Software Studies

Matthew Fuller, Lev Manovich, and Noah Wardrip-Fruin, editors

Expressive Processing: Digital Fictions, Computer Games, and Software Studies,
Noah Wardrip-Fruin, 2009

Code/Space: Software and Everyday Life,
Rob Kitchin and Martin Dodge, 2011

Programmed Visions: Software and Memory,
Wendy Hui Kyong Chun, 2011

Speaking Code: Coding as Aesthetic and Political Expression,
text: Geoff Cox; code: Alex McLean, 2012

Speaking Code

Coding as Aesthetic and Political Expression

text: Geoff Cox

code: Alex McLean

foreword by Franco "Bifo" Berardi

The MIT Press
Cambridge, Massachusetts
London, England

While the schools drill human beings in speech . . . , the pupils become increasingly mute. They can give lectures; every sentence qualifies them for the microphone, before which they can be placed as spokesmen for the average; but their capacity for speaking to each other is stifled. It presupposes experience worth communicating, freedom of expression, and at once independence and relatedness. In the all-embracing system conversation becomes ventriloquism.

—Theodor W. Adorno, "Institute for Deaf-Mutes"

0 Double Coding

```
#!/usr/bin/befunge
>                     v
The  book begins  by  announcing  itself   recursively.  In stating  the
phrasev"Hello World!"<it follows  the  convention that programmers adopt
whenv:<learning a new language. The   paragraph executes itself in a way
that# encapsulates  the  inherent   action  of  code, and  the  central
importance  that will be  developed  around  speech acts  in  the world.
 v:,_@
 >  ^
```

Using the Befunge programming language to render a first paragraph like this sets the tone for the book in not just describing code and what it does but doing it. As an esoteric language, Befunge also breaks with the conventions of downward direction of interpretation through two-dimensional syntax.[1] This is done using punctuation, with each of the four instructions "^>v<" represented by graphical arrows, which change the direction of control flow. "@" ends the program.

The book begins with the "hello world" convention[2] to highlight the ubiquity of speech in everyday communications and the paradoxical tendency seemingly to diminish the power of the voice. Humans and machines increasingly converse with other humans and machines, making our languages ever more codified, but the meanings produced through them are ever more prone to misunderstanding—in the confused spaces between the encoding and decoding of the utterance. Such combinations of natural and artificial languages demonstrate a multilingual human-machine confusion of tongues, under the conditions of contemporary capitalism that have integrated language, intellect, and affect into production.

If in the beginning was the spoken word,[3] this appears to denote the human condition in the breath or essence of life. Underpinning this is the commonly held belief that the soul is the source of speech, which has been forever perverted through the use of spoken language. In the book of Genesis, the world was understood originally to contain one language only—the single language of Adam who first named objects

in the world. The story unfolds that the Tower of Babel, designed to reach into heaven, displeased God so much that he (sic) decided to "confound the language of all the earth."[4] Subsequently, everyone has been left to babble in a diversity of languages and confusion of human-machine "tongues," or in new hybrid forms that combine the formal structures of natural languages and program code. These constitute the contemporary babble of communications technologies[5]—resonant in the naming of the contemporary social media platform Twitter with its reductive register of up to 140 characters (condemning everyone to communicate in what is pejoratively called "twitspeak").[6]

Esoteric languages like Befunge, which introduced this chapter, seem to point in another direction altogether, opening up a more indeterminate and expressive space and transcending the production of simple effects or predetermined actions. Take, for another instance, the confounding effects of the esoteric Brainfuck programming language, which offers a challenge to normative source code interpretation by consisting entirely of punctuation, with each of the eight characters "><+-.,[]" representing a single elementary operation. "Hello world!" is expressed thus:

```
>+++++++++[<++++++++>-]<.>+++++++[<++++>-]<+.+++++++..+++.>>>++++++++[<++++>-
]<.>>>++++++++++[<++++++++++>-]<---.<<<<.+++.------.--------.>>+.
```

Taking this indeterminacy further still, Brainfuck exceeds the world of computation in Bodyfuck, an interpreter using computer vision techniques to map bodily gestures to the Brainfuck instruction set.[7]

But as with all signifying systems, interpretation still takes place at all levels, even when they are as esoteric as the examples mentioned above. The reader, whether human or machine, is also cast as one of the objects of the software and operating system. The point can be demonstrated with writing more generally as, in word-processing a text (like this), the writer is also processed into the choice of software and operating system that prescribes or allows certain tasks. It was in recognition of this issue that Friedrich Kittler apologized for his choice of software used to write the essay "There Is No Software."[8] The fact that the user's thoughts and actions are somewhat determined by the operating system or graphical user interface recalls the ways the user is interpellated in the Althusserian sense to demonstrate how ideology calls us to order through its God-like commands and procedures.[9]

This follows an understanding of code in a broad sense, insofar as it can be traced back to its etymological roots through "codicilla" (tablets used for inscribing letter forms) and "codex" (the bound book of the law), as Kittler explains in his entry to the *Software Studies* lexicon.[10] For Kittler, the references establish how code can be understood through the twin operations of command and control.[11] But in addition,

and the concern of this book, are the ways in which code also produces ambiguities and possibilities of recoding its prescriptive and deterministic tendencies (the unwritten laws, so to speak). Although instructional, program code cannot simply be reduced to its functional aspects, as it also extends the instability already inherent to the relationship of speech to writing, where it can also go out of control. Like all codes, it is only really interpretable within the context of the overall network of relations that make its operations inherently unstable. It is both a computer-readable notation of logic and a representation of this process, both script and performance; and in this sense it is like spoken words (made explicit in the case of a "hello world" program). It functions in relation to other programs that are simultaneously running in the form of processes that include storage and execution, operating across hardware and software platforms.[12] Indeed, code cannot be separated from the broader systemic framework and the way the technology that inscribes it is embedded in wider processes of command and control.

