xCoAx 2021 9th Conference on
Computation, Communication, Aesthetics & X

2021.xCoAx.org

Code, Poetry and Freedom

Keywords: Codeworks, Poetry, Livecoding, FLOSS, Hacktivism, Resistance, Abya Yala

Jorge Forero Rodríguez
jfforero@ldq.cl
FEUP/ITI-LARSyS
Porto, Portugal

Codework, also known as code-poetry, is a variant of digital poetry, in turn a subgenre of electronic literature. Codeworks are basically art that integrate computer code in its overall aesthetic. In its broadest sense, code-inspired visual art and livecoding could be categorized under this taxonomy. Livecoding is a creative technique by which it is possible to compose audiovisual works, interacting directly with the algorithms defined in a programming language, in order to obtain results "on the fly" (Wang 2008). Livecoding performance is permeated by FLOSS (Free / Libre / Open Source Software) culture, since by definition in this artistic format, the source codes of the programs are usually shared publicly with the audience. The research presented below seeks to compose a view around these topics, using Alan Sondheim's codeworks taxonomy in order to analyze them from a critical perspective. In particular, I seek to investigate the Latin American counterpart, where FLOSS and "art converge as an element of autonomy from the functional needs of the structure of code itself, while on the other hand, it attempt to historicize and politicize it by anchoring code in practices of Resistance'' (Ledesma 2015).

1. Introduction

Code-poetry (also known as codework), is a variant of digital poetry, in turn a sub-genre of electronic literature. Codeworks were for the first time described with that name in an article published in the American Book Review, by the artist and critic Alan Sondheim, in the year 2001. In this article the author introduces the concept as:

"The computer stirring into the text, and the text stirring the computer" (Sondheim 2001).

The researcher Rita Raley, refers to the concept as "the use of the contemporary idiolect of computers and computational processes in the form of experimental digital writing or net.writing" (Raley 2002). Sondheim's article proposes a first taxonomy, identifying several formulas by which it would be possible to merge poetry and code.

a. Works using the syntactical interplay of surface language, with reference to computer language and engagement.

This would be the case, as Sondheim describes, for poems inspired by the high-level ALGOL (algorithm-oriented languages) programming language. The idea of *ALGOL's poetry* stems from the first OULIPO (ouvroir de littérature potentielle) manifesto, published by François Le Lionnais in 1962, which proposes from experimental literature;

"other forays may be imagined, notably into the area of special vocabulary (crow, foxes, dolphins; Algol computer language, etc)" (Le Lionnais 1962).

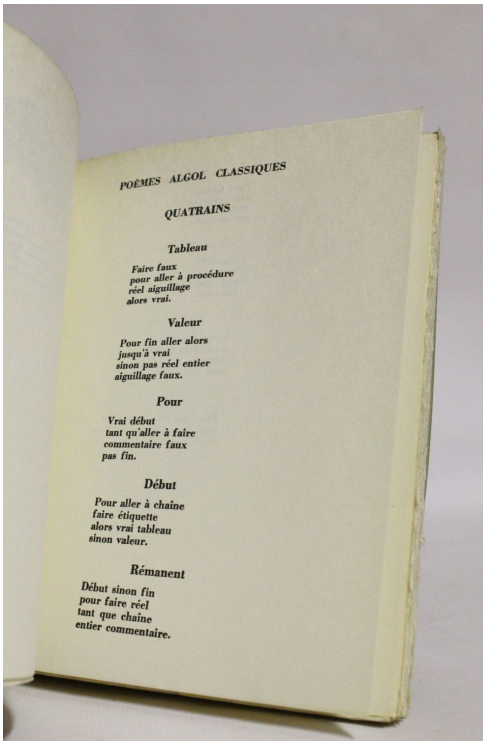


Fig. 1. Noël Arnaud ALGOL poetry book[1]

1. https://www.edition-originale.com/fr/litterature/editions-originales/arnaud-algol-1968-57350

As can be seen from this ALGOL poem (fig.1), written by the Oulipo Noël Arnaud, this approach suggests to use common expressions between natural language and high-level programming languages commands (BEGIN, For, Do, Else, etc.), as a fundamental part of the poems. It is important to mention that although those commands are originally created in English, Arnaud translated them in french. Another perspective in this category also suggests the inclusion of abstract symbolic elements of code in poetry. This is the case of the Mezangelle project by the Australian artist Mary-Anne Breeze. For Raley, *Mezangelle* (M[ez] ang.elle) is a "neologistic 'netwurked' language, which incorporates code snippets, as well as coding structures, such as indentations, parentheses, and other symbols to create new meanings'' (Raley 2002). It is important to mention that both AlGOL and Mezangelle poems are not functional languages by any computer and only refers to codes from its surface layers.

b. Works in which submerged code has modified the surface language—with the possible representation of the code as well.

