Negotiating With The || && !

reading codes and their symbolic structures of control

by

Pedro Cardoso Zylbersztajn
B.A. Visual Communication
Pontifical Catholic University of Rio de Janeiro, 2015

Submitted to the Department of Architecture in partial fulfillment of the
requirements for the degree of Master of Science in Art, Culture and Technology

at the
Massachusetts Institute of Technology
June 2018

Signature of Author: _____

Department of Architecture
May 11, 2018

Certified by: _____

Judith Barry
Professor of Art, Culture and Technology
Director, Program in Art, Culture and Technology
Thesis Supervisor

Accepted by: _____

Sheila Kennedy
Professor of Architecture
Chairman, Department Committee for Graduate Students

Committee

Judith Barry
Thesis Supervisor
Professor of Art, Culture and Technology
MIT Department of Architecture

Nick Montfort
Thesis Reader
Professor of Digital Media
MIT Comparative Media Studies/Writing

Negotiating With the ||&&!

reading codes and their symbolic structures of control

by
Pedro C. Zylbersztajn

Submitted to the Department of Architecture on May 11, 2018 in Partial Fullfiment of the Requirements for the degree of Master of Science in Art, Culture and Technology

Abstract

This thesis investigates software as a textual and aesthetic object through research-based artistic practice and arts-based research. Its main particular interest is in how codes (computer codes, more specifically, but positioned in relation to other linguistic codes) exercise control. It engages Pierre Bourdieu's framework of language and symbolic power, Stuart Hall's encoding/decoding model, and Wendy Chun's notion of programability, aiming to discuss how to read codes in ways that create possible semantic and pragmatic negotiations with their imperatives.

This document draws a model of reading that accounts for the sociological distribution of authority contained in software. It accepts ambiguity in face of invisibility, examines what is the ontological proximity of the code with the performative effect it generates, and how shifts and manipulations of this relational axis may work to oppose or divert prescriptive command structures. The goal of the project is to reflect upon how art practice can provide different modes of reading codes that may prove themselves pertinent to a less passive engagement with this subtle layer of control of everyday life.

Thesis Supervisor: Judith Barry
Title: Professor of Art, Culture and Technology

Acknowledgements

I would like to thank each of the many people who have in different ways contributed to the development of this project and to my two years in the institution and program in which it has taken place. My gratitude is really enormous to all. However, a lesser number of great people warrant special consideration:

My reader Nick Montfort, who has been incredibly generous in helping me think through and write down these ideas. Renée Green, who has been fundamental in my conceptualization of what a thesis could and should be within this context, and in understanding many aspects of my practice. Mario Caro, for pushing me on all aspects of this thesis, consequently making it (ever so slightly) more rigorous. My advisor Judith Barry, for taking on the project in a later stage and helping me get to this result.

My peer group, the people I will miss the most: Jessica Sarah Rinland, Laura Serejo Genes, Nicolás Consuegra and Monica Paéz, Nicolás Kisic Aguirre, and Nolan Oswald Dennis. I can safely say that the most valuable things I have learned in this place have come from you.

Gediminas Urbonas, for his continued support of my work, and willingness to engage and grow with the students. Azra Akšamija, Ben Fry and Lucy Liu for opening their classrooms and giving me the opportunity of teaching alongside them.

The person who most closely understands me, and because of that has provided the most indispensable support over these two years, intellectually, emotionally and otherwise; my parter Alice Noujaim. My admiration for you is too deep.

Finally, my parents Beatriz Cardoso and David Zylbersztajn, who have supported me in every possible sense this word can have, and my sisters Julia, Marilia and Joana, who inspire me continuously.

Table of Contents

Introduction

Software is many things at once, and none of those are easy to define or comprehend. For the past half century it has had an increasing presence in social life, and with that it has articulated a series of changes in our conducts and dispositions, touching everything from the way we listen to music to how cities are designed, with how we turn on the lights and how advertising works in between. These changes are constituted through an assemblage of agents that involve political, technical and aesthetic struggles and can be traced back — among others — to programmers, corporations, policies and politicians, researchers, armed forces, users, and the code itself.

Code, as a system of formal specification of information, and its correlate operations of encoding and decoding are not flawless intermediate channels and carry much agency in the production of meaning. Therefore, it can, and has been, used as a model for the understanding of various phenomena, from biological functions to social behavior. As this term folds out in many different directions, it is necessary to define which are the uses relevant to this project. Here, code is understood as a linguistic system for configuration and exchange of meaning, and the main objects of concern generated by such definition are 1. representation of information for human communication, 2. a parametric set of directions that generates an executable action, i.e. computer programming, and 3. the way these two may intersect.

As much as it is a useful analytical model, code is also a highly manipulable class of objects. Because of its mutual reliance with communication and, therefore, with the weaving of social fabric, it is a fundamental object in the building and maintaining of the mechanisms of symbolic power and control of everyday life. As software takes on the character of omnipresence, its prescriptive agendas also inscribe themselves in our social practices and in the way we formulate our *habitus*.

This thesis investigates software as a textual and aesthetic object through research-based artistic practice and arts-based research. Its main particular interest is in how control is built-in and exercised by codes. For that, it engages Pierre Bourdieu's framework of language and symbolic power, Stuart Hall's encoding/decoding model, and Wendy Hui Kyong Chun's notions of programability, among other concepts, aiming to discuss how to read codes in ways that create possible semantic and pragmatic negotiations with their imperatives. Following Chun, the intent is not to conceive of control through software "at the level of content [or] in terms of the many governmental projects [it has] enabled, but rather at the level of [its] architecture and [its] instrumentality."[1] In that sense, there is no intention of establishing a moral judgement of code, arguing for it being good or bad. There is just an assumption that whether positively or negatively, code does things in the world around it as consequence of its performance, and the actions it generates are, within its current architecture, necessarily involved in the production and maintenance of control and power.

The goal of the project is to show that art practice can provide different modes of reading code which may prove themselves pertinent to a less passive engagement with this subtle layer of control of everyday life. It aims to frame ways in which the manipulation of code operations can be used in art making and, conversely, how art can produce relevant knowledge about code theory, especially in its sociological implications. Reading practices in this context take on the role ascribed to them by thinkers such as Barthes, Foucault, McKenzie, Hall and others, and are thus considered as an active process, a means by which the recipient of a message can (re)configure and generate multiple meanings from it. Therefore, the intention is to apply concepts from a long tradition of media and literary theory about the polysemy of texts to the realm of

1 Wendy Hui Kyong Chun, *Programmed Visions: Software and Memory,* Software Studies (Cambridge, Mass.: MIT Press, 2011). Location 214, Kindle.

code, and by doing so supplying the receiver with agency, dealing with the imminent aspects of control and governmentality present in software without succumbing to a techno-deterministic pessimism that assumes total configurations of power emanating exclusively from the code itself. To the group of strategies that emerge from this vantage point, I designate the term "artistic readings," an idea that is constructed throughout the thesis but eventually culminates in a process of reading codes that understands software as a set of power-embedded performative utterances, tries to account for and amplify the ambiguity contained in the codification of said utterances and considers the multiplicity of sites for reading contained in every program. Most of all, artistic readings of software are any and all practices that are centered in the aesthetic-political action of meaning-making of and through code and its effects.

It is important to notice that while this is to some extent a project about understanding code, it is most definitely not a project about code literacy. While there are clear advantages to knowing how to read and write code (in the narrow sense) in how much one can purport to negotiate with or oppose its prescriptions, it is assumed that on occasion other points of access into the software will provide more relevant engagements. And, most importantly, artistic readings can expand our practices of negotiation precisely by not adhering to the formal grammar of programming. These strategies, then, should be employable in different ways by people with different levels of knowledge of "proper" programming languages.

This project is methodologically organized in a way that straddles the line between theoretical discussion and artistic experimentation. The core of the proposed methodology consists predominantly of an in-depth review of literature that deals in one way or another, frontally or tangentially, with the themes addressed in this introduction. Given the institutional setting in which this research was developed, there is a fair amount of associative and speculative thinking that goes into the selection of top-

ics covered, a freedom secured by the art field that allows us to navigate from sociology to literature, to media theory, to computer sciences, to visual arts, to music, and beyond. The intent is to produce and to propose new artistic and theoretical insights from the juxtaposing of different sources of existing investigation, creating productive dialogues between texts that might have otherwise not cross-pollinated. This aspect of the research is attached to the realization of artistic experiments originating from these readings and writings. These were not necessarily considered to be works of art, but platforms to test the aesthetic-political-epistemological effects and affects of thoughts that emerged or were crystalized in the process of making the thesis. These quasi-examples of suggested modes of reading have provided other avenues of investigation and new insight to the theoretical pursuits of the project, and punctuate the sections not to directly illustrate but to serve the role of simultaneously confirming, thwarting, debunking, distorting and opening any perspectives academically affirmed in this document.

It is important to notice that these artistic counterparts to the writing complement the research not as a means of testing and proving concepts, but as experiments that orient and shift the perception of the theoretical objects at hand. Therefore, both the writing and the artwork are expected to have a certain degree of autonomy, despite their mutual construction. Relatedly, the works here presented were conceived and formatted to be in existence in tandem with the published form and structure this thesis assumes, in a way that avoids representation in favor of presentation and description in favor of encounter. This way, the work is itself contained in the thesis, and doesn't have to rely on documentation and extensive explanation to exercise its role. Reciprocally, this approach transforms the whole thesis in what I see as work, the experiments and the text amounting to a unity of artistic discourse that can be fully realized as artwork. In fact, this is a product that operates as a condensation of

research as well as a tentative script for a publication and/or (&& ||) performance that can almost literally unfold from these pages.

The main body of theoretical content in this thesis is roughly divided into three segments. The first one presents some of the fundamental aspects of the thesis, elaborating on its definitions of code and control and the relationships between the two concepts. Bourdieu's theory of practice and its related concepts of habitus and field are then explicated, moving along to his sociology of language and incorporating into it the idea of code as an agent of symbolic power. Bourdieu's terms and concepts are then applied to understand the agents in the field of coding processes that are in position to struggle for symbolic capital, and how one can push back or negotiate their involvement with the control mechanisms of the "market" one is in. Finally, this segment presents this thesis' allegiance to reading as a potentially emancipatory meaning-making process, by summarizing a brief history of reader-inclined literary and media theory and tying the previous references to Stuart Hall's encoding/decoding model of communication.

The second segment starts by focusing on codes generated with the intention of limiting access to content, namely anti-languages and steganography. It poses questions about the phenomenological relationships between the surface text/crypto-text and the hidden text/plain-text, as to whether they affect each other in any way other than by concealment. These forms are evaluated both in their capacity to serve as a metaphor for the occlusion of software and as strategies for limiting the control that software exerts. This section then poses questions on what are the effects of occlusion over a reader in different scenarios, e.g., full knowledge, partial knowledge, and complete unawareness of the existence of a hidden message or its primary decoding method. These considerations also concern the way in which language shapes and preserves reality, and how concealed forms may still perform their operations just the same

while remaining apparently invisible. From there, computer code is analyzed through the same token of in/visibility and il/legibility. Chun is then referenced to present the limits and implications of visibility in software and to try to achieve an understanding of how one can read something one can only see partially, if at all. A tentative solution is offered through a valorization of ambiguity and opacity in reading and being read.

The last segment examines what is the ontological proximity of the code with the performative effect it generates, and how shifts and manipulations of this relational axis may work to oppose or divert prescriptive command structures. By comparing source code with the running program it is asked: are they one and the same, two completely different objects or something in between, two separate existences that at a certain level collapse into one? Again thinking with Chun, I investigate her notion of code as *logos*, a conflation of order and execution, as a main component of how and why authority is conceded to and constructed through code. Finally, also comparing this relationship with the analogous structure of the division of music in notation and performance, I aim provide a way of thinking about what kinds of results one would obtain in their negotiation with the imperatives of software by reading into the code, into its performative effects, or in the continuum that exists in between the two.

The thesis concludes with a synthetic formulation of what an artistic mode of reading can be and what one can expect by engaging it to negotiate with authority in the form of code architectures.

Code, Command, Control and Communication

Claude Shannon's *A Mathematical Theory of Communication* has set the grounds for one of the most used and reconfigured theoretical models in recent scientific history — the structure in which a sender emits a message through a channel to a receiver, subject to interference (or noise).[2] Initially referring exclusively to signal transmission and explicitly disengaging the idea of *information* from that of *meaning*[3], these ideas, structures, and vocabulary were soon picked up by the social sciences and used to study human communication, education, psychology, and social theory, to name a few. Information theory's sibling discipline, Cybernetics, is according to its forefather Norbert Weiner "the scientific study of control and communication in the animal and the machine."[4] The word is derived from the Greek term κυβερνήτης (*kybernetes*), meaning most literally "steersman" or, by analogy and in the Greek sense, "governor."

Both lines of inquiry were instrumental in establishing a new mode of addressing human experience through an analytical toolset concerned with technological systems. While this has many historical analogs, most famously the bout of metaphors in the late 18th and early 19th centuries that used steam engines and mechanical gears as a base for understanding pretty much anything, this turn in the beginning of the 20th century has many specificities. One of which, importantly, is that it removes the metaphorical component, as it is not a suggestion that, for instance, a casual greeting behaves *like* the transmission of an electrical signal through a copper wire, but that it is indeed practically and structurally the *same thing* as a subject for analysis. Another is that it

2 Claude Elwood Shannon, "A Mathematical Theory of Communication," *The Bell System Technical Journal. Vol. 27*, pp. 379—423 (July 1948), 380

3 Shannon, "A Mathematical Theory of Communication," 379

4 Norbert Weiner. *Cybernetics, Or, Control and Communication in the Animal and the Machine.* (Cambridge, Mass.: M.I.T. Press, 1961),

emphasizes communication as a locus of governance and systematizes it on the basis of technical principles — optimization, efficiency, falsifiability — at the same time as it tries to account for uncertainty — entropy, ambiguity, noise — through precise models.

Control and Communication, the pair Wiener suggests are to be equally evaluated in the animal and the machine, might ring a bell as a doublet. And that might be because one has heard of the military doctrine usually abbreviated as C3: Command, Control and Communication, "the exercise of authority and direction by a properly designated [individual] over assigned [resources] in the accomplishment of a [common goal],"[5] aided by a structure of communication that allows for the proper flow of information and the regulation of the execution of the assignment. Of course, both Wiener and Shannon were at some point or another involved in the war effort: while Shannon was under contract with the National Defense Research Committee through Bell Labs, developing fire-control systems and cryptographic solutions for the American National Forces during World War II, Wiener had a major role developing anti-aircraft control systems at the Massachusetts Institute of Technology during WWII and was actually in the U.S. Army as a foot soldier, during World War I. As the two of them, many of the early developers of what would eventually be the field of computer sciences spent a relevant amount of the late 1930s and early to mid-1940s invested in war labor. As it is now fairly well documented, the history of modern computation is indebted to work produced during and immediately after this historical context by characters such as Vannevar Bush (MIT, head of the U.S. Office of Scientific Research and Development), John van Neumann (Institute for Advanced Study, Manhattan Project scientist), Alan Turing (cryptanalyst at Bletchley Park) and Grace Murray Hopper (Harvard University, Navy officer).

5 Neville Stanton, Christopher Baber, and Don Harris. *Modelling Command and Control: Event Analysis of Systemic Teamwork* (Ashgate Publishing, Ltd., 2008)

Software, Wendy Chun argues, "draw[s] from a series of imperatives that stem from World War II command and control structures."[6] It relies on "'Yes, Sir' in response to short declarative sentences and imperatives that are in essence commands"[7] and such commands "lie at the core of the cybernetic conflation of human with machine."[8] In a certain sense, Cybernetics' desire for control through communication becomes a self-realized prophecy, as new structures of command are built in and via software, based on the principles of executability carried by the discipline, and in fact operate in the world in a way that enforces its imperative structures. It implies a codification of agency, both in the structure of the computation and in the effect it produces.

"Code is law", Lawrence Lessig's maxim, is a very direct way to approach the manner in which control is produced on everyday life through code. The author pairs up what he calls "East Coast Code," the actual U.S. government issued legal code, with "West Coast Code," computer code, suggesting that one is akin to the other, the latter having a similar regulatory effect over people as the former, so much so that it can even do as much as contest the authority of certain legal dispositions.[9] The "architecture", as Lessig puts it, or the infrastructure of the software space is articulated as a series of directives that are determined by the specific objectives of the programmer(s). "Code is never found; it is only ever made, and only ever made by us."[10] As this text will expand on later, code is necessarily, as any other enunciation, a marker of its maker's expectations, social and cultural capital, and the discursive regimes they are a part of.

6 Chun, *Programmed Visions*, location 474

7 Chun, *Programmed Visions*, location 486

8 Chun, *Programmed Visions*, location 489

9 Lawrence Lessig, *Code: And Other Laws of Cyberspace* (New York, N.Y. : Basic Books, 1999).

10 Lawrence Lessig, *Code* (New York : Basic Books, 2006), 6

In a more pragmatic sense, computation is part of virtually every segment of life in the globalized urban environment. Even outside of this scope, few are the people that have not been at least indirectly related to one kind or another of software-based system, from national registration databases to satellite imaging. One might not know of their relationship with software, but the software, its operators, and its operations do. On this scale, it is easy to understand how code is implicated in the control of everyday life (although not so easy to understand how to react to that knowledge). As citizens we are indexed, surveilled, inspected, assessed, admitted/denied, measured, as consumers we are logged, processed, offered, charged, charged off, and as users we are bought, sold, captured, filtered — all by means of code. Channeling Foucault, Chun states that software "coincides with and embodies larger changes within … *governmentality*."[11] This concept is not limited to what is in the state's ability to govern, but comprehends other institutions and actions that can affect a "conduct of conduct,"[12] governing, steering or shaping an individual's or group's posture regarding themselves or others. The software layer that articulates these operations embody and exercise a specific plan for a common infrastructure of governmentality. It is, in that sense, a disciplinary tool for a specific discursive regime.

The spacial arrangements and distributions that this use of code engenders also attest to the effect it possesses in regulatory efforts. Kitchin and Dodge discern four levels of activity through which software is embedded in the fabric of the quotidian: coded objects, coded infrastructures, coded processes and coded assemblages.[13] Coded objects are those that either depend on the execution of code in themselves

11 Chun, *Programmed Visions*, location 171, emphasis in the original

12 Colin Gordon, "Governmental Rationality: An Introduction," in *The Foucault Effect: Studies in Governmentality,* ed. Graham Burchell et al. (Chicago: Chicago University Press,1991), 3 quoted in Chun, *Programmed Visions,* location 174

13 Rob Kitchin and Martin Dodge, *Code/Space: Software and Everyday Life, Software Studies* (Cambridge, Mass. : MIT Press, 2011), 5

to perform their intended function, such as an electronic scale, or that do not have any software enclosed within themselves, but depend on an external code reading to operate, such as a credit card. Coded infrastructures "are both networks that link coded objects together and infrastructures that are monitored and regulated, fully or in part, by software."[14] These might be distributed, like utility networks that coordinate the delivery of water, electricity, gas and sanitation, or closed systems, such as access control in a building. Coded processes are the digital flows and transactions of data through the coded infrastructures. These are not the processes used to regulate the infrastructures themselves, but "structured capta[15] and processed information"[16] that can be accessed in order to "verify, monitor and regulate"[17] individuals' actions and interactions. When associated with relational databases they allow for cross-referencing and precise comparisons aided by software. These processes thus allow, for instance, for credit analysis or other financial practices as simple as withdrawing cash from an ATM. Finally, coded assemblages are the convergence of multiple coded infrastructures in a complex system that might involve nested or parallel arrangements, the presence of coded processes in some but not in other of its components, and that amounts to something that is "greater than the sum of its parts."[18] Passenger air travel and airports, for instance, are an assemblage of the infrastructures and processes involved in "billing, ticketing, check-in, baggage routing, security screening, customs, immigration, air traffic control, airplane instruments, and so on."[19] This particular ex-

14 Kitchin and Dodge, *Code/Space*, 6

15 'capta' is the term used by the authors to refer to "units that have been selected and harvested from the sum of all potential data" — Kitchin and Dodge, *Code/Space*, 261

16 Kitchin and Dodge, *Code/Space*, 6

17 Kitchin and Dodge, *Code/Space*, 6

18 Kitchin and Dodge, *Code/Space*, 7

19 Kitchin and Dodge, *Code/Space*, 7

ample can be a very pronounced argument to describe the spacial configurations that emerge from the structuring and use of code, and also point quite clearly to the twofold aspect of control in software: in one layer, each of these objects, processes and infrastructures is subject to a series of prescriptions on how they themselves must operate, following the military-inherited "command and control" structure of modern computation to the level of execution. In a second layer, each of these prescriptions exert control over an external subject, sanctioning their movements, security, permissions and comfort within a designated space.

While this is a clearer picture, it is very important to highlight, specially within the context of the argument this thesis intends to produce, that it is not only through the manufacturing of explicit authority and management devices that code underlies a regulatory effort. It is deployed in multiple stratas of life and provides imperatives to all kinds of inconspicuous activities. Programs embed power structures in subtle ways. As in the case of digital texts, for instance, Tenen argues that despite not usually figuring in the forefront of our theories of meaning-making, "format governs access."[20] "Code determines its audience, privileging certain voices and modes of reading"[21] and it "is an exercise of power, not its representation"[22] as it is what shapes the visible written word according to its own formulations, relating "matter to content."[23] According to the author's example, it is not equivalent to a restraining order, something that "signifies the calling forth of codified power"[24] but to a handcuff, an actual physical restraint that

20 Dennis Tenen, *Plain Text: The Poetics of Computation* (Stanford, California : Stanford University Press, 2017), location 1910, Kindle.

21 Tenen, *Plain Text*, location 1910

22 Tenen, *Plain Text*, location 1910

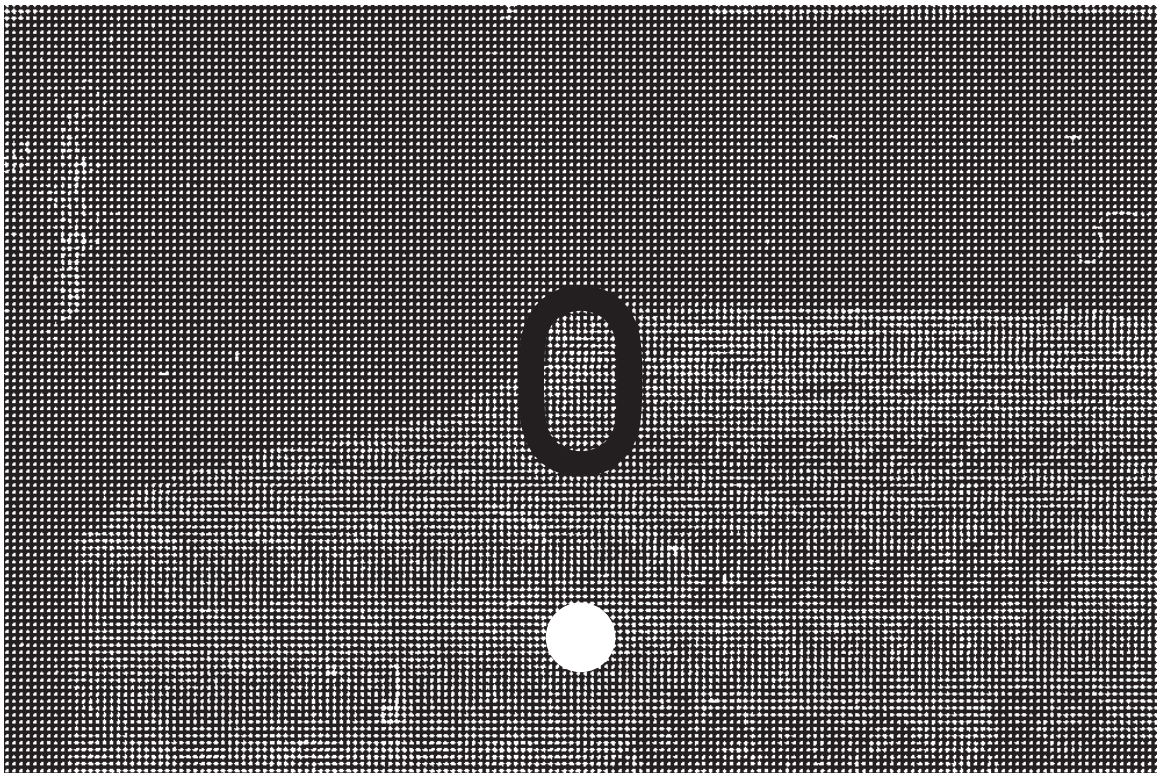
23 Tenen, *Plain Text*, location 1923

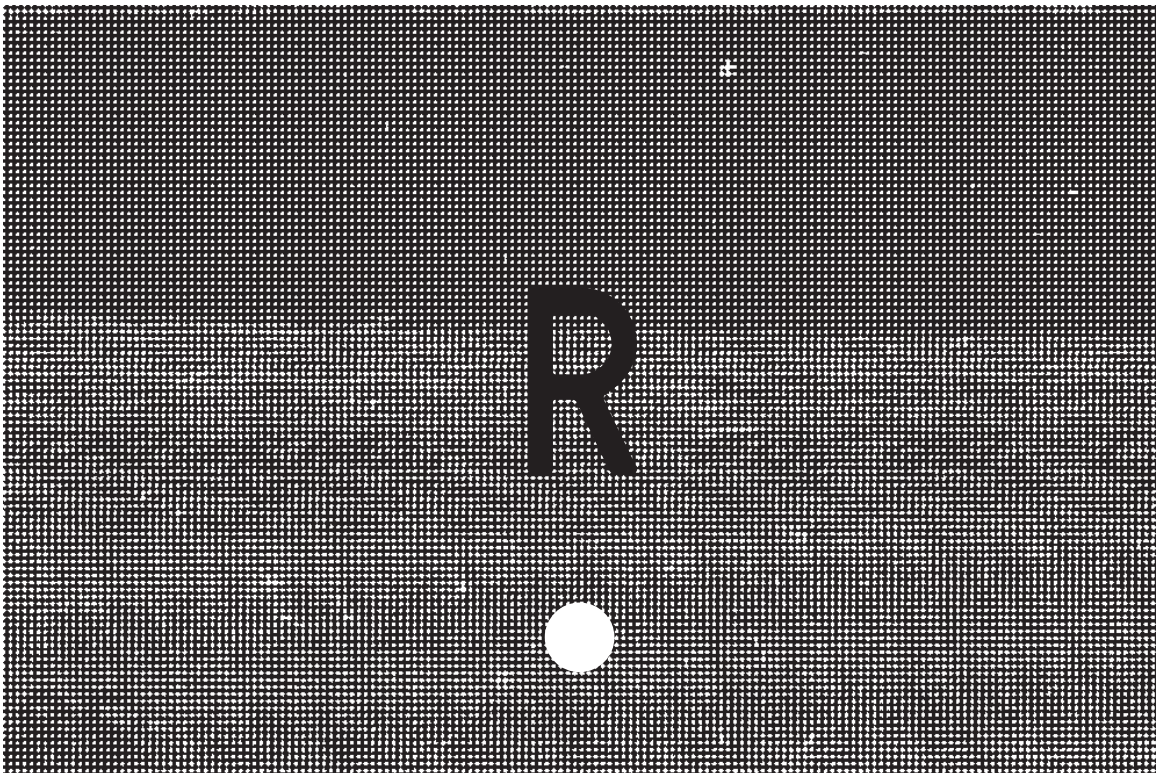
24 Tenen, *Plain Text*, location 1910

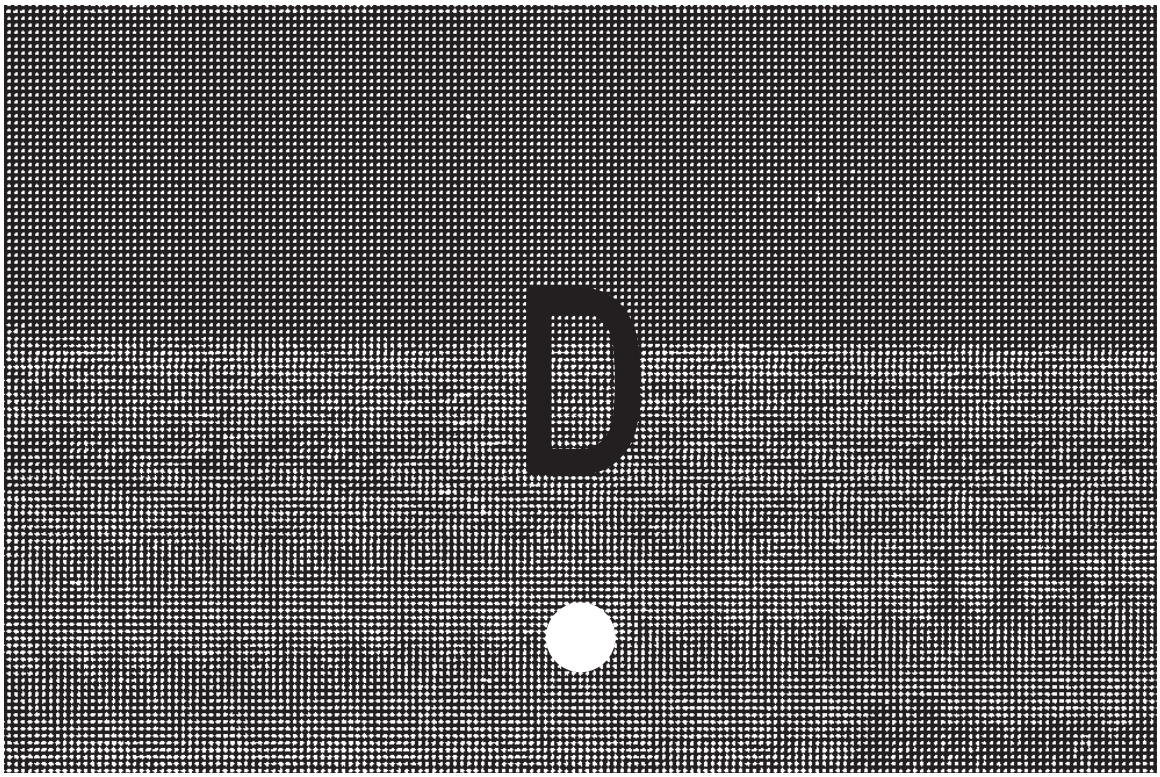
"*enact[s]* the exercise of codified power."[25] As will be argued in larger depth soon, I would equate the code in this metaphor as something closer to the police officer — it signifies the ever-present ability to restrain based on an a priori power structure, but is at times avertible, by means of confrontation, fleeing or negotiation. Code exerts control by having utterances of power embedded into itself — it is an instrument that subtly (or often loudly) directs our dispositions regarding a specific action.