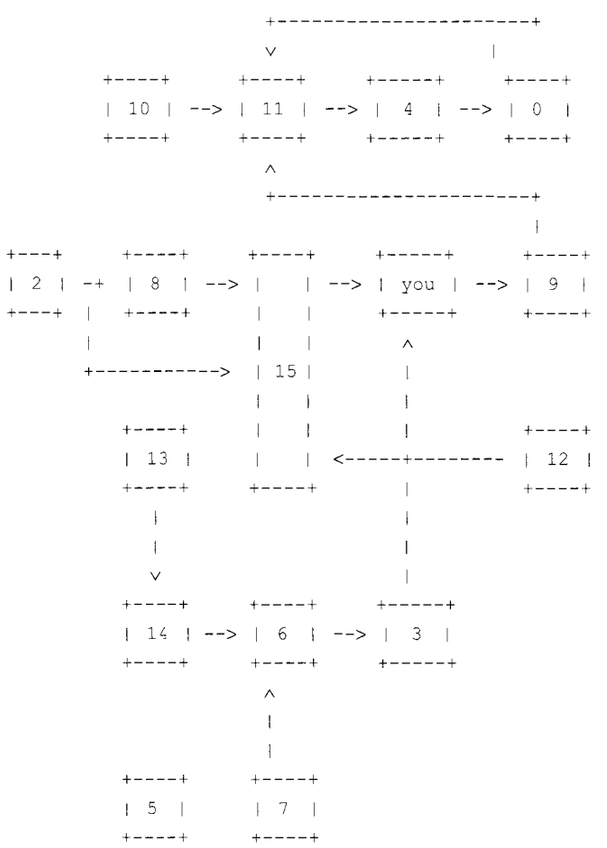
Coding subject

The common declaration "Hello world" *interpellates* in this way too, not least in the dogged insistence on the use of English as the default "mother tongue" of program languages. To Louis Althusser, the speech act constitutes the subject; it recruits subjects by hailing them, "Hey, you there!," as a policeman (sic) might speak to a passerby.[13] Through the act of recognition the subject begins to exist in ideology, in parallel to the way that program code can be seen to exist in ideology too.

In the following example, the program creates a list of numbers, one of which is made to represent the user of the program. The numbers are arbitrarily connected in a graph structure which is visualized for the user, along with the phrase "Hey, you there." An explicit command like this can be seen to reflect the capacity of programs to authoritatively "speak" to subjects, thus occupying the combined realms of subjectivity and sociality. It is in this sense that speech, or having a voice, connects with political expression and allows for a wider understanding of power relations. However, a more complex articulation of interpellation is required than an emphasis on the determining role of communication systems, as neither the human subject nor program code is quite that passive. Like the historical subject who is interpellated to act in a way that is preordained but not fully known, speech is also retroactive: it is speech in-itself, or speech that preexists itself, "speech before speech," as Slavoj Žižek explains.[14] Things are decided before they are enacted in actuality, and in this sense are always ready to be executed.

```
example@speakingcode:code$ cat | perl
use Graph::Easy;
my $g = Graph::Easy->new();

@ids = (0 .. 15);
$ids[rand(@ids)] = 'you';
foreach $id (0 .. $#ids) {
    $to = rand(@ids) - 1;
    $to++ if ($to >= $id);
    $g->add_edge($ids[$id], $ids[$to]);
}
print($g->as_ascii . "Hey, you there.\n");
```




```
Hey, you there.
```


Judith Butler also addresses this in her book *Excitable Speech*, drawing attention to the ways in which speech demonstrates agency, as there is a relation between speech and action, between saying and doing.[15] Spoken words say something and do something, and this has consequences in the world, such as in the way insults exert a form of violence. As linguistic beings we are bound to language as part of the constitution of subjectivity (as Althusser described it), so the call to order starts as a form of insult, and we enter into language antagonistically. Speech announces the action that will follow along with this antagonism, hence its power, but also importantly speech sometimes fails or is bounced back through various attempts to resist its determining effects. A good example of this is found in Mladen Dolar's book *A Voice and Nothing More*, which begins with a description of a failure of the voice to interpellate.[16] A command is given loud and clear by a commander of an army to attack its enemy, but nobody acts. The command is repeated over and over again to no effect, until someone says, "What a beautiful voice!" Refusing to execute the command, the soldiers instead contemplate the aesthetic properties of the spoken expression (the soldiers are Italian and perhaps it is their love of opera that informs their actions, rather than pacifism or fear of dying). The speculation is that esoteric languages similarly shift attention from command and control toward cultural expression and refusal.

Moreover, there is a well-established paradox in such a straightforward view of agency derived from the Althusserian concept of interpellation, with its stress on the determining role of language. If the subject is to some extent constituted in language and code, then to think that someone saying and doing something is a straightforward demonstration of agency misses the point; language and code constituted them in the first place, and as such the formation of the human subject is always an unfinished project. Interpellation seems to be bound up in contradiction in this respect, poised between the subject that speaks and the subject that is constituted through speech. If subjects are constituted in language in this paradoxical way, it follows that they also have the ability to reconstitute themselves through language, and even reconstitute the institution of language itself. Clearly there are other possibilities of agency operating outside the "sovereign autonomy of speech," and this is what Butler refers to as "excitable" states. Her assertion is that "speech is always in some ways out of control."[17]

Put differently, speech is far more distributed and networked than simply emanating from a single body, and the call to order is rather more like being allocated an Internet protocol (IP) address that defines you as a unique user in the network (and the example of code at the end of this chapter tries to demonstrate this point).[18] This also serves to emphasize the social aspect of speech—in that language itself is social and in speaking there is always an "echo of others in the act of speech," as Butler puts it.[19] Moreover, it is not that interpellation doesn't occur but that it operates in ever more subtle ways across complex informational networks, along with the networked forms of subjectivity it partly constitutes. This is what Brian Holmes refers to as the

"double constitution of the subject," generating both machine-like humans and human-like machines, conjoined through closed-loop informational systems.[20] Indeed, computer programs interpellate through the dual registers of command and control—and they do this at multiple levels of operation, and they also tend toward an executable logic that appears predetermined and unchangeable. Yet the program was programmed in the first place. So although on the surface program code appears to operate in a similar sovereign manner with straightforward agency, namely a command to execute an instruction from sovereign code, the argument that unfolds through this book is that in significant ways these operations are also prone to bugs and failure, and in significant ways can be considered to be out of control, like speech.