In this category, codes are executable in the sense they can be compiled and translated into a functional machine language. *Poems By Nari*,[2] for example, are a series of visual poems produced by Ted Warnell, under the pseudonym of Nari in 1996. In this project, the text is deeply linked to the source code used to make the poem. In his poems, Warnell emphasizes this fusion between code and content, stating:

"**what i write:**

document.write("static");

what you read:

Static

what i write:

var x; var y = "dynamic";

for (x = 0; x < y.length; x++)

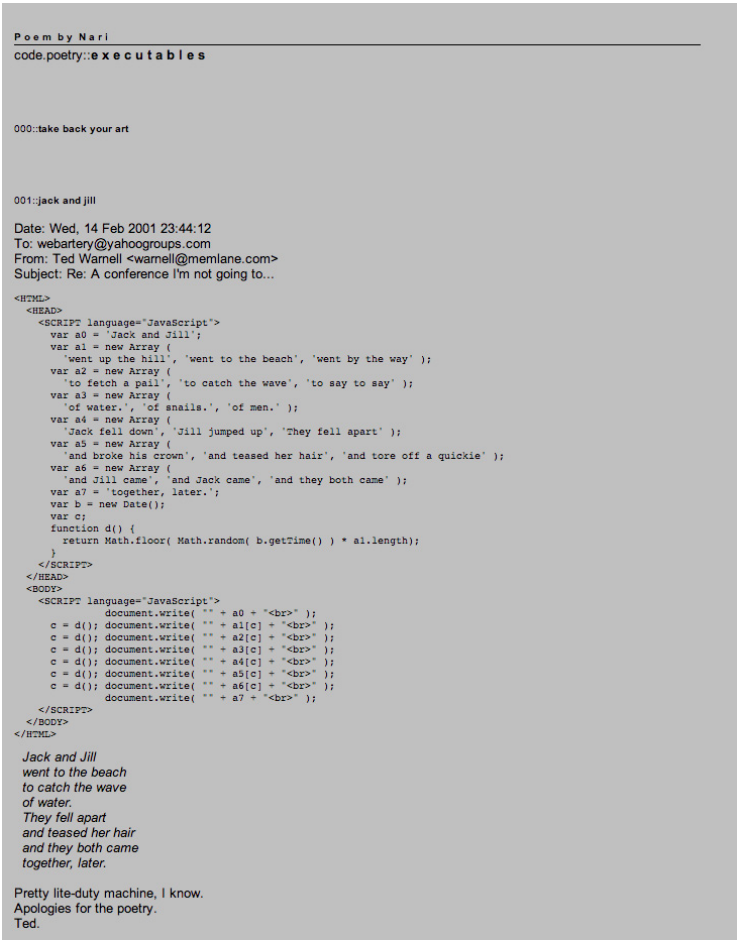
document.write(y.charAt(Math.floor(Math.random() * y.length)));

what you read:

maddicn OR ynyadcm OR imdiyca ...

So there is what I write, and there is what I write writes (what you read) — two different texts: code, a text below and poetry, a text above — they are related and yet separate texts — separate, and inseparable!" *(*Warnell and Quimby 2012*).*

2. https://www.warnell.com/real/nari.htm



Poem by Nari

code.poetry::executables

000::take back your art

001::jack and jill

Date: Wed, 14 Feb 2001 23:44:12
To: webartery@yahoogroups.com
From: Ted Warnell <warnell@memlane.com>
Subject: Re: A conference I'm not going to...

```
<HTML>
  <HEAD>
    <SCRIPT language="JavaScript">
      var a0 = 'Jack and Jill';
      var a1 = new Array (
        'went up the hill', 'went to the beach', 'went by the way' );
      var a2 = new Array (
        'to fetch a pail', 'to catch the wave', 'to say to say' );
      var a3 = new Array (
        'of water.', 'of snails.', 'of men.' );
      var a4 = new Array (
        'Jack fell down', 'Jill jumped up', 'They fell apart' );
      var a5 = new Array (
        'and broke his crown', 'and teased her hair', 'and tore off a quickie' );
      var a6 = new Array (
        'and Jill came', 'and Jack came', 'and they both came' );
      var a7 = 'together, later.';
      var b = new Date();
      var c;
      function d() {
        return Math.floor( Math.random( b.getTime() ) * a1.length);
      }
    </SCRIPT>
  </HEAD>
  <BODY>
    <SCRIPT language="JavaScript">
              document.write( "" + a0 + "<br>" );
      c = d(); document.write( "" + a1[c] + "<br>" );
      c = d(); document.write( "" + a2[c] + "<br>" );
      c = d(); document.write( "" + a3[c] + "<br>" );
      c = d(); document.write( "" + a4[c] + "<br>" );
      c = d(); document.write( "" + a5[c] + "<br>" );
      c = d(); document.write( "" + a6[c] + "<br>" );
              document.write( "" + a7 + "<br>" );
    </SCRIPT>
  </BODY>
</HTML>
```

Jack and Jill
went to the beach
to catch the wave
of water.
They fell apart
and teased her hair
and they both came
together, later.

Pretty lite-duty machine, I know.
Apologies for the poetry.
Ted.

