```
#!/bin/bash

root="thesis"
echo "${root}/code"
becomes=("poetic" "elegent" "abstract
   ")

for path in "${becomes[@]}"; do
    echo "  ${root}/code/${path}/"
done

echo "${root}/syntax/"
echo "Robin/"
echo "  Opheij/"
echo "Running Out of Memory"
```

April 2025

ArtEZ University of the Arts

Bachelor Thesis

Design, Art, and Technology

Poetic Potential

Poetry is often described as the language of emotion, characterized by its ability to condense meaning, and for creating rhythm in structured form. Poets adhere to constraints when writing haikus, and sonnets, the Oulipo literary group a gathering of mostly french-speaking writers and mathmaticians who expiremented with other self-imposed constraints.

If poetry is fundementally
about form, rhythm,
feeling, and interpretation
could programming evoke
poetic ideas and feelings?

Robin Opheij

Like natural language programming has rules, the need in programming for human-readable text over machine code draws inspiration from indo-european language development in structure and syntax, programming had been a western culture rooted development and thus the way we program will come from these cultures, this is being challenged within the field of **Esoteric Programming Languages (esolangs).**

Code can exhibit forms of conciseness like minimalist poems and written with self-imposed or computer forced constraints. Programs can generate inifinite results or endless streams of predefined structured text even used in procedual poetry. Monostich-like single lines of code creating landscapes of new forms behind it. Code like poetry is cultrually significant and has an impact on the community that formed around it.

Austrian philosopher and logician Ludwig Wittgenstein argues that language derives from its use, and not its inherent value of words itself. In code and programming, we can have the argument that code has either beautiful textual qualities or that it needs to be executed to complete itself and make it truely something poetic. English mathmatician Ada Lovelace often considered the first computer programmer viewed her work as a "poetic science"[1] a practice where logic, creativity, and imagination merge togheter. I hope to argue that this is the case for programming as well with code invoking logic, creativity, and imagination in the programmer when they are working on their craft, and thus making code, and programming at least a poetic science.

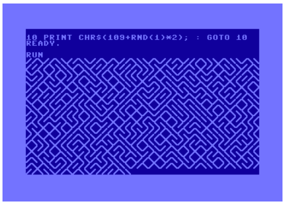
Modern Theorist like Canadian media professor Wendy Hui Kyong Chun, and american poet and digital media professor Nick Montfort suggest that code should be read as a cultural text, same as we do with any other literature. When we read code as a literary text maybe we can find an elegancy otherwise overlooked, not on that is focussed on a technical quality, but one that is based in human expression, looking at code from a aesthetic dimension and to look for a feeling or soul in just the textuality of code.

1. Betty A. Toole, Ada, the Enchantress of Numbers. Strawberry Press, 1992. https://openlibrary.org/books/OL22619654M/Ada_the_enchantress_of_numbers, pp.134

———————— Minimalism and Conciseness

Minimalism in programming is often found in conciesenss, and efficiency. a simple elegance in its inginuity. The UNIX philosophy emphesises doing one thing well, and this thing being simple, compact, clear, and extensible code. This idea also extends to programming languages like LISP where every element is meaningul and needed. The pursuit of efficiency and clarity often leads to code that not only serves some utilitarian function but carries something beautiful, much like a well-crafted poem, the programmers challenge is to convey or computationally express complex problems, or ideas within a limited space. Rather than blocking a creative act, these restraints help the programmer be more decisive in their writing.

The single line of BASIC code **10 PRINT CHR $(205.5+RND(1)); GOTO 10;** shows how minimal code can be whilst generating a vast landscape of an infinite maze.



```
10 PRINT CHR$(109+RND(1)*2); : GOTO 10
READY.

RUN
```

Figure 1: Image of what the line of code generates in the game console Commodore 64 and ran

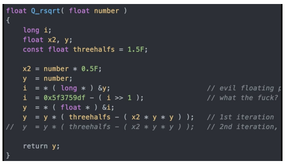
This piece of code when ran on the game computer Commadore 64 would use two very simple characters the forward-slash and backward-slash to create unique structures on the screen. Every new execution would lead to a new completely unique maze to be created. The monostich one liner with unpredictable variations had Montfort et. al. do a close reading on this line due to its ability to be simple enough to easily explain each

aspect, and perfect for explaining
their views on how we should interact with code from the perspective of Platform Studies and Software Studies. In ”10 PRINT CHR$(205.5+RND(1)); GOTO 10;” Montfort and his colleagues do such a close reading talking about how when code is treated as a text we can discover other qualities about it, what it has done to culture, or how it existed and was created by culture. The book talks about whilst the one-liner is a historically located object to study, its not unique or even rare for that matter. It was simple included with each device sold making it a commonplace text. Yet it resonates with people. In the demoscene community, scholars, the hobbyist of the time, and thus even a book.