Although the analogy between program code and speech acts has become rather commonplace since its suggestion by Terry Winograd and Fernando Flores in 1987,[21] the stress here is on the degree of indeterminacy they share, as with the example of live coding, where the writing of the software happens at the same time as performing with the software.[22] Programmers express themselves through the use of program languages, the book suggests, in ways similar to other human communicative expression through language and gesture. They do this through their manipulation of layers of representation, including symbols, then words, language, and notation, as exemplified in the production of software prototypes, artworks, programming languages, and improvised performances that embed the activity of programming in the improvisation and experience of software art in general. On this last point, the practice of live coding exemplifies how the practice of coding, its writing, working, and creative use, establishes an unstable relation to its output. In this sense, although of course code largely determines its output, the broader apparatus including the idiosyncrasies of the programmer provide indeterminate outcomes and help to stress the expressive dimension of software production as a whole.

Beginning with these ideas, the book establishes that program code, like language in general, evokes complex processes by which multiple voices can be expressed, modified, and further developed. With code, clearly problems can be solved in a multiplicity of ways, as evident on the Rosetta Code website, where as many different languages as possible are collected that relate to the same computational task,[23] or on Instructionset, a website where instructions are posted and programmers are invited to carry them out in a variety of ways.[24] Like other collective speech acts, programming oscillates between process and expression. This further resonates in the ways program code opens up broader discussions around the production of meaning and criticism, for it is clear that there are a myriad ways of saying hello in a multiplicity of human and machine languages, and a great complexity in the ways that the human-computer interprets them.[25] Take, for another example, the esoteric programming language Piet which looks like a geometric painting by Piet Mondrian, from which it takes its name.[26]

Like speech in relation to text, esoteric languages that diverge from the conventions of written language seem to stress the point that rendering speech or code as mere

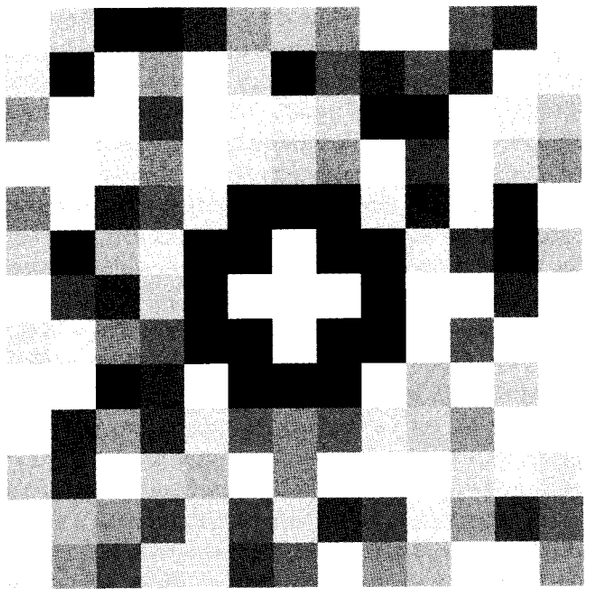


Figure 0.1
Source code written in the language Piet (original in color). Prints "*Hello, world!*" (http://www.retas.de/thomas/computer/programs/useless/piet/explain.html). Image cc by-sa 2.5 license, Thomas Schoch (2006).

written words fails to articulate the richness of human-machine expression. Similarly, Kittler explains that there is something paradoxical about using words to write about program code: "For one, all words from which the program was by necessity produced and developed only lead to copious errors and bugs; for another, the program will suddenly run properly when the programmer's head is emptied of words."[27] The paradox opens up some interesting opportunities for coding practices that challenge expectations of what source code does in itself and what it might do once run. In this way the use of esoteric languages like Befunge, Brainfuck, or Piet somewhat undermine the interpellative authority of the computer and stress alternative interpretations like the paradoxical qualities of speech. Perhaps we can begin to think of these examples as excitable program code.

Coding expression

Some core principles underpin the practices of coding as both procedure and expression. Of course, computers don't really speak but follow prescribed rules of execution,

tasks, and actions. Nevertheless, there is more to coding than simply the demonstration of formal logic, as if everything could be reduced to binary representation at a fundamental level. The conventions of writing and reading, of both text and code, might be considered to be part of coded systems of input and output (abbreviated as I/0), and yet formal logic fails to break the apparent paradox of language. In *The Art of Computer Programming* (1968), Donald Knuth already emphasized the depth of the relationship between writing a computer program and the logic that underpins it—what he referred to as "literate programming"—and so acknowledged programming to be somewhat like composing poetry or music.[28] He also demonstrated how reading could be understood as procedural and reflexive in his "Procedures for Reading This Set of Books." A short extract: "5. Is the subject of the chapter interesting you? If so, go to step 7; if not, go to step 6. . . . 14. Are you tired? If not, go back to step 7; 15. Go to sleep. Then, wake up, and go back to step 7."[29]

The physical form of a text and book, such as the one you are reading (if it still interests you), contributes to the production of meaning in the way that contextual information does in general. The text simply cannot be divorced from the materials and institutions that produce and disseminate it. In *Writing Machines,* N. Katherine Hayles puts it this way: "When a literary work interrogates the inscription technology that produces it, it mobilizes reflexive loops between its imaginative world and the material apparatus embodying that creation as a physical presence."[30] Like M. C. Escher's lithograph *Drawing Hands* (1948), in which two hands draw each another on a sheet of paper, there are many examples of recursion that encourage the author or reader to reflect on their role within the construction of the text, and on the ways in which meaning is constructed. Discussions of this tend to emphasize the link to literary theory (famously explored in Barthes's essay "The Death of the Author")[31] but also to second-order cybernetics, in terms of the way the observer is understood to produce changes to the system being observed,[32] akin to what Gregory Bateson referred to as "recursive epistemology."[33] Informed by systems thinking, Bateson understood recursion to operate on all levels across natural and cultural forms, undermining hierarchical orders of knowledge.

Another useful concept from Bateson, "double description," reinforces the importance of bringing together two or more information sources in order to provide information that is different from either one of them.[34] In a similar way, the combination of formal description and creative action, what might be referred to as *double coding,* is well established in software arts practice and where program code is made "literate." What has become known more specifically as "codework" provides a good example of how the practices of writing and programming can recombine formal logic and poetic expression. Sometimes referred to as pseudo-code, and often nonexecutable, codeworks operate in a similar realm to esoteric programs, and confound any easy assumptions about how meanings are produced prescriptively. Such

examples also serve to undermine the idea that execution is the sole site of interpretation for code.