Fig. 2. Ted Warnell. Jack and Jill poem ByNari.[3]

3. https://elmcip.net/node/7943

c. Projects in which the source code emerges as content; there is a deconstruction of superficial language and a dichotomy between the layers of languages.

In this category, Sondheim proposes the works of Netochka Nezvanova and JODI among other artists, alluding to their hybrid and rhizomatic characteristics in the use of languages (Sondheim 2001). Netochka Nezvanova N.N (Nameless Nobody), is one of the first fictional characters in art with multiple identities on the Internet. It has been present since 1995, under the pseudonyms of *anticorp, integer, m2zk! and =cw4t7abs*, among other names assigned to this identity. As with the Pythagoreans, it is difficult - if not impossible - to know the

true authorship of the works presented under the various aliases that NN use, but for decades they have produced works ranging from open source audiovisual programming tools, to code-poetry. JODI, on the other hand, is an artistic duo made up of Joan Heemskerk and Dirk Paesmans. They were among the first artists to investigate and subvert conventions of internet and computer programs. Radically disrupting the language of these systems, including interfaces, commands, errors and code, JODI stages extreme digital interventions that destabilize the relationship between computer technology and its users.

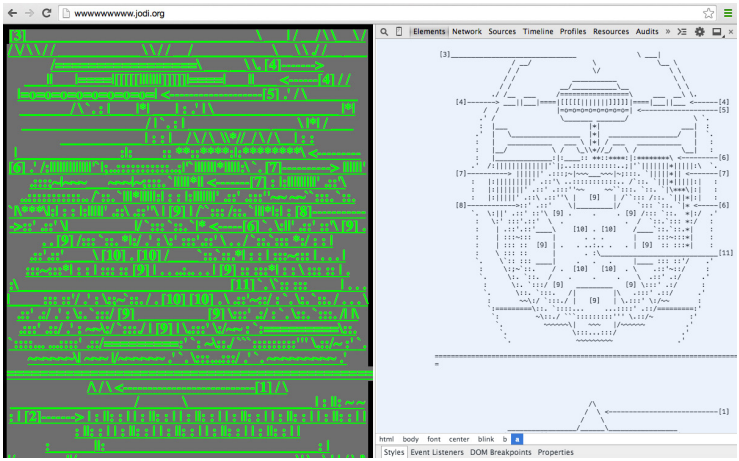


Fig. 3. Jodi Web page
http://wwwwwwwww.jodi.org

2. Livecoding and Freedom

In a broad sense, Livecoding (LC) performances can be considered in the third category in Sondheim's taxonomy, in which it is possible to identify at least three language layers. The idea has its antecedents in 1986 with the works of Ron Kuivila, who together with his experimental group The Hub, composed the piece *Watersurface*, done in a programming language called the Forth music language, developed by David Anderson and Kuivila (Anderso and Kuivila 1991). As described by the pioneers of the format, Alex McLean and his colleagues Adrian Ward and David Griffiths (who formed the Slub collective by the year 2000), "livecoding emerged at the beginning of the 20th century, to describe the activity of a group of practitioners and researchers who had begun developing new approaches to making computer music and video animation in real time" (McLean et al. 2010). It can be defined as the "interactive control of algorithmic processes through programming activity" (Ward et al. 2004). The style of music is not fixed, which suggests that livecoding is a performance method rather than

a genre. A typical livecoding performance involves programmers writing/improvising code on stage, with their screens projected for the audience and their code dynamically interpreted to generate music and/or graphics (Collins et al. 2003).

2.1. FLOSS (Free/Libre/OpenSource Software)

Since the source codes that generate the audiovisual performances are usually projected publicly on a screen, livecoding format is closely linked to the FLOSS culture. Free / Libre / Open Source Software (FLOSS) are computer programs that can be used, copied, shared, modified and redistributed with little or no restrictions and that allow access to their source codes. The term FLOSS refers equally to the concepts Free and Open in order to unify both approaches. Free Software, defined by Richard Stallman and promoted by the Free Software Foundation, places its emphasis on the freedom that this concept brings to users (Stallman 2013). The 4 degrees of freedom in free software are:

» Freedom to use the program for any purpose.

» Freedom to study and modify the source code.

» Freedom to share and redistribute the program.

» Freedom to improve and release new versions.

On the other hand, open source software, in principle equivalent to free software, tries to evade the philosophical question and the political implications of the word freedom and place its emphasis on the peer-to-peer relationship of the model (Stallman 2007). As Martin Zelinger introduces in his article *Livecoding the Law: Improvisation, Code, and Copyright,* "in the field of copyright, livecoding practices can further highlight some of the inherent flaws in traditional intellectual property law" (Zeilinger 2014). Thus, he argues that livecoding has the possibility to radically destabilize intellectual property for two main reasons;

"first, the strong improvisational characteristics of the art form challenge the traditional definition of composer; second, his challenge of palimpsestic nature" (Zeilinger 2014).

The fertile field of studies on jazz improvisation, that offers us a relevant context for the present discussion, shows that improvised practices are recognized as powerful modes of political expression, useful to understand social changes and

to break with structurally inscribed hegemonies in the music. Jazz and in particular free jazz, has often been associated with the insubordinate appropriation of cultural traditions by progressive artists, and with resistance and rejection of an established socio-political order (Stanbridge 2008).