American artist and Alma mater of the University of Cincinnati Massachusetts Institute of Technology, and dutch, belgium artist duo JODI explore these minimal, rule based expressions in code and art. Reas's work involves the making of generative systems that trough simplicity create great imagery, providing another look into how simple ideas can make creative results, creating a form of poetic dance between the programmer and compiler to create something with as little as possible. JODI quite the contrary plays with the missuse of code to create deliborate chaotic outputs, akin to a form of data dadaism, questioning what the digital realm is and how we interact with these structures. With works like their atomic bomb, pruposfully hiding the entire work in its source code creating the whole work a textual feast for the unseen eye. This work forces us to look at what is moslty thought of the purely utilitarian text of a website (its source code) but then has us thinking about it, and what it means to us. Its layered and subjective, much like poetry, it is something that has to be read and examined before it is understood or felt.

In Algorithmic poetry, a few lines of well-crafted rules/code can perform a vast and unique structure of new sentences, which are in turn poetry. The haiku, a rule-based poetry form that forces us to think on the written lines because of its conciseness and need for direct yet vague language has the ability to create imagery in our minds just the same. In code constraints can excist in the form of a lack in memory, or processing power making the programmer have to become creative in their poetic science. **The Fast Inverse Square Root (FISR)** algorithm is a good example of this.

The FISR algorithm first found in the source code of *Quake III Arena*, is an function/algorithms elegant in its condensed-idnes a creative trick which saves many computationally heavy executions from happening providing the developers with the ability to make for its time realistic 3D games at an acceptable FPS (Frames Per Second) yet maintaining the accuracy those heavy computations would yield.American computer scientists Chris Lomont and Charles McEniry provided another detailed analysis in this function specifically, its beauty uncovered from how it used smart bitwise operations on floating-point numbers transplaced as long with a "magic num-



```
float Q_rsqrt( float number )
{
    long i;
    float x2, y;
    const float threehalfs = 1.5F;

    x2 = number * 0.5F;
    y  = number;
    i  = * ( long * ) &y;                       // evil floating p
    i  = 0x5f3759df - ( i >> 1 );               // what the fuck?
    y  = * ( float * ) &i;
    y  = y * ( threehalfs - ( x2 * y * y ) );   // 1st iteration
//  y  = y * ( threehalfs - ( x2 * y * y ) );   // 2nd iteration,

    return y;
}
```

Figure 2: A screenshot of the function found in the source code of *Quake III Arena*

ber"[2] (0x5f3759df), what is also often mentioned and even debated online is its inclusion of another programmer adding the commen "what the fuck"[3] a glimps of humanity in this technical and precise function showing us that even colleagues of the creators were baffled by this technique. The debate usually excisted along the question of the crude comment should be inlcuded in mentioned of it. I believe it should it adds just that touch of real people wrote and interacted with this. The **FISR** algorithms truly encompesses the idea of "poetic science" in use and why programming is part of it or even exists completely within it. Merging both creativity, imaginativity, and logic together to create such a neat, condensed, and efficient piece of code that to computer programmers looks so beautiful (and I hope maybe to you too). [4]

———— Context and Interpretation

Lovelace's concept of poetical science suggests that programming is not purely utilitarian, but a creative interperative act where mathmatics are used not only to solve complex problems, but to explain concepts hard to be found by language. Lovelace saw science as an expressive medium, not rigid in its ways a perspective reiterated in the academic subfield of software studies

2. predefined values built into the computer, common in computer science literature.