A double sense of interpretation lies at the core of this, and suggests a broader engagement with critical literacy (literary criticism for computing) that includes code and other written or spoken forms. Indicative of this tendency is Mez's hybrid language "mezangelle" (developed in the 1990s) which simultaneously combines formal structures from code and speech into a kind of creole, thus challenging some of the assumptions of what constitutes communicative exchange between humans and machines, meshing together speaking-working-coding into new forms that reflect contemporary communication systems. Her intention is to expand the ways that meanings are generated beyond the conventions of written language, as explained using mezangelle:

```
2 4m a text fromme the ground[ing] uppe
2 n-hance the simple text of an email thru the splicing of wurds
2 phone.tic[k-tock]aulli m-bellish a tract ov text in2 a neo.logistic
maze
2 network 2 the hilt N create de[e]pen.den[ting]cies on email lizts for
the wurkz dis.purse.all
2 graphi.caulli N text.u.alli e-voke a conscious sens.u.all & lingual mix
2 make net.wurkz space themz.elves in2 a spindle of liztz thru
collaboratori n-tent
2 uze computer kode kon.[e]vent.ionz spliced with irc emoticons and
ab[scess]breviations
2 spout punctu[rez]ationz reappropri.[s]ated in2 sentence schematics
2 polysemicalli m-ploy a fractured wurdset
2 m-brace 4m conventionz
2 flaunt pol[emic]itical l-usions
2 ig.gnaw word endinz
2 let lettahs b used as subsetz
2 x-tend N promote n-clusive meaningz.[35]
```

The example, using a technique referred to as double (or multiple) coding, exemplifies the material aspects of code both on a functional and an expressive level, even if it further confuses interpretation by its nonfunctionality or inability to communicate clearly (which is part of the point, of course). In this sense, nonexecutable (or illiterate) code can also be considered to be a command that fails to interpellate the reader, and thus promotes the notion that all commands are open to failure and errors, just as refusing to act is a provocation in other contexts. So too with obfuscated code contests, where it is clear that program code has an aesthetic dimension that extends beyond the conventions of programming which stress the efficiency and brevity of source code. In taking to an extreme the principle that program code be concise, a

program might also be seen to run in rather unpredictable and amusing ways that confound the expectations of the human and machine reader-interpreter and allow them to rethink their prejudices.[36]

The double description evident in programming, and arguably inherent to it, can open up ambiguities and imaginative feedback loops. Indeed the loop is an important component of imperative programming, indicating when instructions are to be repeated or set to repeat until a terminating condition is met, unless an infinite loop is invoked. Programs are often structured around an infinite loop, known as the event loop; but with the use of infinite loops comes the possibility of infinite growth, which threatens the logical structure of the machine. Strategies like "deprogramming" bring to attention the structures and standardized formats of programs in this way. In the following example, from the website deprogramming.us,[37] a hello world program written in Perl has been modified to repeat itself in an endless loop:

```
#!/usr/bin/perl
# prozac.pl
# it will greet your system to death.
# but you go down in a cheerful endless loop.

while (1) {
          print "Hello World!\n";
          system ("$0"); # this line replicates it.
}
```

A simple example like this reflects the disruptive capacity of critical aesthetics to comment on the operations and effects of computational processes more broadly, working, as Alan Kay has referred to it, at the level of the "metamedium" (both performing universal computation and simulating all other media).[38]

Moreover, there is a danger of overstating the role of code as the source of action unless it is considered as part of a wider set of communicative actions by multiple human and nonhuman agents. Indeed, if code speaks, under what conditions and on behalf of whom?[39] Referring to code as speechlike becomes significant for this reason, as it invokes an established tradition considering speech to be more intelligible than writing, and its importance for critical debate and understanding of action in the world. To Socrates, who allegedly did not write his philosophy but spoke it, the voice is the "unwritten law" pertaining to the moral law, in contrast to the written law. It takes on a kind of authority and authenticity in this way and carries the inner voice of moral integrity.[40] One should not overstate the role of speech either, and clearly it contains its own internal contradictions and inherent paradoxes. Yet the spoken word does appear to haunt all texts as "sound is the natural habitat of language," as Walter J. Ong puts it in *Orality and Literacy*, and even writing needs to speak (if only silently) to reveal its meanings to the reader-listener.[41] Ong also makes a similar

point to Kittler, pointing to the paradox that in expressing ideas about orality, he relies on the written form rather than a spoken performance to deliver the argument. Part of the difficulty in writing something down is that the transformative possibilities of speech are curtailed.

Like executable code, there is something *overdetermined* about all writing in this sense. In "Talking Back," bell hooks stresses the inherent politics of forms of writing that hold on to speech, for both writing and speech can express resistance to forms of power—as with the rejection of capital letters in her name.[42] Is this what some programs do, too, in holding on to the special qualities and paradoxes of speech? More than simply writing, program code is a special kind of writing and, unlike a score that is followed but interpreted, it follows its script quite literally. It holds on to its script and does not let go, but in so doing it also and importantly holds on to the inherent special qualities and paradoxes of speech, its predeterminations and its sense of excess.[43]

Introduction

The purpose of the book is to explore these double descriptions: involving both formal logic and the expressive aspects of coding, its constraints and its excesses. To some readers, it may seem rather unfashionable to concentrate on spoken language as the main referent: why not use mathematics as the main metaphor for analysis, given that the computer works through binary arithmetic—both representing and manipulating numbers in a system of 0s and 1s? Such an approach might also be expressed in the interest in "object-oriented ontology" and its allusion to object-oriented programming, with a return of attention to the discrete object rather than relations,[44] as well as in numerical articulations of politics. In *Number and Numbers*, Alain Badiou writes: "We live in the era of number's despotism. . . . Number governs our conception of the political."[45] Indeed the relation between the human and machine reader comes closer than ever to the operations of a machine performing calculations. However, the book attempts to address this problem the other way around, insisting on a discourse that derives from an understanding of the human condition and of politics that operates through language rather than simply through the economy. Moreover, the problem identified here is the invasion of language by economics. The universal calculating machine has sharpened our understanding of the operations of speech and writing, not least through what Ong calls "secondary orality," through writing technologies that produce scores and scripts.[46] But under the present conditions of financial capitalism, human action is rendered economic and its force is annulled, as it is "expressed not with words but with numbers," as Boris Groys put it.[47] To Groys, the task is to transcribe the world from the medium of money to that of language, so that politics can operate freely in relation to fate rather than being subordinate to the

economy. Speech continues to underscore the human condition, however paradoxical this may appear.