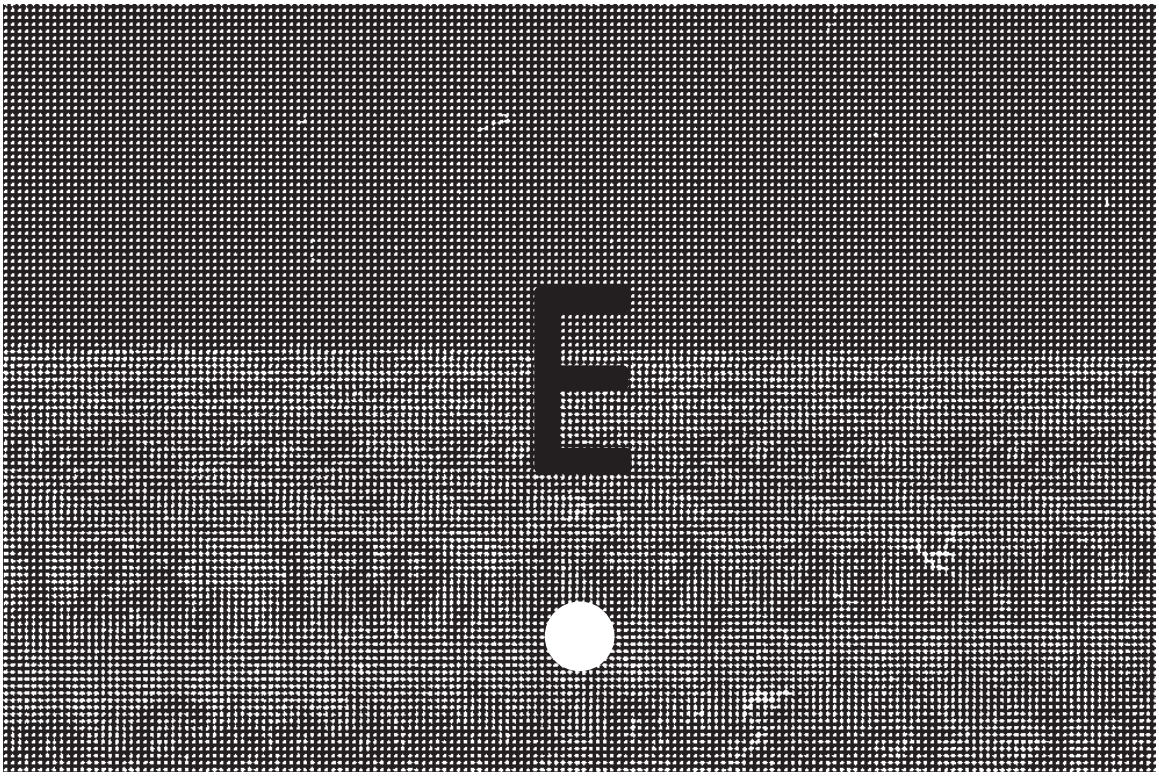
25 Tenen, *Plain Text*, location 1910

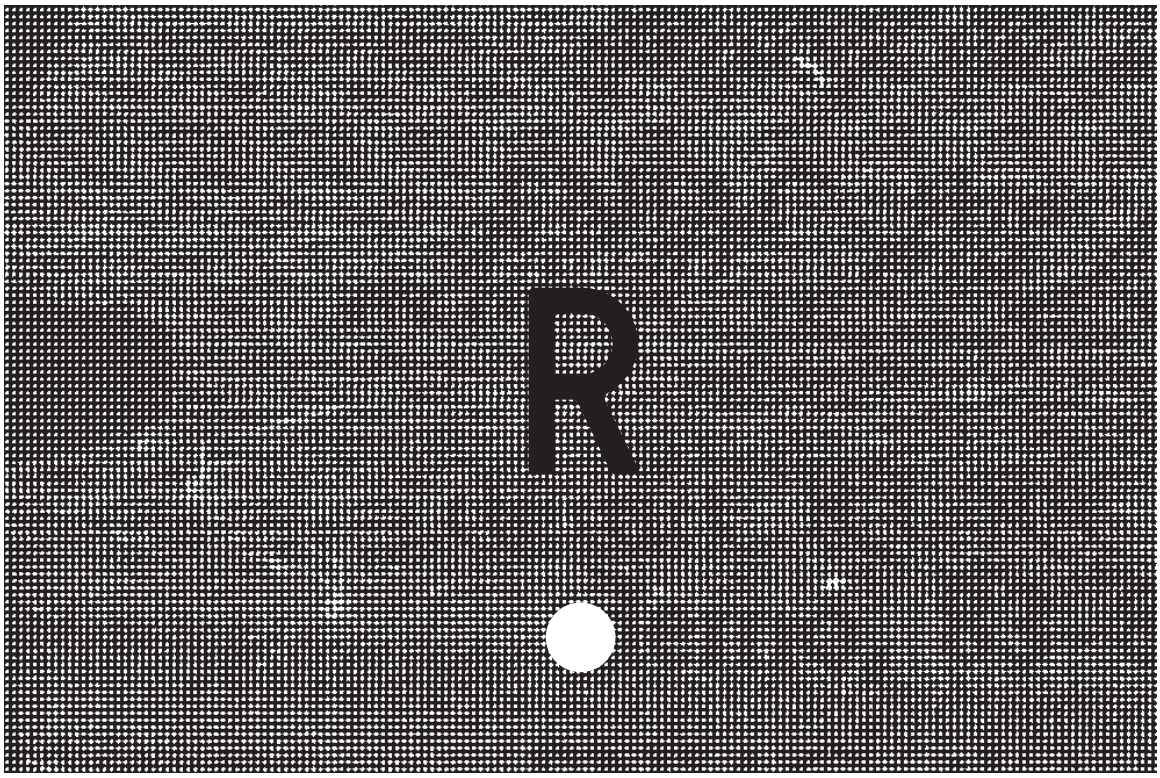
ORDERANDEXECUTION

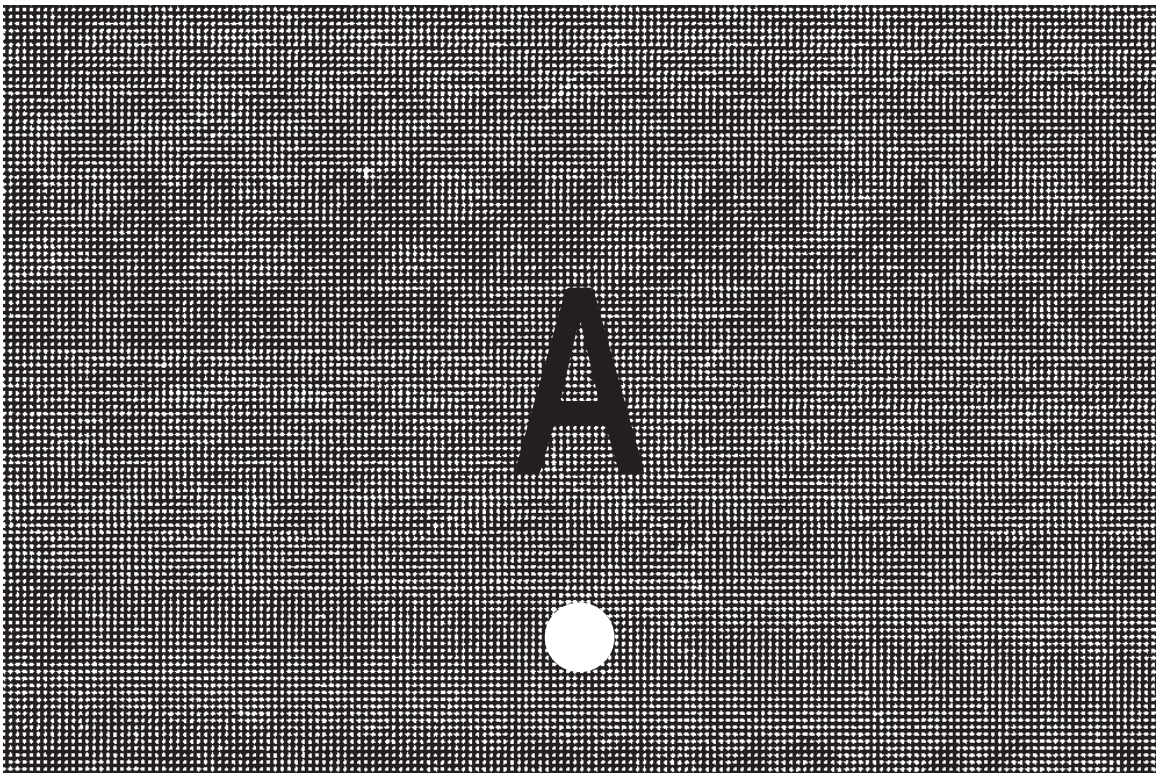


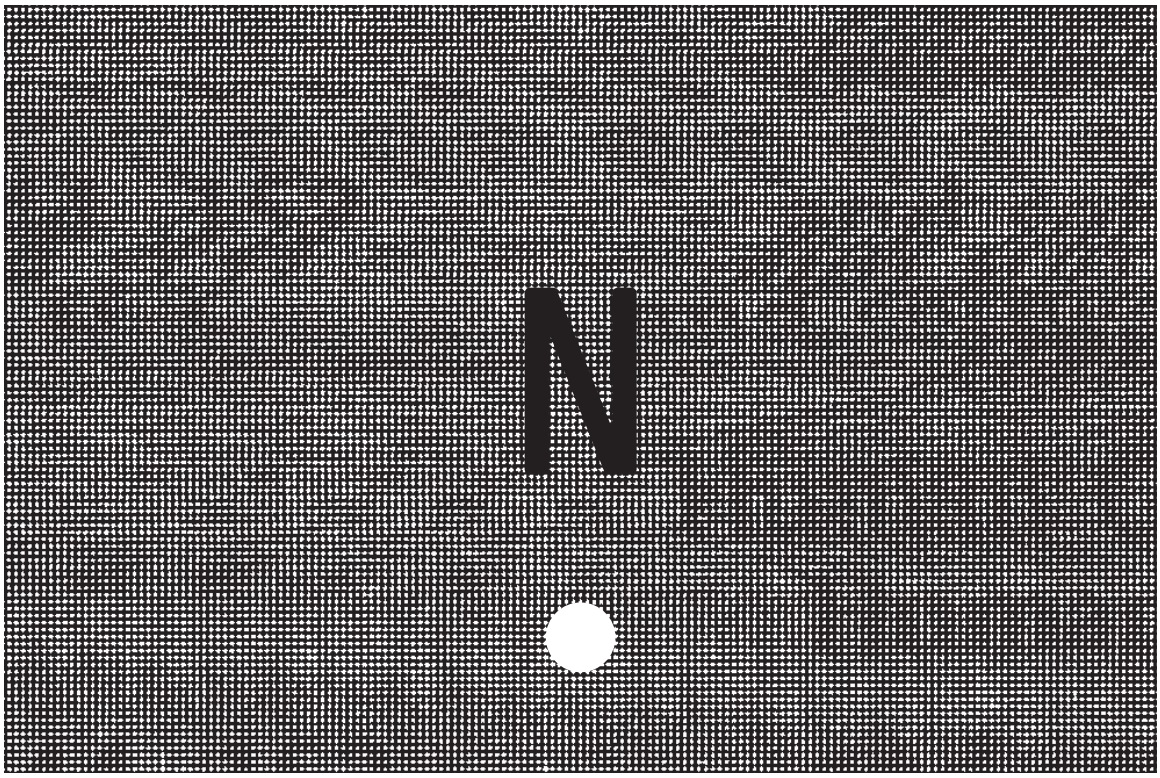

R

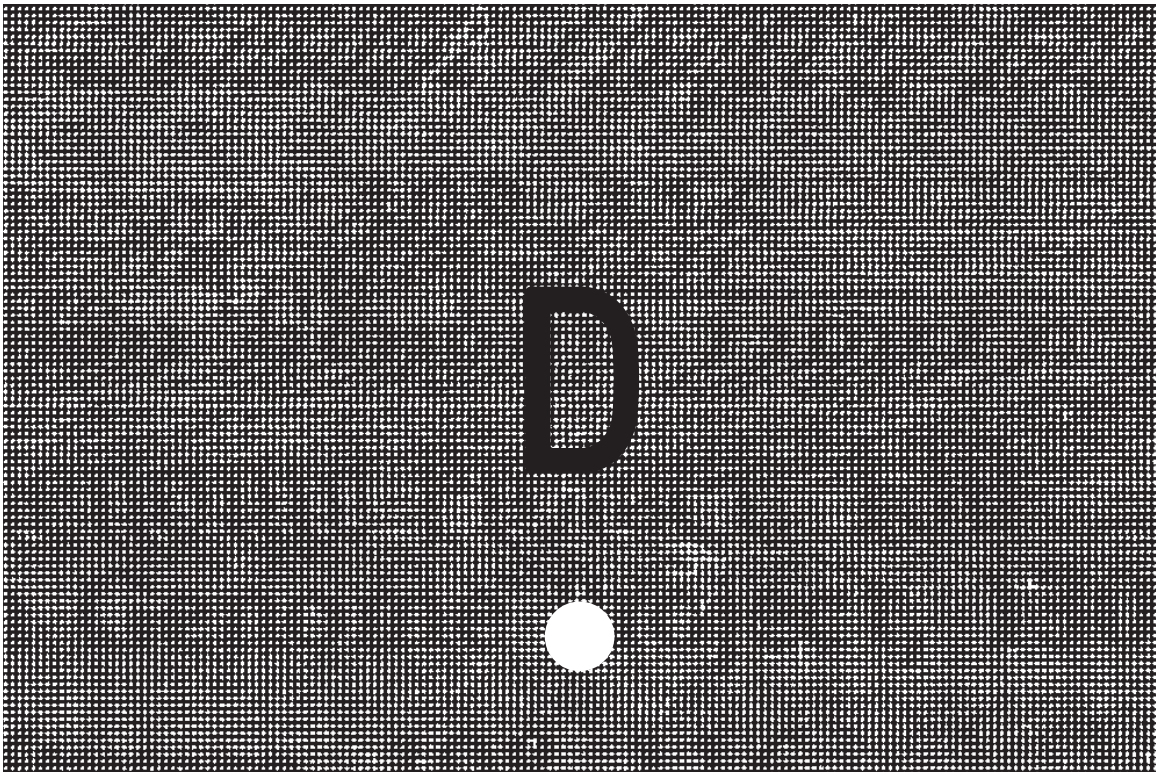

D

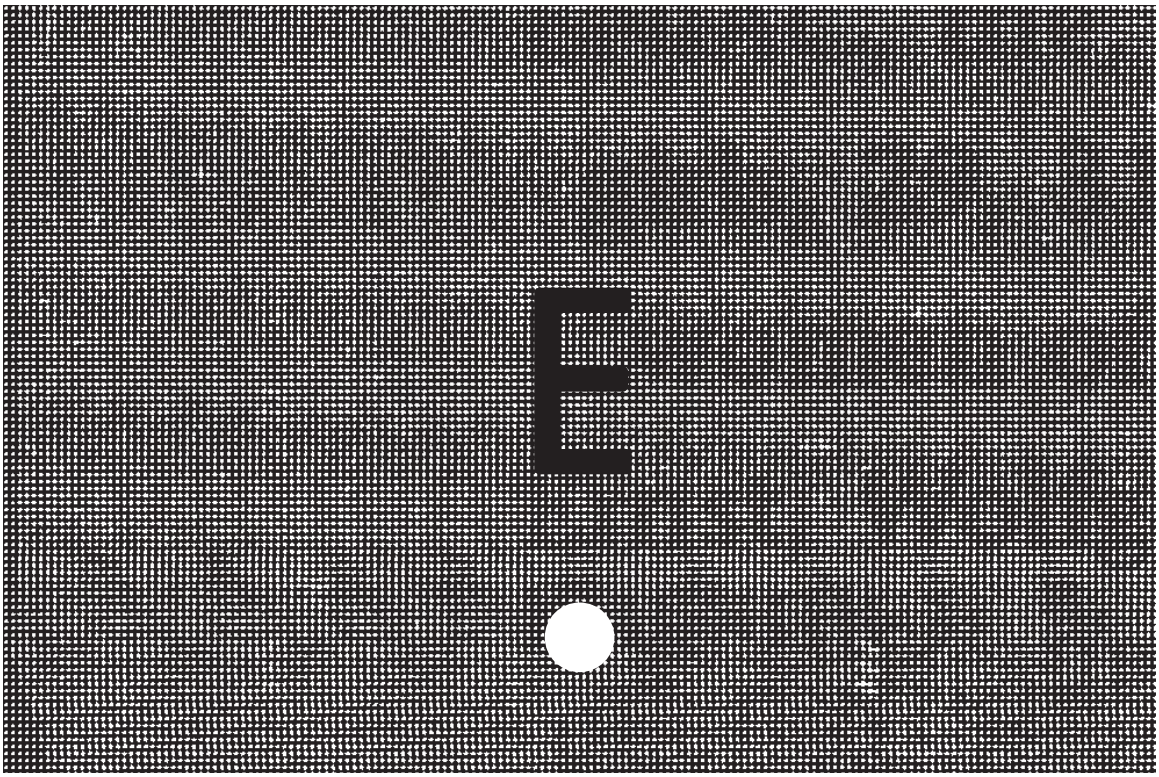


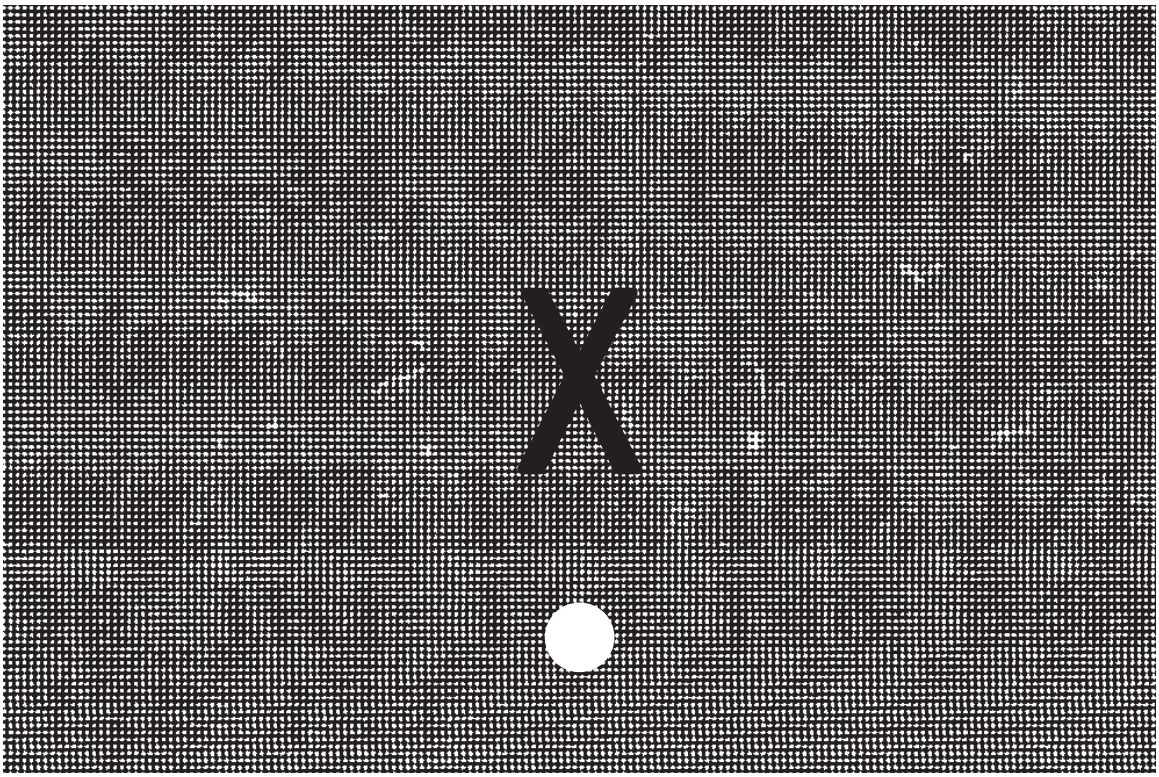

R

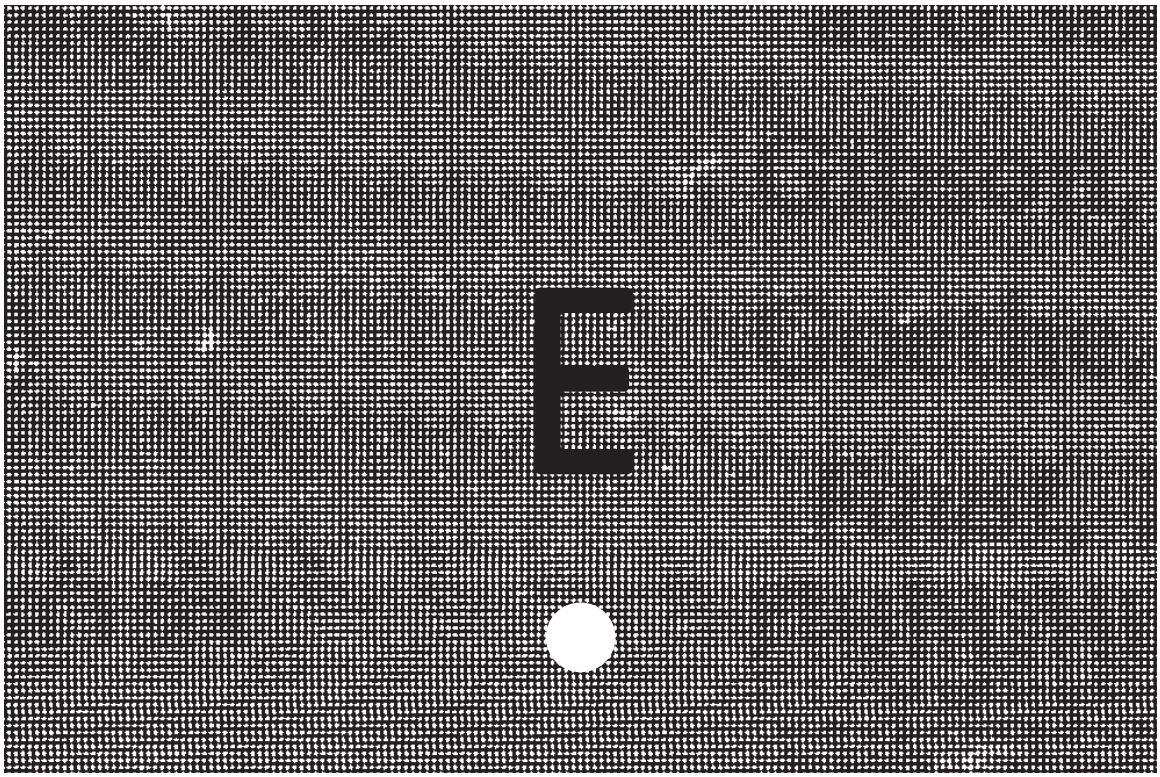

A

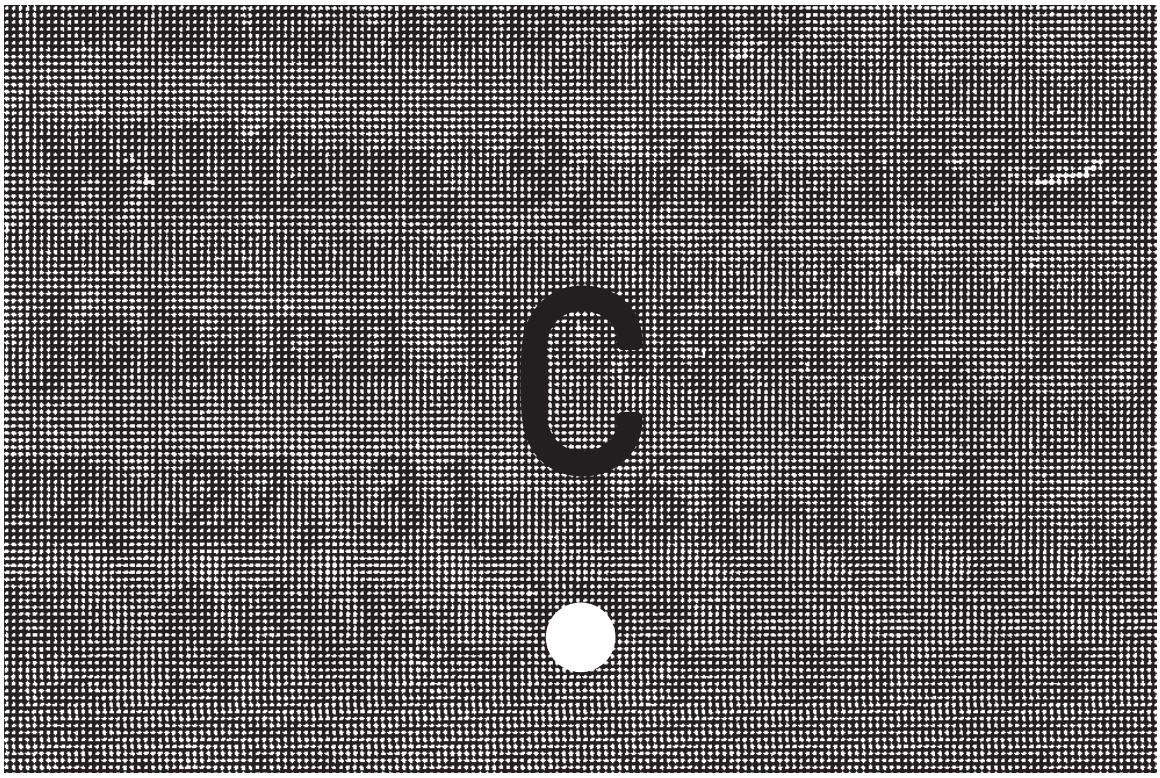

N

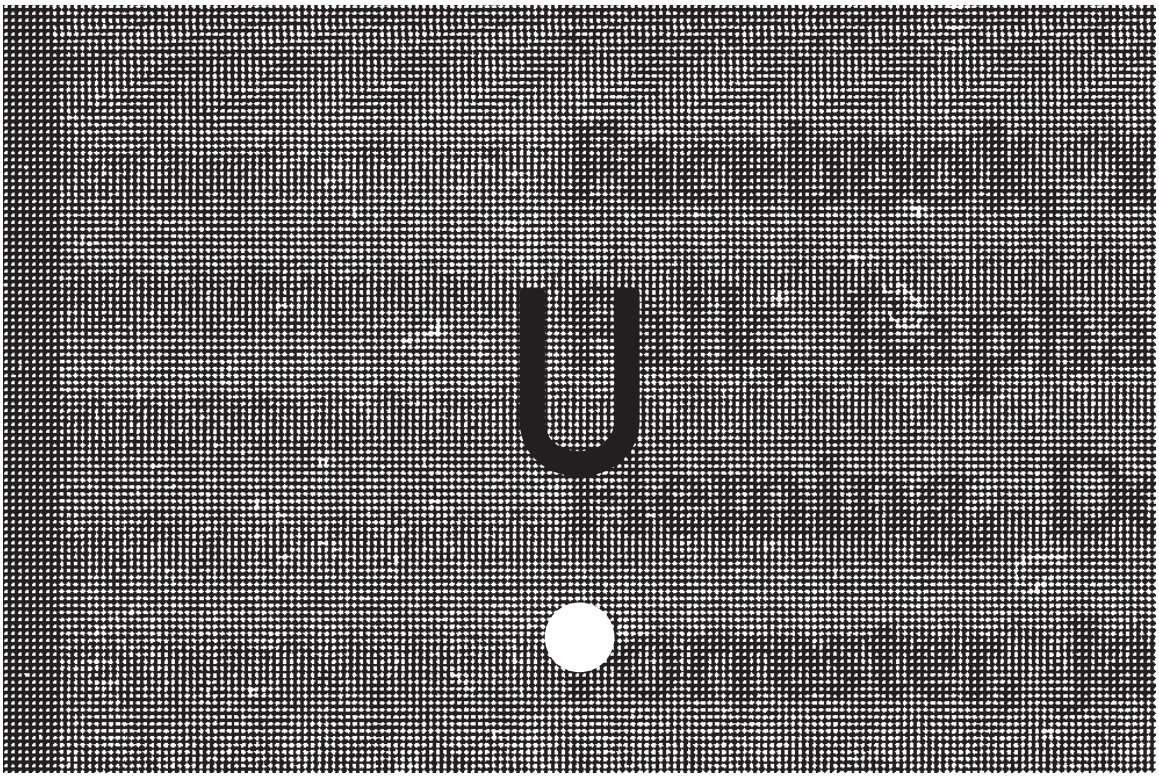

D

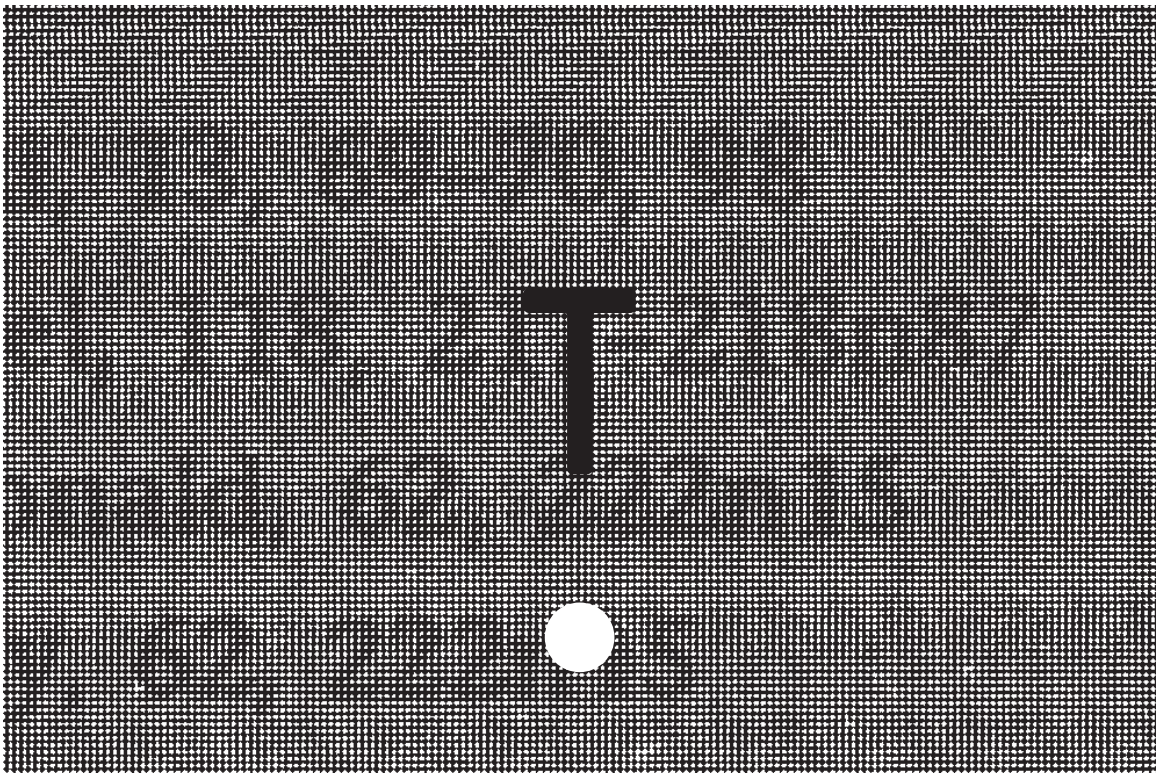


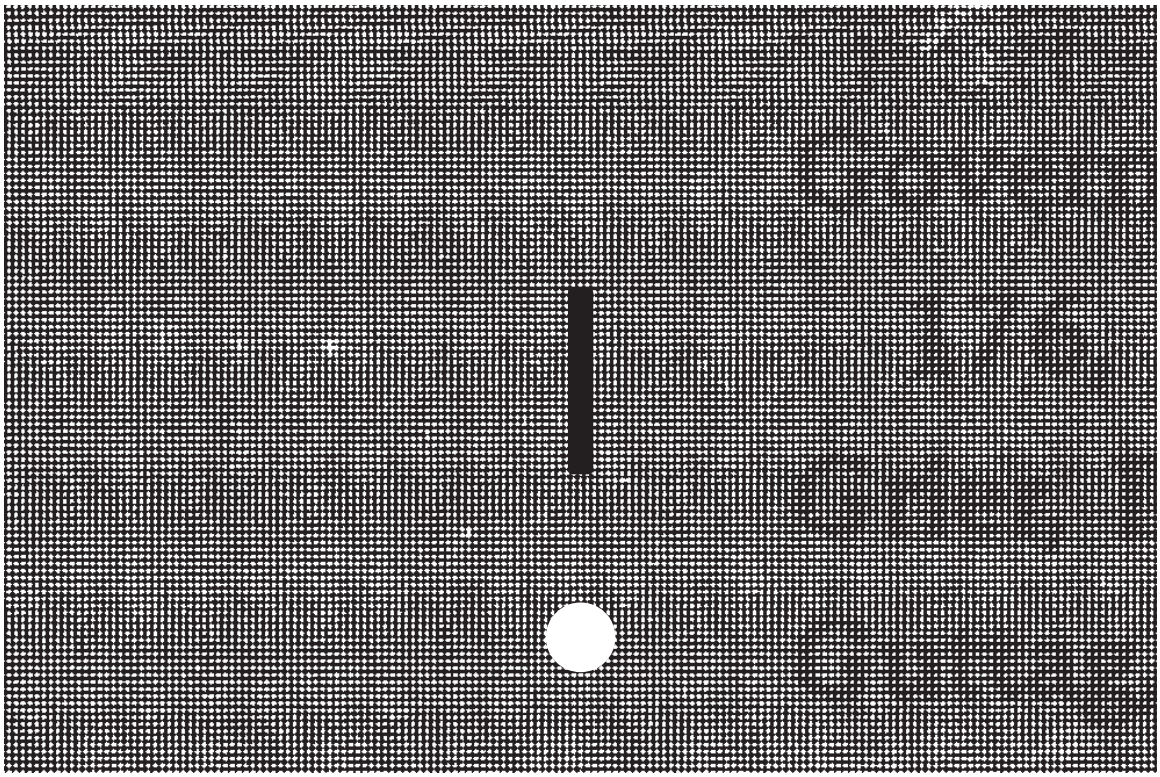

X

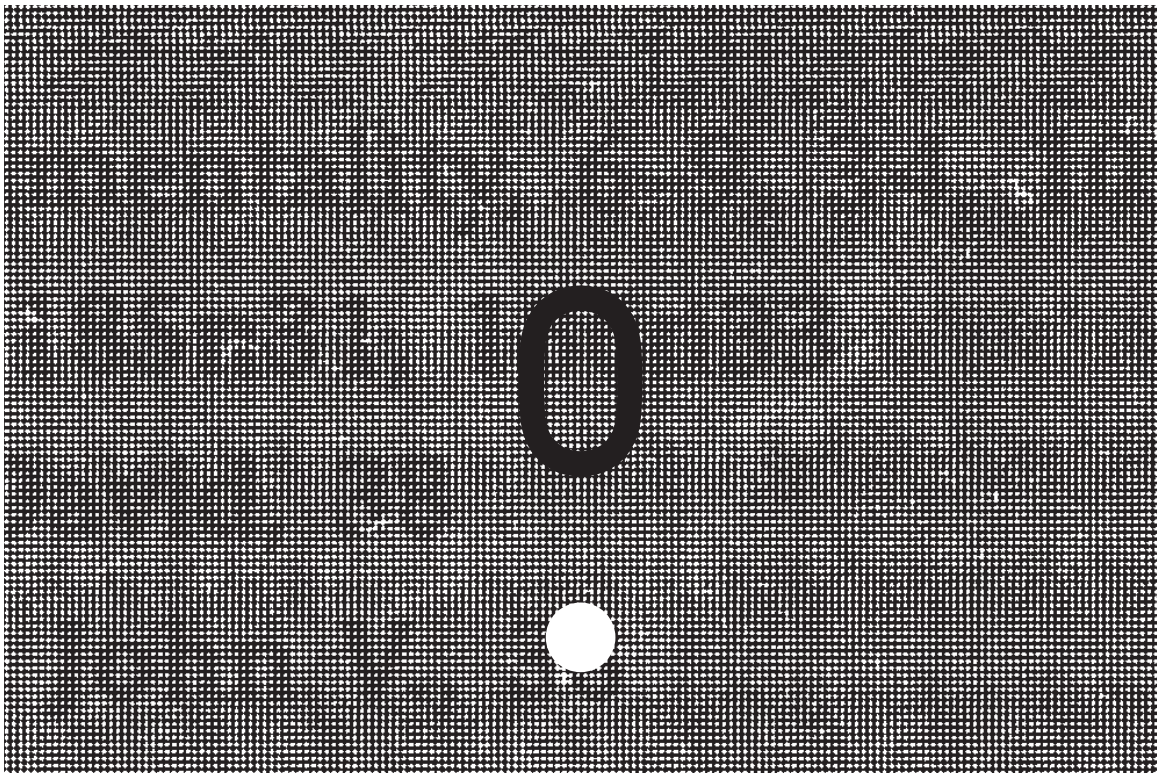

E

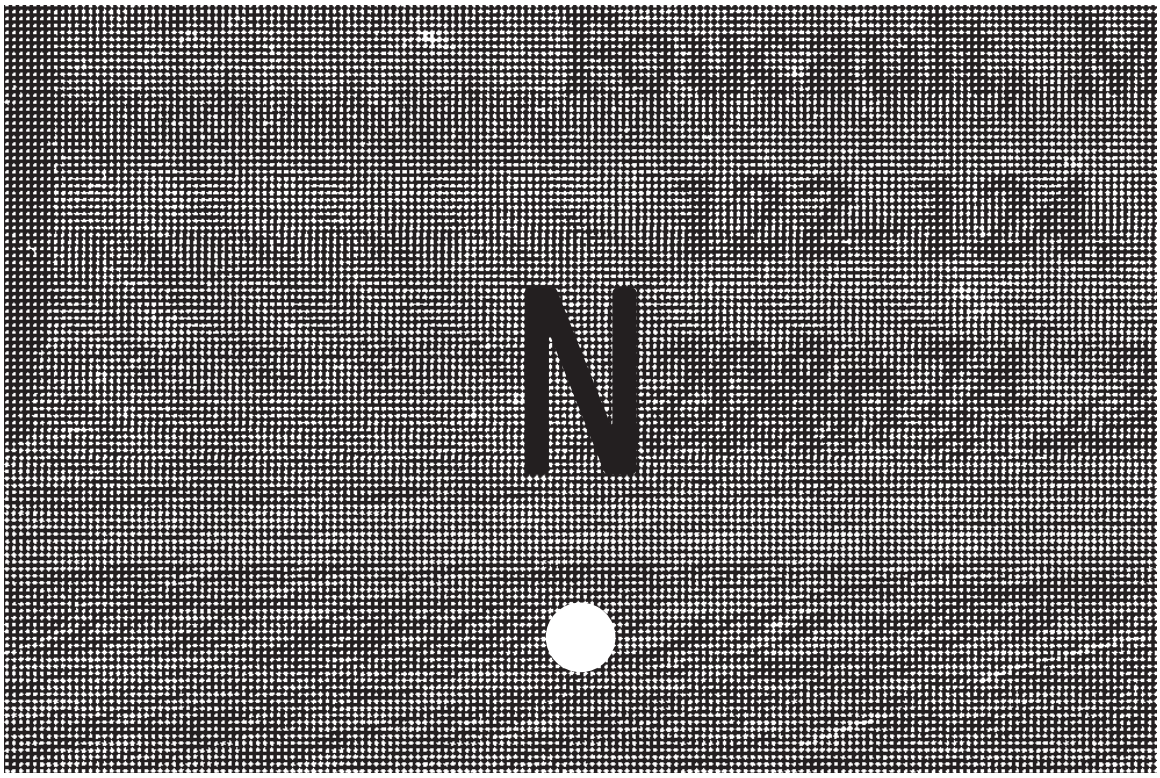

C

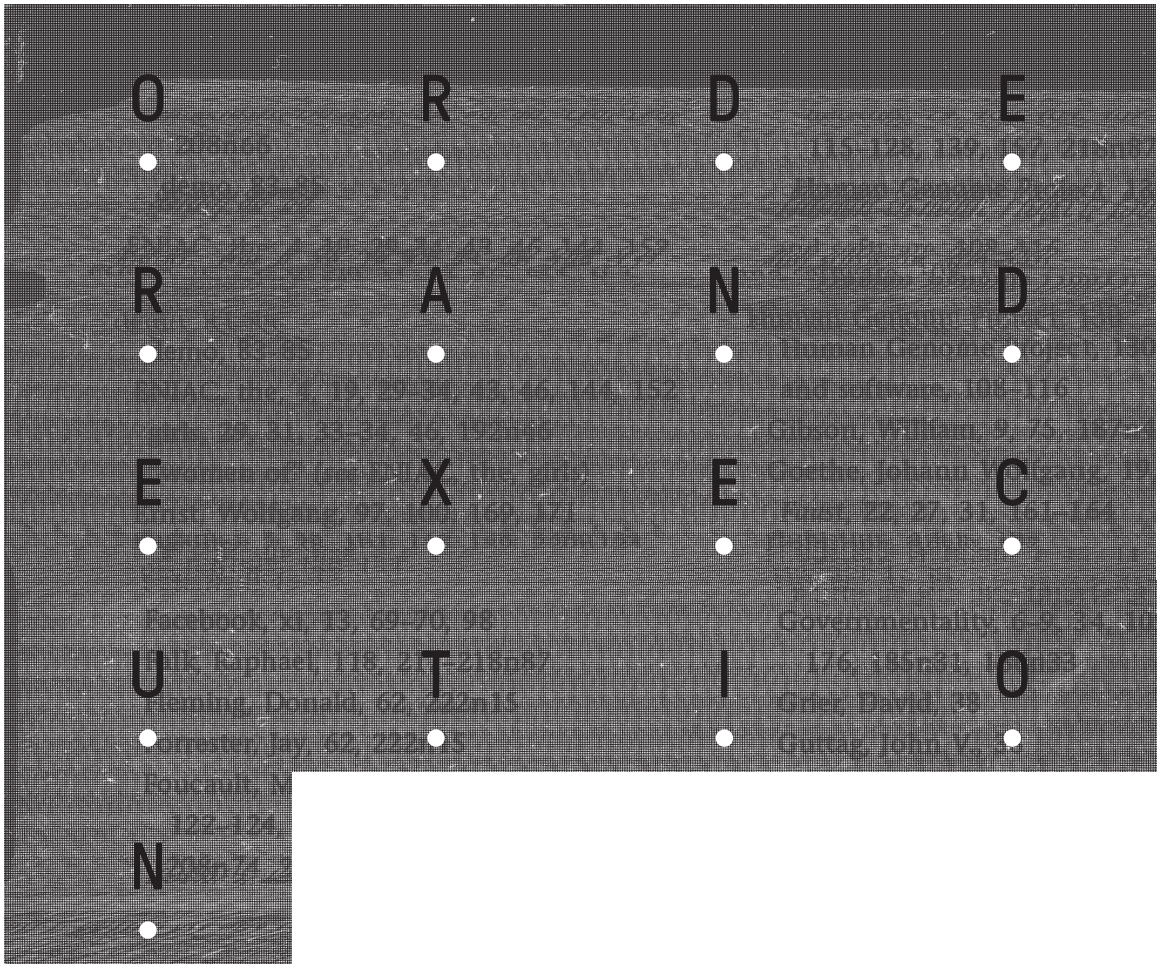

U

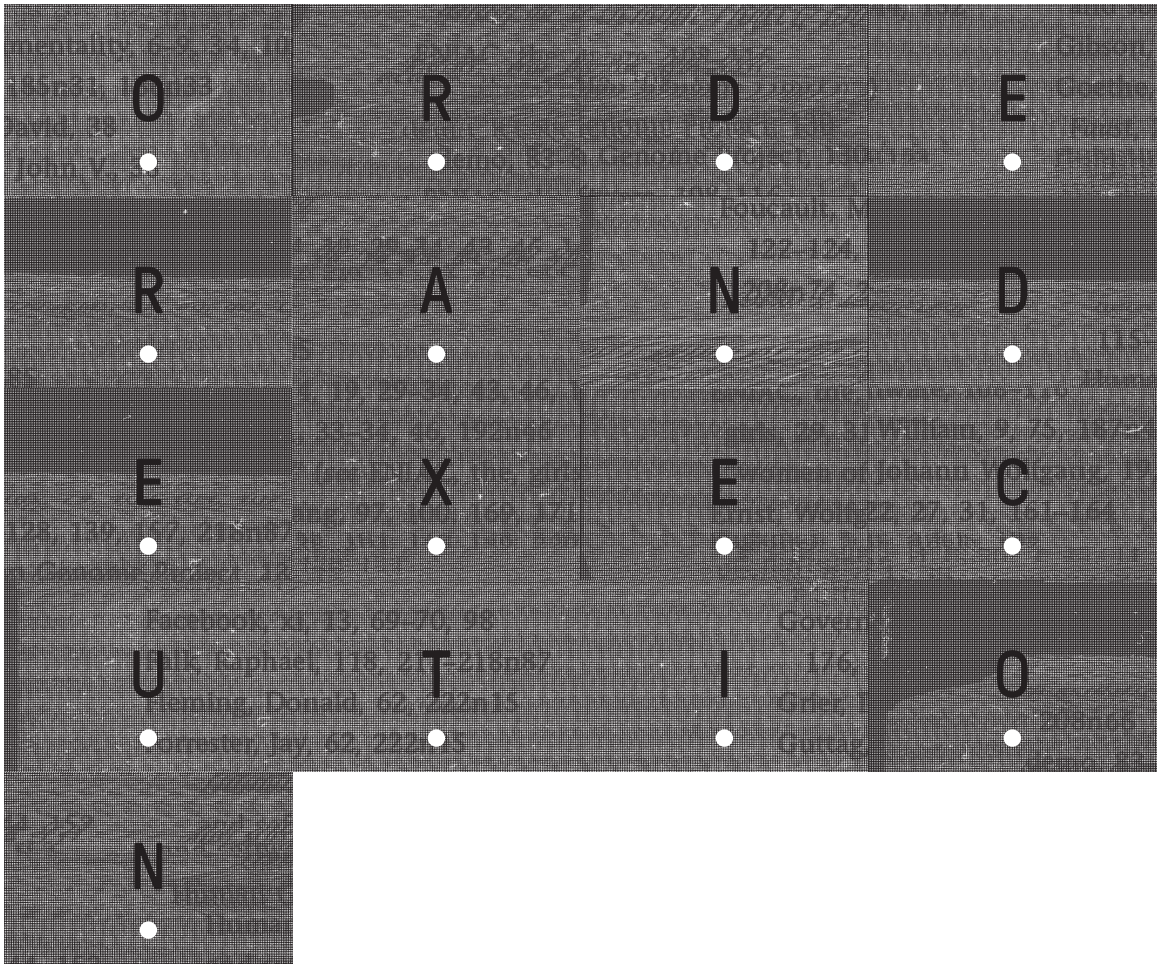


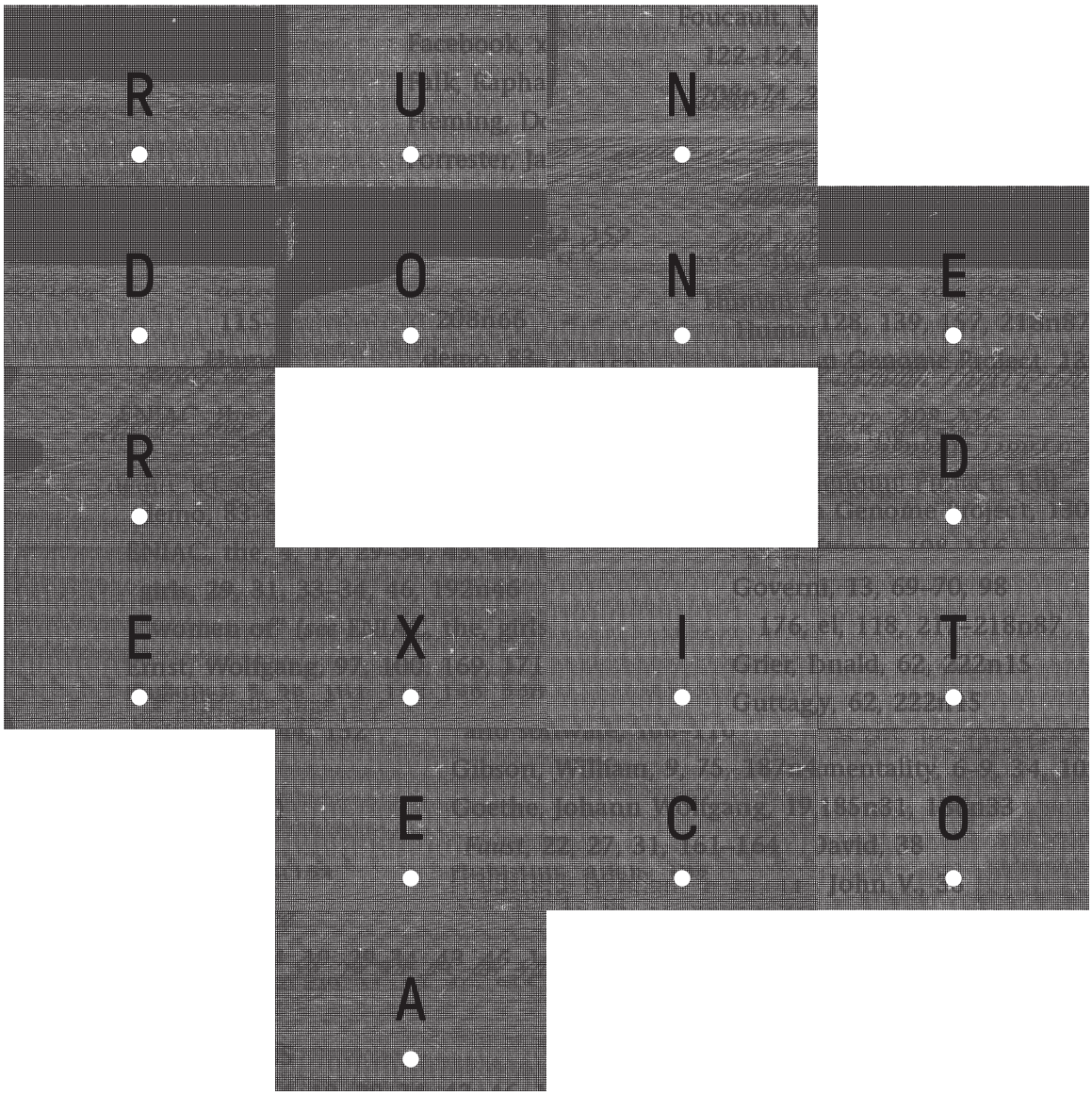


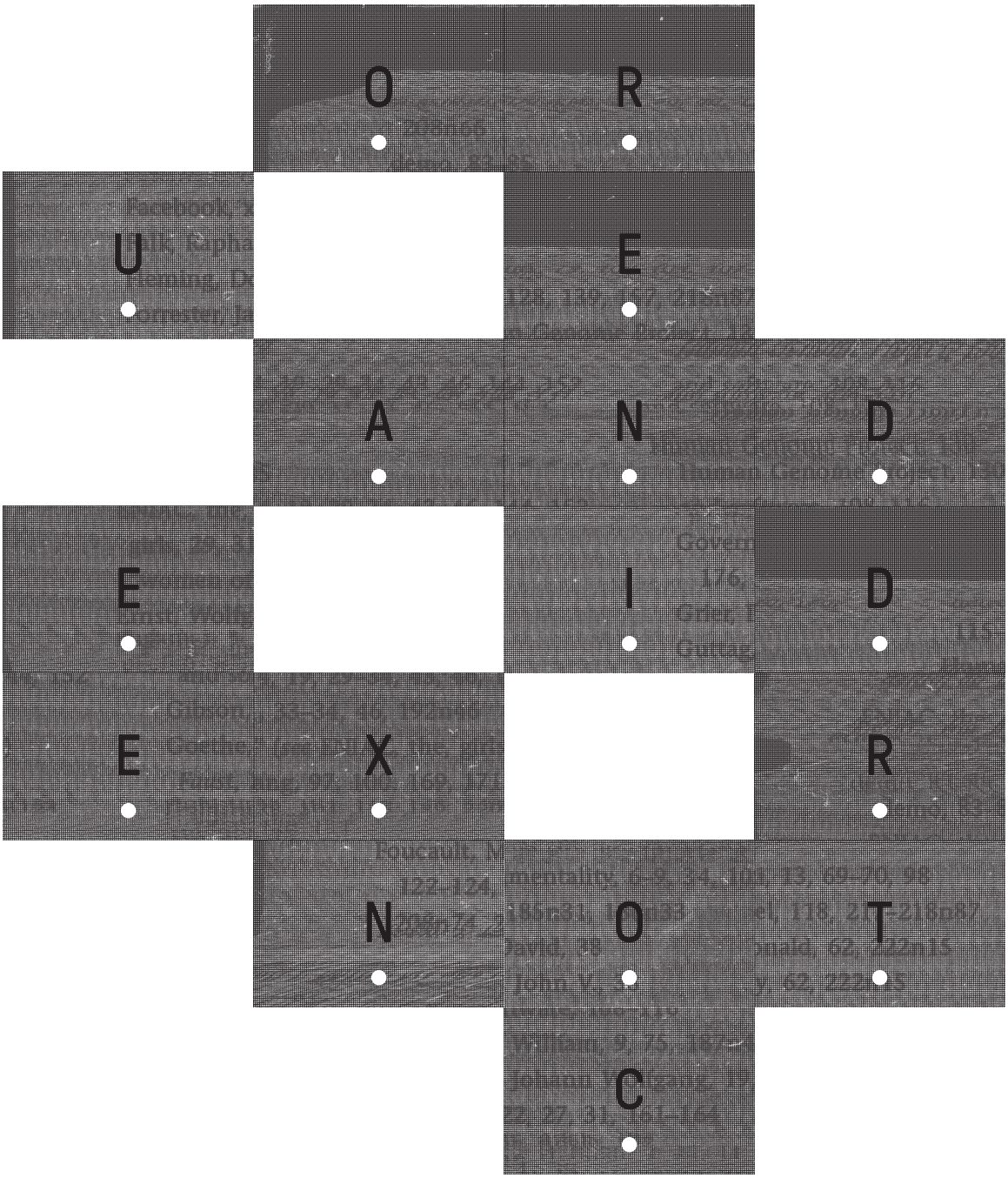


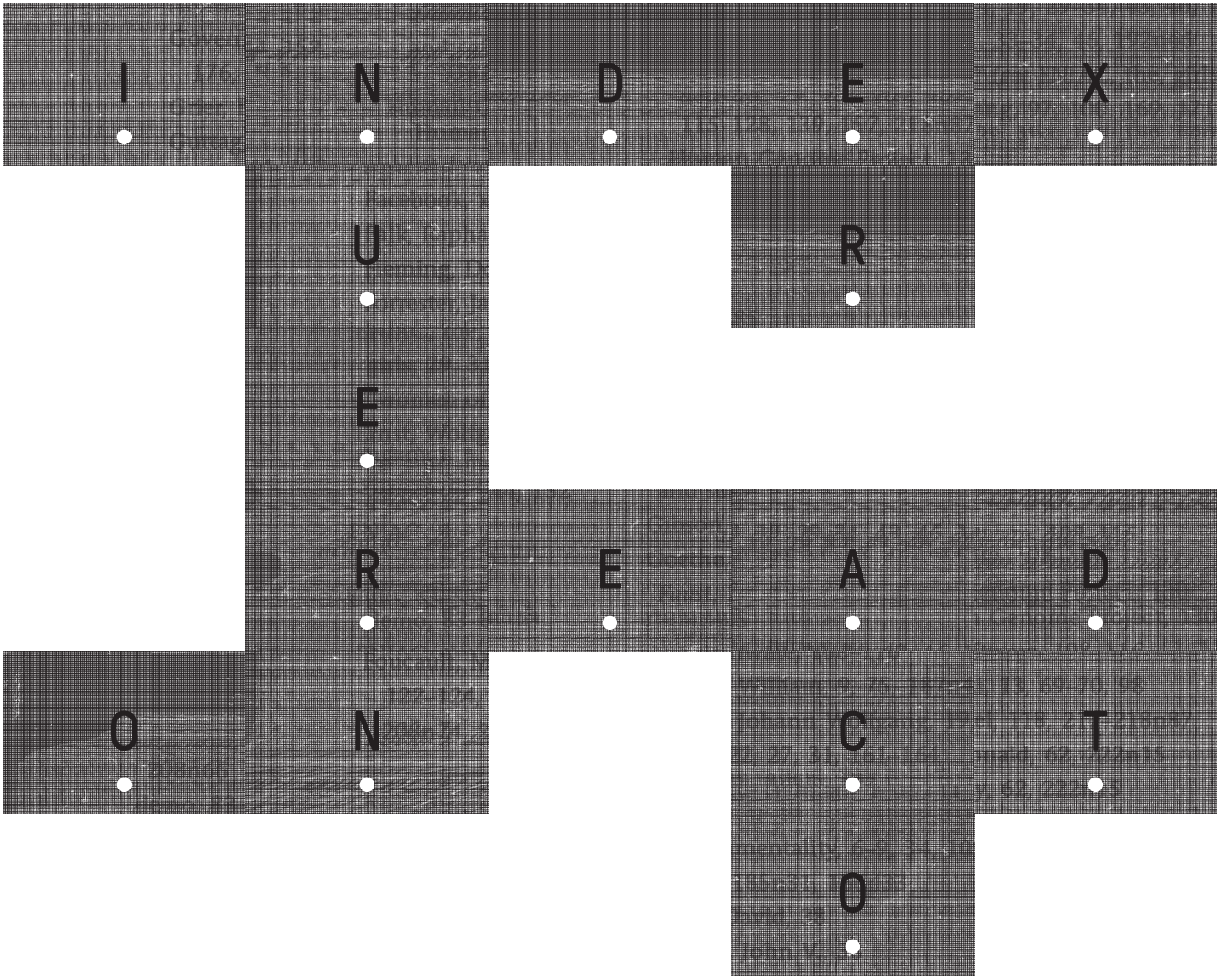

N

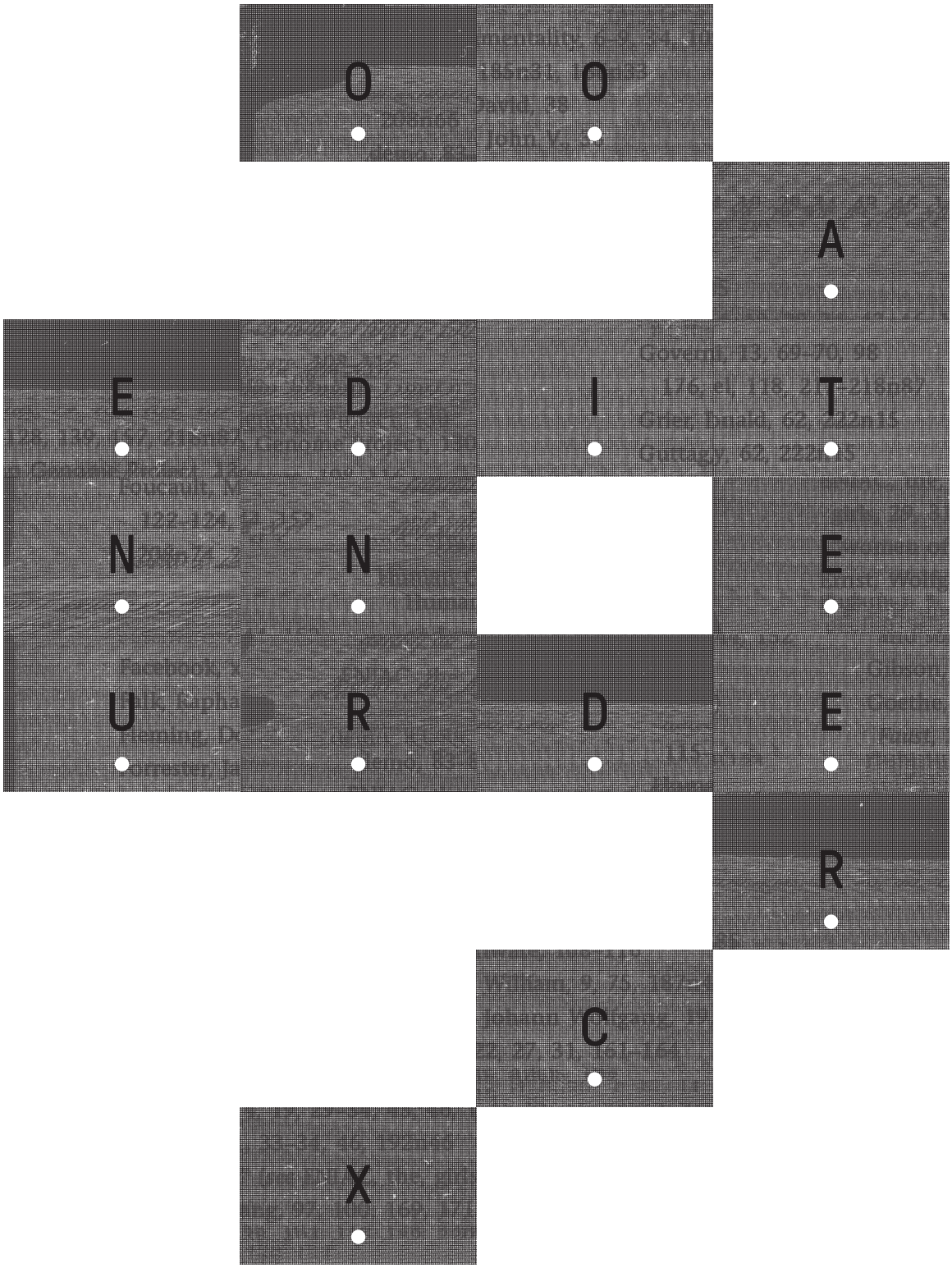

ORDER
RANDEC
EXENTIO
UNTION

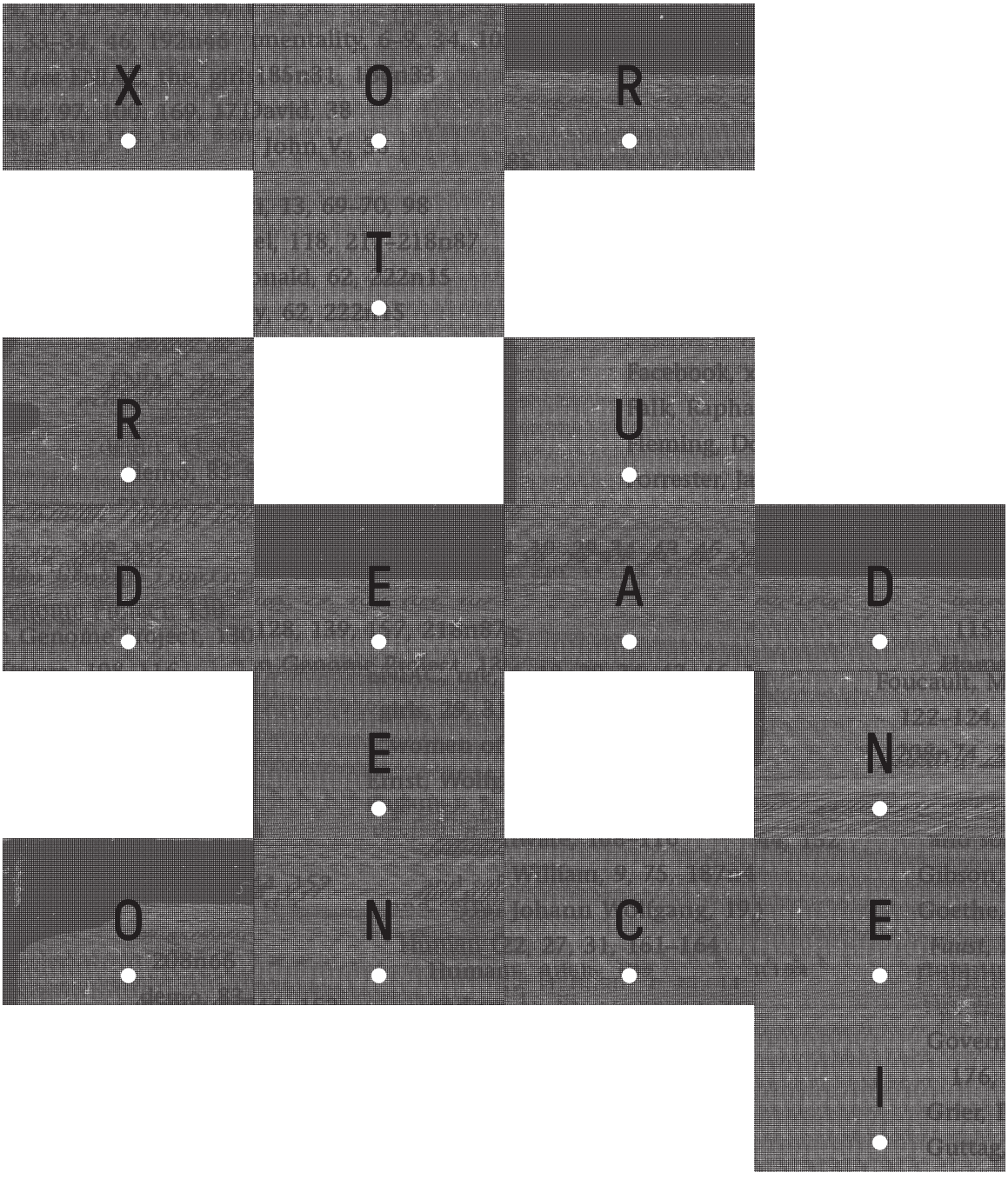

O
R
D
E
R
A
N
D
E
X
E
C
U
T
I
O
N


R U N
D O N E
R D
E X I T
E C O
A


O
R
U
E
A
N
D
E
I
D
E
X
R
N
O
T
C


I N D E X
U R
E
R E A D
O N C T
O


O
O
A
E
D
I
T
N
N
E
U
R
D
E
R
C
X


X
O
R
T
R
U
D
E
A
D
E
N
O
N
C
E
I

How to Do Things With Codes

In his compilation of essays *Language and Symbolic Power*, French sociologist Pierre Bourdieu puts forward a case for an analysis of language that not only involves but centers around the social context of its generation and use. Grounding it into his 'theory of practice' and deploying mainly an economics vocabulary, Bourdieu insists on noticing that language is not only used as a means of communicating verbal messages but also to pursue and advance power in social relations. By juxtaposition, I intend to introduce a reading of the form in which code is implicated in the distribution and operation of power based on the author's sociology of language. To appropriately achieve this, it is first necessary to go over a brief explanation of what such theory of practice entails and how it affects language, so that we may proceed with the discussion that is specific to this thesis.

Bourdieu defines any action or practice as analytically inseparable from its social setting. For this matter, he lays out a methodological interpretation of social contexts as 'fields' (sometimes also referred to as 'markets'), structured relational spaces that are marked by the distribution of resources, *capital*, among different positions.[26] Capital, here, should be understood in a broader sense, as it does not refer only to economic capital (e.g. material wealth), but to other forms such as cultural capital (e.g. knowledge), symbolic capital (e.g. accumulated prestige), political capital (e.g. influence), and so forth. One could talk about, for instance, the literary field, the art field, or the corporate law field, and each will have their set of participants, appreciations and conventions. Within fields, different forms of capital can be converted into one another, as in the quite didactic and schematic example of accumulated wealth allowing for the purchase of

26 John B. Thompson, introduction to *Language and Symbolic Power* (Cambridge, Mass. : Harvard University Press, 1991), 14

high standard education which will then in turn allow for the respectability of the individual, making them eligible for a governmental position.

A field is a locus of continuous struggle regarding the management of the forms of capital it entails, as the individuals holding specific positions in this structured space will attempt to change or retain the distribution of these resources in accordance to their location within the field and what a given position permits. However, for the functioning of a market, there is necessarily a set of common baselines in place for its participants — fundamentally, all must agree upon the "value of what is at stake in the struggles they are waging."[27] So whether a dispute emerges "over the distribution of wealth or over the value of a work of art, [it] always presupposes a fundamental accord or complicity on the part of those who participate"[28] in it. While the terminology of transactions, markets, capital and the formulation of a kind of "game-space" in which social activity is played out might initially suggest a comparison to game-theory (a brainchild of the aforementioned John von Neumann, as a modern discipline) it is important to notice that Bourdieu's account of fields does not by any means ascribe to its subjects the idea of "action as the outcome of conscious calculation,"[29] nor does it take a vacuum-dwelling 'rational agent' as its object. On the contrary, this theory of practice's main contribution is the idea of the *habitus*.

The habitus is a socially constructed set of dispositions within a field of action that inclines agents within such field to operate according to specified conducts. It creates and oversees regularity not by means of conscious coordination but by self-governing attitudes that are inculcated, structured, durable, generative and trans-

27 John B. Thompson, introduction to *Language and Symbolic Power*, 14

28 John B. Thompson, introduction to *Language and Symbolic Power*, 14

29 John B. Thompson, introduction to *Language and Symbolic Power*, 16

posable.[30] That means that such dispositions are the mainly unconscious result of a gradual process of inculcation, in which a group of practices, attitudes and impressions are normalized as a standard, through the reinforcement of their correctness during a person's life. This takes effect within a structured social environment, hence the acquired dispositions will reflect the social conditions in which such process took place. They will also be deep-seated in the individual's being and may last all of one's existence after their acquisition, as they operate "in a way that is pre-conscious and hence not readily amenable to conscious reflection and modification."[31] And, to that effect, the practices that are outcomes of specific dispositions may be generated within different fields other than the ones that engendered them in the first place, but "which concur with the conditions of existence of which the habitus is itself the product."[32] Thus, the habitus is in a circular motion with its environment, as it is shaped by it while also shaping it constantly.

With all that in mind, Bourdieu's gaze upon social practices does indicate that they "never cease to comply with an economic logic"[33] of the pursuit for one's specific interest, but it strays away from the idea that one's interest in necessarily individualistic. Most importantly, it recognizes that the actions used for the advancements of these pursuits, and even the pursuits themselves, are rarely the outcome of a rational calculation, but more generally the results of semi-conscious predispositions that compose the habitus of that individual and the field they are a part of. Moreover, fields are to be regarded in relation to each other, as they have differing degrees of autonomy but are within a complex larger network of habitus that might amount to an entire society. In that way, to understand the

30 John B. Thompson, introduction to *Language and Symbolic Power*, 12

31 John B. Thompson, introduction to *Language and Symbolic Power*, 13

32 John B. Thompson, introduction to *Language and Symbolic Power*, 13

33 Pierre Bourdieu, *The Logic of Practice* (Cambridge: Polity Press, 1990), 122

field of art, one would have to analyze its specificities and the struggles that emerge from it while also understanding what is its proximity to the economy field, politics field, sexual field, etc, in order to grasp what one might exactly mean when speaking of the autonomy of art, what are the overlaps in what forms of capital that are most relevant to each of these fields, what are the kinds of goals that are more usually sought after, and which are the means by which they are; in sum, their differences in habitus.

In what concerns language, the sociologist frames it as a specific form of practice that follows the same constraints as have been laid out in his general theory. Specially in the first section of the aforementioned book, named *The Economy of Linguistic Exchanges*, he develops a strong critique and argues against structural and formalist approaches to linguistics, such as the ones employed by Ferdinand de Saussure and, later, Noam Chomsky. Disputing Saussure's

> inaugural act through which he separates the 'external' elements of linguistics from the 'internal' elements, and, by reserving the title of linguistics to the latter, excludes from it all the investigations which establish a relationship between language and anthropology, the political history of those who speak it, or even the geography of the domain in which it is spoken, because all of these things add nothing to a knowledge of language taken in itself[34]

Bourdieu searches to incorporate such 'externalities', those being the fact that no linguistic interaction would be free of carrying with itself the social structure to which it pertains, by reproducing it and reinforcing it. Thus, the concepts of *langue* and *parole*, brought by Saussure, or competence and performance, their approximation in a Chomskyan vocabulary, are concerned with an abstract and idealized speaker-listener, involved in a homogenous speaker-community and hence removed from any actual