3. comment in the source code of the **FISR** algorithm on line 9

4. The first found use of this function, or almost the **FISR** function was in the comments of a fdlib library file where 2 computer scientist mention an unpublished paper with this algorithm as main topic.

today. Montfort's work on platform poetics in "Platform Studies" explores how code's meaning will shift depending on the hardware its exposed to or even other software it will interact with or is built upon, or even its own execution. He argues that code is not something self-contained, it is embedded in a certain technilogical and cultural enviroment. Code is more than a set of instructions its a performative text that performs what it is asked to do but does so by interacting with everything in its enviroment and does show because of the instructions set by the culture of the creator. Montfort's idea of "platform" is thus not only about technical infrastructure, but the ideas, morals, fundementals, and cultural context in which its created. Much like poetry which is very different based on the culture of the poet, school of thought, and personal quibs. Chun deepens this thought process in "Programmed visions" where she also writes about how not only how we affect code and how software not only shapes how we interact with technology but how we interact with the entire world around us. Adding also how we can learn to understand such a technologically advanced and imbuid world. Chun also engages with the ideas of how/ and when algorithms are used to construct an identity in our digital age in "Algorithmic Authenticity". Opening the discussion of how code has a potential poetic experience. Her argument in the book dives into how computational processes operate more as a performance of meaning than as a transparent execution of it. Like poetry where meaning is never a fixed idea, in programming. Code when viewed through this lens is not something that purely executes but like in poetry and coputer-based or programmically created art the form comes out of not something immidiatly visible or literal, but invites us to look into a deeper contextual interpretation. Chun's critique is showing us code is not an antithetical of poetics but an extension of it.

Esoteric programming languages (esolangs) like Brainfuck or Malbolge push this idea of context and interpretation to its extreme, by even questioning the readability of code to even the usuability of it. Making the writing in an esolang an already poetic act where code is both an object and event. Languages like these have limited syntax, that are juxtaposed to "readable"[5] programming langauges, making programming in these languages almost as hard as writing straight machine code, doable or small examples but hard to near impossible for larger structures. Just as how large scale poetic works are almost unheard of outside of the ancient Epic's, tragedies, and comedies.

By applying Wittgenstein's ideas of "language games"[6] to programming languages we find how they create meaning trough context. Poetic forms that impose constraints on the poet lead to different forms of creative expression, different languages have different areas they excell or fail at and thus create their own context of writing for these applications. Meaning of code doesn't have to lie only with their textual qualities, but also from how, when, and where its executed. "Code runs. Code does something. Code executes on the computer and has operational semantics. But code means things to people as well, both implic-

5. readbale programming langauges include mainstream languages like C, C++, python and such

6. A concept referng to simple examples of language use, deriving meaning only from the context of which it is found.

itly and explicitly”[7] examplifying Wittgenstein’s idea that the meaning of a word is in its use in the language or for code the meaning in the code is in the execution of its processes.

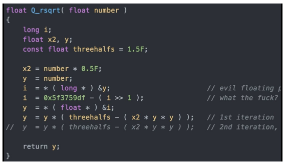
———— Language Games

Wittgenstein’s ”Philosophical Investigations” introduces the idea that meaning in language derive from the context in which it is found. He argued that words don’t have inherent meaning, they acquire meaning trough their role in various ”language games”. These affornmentioned games are more ruled activities which language is used in to achieve their purpose. Programming likewise has meaning emerge from how it interacts with its ”platform” be that compuler, OS, or human interpreter. The act of programming becoming a process of partisipation in forms of language games, programmer manipulating syntax and structure according to a rule-set that will achieve its requested outcome.

Livecoding a practice that has programmers write in a truely performative matter. Writing code for music or visuals live infront of an audience. This live audio-visual programming performance is a direct example of a language game withing the field of computer generated art.

7. Nick Montfort et. al., 10 PRINT CHR$(205.5+RND(1)); : GOTO 10 (MIT Press, 2012) pp.263

In the algorave movement code is used as the tool for expression and setting for their performance. TidalCycles, for example code is used to generate audio patterns, and sound. Where the act of writing and editing code not only to make a final product, but as the process of creation an expierence, blurring the lines of what programming is, moreso positioning itself as the instrument and composer. This "liveness" Hong Kong-born artist, coder, and researcher Winnie Soon would call it is crucial: code is not something static, but dynamic evolving in time. Soon explores this idea in "Executing Liveness: an examination of the live dimensions of code inter-actions in software (art) practice". "Liveness" in the sense that there is more then real-time execution to the activity of code. Its about a shared responsiveness and presence to the programmer, code, and "audience"[8] in the sense of live coding, but user for standalone software. Programming thus is an embodiement of creative thinking, and entangled with human gestures or processes, errors, and improvisations.



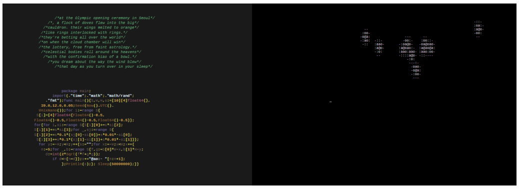
```
float Q_rsqrt( float number )
{
    long i;
    float x2, y;
    const float threehalfs = 1.5F;

    x2 = number * 0.5F;
    y  = number;
    i  = * ( long * ) &y;                       // evil floating p
    i  = 0x5f3759df - ( i >> 1 );               // what the fuck?
    y  = * ( float * ) &i;
    y  = y * ( threehalfs - ( x2 * y * y ) );   // 1st iteration
//  y  = y * ( threehalfs - ( x2 * y * y ) );   // 2nd iteration,

    return y;
}
```

Figure 3: A screenshot of the function found in the source code of Quake III Arena

In Code Poetry, like those of the perl poetry contest or the book "Code Poetry" by programmer and writer Daniel Holden and Poet Chris Kerr showcasing how structures of code can be used as an aesthetic, much like concrete poetry where

8. audience in the sense ranging from live performance audience to a more individual observer

the literary matters as much as the way it's displayed.



```
        /*at the Olympic opening ceremony in Seoul*/
     /*, a flock of doves flew into the big*/
   /*cauldron. their wings melted to orange*/
  /*like rings interlocked with rings.*/
 /*they're betting all over the world*/
 /*on when the cloud chamber will win*/
 /*the lottery, free from faint astrology.*/
  /*celestial bodies roll around the heavens*/
   /*with the confirmation bias of a bowl.*/
     /*you dream about the way the wind blew*/
        /*that day as you turn over in your sleep*/



           package main;
       import(."time";."math";."math/rand";
    ."fmt");func main(){S,W,H,s:=[10][4]float64{},
  39.0,12.0,0.05;Seed(Now().UTC().
 UnixNano());for i:=range S{
S[i]=[4]float64{Float64()-0.5,
Float64()-0.5,Float64()-0.5,Float64()-0.5}};
for{for i,si:=range S{S[i][0]+=s*si[2];
S[i][1]+=s*si[3];for _,sj:=range S{
S[i][2]+=s*0.1*(sj[0]-si[0])+s*0.01*-si[0];
 S[i][3]+=s*0.1*(sj[1]-si[1])+s*0.01*-si[1]}};
 for y:=-H;y<H;y++{s:="";for x:=-W;x<W;x++{
  n:=5;for _,b:=range S{f,g:=b[0]*W-x,b[1]*H-y;
    d:=int(2*Sqrt(f*f+g*g));
       if d<n{n=d}};s+="@ao:- "[n:n+1];
           };Println(s);}; Sleep(50000000);}}
```

Figure 4: Screenshot taken from https://code-poetry.com/flocking named flocking

These functional pieces of code are written in such a way that they also say something about it what is its execution, referencing both an aesthetic quality of text and how it can be used to create form, but also about the recursive loops found in programming. The reading or interpretation doesn't come from a strict form but from the observers cultural context.

Esolangs also play with these langauge games, they are made for a non-practical use. Esolangs exist to explore the boundaries in computer science; challenging conventional notions of what a programming language is supposed to be. Take for instance Whitespace a language designed to be programmed with only white-space characters like SPACE, RETURN, AND TABS. This brings what we normally overlook to the foreground yet keeping it also secret to the eye. Whitespace prioritizes a conceptual and aesthetic poetic writing over any readbility and useablity. Another thing that makes whitespace so interesting is that because it is a langauge that is written using only whitspaces it can be written inside code of langiages that ignore said whitespace. Making that text a polyglot adding another layer of expression within functionality.

———————— Complexity in Simplicity

The algorithmic art of german artist Manfred Mohr like the *Cubic Limit* series shows the rule-based plotting of simple geometric shapes, yet requiring a complex computer program built from scratch in the programming language Fortran V. *Cubic Limit* was a series of studies in algorithmic design, creating works with the simple requirement of showing lines plotted in different matters, all revolving around cubes, cubic forulas, and lines. Other interesting experiments were for instance Visual Equations which was computer generated plotter image with different line structures who were put into an array, then equeling their final structure. Mohr's work which involves the manipulation of geometric shapes changes the artists role from a creator of detail, to the designer of systems which in turn will generate new endless streams of imagery. This way of working is remarkably similar to the field of generative literature and procedural poetry. Creating a form of poetics within the dance of code, computer, and author. Argentine writer Jorge Luis Borges wrote in his short story "The Library of Babel",

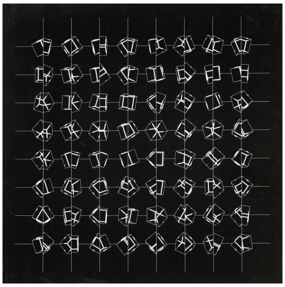


Figure 5: *Cubic Limit*, Manfred Mohr, 1973-75

the story of a library where all possible versions of any 410-page book has been written/generated/exists in. While the library contains all answers to the universe, of how it ends, how to cure any disease the vast majority of the books will either be filled with noise, random sets of letters that have no relation to eachother. And even if the text is legible who can tell the truth from the lies. Whilst fictional this book relates heavily to computer science and generative systems asking the question if all that is generated is not just noise until curated by a person, or is there already inherent quality in this generated "noise"[9]. If we look at this idea and relate it to for instance 10 PRINT which produces and endless and unique maze every execution which has inherent aesthetic qualities even if it is just noise, we could argue that that is us finding meaning in the aesthetics, thus the same as finding meaning in the books that are coherent in the Library of Babel just with a lower threshold of what it means to be "of quality".

———— Writing as a Poetic Act

In "Critical Code Studies" by american writer and scholar Mark C. Marino argues that code should be analyzed as more then functional software, but as cultural writing. The field of critical code studies is an interdisplinary field that examines the social, cultural, and political dimensions of code, argueing that code is not neutral, in the sense that its been written by people, shaped by institutions, used in social systems, and filled with

9. In computer science noise refers to data that is unwanted, or unknown

cultural meaning. Code for login systems that assume gender binaries isn't a technical decision but one rooted in their authors worldview. Any written code is by design, not a given by the code itself, revealing a form unnaturalness. We can also find this inside programming languages themselves where some are made for finance and other more focussed on data. Code isn't something passive, its governs how people live as a regolatory force decided by design but not executed in perfect unity with all that is well. Take for instance redlining (a financial discrimination against residents of certain areas). To fully understand this topic it might be useful to do close reading of a bank's code and see how they decide on approving or disaproving mortgages, not simply the end result of these systems.

Writing code like writing poetry requires a form of rhythm, clear choices and aaesthetic decisions. Programmers have to decide on naming conventions, structure, code style, and a dance of how the execution should run, all having their impact on readability and maintainability, form and aesthetics, and efficiency.

Live coding performances align with this idea of code being something curtural and expressive by performing live programming becomes not just the underlying tool but part of its architecture. The normally hidden black box like code is being forced into the forground as an aesthetic, and social part of a performance artwork. exploring the liveness of code more directly.

Soon in "Executing Liveness" discussed how code operates dynamically. She does this by looking at the execution of software and the underlying code. By highlighting code as not just something written by a programmer, just a set of instructions, but something that is shaped by the process of both execution and the input set. Both Marino and Soon ask the question "what does it mean to run code?" where Marino asks the context of who, and what is ran, Soon focusses more on how it is executed plus how that positions itself comparitavly to the text.

> Programming as an poetic act, is to write code to confront the power.
>
> Robin Opheij

French Algerian philosopher Albert Camus in his speech "Create Dangerously" talked about what it means to make art in times of crisis. This is not unlike what a programmer should and could be thinking of, using code as a protest tool has been done many times, or the other way around code has been censored for being a weapon of war, such is the case for Pretty Good Privacy (PGP), when this software was first written it was being shared online until the US government ended that saying that it fell under international arms dealing and being a theat to national security. So the developer of PGP released the source code as book form which was defended under the free speech act allowing the program to be sent worldwide. To create dangerously is to reveal, not obscure; to break; or to invite. Camus believed that art must confront power, and to write code as an poetic act I believe that one should write code to confront and critique said powers. Code should be used to break sociatal norms, not just reinforce them by being complacent with what is expected. We have been given infinite possibilites, we should explore.

———————— Rhythm and Movement in Code

As we have discussed before code isn't something static or rigid, rather something dynamic and moving. Trough execution, thorough reading, and the act of writing. Code is like a musical score. In algoriymic music composition we can hear the code's aesthetic, repetitve recursion, and structured states. When we use the computer, program, or compiler as a co-performer the code becomes alive or has this liveness, it is transformed into a time-based medium. The poetics of a well performed improvised jazz concert not to unfamiliar to the act of generative music composition. This is not everything though British author and theorist Geoff Cox writes in "Aesthetic Programming A Handbook of Software Studies" that its not just in the liveness of code, but also the act of writing code itself brings rhythm and movement. When writing code, waiting for compilation, rinse and repeat is bound into the program itself and effecting how it is executed in the end by the pauses changing the authors mind and "flow". Flow is a term coined by Mihaly Csikszentmihalyi who explains it as an allencompassing feeling of being "in the zone" fully absorbed by the activity at hand. When a skilled programmer is programming the only thing that will take him out of said zone when and if achieved is the waiting period between input and result. If an error occurs the zone is destroyed, but can still leave a poetic beauty otherwise not found by being pushed out of the zone in this way the programmer can find something new and interesting that was before unseen. Programming is more a time-based practice akin to dance or music.

Glitch art, and error disruption brings up the topic of movement in code or more the lack there of, when a glitch happends the code absruptly breaks, stops in its execution and opens up the black box revealing what is inside. Dutch art theorist Rosa Menkman in "The Glitch Moment(um)" brought up her concept of resolution disruption. Resolution not in the term of image pixel count but expanding the term to include rules, and exceptions. This is how we shape or media, encode it, transmit it, and decode it. The disruption is when this idea fails, artifacts arise and cracks form. Its a poetic display of decay, and rhyhtm. The absence of motion in the form of crashes or errors that create unimagined results holds a tension to the execution that can lead to new meaning, maybe even more profound than what is meant or expected.

The poetics of code is not something really the syntax, or structure, the liveness, or dynamics, its all of it combined and yet all of it on its own. There is a tempo, code moves across time, ans systems, and rewires the mind opening up new ways of thinking about the world, art, literature, and poetry. Becoming its own form of poetry itself.

———————— The Poetics of Code

This exploration that tried to explain the ways that code is a poetic medium over its more utilitarian executuin based use-case has shown that there is a playfullness in the textual aesthetics, processes, and cultural context of code. Exemplifying code as an expressive medium right for poetic expressions.

Programming is a poetic science with its own rules and constraints, opening up the possibilities for creativity and imagination. The programmer is forced to think more elegently by these constraints as we could find in the examples of **FISR** and **10 PRINT**, small, concise, elegent, and rooted in creativity. **10 PRINT** even being rewritten with many different variations exploring this simple algorithm in further depth.

Code should not only be executed, but read, in close reading form as is done with literature to reveal inherent biases, beauties, and cultural impregnations of the time and place of the author. Like poetry is a sign of its time, the culture and ideas of their author, emotions, and feelings so is code an exploration in what is valued and what is percieved at the time of creation. To writ code is to interact with it with the methodology of a poetic science, its not just a science or just designing a rule-set but a play with creativity, and imagination in the forefront. To write code is to interact with it in time and society, execution, and errors further influencing the former and the former influencing the later.

Programming I believe to be essentially a poetic act, code itsel doesn't have to be poetic in and of itself even if it could be take for instance the examples of "Code Poetry" but does realise a poetic expierence between computer, writer, and audiance. An expressive medium that reflects the creativity, and cultural context of the author eveoking emotion in many ways. Be that from the raw textual qualities, the relation between, or its ouptut. Looking at code with this mindset opens us up to interact and engage more with the digital world we have created in a meaninful and determined fashion. So I like to conclude that yes code is poetic.

References

[1] Alan F. Blackwell, et al. *Live Coding: A User's Manual.* MIT Press, 2022.

[2] Jorge Luis Borges. *The Library of Babel*, 1941. https://en.wikipedia.org/wiki/The_Library_of_Babel

[3] James Bridle. *Ways of Being: Animals, Plants, Machines: The Search for a Planetary Intelligence.* Picador USA, 2023.

[4] Anthony Glyn Burton and Wendy Hui Kyong Chun. *Algorithmic Authenticity: An Overview.* Meson Press, 2023.

[5] Albert Camus. *Create Dangerously*, 1957.

[6] Wendy Hui Kyong Chun. "Programmed Visions: Software and Memory." *Choice Reviews Online*, vol. 49, no. 05, 2012, pp. 49–2711. https://doi.org/10.5860/choice.49-2711

[7] Geoff Cox and Alex McLean. *Speaking Code: Coding as Aesthetic and Political Expression.* MIT Press, 2012. http://ci.nii.ac.jp/ncid/BB13182615

[8] Geoff Cox and Winnie Soon. *Aesthetic Programming: A Handbook of Software Studies.* Open Humanities Press, 2020. https://library.oapen.org/bitstream/20.500.12657/46909/1/Soon-Cox_2020_Aesthetic-Programming.pdf

[9] Florian Cramer. "Digital Code and Literary Text." *P0es1s*, 27 Sept. 2001.

[10] Martin Dodge and Rob Kitchin. *Code/Space*. MIT Press, 2011. https://doi.org/10.7551/mitpress/9780262042482.001.0001

[11] Anthony Dunne and Fiona Raby. *Speculative Everything: Design, Fiction, and Social Dreaming*. MIT Press, 2013. https://doi.org/10.5860/choice.51-5390

[12] Clement Greenberg. *Homemade Esthetics: Observations on Art and Taste*. Oxford University Press, 2000.

[13] Heidi Hart. "Timothy Morton (2021). All Art Is Ecological." *Journal of Ecohumanism*, vol. 1, no. 1, 2022, pp. 77–80. https://doi.org/10.33182/joe.v1i1.2061

[14] William Kahan and K.C. Ng. "Sqrt Implementation in Fdlibm." 1986. https://www.netlib.org/fdlibm/e_sqrt.c

[15] David Kushner. "The Wizardry of Id [Video Games]." *IEEE Spectrum*, vol. 39, no. 8, 2002, pp. 42–47. https://doi.org/10.1109/mspec.2002.1021943

[16] Benjamin Labatut. *The MANIAC*. Penguin Group, 2024.

[17] Chris Lomont. "Fast Inverse Square Root." Dept. of Mathematics, Purdue University, 2003.

[18] Lev Manovich. *Software Takes Command*. Bloomsbury Academic, 2013. https://doi.org/10.5040/9781472544988

[19] Lev Manovich. *The Language of New Media*. MIT Press, 2001. http://dss-edit.com/plu/Manovich-Lev_The_Language_of_the_New_Media.pdf

[20] Mark C. Marino. *Critical Code Studies*. MIT Press, 2020. https://doi.org/10.7551/mitpress/12122.001.0001

[21] Charles McEniry. "The Mathematics Behind the Fast Inverse Square Root Function Code." 2007.

[22] Nick Montfort. "A Platform Poetics." https://thedigitalreview.com/issue01/montfort-a-platform-practice/begin.html

[23] Nick Montfort. "Generating Narrative Variation in Interactive Fiction." 2007.

[24] Nick Montfort, et al. *10 PRINT CHR$(205.5+RND(1)); GOTO 10*. MIT Press, 2012. https://doi.org/10.7551/mitpress/9040.001.0001

[25] Rys. "Beyond3D - Origin of Quake3's Fast InvSqrt()." https://www.beyond3d.com/content/articles/8/

[26] Rys. "Beyond3D - Origin of Quake3's Fast InvSqrt() - Part Two." https://www.beyond3d.com/content/articles/15/

[27] Winnie Soon. "Executing Liveness: An Examination of the Live Dimension of Code Inter-actions in Software (Art) Practice." *Leonardo*, vol. 51, no. 5, 2018, p. 530. https://doi.org/10.1162/leon_a_01669

[28] Betty A. Toole. *Ada, the Enchantress of Numbers*. Strawberry Press, 1992. https://openlibrary.org/books/OL22619654M/Ada_the_enchantress_of_numbers

[29] John von Neumann and Herman H. Goldstine. “Planning and Coding of Problems for an Electronic Computing Instrument.” 1947. http://ci.nii.ac.jp/ncid/BA29275660

[30] Ludwig Wittgenstein. *Philosophical Investigations.* Wiley-Blackwell, 2010.

Figures