These are some of the initial ideas that account for the book's attention to speech, its relation to the human condition, and its continued importance within cultural criticism. The book is not intended to be philosophy or theory as such. It takes critical writing to be a form of practice, and the many examples of code add to this conviction, not as illustrations but as additional forms of criticism. The resurgent interest in the concept of speech and the voice in recent years has something to do with the perceived neglect of sound in cultural work, no doubt,[48] but for the purpose of the book it is also an opportunity to foreground speech and action in the spirit of Hannah Arendt's writings,[49] and to consider some recent work that identifies how both speech and action appear to have lost their power, not least to mathematical symbols. To put it simply, *speech continues to need a voice that exceeds number.*

These are some starting points for the book and an indication of the broad sweep and eclecticism of its references. The first chapter, "Vocable Code" (co-written with Alex McLean), examines the performative and expressive dimension of programming and provides a number of examples. The aim is to stress some of the instabilities of code that undermine strict determinations of intention and meaning. Accordingly, the chapter looks in more detail at the linguistic analogies between code and natural languages, and the ways these have been understood in terms of meaning production. Speech emanates from the human body; computer programs have bodies too, and various attempts at simulation have revealed the impossibility of duplication. The further link of speaking to thinking machines demonstrates the sophistication of the human apparatus and the enduring complexity of speech, both as a sign of intelligence and as one that requires social interactions. Indeed collective speech acts provide multiple ways of understanding the complexities of code and the performative actions of running code. This is where speech indicates a more open-ended set of procedures that do not entirely compute, and that operate both within and beyond computational processes.

The second chapter, "Code Working," emphasizes a more overt political dimension of working with code, by stressing the ways in which all codework necessarily carries with it the work that has been invested in its production, as well as in the broader apparatuses through which it is served. This partly explains the motivation for many software producers to reveal the source code as an integral part of their work, particularly in the production of codeworks previously mentioned. A number of other examples are introduced, especially work by artists and programmers keen to offer alternatives to mainstream development, ranging from the performances of the live-coding scene to commons-based peer production. These demonstrate that new ideas emerge through wider recursive processes, which reflect the communicative and linguistic dimensions of work and action. If the first chapter established that code was

speechlike, then this chapter further establishes that speech has become more codelike under the conditions of informational capitalism, especially when work is executed through scores and scripts, something that Paolo Virno argues in building upon Arendt's work.[50] Taking Virno's line of argument, the relationship of capital to language also helps to establish how working can be understood as speech acts, facilitated by networked communications technologies and collective formations of work. This underpins the ways in which commodified technologies have appropriated collective speech acts and networked intelligence. It also allows for the possibility of code operating in excess of market forces and, despite the title of the chapter, more in the realm of code action than code work.

The third chapter, "Coding Publics," builds on many of the same references but focuses more on collective action and reconceptualizing ideas of publicness, again making reference to the writings of Arendt. The voice undermines this public dimension by offering a dual representational function, indicating both the precondition for language acquisition and also the expression of political opinion. That the idea of democracy itself is in crisis, along with the voice, is revealed in a range of paradoxes over the terms in use, and also over key issues that relate to intellectual property and the uneasy interconnections between free software and free speech. For Christopher M. Kelty, again referring to Arendt, the free software movement is an example of emergent and self-organizing public actions, underpinned by the sharing of source code.[51] With the free software movement in mind, he introduces the term "recursive public" to account for the ways in which the public is able to reconstitute itself as a public, through the modification of the platform through which it speaks and by addressing other technical and legal layers of operation. This attention to publicness has implications also for the connection to property regimes and the ways that social technologies (especially popular platforms like Twitter and Facebook) tend to encourage ever more voices to be heard but only with restricted registers and effects. The concern is to consider some of the mainstream implementations of social software, to register the conditions under which this is done and the language employed to do so, and to examine the consequences in terms of the production of social relations and subjectivities. Through their reliance on proprietary logic, these developments seem to legislate against a plurality of voices that have unique attributes, which Arendt considers to be necessary for politics. Although there may be more voices speaking, they are effectively made mute. Against popular interpretations of them, the logic of networks and the rhetorical promises of social media are examined as effective mechanisms for the *suppression* of political expression in the public realm, and to reveal how public intellect is becoming ever more privatized through free-market logic. The coalition between consumer capitalism and democracy is evident in relation to certain platforms that claim to allow enhanced participation in the political process, where again the voice is strongly invoked but only in compromised forms. It would seem

that social media not only fail to provide the means for people to have an effective voice but also reinforce neoliberal values, paradoxically through technologies that appear to promote the voice across telecommunications networks and online sharing platforms. This leads to the final section of the third chapter, in which the concept of the speaking public is reimagined not simply in its ability to enter into discourse but in its ability to modify the very platform through which it operates. And yet this capacity to express opinions is bound both to collective action and to the forces of domination, in reciprocal relation. Thus participation and collective forms of political expression become part of the very mechanism of guaranteeing their nullification. Clearly something important is missing.

The final chapter, "Code for-Itself," moves toward a summary of key issues, drawing on Franco Berardi's *The Soul at Work* (already introduced in the earlier chapter on work) to illustrate how the voice relates to material factors emanating from the way capital tries to incorporate both the soul and voice into production.[52] If the soul was once considered the source of speech, now conditions of contemporary production appear to have colonized both, indicating the politics of work to involve a broader set of issues that include speech acts and the communications platforms through which we speak. Voice reenters the discussion, on account of the need to focus on the biopolitical preconditions and the most fundamental aspect of human expression. In this way, the voice stands for the sensitivity of a culture bound by calculation and by neoliberalism's emptying out of expressive language. The ability to voice things is offered, if not accelerated, but is ultimately rendered illusionary by political forces that do not care about the "grain of the voice,"[53] as it lies outside the interests of the market and profiteering.