34 Pierre Bourdieu, *Language and Symbolic Power* (Cambridge, Mass. : Harvard University Press, 1991), 33

use that could elucidate anything about the production of meaning. While it may be true that language is a generative process that produces unlimited amounts of grammatically sound utterances, and that this seems to be a uniquely human capacity, the actual competence that is required from humans within their use of language is not that of unlimited utterances, but that of precise and appropriate utterances à propos, within the constraint of a particular social arrangement.

This happens through a 'linguistic market' that informs how the enunciator will consider the enunciated according to its interlocutors and their position in a power scale, and a 'linguistic habitus.' It is also embodied in its practitioners, in what Bourdieu calls a bodily *hexis*. The hexis is the physical incorporation and manifestation of the habitus, in the way one behaves in that social condition. In other circumstances, it can be observed for instance in the ways women are expected to sit or walk differently to men. In the case of language, it is represented by accents, certain uses of the mouth, vocal posture, etc.[35] As in other fields, all of these dispositions also become markers of authority, as they necessarily embed in them the whole social structure of their production. That means that stratified environments will value certain dispositions that compose a specific habitus, like that of the way of speaking of the aristocracy, and devalue others, like that of the working-class.

Linguistic exchanges are then symbolic exchanges, administered in a market that circulates different kinds of symbolic capital. Holders of such capital will usually exert their power, reinforcing a specific habitus by making it official, thus allowing for a larger accumulation of it. One of the instruments for the officialization of a certain set of uses of a language is a grammar. Through that operation, not only specific meanings but also the very structure of enunciation are transformed by their social context.

35 John B. Thompson, introduction to *Language and Symbolic Power*, 17

However, "a discourse can only exist, in the form in which it exists, so long as it is not simply grammatically correct but also, and above all, socially acceptable, i.e. heard, believed and therefore effective."[36]

The field of language provides a group of directives for a tentative mapping of Bourdieu's framework onto the realm of code. In fact, not only are there similarities, but considerable overlap between these fields, as a great deal of the in(put)s and out(put)s of software are manifest as linguistic objects. I will not attempt to produce a full reconstruction of the field of coding. While this seems like a worthy pursuit that could illuminate many aspects of software and society, that would be too deep a sociological endeavor that far exceeds my abilities and objectives. The interest here is, simply put, to address the way in which code is a product of a habitus and its agency upon the world creates and changes dispositions that recompose that same habitus in an orbital, circular motion. In that way, the making of codes, as any other practice, can never escape from containing in itself the social structures in which it was produced, and the dynamics of power from which it originates. The enunciation of power, or lack thereof, is embedded in the code itself and indissociable from it. Software, as is language, is a mechanism for the exchange of symbolic power, and any instance of it is necessarily considered through an understanding of the position from where it came from the position that is receiving it.

As a very clear example, when a national government commissions a coded infrastructure and coded processes to intercept and analyze email messages, it is mobilizing a massive amount of political, pecuniary, technological and other forms of capital to promote a strategy of maintenance of political control of its field. These infrastructures and processes will affect different agents in this field in different ways,

36 Boudieu, *Language and Symbolic Power*, 76

and the relation and reaction to them will have to differ as well — one would have to consider that, if there is a specific bias in the way the national government treats a specific population, the codes that organizes the analytical system will conform to the bias in the very way they are manifest.

Alternatively, one can think of subtler forms of control, that have altogether very clear disciplinary effects on how an individual is to think about and perform certain actions, such as web searching, GPS navigation, and water usage. In all of these cases the software layer is responsible for upholding compliance with a pre-existing form of competent use, or determining what this competent use must be. Competence, in this case as in the case of language, is judged upon concordance to the habitus. It might be the case that this happens through a nearly insurmountable software structure that puts the interlocutor in a stronghold, or through a nudge or suggestion. It is impossible to search on Google without having your search terms indexed and establishing a suggestion pattern for your device (if not for yourself). But even though Waze won't conform to any path that is not strictly coded into its maps, one can still, while driving and making use of the app, make a left turn over a grass patch to reach their destination faster.

Code as a practice, however, has a double agency, in that it also has an operational effect. It is not equivalent to just any linguistic action as it is not just working at the level of production and exchange of meaning, but also for the production of an order that has very real, direct symbolic effects. Software, indeed, is marked by its possibility to "do work in the world."[37] In that sense, Bourdieu's analysis of J.L. Austin's work on speech-acts might provide insight into the way the operational performability of code relates to the world around it. Austin, in his aptly titled book *How to Do Things*

37 Kitchin and Dodge, *Code/Space*, 20

with Words, puts forward a notion of language that has been categorized under the guise of ordinary language philosophy, an attempt at understanding language and the production of meaning through an analysis of its use over anything else. He argues that, within the range of possible linguistic utterances, only a few can be assessed according to their truth-value, that is, whether they express something that is true or false according to an existing described reality.[38] A vast number of sentences are used to affect change in their environment. These are what he names *performative utterances* (later *illocutionary acts*), statements that are in themselves the "doing" of something: "I promise I'll be there" by itself instates a promise, while "I name this ship the *Queen Elizabeth*" is the actualization of the christening of a ship.

These may only be gauged by whether they were "felicitous" or "infelicitous," in Austin's words, meaning their success is based on meeting a certain set of conditions that allow the locution to do what it intends on doing.[39] Bourdieu stresses that these felicity conditions are determined and adjudicated by means of a sociological more than a linguistic appraisal, in that the allowance for the effect of a performative is based on an exercise of authority, a displacement of symbolic capital from someone who is in a specific, legitimate position to do so, and the field's acceptance of it. As he puts it, "only a helpless soldier (or a 'pure' linguist) could imagine that it was possible to give his captain an order."[40] Despite the fact that there might be nothing audibly divergent in a private's and a sergeant's enunciation of the order "Forward, march," it is reasonably expected that only the latter will be felicitous and actually deploy the execution of the call by the appropriate interlocutors (as expected by the aforementioned C3 doctrine). As such,

38 J. L. Austin, *How to Do Things with Words*, The William James Lectures Delivered at Harvard University in 1955 (Cambridge : Harvard University Press, 1975)

39 Austin, *How to Do Things With Words*

40 Bourdieu, *Language and Symbolic Power*, 75

> the logical exercise of separating the act of speech from its conditions of execution shows, through the absurdities that this abstraction engenders, that the performative utterance, as an act of institution, cannot socio-logically exist independently of the institution which gives its *raison d'étre*, and if it were to be produced in spite of everything, it would be socially deprived of sense.[41]