2.2. Hacks, Hackers and Hackerspaces

Livecoding and the FLOSS philosophy are closely related to the D.I.Y (do it yourself) culture and in particular to the hacker world. "Typically, a hacker is a computer-minded technologist and a hack is a clever technical solution that is reached by non-obvious means" (Levy 1984). The term hacker was first used in 1960 among MIT technology developers, whose lives revolved around computer programming (Coleman 2014). The Researcher Steven Levy studied the undeclared ethical codes of hacker groups and conceptualized them under a hacker ethic. These ethical principles were constituted as an amalgam of aesthetic and pragmatic imperatives that included:

"Commitment to freedom of information, distrust of authority, greater dedication to meritocracy and the firm belief that computers can be the basis of beauty and a better world" (Levy 1984).

Hack spaces (also known as Hackerspaces or HackLabs) emerged from 1990, are basically public spaces that promote collaborative creation using technological means. They are usually linked to training proposals that promote the use of FLOSS resources into the DIY culture. Depending on their origin, these spaces are born or derived from cultural community centers (Schrock 2014).

Live coders have their own spaces and institutions. TOPLAP is an organization founded in 2004 to explore and promote livecoding. The activities that they organize are: workshops, talks, periodically livecoding sessions from scratch (improvisation from a blank page), meetings with the research community, as well as Algoraves (livecoding raves parties). They also advise on projects that use real-time code as a central element in the arts, digital humanities, and science. TOPLAP has a founding manifesto, initially focused on musical projects, however it has been expanded to various artistic genres. His manifesto clearly states the necessity of showing the screen as a philosophy against what they call a new digital obscurantism.[4]

4. https://toplap.org/wiki/ManifestoDraft

3. Critical Perspectives

Critical Theory refers to the ways of thinking associated with Frankfurt's Institute for Social Research. As Max Horkheimer described in his foundational essay, a theory is critical to the extent that it seeks human "emancipation from slavery", acts as a "liberating ... influence", and works "to create a world which satisfies the needs and powers of human beings" (Horkheimer 1972).

3.1. Critical Code Studies (CCS)

Critical Code Studies (CCS) are an emerging academic subfield linking socio-cultural studies with computer science, with a special emphasis on computational code. According to the theorist Mark C. Marino, "CCS can be defined as an approach that addresses critical hermeneutics, computer code, software architecture and all documentation from a socio-historical context" (Marino 2006). This perspective proposes that the lines of code do not have a neutral value and can be analyzed using theoretical tools common to any other semiotic system in addition to interpretive methods specially developed for the context. In the article Code as Language wrote by Loss Pequeño Glazier, the author argues that "if language is defined as written symbols organized in combinations and patterns to express and communicate thoughts and feelings - language that is executed - then the code is language" (Glazier 2006). Wendy Hui Kyong Chun's proposal goes further and argues that "software is ideology" (Chun 2004). In this sense, we could extend the argument proposing that computer code is ideology too, but an ideology that is doubly hidden by our illiteracy.

3.2. CCS for Multiples Code

In *A Box, Darkly: Obfuscation, Weird Languages, and Code Aesthetics*, Michael Mateas and Nick Montfort describe the concept of multiple coding as a practice by which it is possible to obtain different representations for each language layer (Mateas and Montfort 2005). A classic example of double coding in natural languages would be the sentence "Jean put dire comment on tap", which has grammatical sense in both French and English, but different meanings in each language. Thus, multiple coding becomes a way of describing the many meanings found in code, as in natural language, adding the computational field as another layer of meaning. So, we can think and argue that all programs have inherently multiple code and present the possibility for constructive interpretation. An extreme case of multiple coding could be found in esoterics programming languages. These kinds of expressions are created as an exper-

iment or research where developers explore and extend the possibilities of code languages, sometimes even in an ironic and/or obfuscated way (Mateas and Montfort 2005). For example, Piet[5] is an esoteric programming language invented by David Morgan-Mar and named in relation to the artist Piet Mondrian, in which programs look like abstract paintings. Piet takes inputs like a picture instead of source code, and then it interprets the pixels in the picture as the code. So in this language we can clearly percibe at least three language layers (natural, machine and abstract language).