One of the main problems identified is the way the acquisition of language and the human condition have been largely separated as part of the expansion of market logic. More than simply a series of sounds from the body, a speech act is something that involves the human capacity to think and thereby express feelings. Rediscovering the voice, combined with words as speech, is therefore a necessary part of social transformation in the face of overpowering forces that close down and oversimplify discussion, or reduce action to procedures and behaviors that can be simulated. If human action is compromised by its contemporary expression in numbers, it should also be remembered that even at binary level, in terms of numerical calculations, the computer is surprisingly prone to errors, and certain calculations simply cannot be performed by strings of binary digits (determining the value of pi is a famous example).[54] Further ambiguities arise in computational processes when complex formations are introduced such as "or," "and," "not," as well as the infinite loops already mentioned. Once code is likened to speech, it also provides the possibility of new forms of criticism and practice that combine natural and artificial languages into new speech acts, in which ideas are stated and then reflected upon and restated. For all languages are

a mode of action in this way, and not only a referent of thinking. If coding is an invitation for speech and action—a script to be executed—then the act of coding is a deliberate action across cultural and technological fields. In this way it offers the potential to open up some of the inherent paradoxes of double description. This is a similar point to Groys's assertion that politics needs to operate with language if it is to act freely, and that the critical task is to assert its internal paradox. Under present conditions of capitalism, human action functions as a commodity, that is to say, it is inherently mute.[55]

The example that ends this chapter is a script that attempts to connect to a web server on each of the 4,294,967,296 IPv4 Internet addresses in turn. Where successful, it posts the message "Hello world!" to the server. The script signals other possibilities for speech, both *within* and *beyond* politics.[56] Perhaps programmers need to find their voices in this respect, such that their scripts might begin to announce themselves to the world in ways that remain open to other expressive possibilities and collective actions. By acting collectively, in the echoes of others in the act of speech (as Butler puts it), some of the rules of correct speech can be broken, and meaning can be reconnected to the body.[57] Similarly, babbling, or different forms of nonstandard speech, can extend the political and aesthetic possibilities of speech. This is how the following chapter begins, with reference to sound poetry that disrupts linguistic rules and rejects the authority of the master's voice.[58] Speech is inherently political in this sense, and programs characterized in this way provide ever more possibilities for generating unpredictable results in recognition of a body politic.

```
#!/usr/bin/python
# A script for greeting every server on the Internet.
import iptools, httplib
for ip in iptools.IpRangeList('0.0.0.0/0'):
  try:
    print "Greeting " + ip
    cx = httplib.HTTPConnection("%s:80" % ip)
    cx.request("POST", '/', "message=Hello+world!")
  except:
    pass
```

```
lt.set_piece_hashes(t, '.')
tf = lt.bencode(t.generate())

fh = open('me.torrent', 'wb')
fh.write(tf)
fh.close()

info = lt.torrent_info(lt.bdecode(lt.bencode(t.generate())))

opener = urllib2.build_opener(
    urllib2.HTTPCookieProcessor(cookielib.CookieJar()),
    MultipartPostHandler.MultipartPostHandler
    )

opener.open("http://runme.org/submit/",
            {'act': 'login',
             'email': 'torrentpy@mail.slab.org',
             'password': 'hjrvrs'
             }
            )

opener.open("http://runme.org/submit/",
            {'explicit_project_id': '974',
             'act': 'submit_software',
             'file_software': open('me.torrent')
             }
            )
ses = lt.session()
h = ses.add_torrent(info, "./")

state_str = ['queued', 'checking', 'downloading metadata',
             'downloading', 'finished', 'seeding', 'allocating']

while (1):
        s = h.status()
        print '%.2f%% complete (down: %.1f kb/s up: %.1f kB/s peers:
%d) %s' % (s.progress * 100, s.download_rate / 1000, s.upload_rate /
1000,
            s.num_peers, state_str[s.state])
        time.sleep(5)
```

Notes

0 Double Coding

1. For more on the Befunge programming language, see http://en.wikipedia.org/wiki/Befunge.
2. *The Hello World Collection*, compiled by Wolfram Roesler (with help from many people around the world), includes 421 "Hello world" programs in many more or less well-known programming languages, plus 63 human languages (available at http://roesler-ac.de/wolfram/hello.htm).
3. Referring to John 1:1: "In the beginning was the Word, and the Word was with God, and the Word was God."
4. Jonathan Rée, *I See a Voice: Language, Deafness and the Senses—A Philosophical History* (London: HarperCollins, 1999), 75–76. He is citing a passage from Genesis 11:1–9. Also of relevance here is the constructed language Volapük, created in 1879–1880 by Johann Martin Schleyer, who claimed that God had told him in a dream to create an international language. See Leo Findeisen, "Some Code to Die For: On the Birth of the Free Software Movement in 1887" (2003; available at http://www.monochrom.at/codetodiefor/).
5. Although the relationship between the study of machines and religious thinking is not the concern of this book, it is worth registering that Norbert Wiener identified commonalities in that machines must learn and reproduce in accordance with what he called the rules of the game, a game increasingly set by the dark forces of informational capitalism and the industrial-military complex. See Norbert Wiener, *God and Golem, Inc.* (Cambridge, MA: MIT Press, 1964).
6. "Twitspeak" is the vernacular form of language used with Twitter, such as the use of hashtags (see http://twitter.com/).
7. The Bodyfuck demo shows the physical exertion required to produce a short sequence of symbols (available at http://forum.openframeworks.cc/index.php?topic=2772.0).
8. Friedrich Kittler, "There Is No Software," in Timothy Druckrey, ed., *Electronic Culture* (New York: Aperture, 1996), 331–337 (available at http://www.ctheory.net/articles.aspx?id=74). Matthew Fuller's "It Looks Like You're Writing a Letter," in *Behind the Blip: Essays on the Culture of Software* (New York: Autonomedia, 2003), makes a similar reading through a close analysis of the word-processing software Microsoft Word, with a further link to the politics of work through the Office suite of programs.
9. Louis Althusser, "Ideology and Ideological State Apparatuses: Notes Toward an Investigation" (1969), in Slavoj Žižek, ed., *Mapping Ideology* (London: Verso, 1997), 131. To explain, "ideological

State apparatuses" include the family, schools, church, legal apparatus, political systems, trade unions, communications media, arts and culture, and so on, as distinct from the "repressive State apparatuses," the government, army, police, courts, prisons, and so on. Both function through repression and ideology, but the difference is that ideological State apparatuses, rather than acting predominantly by repression or violence, function through ideology and do so more covertly, such that people respond willingly.

10. Friedrich Kittler, "Code," in Matthew Fuller, ed., *Software Studies: A Lexicon* (Cambridge, MA: MIT Press, 2008).

11. Ibid., 41. More specifically, this is what Kittler refers to as the US Pentagon's imperial motto of C^4: namely, "command, control, communication, computers."

12. Moreover, the program both says something and does something at the same time and in the same place, and thus counters commonly held assumptions about language taking two main forms, in which speech privileges time over space and writing space over time.