Chun warns us of the perils of conflating order and execution, something that will be addressed in greater length in further sections, and to do so she approximates Judith Butler's reading of Austin to a possible application of it to the analysis of code: to think of the performative utterance as "simply doing what they say posit the speaker as 'the judge or some other representative of the law.' It resuscitates fantasies of sovereign — that is *executive* (hence executable) — structures of power."[42] In contrast, Butler argues, "the subject who 'cites' the performative is temporarily produced as the belated and fictive origin of the performative itself."[43] And Chun ensues, "the programmer/user, in other words, is produced through the act of programming."[44] To think about source code as performative utterances is to make explicit the felicity conditions that afford a program's effectivity, both in terms of machine executability and, in Bourdiesian fashion, the acceptance of its authority over the social reality it affects. To Bourdieu, even the "judge or some other representative of the law," as Butler says, has to adhere to an institutional framework that grants them the right to expect execution from their order, and guarantee that the sentence will be consummated.

> The real source of the magic of performative utterances lies in the mystery of ministry … More precisely, it lies in the social conditions of the *institution* of

41 Bourdieu, *Language and Symbolic Power*, 74

42 Judith Butler, *Excitable Speech: A Politics of the Performative* (New York : Routledge, 1997), quoted in Chun, *Programmed Visions*, location 446

43 Butler, *Excitable Speech: A Politics of the Performative*, 48 quoted in Chun, *Programmed Visions*, location 446

44 Chun, *Programmed Visions*, location 446

> the ministry, which constitutes the legitimate representative as an agent capable of acting on the social world through words, by instituting him as a medium between the group and the social world; and it does that, among other things, by equipping him with the signs and the insignia aimed at underlining the fact that he is not acting in his own behalf and under his own authority.[45]

In our case, then, it is not only the ability of the code to be read by the machine, — i.e. it is compilable if it is a compiled language, it is properly described in its logic operations in the case of assembly, there is enough processing power to run the expected commands, among many other material constraints that deserve a lot of theoretical attention each in their own respects — but the constitution of a ministerial agency to the program and, therefore, the programmer or programming body, that affords a software's authority to be recognized. In that sense, much in the same way that code reflects the same habitus it simultaneously shapes, the authority that renders the performative utterances of software felicitous is mutually constructed. Evidently, as we have seen so far, this mutual relationship conforms to a field and the distribution of capital within it, and both in the case of judicial courts and software infrastructure, there is an accumulation of capital that makes possible the use of violence, symbolic or otherwise, for the maintenance and application of power.

But what that means is that there is only space for algorithmic control and sovereignty, for instance, because there is the construction of a discursive regime, usually the same discursive regime the algorithm will embody in itself, that makes sure that the actions and performatives of the code will be generally accepted. The idea of the "smart city," the 'optimization' of the urban space in which "our streets will be embedded with sensors, our buildings plugged into the internet of things, our commons monitored by cameras and drones, our urban systems recalibrated by real-time data

45 Bourdieu, *Language and Symbolic Power*, 75

on energy, water, climate, transportation, waste and crime,"[46] is only attainable via the inculcation of its underlying Laplacian[47] assumptions that amounting and processing heaps of data will result in a more efficient city, or that this is what an efficient city is. And, more importantly, this gives way to an acceptance of a concentration of power, a letting go of privacy concerns, and the embrace of a technological structure of control by the ones who inhabit the city and are on the other end of the emission of the commands. That means that the way the structure of command employed by a code will actually become an infrastructure of control and how it will act on the world is as much a function of the 'emitter' as it is of the 'receiver.' Or, to put it in better terms, the decoder, or the reader.

46 Shannon Mattern, "Interfacing Urban Intelligence," Places Journal, April 2014. Accessed 17 Mar 2018. https://doi.org/10.22269/140428

47 "We ought to regard the present state of the universe as the effect of its antecedent state and as the cause of the state that is to follow. An intelligence knowing all the forces acting in nature at a given instant, as well as the momentary positions of all things in the universe, would be able to comprehend in one single formula the motions of the largest bodies as well as the lightest atoms in the world, provided that its intellect were sufficiently powerful to subject all data to analysis; to it nothing would be uncertain, the future as well as the past would be present to its eyes. The perfection that the human mind has been able to give to astronomy affords but a feeble outline of such an intelligence." — Pierre-Simon Laplace, *A Philosophical Essay on Probabilities* (1820; reprinted, New York: Dover, 1951) quoted in Chun, *Programmed Visions*, location 1492

non-authoritative program

```
please( )
        i politely request (attention)
        i mustn't, but:
                if you will:
                        explain( )
                otherwise:
                        it's ok(really)
        as long as it is not a bother:
                listen( )
        if this is somewhat similar to that:
                could(i?)
        otherwise:
        counter-propose( )

i'm sorry to ask( )
```

An Amicable Halving of the Matters of Meaning

The history of the reader has been one of success. In a long coming climb, the reader has risen to the top ranks of the meaning-making schemas of Western literature and literary thought. As Donald McKenzie points out, we have generally moved from John Milton's *Areopagidica*'s view of the book (and writing) as "the pretious life-blood of a master-spirit, imbalm'd and treasur'd up on purpose to a life beyond life," a "sacred but expressive form, one whose medium gives transparent access to the essential meaning,"[48] to what some one hundred years later Laurence Sterne wrote in his *Tristram Shandy*:

> No author, who knows the just boundaries of decorum and good-breeding, would presume to think all: The truest respect which you can pay to the Reader's understanding, is to halve this matter amicably, and leave him something to imagine, in his turn, as well as yourself. For my own part, I am eternally paying him compliments of this kind and do all in my power to keep his imagination as busy as my own.[49]

And, notably, Roland Barthes published *The Death of the Author* in 1967, dissociating the creator of the text and their personal, psychological and biographical information from its interpretation. Barthes emphatically proposes that reading and textual criticism should be decoupled of the idea of the Author as a major authority figure over the text. He argues that whereas one can try to pursue an essential meaning in a text by relying on aspects of the author's life, context or psychology, this is ultimately "to

48 John Milton, *Areopagitica: a Speech for the Liberty of Unlicensed Printing, to the Parliament of England* (London : reprinted for R. Blamire, 1792) quoted in D. F. McKenzie, *Bibliography and the Sociology of Texts* (Cambridge, U.K. ; New York : Cambridge University Press, 1999), 31

49 Laurence Sterne, *Tristram Shandy* (London : s.n., 1795) quoted in McKenzie, *Bibliography and the Sociology of Texts*, 35

impose a limit on that text."[50] In place of this method, he suggests an active role of the reader in co-writing the text, which is "eternally written here and now"[51] as meaning derives only from language itself and what the reader makes of it.[52] As he famously and dramatically concludes, "the birth of the reader must be ransomed by the death of the Author."[53]

Michel Foucault, in his turn, in the lecture *What is an Author*?, breaks down the idea of authorship by questioning the definition of a body of work and the decisions that comprise its making. He poses that a writer's oeuvre is composed of their writing. However, which writings should be considered part of the *oeuvre*? Publications, notes, and shopping lists are all written by the so called author but assume different values. He takes this position in a non-explicit conceptual argument with Barthes — while he agrees that this category of the Author is an overblown historical construct, he goes further and says that by preserving the notion of work and the suggestion of unity provided by it, Barthes's account of the Death of the Author is actually perpetuating authorship.[54] Both inextricably link *authorship* and *authority*, while simultaneously stripping the author of power, and conceding it to the reader.

