

soap box software | permute

- 1
- 2 code
- 3 zines
- 4 diy
- 5 crap hound
- 6 dissent
- 7 temp slave
- 8 print matter
- 9 print function
- 10 project
 - knuth
 - syntax tactics
 - office stationary
- 11 gnu
- 12 mythical man-month
- 13 wwwwww
- 14 comments

1

this text deals with two things: code and self-publishing. self-publishing can be almost anything -- handmade snail mail spam. the focus of the essay is on zine culture of the 1980s and 1990s. the utopian print of those years was an afterthought of consumer computing. leaving it at that is boring. let's reverse that: does utopian print think of consumer computing? i will attempt to tackle that question by way of code. can code be utopian? is code self-publishing?

2 code

i think of code as consumer text. its purpose is to give us the illusion of agency regarding the technologies upon which we have come to depend. if the internet was invented so that banks could erase their end-user support department, then we can always daydream about the promises of free software.

in this essay, code is the textual aspect of computer technology that may be loaded up on a text editor and easily changed. i'm consciously avoiding any discussion on the subject of text editors -- jot down your edits on a piece of paper and then write them to newfile with echo. what's important to me is that this be easily accomplished. with enough time and energy, anyone could write interesting code. the best project would be if my grandmother took the time to re-write the linux kernel from scratch, and if she kept a record of her reflections about code. as much as i like pierre menard, this is not feasible. the processes of code should be manageable without the need of resorting to too much external technical support. this means that code is relative. what's code for some will not be code for others.

when it comes down to it, this means that code is text written in one of the computer languages. code is the active practice of altering half-understood text files. code are the static characters that silently stare back at you, and that will not even you give the illusion that you're double-guessing a machine. it's always evident that someone else was there before you, and that that person was sloppy. it's just a matter of playing along and locating those three characters in a text file that will make all the difference on whether your computer can display postscript or whether it will keep that as a secret to itself. the code is as flawed as the coder.

binary. a piece of binary would be code as long as it's isomorphic to something that is not binary, and as long as the isomorphism is manageable for consumers like us. again, it's a question of it not getting too out of hand. i should be able to predict the changes in the binary by editing the nonbinary, or similar. though i will not speak about things like data flow programming, i wonder how of this essay is extensible. what would it mean to start editing pd patches in vim?

3 zines

seldom does the small press have any real political weight. it's very nice when it does, but most of the time it's just poetry by way of fragile, limited, utopian a4s. the small press may be weak, but this is no reason to get rid of it. likewise for code. a lot of free software activism may be hot air, but this is no reason to ban floss practices. the code of free software is poetry way beyond high brow computer science. some people have pointed to its infinite reproducibility. some parts of this essay deal with its glitches.

there are two things that interest me about the zines of the 1980s and the 1990s. not all utopian publishing is like this, and it's not my intention to define exactly what happened in those publications. it's more like i want to see how these tendencies could evolve:

A zines were media whose design was self-explanatory
B zines were media used to dissent and critique

B is a rephrasing of underground, and A is a rephrasing of diy. A means that zines wore their design on their jacket. i'm using 'design' in a perhaps idiosyncratic way that includes not only layout but also manufacture and distribution. in their manufacture, there was little or no professionalization, and their distribution was at times more like a conversation rather than a monologue. the next two sections go into more detail about A.

4 diy

these days i tend to seek disconnection rather than connection, but back then the design of some zines went the other way around. sometimes connection was pursued as a goal by itself. address listings were ubiquitous. three of the sources i consulted took care to include them. factsheet five's 'the world of zines', v.vale's 'zines!', and fred wright's 'personality on parade' all have them. open karel marten's drukwerk on page 18 and imagine that these listings are typeset with the selectric's 6 pt classified news bold.

part of the strength of this type of design lay on its minimality. page 0322 of ovo (in its open office remediation) had as much weight as anything else. for me, the poetry of zines lies in this unassuming quality. they were transitory. there was no intention of permanence or institution. for good zines, it was always a good time to leave.

there's no need for depression in seeing zines as concocted by xerox-parc, adobe and apple. a theory regarding this should be written, based on fiction as well as fact. it should stress that zines were a tactical appropriation of the digital white cube office of the 1990s. in this, there's an intersection with code. code is the tactical appropriation of computational household appliances.

5 crap hound

both photocopiers and photocopies may be used as publishing tools. more than other media, photocopies invite photocopying. their learning curve is small. xerox collages are self-explanatory, and anyone can do them.

the zine ovo of the late 1980s approached this in an interesting way. it described itself as ``a magazine published on an irregular basis introducing new works into the public domain.'' in this respect, it's now is available not only in pdf format, but also in open office ``source.''

another zine that thought of itself as a tool was crap hound. it was originally published by sean tejaratchi between 1994 and 1998. it was taken up again in 2005. tejaratchi's original intentions were entirely pragmatic. it was to be merely an ``encyclopedia of clipart'':

i was getting paid corporate wages at adidas [...] and i wanted to do something worthwhile with my earnings [...] if i was going to do a zine, i wanted it to be something useful and relevant. i started calling it clip art, but it's changed--my motives are now officially different.

the zine functions as a catalogue of re-usable clip art samples. it's also a collection of collages that sample someone's floatsam of found black and white imagery. it even functioned as a pre-digital distribution medium for fonts:

i've been making fonts for a while, and i've been putting a few in each issue of crap hound [...] [they] are a throwback to dover books. not everyone wants to use a computer. there's nothing wrong with scissors and glue sticks.

6 dissent

it's nice to write about zines in the past tense. there's the possibility that there will be some nostalgia in the resulting text. it's like some time has passed since factsheet five became no more. in the short text 'the society of the unspectacular', this passage of time has meant an acute growth in self-mediation. in relation to that essay, in this section i want to speak about point B of the previous 'zines' section, namely zines as a medium for dissent and critique:

B zines were media used to dissent and critique

i have an affinity for paranoia -- it may be an artistic practice. it's a constant struggle in my head to keep hatred at bay. in 'society of unspectacular' there's a 'current excess' of self-mediation and what's getting shown in those media are four things: xenophobia, racism, hatred, and paranoia. i couldn't agree more. it's a natural side effect of open channels. and at least hate may be a productive aspect of dissent.

one danger that kluitenberg sees in paranoia is that it may lead to disinformation. i think that disinformation and confusion may be tools. the stuff regarding 9/11 that he mentions was actually orchestrated by a small group digital tree huggers as a countermeasure. kluitenberg's arguments can easily be morphed into dialing the police because the self-mediating neighbours are too loud. the paranoia can also turn into insomnia, and in some nights i wasn't able to sleep thinking that 'society of unspectacular' was being used to close down some sort of poetic medium in some place somewhere.

in other more delicate situations paranoia is just common sense. sometimes it may be better to fall prey to its apparent paralysis and disappear. hopefully disappearance is still possible.

less polemically, i would add the possibility of readership to kluitenberg's analysis. ten years ago, the adage was that only 0.1% of zines were any good. i like some of the stuff in the zine ovo a lot. some of the stuff in it makes me cringe. the crap hound zine is based on the premise of sorting through the crap. in general, i think that in the 'current excess', filtering dissent is an interesting design problem. like zines and the small press did, how can design articulate dissent? how can design help to read through the 'current excess'?

i think that it's naive to link 'public discourse breakdown' and self-mediation. media or no media, agency has a natural tendency to remove itself from the public sphere. for many years, the radical aspects of the left to right gradient have known that in order to gain some agency, they have to position themselves outside of the domain of visibility. in this they're not even innovating. when it comes to decision making, the faces of those whose opinions really matter are most of the time unknown.

7 temp slave

i have no idea how to start articulating an answer to the question of the previous section, but the essay can't end here. a good way of carrying on may be to look into a text from the zine 'temp slave'.

it may be said that one of the premises of that zine was a tactical approach to temporary, odd jobs. from it, i will take the cue for the tactical approach to coding of the next sections. the text at hand is called 'temporary insanity.' it proposes detailed, cold, detached classification as a way to not go crazy inside a white cubicle:

emotionless action and emotionless reaction is the solitary buffer between your sanity and your soul [...] if you forget this principle, you begin to revel in the act and the outcome of the organizational process.

post-its get stacked according to ``the laws of diffracted light as exhibited by the rainbow''; productivity is increased by meticulously re-organizing the file cabinets; the filesystem of the computer is cleaned and the temporary worker hides behind the monitor playing the version of tetris that was found; etc.

the text ends on a sad note. the temporary worker forgets the principle of detachment s/he spoke about, and tries to see some sense in the inventory. it's fitting that the arbitrariness of the encoding starts to manifest in front of the computer:

you despise the random pattern of stars and have done away with the frivolity of screen savers. the screen of your computer terminal is clear and black.

a few lines after the temporary worker has a temporary crash. likewise, this would be a good point for my computer to crash.

8 print matter

i would hope that all of this would at this point in the essay lead the reader to re-consider what print matter is or could be. for the rest of the essay, print matter is something that manifests itself through or inside of computers. among other things, this means that print matter is no longer fixed. it becomes harder and harder to say that you will set something on paper.

let's imagine the following situation. there's a hypothetic perl script S that when run prints the following to standard output: ``hello world!`` for me that script is print matter. what i mean to stress with the example is the fluid, variable, unstable form of print matter. print matter is text that at some point in time manifests itself on or through the computer in some way.

this is not as strange as it seems. a similar transformation happens every time a browser renders an html document. under this light, dynamic pages are work that exploits the variable nature of a script's output.

these are easy arguments. print is re-mediated by digital media or whatever. to that i would add that some aspects of digital media are re-mediations of print. let's briefly look at perl.

9 print function

the idea of print that i'm trying to push feels very natural, at least to me. this is the reason why i'm surprised to see how much in common it has with the usual print functions of the computer languages.

the perlfunc man page says that 'print prints a string or a list of strings.' that is, print and strings are intimately tied in perl. with regards to strings, the introductory book 'learning perl' says that strings are sequences of characters, typically plucked from the ascii 32 to ascii 126 range. this means that printing in perl is very often made up of the same thing that text is made of.

but then there are things that we would have never thought of printing had it not been for the print function. browsing through the index we get at least two ideas, printing time and printing databases:

3.8. printing a date

11.11. printing data structures

print function, 29

databases records and, 227

perhaps more interesting is the possibility for a text to be empty, and for there to still be printing. printing and text can be null. null is as natural as whatever: trees, squirrels, bees, and null: 'the shortest possible string has no characters' (learning perl, page 22).

but basically the idea of print in perl is to show the world your stuff, much like print matter is used. in print culture, this happens on paper, most of the time. likewise, in perl theres a standard medium for print namely the so-called 'standard output'. this often means an aggregation of linebreaks: a terminal.

write something about text processing: reading a file backwards by line or paragraph, trailing a growing file, randomizing all lines.

10 project

the previous sections have been dealing with what could be called generative printing. as such, this zine that this essay is printed on has been describing itself. this zine contains within itself a description of its syntax. the specification is included in the cover price. this zine is syntax at the level of the code that has been written, and at the level of the pseudocode that you're reading. for the purposes of the next few sections let's idiosyncratically consider the syntax of pseudocode.

knuth

if knuth talks about those books that precisely describe how they're made, then the final project isn't one of them. it belongs to the subset of books that sort of precisely describe how they're made. it also belongs to the subset of books that are a particular case of a specification that they describe. knuth's books are probably very pretty, but i don't know whether they're accessible outside of his cult. besides, there's little value in a precise description. for me, the code is as precise as i would like to get. as i've already mentioned, everyday we work with chunks of code that aren't always entirely, precisely understood.

syntax tactics

in order to start coding, consumers dont need to wait for a standards body to define a syntax. if we're already coding, i don't see why we shouldn't start writing our own syntax. this means that there's room for both [1] syntax that's functional from an engineer's point of view [2] syntax that's not functional from an engineer's point of view.

on the one hand, [2] above means code that crashes. on the other hand, it also means that we can start looking for computations outside of computers. on the city like socialfiction did, or at a temp job like tempslave did. this also means that the languages will proliferate. how far will the incompatibilities get?

in the final project, functional code is being used to typeset this. a possible direction to explore would be the actual writing of a functional syntax for a publication. i really wonder whether this is possible or meaningful. what publication would require its own software? the only example that i can think of is the original indymedia of seattle.

the final project includes an open call for pseudocode, pseudosyntax, pseudospecifications that still needs to be written. the final project so far includes: [1] its own description [2] the .walk font specification [3] the postscript versioning pseudoscript.

towards an office stationary classification if it's not too presumptuous to say, the final project is research about the code of an a4. this happens at two levels: [1] at the command line. there's code that produces a4 documents, and there's code that sends these documents to the printer. [2] at the printer. this is an office setting. this is where the folding and the stapling of the zine takes place.

in typewriters, printing acts like a screen in that it displays what's typed. unless it's imagined, a typewriter has no random access memory, no disk space, no processor memory. physical traces are left on the typewriter stationary. the erased characters are all there. teletypes received instructions and displayed computations by printing. at the shell, which belongs to the teletype family tree, printing has been abstracted with stuff like lpr. on youtube theres a video where a frustrated office worker starts photocopying the computer's screen directly after the printer sprays him with toner.

could we start coding the office stationary? could we start coding the different paper-printer-staple-computer arrangements in an office or on a desk? calculating their permutations is trivial. how much could we calculate by moving stuff around a desk? the installation aspect of the final project is related to these questions.

in the user manual of the printer where this was printed, there are the following specifications that may come in handy to the end-user:

- model name
- print speed
- acoustic noise
- weight
- package weight
- external dimensions
- emulation
- ram
- supported sizes of paper
- paper smoothness

in temp slave, the insanity manifests itself by classification. in youtube, it becomes anger directed at the printer. it gets thrown out the window, clubbed with a baseball bat, shot at.

11 gnu

in the gnu coding standards, the terms documentation and manual practically mean the same thing.

specifically, a manual is a texinfo file, and documentation includes manuals and other things like NEWS files and change logs. in practice, the latter term practically takes the place of the former.

the people of gnu distinguish between the way in which a program was built, and the way in which it's used. they warn about modelling the documentation after the software:

programmers tend to carry over the structure of the program as the structure for its documentation. but this structure is not necessarily good for explaining how to use the program [...] learn to notice when you have unthinkingly structured the documentation like the implementation, stop yourself, and look for better alternatives.

the coding standards advise authors to approach users pedagogically, thinking about ``the concepts and questions that a user will have in mind when reading it.'' what's more, the manuals that they write should admit two types of reading: tutorial and reference. it's interesting to note that by tutorial they mean something that a user may want to read straight through.

regarding unix man pages, gnu suggests that `-if` adequate- `help2man` be used to extract a man page from the texinfo file. unlike man pages, gnu manuals should have a ``coherent topic''. as an example, the coding standards note that `diff` and `diff3` are both covered in a single manual, whereas there are two man pages, one for each command.

12 mythical man-month

according to the mmm, software has two faces, both as important. the code speaks to the machine, and the documentation ``tells its story to the human user.''

in contrast to the approach of gnu, where -in the guise of documentation as tutorial- writers are asked to provide for readers who ``know[s] nothing about the topic,''' brooks assumes that all users are acquainted with the software. for him, there are three types of users:

the casual user of a program, [...] the user who must depend upon a program, and [...] the user who must adapt a program

brooks attributes the poverty of most documentation to the difficulty inherent in trying to keep any two pieces of data in sync. in particular, it will be difficult to keep the ``two faces'' in sync. in light of this he proposes self-documentation:

the solution, i think, is to merge the files, to incorporate the documentation in the source program.

he points to three levels where he could see this happening: [1] ``labels, declaration statements, and symbolic names [...] convey[ing] as much meaning as possible to the reader''; [2] ``use space and format as much as possible to improve readability and show subordination and nesting''; [3] ``insert the necessary prose documentation into the program as paragraphs of comment.''' before proceeding, i should mention that gnu follows brooks in self-documentation. what they deem to be documentation is included in their source code distributions.

13 wwwwwwwww

we could work with the format of software documentation. in brooks' second point, the jump to ascii art is implicit. it could be said that the pre tag goes towards abiding by that point. however, as this well-known work <http://wwwwwwwww.jodi.org/> does we may choose to ignore these code display conventions. in it, the ascii art looks like noise because the whitespace has collapsed.

i like jodi's approach because it uses obscurity at two levels. on the one hand, it looks obscure -- some people mistake their work for a computer failure. on the other hand, it points to how arbitrary and obscure brooks' conventions can be outside of software engineering.

14 comments

it's strange to think about how some of these coding conventions may have shaped me. it's as if some of my history follows those conventions.

if they've not been modified, timestamps tie a file to a moment in the past. it's an issue of compatibility how far back in time we can go. comments say a lot concerning the relationship between the coder and the code. the coder's ideas about the code get filtered into the comments' prose. what sections of the file did s/he single out for commenting? what's the tone of the commentary? was there time aplenty to develop the project? or not? what's the coder's ascii style? what's the variable's name? when reading own code, the answers may be melancholy. code was written on days and in cities, for particular purposes, in specific machines and languages. my own personal history may start with a child and a 'scientific' calculator.

it would not be smart to write an entire essay on code aesthetics. it would not be smart to entirely avoid the subject. the world (and the other ninety lines) may be falling apart, but theres the possibility of complete description at the inane level of ten lines of code. it may be that those lines completely account for a particular process somewhere in some way at some point. more than i care to admit, i do enjoy seeing how other people come to grips with that.

the catch to code aesthetics is that it's entirely formal. where is that 'somewhere'? in what 'way'? what's the point? i don't know. people seem to be writing their own formalisms underground, utopically.