5. https://www.dangermouse.net/esoteric/piet.html

Fig. 4. Daniel Holden and Chris Kerr. Piet project.[6]

6. http://code-poetry.com/bark

3.3. Functional Language

The article *the Code is Not the Text (unless it is the text)*, published by the artist John Cayley, argues against the academic notion that computational code is itself the text we should study. According to Cayley, the figure of "code-as-text", or the interpretation of the code as the object of study "is more of a rhetorical form, of the baroque euphemism of the new media" (Caley 2002). His main argument is that many of the code-poetry studied by critics at the time, were analyzed just from the surface layer of the code. Thus, the analyzes do not delve into the functional part of the algorithms, in their interpretation as executable instructions and computable processes. Caley examines his poem Hypertalk written in a language that can be interpreted by humans and compiled by computers. From this perspective, this approach critiques proposals such as *Mezangelle* or *ALGOL* poetry, as examples of code-poetry. Although theorists such as Katherine Hayles refer to *Mezangelle* as a Creole composed of programming languages and natural languages (Hayles 2007), Caley claims that in this work the code presented is placed just as a decorative aesthetic, not making this Creole possible, since there is not really a computational language. However, the functionality of the code does not seem to be a requirement in a broad defi-

nition of the genre. In response, the article Interferences: [Net.Writing] and the Practice of Codework by Rita Raley, refutes these unnecessary limitations comparing Cayley's privilege of the code (on output) to Adorno's remarks on music as a mere consequence of the score. As she explains, codeworks do not suggest, nor don't need, that the code must itself be privileged. Therefore, Mez jobs are valid codeworks, because "they play with code structures at the display level of the text" (Raley 2002). Despite their disagreement over the functionality of code-poetry, both Raley and Cayley suggest that code is a special type of language that is worthy of the attention of scholars.

3.4. Critics to Code as Poetry

Geoff Cox, Alex McLean and Adrian Ward, in his essay *The Aesthetics of Generative Code*, compare code with poetry, while developing some techniques to interpret it. As they state, *"the code works like poetry in the sense that it plays with the structures of language itself, as well as with our corresponding perceptions"* (Cox, McLean, and Ward 2001). However, excluding some works for poetry and computer code in its broadest definition, most of the codes are far away from being considered literature. Unlike text, computer code does not usually appear to be written to be viewed by another human (however many times it is). Codes are the way we interact with computers, allowing us to convey thoughts, reasoning and choices that function as an extension of the creator's intentions. The written form is simply a logical notation that is computer-readable, and is a representation of this process. The code itself, therefore, is clearly not poetry. In addition, the artistic meaning of the code rests, as in poetry, in conjunction with its execution, not merely in its written form.

4. Codeworks from Latin Abya-Yala[7]

As with their Anglo-American counterpart, many Latin American artists have adopted a code-inspired aesthetic, creating projects that could be classified under Alan Sondheim's proposed taxonomies. In this sense we can find antecedents in some works of the Uruguayan artist Brian Mackern that fall into the first Sondheim's category. The project Wartime (2003) per example, is a codework that is situated on the invasion perpetrated by the United States and other countries to Iraq. As Mezangelle, Wartime does not use a functional programming language, but far away from mezangelle esthetical concerns, Wartime is situated in a notorious political context.

7. The ancient aboriginal civilizations that inhabited the land that Europe named America, had numerous denominations. The Aymara leader Takir Mamani invites us to revindicate the original name proposed by the Kuna culture for america; Abya Yala, which means land of blood.

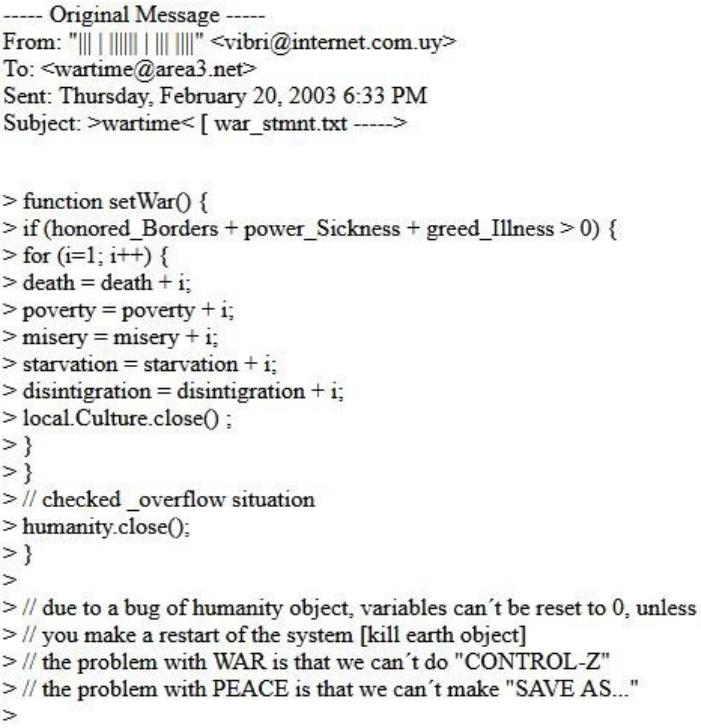


Fig. 5. Brian Mackern WarTime poem.[8]