Reading, then, becomes a rich site for it carries in itself the aesthetic-political action of making meaning, and with that of validating or not what is being authoritatively put forth. In *Bibliography and the Sociology of Texts*, McKenzie not only points to the relationship between the writer and the reader in the process of signification (to

50 Roland Barthes, "The Death of the Author," *Image, Music, Text.* ed. Stephen Heath (London: Fontana, 1984) p. 143-148

51 Barthes, "The Death of the Author," p. 143-148

52 Going in a somewhat opposite direction of Bourdieu, by leaning in an obviously more structurally-inclined interpretation.

53 Barthes, "The Death of the Author," p. 148

54 Michel Foucault,"What is an Author?" *Aesthetics, Method, and Epistemology*. ed. James D. Faubion (New York, NY: The New Press, 1998), 206-222

which he rightfully adds the editor, the printer, the book/object, and others, but that is a matter for a different discussion) but also promotes an expansion of the notion of "text" itself, as not only the written records of language but as the multiplicity of forms that have a "textual function," of recording and embodying meaning "which is subject to bibliographical control, interpretation, and historical analysis."[55]

"It is more convenient," he says, "to think simply in terms of homologies, of correspondent structures, suggesting that, whatever our own special field — be it books, maps, prints, oral traditions, theatre, films, television, or computer-stored databases — we note certain common concerns."[56] This conception of the textual artifact permits a "readerly" engagement with a wider range of phenomena, to make explicit the interlocutor's agency in their configuration and understanding. From that, we can analyze software within a textual framework, and analyze what are the positions in which, how and by whom it is being read, and formulate a proposal for how art practice can suggest forms of reading that shift the current power and control structures of the code.

In the past few years, much attention has been given to developing strategies for reading code, usually on the basis of source code, by employing well established literary and media theory technics. The emergent field of Critical Code Studies relies on the idea that "code, especially mid- to high-level languages, exists not solely for computers, which could operate on machine language (essentially, representations of electronic signals), but for programmers as well."[57] From this rationale, Mark C. Marino proposes "that we no longer speak of the code as a text in metaphorical terms, but that we begin to analyze and explicate code as a text, as a sign system with its

55 McKenzie, *Bibliography and the Sociology of Texts*, 39

56 McKenzie, *Bibliography and the Sociology of Texts*, 39

57 Mark C. Marino, "Critical Code Studies," *Electronic Book Review*, 12/04/2006, http://www.electronicbookreview.com/thread/electropoetics/codology

own rhetoric, as verbal communication that possesses significance in excess of its functional utility."[58] That is, one must apply "critical hermeneutics to the interpretation of computer code, program architecture, and documentation within a socio-historical context,"[59] as "meaning grows out of the functioning of the code but is not limited to the literal processes the code enacts."[60] One example of the application of socio-historical inclined hermeneutics applied to code is *10 PRINT CHR$(205.5+RND(1)); : GOTO 10*, by Nick Montfort, Patsy Baudoin, John Bell, Ian Bogost, Jeremy Douglass, Mark C. Marino, Michael Mateas, Casey Reas, Mark Sample and Noah Vawter. This collectively authored book takes as its subject the one-line BASIC program that lends itself as the title. As the authors state in the opening of the book

> The way in which code connects to culture, affecting it and being influenced by it, can be traced by examining the specifics of programs by reading the code itself attentively.
>
> Like a diary from the forgotten past, computer code is embedded with stories of a program's making, its purpose, its assumptions, and more. Every symbol within a program can help to illuminate these stories and open historical and critical lines of inquiry.[61]

By dissecting this readily available and extremely simple piece of software, published in the 1982 *Commodore 64 User's Guide*, an automatic generator of maze-patterns based on extended ASCII characters, they are able to extrapolate a series of arguments and understandings about software culture and cultural history, such as the circulation and modification of programs, platform design, emulation, regularity, and randomness. "Code," according to them, "should be valued as text

58 Marino, "Critical Code Studies," Making the Code the Text

59 Marino, "Critical Code Studies," Critical Code Studies

60 Marino, "Critical Code Studies," Critical Code Studies

61 Nick Montfort et al., *10 Print Chr$(205.5+rnd(1));:goto 10*, Software Studies (Cambridge, Massachusetts ; London, England : MIT Press, 2013), 3

with machine and human meanings, something produced and operating within culture."[62] To that, I would add that it also produces and operates culture, as discussed previously. While this close-reading approach has much to offer in the study of the meanings and cultural-historical positions embedded and produced in and by the code, other forms of reading might enact other forms of engagement with software.

In an analysis of mass media (and more specifically television) in the 1970s, Stuart Hall establishes his Encoding/Decoding model of communication. He refers to how media messages are "produced, disseminated, and interpreted,"[63] with emphasis on television, and claims that a message is produced in a coded system of meanings and later *decoded* by its audience, in an active process that involves a reliance in one's social context. He also considers such a process to be divided in four stages, Production, Circulation, Use, and Reproduction, each with its own encoding procedures that carry on to the next.

Beyond approximating theories pushed in the literary and media theory world, as we have seen, to the language of signals and code, Hall speaks of three positions from which the decoding end can occur: *dominant-hegemonic*, *oppositional*, or *negotiated*. Each represents a stance of the reader/receiver/decoder in regards to the control structure carried by the code employed. A dominant-hegemonic position happens when the reader takes the media text, in his example a newscast but easily transposable to a news feed, "full and straight, and decodes the message in terms of the reference code in which it has been encoded,"[64] that is, accepts all of the authority that comes with that utterance. An oppositional reading is one that "decode[s] the message in a *globally* contrary way,"[65] despite a clear literal and connotative understanding of the message's discourse. And finally, the position from which

62 Montfort et al., *10 Print Chr$(205.5+rnd(1));:goto 10*, 8

63 Hall, "Encoding/Decoding," 128

64 Hall, "Encoding/Decoding," 136

65 Hall, "Encoding/Decoding," 137-138, emphasis his

most decodings of television content occur is a negotiated one, where there is a perfectly adequate understanding of the dominant definitions but read through the hegemonic assertions, acknowledging their legitimacy while simultaneously setting "its own ground rules."[66]

The establishment of new ways of reading software has to go through the possibility of creating larger windows for negotiation and opposition. As we will explore in further chapters, software, as present and heavy-handed as it is in the articulation and control of everyday life, is largely occluded, and its actions are hard to locate within the different fields it is inserted in, and even in the field of code practices itself. This facilitates the maintenance of a large body of individuals that take on a dominant-hegemonic position for the decoding of their relationship to the effects of software, in turn lending authority to its prescriptions. Artistic readings can offer modes for the articulation of negotiated meanings that emerge from discrepant engagements with code, that do not necessarily fall into the category of education or code-literacy, but may nonetheless provide emancipatory knowledge and praxis.

66 Hall, "Encoding/Decoding," 137

I
I m
I mu
I mus
I must
I must
I must n
I must no
I must not
I must not
I must not s
I must not sa
I must not saw
I must not saw
I must not saw o
I must not saw of
I must not saw off
I must not saw off
I must not saw off t
I must not saw off th
I must not saw off the
I must not saw off the
I must not saw off the b
I must not saw off the br
I must not saw off the bra
I must not saw off the bran
I must not saw off the branc
I must not saw off the branch
I must not saw off the branch
I must not saw off the branch o
I must not saw off the branch on
I must not saw off the branch on
I must not saw off the branch on w
I must not saw off the branch on wh
I must not saw off the branch on whi
I must not saw off the branch on whic
I must not saw off the branch on which
I must not saw off the branch on which
I must not saw off the branch on which i
I must not saw off the branch on which i
I must not saw off the branch on which i a
I must not saw off the branch on which i am
I must not saw off the branch on which i am
I must not saw off the branch on which i am s
I must not saw off the branch on which i am si
I must not saw off the branch on which i am sit
I must not saw off the branch on which i am sitt
I must not saw off the branch on which i am sitti
I must not saw off the branch on which i am sittin
I must not saw off the branch on which i am sitting
I
I m
I mu
I mus
I must
I must
I must n
I must no
I must not
I must not
I must not s
I must not sa
I must not saw
I must not saw
I must not saw o
I must not saw of
I must not saw off
I must not saw off
I must not saw off t
I must not saw off th
I must not saw off the
I must not saw off the
I must not saw off the b
I must not saw off the br
I must not saw off the bra
I must not saw off the bran
I must not saw off the branc
I must not saw off the branch
I must not saw off the branch
I must not saw off the branch o
I must not saw off the branch on
I must not saw off the branch on
I must not saw off the branch on w
I must not saw off the branch on wh
I must not saw off the branch on whi
I must not saw off the branch on whic
I must not saw off the branch on which
I must not saw off the branch on which
I must not saw off the branch on which i
I must not saw off the branch on which i
I must not saw off the branch on which i a
I must not saw off the branch on which i am
I must not saw off the branch on which i am
I must not saw off the branch on which i am s
I must not saw off the branch on which i am si
I must not saw off the branch on which i am sit
I must not saw off the branch on which i am sitt
I must not saw off the branch on which i am sitti
I must not saw off the branch on which i am sittin
I must not saw off the branch on which i am sitting
I
I m
I mu
I mus
I must
I must
I must n
I must no
I must not
I must not
I must not s
I must not sa
I must not saw
I must not saw
I must not saw o
I must not saw of
I must not saw off
I must not saw off
I must not saw off t
I must not saw off th
I must not saw off the
I must not saw off the
I must not saw off the b
I must not saw off the br
I must not saw off the bra
I must not saw off the bran

```
I must not saw off the branc
I must not saw off the branch
I must not saw off the branch
I must not saw off the branch o
I must not saw off the branch on
I must not saw off the branch on
I must not saw off the branch on w
I must not saw off the branch on wh
I must not saw off the branch on whi
I must not saw off the branch on whic
I must not saw off the branch on which
I must not saw off the branch on which
I must not saw off the branch on which i
I must not saw off the branch on which i
I must not saw off the branch on which i a
I must not saw off the branch on which i am
I must not saw off the branch on which i am
I must not saw off the branch on which i am s
I must not saw off the branch on which i am si
I must not saw off the branch on which i am sit
I must not saw off the branch on which i am sitt
I must not saw off the branch on which i am sitti
I must not saw off the branch on which i am sittin
I must not saw off the branch on which i am sitting
d
de
def
def
def b
def br
def bra
def bran
def branc
def branch
def branch(
def branch()
def branch():
def branch():
def branch(): s
def branch(): sa
def branch(): saw
def branch(): saw
def branch(): saw =
def branch(): saw =
def branch(): saw = “
def branch(): saw = “i
def branch(): saw = “i
def branch(): saw = “i m
def branch(): saw = “i mu
def branch(): saw = “i mus
def branch(): saw = “i must
def branch(): saw = “i must
def branch(): saw = “i must n
def branch(): saw = “i must no
def branch(): saw = “i must not
def branch(): saw = “i must not
def branch(): saw = “i must not s
def branch(): saw = “i must not sa
def branch(): saw = “i must not saw
def branch(): saw = “i must not saw
def branch(): saw = “i must not saw o
def branch(): saw = “i must not saw of
def branch(): saw = “i must not saw off
def branch(): saw = “i must not saw off
def branch(): saw = “i must not saw off t
def branch(): saw = “i must not saw off th
def branch(): saw = “i must not saw off the
def branch(): saw = “i must not saw off the
def branch(): saw = “i must not saw off the b
def branch(): saw = “i must not saw off the br
def branch(): saw = “i must not saw off the bra
def branch(): saw = “i must not saw off the bran
def branch(): saw = “i must not saw off the branc
def branch(): saw = “i must not saw off the branch
o
on
on
on w
on wh
on whi
on whic
on which
on which
on which i
on which i
on which i a
on which i am
on which i am
on which i am s
on which i am si
on which i am sit
on which i am sitt
on which i am sitti
on which i am sittin
on which i am sitting
on which i am sitting
on which i am sitting w
on which i am sitting wh
on which i am sitting whi
on which i am sitting whil
on which i am sitting while
on which i am sitting while
on which i am sitting while T
on which i am sitting while Tr
on which i am sitting while Tru
on which i am sitting while True
on which i am sitting while True:
on which i am sitting while True:
on which i am sitting while True: f
on which i am sitting while True: fo
on which i am sitting while True: for
on which i am sitting while True: for
on which i am sitting while True: for i
on which i am sitting while True: for i
on which i am sitting while True: for i i
on which i am sitting while True: for i in
on which i am sitting while True: for i in
on which i am sitting while True: for i in r
on which i am sitting while True: for i in ra
on which i am sitting while True: for i in ran
on which i am sitting while True: for i in rang
on which i am sitting while True: for i in range
on which i am sitting while True: for i in range(
le
len
len(
len(s
```

len(sa
len(saw
len(saw)
len(saw))
len(saw)):
len(saw)):
len(saw)): p
len(saw)): pr
len(saw)): pri
len(saw)): prin
len(saw)): print
len(saw)): print
len(saw)): print s
len(saw)): print sa
len(saw)): print saw
len(saw)): print saw[
len(saw)): print saw[0
len(saw)): print saw[0:
len(saw)): print saw[0:i
len(saw)): print saw[0:i+
len(saw)): print saw[0:i+1
len(saw)): print saw[0:i+1]
len(saw)): print saw[0:i+1]
len(saw)): print saw[0:i+1] b
len(saw)): print saw[0:i+1] br
len(saw)): print saw[0:i+1] bra
len(saw)): print saw[0:i+1] bran
len(saw)): print saw[0:i+1] branc
len(saw)): print saw[0:i+1] branch
len(saw)): print saw[0:i+1] branch(
len(saw)): print saw[0:i+1] branch()
l
l m
l mu
l mus
l must
l must
l must n
l must no
l must not
l must not
l must not s
l must not sa
l must not saw
l must not saw
l must not saw o
l must not saw of
l must not saw off
l must not saw off
l must not saw off t
l must not saw off th
l must not saw off the
l must not saw off the
l must not saw off the b
l must not saw off the br
l must not saw off the bra
l must not saw off the bran
l must not saw off the branc
l must not saw off the branch
l must not saw off the branch
l must not saw off the branch o
l must not saw off the branch on
l must not saw off the branch on
l must not saw off the branch on w
l must not saw off the branch on wh
l must not saw off the branch on whi
l must not saw off the branch on whic
l must not saw off the branch on which
l must not saw off the branch on which
l must not saw off the branch on which i
l must not saw off the branch on which i
l must not saw off the branch on which i a
l must not saw off the branch on which i am
l must not saw off the branch on which i am
l must not saw off the branch on which i am s
l must not saw off the branch on which i am si
l must not saw off the branch on which i am sit
l must not saw off the branch on which i am sitt
l must not saw off the branch on which i am sitti
l must not saw off the branch on which i am sittin
l must not saw off the branch on which i am sitting
l
l m
l mu
l mus
l must
l must
l must n
l must no
l must not
l must not
l must not s
l must not sa
l must not saw
l must not saw
l must not saw o
l must not saw of
l must not saw off
l must not saw off
l must not saw off t
l must not saw off th
l must not saw off the
l must not saw off the
l must not saw off the b
l must not saw off the br
l must not saw off the bra
l must not saw off the bran
l must not saw off the branc
l must not saw off the branch
l must not saw off the branch
l must not saw off the branch o
l must not saw off the branch on
l must not saw off the branch on
l must not saw off the branch on w
l must not saw off the branch on wh
l must not saw off the branch on whi
l must not saw off the branch on whic
l must not saw off the branch on which
l must not saw off the branch on which
l must not saw off the branch on which i
l must not saw off the branch on which i
l must not saw off the branch on which i a
l must not saw off the branch on which i am
l must not saw off the branch on which i am
l must not saw off the branch on which i am s
l must not saw off the branch on which i am si
l must not saw off the branch on which i am sit

Occlusions and In/Visibilities

I.

'Code' can be a confusing term for it has a multiplicity of common-sense meanings. While so far it has mostly been used in this text as a synonym for the information layer that determines the behavior of a computational procedure — more or less interchangeably with 'software' — and only slightly and strategically approximated to other definitions (e.g. legal regimen of a determinate social context, a system of formal specification of information) this chapter will start by addressing a more specific notion of the same term before returning to our original conception. Namely, this section will seek to discuss secret codes, forms of encoding that are purposefully designed to conceal the intended meaning of a message. By doing so, I aim to speculate on how one can read things that are hidden, as are most of the control structures of (and proceeding from) the programs in our everyday life, while simultaneously bringing to the fore encoding/decoding strategies that create parallel signification architectures which negotiate with the given power dynamics and create spaces of resistance.

One such strategy is the development of what M.A.K. Halliday have named "anti-languages." These forms of speech are constructions of "anti-societies," or marginalized groups within a larger society, to communicate between themselves in the open while still not being understood by those outside their linguistic and social domain.[67] In this way, they are able to reconfigure and maintain their reality through concealment and secrecy in their use of heavily encoded lexicons. In a similar analysis of language than that of Bourdieu, Halliday is pointing out to the use of language primarily as a tool for social action, part of a socio-political struggle within a particular context. Howev-

67 Halliday, M.A.K. "Anti-Languages", *American Anthropologist, Volume 78* (Sep. 1976). pp. 570

er, this approach inverts the gaze to look at it through its defensive competencies, a means for the holders of the least amount of capital in a field to assert their ways of being, facing the imperatives of the society at large with which they might not have allegiance in one or more ways.

This is the case with Polari, for instance, which, as an anti-language used mostly by the British gay community while homosexuality was still criminalized in the country during the 20th Century, allowed BBC Radio host Kenneth Williams to air a comment about having erections on public toilets that would only be comprehended by fellows in his "anti-societal" group, simply by mentioning "a miracle of dexterity at the cottage upright."[68] As exemplified by this utterance, anti-languages typically operate with the principle of partial relexicalization of the language in which they are situated. That is, it is based on the exchange of certain words for other, new ones, that may or may not be familiar to the speakers of the "original" language, which amounts to "same grammar, different vocabulary; but different vocabulary only in certain areas, typically those that are central to the activities of the subculture and that set it off most sharply from the established society."[69]

What that entails is that a member of the society that has found its opposition in an anti-society — a speaker of the language that serves as the contentious basis for the anti-language — may be able to extract some meaning from the anti-linguistic utterance. The sentence pronounced by the radio host was indeed transmitted to thousands of homes in England, and those who have not grasped its meaning within the boundaries of Polari have still read and created a semantic interpretation of this string of words. This may rely on a very active participation of the reader, by activating their

68 David Robson. *The Secret "Anti-Languages" You Are Not Suposed to Know.* accessed 27 February 2017, http://www.bbc.com/future/story/20160211-the-secret-anti-languages-youre-not-supposed-to-know

69 Halliday, "Anti-Languages," 571

own decoding and producing meaning from what seems to be disparate or loosely connected parts in a *post hoc ergo propter hoc* fashion. And while it is evident that there is a difference between the transmission of a secret or an ambiguous message, one can take each of these positions based exclusively on the nature of the mutually established encoding/decoding system.

However, ambiguity has a life of its own. Following an information-theoretic approach, the amount of ambiguity and the amount of information in a message are one and the same. To put it in an extremely simplified way, if one can always tell what is the next bit of information following a certain state in a message, there is no meaning being produced, only redundancy. Conversely, if there is no redundancy surrounding the bit in a way that could guide a disambiguation process, the multiplicity of the message can break itself. The way these ambiguities and redundancies are negotiated in order to make sense are part of the code applied upon them, and the actions each code takes to articulate these two axes will have a major outcome in the decoded message.

With that in mind, questions arise from the interpretation of anti-languages. Because they are generally coded on top of an existing system, using a similar syntax, morphology and even vocabulary to one or more base languages, one might be able to produce attempts at decoding an anti-linguistic message with reasonable rates of success, without even having knowledge of the existence of an occluded sign. This may result in a secondary message, not less valid in terms of circulation than the first one, with its own offshoots of meaning. To use the previous example, “a miracle of dexterity at the cottage upright” could very well be relative to an incredibly well built farm house.

What is not as clear, however, is whether this secondary message is indelibly connected to the primary one — could it exist in absolute autonomy, once it is decoded as something else? Or does it necessarily carry, albeit silently, a trace of its “origi-

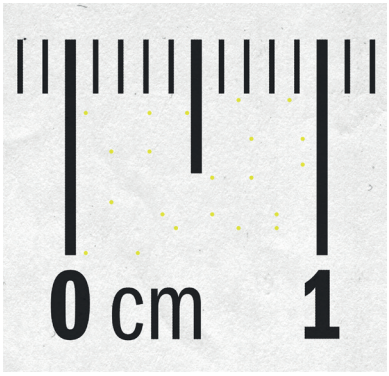
nal" meaning? This is, in part, a function of the receiver. The level of engagement one has with the fuller context of the message will determine how much of other readings might creep into the semantic background of the message. What are the possibilities of a surface text to acquire an existence of its own, seemingly disconnected from its origin, even while it still carries the occluded text within itself? This also concerns the way in which objects, practices and codes do things that shape and preserve reality, and how concealed forms may still perform these operations just the same while being apparently invisible. Steganography, the practice of concealing a message within another message, raises many of the same questions, and sometimes in much more explicit ways. This millennia old method, first described with this name by the German abbott Johannes Trithemius in 1499, is based on the existence of two layers of text: a covertext, something immediately visible, that has its own set of meanings, usually unrelated to the hidden text, which is somehow incorporated invisibly to the same file or message. Trithemius' *Steganographia* is in itself an example of the inner workings of this technique: written as a treatise on magic and demonology, allegedly about using spirits to convey messages at long distances, the book secretly describes steganographic and cryptographic processes throughout, but is only entirely legible for those who have a decryption key and can thus access the second layer of occluded text.[70] Before its decryption, though, and even after, for those who are unfamiliar with the nature of this double encoding, there was the constitution of an understanding of this book solely as a volume on the occult, and that meaning took prevalence over the "original" one.

Steganography differs from cryptography in that the latter enciphers the secret

70 Jim Reeds, "Solved: The Ciphers In Book III Of Trithemius's Steganographia," *Cryptologia*, 22:4, 293, https://doi.org/10.1080/0161-119891886948

content, but maintains visibility to the existence of a hidden message; the confrontation with the cryptotext is one of immediate opposition, a desire for its breakage and an acknowledgement of the meaninglessness of the extant object. The steganotext, however, has the ability to infiltrate, for it has no indication for the unintended or inattentive receiver that it is not a full message in itself. It not only hides a message but the sheer fact that there is anything hidden. And this gentler intrusion allows for the resocialization of the message, an autonomous circulation of the two intertwined textual layers. Far from an esoteric gimmick, this is an usual event — from transit of politically sensitive information to hacking for personal gain, and other things in between, steganographic texts abound in the physical and digital worlds. In fact, since 2005 the Electronic Frontier Foundation has been researching and advocating against the use of steganographic timestamps and serial information on consumer-level laser printers, that get printed without the knowledge or consent of the user onto the paper document, as barely visible markings. According to the EFF,

> Some of the documents that we previously received through FOIA [Freedom of Information Act] suggested that all major manufacturers of color laser printers entered a secret agreement with governments to ensure that the output of those printers is forensically traceable.[71]



In any case, both of these encoding/decoding methods, anti-languages and steganography, propose a mode of occlusion based on apparent ambiguity. Ambiguity, then, becomes a duplicitous strategy for dealing with our problems in reading. To the concern of reading what is not there, or what cannot be perceived as im-

71 Eletronic Frontier Foundation, "List of Printers Which Do or Do Not Display Tracking Dots," https://www.eff.org/pages/list-printers-which-do-or-do-not-display-tracking-dots

mediately visible, it becomes obvious that the only way to do so is by incorporating the ambiguous nature of the task and sticking to the ambiguity of the generated meanings. By refusing redundancy and disambiguation, thus amplifying the grey areas between prescriptive intention and autonomous reception, one takes a liberating position that permits a larger flexibility in political-aesthetic meaning-making. Conversely, on the problem of being read — becoming an object under constant data assessment within an infrastructure of control — producing ambiguity is a way of making oneself indiscernible, so unattainable.

II.

"Software is extremely difficult to comprehend. Who really knows what lurks behind our smiling interfaces, behind the objects we click and manipulate?"[72] asks Wendy Chun.

> Its combination of what can be seen and not seen, can be known and not known — its separation of interface from algorithm, of software from hardware — makes it a powerful metaphor for everything we believe is invisible yet generates visible effects, from genetics to the invisible hand of the market, from ideology to culture.[73]

How is a user supposed to understand the inner workings of a software object? This question is much more than a matter of knowing how to program — while the vast majority of people in the world does not comprehend the operation of code for not knowing *how to code*, even experienced programmers are extremely limited in their access and understanding of many aspects of software. "Regardless of myths of all-powerful

72 Chun, *Programmed Visions,* location 104

73 Chun, *Programmed Visions*, location 110

hackers who 'speak the language of computers as one does a mother tongue,'"[74] in the most cases code and its primary operations are separated from us through a bulkhead.[75] Proprietary software is distributed as binary executable files, sealing off analysis of the source. Much of the software that in one way or another affects an individual's life is not being run on the same space occupied by the person. It might be running in a server miles away, as in a search query, or it might be running locally where the main constituency of its function takes place but the affected individual is in one of its "terminations," which would be the case of something as simple as turning on a water tap.

All software architecture needs to provide points of access, nonetheless, and it is via these *interfaces* that we may be able to read them. These may vary, from the most obvious physical ones, such as screens, keyboards, mouses and touchscreens that host our daily interaction with Graphic User Interfaces, to the inconspicuous trails left by coded processes: "official form letters, statements, bills, receipts, printouts, licenses … bank and credit cards, library cards, transportation cards"[76] and so on. No matter how efficiently the interruption of visibility of code is, it has to be betrayed by a link with its subject of operation. These slight denunciations are akin to the relexicalized nature of anti-languages, or the familiar form of the steganotext; they provide a ledge to hold onto, a base unit to establish a decoding system that may or may not coincide with the one used for encoding. But the fact is that software is occluded on a regular basis, and the ones which are explicitly or actively engaged in strategies of control are usually designed with emphasis on that feature — out of sight, out of mind.

74 Alexander Galloway, *Protocol: How Power Exists after Decentralization*, (Cambridge, Mass.: MIT Press, 2004), 164 quoted in Chun, *Programmed Visions*, location 346

75 Not less importantly, even the notion that acquiring access to source code is enough to understand its effect is inappropriate, but this will be further discussed in the next chapter.

76 Kitchin and Dodge, *Code/Space*, 7

Tenen makes a clear point of this by stating that "code is not usually meant to be decoded by those it acts upon. Recipients of codified control are spared the friction of signification, remaining instead in the state of *asemiosis* and therefore nescience."[77] To him, "programming at its essence is a phatic activity," recalling Roman Jakobson's definition of the phatic function of language. "Code shapes and commands. At the same time, it conjures fantastical metaphors to occlude the structure of shaping and commanding."[78]

At the level of personal computing, this is made clear in the transition between what was known as literal computing into what Ben Shneiderman termed the 'direct manipulation' model. The elements that compose a current computer interface, deriving from this model, are folders, buttons, garbage bins, pages, etc. and it allows for us to perform simulations of actions we are used to perform when engaging with those objects. These metaphors propose a method of operating by means of intelligible visuality that is affected by an invisible set of parameters, and in turn affects the behavior of such parameters through manipulation of the visible component. This configures a "complex process of transfiguration between the visible sign and the sign at the site of its inscription."[79] As Chun points out in the quote used for the opening of this section, software not only uses metaphor, but it serves as a metaphor itself. And in this metaphoric representation of all that is visibly invisible, "software as metaphor combines what we can only vaguely understand with something equally vague. It is not simply, then, that one part of the metaphor is 'hidden,' but rather that both parts — tenor and vehicle[80] — are invisibly visible."[81] And it is from this perspective that code is able to

77 Tenen, *Plain Text*, location 518, emphasis in the original

78 Tenen, *Plain Text*, location 543

79 Tenen, *Plain Text*, location 543

80 'Tenor' and 'vehicle' are terms regularly used in theories of metaphor to signify the idea represented and the idea expressly used, respectively — Chun, Programmed Visions, location 893

81 Chun, *Programmed Visions*, location 896

stand for the ambiguous relationships between all we can and cannot perceive.

It is software's "ambiguous thingliness" that foregrounds its apparent ability to understand "how power operates in a world marked by complexity and ambiguity, in a world filled with things we cannot fully understand, even though these things are marked by, and driven by, rules that should be understandable."[82] That, according to the author, causes a difficulty in distinguishing the personal and the impersonal, empowerment and surveillance. By presenting in its interfaces what is supposed to be a map to engage a complex world "driven by the invisible laws of late capitalism," code is also "induc[ing] the user to map constantly so that the user in turn can be mapped."[83] Simultaneously, however, Chun calls for making our interfaces "more productively spectral — by reworking rather then simply shunning the usual modes of 'user empowerment.'"[84]

It is within that framework that artistic readings are able to establish other modes of relation with the in/visibilities of software. However obscured code can be, its operations continue without regards to immediate legibility. There is a calculated point of interaction and distribution that is ostensibly visible, but offers not more than an anti-linguistic utterance does to its eclipsed message. So, in that sense, one can only expect to encounter the same relationship with reading the occluded text, through and beyond the interface, by an embrace of ambiguity, or 'spectrality.' If one has no way of decoding by using the same encoding structure, one is to embrace ambiguity and confront software with the most convenient interpretation for each particular reading. By way of doing so, the ambiguity contained in code is not used for the constitution of software as metaphor for all things on the edge of visibility, but precisely undoes it

82 Chun, *Programmed Visions*, location 899

83 Chun, *Programmed Visions*, location 910

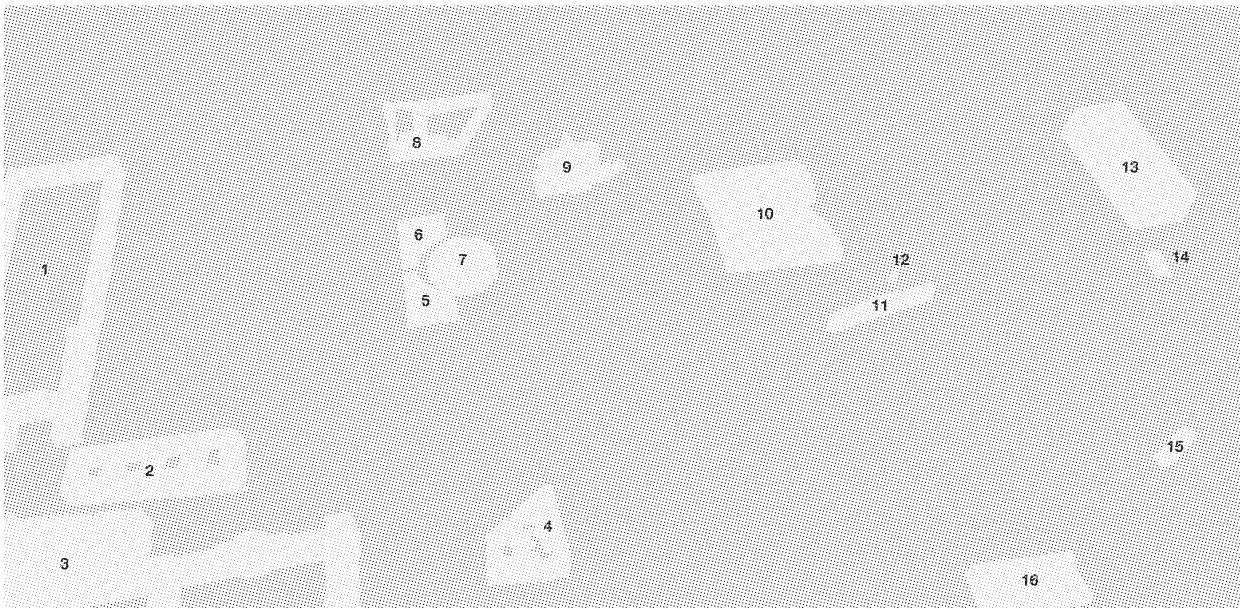
84 Chun, *Programmed Visions*, location 921

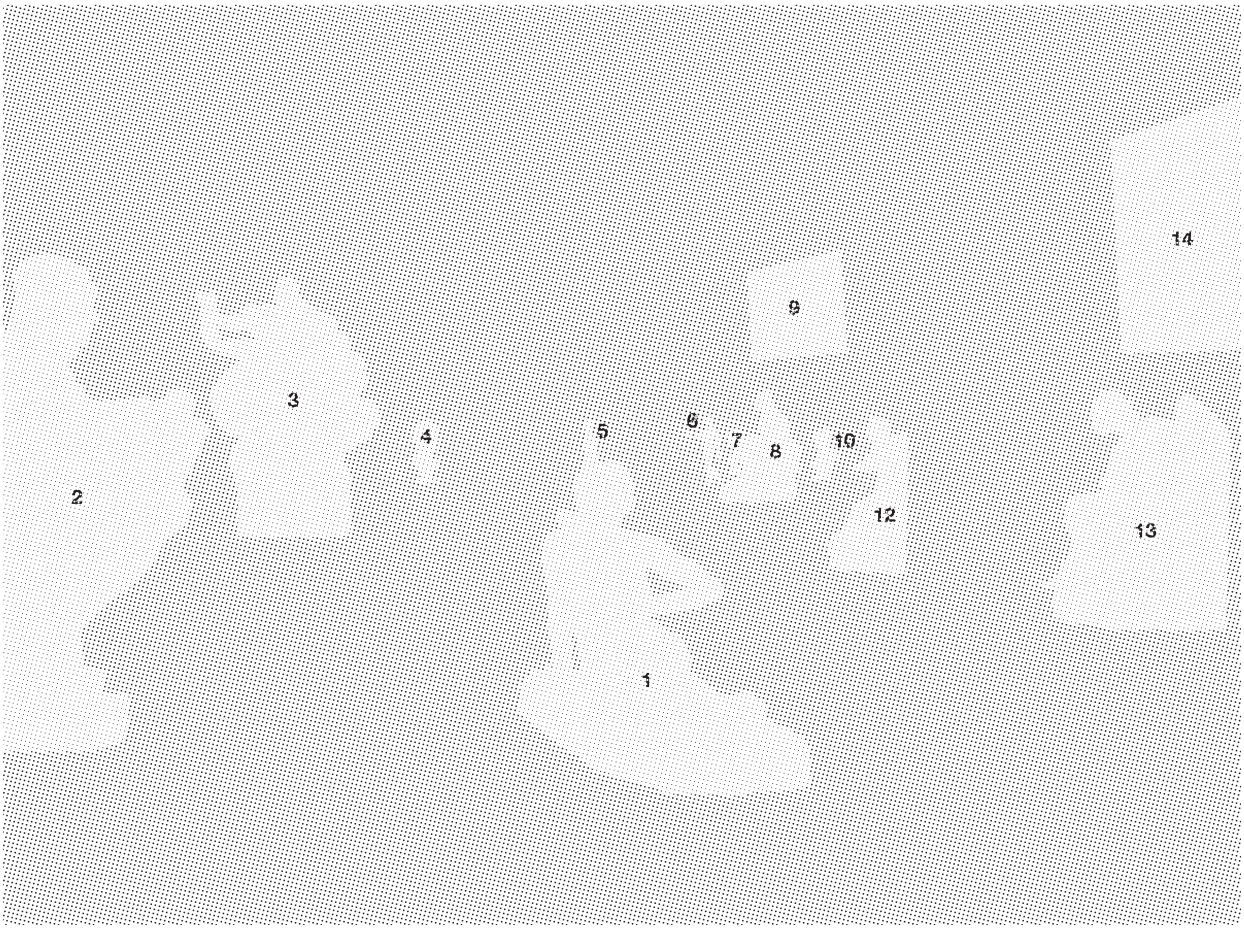
— to expect a clearly produced meaning is to expect a simple map of power, and to comply with this mapping is to decode the prescriptions of the code hegemonically. In Édouard Glissant's words,

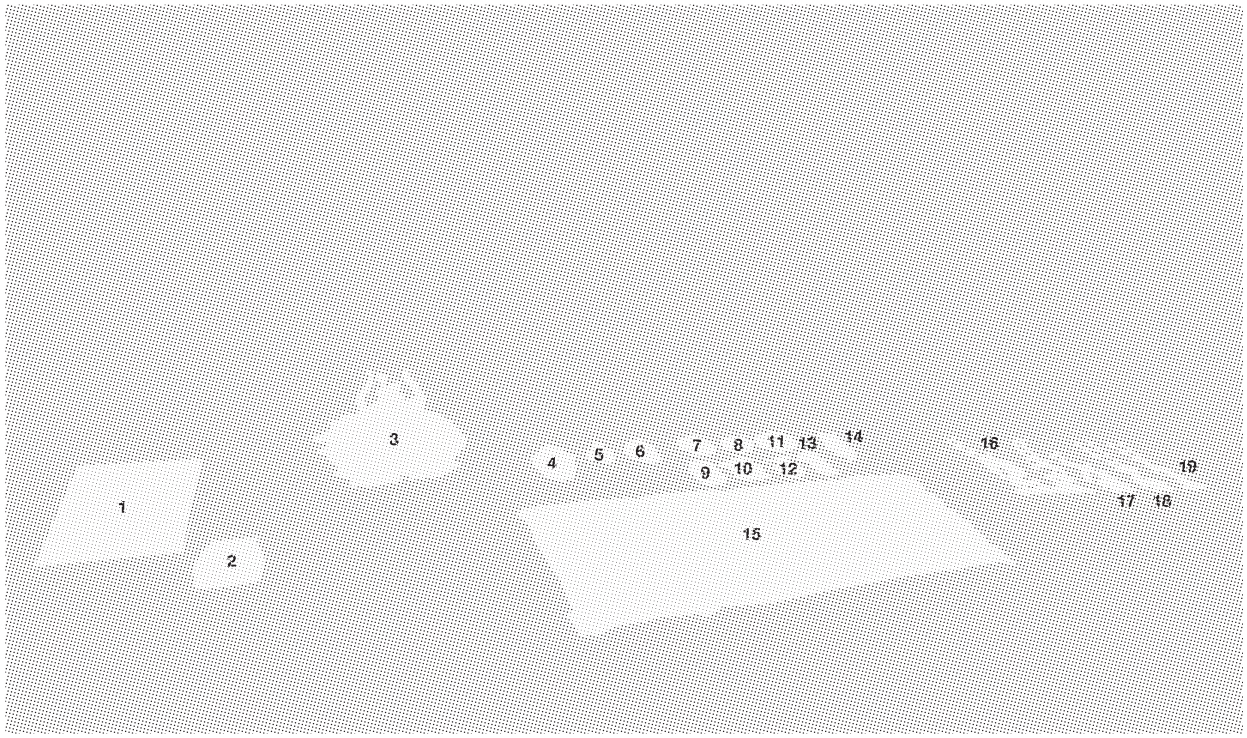
> the written text opposes anything that might lead a reader to formulate the author's intention differently. At the same time he can only guess at the shape of this intention. The reader goes, or rather tries to go back, from the produced opacity to the transparency that he read into it.[85]

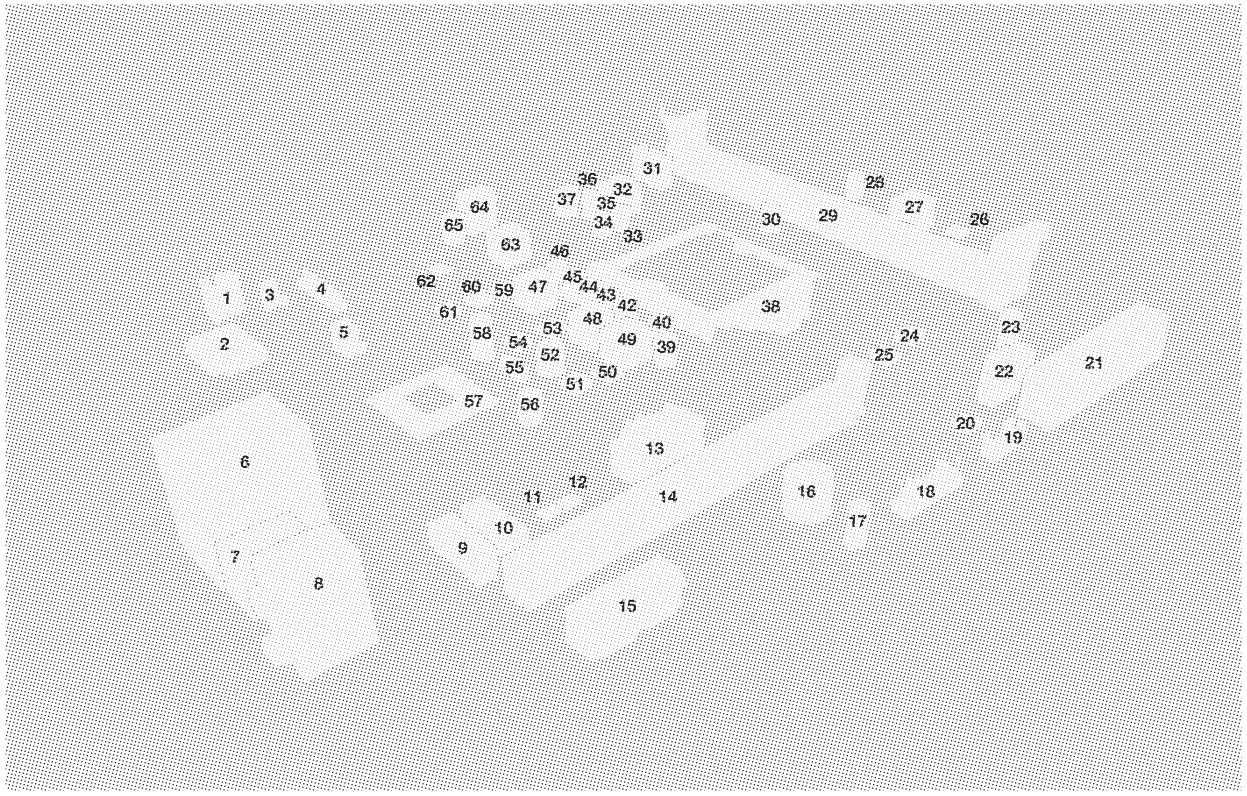
85 Édouard Glissant, *Poetics of Relation* (Ann Arbor : University of Michigan Press, 1997), 115

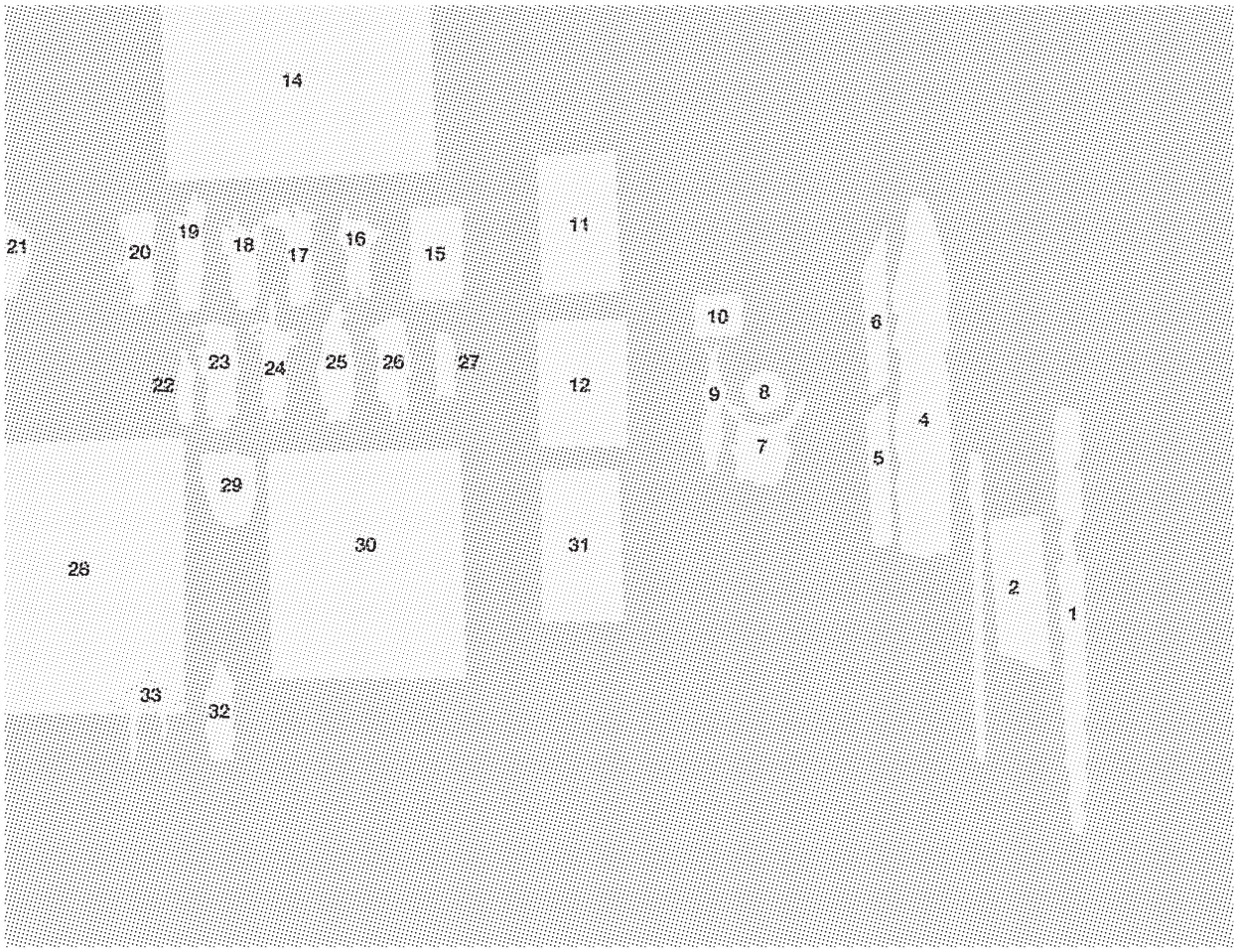
Catalogue of Unidentified Parts

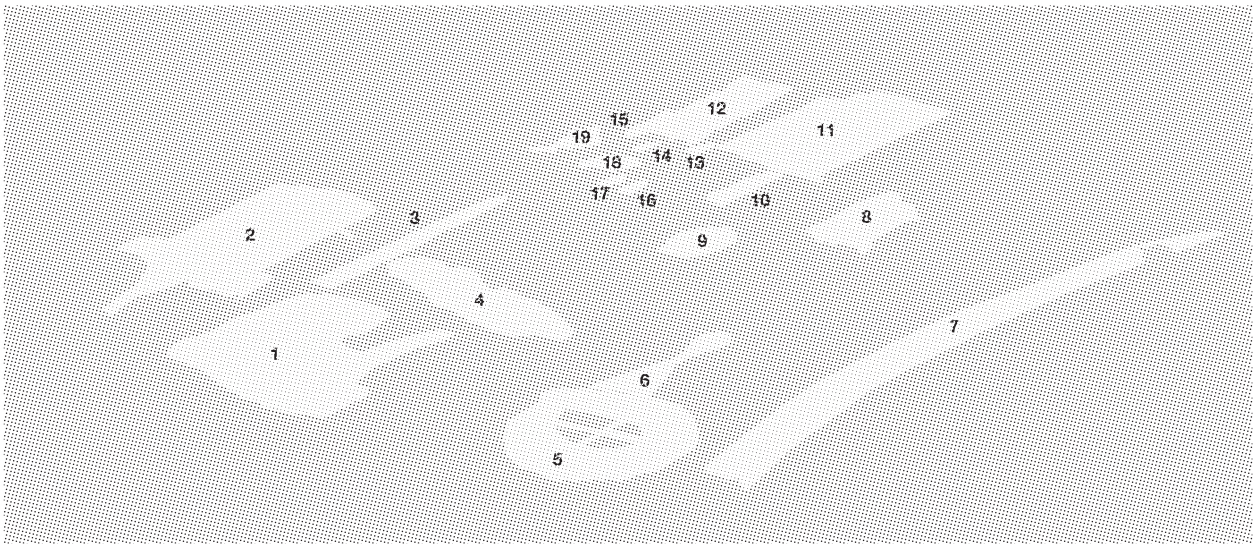


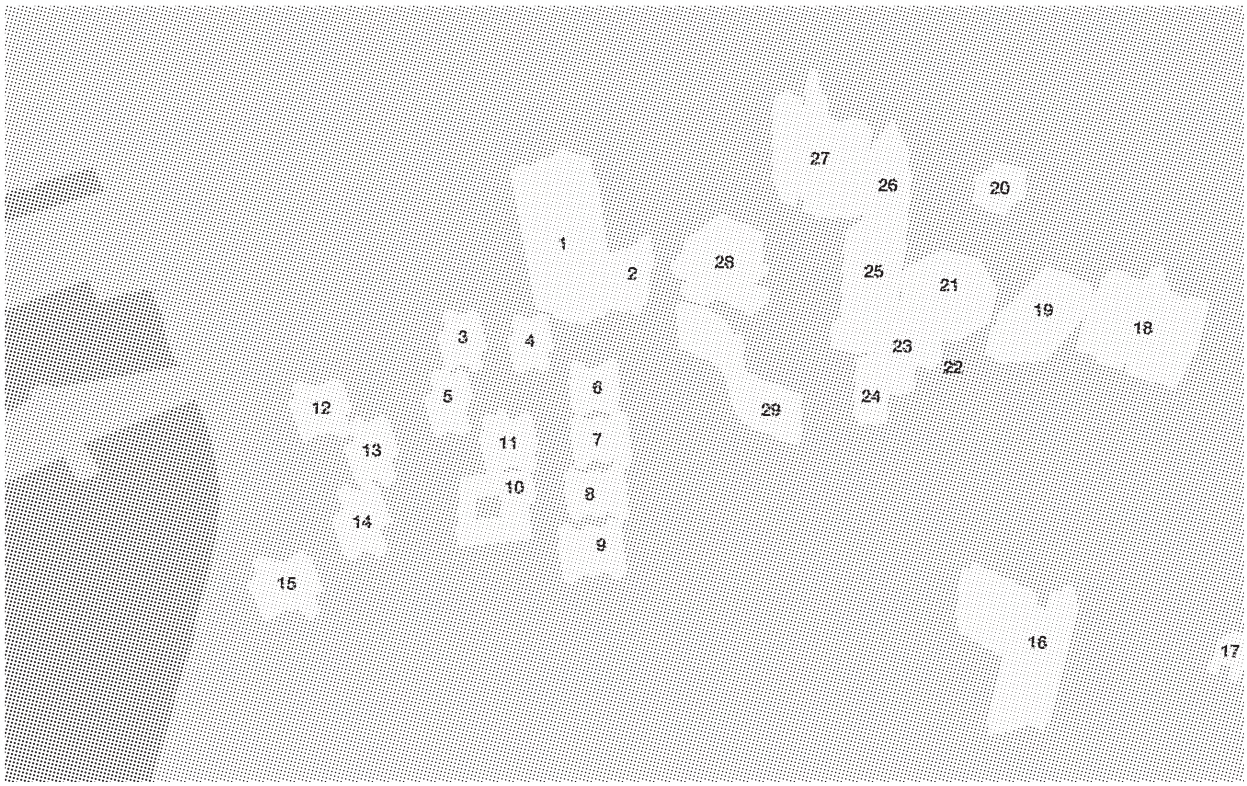


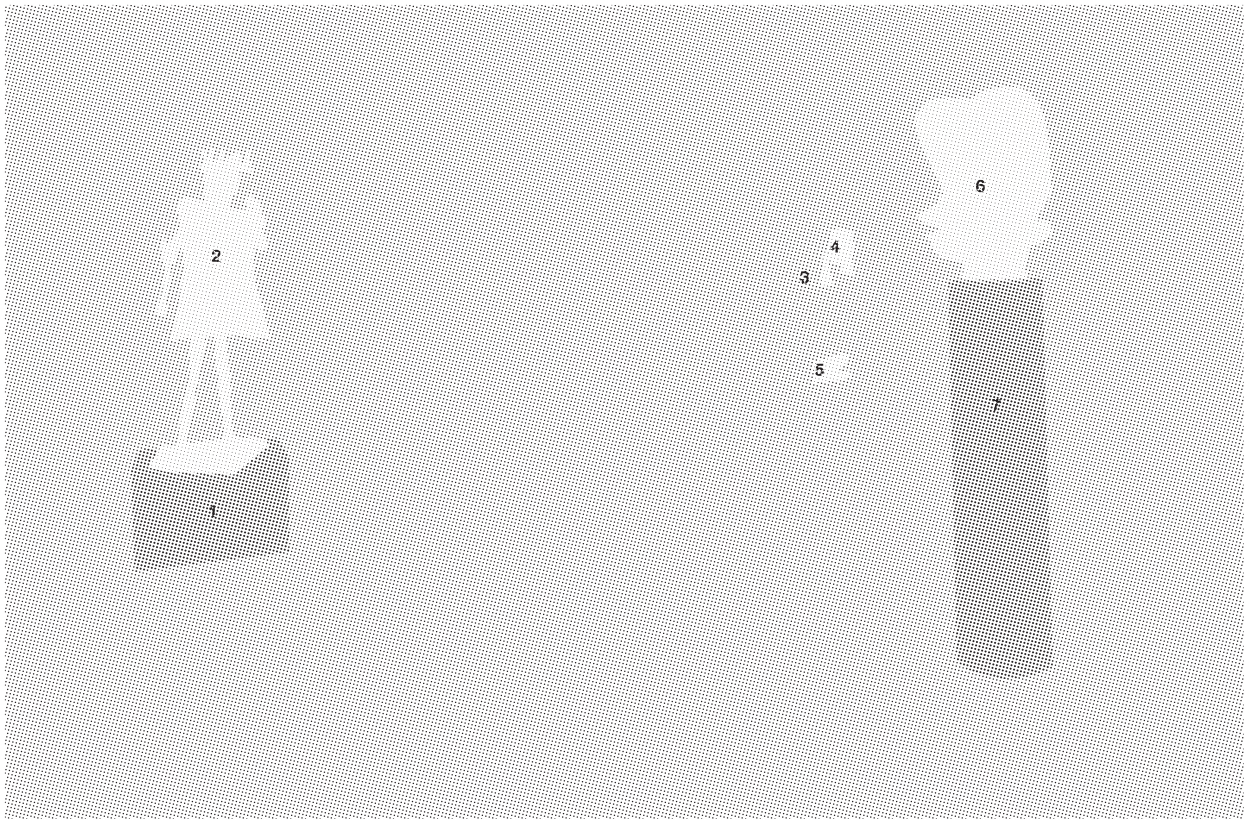

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19

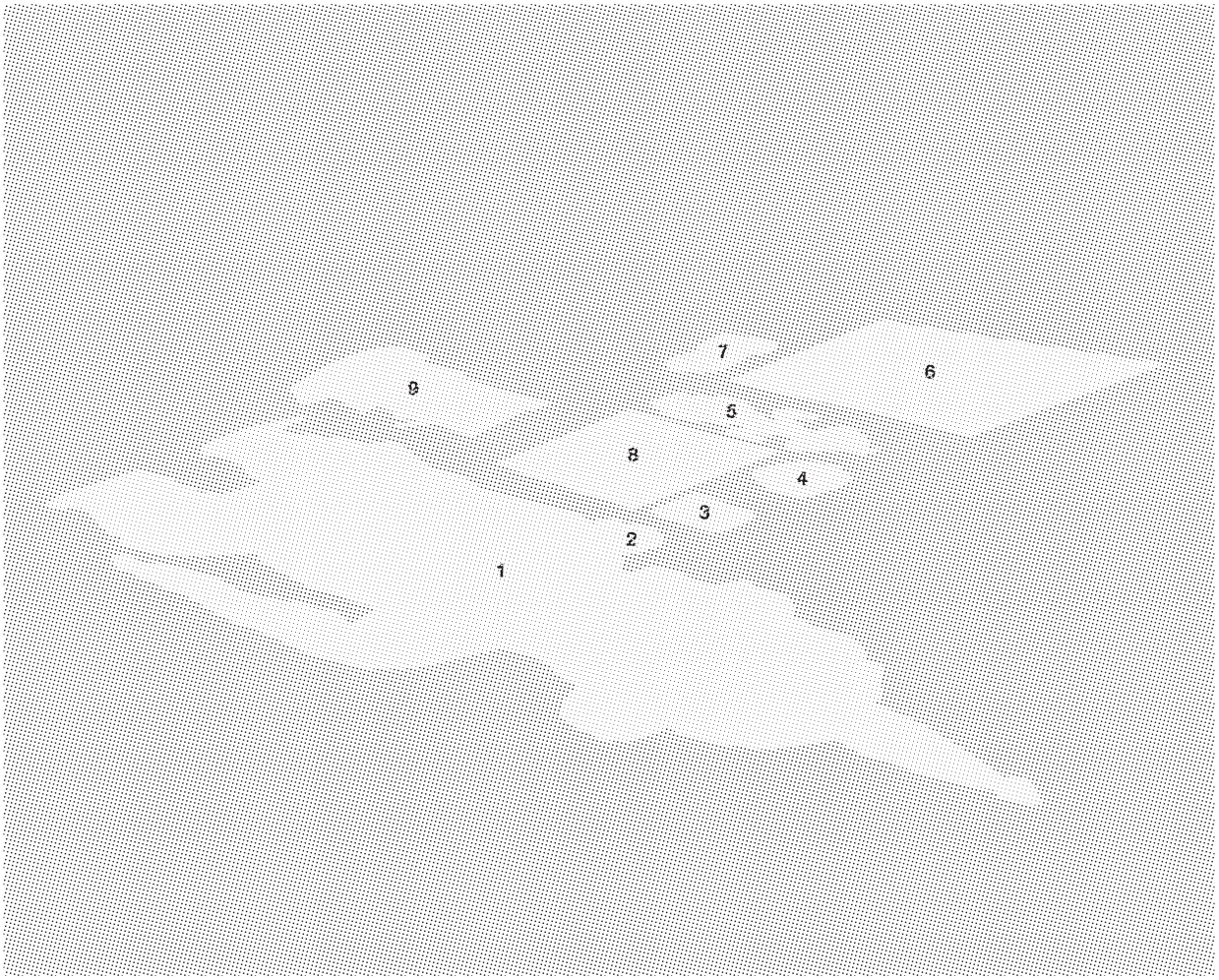

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

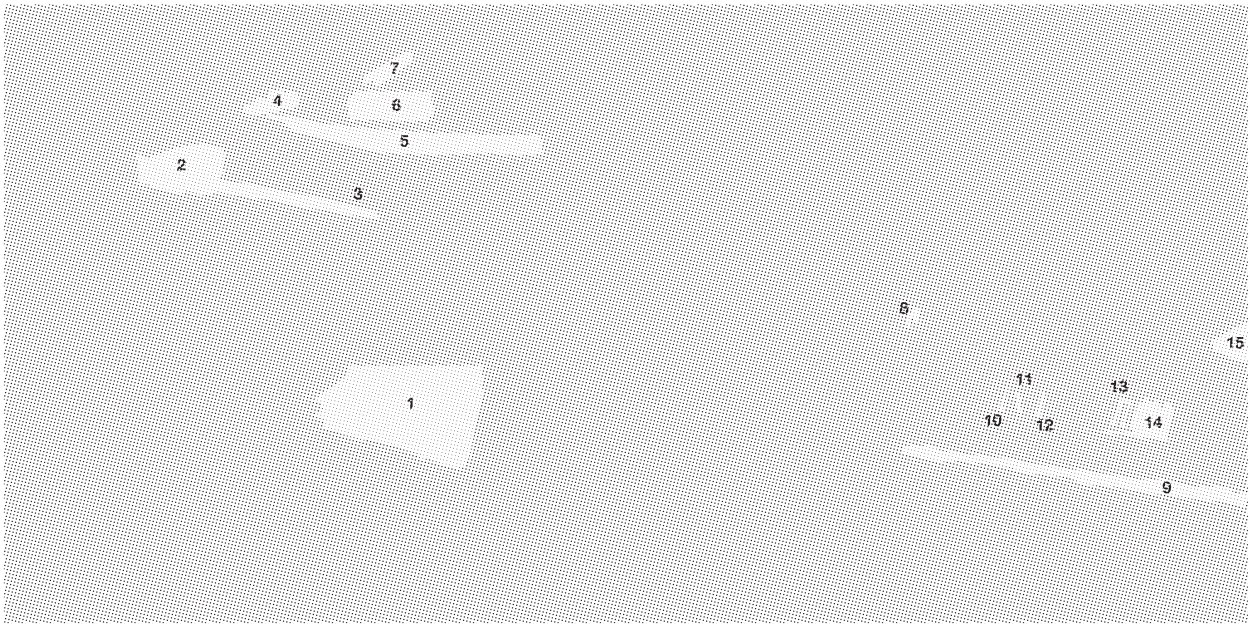

14
21
20
19
18
17
16
15
11
10
6
23
24
25
26
27
12
9
8
4
22
7
5
29
30
31
28
2
1
33
32