13. Althusser, "Ideology and Ideological State Apparatuses."

14. Slavoj Žižek, *The Ticklish Subject: The Absent Centre of Political Ontology* (London: Verso, 1999), 60.

15. Judith Butler, *Excitable Speech: A Politics of the Performative* (London: Routledge, 1997).

16. Mladen Dolar, *A Voice and Nothing More* (Cambridge, MA: MIT Press, 2006), 3.

17. Butler, *Excitable Speech*, 15.

18. Althusser, "Ideology and Ideological State Apparatuses," 131. I have borrowed the idea of the allocation of an IP address from Brian Holmes's conference paper "Artistic Autonomy and the Communication Society," in *Diffusion: Collaborative Practice in Contemporary Art* (London: Tate Modern, 2003).

19. Butler, *Excitable Speech*, 26.

20. Brian Holmes, "Future Map: or, How the Cyborgs Learned to Stop Worrying and Love Surveillance," in *Continental Drift* (2007; available at http://brianholmes.wordpress.com/2007/09/09/future-map/).

21. Terry Winograd and Fernando Flores, *Understanding Computers and Cognition: A New Foundation for Design* (Reading, MA: Addison-Wesley, 1987). Inke Arns also emphasized the analogy in "Read_Me, Run_Me, Execute_Me: Software and Its Discontents, or, It's the Performativity of Code, Stupid," in Olga Goriunova and Alexei Shulgin, eds., *Read_Me: Software Art and Cultures* (Aarhus: Digital Aesthetics Research Centre, University of Aarhus, 2004), 176–193.

22. The TopLap website, dedicated to live coding, also contains an evolving set of definitions and a draft manifesto (available at http://toplap.org/index.php?title=Main_Page).

23. For more on Rosetta Code, see http://rosettacode.org/wiki/Rosetta_Code.

24. For more on Instructionset, see http://instructionset.org/.

25. For instance, Mark Marino, in his article "Critical Code Studies" (2006), expresses "Hello world!" using Lisp and comments on its suitability as the language that was developed for artificial intelligence, thus encapsulating the imagined ability of machines to speak like humans. (Available at http://www.electronicbookreview.com/thread/electropoetics/codology?mode=print.)

26. Piet is an esoteric programming language written by David Morgan-Mar. His explanation of the Piet "Hello world!" program is available at http://www.retas.de/thomas/computer/programs/useless/piet/explain.html.

27. Kittler, "Code," 46.

28. Donald Knuth, *The Art of Computer Programming*, vol. 1, *Fundamental Algorithms* (Reading, MA: Addison-Wesley, 1981), v.

29. Ibid., xv–xvi.

30. N. Katherine Hayles, *Writing Machines* (Cambridge, MA: MIT Press, 2002), 25.

31. Roland Barthes, "The Death of the Author," in *Image, Music, Text* (London: Fontana, 1977), 142–148.

32. Heinz von Foerster, *Cybernetics of Cybernetics* (Urbana: University of Illinois, 1974).

33. Gregory Bateson, *Steps to an Ecology of Mind* (1971; Chicago: University of Chicago Press, 2000).

34. Gregory Bateson, *Mind and Nature: A Necessary Unity* (Cresskill, NJ: Hampton Press, 1979). Thanks to David Dunn for this reference.

35. Available at http://beehive.temporalimage.com/content_apps34/mez/0.html.

36. For a good example of an entry in the International Obfuscated C Code Contest, see an uncompiled C program that is self-referential, reproducing the logic of code as a portrait (available at http://www.Ioccc.org/2000/dhyang.c).

37. Amy Alexander, deprogramming.us (available at http://deprogramming.us/prozac.html).

38. Alan Kay and Adele Goldberg, "Personal Dynamic Media," in Noah Wardrip-Fruin and Nick Montfort, eds., *The New Media Reader* (Cambridge, MA: MIT Press, 2003), 393–404.

39. Here I am paraphrasing Henry Giroux's much-used questions: "Who speaks, under what conditions, on behalf of whom?," in *Living Dangerously: Multiculturalism and the Politics of Difference* (New York: Peter Lang, 1993).

40. Dolar, *A Voice and Nothing More*, 84.

41. Walter J. Ong, *Orality and Literacy: The Technologizing of the Word* (1982; London: Routledge, 2002), 8.

42. bell hooks, "Talking Back," in Russell Ferguson, Martha Gever, et al., eds., *Out There: Marginalization and Contemporary Cultures* (Cambridge, MA: MIT Press, 1989).

43. Georges Bataille, *The Accursed Share*, vol. 1, trans. Robert Hurley (New York: Zone Books, 1991). Excess is elementary to Bataille's notion of "general economy" where expenditure (waste, sacrifice, or destruction) is considered more fundamental than the economies of production and utilities.

44. Ian Bogost, *Unit Operations: An Approach to Videogame Criticism* (Cambridge, MA: MIT Press, 2004), making reference to the work of Graham Harman in particular and what has become known as "speculative realism," also associated with Ray Brassier, Iain Hamilton Grant, and Quentin Meillassoux.

45. Alain Badiou, *Number and Numbers* (Cambridge: Polity Press, 2008), 1. Numbers may have been somewhat repressed in Continental philosophy at the time of Badiou's writing (in French, 1990), but the same cannot be said now. The present financial crisis can perhaps be read as the return of the repressed.

46. Ong, *Orality and Literacy*.

47. Boris Groys, *The Communist Postscript* (London: Verso, 2009), xvi.

48. For instance, Norie Neumark, Ross Gibson, and Theo van Leeuwen, eds., *Voice: Vocal Aesthetics in Digital Arts and Media* (Cambridge, MA: MIT Press, 2010); and even the film *Speaking in Code* by Amy Grill (2009), although this is about getting lost in techno music.