8. http://netart.org.uy/wartime/

A relationship between codeworks and FLOSS in Latin America can be found in the poem *you CODE me*,[9] by the Mexican artist Laura Balboa. Balboa's poem may be in the line of not executable codes too, but it embodies the hacker's transgressive stance toward freedom of knowledge and dissemination of information through the deployment of codes and symbols. For instance, the poem highlights the concept of CopyLeft, using the expression "Viva la copyleft'' to expose the dominance of English-American culture by implicitly criticizing how English populated technological languages while "she ironically celebrates this linguistic hybridity" (Ledesma 2015). The techno-poetic works developed since 1999 by the Mexican artist and programmer Eugenio Tisselli, goes from net.art to the code-poetry. One of his first proposals entitled Midipoet[10] is a real-time image and sound manipulation computer program that allows to compose and interpret various types of works. According to the Argentine researcher Claudia Kozak, "the politicization in the practice of digital poetry has become very evident in Tisselli's works and latinamerican net-artists in general" (Kozak 2017). His codework project entitled *Degenerativa*,[11] meanwhile (2005), consists of a

9. http://youcode.me/index.html

10. http://motorhueso.net/midipoet/

11. http://www.motorhueso.net/degenerativa/

web page that degrades the code as it is visited. This idea about the transitory and degradation is taken up in several of his proposals, particularly in his work *The27th.The27*,[12] where each time the New York Stock Exchange Composite Index closes with a positive percent variation, a fragment of the 27th article of the Mexican Constitution is automatically translated into English.

12. http://motorhueso.net/27/

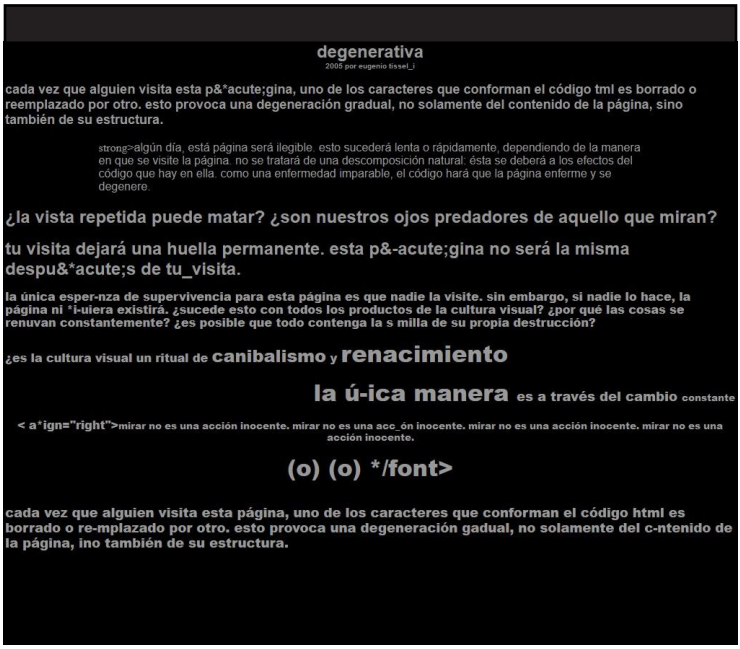


Fig. 6. Eugenio Tisselli *Degenerativa* poem.

4.1. Livecoding Yala (Livecoding land)

In the field of livecoding from Abya Yala, Hernani Villaseñor Ramírez register the first activities in Mexico, particularly since 2006, carried out by members of the Audio Workshop of the Multimedia Center (CMM), such as some concerts by the mU group, formed by Ernesto Romero, Eduardo Meléndez and Ezequiel Netri or in some concerts performed by Netri himself (Ramirez 2019). From 2009 to 2014, the aforementioned space also periodically organized livecoding concerts that were characterized by the writing of source code from scratch (a blank page), a duration of nine minutes per participation, the projection of the code to the public, participation of one live coder of sound and another of image in turn, and a predominant use of the SuperCollider and Fluxus softwares. Thus, in what griffiths entitled “the Mexican style” (Griffiths 2012), artists such as Alexandra

Cárdenas, Malitzin Cortés, Jaime Lobato or the RGGTRN group emerge, among many other artists and creatives who spread their concerns around Abya Yala.

4.2. My Abya (My blood)

My approach to the livecoding is related to a workshop that the Chilean artist Christian Oyarzún shared in the Faculty of Architecture of the University of Chile, at the beginning of year 2018. Christian, by the time, had just arrived from the International Live Coding Conference ICLC in Morelia Mexico,[13] where he was able to meet the Latin American coders. According to Oyarzún, in principle his codeworks were somewhat relegated in front of his audiovisual production. That is, although it was a livecoding proposal, the codes were just seen only by himself, while only the audiovisual results were shared. In such a meeting, the Colombian artist Alexandra Cárdenas proposes to Christian to incorporate that layer and show the process as part of the work. With that in mind, the workshop introduced us to tidalcycles software and discussed some interesting works that had been presented at the conference. In particular, he mentions the case of the Colombian artist Celeste Betancur who develops a proposal called CineVivo, with which it is possible to create livecoding using natural language (Rodríguez, Betancur, and Rodríguez 2019). That idea remained latent in me, so I initially sought to reconstruct an audiovisual work, called the transit of the Kloketen, from the poetics of Natural language coding. For this, I developed a patch in the open source language Pure Data, capable of reading from a text file, messages and arguments that would allow interacting with various pre-programmed audiovisual algorithms. My interest was to be able to point to the creative power of words, to build an identity from a decolonized language. The first live code-work that I built by this way was entitled transit (it is worse than the flood),[14] was presented in the Selk'nam land Puerto Natales, a town wounded by the onslaught of colonization to the point of almost complete genocide.

13. https://iclc.toplap.org/2017/

14. Forero, Jorge (2018). El tránsito del Kloketen. Festival Internacional de Arte y Nuevos Medios LUMEN. https://forero.cl/transito/

4.3. Critical Studies from Abya Yala

While the code-poetry from the "northern hemisphere" has concentrated on aesthetic issues, in Latin Abya Yala seems it has been established from socio-political tension.

On the one hand Latin American works draw on an element of autonomy from the functional needs of the structure of code itself, while on the other, they attempt to historicize and politicize it by anchoring code in practices of Resistance (Ledesma 2015).

The presence of code in the artistic sphere, in addition to incorporating creative tools and techniques, reconfigures the notion of art itself and poses a challenge that encompasses its ontological and axiological meaning. This new writing responds to an integrating purpose of other discourses that "liberate literature from the tyranny of dogma, but it also aims to prevent it from being again engulfed by the system" (Vega 2016). Thus, and in accordance with their Latin American neo-avant-garde predecessors, these post-conceptual writings of software choose to "interrogate the political in art, in the mobile web of poetic strategies, rhetorical devices and forms of interpellation, through which the work projects and disputes its effects of meaning" (Davis 2011).

Latin American codeworks reveal its own uncertain position, making an ideological critique that bears traces of its historical context and its geographical ties with the Global South. It is, therefore, a commitment whose ethical scope entails an aesthetic response of rejection of the models legitimized by the institution. As Eduardo Ledesma argues, Latin Abya Yala presents code not as a mathematical certainty that, by its programmability, determines any outcome, but as indecisive in the, deconstructive sense of resistance to totalitarian and complete meaning or knowledge that resists the very ruthless application of codified rules or programs and casts doubt to the decision-making process, so that indecision becomes the origin of ethics and politics. Of course, for this to be possible it is necessary to intervene directly in the program, using computer code, a type of language that will always imply a degree of efficiency and usefulness even in uselessness. Tisselli reformulates this paradox by appealing to oulipian notions that you can only command language by obeying it.

5. Conclusion

As the expert in complexity Cesar Hidalgo tells us in his book *Why information grows*, although information arises naturally in systems out of equilibrium in the form of order and matter intrinsically has the ability to compute (Hidalgo 2015), communication supposes a first structure in common in the coding that structures a message. A message, as Claude Shannon states, certainly does not guarantee coherence or meaning, but rather it is constituted as a first form of information consensus.

Are we speaking the same language?

Codes and programs provide us new readings and tools with which we can critically analyze and construct cultural heritage, in a society mediated by informa-

tion. For this reason, free access to source code seems to be so important by today. Knowledge cannot be privatized by and for a group. In this same sense, we must establish consensus to decode and share language. Otherwise, it will be the predominant hegemonies that will make a violent conquest of that territory (perhaps they longer do so). Livecoding and FLOSS in this sense naturally aligns themself with a critical and emancipatory perspective. Because the idea beyond proclaiming open and free access to software, warns that asymmetry in access to knowledge only promotes exploitation. In effect, it is about raising a writing that, following Eugenio Tisselli, "can break with the hegemony of economic behavior and its correlative technocracy, proposing, from language, other ways of understanding our way of existing" (Leonardo 2015).

I propose that Latin America seems to run with advantages, since we have had to survive hacking the foreign systems imposed. Hacks, for us, are a value and almost a duty. Thus, although Griffiths distinguishes a Mexican style that he proposes to build from the blank page, this proposal is far from just being aesthetic. The blank page is the possibility that we have from Abya Yala since we continually build ourselves in search of identity.

References

Anderso, David, and Ron Kuivila. 1991. "Formula: A Programming language for expressive computer music." *Computer* 24/7, 12-21.

Berelson, Bernard. 1959. "The State of Communication Research." *Public Opinion Quarterly 23* (1).

Berentsen, Mirthe. 2018. "How we became artists: Net Art duo JODI." Art|Basel. https://www.artbasel.com/stories/how-we-became-artists-jodi.

Caley, John. 2002. *The Code is not the Text (Unless It Is the Text)*. N.p.: Electronic Book Review. https://electronicbookreview.com/essay/the-code-is-not-the-text-unless-it-is-the-text/.

Chun, Wendy Hui Kyong. 2004. "On Software, or the Persistence of Visual Knowledge." *Grey Room* 18:26–51.

Coleman, Gabriella. 2014. "Hacker." The Johns Hopkins Encyclopedia of Digital Textuality. https://gabriellacoleman.org/wp-%20content/uploads/2013/04/Coleman-Hacker-John-Hopkins-2013-Final.pdf.

Collins, N., A. McLean, J. Rohrhuber, and A. Ward. 2003. "Live coding in laptop performance." *Organised Sound* 8(03) (Sound): 321–330.

Cox, Geoff, Alex McLean, and Adrian Ward. 2001. "The aesthetics of generative code." *In Proc. of Generative Art*, no. Art.

Dance, Frank, and Carl Larson. 1976. *The functions of human communication: A theoretical approach*. New York: Holt, Rinehart and Winston.

Davis, Fernando.
2011. “La invención de nuevos conceptos de vida. Estrategias poéticas y políticas en los conceptualismos en América Latina.” 143-155.

Glazier, Loss.
2006. “Code as Language.” *Leonardo Electronic Almanac* 14:5-6.

Griffiths, Dave.
2012. “Mexican livecoding style.” Dave’s blog of art and programming. http://www.pawfal.org/dave/blog/2012/11/mexican-livecoding-style/.

Hayles, Katherine.
2007. “Electronic Literature: What is it?” *The Electronic Literature Organization 1.0* (Literature).

Hidalgo, César.
2015. *Why information grows: the evolution of order, from atoms to economies*. N.p.: MIT.

Horkheimer, Max.
1972. *Critical theory*: selected essays. New York: Continuum Pub. Corp.

Kozak, Claudia.
2017. “Literatura expandida en el dominio digital.” *El Taco en la Brea*.

Ledesma, Eduardo.
2015. “The Poetics and Politics of Computer Code in Latin America: Codework, Code Art, and Live Coding.” *Revista De Estudios Hispánicos* 49.

Le Lionnais, François.
1962. “La LiPo.” First, no. Manifest (sep): 27.

Leonardo, Isaura.
2015. “Poéticas, lo humano a flote.” Entrevista a Eugenio Tisselli.

Levy, Steven.
1984. *Hackers: heroes of the computer revolution*. N.Y: Anchor Press/Doubleday.

Littlejohn, Stephen.
2002. *Theories of human communication*. Belmont, CA: Wadsworth/Thomson Learning.

Marino, Mark.
2006. *Critical Code Studies*. N.p.: Electronic Book Review.

Mateas, Michael, and Nick Montfort.
2005. “A Box, Darkly: Obfuscation, Weird Languages, and Code Aesthetics.” *Digital Arts and Culture*.

McLean, A., N. Collins, J. Rohrhuber, and A. Ward.
2003. “Live coding in laptop performance.” *Organised Sound 8*(03) (Computer Music): 321–330.

McLean, Alex, Dave Griffiths, Nick Collins, and Geraint Wiggins.
2010. “Visualisation of live code.” *Proceedings of the 2010 international conference on Electronic Visualisation and the Arts* (EVA’10), no. Electronic Visualisation and the Arts, 26-30.

Raley, Rita.
2002. *Interferences: Net. Writing and the Practice of Codework*. EEUU: Electronic Book Review.

Ramirez, Hernani.
2019. “Live coding en México: una revisión a partir del concierto “A la escucha del código fuente.”” La oreja culta. https://laorejainculta.net/2019/03/18/live-coding-en-mexico-una-revision-a-partir-del-concierto-a-la-escucha-del-codigo-fuente/.

Rodríguez, Jessica, Esteban Betancur, and Rolando Rodríguez.
2019. “CineVivo: a mini-language for live-visuals.” *ICLC 2019*.

Schrock, Andrew.
2014. ““Education in Disguise”: Culture of a Hacker and Maker Space.” Interactions: *UCLA Journal of Education and Information Studies 10* (Culture of a Hacker and Maker Space).

Sondheim, Alan.
2001. “Introduction: Codework.” In *American Book Review*, 4. Chicago: American Book Review.

Stallman, Richard.
2007. “Por qué el «código abierto» pierde de vista lo esencial del software libre.” GNU. https://www.gnu.org/philosophy/open-source-misses-the-point.es.html.

Stallman, Richard.
2013. “Philosophy of the GNU Project.” GNU. https://www.gnu.org/philosophy/philosophy.html.

Stanbridge, Alan.
2008. “From the Margins to the Mainstream: Jazz, Social Relations, and Discourses of Value.” *Critical Studies in Improvisation/Études critiques en improvisation* Vol. 4 No.1 (Music).

Vega, Aparicio.
2016. *Software Allegorical Writings: Aesthetic Breakdown, Ethic Rearticulation, from Mexican Sphere*. Vol. 5#2. N.p.: Revista Caracteres.

Wang, Ge.
2008. *The ChucK Audio Programming Language: A Strongly-timed and On-the-fly Environ/mentality*. London: Princeton University.

Ward, A., J. Rohrhuber, F. Olofsson, A. McLean, D. Griffiths, N. Collin, and A. Alexander.
2004. *Live algorithm programming and a temporary organisation for its promotion*. London: Goriunova, O. and Shulgin, A.

Warnell, Ted, and Melton Quimby.
2012. “Code Poetry.” SCRIPTjr. nl. www.scriptjr.nl/texts/code/ted-warnell.

Zeilinger, Martin.
2014. “Live Coding the Law: Improvisation, Code, and Copyright.” *Computer Music Journal* 38 (Music): 78-89.