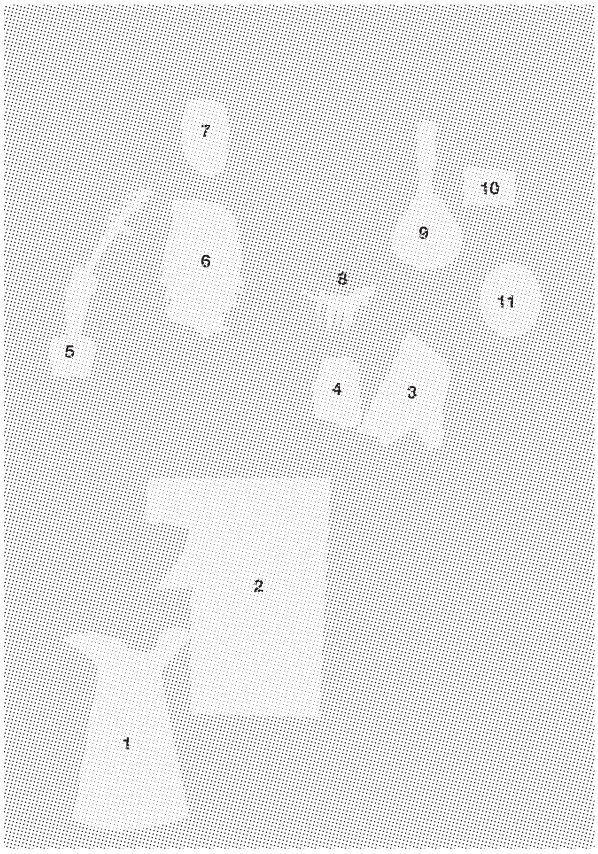

12
15
19
11
18
14
13
17
16
10
8
3
2
9
4
7
1
6
5

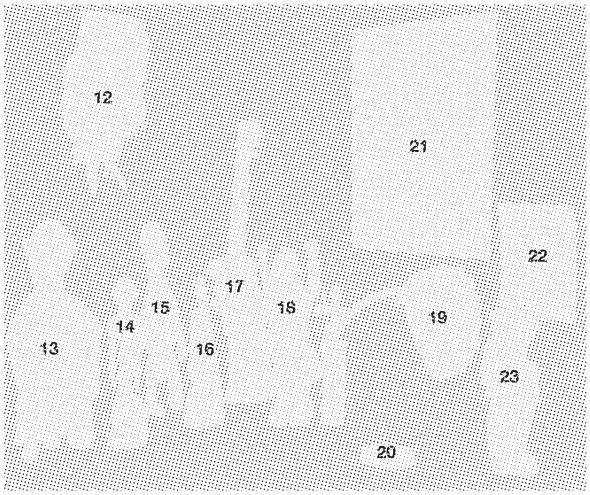

27
26
20
1
28
25
21
19
18
2
3
4
23
22
6
5
24
12
29
13
11
7
10
8
14
9
15
16
17

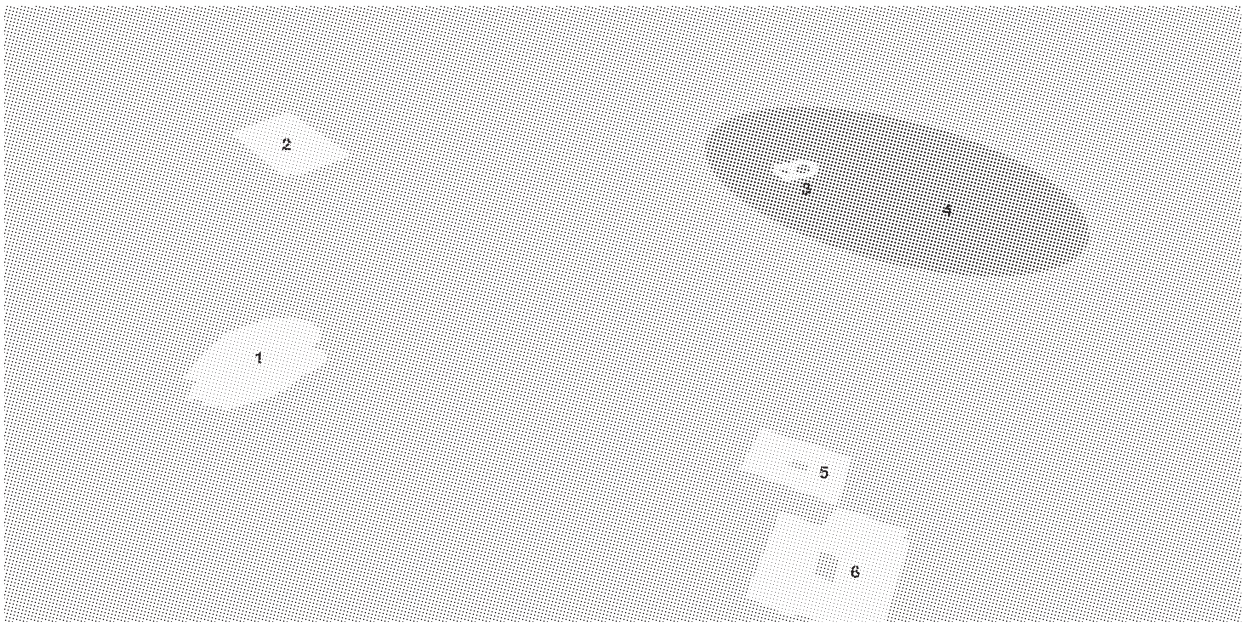

6
2
4
3
5
7
1

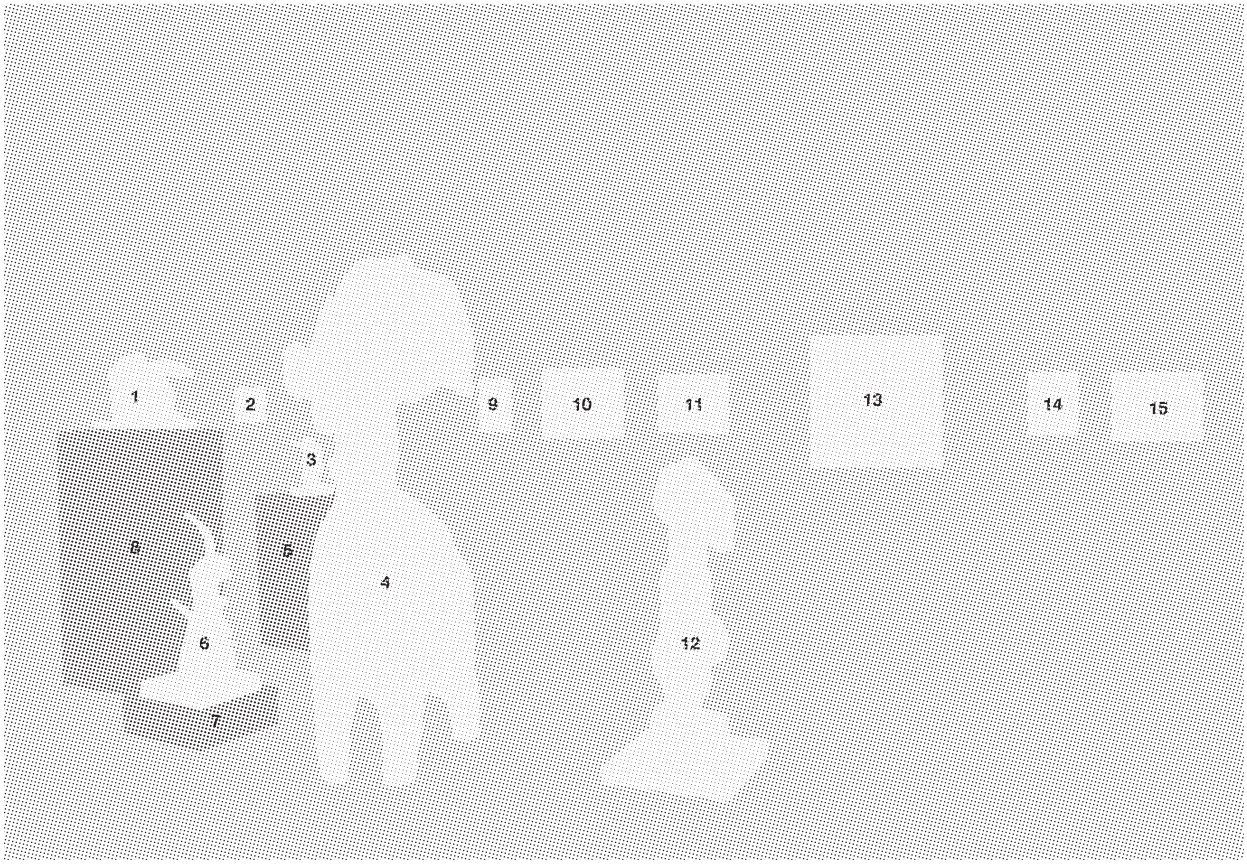

7
6
9
5
8
4
3
2
1

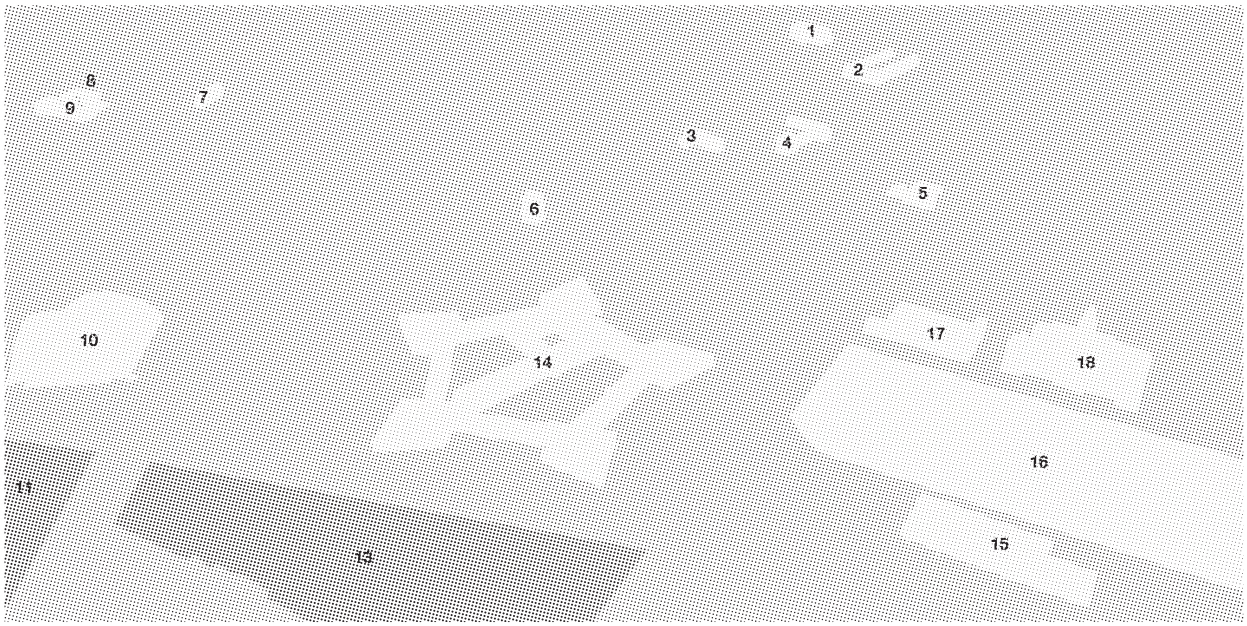

7
4
6
5
2
3
8
15
11
13
1
10
12
14
9

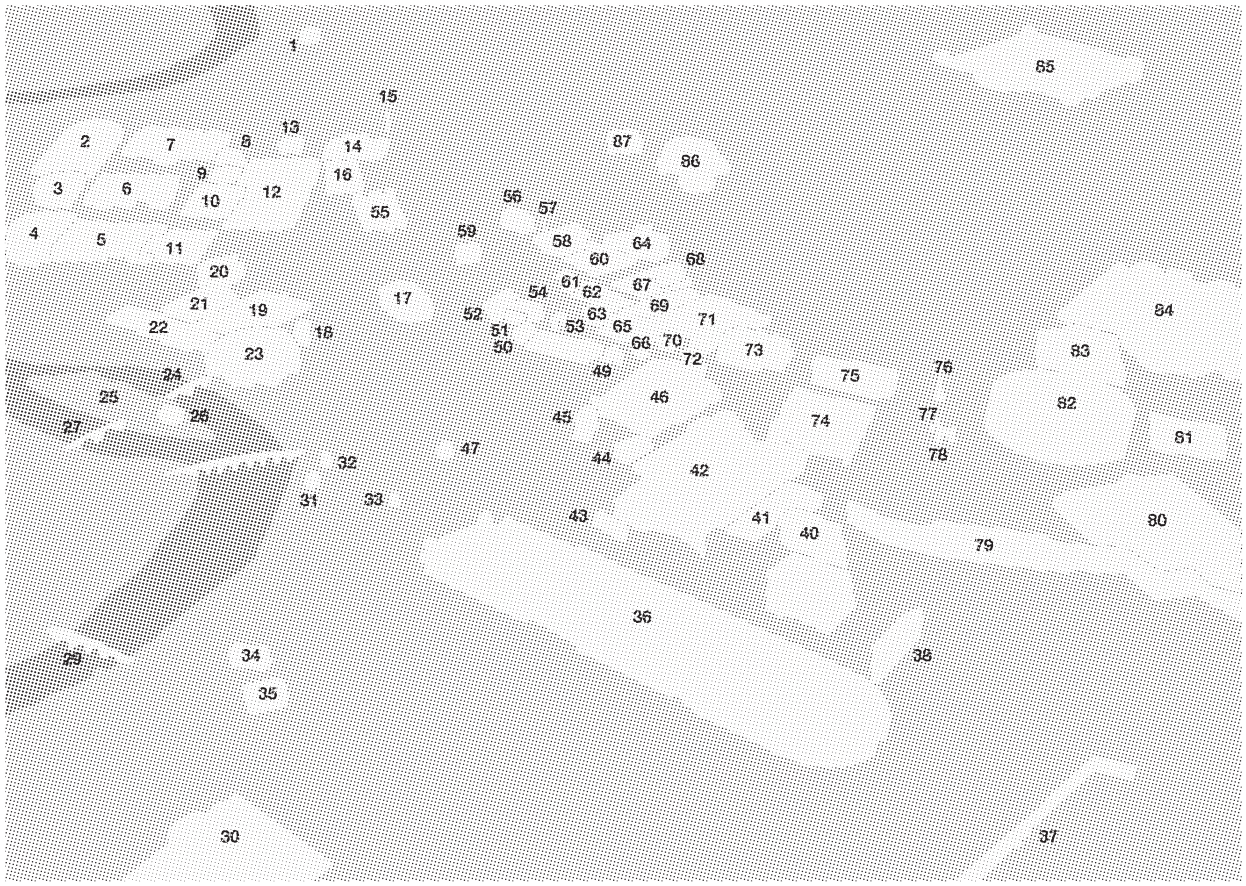

7
10
9
6
8
11
5
4
3
2
1


12
21
22
17
15
18
19
14
13
16
23
20


2
3
4
1
5
6


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87

One and Three Programs

Joseph Kosuth's 1965 work *One and Three Chairs* is often regarded as one of the most significant works of 1960s conceptual art. Its simple but at the time groundbreaking form is anchored by a wooden chair, presented close to the wall in a gallery space. In that wall, two panels hang, aligned by their upper edge, one to each side of the chair. To the left of the viewer, the panel presents a photograph of the same chair in the same space, enlarged to the actual size of the physical chair. To the right, a dictionary definition of 'chair,' reproduced as is. The work changes with every installation; different chairs can be selected as the center-piece, and the photograph must always correspond to that chair at that location. The dictionary extract is also undefined, so at times it may read as the OED, and at others as the Merriam-Webster, and maybe if it is installed in Mexico it will define *silla*, instead of chair. The work is determined by its compliance with a set of diagrammatic instructions by the artist that describe the kinds of objects and their relative positions. In its most evident interpretation, it deals with semantic in/congruities and with the attainment and production of meaning and definition. It begs the question of how these three objects we are seeing relate to each other. They might be perceived as instances of a same theoretical class, actually the same thing, multiple representations of a single object, an object and its abstractions, and so forth. The title forces the suggestion of singularity and multiplicity, synchronicity and rupture. It makes itself ambiguous in its declaration of one concept and three occurrences or one object and three notations or one notation and three performances. To make it even more satisfactorily unsatisfactory in its refusal to provide answers, the fact that it is reassembled from a virtual infinity of chairs, representations and denotations with the basis on instructions and not on specific embodied elements makes the ones and threes turn into thousands, and the whole roster of questions the work im-

plies turn onto itself: is it one and three works, or a single one? Additionally, does every execution compliant with the same order generate the same result? Is it supposed to?

One and Three Chairs, Joseph Kosuth, 1965 © 2018 Joseph Kosuth / Artists Rights Society (ARS), New York, Photo credits: MoMA Archives

Many of the same concerns surround the ontologies of code. Software is a set of instructions but is also the operation of hardware and the performance of the program. The discernment among these stages or interpretations of code has important political effects, as assuming total integrability between command and operation also assumes total control and impermeability. Looking back at Chun, Butler and Bourdieu, thinking of the performative utterance as a means of enacting by plain enunciation, without regards to the social (and technical) conditions that provide its authority and capabilities, is "a wish to return to a simpler and reassuring map of power, one in

which the assumption of sovereignty remains secure."[86]

Before advancing any further, it is important to clarify terminologically the main object of this section: source code. This fundamental aspect of modern computation is the human-legible textual layer that contains the instructions for the operation of the software. It is the most visible aspect of a programming language, and generally, it is compiled or interpreted into binary machine code to be stored as an executable file or to run as a program. It stands on itself as a multiple construct: a text to be read, a series of commands to be executed, a notation for a certain number of actions, and a denotative description of how to understand a specific software operation. Not only that, what "came to be know as the 'von Neumann architecture' … presented a single structure to hold both the set of instructions on how to perform the computation and the data required or generated by the computation."[87]

This merger, added to a paradoxical conceptual and discursive separation of software and hardware, were instrumental in establishing what Chun understands to be a process of "fetishization," the making of code into *logos* — source code as a "true representation of action"[88] or, rather, action itself. "In the beginning was the word, the instruction."[89] This notion of source as sorcery, able to affect change in the world by the naming of the right words, is what allows the program to move from text into time/process/space seamlessly. Exemplifying the tenet of source code as logos, the divine incantation that makes world from word, Florian Cramer's book *Words Made Flesh* traces a history of code and computation that expands far beyond the usual 19th

86 Butler, *Excitable Speech: A Politics of the Performative*, 78 quoted in Chun, *Programmed Visions*, location 442

87 Joasia Krysa and Grzesiek Sedek, "Source Code," *Software Studies: A Lexicon*, Leonardo (Cambridge, Mass. : MIT Press, 2008), 238

88 Chun, *Programmed Visions*, location 315

89 Chun, *Programmed Visions*, location 315

century cutoff with Charles Babbage and Ada Lovelace, and into the realms of magic, "the speech act that affects physical matter instantly and directly."[90] By approximating software to what he calls an "often utopian cultural imagination,"[91] the author points to the "obsessive persistence and contradictory mutations of the phantasm that symbols turn physical, and words are made flesh."[92] These phantasmal fears are grounded in a "seeming opacity and the boundless, viral multiplication of its output in the execution" and the "irrationality of rational systems,"[93] a certain agency that code as logos, like magic, has by itself.

Interestingly, John Hamilton's *Philology of the Flesh — philia* for the fleshly *logos* — speaks of a philological analysis that "exhibits a love that never wants to part with the word's material manifestation. It effectively denies the separateness of *logos* and its physical form."[94] Even to read code as *logos* then, one must read it as it is *incarnate*, made flesh, from its outset. This analysis "suggests that reading must resolutely attend to the flesh of language, that it must not neglect the primacy of the medium, even or precisely when that medium exceeds what it ostensibly aims to mediate."[95] The operational language of code must then be attended at the level of its physical realities.

Chun contends with the stance commonly taken by new media theorists such as N. Katherine Hayles and Alexander Galloway that conflates order and execution, assigning to the source code an ideal power of automatic causation. Galloway, to

90 Florian Cramer, Words Made Flesh: Code, Culture, Imagination (Rotterdam: Media Design Research, Piet Zwart Institute, 2005), 14

91 Florian Cramer, Words Made Flesh, 8

92 Florian Cramer, Words Made Flesh, 3

93 Florian Cramer, Words Made Flesh, 9

94 J. Hamilton, Philology of the Flesh (Chicago: The University of Chicago Press, 2018), 8

95 J. Hamilton, Philology of the Flesh, 9

make possible the erasure of the process of execution and substitute it by the source itself, "logically" equates one thing to another:

> One should never understand this 'higher' symbolic machine as anything empirically different from the 'lower' symbolic interactions of voltages through logic gates. They are complex aggregates yes, but it is foolish to think that writing an "if/then" control structure in eight lines of assembly code is any more or less machinic than doing it in one line of C … the relationship between the two is *technical*.[96]

However, to perceive it as such is to ignore that execution depends on translations, methods of compilation and interpretation that will take agency in the form through which the physical operation of the code takes place. "Source code as source," then, "means that software functions as an axiom, as 'a self-evident proposition requiring no formal demonstration to prove its truth, but received and assented to as soon as it is mentioned.'[97]"[98] This axiomatic quality "temporarily limits what can be decoded, put onto motion, by setting up an artificial limit — the artificial limit of programmability — that seeks to separate information from entropy, by designating some entropy information and other 'non intentional' entropy noise."[99]

More importantly, while the source code by definition maps out the entire range of possible actions of a program, "lines are read in as necessary,"[100] that is, a running program responds to its effects and inputs and uses functions and creates flows of action within the source code as they are made useful. Which is to say that the running

96 Alexander Galloway, "Language Wants to be Overlooked: Software and Ideology," *Journal of Visual Culture 5, no. 3* (2006), 321 quoted in Chun, Programmed Visions, location 381. emphasis on source.

97 *The Oxford English Online Dictionary* (Oxford and New York: Oxford University Press, 1992), quoted in Chun, *Programmed Visions*, location 777

98 Chun, *Programmed Visions*, location 777

99 Chun, *Programmed Visions*, location 783

100 Chun, *Programmed Visions*, location 394

program in fact reconstitutes the way in which the source is source, making it a "*re-source*,"[101] a map of what *has* happened at the execution, after it is merged with "code burned into silicon chips; and after all these signals are carefully monitored, timed, and rectified."[102] As Kosuth's chairs, software reconstitutes itself in every execution, while still composing a single unit — it is one and three source codes. Also like the chairs, there is a shifting flow of precedence between its elements in regards to which is representing which or which one is the origin of the other. While it is important to understand that there is necessarily a distance between code and performance, order and execution, and as such a space for agency in between, it may also be equally important to understand that there is no static supremacy of one form over the other and that the topological space between the source code and the running program is not constant but relative across different object pairings.

This presents itself as an important factor to be considered when producing readings of code, as the point in which one decides to interpret and interact with code produces specific results. Reading the source code is not equivalent to reading the executable file, which by its turn is not equivalent to reading the program while it is running. Each of these instances of software carry in themselves the possibilities of different meanings, as the way they do things in the world is particular to each and they are not logically or technically equivalent but are constantly constituting and reconstituting their performances and performatives.

To produce a method for the reading of software in its multiple dimensions, it is important to consider all products of the command and control chain of software as relevant sites. Consider an analogous structure in music: the relationship between performance and notation is equally suggestive to debate on whether these two con-

101 Chun, *Programmed Visions*, location 400

102 Chun, *Programmed Visions*, location 400

stitute two different objects, a single thing or a continuum of possibilities in between. A particularly common notion regarding musical notation is that it was conceived and serves a purpose as a memory technology. Adorno, however, contends with that conception by stating that notation is "not so much the preservation of something already present in tradition as the disciplinary function of the traditional exercise."[103] It represents a qualitative change in how music is made and understood, and "inaugurates a peculiar situation in which it exists, simultaneously, in two distinct yet related forms."[104] The very making of music is changed by that introduction, as notation becomes an object that constitutes a form to which certain kinds of music must agree. So to infer that the performance is always preponderant to the sheet is to discard the ordering effect that staff notation has on the exercise of European classical music, for instance. Conversely, to assume the sheet's superiority is to subdue the act of performance and the sounding of music to the prescriptions of the procedural signs, to disembody sound and disconsider that to be played, the music has to be interpreted, and is subjected to variation, error and, simply, the physical conditions of the world. One can think of it in terms of two extremes, represented by Adorno's view of music as a consequence of the score, to be expected from a musician grounded in twelve-tone serialism, to the experiments with scores and performance instructions that so relied on chance and experimentation, by John Cage and *Fluxus* artists such as Yoko Ono or George Brecht.

Going back to code, but with that in mind, Tenen reminds us that "formats relate matter and content. They are techniques in which immanent inscriptions, the electromagnetic charge, are transformed into transcendent digital objects. … Format-

103 Theodor W. Adorno, *Towards a Theory of Musical Reproduction: Notes, a Draft and Two Schemata* (Cambridge: Polity, 2006), 171

104 David Addyman, Matthew Feldman, and Erik Tonning, *Samuel Beckett and BBC Radio: A Reassessment* (Springer, 2017), 271

ting imposes structure."[105] And "reading for format explicitly … involve[s] the deconstruction, both literal and figurative, of the *textual laminate*,"[106] the combine of logic and physics, the instructions, the interfaces, the metals, the electricity, and so on, "that hold inscription in suspense."[107]

105 Tenen, Plain Text, location 1923

106 Tenen, Plain Text, location 2349

107 Tenen, Plain Text, location 2362

Score for Two Negotiative Events

first event

a conversation

props: yourself, a programmer, a program

talk about the program and its command structure

decide upon an alternative representation for the performance of the code

the programmer must lead this decision

this representation must be a performance in itself (of any kind)

it must not represent what the code aims to do as a program, but what it instructs as a code

it must not explicitly use any part of the computer code

it must not use any additional computer code

the making of a notational device or script is encouraged

second event

a performative situation

props: to be defined in previous conversation

performer: a programmer (the same)

others, human or not, may be added if needed

programmer enacts the orders contained in their program

follow definitions previously set in conversation (first event)

if notation is used, programmer is interpreter as well as performer

there need not be an audience

Conclusion

In his 1948 essay What is Literature? Sartre claims that to write is to appeal to the reader

> that he lead into objective existence the revelation which [the writer has] undertaken by means of language. And if it should be asked to what the writer is appealing, the answer is simple. As the sufficient reason for the appearance of the aesthetic object is never found either in the book (where we find merely solicitations to produce the object) or in the author's mind, and as his subjectivity, which he cannot get away from, cannot give a reason for the act of leading into objectivity, the appearance of the work of art is a new event which cannot be explained by anterior data. And since this directed creation is an absolute beginning, it is therefore brought about by the freedom of the reader, and by what is purest in that freedom. Thus, the writer appeals to the reader's freedom to collaborate in the production of his work.[108]

As I have tried to demonstrate in this thesis, to assume a readerly position in face of any text, verbal or otherwise, is to put oneself in a position of relevance in the construction of its meaning. When it comes to software, the discursive regimes that surround it have pushed regular users and affected individuals out of this position, and meaning-making authority has been highly concentrated on one end of this circulation system. Relying on Hall's terminology once again, the status of code today is to its larger extent based on dominant-hegemonic decodings.

While much of this text was spent arguing for an association of software with the exercise of control, such breath of analysis is hardly necessary today to justify this notion. We are, for the most part, increasingly aware of the ways in which code-running technology manages our behaviors and affects and is affected by our habitus, only not necessarily with the same vocabulary or clarity. What this thesis tries to get at

108 Jean-Paul Sartre, *What Is Literature?* (Gloucester, Mass. : P. Smith, 1978), 46

is that authority is mutually constructed in a process that is both political and aesthetic and we can produce negotiations with containers of power by using any and all of these two means. Refusing the notion of art as a mere reflection or representation of a current state, and aiming for an understanding of art as being able to disturb and sway norms, practices and dispositions, I suggest that this active process of reading can lend us back agency in dealing with the overbearing command structures contained in modern computation, from the most obvious Leviathan-seeming intelligence gathering project to the subtlest nudging of social media.

Far from taking on a luddite attitude, or, as my mother would say, throwing the baby out with the bathwater, I find extremely relevant to notice that this is in no way a condemnation of software, computation, code or digital technology, and that all of these have profoundly positive, interesting and beautiful outcomes in the contemporary world. To a similar extent, however, they are generating or are implicated in very problematic social arrangements, and it is our responsibility as users, cultural producers, technologists, theorists, and the like to re-architect the technology and, maybe most easily and most importantly, its current uses. Making a constant effort to avoid illusions of symmetric distributions of power in the reading process itself, an artistic mode of reading is one that acknowledges that all coding and decoding practices emerge from a structured social context, but that allows itself to betray this context and uphold the relevance of uncertainty and ambiguity. It is a way of looking that considers that everywhere you look at is a specific site, and that no other site will be equivalent — and can only be closely related — to that one. And most of all, apart from all other things it can also be (but that this document was too short and this author too limited to encompass), it is committed to negotiating the (op)positions of the ifs, thens, ors, ands and nots.

Bibliography

Adorno, Theodor W. *Towards a Theory of Musical Reproduction: Notes, a Draft and Two Schemata*. Cambridge: Polity Press, 2006.

Addyman, David, Feldman, Matthew, and Tonning, Erik. *Samuel Beckett and BBC Radio: A Reassessment*. New York: Springer, 2017.

Austin, J. L. *How to Do Things with Words*, The William James Lectures Delivered at Harvard University in 1955. Cambridge: Harvard University Press, 1975.

Barthes, Roland. "The Death of the Author". *Image, Music, Text.* ed. Stephen Heath. London: Fontana, 1984. p. 143-148

Berger, Peter L., and T. Luckmann. *The Social Construction of Reality: A Treatise in the Sociology of Knowledge*. New York, NY: Doubleday, 1966

Berry, David. *The Philosophy of Software*. Basingstoke: Palgrave Macmillan, 2011

Bourdieu, Pierre. *The Rules of Art: Genesis and Structure of the Literary Field*. Stanford, CA: Stanford University Press, 1996.

Bourdieu, Pierre. *The Field of Cultural Production: Essays on Art and Literature*. Cambridge: Polity Press, 1993.

Bourdieu, Pierre. *Language and Symbolic Power*. ed. John B. Thompson. Cambridge, MA: Harvard University Press, 1991.

Butler, Judith. *Excitable Speech: A Politics of the Performative.* New York : Routledge, 1997.

Chartier, Roger. *Forms and Meanings: Texts, Performances, and Audiences from Codex to Computer*. Philadelphia, PA.: University of Pennsylvania Press, 1995.

Chun, Wendy Hui Kyong. *Programmed Visions*. Cambridge, MA: The MIT Press, 2011

de Certeau, Michel. *The Practice of Everyday Life*. Berkley, CA: University of California Press, 1984.

Cox, Geoff and McLean, Alex. *Speaking Software*. Cambridge, MA: The MIT Press, 2012.

Cramer, Florian. *Words Made Flesh: Code, Culture, Imagination*. Rotterdam: Media Design Research, Piet Zwart Institute, 2005.

Eletronic Frontier Foundation, "List of Printers Which Do or Do Not Display Tracking Dots," https://www.eff.org/pages/list-printers-which-do-or-do-not-display-tracking-dots

Foucault, Michel."What is an Author?" *Aesthetics, Method, and Epistemology*. ed. James D. Faubion (New York, NY: The New Press, 1998), 206-222

Frabetti, Federica. *Software Theory*. London: Rowman & Littlefield International, 2014.

Fuller, Michael. *Software Studies: a Lexicon*. Cambridge, MA: The MIT Press, 2008.

Galloway, Alexander. *Protocol: How Power Exists after Decentralization*. Cambridge, Mass.: MIT Press, 2004.

Galloway, Alexander. "Language Wants to be Overlooked: Software and Ideology," *Journal of Visual Culture 5, no. 3*, 2006.

Glissant, Édouard. *Poetics of Relation*. Ann Arbor : University of Michigan Press, 1997.

Gordon, Colin. "Governmental Rationality: An Introduction," in *The Foucault Effect: Studies in Governmentality*, ed. Graham Burchell et al. Chicago: Chicago University Press, 1991

Hall, Stuart. "Encoding/Decoding." *Culture, Media, Language: Working Papers in Cultural Studies, 1972-1979*. ed. Stuart Hall et tal. (London, Eng.: Routledge, 1991), 128-137

Halliday, M. A. K. "Anti-Languages." *American Anthropologist 78* (1976): 570-584

Hamilton, J. *Philology of the Flesh*. Chicago: The University of Chicago Press, 2018.

Hayles, N. Katherine. *My Mother Was a Computer*. Chicago, IL: The University of Chicago Press, 2005.

Hui, Yuk. *On the Existence of Digital Objects*. Minneapolis, MN: University of Minnesota Press, 2016

Laplace, Pierre-Simon. *A Philosophical Essay on Probabilities*. 1820; reprinted, New York: Dover, 1951.

Mackenzie, Adrian. *Cutting Code: Software and Sociality*. New York, NY: Peter Lang, 2006

Manovich, Lev. *Software Takes Command*. New York, NY: Bloomsbury Academic, 2013.

Marino, Mark C. "Critical Code Studies," *Electronic Book Review*, 12/04/2006, http://www.electronicbookreview.com/thread/electropoetics/codology

McKenzie, D. F. *Bibliography and the Sociology of Texts*. Cambridge: Cambridge University Press, 1999.

Milton, John. *Areopagitica: a Speech for the Liberty of Unlicensed Printing, to the Parliament of England*. London : reprinted for R. Blamire, 1792.

Montfort, Baudoin, Bell, Bogost, Douglass, Marino, Mateas, Reas, Sample, Vawter. *10 PRINT CHR$(205.5=rND(1)); GOTO 10*. Cambridge, MA: The MIT Press, 2012.

Melzer, Tine. *Taxidermy for Language-Animals*. Zürich: Rollo Press, 2016.

Reeds, Jim. "Solved: The Ciphers In Book III Of Trithemius's Steganographia," *Cryptologia*, 22:4. https://doi.org/10.1080/0161-119891886948

Robson, David. "The Secret 'Anti-Languages' You Are Not Suposed to Know." accessed 27 February 2017, http://www.bbc.com/future/story/20160211-the-secret-anti-languages-youre-not-supposed-to-know

Sartre, Jean-Paul. *What Is Literature?* Gloucester, Mass.: P. Smith, 1978.

Shannon, Claude Elwood. "A Mathematical Theory of Communication," *The Bell System Technical Journal. Vol. 27* (July 1948): 379–423.

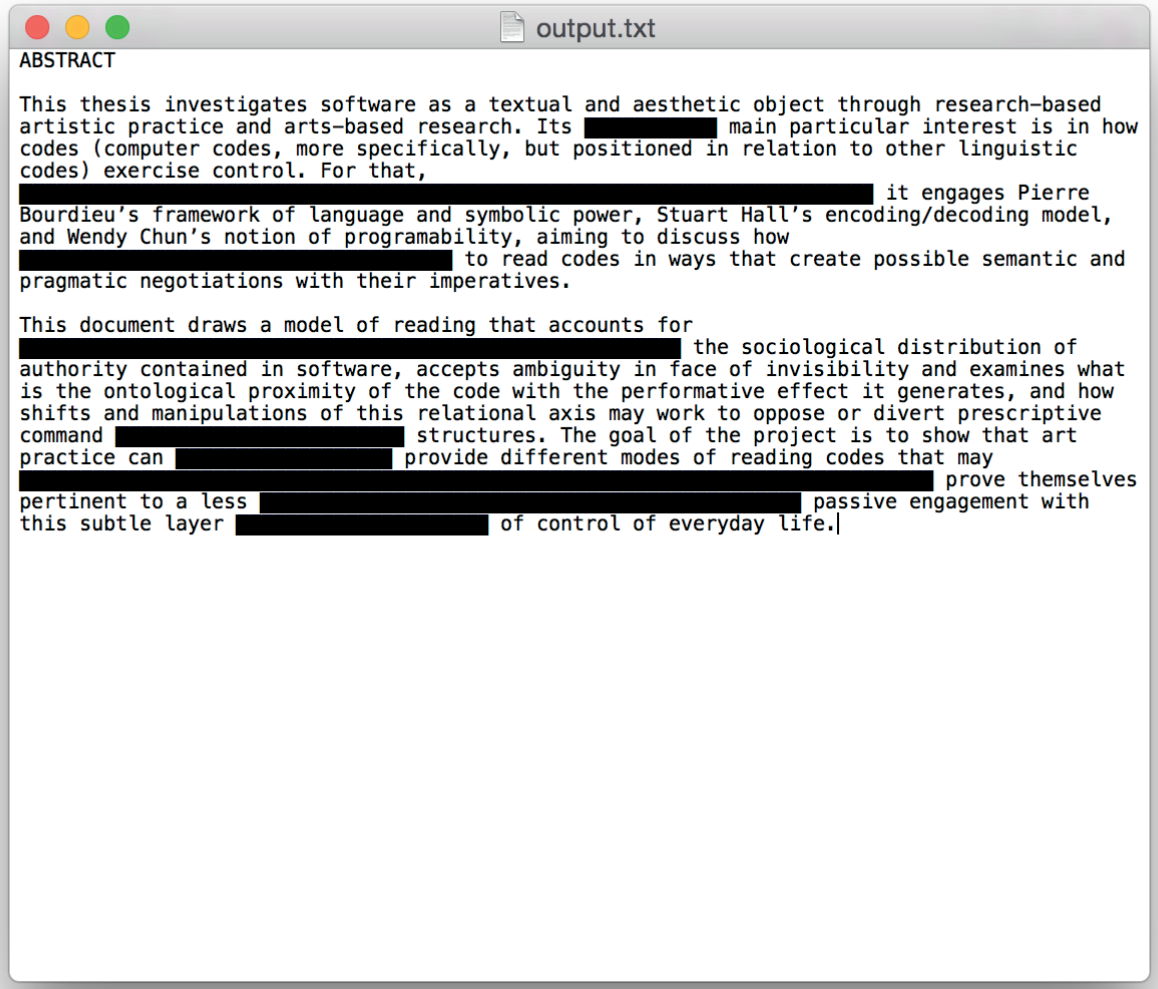
Stanton, Neville, Baber, Christopher, and Harris, Don. *Modelling Command and Control: Event Analysis of Systemic Teamwork*. Ashgate Publishing, Ltd., 2008.

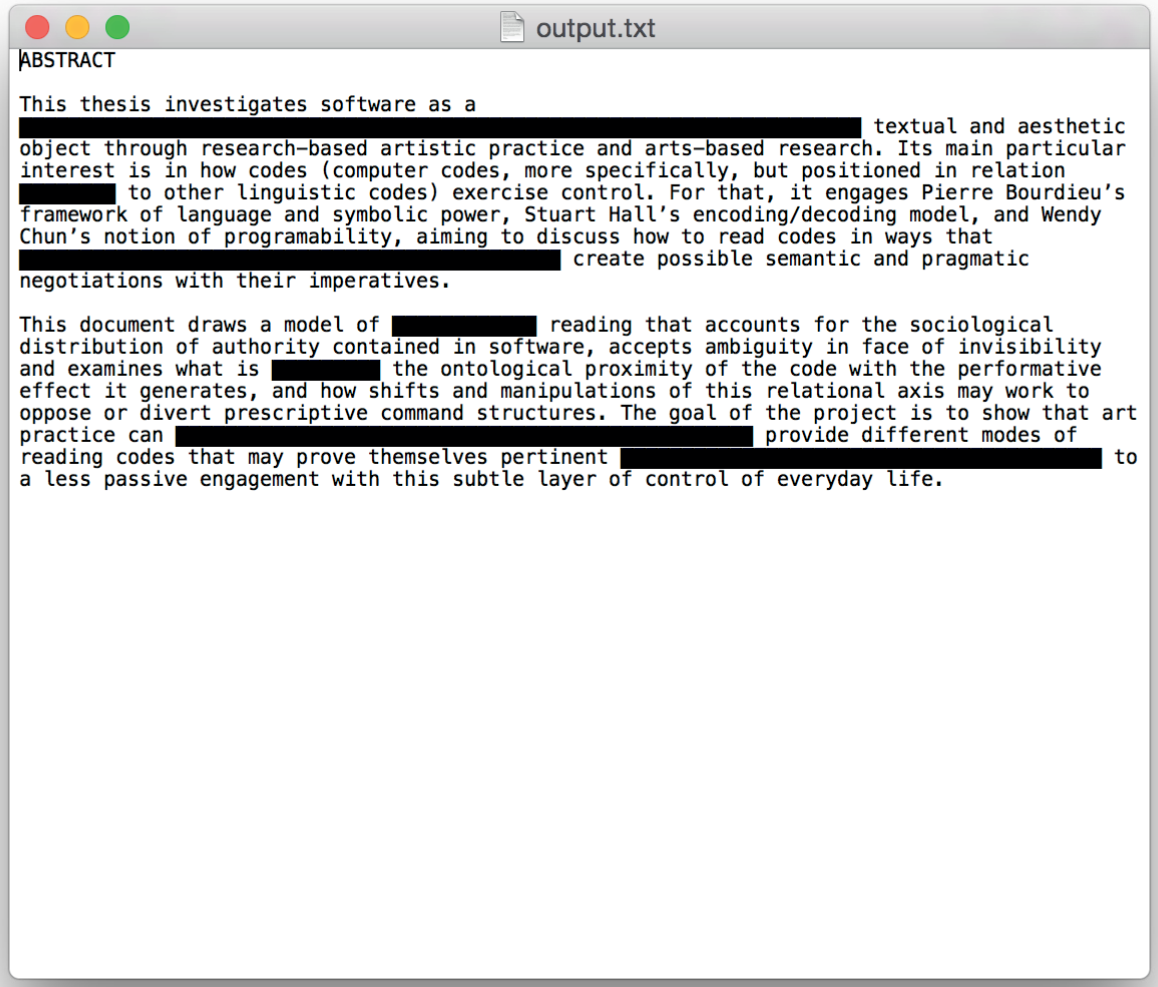
Sterne, Laurence. *Tristram Shandy*. London : s.n., 1795.

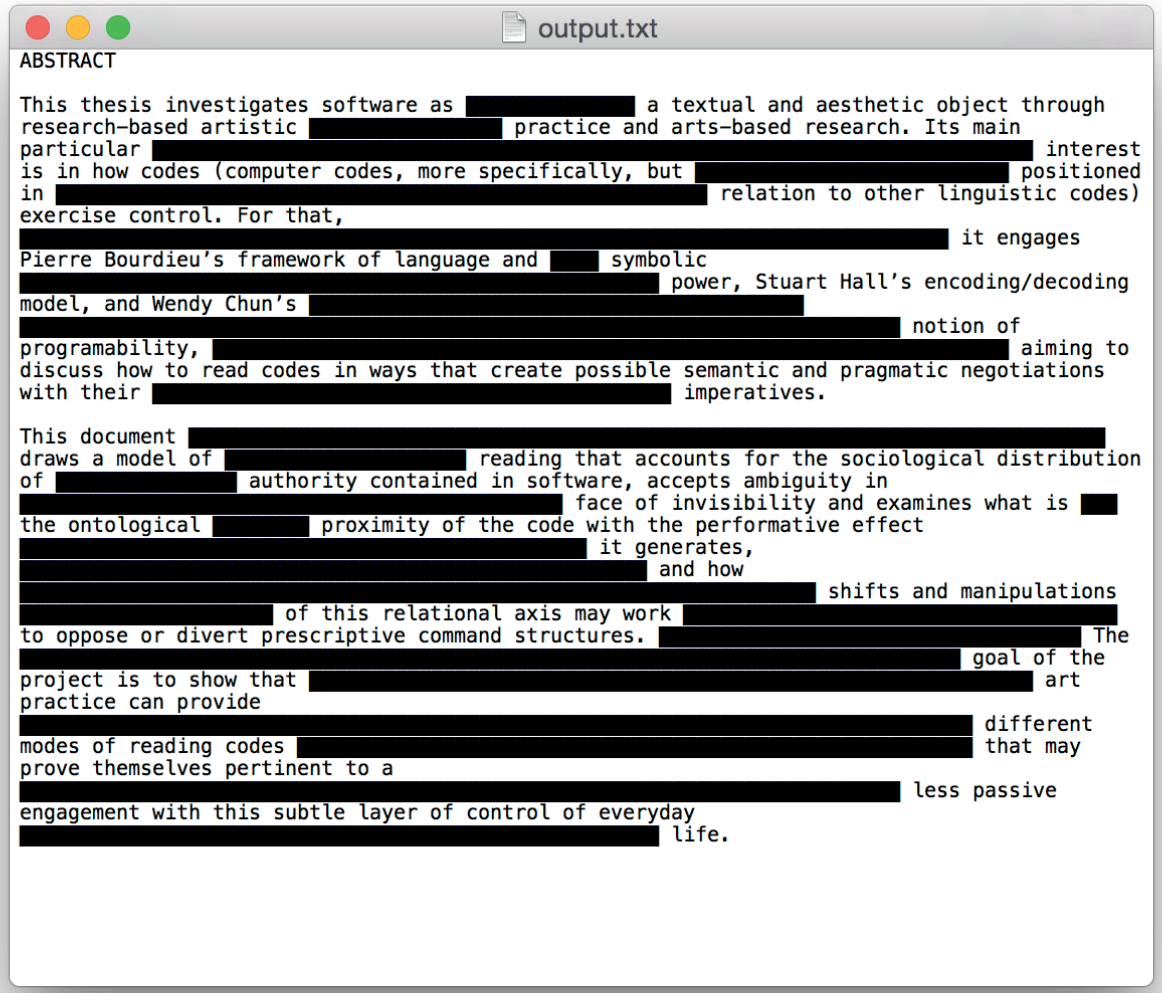
Tenen, Dennis. *Plain Text: The Poetics of Computation*. Stanford, CA: Stanford University Press, 2017

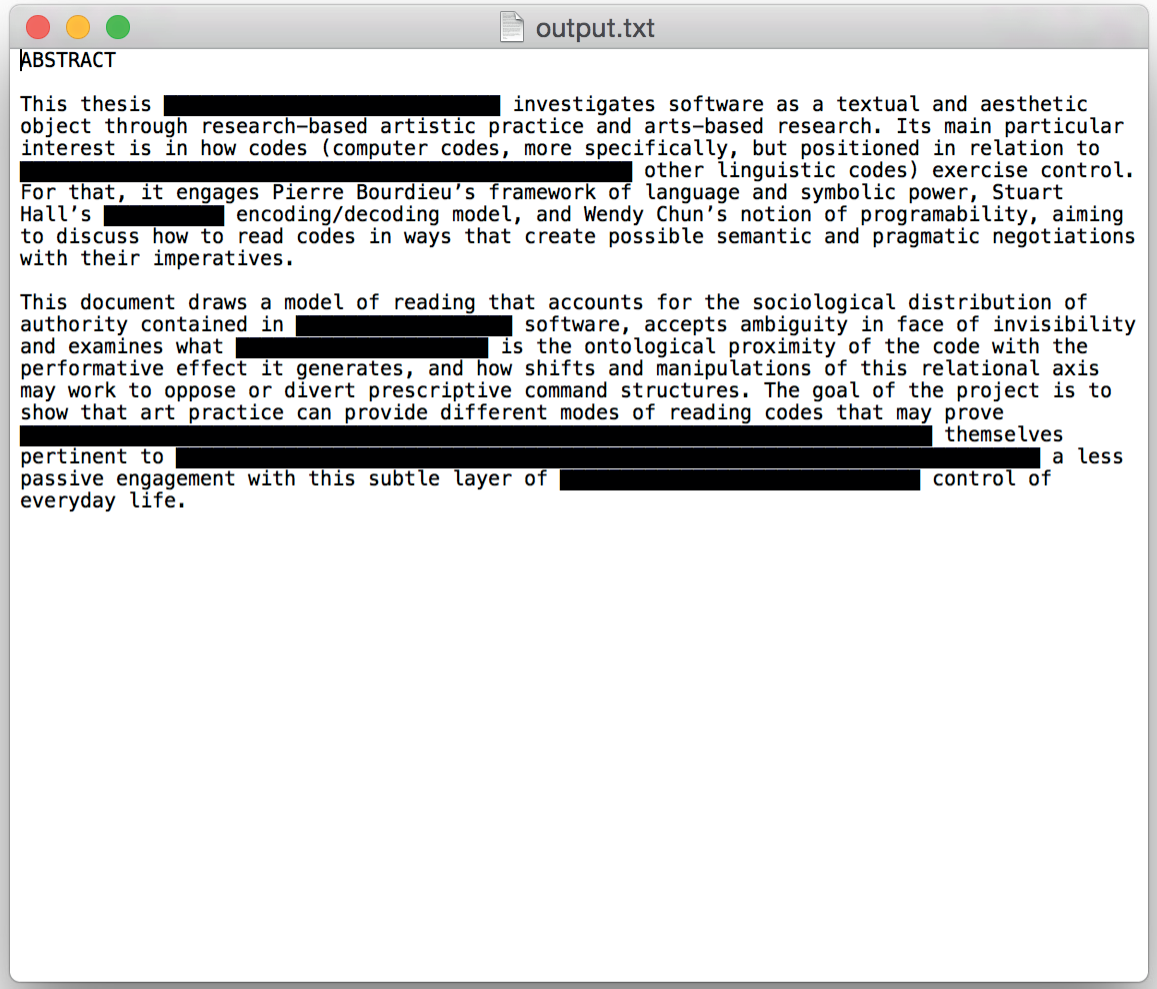
Weiner, Norbert. *Cybernetics, Or, Control and Communication in the Animal and the Machine*. Cambridge, Mass. : M.I.T. Press, 1961.

Wittgenstein, Ludwig. *Philosophical Investigations*. Oxford: B. Blackwell, 1986.


output.txt
ABSTRACT
This thesis investigates software as a textual and aesthetic object through research-based artistic practice and arts-based research. Its [redacted] main particular interest is in how codes (computer codes, more specifically, but positioned in relation to other linguistic codes) exercise control. For that, [redacted] it engages Pierre Bourdieu's framework of language and symbolic power, Stuart Hall's encoding/decoding model, and Wendy Chun's notion of programability, aiming to discuss how [redacted] to read codes in ways that create possible semantic and pragmatic negotiations with their imperatives.
This document draws a model of reading that accounts for [redacted] the sociological distribution of authority contained in software, accepts ambiguity in face of invisibility and examines what is the ontological proximity of the code with the performative effect it generates, and how shifts and manipulations of this relational axis may work to oppose or divert prescriptive command [redacted] structures. The goal of the project is to show that art practice can [redacted] provide different modes of reading codes that may [redacted] prove themselves pertinent to a less [redacted] passive engagement with this subtle layer [redacted] of control of everyday life.


output.txt
ABSTRACT
This thesis investigates software as a [redacted] textual and aesthetic object through research-based artistic practice and arts-based research. Its main particular interest is in how codes (computer codes, more specifically, but positioned in relation [redacted] to other linguistic codes) exercise control. For that, it engages Pierre Bourdieu's framework of language and symbolic power, Stuart Hall's encoding/decoding model, and Wendy Chun's notion of programability, aiming to discuss how to read codes in ways that [redacted] create possible semantic and pragmatic negotiations with their imperatives.
This document draws a model of [redacted] reading that accounts for the sociological distribution of authority contained in software, accepts ambiguity in face of invisibility and examines what is [redacted] the ontological proximity of the code with the performative effect it generates, and how shifts and manipulations of this relational axis may work to oppose or divert prescriptive command structures. The goal of the project is to show that art practice can [redacted] provide different modes of reading codes that may prove themselves pertinent [redacted] to a less passive engagement with this subtle layer of control of everyday life.


output.txt
ABSTRACT
This thesis investigates software as [redacted] a textual and aesthetic object through research-based artistic [redacted] practice and arts-based research. Its main particular [redacted] interest is in how codes (computer codes, more specifically, but [redacted] positioned in [redacted] relation to other linguistic codes) exercise control. For that, [redacted] it engages Pierre Bourdieu's framework of language and [redacted] symbolic [redacted] power, Stuart Hall's encoding/decoding model, and Wendy Chun's [redacted] notion of programability, [redacted] aiming to discuss how to read codes in ways that create possible semantic and pragmatic negotiations with their [redacted] imperatives.
This document [redacted] draws a model of [redacted] reading that accounts for the sociological distribution of [redacted] authority contained in software, accepts ambiguity in [redacted] face of invisibility and examines what is [redacted] the ontological [redacted] proximity of the code with the performative effect [redacted] it generates, [redacted] and how [redacted] shifts and manipulations [redacted] of this relational axis may work [redacted] to oppose or divert prescriptive command structures. [redacted] The [redacted] goal of the project is to show that [redacted] art practice can provide [redacted] different modes of reading codes [redacted] that may prove themselves pertinent to a [redacted] less passive engagement with this subtle layer of control of everyday [redacted] life.


output.txt
ABSTRACT
This thesis [redacted] investigates software as a textual and aesthetic object through research-based artistic practice and arts-based research. Its main particular interest is in how codes (computer codes, more specifically, but positioned in relation to [redacted] other linguistic codes) exercise control. For that, it engages Pierre Bourdieu's framework of language and symbolic power, Stuart Hall's [redacted] encoding/decoding model, and Wendy Chun's notion of programability, aiming to discuss how to read codes in ways that create possible semantic and pragmatic negotiations with their imperatives.
This document draws a model of reading that accounts for the sociological distribution of authority contained in [redacted] software, accepts ambiguity in face of invisibility and examines what [redacted] is the ontological proximity of the code with the performative effect it generates, and how shifts and manipulations of this relational axis may work to oppose or divert prescriptive command structures. The goal of the project is to show that art practice can provide different modes of reading codes that may prove [redacted] themselves pertinent to [redacted] a less passive engagement with this subtle layer of [redacted] control of everyday life.