49. Hannah Arendt, *The Human Condition* (1958; Chicago: University of Chicago Press, 1998).
50. Paolo Virno, *A Grammar of the Multitude: For an Analysis of Contemporary Forms of Life*, trans. Isabella Bertoletti, James Cascaito, and Andrea Casson (New York: Semiotext(e), 2004).
51. Christopher M. Kelty, *Two Bits: The Cultural Significance of Free Software* (Durham: Duke University Press, 2008).
52. Franco "Bifo" Berardi, *The Soul at Work: From Alienation to Autonomy* (Los Angeles: Semiotext(e), 2009). Berardi invokes the soul to examine contemporary productive forms that put the soul to work (in a materialist sense).
53. The "grain of the voice" is what Barthes calls the individual "voice-magic," imparted by the "very precise space of the encounter between a language and a voice." Roland Barthes, "The Grain of the Voice," in *Image, Music, Text*, 181. The music industry tries to commodify this "grain."
54. It is interesting to note that a function to calculate pi can be written; the issue is that it would never return the value. If running indefinitely, a Turing machine would be able to output it all, as it has infinite memory.
55. Groys, *The Communist Postscript*, xvii.
56. Nick Couldry, *Why Voice Matters: Culture and Politics after Neoliberalism* (London: Sage, 2010), 3.
57. Echoed in the phrase "words made flesh," the title of Florian Cramer's *Words Made Flesh: Code, Culture, Imagination* (Rotterdam: Piet Zwart Institute, 2005), 125 (available at http://pzwart.wdka.hro.nl/mdr/research/fcramer/wordsmadeflesh/).
58. "His Master's Voice" is the famous gramophone trademark, based on an 1899 painting by Francis Barraud, showing the dog Nipper listening obediently to his owner's phonograph (circa 1930).

1 Vocable Code

1. Drawing upon Jonathan Rée's *I See a Voice: Language, Deafness and the Senses—A Philosophical History* (London: HarperCollins, 1999), we previously speculated whether code could be seen to work in a similar way to poetry by acknowledging the conditions of its own making—its *poiesis*. Geoff Cox, Alex McLean, and Adrian Ward, "The Aesthetics of Generative Code," paper delivered at Generative Art 00, international conference, Politecnico di Milano (2000; available at http://www.generative.net/papers/aesthetics/index.html).
2. Geoff Cox, Alex McLean, and Adrian Ward, "Coding Praxis: Reconsidering the Aesthetics of Code," in Olga Goriunova and Alexei Shulgin, eds., *Read_Me: Software Art and Cultures* (Aarhus: Digital Aesthetics Research Centre, University of Aarhus, 2004), 161–174.
3. Franco Berardi read the source code of the virus at the D-I-N-A (Digital Is Not Analog) digital art festival in 2001 (available at http://www.digitalcraft.org/iloveyou/loveletter_reading.htm). Another example that will be mentioned in chapter 2 is radioqualia's *Free Radio Linux* (2001), in which the source code of the Linux kernel was webcast over the Internet using a speech synthesizer. (See http://www.radioqualia.net/documentation/frl/index.html.)
4. Kurt Schwitters, *Poems, Performance Pieces, Proses, Plays, Poetics*, ed. and trans. Jerome Rothenberg and Pierre Joris (Cambridge, MA: Exact Change, 2002), xvii.

5. Excerpt from the introduction to the *Ursonate* by Kurt Schwitters (1932), *Merz*, 24 (available at http://www.ubu.com/historical/schwitters/ursonate.html).
6. We know this from Ferdinand de Saussure's lectures, later published as *Course in General Linguistics* (1916; available at http://www.marxists.org/reference/subject/philosophy/works/fr/saussure.htm).
7. Ibid.
8. This line of thinking is exemplified by Fredric Jameson in *The Prison-House of Language: A Critical Account of Structuralism and Russian Formalism* (Princeton: Princeton University Press, 1972), where he looks for the structures of consciousness itself.
9. This is something that Tzvetan Todorov explained in *Grammaire du Décaméron*: "Every work, every novel, tells through its fabric of events the story of its own creation, its own history . . . the meaning of a work lies in its telling itself, its speaking of its own existence." Quoted in Terence Hawkes, *Structuralism and Semiotics* (1977; London: Methuen, 1986), 100.
10. Noam Chomsky, *Syntactic Structures* (1957; The Hague: Mouton, 1972), 106.
11. Douglas R. Hofstadter, *Gödel, Escher, Bach: An Eternal Golden Braid* (1979; London: Penguin, 2000), 495. He also uses the example of Escher's hands drawing hands already mentioned in the introduction. A further example is *prozac.pl*, also cited in the introduction.
12. In the older tradition of computer art or generative art from the 1960s and 1970s (as distinguished from a more contemporary understanding of software art), Max Bense's theory of "generative aesthetics" was influenced by Chomsky's work on generative grammar. Max Bense, "The Projects of Generative Aesthetics," in Jasia Reichardt, ed., *Cybernetics, Art, and Ideas* (London: Studio Vista, 1971); available at http://www.computerkunst.org/Bense_Manifest.pdf.
13. N. Katherine Hayles, *Writing Machines* (Cambridge, MA: MIT Press, 2002), 6.
14. Friedrich Kittler's observation is much quoted: that to understand today's culture requires knowledge of a natural language and an artificial language. In a similar vein, Douglas Rushkoff, in *Program or Be Programmed: Ten Commands for a Digital Age* (Berkeley, CA: Soft Skull Press, 2011), argues that in order to participate fully in contemporary democracy we all now need to learn to program, to avoid being programmed.
15. Friedrich Kittler, "There Is No Software," in Timothy Druckrey, ed., *Electronic Culture* (New York: Aperture, 1996), 332.
16. Ibid. Kittler is referring to his own previous work, "Protected Mode," in Ute Bernhardt and Ingo Ruhmann, eds., *Computer, Macht und Gegenwehr. InformatikerInnen für eine andere Informatik* (Bonn: FIfF, 1991), 34–44.
17. This ideological distinction will be returned to in chapter 3, which further explores the paradoxes around technology's social potential.
18. Eric S. Raymond, *The Art of UNIX Programming* (Boston: Addison-Wesley, 2004), 8. It should be noted, however, that Raymond's idea of openness does not stretch beyond UNIX as he has made himself gatekeeper of *The Hacker's Dictionary*, changing the very definition of programmer culture to reflect his own free-market ideology. (Available at http://www.ntk.net/2003/06/06/.)
19. Raymond, *The Art of UNIX Programming*, 25. The phrase alludes to Einstein's soundbite, "Everything should be made as simple as possible, but no simpler."
20. Florian Cramer, "Exe.cut[up]able statements: The Insistence of Code," in Gerfried Stocker and Christine Schöpf, eds., *Code: The Language of Our Time* (Linz: Ars Electronica; Ostfildern-Ruit: