convidado a manipular e explorar a obra. No terceiro grau de abertura, que representa a arte interativa, o público participa da obra por meio de interfaces técnicas. Nesse tipo de arte, o artista e a obra “só existem pela participação efetiva do público, o que torna a noção de ‘autor’, consequentemente, mais problemática” (PLAZA, 2000, p.19). Dessa forma, o papel central da interação do público na obra requer uma redefinição do próprio conceito de artista. Plaza destaca que a questão da “dissolução do autor” não surgiu com a arte interativa; ela já era abordada nas discussões da “intertextualidade”. A discussão que a arte interativa põe em pauta é que artista, obra, e receptor não são conceitos tão separados e distintos entre si, mas que eles “cruzam-se, confundem-se, e se contaminam.” (PLAZA, 2000, p.26).

Figura 3.1: Waldemar Cordeiro. *Gente - Grau 3* (1972) na capa da revista *Computer Graphics and Art* (v.1, n.3, ago. 1976). Fonte: *Computer Graphics and Art*, v.1, n.3, ago. 1976.

Os textos de Cordeiro e Plaza são importantes na história da artemídia brasileira, mas pertencem a um outro contexto histórico e não lidam especificamente com a questão da programação no processo de criação e projeto. Para discutir essa questão no contexto brasileiro contemporâneo, foram conduzidas entrevistas com 3 designers. Foram entrevistados os designers Hugo Cristo Sant’anna, Gustavo Ferreira, e Dimitre Lima. Esses profissionais foram selecionados porque possuem formação em design, refletem sobre a aplicação da computação em design, e desenvolvem trabalhos em design computacional. Sant’anna possui um vasto portfolio de trabalhos em design computacional e pesquisa o tema na Universidade Federal do Espírito Santo (Ufes). Ferreira explora a computação em seus trabalhos de design tipográfico, e oferece workshops de programação para designers tipográficos. Lima é designer gráfico e participou de trabalhos de grande alcance nacional e internacional.

A seguir, são apresentados e discutidos os depoimentos de designers estrangeiros.

3.1 Depoimentos

Foram identificados alguns temas recorrentes nos depoimentos pesquisados. O primeiro tema verificado é a importância da computação ao habilitar o designer a construir suas próprias ferramentas, criando seu próprio software. Em seguida, o segundo tema identificado é o uso da computação para exploração de fenômenos emergentes em processos autônomos; ou seja, a busca por resultados incertos a partir de sistemas complexos. Por fim, o último tema observado é uma visão filosófica da computação como extensão da mente.

3.1.1 Espaço das ideias e espaço das ferramentas

O pesquisador e professor americano A. Michael Noll é um dos pioneiros da arte computacional, tendo atuado como pesquisador no Bell Labs e como conselheiro de Ciências do governo dos EUA². Já nos primórdios da arte computacional, na década de 60, Noll observava que a computação não era levada muito a sério como uma ferramenta

²<http://noll.uscannenberg.org/>

criativa inovadora:

A noção de criar arte por meio de máquinas pode parecer um pouco estranha. A maioria das pessoas que ouviram falar do uso experimental de computadores digitais em esforços criativos provavelmente menosprezou como sendo de nenhuma consequência. (NOLL, 1967, p.89)³

Embora esse depoimento tenha sido publicado em 1967, esse tipo de reação pode ser observada ainda hoje. Muitas vezes, a computação ainda é vista apenas como uma simples ferramenta, como mera simulação de outras mídias. Essa visão limitada do potencial da computação passa também pela predominância de alguns poucos programas comerciais na prática profissional do design. Ou seja, entende-se que as possibilidades da computação são restritas às possibilidades oferecidas por tais programas.

Um dos trabalhos mais conhecidos de Noll, e que levanta questões a respeito das possibilidades criativas da computação, é o experimento que fez com a composição apresentada na figura 3.2.

Figura 3.2: A. Michael Noll. *Computer composition with lines*, 1964. Fonte: NOLL, 1964, p.2.

³The notion of creating art works through the medium of machines may seem a little strange. Most people who have heard about the experimental use of digital computers in creative endeavors have probably shrugged them off as being of no consequence (NOLL, 1967, p.89).

Esta composição é baseada no quadro *Composição com linhas* (1917) de Piet Mondrian. Noll criou um programa de computador para produzir uma composição semelhante à de Mondrian, mas na qual as posições e as dimensões dos retângulos que formam a imagem foram determinadas aleatoriamente. Noll realizou uma pesquisa comparativa entre as 2 composições, exibindo reproduções das 2 imagens para um grupo de 100 pessoas, que deveriam identificar qual das composições havia sido gerada computacionalmente, e indicar qual das duas imagens elas preferiam. Os resultados da pesquisa mostraram que a maioria dos participantes não conseguiu identificar corretamente a imagem gerada por computador, e que a maioria preferiu a imagem gerada por computador (NOLL, 1966). Ao apresentar os resultados de sua pesquisa, Noll ressaltou que a preferência demonstrada pelos participantes da pesquisa não deveria ser entendida como sendo relacionada ao fato da imagem ter sido produzida por computador, e sim ao maior grau de aleatoriedade que a imagem computacional exibia. De qualquer forma, este experimento de Noll despertava, na década de 60, importantes reflexões a respeito da participação do computador na produção de imagens artísticas.

Outra figura importante que comentou a respeito dessa postura conservadora em relação à computação no design é o designer e professor americano John Maeda.

Maeda é uma das figuras mais importantes na história recente do design computacional. Em 1999, Maeda lançou o projeto *Design by numbers* (ou DBN)⁴, que consistia em uma linguagem e em um ambiente de programação. O objetivo do DBN era introduzir conceitos da computação para designers e artistas. Desse modo, sua linguagem de programação foi projetada para ser descomplicada, fácil de entender para pessoas não-familiarizadas com a programação de computadores. Uma consideração importante no projeto do DBN era que ele fosse acessível. Assim, o DBN foi disponibilizado gratuitamente pela internet, com versões disponíveis para os vários sistemas operacionais existentes. Além disso, o DBN permitia que os programas produzidos nele fossem compartilhados na internet na forma de *applets* Java, incorporáveis em qualquer site. Além do DBN, Maeda foi docente do MIT, onde atuou com o grupo de pesquisas *Aesthetics and Computation Group* (ACG). Assim como Noll comentou acerca do desprezo da computação como ferramenta criativa, Maeda recorda:

⁴<http://dbn.media.mit.edu/>.

Eu consumi muitos esforços no início da minha carreira tentando negar a crença convencional, nascida nos anos 80, de que o computador não passava de um novo tipo de lápis. (MAEDA, 2004, p.113)⁵

Para Maeda, a computação não é apenas um “novo tipo de lápis”, ou seja, não é apenas uma nova forma para um instrumento já existente. Em outras palavras, a computação não é apenas uma simulação de outras mídias, como o lápis. Percebe-se então uma relação entre o argumento de Maeda e a ideia de meta-mídia de Kay. Essa relação também pode ser observada quando Maeda fala da particularidade da mídia computacional:

Computação é intrinsicamente diferente de outras mídias porque é a única mídia na qual o material e o processo para modelar o material coexistem na mesma entidade: números. (MAEDA, 2001, p.251)⁶

Isto é, a computação é baseada em números e operações matemáticas que manipulam esses números. Isto confere à computação uma característica “meta-material”, pois tanto o programa de computador, quanto os dados utilizados por esse programa, consistem de números. Por exemplo, um vídeo armazenado em um computador consiste de uma sequência de bytes⁷. Para reproduzir esse vídeo, é necessário utilizar algum software, que também é constituído por bytes. Ou seja, tanto o vídeo quanto o programa que reproduz esse vídeo são formados por bytes. Desse modo, pode-se dizer que o que torna o computador uma meta-mídia é essa natureza “meta-material”, na qual representações e programas são baseadas em operações numéricas. Para Maeda, o domínio da computação seria uma condição importante para o desenvolvimento de um design mais verdadeiro à natureza das mídias digitais. Essa foi a motivação que o levou a desenvolver o DBN, cujo nome (“design por meio de números” em português) se refere ao papel central dos números na computação.

⁵I expended much effort in my early career trying to negate the conventional belief, borne in the 1980s, that the computer was nothing more than a new kind of pencil (MAEDA, 2004, p.113).

⁶Computation is intrinsically different from existing media because it is the only medium where the material and the process for shaping the material coexist in the same entity: numbers (MAEDA, 2001, p.251).

⁷Bytes são sequências de 8 bits. Um bit é a unidade mais básica de informação em sistemas computacionais, e consiste em um dígito binário, cujo valor pode ser 0 ou 1.

Um trabalho que exemplifica a abordagem numérica de Maeda em design gráfico pode ser visto no pôster que ele desenhou para a marca de cosméticos japonesa Shiseido em 1995 (figura 3.3). A composição visual do pôster consiste em uma área quadrada apoiada sobre uma superfície preta, e um conjunto de curvas douradas ao lado. Essa composição é resultado de uma interpretação gráfica do banco de dados dos filmes comerciais da Shiseido. O quadrado é composto por *frames* (quadros) dos filmes, mostrando a evolução desses comerciais ao longo do tempo. A superfície preta na parte inferior é uma representação gráfica das trilhas sonoras dos comerciais. E as curvas douradas são desenhadas em função dos códigos alfanuméricos das letras que compõem os títulos dos comerciais. Pode-se identificar neste trabalho (de 1995) um precedente para as chamadas “visualizações de dados” interativas e dinâmicas que têm se tornado mais comuns recentemente. Este pôster mostra como a computação contribui para o designer pensar formas a partir de bancos de dados. Mas, a composição numérica não prescinde da sensibilidade humana, visível na escolha das formas e na disposição delas no campo visual deste pôster.

A atuação de Maeda com o Design by numbers têm um papel importante na história recente do design computacional, pois inspirou o desenvolvimento da plataforma Processing por dois de seus alunos no MIT: Ben Fry e Casey Reas. O Processing foi lançado em 2001 e por sua vez inspirou o desenvolvimento de tantas outras plataformas voltadas para artistas e designers computacionais, como o Nodebox⁸, DrawBot⁹, OpenFrameworks¹⁰, e Cinder¹¹.

O que torna o Processing (figura 3.4) mais acessível para designers e artistas é sua simplicidade em relação às linguagens de programação mais tradicionais. No Processing, é possível executar um programa com o clique de um botão. Outras linguagens exigem a criação de um “ambiente de programação” mais complexo, com um número de componentes que devem ser configurados. Além disso, o Processing automatiza certas tarefas elementares, simplificando o código que precisa ser escrito para criar aplicações gráficas e interativas. Por exemplo, para desenvolver uma aplicação gráfica

⁸<http://nodebox.net/>

⁹<http://drawbot.readthedocs.org/>

¹⁰<http://openframeworks.cc/>

¹¹<http://libcinder.org/>

Figura 3.3: John Maeda. Pôster comemorativo do aniversário de 30 anos de comerciais da marca japonesa de cosméticos Shiseido, 1995. Fonte: site de John Maeda. Disponível em: <http://www.maedastudio.com/1995/shisposter/>.

Figura 3.4: Processing. Fonte: imagem do programa em execução.

em Java, é necessário escrever algumas linhas de código apenas para fazer aparecer uma janela gráfica no computador. Com o Processing, o programador precisa escrever apenas 1 linha de código, informando as dimensões da janela desejada. O Processing fornece um grande número de comandos que facilitam a produção de imagens e a programação de interações. Outro fator importante para a difusão do Processing como ferramenta criativa foi a construção de uma comunidade forte de usuários. O site do Processing apresenta tutoriais para os usuários iniciantes, apresenta um fórum de discussões onde usuários compartilham informações e dúvidas, e apresenta também um espaço no qual usuários avançados contribuem com bibliotecas de código que expandem as funcionalidades embutidas no Processing. Sobre a motivação para a criação do Processing, Fry diz:

Um dos meus objetivos pessoais para este projeto é facilitar a tomada de controle de suas próprias ferramentas pelo designer. Faz mais de 20 anos desde que a *desktop publishing* ajudou a reinventar o design na metade da década de 80, e estamos atrasados em mais inovações. Assim que os designers se cansaram das ferramentas disponíveis, a programação começou a preencher a distância entre o que está na mente do designer e a capacidade do software que eles compraram. Enquanto a maioria dos usuários do Processing irão aplicá-lo em seus próprios trabalhos, eu espero que ele [Processing] também ajude outros [designers] a criar novas ferramentas de design que provém não de corporações ou de cientistas da computação, mas dos próprios designers. (FRY; REAS, 2007, p.xxv)¹²

Pode-se identificar ideias de Kay e de Flusser neste depoimento de Fry. Uma das premissas centrais no projeto Dynabook de Kay era a possibilidade do seu usuário programar novos tipos de conteúdos para o dispositivo. Para Kay, o domínio de um computador pelo seu usuário está associado com a capacidade desse usuário de programar

¹²One of my personal goals for this project is to facilitate designer's taking control of their own tools. It's been more than twenty years since desktop publishing helped reinvent design in the mid-1980s, and we're overdue for more innovations. As designers have become fed up with available tools, coding and scripting have begun to fill the widening gap between what's in the designer's mind and the capability of the software they've purchased. While most users of Processing will apply it to their own work, I hope that it will also enable others to create new design tools that come not from corporations or computer scientists, but from designers themselves (FRY; REAS, 2007, p.xxv).

o computador; essa seria a base de um “letramento computacional”. Assim como Kay projetou a linguagem Smalltalk com o objetivo de permitir que crianças programassem o Dynabook, a linguagem Processing foi projetada tendo em vista a utilização por designers e artistas. Dessa forma, a linguagem traz comandos simples para a produção de imagens e sons. A possibilidade de um designer programar representa um maior domínio da computação por esse designer. E aqui é possível identificar um paralelo com a filosofia da caixa preta de Flusser. Quando um designer trabalha com um software comercial de criação gráfica, ele trabalha dentro das limitações desse software. Nessa situação, o software é uma caixa preta pré-programada, e o designer trabalha dentro do universo de possibilidades inscrito dentro dele. Além disso, é interessante observar que os programas de criação têm se tornado cada vez mais complexos e difíceis de usar. Para Maeda, esse grau de complexidade dos programas pode se tornar tão alto a ponto de constituir obstáculos para a criatividade (MAEDA, 2004). A complexidade do software contribui para tornar a caixa preta ainda mais indecifrável. No entanto, quando o designer passa a produzir seu próprio software e suas próprias ferramentas, a caixa preta da computação se torna mais branca. O designer passa a definir seus próprios limites, ao invés de trabalhar dentro dos limites impostos pela corporação ou pelo programador que desenvolveu algum software. Um depoimento que sintetiza em poucas palavras esta preocupação com a invenção de ferramentas pelo próprio designer é o do artista John Simon Jr., em artigo publicado no livro *Creative code* organizado por Maeda:

A razão óbvia (...) para um artista ou designer programar é quebrar os limites das ferramentas comerciais. Programação criativa oferece a possibilidade de ativar seus próprios modelos e inventar novos tipos de software. (SIMON JR, 2004, p.46)¹³

Ainda nesse tema, outro depoimento importante é o do designer holandês Erik van Blokland, que ao lado de Just van Rossum, forma a Letterror. A Letterror atua desde 1989, principalmente no campo do design tipográfico digital. Deve-se ressaltar que o

¹³The obvious reason, (...) for an artist or designer to program is to break the boundaries of commercial tools. Creative programming offers the possibility of activating your own models and inventing new kinds of software (SIMON JR, 2004, p.46).

desenho tipográfico é um campo do design com uma vasta tradição histórica, e que a Holanda é um dos países mais tradicionais nesse campo. Uma das características principais da obra da Letterror é a fusão dessa tradição tipográfica com a prática computacional. O texto abaixo faz parte de um artigo escrito por van Blokland em parceria com o escritor e designer Jan Middendorp. No seguinte trecho, Middendorp e van Blokland falam sobre as limitações dos programas comerciais:

Software é um produto comercial de massa, então ele não é feito para indivíduos, mas para um grupo-alvo com desejos e requisitos hipotéticos. Se na prática acontecer do grupo-alvo ter desejos não-atendidos, funções extras são adicionadas com um upgrade, mas apenas se houver interesse suficiente. Você pode fazer tudo com um programa, enquanto houver um número suficiente de pessoas que queiram fazer a mesma coisa. Nada disso levanta problemas para o usuário médio, que normalmente não exige do computador mais do que ele é capaz de fazer, e que na verdade encontra segurança em sua capacidade limitada. Esse não deve ser o caso para usuários em uma profissão criativa. É precisamente a tarefa de designers (gráficos, industriais, etc) descobrir novas possibilidades. (MIDDENDORP; VAN BLOKLAND, 2000, p.23)¹⁴

Ou seja, programas comerciais (como Photoshop e Illustrator) são produtos feitos para atender às necessidades de um grupo amplo de profissionais. Nesse sentido, a adição (ou remoção) de uma função a esses programas depende das demandas coletivas desse grupo. Os fabricantes desses programas não têm interesse em atender a necessidades individuais de um ou outro membro desse grupo. Novamente, esta observação remete ao discurso de Flusser em relação à programação e aos limites das caixas pretas. Mas, se Middendorp e van Blokland até admitem que as limitações dos

¹⁴Software is a commercial mass product, so it is not made for individuals but for a target group with hypothetical wishes and demands. If in practice that target group turns out to have unfulfilled desires extra functions are added with an upgrade, but only if there is enough interest. You can do everything with a program as long as there are enough people who want to do the same thing. None of this raises any problems for the average user, who usually does not demand more of the computer than it is capable of doing, and in fact finds reassurance in its limited capacity. That should not be the case for users in a creative profession. It is precisely the task of designers (graphic, industrial or whatever) to discover new possibilities (MIDDENDORP; VAN BLOKLAND, 2000, p.23).

programas comerciais não é necessariamente um empecilho em outras atividades profissionais (ao contrário de Flusser), eles defendem que a busca pela ruptura dos limites, e consequentemente pela descoberta de novas possibilidades, é uma responsabilidade de designers e de outros profissionais criativos. E é justamente na tradição do design holandês que eles vão buscar uma justificativa:

Artesãos do passado – os predecessores de nossa guilda de designers – raramente estavam satisfeitos com as ferramentas da forma como eles encontravam nas lojas. Eles sempre tinham a tendência de personalizar suas ferramentas, para apropriá-las afiando, convertendo, ou expandindo elas. Quanto mais especializado o trabalho, maior a demanda por instrumentos customizados ou individualmente fabricados. Por exemplo, tipógrafos do passado pensaram métodos para trabalhar mais rapidamente e mais meticulosamente, e para esse fim eles projetaram não apenas novas fontes, mas também contrapunches e outras novas ferramentas. Deve ser possível fazer algo parecido com software. Afinal, programar programas gráficos é muito importante para deixar para programadores. (MIDDENDORP; VAN BLOKLAND, 2000, p.25)¹⁵

Abraçar suas ferramentas, questionando os seus limites, e explorando seus potenciais ao máximo, é uma tradição das artes e do design. Compreender a computação em seus múltiplos aspectos, e fugir de uma visão reducionista da computação como mera simulação de outras ferramentas é uma continuação dessa tradição. O desenvolvimento de ferramentas próprias não é uma peculiaridade do meio computacional. Designers com ideias novas frequentemente buscam criar ferramentas novas, e são as pessoas mais indicadas para essa tarefa. Não está se propondo uma revolução do

¹⁵Artisans of the past – the predecessors of our designers' guild — were rarely satisfied with tools as they found them in the shop. They always had the tendency to personalize their tools, to appropriate them by honing them, converting them or expanding them. The more specialized the work, the greater the demand for customized or individually made instruments. For instance, letter-cutters in the past thought up methods of working faster and more meticulously, and to that end they designed not only new fonts, but also counterpunches and other new tools. It must be possible to do something similar with software. After all, programming graphic programs is much too important to leave to programmers (MIDDENDORP; VAN BLOKLAND, 2000, p.25).

design pela computação, e sim uma melhor compreensão de suas características intrínsecas, tendo em vista que a computação é uma das ferramentas mais presentes na prática profissional dos designers.

Um esquema (figura 3.5) que acompanha o artigo de Middendorp e van Blokland ilustra com bastante clareza essa relação da ferramenta com as soluções possíveis aqui discutida.

Figura 3.5: Lettererror. Espaço das ideias versus espaço das ferramentas. Fonte: MIDDENDORP; VAN BLOKLAND, 2000, p.25.

Nesse esquema, o quadrado com fundo azul claro representa o “espaço das ideias”, ou seja, o universo de ideias imagináveis. A área em amarelo é o “espaço da ferramenta”, e a área em verde é “sua própria ferramenta”, a ferramenta criada pelo próprio designer. Neste mapa, há 3 pontos marcados em vermelho. O ponto número 1, que está dentro da área amarela, é uma “ideia possível”, pois é um ponto do espaço de ideias que também é coberto pelo espaço da ferramenta. Já o ponto número 2, que está na área azul, é uma “ideia impossível”, pois ela está fora do espaço da ferramenta. O ponto número 3 representa uma ideia que passa a ser possível graças à construção de uma nova ferramenta, que está fora dos limites da ferramenta original. Em outras palavras, construir ferramentas próprias permite ao designer explorar ideias que não são praticáveis com as ferramentas existentes. Não é o software que deve orientar as ideias, e sim o inverso:

Escrever suas próprias ferramentas faz as ideias direcionarem o desenvolvimento do software, ao invés do contrário. (VAN BLOKLAND, 2007, p.170)¹⁶

A fonte digital *Beowulf*, desenvolvida em 1990 pela Letterror, introduz variações aleatórias nas letras de um texto composto nessa fonte. Na figura 3.6, é possível observar algumas palavras em que letras repetidas apresentam contornos diferentes, simulando falhas de impressão, como por exemplo a letra “n” em “remnant”. Em um texto composto com uma fonte tipográfica tradicional, todas as letras “n” seriam iguais entre si. Com *Beowulf*, cada letra possui um contorno aleatório e único. Os pontos que definem os contornos de cada letra sofrem pequenos deslocamentos, cujos valores são determinados aleatoriamente. Essa fonte foi originalmente executada utilizando os formatos de fontes digitais Type 1 e Type 3, baseados na linguagem de programação Postscript¹⁷, e tirando proveito de brechas existentes nesses formatos que permitiam a inserção de código a ser executado no momento da impressão. No mesmo ano de 1990, a Adobe fechou essas brechas, limitando os recursos disponíveis nas fontes para

¹⁶Writing your own tools makes the ideas direct the development of software, rather than the other way round (VAN BLOKLAND, 2007, p.170).

¹⁷Postscript é uma linguagem de programação voltada para a produção de imagens gráficas, criada pela Adobe na década de 80 e utilizada até hoje nos programas de criação gráfica.

aumentar a velocidade de impressão. Embora essa limitação tenha beneficiado um grande número de usuários, ao mesmo tempo ela eliminou algumas possibilidades criativas. Mesmo com tecnologias mais recentes para fontes digitais, a Letterror conseguiu apenas simular, e não reproduzir, a aleatoriedade do trabalho original.

roar
ravage
speedeth
remnant

Figura 3.6: Letterror. Exemplos de aplicação da fonte tipográfica FF Beowolf. Fonte: site FontFont. Disponível em: <https://www.fontfont.com/fonts/beowolf/in-use>.

Os conhecimentos em computação permitem aos designers proporem novas ferramentas que permitem executar ideias impraticáveis com as ferramentas disponíveis. Um outro tema identificado nos depoimentos pesquisados é que a computação permite a exploração de novas ideias.

3.1.2 Expansão do espaço das ideias

No diagrama da Letterror, que procura imitar o estilo gráfico de um mapa, o espaço de ideias é delimitado por uma borda bem definida. No entanto, talvez fosse mais interessante representar esse espaço de ideias como um espaço sem contornos bem-definidos, cuja área pode ser expandida. Como foi discutido no capítulo anterior, a prática generativa propõe a utilização de sistemas autônomos na arte como uma forma de se obter resultados variáveis e inesperados. Alguns designers apontam esta característica como uma das maiores motivações para uma prática programática / computacional do design.

O artista e programador inglês Ed Burton é conhecido por ter criado o projeto Sodaconstructor¹⁸, que lançou em 2000. Sodaconstructor pode ser descrito como um tipo de brinquedo de montar (como o popular Lego), no qual os elementos a serem montados são simples pontos e linhas. Esses elementos se comportam como simulações de conceitos da física: os pontos se comportam como massas e as linhas se comportam como fios ou molas deformáveis. O usuário que acessa o Sodaconstructor inventa “criaturas” com esses pontos e linhas e observa como elas reagem aos parâmetros da simulação física. Além disso, o Sodaconstructor possui uma plataforma online na qual usuários disponibilizam suas criações para outros usuários. A figura 3.7 mostra um exemplo de criatura criada com o Sodaconstructor. O usuário desenha os pontos e linhas que definem a criatura, e configura uma série de variáveis que determinam o seu movimento. As linhas atuam como molas, que podem se contrair automaticamente, e o usuário determina a velocidade e a amplitude dessas contrações. Além disso, é possível configurar a gravidade e a fricção do ambiente simulado, e a elasticidade das linhas. A computação é utilizada neste projeto para a criação de um sistema interativo, aberto à participação ativa do público, que propicia o surgimento de formas inesperadas.

No Sodaconstructor, formas inesperadas surgem da combinação de elementos simples como pontos e linhas. Da simulação de forças físicas nessas formas surgem movimentos inesperados. Das trocas de informações realizadas pela plataforma online, surgem novas ideias. Para Burton, essa possibilidade de explorar fenômenos emergentes é um reflexo da prática computacional:

¹⁸<http://sodaplay.com/creators/soda/items/constructor>.

Figura 3.7: Ed Burton. Sodaconstructor, 2000. Fonte: site Soda. Disponível em: <http://soda.co.uk/work/sodaconstructor>.

Eu tendo a estar mais interessado no processo do que no produto. Ao invés de criar formas estáticas singulares, eu quero criar processos que podem resultar em incontáveis formas em potencial emergindo ao longo do tempo. Minha motivação é ser surpreendido pelo processo de emergência. Pode soar contraintuitivo que a programação (de computadores) pode ser surpreendente. Programas de computador são construídos de operações lógicas determinísticas, afinal. Entretanto, até as operações mais simples podem ter consequências inesperadas, especialmente quando executadas com feedback e interatividade. Usando programação para tirar proveito de feedback e interatividade é possível ser repetidamente premiado com fenômenos emergentes e gratas surpresas. (BURTON, 2007, p.264)¹⁹

¹⁹I tend to be more interested in process than product. Rather than create singular static forms I want to create processes that may result in countless potential forms emerging over time. My motivation is to be happily surprised by the process of emergence. It may sound counterintuitive that programming

Quando Burton fala em ter mais interesse “no processo do que no produto”, é possível identificar uma relação direta com a prática artística generativa. Além disso, o depoimento de Burton é interessante por demonstrar como a lógica computacional pode gerar resultados imprevisíveis. Embora os programas de computador sejam constituídos por operações determinísticas, é possível induzir fenômenos emergentes no nível do código. Emergência se refere à “revelar, aparecer, ou ‘tornar visível’ um evento, objeto, ou resultado de um processo” (MCCORMACK; DORIN, 2001, p.4). Em fenômenos emergentes, a complexidade **emerge** como resultado das interações entre partes simples de um sistema. Nesse sentido, as instruções de um programa são partes simples, que interagem entre si para produzir resultados complexos e inesperados. Ou seja, não é possível determinar com antecedência o resultado dos processos em andamento em um sistema emergente. Quando um fenômeno emergente gera resultados não-planejados, ele potencialmente expande o universo de ideias (e de soluções de projeto) do designer.

O programador e artista computacional Jared Tarbell compartilha dessa visão. Para Tarbell, o resultado de um algoritmo pode ser mais complexo do que a ideia original por trás desse algoritmo.

Antes de escrever o programa, eu tinha apenas uma vaga ideia do que aquilo poderia parecer. Foi nas primeiras execuções sem erro que me dei conta de que algo incrível estava acontecendo. A forma resultante era muito mais complexa que o algoritmo originador. Esta particular qualidade do software é o que me mantém interessado. (TARBELL, 2007, p.157)²⁰

Assim como Burton, Tarbell percebe que o código computacional é um meio que permite a emergência de resultados inesperados. Uma consequência do código ser construído com instruções individuais é que a formulação da sequência de instruções

can be surprising. Computer programs are constructed from deterministic logical operations, after all. However, even the simplest operations can have unexpected consequences, especially when executed with feedback and interactivity. By using programming to harness feedback and interactivity it is possible to be repeatedly rewarded with emergent phenomena and happy surprises (BURTON, 2007, p.264).

²⁰Before writing the program, I only had a vague idea of what it might look like. It wasn't until the first couple of bug-free executions that I realized something incredible was happening. The resulting form was much more complex than the originating algorithm. This particular quality of software is what keeps me interested (TARBELL, 2007, p.157).

adequada para a obtenção de um determinado resultado não é trivial. Mesmo um código de poucas linhas pode ser de alta complexidade. Muitos programas possuem bugs; ou seja, erros ou comportamentos inesperados / indesejados. Explorar a complexidade da computação para alcançar resultados imprevistos é uma das motivações do design computacional.

Figura 3.8: Jared Tarbell. *Substrate*, 2003. Fonte: página do artista no Flickr. Disponível em: <http://www.flickr.com/photos/generated/416787/in/set-72157594149046982>.

Um trabalho de Tarbell que explora essa qualidade do software para obter formas inesperadas é *Substrate* (figura 3.8). Este trabalho consiste em um programa (feito com Processing) que gera inúmeras variações de imagens a partir de um mesmo sistema de regras. O programa inicia com uma tela em branco, e logo algumas linhas com diferentes rotações começam a preencher a tela. A seguir, surgem novas linhas, que são perpendiculares à alguma linha previamente desenhada. Linhas novas se expandem

até encontrarem alguma linha já desenhada. Cada linha acompanha uma mancha de cor semi-transparente que cria um efeito gráfico similar à pintura com aquarela. A partir desse algoritmo simples, imagens complexas são obtidas. A complexidade resultante remete a imagens de aglomerações urbanas, estruturas moleculares, e pinturas abstratas. Tarbell considera este um de seus trabalhos favoritos, por combinar uma “regra simples, resultados complexos, e enorme potencial para modificações”²¹.

A computação permite ampliar o espaço das ideias (o espaço das soluções), por meio da exploração de ideias que não seriam pensáveis de outra forma. Desse modo, alguns designers computacionais passam a compreender a computação como extensão da mente. Nesse caso, o software faz uma ponte entre mente humana e máquina computacional, resultando na percepção de um espaço mental contínuo entre o homem e o computador.

3.1.3 Espaço contínuo mente-computação

A ideia de que a computação poderia ser uma extensão da mente pode ser encontrada também nos textos de Licklider, Engelbart, Bush, e Kay. Aliás, McLuhan já falava da eletricidade como “extensão tecnológica de nosso sistema nervoso central” (MCLUHAN, 1964, p.336). Rheingold também observava que:

(...) qualquer hacker atual admitirá que a coisa mais fascinante em sua vida é sua própria mente, e dirá que considera uma interação prolongada e intensa com um programa de computador como um tipo de diálogo particularmente satisfatório com seus próprios pensamentos. (RHEINGOLD, 1984)²²

Ou seja, um programa de computador pode ser visto como um tipo de “espelho” dos pensamentos de seu programador. Quando Rheingold fala em “diálogo”, remete mais uma vez à proposta de simbiose de Licklider, na qual o computador atuaria quase como um “segundo cérebro”, que responderia ativamente às interações que recebe. A

²¹<http://www.complexification.net/gallery/machines/substrate/>

²²(...) any later-day hacker will admit that the most fascinating thing in his own life is his own mind, and tell you that he regards intense, prolonged interaction with a computer program as a particularly satisfying kind of dialogue with his own thoughts (RHEINGOLD, 1984).

semelhança do funcionamento dos aparelhos com o pensamento humano também foi observada por Flusser:

Em suma: aparelhos são caixas pretas que simulam o pensamento humano, graças a teorias científicas, as quais, como o pensamento humano, permutam símbolos contidos em sua “memória”, em seu programa. Caixas pretas que brincam de pensar. (FLUSSER, 2002, p.17)

Ou seja, caixas pretas “brincam de pensar” em função de um programa que está embutido nelas. No caso do computador, o programa embutido nele pode ser criado pelo usuário do computador. Sendo assim, pode-se considerar a possibilidade do homem voltar a “possuir” o aparelho. Para o professor e artista computacional Golan Levin, o computador pode se tornar inseparável da mente humana:

Gosto da pergunta: “Em sua mente, o computador é somente uma ferramenta?”, porque minha resposta está na pergunta; eu acredito que mente e computador são coextensivos. (...) Se a escrita é um veículo para o pensamento, então o software é um agente de vontade. Quando ele executa minha vontade, software e eu formamos uma única, coextensiva unidade de pensamento e propósito. (...) Nossos intelectos crescem com nossas ferramentas, eventualmente se tornando inseparáveis. (...) Acredito que computadores representam uma forma híbrida de ferramenta que combinam as capacidades de ferramentas convencionais para agir no mundo com os atributos especiais de ferramentas como a linguagem e a escrita para permitir substratos para o pensamento. (LEVIN, 2004, p.140)²³

Quando fala em coextensão, Levin quer dizer que mente e computador ocupam um mesmo espaço, um contínuo que flui de um para outro. Em outras palavras, para Levin

²³I like the question: “In your mind, is the computer just a tool?”, because my answer is embedded in the question; I believe that mind and computer are coextensive. (...) If writing is a medium of thought, then software is an agent of will. When it executes my will, software and I form a single, coextensive unit of thought and purpose. (...) Our intellects grow with our tools, eventually becoming inseparable. (...) I believe computers represent a hybrid form of tool in that they combine the abilities of conventional tools to act on the world with the special attributes of tools like language and writing to enable substrata for thought (LEVIN, 2004, p.140).

o software não é apenas um registro estático e externalizado de uma ideia. Enquanto a palavra escrita funciona como uma forma de registrar e expressar pensamentos, o software funciona como uma forma de executar vontades. Ou seja, software contém pensamentos e formas de atuar no ambiente de acordo com esses modelos. E assim, é como se o software se tornasse uma extensão ativa da mente. A linguagem (falada e escrita) já faz parte do homem; regiões inteiras do cérebro humano são dedicadas à sua compreensão e aplicação. Da mesma forma, Levin acredita que a computação pode se tornar parte inseparável do intelecto humano.

Na figura 3.9 é possível observar uma imagem de *Meshy*, trabalho interativo feito com Processing por Levin.

Figura 3.9: Golan Levin. *Meshy*, 2006. Fonte: site do artista. Disponível em: <http://www.flong.com/projects/meshy/>.

Em *Meshy*, o usuário desenha 2 linhas na tela, e o programa gera uma animação de elementos (losangos), que se deslocam ciclicamente entre essas 2 linhas. A forma

resultante lembra uma estrutura tridimensional, embora seja composta apenas de elementos bidimensionais. Assim como *Substrate*, este projeto se caracteriza por apresentar um sistema de regras simples que gera formas complexas. Além disso, o trabalho responde ativamente às interações de seu usuário. Ou seja, o programa não contém apenas o registro de uma forma gráfica imaginada por Levin, ele contém também instruções de como desenhar esse tipo de forma, levando em considerações as interações que recebe. *Meshy* é uma coextensão da mente de Levin por conter uma pequena ideia da mente do artista, e por ser capaz de atuar ativamente para realizar essa ideia.

Enquanto Levin fala da computação como algo que permite executar “vontades”, Tarbell fala da computação como veículo para a imaginação:

Com software, qualquer coisa que pode ser imaginada pode ser construída. O software tem uma borda misteriosa, indefinida. A programação é verdadeiramente um processo de criar algo do nada. (TARBELL, 2007, p.158)²⁴

Ou seja, criar um software é quase uma experiência metafísica, pois envolve a transformação de um “nada” em um “algo”. No entanto, é importante ressaltar que esse é o ponto de vista de um artista computacional. Essa “borda misteriosa e indefinida” existe para aquele que cria e domina o software, mas não necessariamente para aquele que apenas utiliza esse software. Outro depoimento que traz uma visão semelhante é o do artista e programador Scott Snibbe:

Uma obra de arte computacional existe somente como uma continuação dos pensamentos do programador através do meio computacional e em direção à mente do espectador. Não há como remover ou separar esses componentes. (SNIBBE, 2004, p.228)²⁵

Enquanto Levin fala da coextensão entre mente e computação, Snibbe entende que a mente do espectador também faz parte dessa continuidade. Para Snibbe, são as relações de interdependências entre os diversos elementos envolvidos em um sistema

²⁴With software, anything that can be imagined can be built. Software has a mysterious, undefined border. Programming is truly a process of creating something from nothing (TARBELL, 2007, p.158).

²⁵Computational artwork exists only as a continuation of the programmer’s thoughts through the computational medium and into the mind of the viewer. There is no way to remove or separate these components (SNIBBE, 2004, p.228).

computacional (programador, espectador, código, computador) que configuram a obra de arte computacional. Portanto, essa obra deve ser considerada em todos esses aspectos, e não apenas em uma de suas partes. A obra não é apenas o código, ou apenas a imagem gerada por esse código, é o conjunto dos elementos envolvidos em sua concepção.

Um dos trabalhos recentes de Snibbe é *Rework* (figura 3.10), aplicativo que é um complemento visual e interativo para o disco homônimo do compositor minimalista americano Philip Glass.

Figura 3.10: Scott Snibbe. *Rework*, 2012. Fonte: imagem do aplicativo em execução.

O aplicativo consiste em uma coleção de 11 peças interativas, correspondentes às 11 faixas do disco. As peças interativas possuem em comum uma linguagem gráfica minimalista baseada em formas geométricas, e a aplicação de simulações de forças físicas em suas interações. Assim como *Meshy*, este é um trabalho que utiliza da computação para criar uma experiência interativa. No caso de *Rework*, ainda se busca utilizar

a capacidade da computação de produzir inúmeras variações a partir de um mesmo conjunto de regras para produzir uma experiência que não se repete a cada vez que é reproduzida. A experiência audiovisual do usuário no aplicativo é resultado de suas interações. Dessa forma, o espectador é um componente central do trabalho.

Pode-se dizer que a noção de unidade entre mente e computação apresentada por Levin e Snibbe representa o domínio que o artista / designer / programador tem sobre a mídia computacional. Como diz Vassão, uma ferramenta “se entranha em nós, e nós nela” (VASSÃO, 2008, p.284). Nesse estado de fusão (ou simbiose, em referência à Licklider), não há como separar homem e tecnologia, mente e computação. É uma experiência quase transcendental, na qual o domínio da ferramenta é tão alto que os limites do homem e os limites da ferramenta se confundem. Não há mais manipulação de símbolos pré-programados, e sim a invenção de novos símbolos, novos programas. A computação deixa de ser caixa preta intransponível e indecifrável, e se abre para novas possibilidades inimaginadas. Ao utilizar o computador como extensão da mente, num ciclo de feedback entre algoritmo e imaginação, o artista / designer / programador concebe uma forma de romper a caixa preta e tornar a tecnologia mais humana novamente.

A seguir, são apresentadas e discutidas as entrevistas realizadas com os 3 designers brasileiros contemporâneos.

3.2 Entrevistas

As entrevistas foram realizadas por email entre janeiro de fevereiro de 2014. Cada entrevistado recebeu 5 perguntas, sendo que 4 dessas perguntas eram comuns a todos os entrevistados. Uma pergunta foi formulada especificamente para cada entrevistado, tratando de algum trabalho ou projeto realizado por ele.

Foram selecionados 3 designers brasileiros para as entrevistas. Cada um possui um *background* acadêmico e profissional diferente, mas todos exploram a computação no design de forma programática. Hugo Cristo Sant’anna é designer, professor de design, e doutorando em psicologia (na Ufes), que trabalha e pesquisa nas intersecções entre computação, design, e educação. Gustavo Ferreira é designer tipográfico, que

vêm aplicando uma abordagem sistemática e paramétrica para o desenho de famílias tipográficas. Dimitre Lima é designer gráfico, com um repertório diverso de projetos, que abrange famílias tipográficas, vinhetas para televisão, e projeções interativas para shows, concertos, e eventos.

3.2.1 Entrevista com Hugo Cristo Sant’anna

1. Hugo, em primeiro lugar, agradeço sua participação nesta pesquisa. Para iniciar esta entrevista, gostaria que você se apresentasse e contasse um pouco de sua trajetória acadêmica e profissional: onde estudou, onde trabalhou, e como essas experiências influenciam seu trabalho hoje.

Sou designer formado pela Ufes (2002), mestre (2007) e doutorando em psicologia pelo programa de pós-graduação em psicologia da Ufes (previsão de defesa março deste ano). Programo desde os 9 ou 10 anos, começando no MSX-Basic e depois partindo pro C no IBM-PC. Basicamente me interessei por programação por causa da *demoscene*²⁶, então meus interesses sempre orbitaram em torno disso que hoje se chama de design computacional e da computação gráfica mais *oldskool*²⁷.

2. Quando você começou a programar, e o que o motivou a aplicar a computação em trabalhos de design?

Por causa desses interesses eu comecei a trabalhar cedo como webdesigner no finado ZAZ de Vitória, tanto com Flash como programação server-side (CFML depois PHP) e aplicações Java em geral. Já fiz de tudo com programação e design interliga-

²⁶Comentário do autor: *demoscene* é “uma subcultura de programadores centrada no design e na manipulação de software audiovisual em tempo real” (MONTFORT et al, 2012, p.240). A *demoscene* tem origem nos primórdios da pirataria de software. Em geral, programas comerciais possuem mecanismos de verificação que certificam que apenas usuários que compraram uma licença do software possam utilizá-lo. Programadores passaram a quebrar esses mecanismos de verificação, permitindo que qualquer usuário pudesse utilizar os mesmos programas, sem pagar por uma licença de uso. As versões desbloqueadas eram distribuídas gratuitamente pela internet. Essas versões modificadas acompanhavam assinaturas gráficas ou textuais incluídas pelos programadores para sinalizar suas proezas técnicas. Essas assinaturas se tornaram cada vez mais sofisticadas, com animações e músicas executadas em tempo real. Com o tempo, a *demoscene* passou a se concentrar na produção dessas animações, os chamados *demos*. O desafio de explorar as limitações técnicas do computador é o aspecto central da *demoscene*.

²⁷Comentário do autor: esta grafia alternativa do termo *oldschool* é característica da subcultura da *demoscene*.

dos, mas de maneira geral foi o design computacional que me levou para a programação de sistemas mais hardcore. Atualmente na Psicologia tenho trabalhado muito com desenvolvimento de software para coleta, processamento, análise e visualização de dados de pesquisa, que na minha cabeça continuam dentro do mesmo universo de sempre (processar dados, gerar imagens e afins).

3. Qual o papel da computação em seu processo de trabalho (enquanto designer)? Você acredita que os conceitos da computação contribuem para o processo de design como um todo, e não apenas em etapas específicas do processo?

Eu só sou o designer que sou porque sei programar e só sou docente/pesquisador na área que escolhi porque tenho os fundamentos da Computação. Toda a minha estratégia de projeto é de natureza computacional: sempre tento criar ferramentas para me auxiliar nos processos de análise e síntese. Discuto problemas de design em termos de algoritmos com meus alunos.

4. Seu portfolio de projetos ²⁸ compreende trabalhos em diversas tecnologias e mídias, incluindo músicas, software, hardware, textos, e games. Gostaria de você destacasse um projeto para ilustrar a importância da computação em seu processo pessoal de design.

Eu sempre falo dos *demos*, mas há outras coisas que combinam todos os meus interesses e que são mais recentes... por exemplo, as projeções (Derridaw) do KFK²⁹ feitas em PureData³⁰. Seria impossível desenvolver um projeto daqueles sem utilizar princípios da Computação em sentido amplo: a própria rede que eu tive que construir para conectar o hardware e software, os algoritmos para gerar as imagens a partir das instruções Midi e OSC³¹, a interface para controlar tudo isso ao vivo e assim por diante. Não é só música, nem só animação, nem só código. Dar conta de diversos domínios da Computação é que viabilizaram a concepção de um projeto desses e acho que isso se aplica a todos os projetos que desenvolvo, em maior ou menor grau.

5. Você acredita que os conceitos da computação são importantes na formação de

²⁸<http://www.hugocristo.com.br/v32/>

²⁹Comentário do autor: KFK é sigla para Kung Fu Ko, nome de um dos projetos musicais de Sant'anna.

³⁰Comentário do autor: PureData é uma linguagem de programação que segue um paradigma de programação baseado em diagramas gráficos, e não em instruções textuais.

³¹Comentário do autor: Midi e OSC são formatos de dados que permitem a troca de informações musicais entre diferentes computadores, dispositivos, e programas.

designers?

Acredito que são importantes para qualquer pessoa. Computadores são onipresentes, problemas computacionais estão por toda parte e definitivamente saber se relacionar com esse tipo de conhecimento para além do uso é uma demanda real. As pessoas só dominam um meio quando também sabem escrever nele ao invés de apenas ler. Tem gente que sugere que o letramento computacional seria tão importante quanto saber ler e escrever no idioma nativo ou aprender matemática. A Computação, para mim, é a fronteira entre a cidadania de fato e o analfabetismo funcional de hoje.

3.2.2 Entrevista com Gustavo Ferreira

Gustavo, em primeiro lugar, agradeço sua participação nesta pesquisa. Para iniciar esta entrevista, gostaria que você se apresentasse e contasse um pouco de sua trajetória acadêmica e profissional: onde estudou, onde trabalhou, e como essas experiências influenciam seu trabalho hoje.

Estudei design gráfico e de produto na Esdi, no Rio de Janeiro, entre 1997 e 2003. Depois fiz um mestrado em comunicação e planejamento na HfG Schwäbisch Gmünd, na Alemanha, em 2004-05. E logo em seguida fiz o curso de pós-graduação em design de tipos na KABK (Type&Media)³², em 2005-06.

Trabalhei como designer gráfico na Redley, uma marca de *surfwear* do Rio, entre 1999 e 2001, e depois passei algum tempo trabalhando como designer freelancer. Durante os cursos na HfG Schwäbisch Gmünd e na KABK recebi bolsas de estudo e pude me dedicar em tempo integral aos meus próprios projetos. De volta ao Brasil em 2007, fiz alguns projetos como freelancer (família tipográfica para a UnB, proposta de identidade visual para os CVTs) e comecei a realizar oficinas de desenho de letras e design de tipos.

Acabei retornando para a Holanda em 2009, onde fundei minha empresa-de-um-homem-só Hipertipo (um neologismo que define minha área de atuação, criado a partir da fusão dos conceitos de *tipografia* e *hipertexto/hipermídia*). Durante 2009 trabalhei meio expediente para a FontShop International como administrador da comunidade

³²Comentário do autor: KABK é a sigla da Koninklijke Academie van Beeldende Kunsten, ou Academia Real de Belas Artes, situada em Haia, na Holanda.

Figura 3.11: Hugo Cristo Sant'anna. Derridaw, 2012. Fonte: site do designer. Disponível em: <http://www.hugocristo.com.br/v32/portfolio/derridaw/>.

Derridaw é um conjunto de pequenos programas (*patches*) construídos na linguagem PureData. O nome do projeto é uma corruptela que mistura o sobrenome do conhecido filósofo francês com a sigla DAW, que significa *digital audio workstation*. O *Derridaw* gera animações em tempo real que funcionam como complemento visual das performances ao vivo do projeto musical *Kung Fu Ko*, do próprio Sant'anna. As informações musicais produzidas pelo software de áudio (Ableton Live) são detectadas pelo *Derridaw*, que as utiliza como *inputs* numéricos que alteram os parâmetros das animações geradas. Sant'anna reconhece que seus conhecimentos em computação é que viabilizaram a realização desse projeto, que une música e animação.

online FontStruct. Em 2010 trabalhei meio expediente como assistente e suporte técnico para a Typotheque. A partir de 2011 passei a trabalhar 100% por conta própria, desenvolvendo ferramentas e fontes, participando de conferências, escrevendo artigos. Em 2012 cheguei a trabalhar dois meses com o Petr van Blokland para o FontBureau, como desenvolvedor de ferramentas, mas o trabalho não foi adiante.

Nos últimos anos tenho trabalhado principalmente nos meus próprios projetos. Participei da RoboThon 2012 com uma apresentação sobre meu kit de ferramentas para produção de fontes, hTools2. No ano seguinte comecei a dar aulas de programação para designers, primeiro com o curso Programação Visual (no Rio de Janeiro), e depois com um workshop mais longo chamado Python para designers (tipo)gráficos, em parceria com o também designer e programador Joancarles Casasín. Ao longo de 2013 participei de conferências na Polônia e na Espanha. Escrevi artigos sobre meu trabalho e fiz algumas entrevistas com outros designers. Em paralelo, sigo trabalhando nas minhas fontes e ferramentas de produção. Ocasionalmente também faço serviços de produção de fontes e desenvolvo ferramentas sob medida para outros designers de tipos.

Resumindo, tenho feito de tudo um pouco, buscando principalmente independência e autonomia criativa. Tem sido um período muito estimulante e produtivo, mas ao mesmo tempo é difícil sustentar essa condição por muito tempo. Agora estou em busca de algo que me dê um pouco mais de estabilidade. Estou me preparando para iniciar um doutorado na Inglaterra nesse ano ou no próximo. Pretendo pesquisar diferentes sistemas de hipertexto e desenvolver uma linguagem de padrões tipográficos para conteúdo não-linear e multidimensional.

Quando você começou a programar, e o que o motivou a aplicar a computação em trabalhos de design?

É difícil determinar uma data, pois foi um processo gradual e muito lento. Não tenho formação em ciência da computação. Mas sempre me interessei por história da ciência e tecnologia, acho que de alguma forma intuitiva sempre tive consciência das oportunidades de inovação abertas por novas mídias e tecnologias.

No final dos anos 1990 comecei a me interessar por design para a web. Brincando com o Dreamweaver pude aprender como HTML e CSS funcionam. Mais tarde par-

ticipei de um curso na Esdi sobre mídias e interfaces digitais com o Gui Bonsiepe, o que abriu os meus horizontes e aumentou ainda mais meu interesse pela área. Alguns anos depois tive aulas de Lingo (linguagem de programação usada no Director da Macromedia) na HfG Schwäbisch Gmünd, mas não consegui aprender muita coisa.

No ano seguinte participei de um curso de tecnologias para web na mesma escola, e com a ajuda do instrutor consegui desenvolver uma paleta simples de navegação para o meu sistema de fontes Elementar, usando HTML, CSS e PHP. Essa ferramenta foi muito útil para o meu trabalho e me motivou a aprender mais. Na mesma época um outro professor da escola desenvolveu uma ferramenta de desenho simples para o FontLab, em Python, para auxiliar no desenho do mesmo sistema de fontes. Minha produtividade aumentou muito graças a essa nova ferramenta.

Finalmente, consegui aprender uma boa base de programação em Python durante os meus estudos no Type&Media, na Holanda. Tive o privilégio de ter aulas de introdução com o Just van Rossum e com o Erik van Blokland. Durante o inverno aproveitei para estudar por conta própria com a ajuda de um livro. Também tive a sorte de ser colega de turma do Frederik Berlaen, com quem colaborei em um projeto ao longo do curso: a automação do método de geração de variações tipográficas do sistema Elementar.³³

Qual o papel da computação em seu processo de trabalho? Você acredita que os conceitos da computação contribuem para o processo de design como um todo, e não apenas em etapas específicas do processo?

Conhecimentos de programação me permitem trabalhar com mídias digitais de forma autônoma, sem depender de um programador para implementar minhas ideias. Essa independência é muito importante para mim, pois me dá maior liberdade para experimentar descompromissadamente.

Além disso, apesar de parecer um contrassenso, um programa (na forma de código) pode ser a linguagem mais natural para expressar um sistema gráfico. Acho muito interessante que a gente use em português a expressão *programação visual* para se referir à atividade do designer gráfico. A noção de que um manual de identidade visual é um programa – um conjunto de regras, instruções e normas – é anterior ao

³³Comentário do entrevistado: Hoje o Frederik é o principal desenvolvedor do RoboFont (editor de fontes) e do DrawBot (ambiente para programação gráfica usando a linguagem Python).

computador. Há toda uma linguagem gráfica moderna fundamentada em conceitos como sistemas, séries, transformações, progressões, proporções etc. Hoje podemos produzir esses tipos de formas de dentro para fora, a partir de sua lógica generativa expressa em código.

Não acho que saber programar é imprescindível para todo designer, mas acredito sim que conhecimentos de programação podem ajudar o trabalho do designer de várias formas. Uma delas é aumentando o espaço de possibilidades do projeto. Quando usamos as mesmas ferramentas existentes (por exemplo, a suíte de programas da Adobe), o espaço de possibilidades do projeto fica limitado pelo espaço de possibilidades da ferramenta. Quando criamos (programamos) nossas próprias ferramentas sob medida para um projeto, podemos moldar o espaço de possibilidades da ferramenta de acordo com as nossas ideias – e não ao contrário. O limite passa a ser a nossa capacidade de expressar nossas ideias em código.

A abordagem programática também permite uma iteração mais rápida, já que o percurso entre a expressão de uma ideia e sua execução é reduzido ao máximo: os resultados estão disponíveis imediatamente para avaliação, ajustes, nova execução, avaliação, ajustes etc. Dessa forma o designer pode se concentrar no projeto em si, com a produção automatizada por código.

Você desenvolveu e ministra um workshop sobre a linguagem de programação Python, voltado para designers tipográficos ³⁴. O que o motivou a desenvolver esse workshop?

Foram várias coisas juntas. A principal talvez seja a vontade de ensinar, é algo que me dá uma satisfação pessoal muito grande. Mais do que isso, sinto que essa é a minha missão, pois quando saí do Brasil para estudar na Alemanha (com bolsa dos governos brasileiro e alemão), meu objetivo era aprender o máximo possível para depois retornar e passar o conhecimento adiante. Infelizmente, quando retornei ao Brasil em 2007 não consegui obter uma posição de professor, e acabei retornando para a Holanda em 2009. Os workshops têm me possibilitado trabalhar com ensino de forma independente, suprimindo as carências (em design de tipos e programação) de cursos de design existentes, tanto no Brasil quanto na Europa.

³⁴<http://pyftgd.org>

Além disso o meu próprio interesse pelo assunto me motiva, é claro, pois para ensinar preciso estar sempre aprendendo. A parceria com o Joancarles tem funcionado muito bem nesse sentido, nós aprendemos bastante um com o outro.

Você acredita que os conceitos da computação são importantes na formação de designers?

Sim. Não apenas conceitos mas também toda a história da computação e do computador pessoal: de onde essas máquinas vieram, quem as criou, com quais objetivos etc. Todo designer conhece e sabe da importância da invenção de Gutenberg, mas poucos ouviram falar em Doug Engelbart, por exemplo – apesar de trabalharem diariamente com computadores que existem graças às suas ideias e protótipos.

O uso de computadores também tem necessariamente uma dimensão política. As mesmas questões relacionadas a autonomia e autodeterminação com as quais eu lido no meu trabalho criativo precisam ser pensadas no nível das organizações, sejam elas públicas ou privadas. No ano passado foram feitas revelações muito graves a respeito da coleta de dados em larga escala na Web, por exemplo³⁵. No Brasil todos parecem estar alheios a esse tipo de questão. Isso é muito preocupante.

Do ponto de vista artístico, acredito que conhecimento sobre computadores permite um uso mais consciente e apropriado da máquina como ferramenta e mídia, e resultados mais honestos no sentido defendido por proponentes do design moderno. Ou seja, ao invés de usar o computador para simular técnicas e características de outras mídias, podemos focar nossos esforços na exploração de suas propriedades únicas, e na busca por novas linguagens e possibilidades.

3.2.3 Entrevista com Dimitre Lima

1. Dimitre, em primeiro lugar, agradeço sua participação nesta pesquisa. Para iniciar esta entrevista, gostaria que você se apresentasse e contasse um pouco de sua trajetória acadêmica e profissional: onde estudou, onde trabalhou, e como essas experiências influenciam seu trabalho hoje.

³⁵Comentário do autor: Em 2013, Edward Snowden revelou uma rede global de vigilância operada pela National Security Agency (NSA) dos EUA, com o objetivo de interceptar comunicações pela internet. Essa operação teria acesso à informações privadas dos usuários dos maiores serviços online de email e de redes sociais.

Figura 3.12: Gustavo Ferreira. *Tuplefont*, 2011. Fonte: site do designer. Disponível em: <http://hipertipo.com/content/hyperdimensional-typefaces/>.

Este desenho tipográfico foi criado diretamente em código de computador, ao invés de ter sido desenhado em algum software de desenho. Os pontos que definem os contornos das letras são armazenadas na forma de listas de números, em código na linguagem de programação Python. Essa descrição numérica das letras permitiu a Ferreira produzir diversas formas de visualização das mesmas letras. Além disso, a construção de um programa próprio para desenhlar letras possibilitou a superação de certas limitações técnicas encontradas nos atuais formatos digitais de fontes tipográficas, como por exemplo a aplicação de cores nas letras. Nas letras da coluna da direita, observa-se que a cor **está incorporada** ao desenho das letras; a cor não foi aplicada após o desenho. É como se o “pincel virtual” que desenha as letras mudasse de cor a cada pequeno deslocamento no espaço da tela. Este é um exemplo de como a construção de ferramentas próprias permite a expansão do espaço de ideias, a exploração de ideias que são limitadas pelas ferramentas comerciais disponíveis.

Estudei Desenho Industrial / Programação Visual na UFSM (Santa Maria) e até então eu dedicava muita energia à música. Pensava que ia fazer isso da vida.

Logo depois de formado (2001) mudei para São Paulo e comecei a trabalhar em uma gravadora (Trama).

Foi uma maneira de aliar os dois interesses (música e desenho), e na época foi ótimo conviver com colegas talentosos, todo mundo tinha projetos paralelos dos mais variados, um gravando um disco, outro fazendo um jogo, outro estudando 3D. Foi minha única experiência como funcionário, sem ser estágio e foi em um lugar bem diferente, num contexto pós-bolha da internet.

Depois disso comecei a trabalhar por conta e venho desde então trabalhando ora sozinho, ora colaborando com amigos, designers e artistas e tentando encontrar o equilíbrio entre projetos pessoais e comerciais, entre a liberdade de não ser funcionário e a prisão dentro da própria liberdade.

2. Quando você começou a programar, e o que o motivou a aplicar a computação em trabalhos de design?

Era 1995, havia um computador em casa somente com DOS 6.2 e alguns softwares aborrecidos. A linguagem de programação QBASIC apresentava uma alternativa e muitas possibilidades, haviam jogos escritos nele. Comecei a fazer programas simples e depois de um tempo consegui fazer um programa para estudar escalas no violão, onde uma escala se desdobrava em diversas possibilidades de digitação.

Meu irmão estudava engenharia na época, e quando viu que eu estava interessado em programação, começou a me mostrar coisas do Turbo Pascal, que acabei usando por um tempo também.

3. Qual o papel da computação em seu processo de trabalho? Você acredita que os conceitos da computação contribuem para o processo de design como um todo, e não apenas em etapas específicas do processo?

No meu processo a computação tem dois papéis importantes: um da automatização de resolução de problemas e um segundo é o que vou chamar de entropia. Uma forma de chegar em resultados não-desenháveis, não-pensáveis quando se trabalha direto no resultado.

Trabalhando estritamente no processo (código) é como trabalhar na genética do

resultado, trazendo surpresa mesmo para quem está criando.

4. Você desenvolveu projeções de imagens interativas para grandes eventos nacionais e internacionais, como o evento de premiação do VMB ³⁶ e o festival de música Coachella ³⁷. O que motivou a utilização da computação nesses projetos?

Na verdade, talvez tenha acontecido o processo reverso, a minha familiaridade com computação tenha alinhado com as necessidades específicas destes projetos.

5. Você acredita que os conceitos da computação são importantes na formação de designers?

Acredito ser importante mostrar como possibilidade, somente isso. Seria o suficiente para despertar interesse em quem tem afinidade com isso.

Também entendo que nem todo mundo se sente confortável com um método de desenho tão abstrato.

3.2.4 Discussão sobre as entrevistas

A seguir, é apresentada uma pequena discussão a respeito dos depoimentos obtidos por meio das entrevistas.

Em relação às trajetórias acadêmicas e profissionais dos entrevistados, verifica-se que os 3 entrevistados atuam em cidades diferentes, estudaram em instituições diferentes, e tiveram trajetórias profissionais diversas. Portanto, não se deve atribuir a prática computacional desses designers à alguma experiência acadêmica ou profissional em comum entre eles. O que os 3 entrevistados têm em comum é que todos eles possuem formação acadêmica em design (embora as denominações dos cursos tenham sido diferentes em cada instituição), e que nenhum deles possui formação acadêmica adicional em ciências da computação. E mesmo assim, todos trabalham programaticamente com a computação, cada um à sua maneira.

Tendo isso em vista, é interessante observar que Ferreira estudou computação em disciplinas oferecidas nos cursos de pós-graduação que frequentou. Inclusive, é importante lembrar que a dupla de designers Letterror faz parte do corpo docente da KABK, onde Gustavo estudou. Existe uma tradição no design holandês de “afiar as

³⁶<http://dmtr.org/vmb2010/>

³⁷<http://dmtr.org/coachella/>

Figura 3.13: Dimitre Lima. Vinheta MTV Brasil, 2009. Fonte: site do designer. Disponível em: http://dmtr.org/mtv_rewind/.

Nas vinhetas animadas produzidas por Lima para a MTV Brasil em 2009, o logotipo da emissora aparece formado por sistemas de partículas em movimento. Vinhetas variadas foram produzidas com a utilização de diferentes algoritmos regendo os comportamentos das partículas. Observa-se que essa série de vinhetas possui uma linguagem gráfica bastante particular, que foge ao estilo resultante da utilização dos programas comuns de animação. Ao criar seus próprios programas para gerar as animações, Lima conseguiu produzir uma linguagem gráfica peculiar para as vinhetas. Desenvolvido com Processing, este trabalho obteve reconhecimento internacional, e aparece na galeria de projetos selecionados pela equipe do Processing (http://processing.org/exhibition/curated_page_10.html).

ferramentas”, como se observa no texto de Middendorp a respeito da obra do Letter-ror. Nesse sentido, estudar a computação é uma premissa importante do designer que trabalha com a computação. Para aqueles que ainda acreditam que o estudo da computação prejudica a formação do designer, a produção holandesa de design mostra o contrário. Não se propõe que todos os designers trabalhem programaticamente com a computação. O que se propõe é que essa abordagem seja mais explorada nos cursos de design.

Por outro lado, Lima e Sant’anna (assim como o autor deste texto) demonstraram interesse pela computação antes mesmo de ingressarem na graduação em design. Este interesse precoce pela computação pode ser relacionado com a crescente popularização da computação pessoal a partir da década de 80. Nos EUA, a porcentagem de domicílios que tinham um computador pessoal quase dobrou entre 1984 e 1989, passando de 8,2% para 15% em um intervalo de apenas 5 anos³⁸ (MONTFORT et al, 2013, p.212). O computador MSX, mencionado por Sant’anna, foi uma das plataformas dessa popularização dos computadores pessoais (LUZ, 2009, p.51). A disponibilidade de um computador em casa permite que as pessoas tenham um contato precoce com a computação. Os depoimentos de Lima e Sant’anna indicam que os conhecimentos da computação adquiridos antes da graduação permeiam suas atividades profissionais após a graduação, embora a profissão que tenham escolhido não esteja diretamente relacionada com a computação.

Quanto ao papel da computação nos processos de trabalhos dos entrevistados, constata-se que a computação é valorizada como uma ferramenta de automação e ferramenta de exploração de ideias. Lima exalta a possibilidade da computação gerar resultados inesperados, auxiliando na obtenção de resultados que são a princípio “não-desenháveis” e “não-pensáveis”. Ou seja, a prática programática ajuda o designer a pensar soluções que não seriam “pensáveis” de outras maneiras. Dos depoimentos pesquisados, Tarbell também compartilha desta opinião. Nesta constatação é possível identificar um eco da ideia de simbiose de Licklider. Ferreira aborda um ponto

³⁸Não foi encontrada pesquisa correspondente ao mesmo período no Brasil. No entanto, artigo publicado pelo jornal O Estado de São Paulo em 2008 indica que um crescimento de magnitude semelhante ocorreu no país somente na metade da década de 90. Segundo o artigo, em 1996, 9,6% dos domicílios brasileiros dispunham de um computador pessoal. Esse número saltou para 24% em 2007 (O ESTADO DE SÃO PAULO, 2008).

que remete à caixa preta de Flusser: que os programas comerciais de desenho possuem limitações e acabam limitando as soluções possíveis. Para ele, que possui forte influência da visão holandesa, trabalhar com a computação permite criar ferramentas próprias, mais adequadas para as ideias que quer explorar em seus projetos. Por fim, Sant’anna adota a computação como essencial em seu processo de trabalho, tanto como ferramenta de análise (especialmente na visualização e interpretação de dados) quanto como ferramenta de síntese (ou seja, na produção de soluções visuais). É interessante observar que, enquanto Ferreira e Lima falam da computação com foco em linguagens de programação, Sant’anna aborda outros aspectos como as redes, o hardware, e as interfaces entre diversos elementos de um sistema computacional. Analisando os trabalhos produzidos pelos entrevistados, também é possível observar algumas diferenças nas formas como eles trabalham com a computação. Muitos dos trabalhos de Lima são resultado de um programa criado especificamente para o projeto. Ou seja, cada projeto envolve um programa diferente e único. Já Ferreira tem uma preocupação em criar ferramentas utilitárias genéricas que ele utiliza em diversos projetos diferentes. Seu kit de ferramentas *hTools2* é disponibilizado gratuitamente nos moldes do software *open-source*.

Em relação à importância da computação na formação de designers, Ferreira aponta para a importância de conhecer a história da computação, não só no sentido tecnológico, mas no sentido das ideias que levaram ao desenvolvimento das tecnologias, e menciona os trabalhos de Engelbart. A importância da sua obra foi tema de um artigo publicado pelo pesquisador Victor Bret um dia após o falecimento de Engelbart em 2 de julho de 2013. No artigo, Bret defende que o grande legado de Engelbart não é a invenção do email, do mouse, ou da videoconferência, mas uma visão da computação enquanto amplificadora do intelecto. Esta visão histórica e humana da computação foi um dos objetivos do autor no primeiro capítulo desta dissertação. No entanto, como lembra Lima, o estudo da computação não necessariamente tem apelo para estudantes e profissionais de design. E esse é um problema que já foi enfrentado por Kay no projeto do Dynabook. O que Kay constatou é que a computação é que precisa se adaptar ao seu usuário. Uma linguagem de programação pode ser utilizada por crianças se for projetada com esse público-alvo em mente. Assim como os designers devem

se aproximar mais da computação, a computação também deve se aproximar mais de seus usuários. Em seu artigo *The south face of the mountain* (1999), Maeda imagina como ideal um profissional “híbrido”, com formação tecnológica e formação humanística em medidas iguais.

Os depoimentos de Ferreira e Sant’anna apontam para a importância de uma discussão politizada da computação no contexto brasileiro. Nesse sentido, também é importante mencionar um comentário de Machado a esse respeito:

Em um país como o Brasil, deslocado geograficamente em relação aos países produtores de tecnologia e em que o acesso aos bens tecnológicos é ainda seletivo e discriminatório, uma discussão séria sobre o tema das novas tecnologias deve necessariamente refletir esse deslocamento e essa diferença (...). (MACHADO, 2007, p.31)

A grande parte dos sistemas computacionais são projetados e projetados em outros países e importados para o Brasil. Se os programas que são utilizados aqui são produzidos em outros países, eles não levam em consideração os valores e as necessidades dos usuários daqui. Machado aponta para a importância de reconhecer essa posição do país no cenário internacional em uma discussão das novas tecnologias. Como apontaram Blokland e Middendorp, programas comerciais buscam atender às demandas de um grande grupo de usuários, e assim acabam deixando de atender às demandas de grupos específicos. Nesse sentido, as grandes corporações que desenvolvem software acabam visualizando seus usuários como um grupo quase homogêneo. Machado propõe que é necessário considerar as especificidades locais. Assim, estudar a computação de forma crítica é um primeiro passo para compreendê-la melhor, para poder propor coisas novas dentro desse universo, podendo identificar as limitações existentes, e enxergar outras possibilidades. Construir as próprias ferramentas é um caminho alternativo à homogeneidade proposta pelos programas comerciais; é um caminho para a construção de tecnologias sensíveis ao contexto em que estão inseridas, construção da qual o design deve participar. É possível observar iniciativas nesse sentido no ensino de design no Brasil. No curso de Graduação em Design da Fauusp, existem disciplinas de introdução à computação, nas quais os alunos têm

primeiros contatos com plataformas de programação como o Processing. Os entrevistados Sant'anna e Ferreira também possuem experiências nesse sentido. Sant'anna utiliza a computação nas aulas que ministra no curso de graduação em Desenho Industrial da Ufes. Ferreira oferece um workshop de programação Python para designers. É importante que o interesse pela computação parta daqueles que estão envolvidos com o seu ensino.

Por fim, como diz Sant'anna, o conhecimento da programação é um elemento essencial para o exercício da cidadania hoje. E se Kay já relacionava a capacidade de programar com o letramento digital, outros autores contemporâneos como Ronaldo Lemos (2013), Silvio Meira (2012), e Douglas Rushkoff (2010) têm abordado esse tema recentemente. Lemos, que foi pesquisador do MIT Media Lab, é doutor em direito pela USP, e atua em iniciativas de inclusão digital no Brasil. Meira é doutor em ciência da computação pela University of Kent (Inglaterra), docente na Universidade Federal do Pernambuco (Ufpe) e cientista-chefe do Centro de Estudos e Sistemas Avançados de Recife (Cesar). Rushkoff é doutor em novas mídias e cultura digital pela Utrecht University (Holanda), escritor, e professor. Para esses autores, a falta de conhecimento em programação corresponde ao analfabetismo digital, que leva a uma menor capacidade de compreender as questões da sociedade e de agir nessas questões. Segundo Lemos, “a tecnologia abre possibilidades extraordinárias de participação na vida pública” (LE-MOS, 2013), mas o conhecimento de programação é essencial para a manutenção da democracia. Sem esse conhecimento, mesmo que em nível básico, as pessoas não conseguem formar opiniões sobre os sistemas computacionais que utilizam, e ficam reféns das decisões tomadas por esses sistemas. Para Meira, não entender como os sistemas computacionais funcionam e não conseguir reprogramar esses sistemas “será um problema tão fatal (...) como não entender porcentagens” (MEIRA, 2012). Meira observa que uma população que não sabe resolver problemas com porcentagens não consegue compreender e participar de discussões sobre os problemas econômicos de seu país. Do mesmo modo, uma população que não sabe programar computadores (e suas redes) não consegue utilizar essa tecnologia a seu favor e fica refém dos interesses daqueles que programam essas máquinas. Para Rushkoff, “uma vez que as pessoas enxergam a forma como as tecnologias são programadas, elas passam a re-

conhecer os programas existentes em todos os outros lugares”³⁹ (RUSHKOFF, 2010). Ou seja, esse conhecimento permite que as pessoas reconheçam as intenções embutidas nas tecnologias que utilizam. Como exemplo, ele menciona o Facebook, que é uma rede social na qual as pessoas mantêm amizades e relacionamentos interpessoais, mas cujo objetivo é a obtenção de lucro a partir das informações pessoais que são compartilhadas e publicadas dentro de sua plataforma. Essa intenção não é explícita, e conseqüentemente pode não ser percebida por alguém que ignora como funcionam os sistemas computacionais. Conhecer computação e programação permite ao indivíduo compreender melhor como os sistemas computacionais com que interage são construídos, e dessa forma permite que ele compreenda melhor as intenções por trás desses sistemas.

Ao lembrar das revelações acerca da NSA em 2013, Ferreira indica que a computação faz parte das vidas das pessoas e tem conseqüências importantes até mesmo com relação às suas liberdades individuais. A revelação desse esquema global de espionagem mostra que qualquer comunicação realizada pela internet pode ser alvo de interceptação. Mas não é apenas o governo americano que está interferindo na privacidade dos usuários da internet. Em troca de serviços gratuitos de comunicação pela internet (como email, mensagens, videoconferências, e redes sociais), as pessoas vêm (voluntariamente) abrindo mão de suas próprias privacidades, fornecendo informações pessoais para as grandes corporações. O grande volume de informações fornecido pelos próprios usuários é utilizado por essas empresas para prever comportamentos futuros e criar campanhas de marketing mais precisas. No entanto, essa troca é apenas implícita, e não é clara para todos os usuários desses serviços. Sendo assim, tendo em vista que as comunicações são cada vez mais mediadas por sistemas computacionais, é necessário refletir sobre as relações entre computação e liberdade individual. O momento é oportuno para o desenvolvimento de novas propostas, alternativas às soluções existentes. Para tal, o conhecimento da computação é essencial.

³⁹For once people come to see the way their technologies are programmed, they start to recognize the programs at play everywhere else (RUSHKOFF, 2010).

Considerações finais

Esta dissertação buscou investigar vários aspectos e precedentes para a prática contemporânea do design computacional. No primeiro capítulo, foram estudados os efeitos da computação na sociedade, e algumas ideias relativas ao potencial da computação como amplificadora do intelecto humano. No segundo capítulo, estudou-se a prática artística generativa e o precedente histórico da obra de Gerstner no design gráfico. No terceiro capítulo, foram apresentados depoimentos de designers que trabalham computacionalmente, procurando identificar os temas comuns e as motivações por trás da prática do design computacional.

Se constata que, já se discute há muito tempo sobre os efeitos da computação na sociedade. A crescente presença da computação em nosso ambiente traz inúmeras mudanças na forma como as pessoas se relacionam entre si, com o trabalho, e com o lazer. Essas mudanças já seriam suficiente para justificar uma maior atenção do design à computação. Mas, além disso, observou-se que o desenvolvimento da computação foi movido por cientistas que enxergavam na computação uma forma de expandir a inteligência do homem, permitindo ao homem explorar novas soluções para seus problemas, e novas formas de agir no mundo. Para esses cientistas, o objetivo era tornar a computação uma ferramenta de criatividade, uma “amplificadora de fantasias”. No entanto, esse objetivo ainda não foi alcançado para a maioria dos usuários de computador. Ainda se vê a computação mais como ferramenta de produtividade do que de criatividade. Tendo em vista que buscar novas ideias e soluções faz parte da atividade de designers, arquitetos, e artistas, conclui-se que a computação pode contribuir criativamente nos processos de design. Essa visão foi corroborada pelos depoimentos coletados, especialmente nas entrevistas com os designers brasileiros. Os entrevista-

dos ainda ressaltaram a importância do conhecimento em computação como entendimento dos fatos do mundo, como uma forma de letramento que permite o exercício da cidadania.

Com relação a trabalhos futuros, aqui são apresentados dois possíveis caminhos. Uma primeira possibilidade está associada às críticas das plataformas digitais de comunicação (redes sociais, especialmente o Facebook por seu maior alcance) e de distribuição de conteúdo (como a App Store da Apple) que existem atualmente. Esta discussão têm ganho destaque recentemente, e alguns dos autores mais engajados nesse movimento crítico contemporâneo são Jaron Lanier (2010), Douglas Rushkoff (2013), e Evgeny Morozov (2013). O que esses autores dizem é que o desenvolvimento da internet favoreceu o estabelecimento de monopólios digitais, como por exemplo o Google (que domina grande parcela das buscas e da publicidade online), o Facebook (que domina as redes sociais na maior parte do mundo ocidental), e a Apple (que exerce papel central no mercado de dispositivos móveis e de aplicativos para esses dispositivos). O problema com o surgimento desses monopólios é que as plataformas digitais têm se tornado cada vez mais fechadas em si mesmas. Ao invés de promover o diálogo entre diferentes plataformas, a cultura digital favoreceu o fortalecimento de algumas plataformas específicas em detrimento de outras. Isso coloca uma questão importante para os designers, por dois motivos. Por um lado, o próprio trabalho do designer acaba sendo limitado a determinadas plataformas digitais. Por outro lado, aqueles que vão utilizar os produtos e as interfaces projetadas por esses designers também estão limitados dentro dessas plataformas fechadas. Se Kay imaginava que um Dynabook facilitaria a comunicação e a troca de informações entre as pessoas, o que se vê hoje é que os iPads e iPhones são plataformas fechadas que não favorecem essas trocas. Nesse contexto, um trabalho futuro pode discutir: o que pode propor um designer? O que pode ser proposto para reverter essa tendência de fechamento das plataformas digitais?

Outro caminho possível para trabalhos futuros é seguir o caminho trilhado por Kay, Papert, Maeda, Fry, Reas, e Bret: investigar e projetar novos tipos de interfaces e de linguagens de programação, que sejam mais fáceis de aprender e de usar. Como observou Lima em entrevista, a programação de computadores é uma tarefa abstrata, com a qual muitos designers não possui afinidade. Existem linguagens de programa-

ção de computadores que não são baseados em texto, como por exemplo o Scratch, uma linguagem de programação voltada para crianças (assim como o Smalltalk e o Logo). No Scratch, um programa é construído por meio de uma interface gráfica, e é constituído por vários blocos que são encaixados uns nos outros. Em outras linguagens como o PureData, Max, e vvvv, voltadas para performances musicais e visuais em tempo real, os programas são constituídos por caixas que são conectadas por fios. Esses são alguns dos modelos existentes, nem sempre pensados por designers. Nesse sentido, é interessante que designers também participem do projeto de novos modelos de linguagens de programação. Como constataram Kay e Papert, a linguagem de programação precisa se aproximar das pessoas que irão utilizá-la. Do contrário, as pessoas passam a ser apenas usuários passivos de sistemas programados por grandes corporações e governos. Tornam-se usuários ingênuos, aficionados por tecnologias, que não conseguem avaliar os verdadeiros potenciais e impactos delas. Neste mundo cada vez mais permeado pela computação, o projeto de novas interfaces e linguagens que permitam maior domínio da máquina pelo homem é um importante problema de design.

Referências bibliográficas

- BENSE, Max. The projects of generative aesthetics. In: Reichardt, Jasia. **Cybernetics, art and ideas**. Londres: Studio Vista, 1971.
- BRET, Victor. **A few words on Douglas Engelbart**. 3 jul. 2013. Disponível em: <http://worrydream.com/Engelbart/>. Acesso em: 1 jan. 2014.
- BURTON, Ed. Sodaconstructor. In: FRY, Benjamin; REAS, Casey. **Processing: a programming handbook for visual designers and artists**. Cambridge: MIT Press, 2007. p.262-265.
- BUSH, Vannevar. “As We May Think” (1945). In: PACKER, Randall; JORDAN, Ken (ed.). **Multimedia: from Wagner to virtual reality**. Expanded edition. Nova Iorque: W.W. Norton & Company, 2001. p.141-159.
- COMPUTER SCIENCE TEACHERS ASSOCIATION; INTERNATIONAL SOCIETY FOR TECHNOLOGY IN EDUCATION. **Computational thinking: teacher resources**. 2a edição. 2011. Disponível em: http://csta.acm.org/Curriculum/sub/CurrFiles/472.11CTTeacherResources_2ed-SP-vF.pdf. Acesso em: 20 jan. 2014.
- CORDEIRO, Waldemar. **Arteônica**. São Paulo: Editora Universidade de São Paulo, 1971. Reimpresso em: MASCARENHAS, Nelson; VELHO, Luiz; HESS, Lilia (org.). Waldemar Cordeiro: Arteônica. Disponível em: <http://www.visgraf.impa.br/Gallery/waldemar/catalogo/catalogo.pdf>. Acesso em: 11 abr. 2013.

DELEUZE, Gilles; GUATTARI, Felix. **Mil platôs. Volume 1.** São Paulo: Editora 34, 1980.

DENNING, Peter J. Beyond computational thinking. **Communications of the ACM**, v.52, n.6, p.28-30, 2009.

_____. The great principles of computing. **American Scientific**, v.98, n.5, p.369-372, 2010.

DIAZ-JEREZ, Gustavo. The Classical period. In: **Algorithmic music: using mathematical models in music composition**. Tese de doutorado. The Manhattan School of Music, Nova Iorque, 2000. p.21-23.

ENGELBART, Douglas. "Augmenting Human Intellect: A Conceptual Framework" (1962). In: PACKER, Randall; JORDAN, Ken (ed.). **Multimedia: from Wagner to virtual reality**. Expanded edition. Nova Iorque: W.W. Norton & Company, 2001. p.64-90.

FERREIRA, Marcelo de Paula. **Um roteiro para a criação de programa computacional dedicado à geração de pré-projeto arquitetônico de habitações sociais: realimentação de modelo a partir de Avaliações Pós-Ocupação**. Dissertação de mestrado. Faculdade de Arquitetura e Urbanismo, Universidade de São Paulo, São Paulo, 2009.

FISCHER, Gerhard; GIACCARDI, Elisa. Meta-Design: A Framework for the Future of End-User Development. In LIEBERMAN, H., PATERNÒ, F., WULF, V. (Eds.) **End User Development - Empowering people to flexibly employ advanced information and communication technology**. Dordrecht: Kluwer Academic Publishers, 2004.

FLUSSER, Vilém. **Filosofia da caixa preta**. Rio de Janeiro: Relume Dumará, 2002.

FRY, Benjamin; REAS, Casey. **Processing: a programming handbook for visual designers and artists**. Cambridge: MIT Press, 2007.

GABBAI, Livia Escobar. **Arquitetura da imagem - processo de criação da imagem**

- em computação gráfica como leitura da imagem cinematográfica.** Dissertação de mestrado. Faculdade de Arquitetura e Urbanismo, Universidade de São Paulo, São Paulo, 2006.
- GALANTER, Philip. What is Generative Art? Complexity theory as a context for art theory. In: **GA2003–6th Generative Art Conference**. 2003. Disponível em: http://www.philipgalanter.com/downloads/ga2003_paper.pdf. Acesso em: 11 abr. 2013.
- _____. Generative art and rules-based art. **Vague Terrain**, v. 3, jun. 2006.
- GERSTNER, Karl. **Designing programmes**. Baden: Lars Müllers Publishers, 1964, 2007.
- GESSERT, George. An introduction to genetic art: conflicts of interest jeopardize scientific integrity and public health. **GeneWatch**, v. 17, n. 2, mar.-abr. 2004.
- GONÇALVES, Marly de Menezes. **O uso do computador como meio para a representação do espaço: Estudo de caso na área de ensino do Digital & Virtual Design**. Tese de doutorado. Faculdade de Arquitetura e Urbanismo, Universidade de São Paulo, São Paulo, 2009.
- GOOGLE. **Exploring computational thinking: what is CT?** 2011. Disponível em: <http://www.google.com/edu/computational-thinking/what-is-ct.html>. Acesso em: 20 jan. 2014.
- GREDDINGER, Paul. Pro-Programmatic. In: GERSTNER, Karl. **Designing programmes**. Baden: Lars Müllers Publishers, 1964, 2007. p.4-5.
- GREELISH, David. An interview with computing pioneer Alan Kay. **Time**. 2013. Disponível em <http://techland.time.com/2013/04/02/an-interview-with-computing-pioneer-alan-kay/>. Acesso em: 20 jan. 2014.
- HOLLIS, Richard. **Design gráfico: uma história concisa**. São Paulo: Martins Fontes, 2001.

_____. “To describe the problem is part of the solution.” In: **Designing programmes**. Baden: Lars Müllers Publishers, 2007. p.7.

_____. The designer as programmer. **Eye**, n.43, 2002. Disponível em: <http://www.eyemagazine.com/review/article/the-designer-as-programmer>. Acesso em: 20 jan. 2014.

JOINT TASK FORCE FOR COMPUTING CURRICULA 2005. **Computing curricula 2005: the overview report**. 2006. Disponível em: http://www.acm.org/education/curric_vols/CC2005-March06Final.pdf. Acesso em: 20 jan. 2014.

KAY, Alan. A personal computer for children of all ages. In: **Proceedings of the ACM National Conference**, Boston, 1972.

_____. User interface: a personal view. 1989. Reimpresso em: PACKER, Randall; JORDAN, Ken (Eds.). **Multimedia: From Wagner to virtual reality**. Nova Iorque: W.W. Norton, 2002. p.121-131.

KAY, Alan; GOLDBERG, Adele. Personal dynamic media. In: IEEE Computer, v. 10, n. 3, p.31-41, mar. 1977. Reimpresso em: WARDRIP-FRUIN, Noah; MONTFORT, Nick. **The new media reader**. Cambridge: The MIT Press, 2003. p.391-404.

LANIER, Jaron. **You are not a gadget**. Londres: Allen Lane, 2010.

LEMOES, Ronaldo. Não saber programar é o analfabetismo do futuro. **Folha de São Paulo**. São Paulo, 4 nov. 2013. Disponível em: <http://www1.folha.uol.com.br/colunas/ronaldolemos/2013/11/1365651-nao-saber-programar-e-o-analfabetismo-do-futuro.shtml>. Acesso em: 20 jan. 2014.

LEVIN, Golan. Is the computer a tool? In: MAEDA, John. **Creative code**. London: Thames & Hudson, 2004. p.140.

LICKLIDER, J.C.R. “Man Computer Symbiosis” (1960). In: PACKER, Randall; JORDAN, Ken (ed.). **Multimedia: from Wagner to virtual reality**. Expanded edition.

- Nova Iorque: W.W. Norton & Company, 2001. p.55-63.
- LIEBERMAN, Henry. Intelligent Graphics. **Communications of the ACM**, v. 39, n. 8, p. 38-48, 1996.
- LUZ, Alan Richard da. 1984~1989: O Renascimento com a Nintendo. In: **Linguagens gráficas em videogame**. Dissertação de mestrado. Faculdade de Arquitetura e Urbanismo, Universidade de São Paulo, São Paulo, 2009. p.51-61.
- MACHADO, Arlindo. **Arte e mídia**. Rio de Janeiro: Jorge Zahar, 2007.
- MAEDA, John. The south face of the mountain. **MIT Techonology Review**, v.101, n.4, jul./ago. 1998, p.76-80.
- _____. **Design by numbers**. Cambridge: MIT Press, 2001.
- _____. **Creative code**. London: Thames & Hudson, 2004.
- MAIA, Marcelo. **Cidade instantânea**. Tese de doutorado. Faculdade de Arquitetura e Urbanismo, Universidade de São Paulo, São Paulo, 2013.
- MANOVICH, Lev. **Software takes command**. Edição Kindle. Nova Iorque: Bloomsbury Academic. 2013.
- MCCORMACK, Jon; DORIN, Alan; INNOCENT, Troy. Generative design: a paradigm for design research. In: **Proceedings of Futureground**, Design Research Society, Melbourne, 2004. Disponível em: <http://directory.umm.ac.id/sistem-pakar/genDesignFG04.pdf>. Acesso em: 11 abr. 2013.
- MCCORMACK, Jon; DORIN, Alan. Art, Emergence, and the Computational Sublime. In: **Second Iteration: a conference on generative systems in the electronic arts, CEMA**. Melbourne: 2001. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.16.6640&rep=rep1&type=pdf>. Acesso em: 11 abr. 2013.

MCLUHAN, Marshall. **Understanding media: the extensions of man**. Edição eletrônica. Berkeley: Gingko Press, 1964, 2013.

MCLUHAN, Marshall; FIORE, Quentin; AGEL, Jerome (coord.). **The medium is the massage**. Londres: Penguin Books, 1967, 2008.

MEIRA, Silvio. Programar é preciso... **Terra Magazine**, 19 jun. 2012. Disponível em: <http://terramagazine.terra.com.br/silviomeira/blog/2012/06/19/programar-preciso/>. Acesso em: 20 jan. 2014.

MIDDENDORP, Jan; VAN BLOKLAND, Erik. Toolspace. In: KUIJPERS, Els (ed.). **LettError (Charles Nypels Award. Goodwill series, v.49)**. Nuth: Drukkerij Rosbeek bv, 2000. p.20-29. Disponível em: <http://lettererror.com/writing/toolspace/>. Acesso em: 1 jan. 2014.

MINSKY, Marvin. **The society of mind**. Nova Iorque: Simon & Schuster, 1988.

MOHR, Manfred. Depoimento do artista. In: **Manfred Mohr computer graphics - une esthétique programmée**. Paris: ARC, Musée d'Art Moderne de la Ville de Paris, 1971.

_____. Manfred Mohr. **Computer Graphics and Art**, v.1, n.2, mai. 1976.

MONTFORT, Nick et al. **10 PRINT CHR\$(205.5+RND(1)); : GOTO 10**. Cambridge: The MIT Press, 2012. Disponível em: <http://10print.org/>. Acesso em: 11 abr. 2013.

MOROZOV, Evgeny. **Open and closed**. New York Times, 16 mar. 2013. Disponível em: <http://www.nytimes.com/2013/03/17/opinion/sunday/morozov-open-and-closed.html>. Acesso em: 1 jan. 2014.

MOURA, Paulo Henrique Frediani de. **A influência da ferramenta “computador” no processo de desenvolvimento de uma marca – o caso da Caixa Econômica Federal**. Dissertação de mestrado. Faculdade de Arquitetura e Urbanismo, Universidade de São Paulo, São Paulo, 2006.

NAKE, Frieder. There should be no computer art. **Bulletin of the Computer Arts Society**. p.18-19, out. 1971.

NATIVIDADE, Verônica Gomes. **Fraturas metodológicas nas arquiteturas digitais**. Dissertação de mestrado. Faculdade de Arquitetura e Urbanismo, Universidade de São Paulo, São Paulo, 2010.

NIERHAUS, Gerhard. Language and music generators - a book of books. In: **Algorithmic composition: paradigms of automated music generation**. Mörlenbach: Springer, 2009. p.36-39.

NOLL, A. Michael. Human or machine: a subjective comparison of Piet Mondrian's "Composition with lines" (1917) and a computer-generated picture. **The Psychological Record**, v.16, n.1, p.1-10, 1966. Disponível em: <http://noll.uscannenberg.org/Art%20Papers/Mondrian.pdf>. Acesso em: 1 jan. 2014.

_____. The digital computer as a creative medium. **IEEE Spectrum**, v.4, n.10, out. 1967, p.89-95.

O ESTADO DE SÃO PAULO. Computador em casa. **O Estado de São Paulo**. São Paulo, 1 jan. 2008. Caderno principal, p.3. Disponível em: http://www.observatoriodaimprensa.com.br/news/view/uso_de_computador_pessoal_cresce_no_brasil. Acesso em: 1 jan. 2014.

OLIVEIRA, Kleber Adriano Silva e. **O ponto e o pixel: novas mídias, novas linguagens**. Dissertação de mestrado. Faculdade de Arquitetura e Urbanismo, Universidade de São Paulo, São Paulo, 2006.

PACKER, Randall; JORDAN, Ken (ed.). **Multimedia: from Wagner to virtual reality**. Expanded edition. Nova Iorque: W.W. Norton & Company, 2001.

PLAZA, Julio. **Arte e interatividade: autor-obra-recepção**. Revista de Pós-graduação, CPG, Instituto de Artes, Unicamp, 2000. Disponível em: <http://www.cap.eca.usp.br/ars2/arteeinteratividade.pdf>. Acesso em: 11 abr. 2013.

RAWLINS, Gregory. **Slaves of the machine**. Cambridge: MIT Press, 1998.

RHEINGOLD, Howard. **Tools for thought: the history and future of mind-amplifiers**. 1984. Disponível em: <http://www.rheingold.com/texts/tft/>. Acesso em 11 abr. 2013.

ROYZEN, Beatriz Gonçalves Boskovitz. **Os recursos da computação gráfica na elaboração de projetos**. Dissertação de mestrado. Faculdade de Arquitetura e Urbanismo, Universidade de São Paulo, São Paulo, 2009.

RUSHKOFF, Douglas. Why Johnny can't program: a new medium requires a new literacy. **Huffington Post**, 30 set. 2010. Disponível em: http://www.huffingtonpost.com/douglas-rushkoff/programming-literacy_b_745126.html. Acesso em: 20 jan. 2014.

_____. Why I'm quitting Facebook. **CNN**, 25 fev. 2013. Disponível em: <http://edition.cnn.com/2013/02/25/opinion/rushkoff-why-im-quitting-facebook/>. Acesso em: 1 jan. 2014.

SANTOS, Ivan Lubarino Piccoli dos. **Mídia eletrônica e agenciamentos de significados para a arquitetura e design**. Tese de doutorado. Faculdade de Arquitetura e Urbanismo, Universidade de São Paulo, São Paulo, 2010.

SIMON JR, John. Authorship, Creativity, and Code. In: MAEDA, John. **Creative code**. London: Thames & Hudson, 2004. p.46.

SIMONDON, Gilbert. **El modo de existencia de los objetos técnicos**. Buenos Aires: Prometeo Libros, 2007.

SNIBBE, Scott. The emptiness of code. In: MAEDA, John. **Creative code**. London: Thames & Hudson, 2004. p.228.

SODDU, Celestino. **Generative Art**. Milão: Politecnico di Milano, 2008. Disponível em: <http://web.archive.org/web/20080219050736/http://www.generativeart.com/>. Acesso em: 1 jan. 2014.

- STOLFI, Ariane. **World wide web: forma aparente e forma oculta**. Dissertação de mestrado. Faculdade de Arquitetura e Urbanismo, Universidade de São Paulo, São Paulo, 2010.
- TARBELL, Jared. Fractal. Invaders and Substrate. In: FRY, Benjamin; REAS, Casey. **Processing: a programming handbook for visual designers and artists**. Cambridge: MIT Press, 2007. p.156-159.
- TURKLE, Sherry. **The second self: computer and the human spirit**. Edição do 20º aniversário. Cambridge: The MIT Press, 2005.
- VAN BLOKLAND, Erik. RandomFont Beowolf. In: FRY, Benjamin; REAS, Casey. **Processing: a programming handbook for visual designers and artists**. Cambridge: MIT Press, 2007. p.168-171.
- VASSÃO, Caio Adorno. **Arquitetura livre: complexidade, metadesign e ciência nômade**. Tese de doutorado. Faculdade de Arquitetura e Urbanismo, Universidade de São Paulo, São Paulo, 2008.
- WIENER, Norbert. "Cybernetics in History," The Human Use of Human Beings (1954). In: PACKER, Randall; JORDAN, Ken (ed.). **Multimedia: from Wagner to virtual reality**. Expanded edition. Nova Iorque: W.W. Norton & Company, 2001. p.47-54.
- WILSON, Mark. **Drawing with computers**. Nova Iorque: Perigee Books, 1985.
- WING, Jeannette M. Computational thinking. **Communications of the ACM**, v.49, n.3, p.33-35, 2006. Disponível em: <http://portal.acm.org/citation.cfm?doid=1118178.1118215>. Acesso em: 18 nov. 2012.
-
- _____. Computational thinking and thinking about computing. **Philosophical transactions of The Royal Society A**, v.366, n.1881, p.3717-3725, 2008. Disponível em: <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2696102&tool=pmcentrez&rendertype=abstract>. Acesso em: 18 nov. 2012.

Cólofon

Esta dissertação foi composta com a família tipográfica Quadraat, do designer holandês Fred Smeijers, publicada pela FontFont em 1992. Foi utilizado o sistema de composição tipográfica $\text{X}_{\text{T}}\text{E}_{\text{X}}$, baseado no sistema $\text{T}_{\text{E}}\text{X}$ de Donald Knuth. O sistema $\text{T}_{\text{E}}\text{X}$ teve seu desenvolvimento iniciado em 1978 e é um sistema computacional baseado em comandos textuais voltado para a composição tipográfica de livros e publicações científicas (especialmente nas áreas relacionadas com a matemática e a computação).

A imagem da capa desta dissertação foi criada com a linguagem de programação Python e o software Drawbot, de Just van Rossum, Erik van Blokland, e Frederik Berlaen.