DANCING CODE, SHAKE YOUR PARAMETERS

Alessandro Ludovico

Humans have a unique ability to build formal languages. We use them to both communicate among us, but also to communicate with the machines we assemble. Computer programming languages and natural languages are both formal languages. Nonetheless they stay at the antipodes: one is close to our anthropological way of communicating and the other is close to how the inner machine logic works.

```
From: ".dirtee codah." <netwurker@hotkey.net.au>
Date: Mon Feb 11, 2002 5:20 pm
Subject: N.formation.sources|{i. am. [trapped. in. seizure. language.

>+<
][][][][][][][][in][Form.ational Sauces
>+<living. in. charl.e.tan][ned & lurvely][. glass.][topped danca dangerous][
::drenching wurds with cauls of gritty re:][d][wined rims & gra.][k][nit.e longing
>+++< hearts of c.hun.king stone
::shifting l][iquid polyvalent][ucre melts gigabyting fronts
>+.+.< removable feldspar ][s(ta][c][tic)nakes & jacob][ladders
::whole twitching N.titees d][cl][own.loading l][m][uddite dust
][][][][][][]
][][][][][][][][][][][*][
<=>
][][][][][][][by-][Pa][r][s][e][Sable Violences
<=> non-metallic so][urced[][pi][ngs & channel lusts
<=>][band][width][ing][ & length][ening][ of p][D][ol][l][y][paRA][Me][te][rs
<=> d][t][iz][zy][course. 4. dizzy.cour.sets][N tangled arrays][
][][][][][][][][][][][][][][]
][][][][][-][][][][][][][][.+.][
.+.
][pro][Duc][t][ing & C][iphered][on.tus][sles][ions
.+.][or min.us * owe][
.+.+.][or mining (g)olden [out-of-]boundaried veins][
.+.-.+.][or mined brain-.wavs & glitching heart-r][ot][ational][e][s][.
][][][][][][][][]
][ ][ ][] [][ ]] ][ ]]] ]]][[ ][
]] ] ] [ ] ] [

)( [plus awe
minus |
```

Fig 1. Mez Breeze, mezangelle example.

Fig 2. Open Framework logo.

Fig 3. Dance position with dance notation (Benesh)

Humans have a unique ability to build formal languages. We use them to both communicate among us, but also to communicate with the machines we assemble. Computer programming languages and natural languages are both formal languages. Nonetheless they stay at the antipodes: one is close to our anthropological way of communicating and the other is close to how the inner machine logic works. But, they both instantly establish an understandable abstract environment to describe processes. Their point of contact is centered in the way we're able to write programming language code closer to our natural language (English is the universally adopted one) transversally modifying the way we formulate what we'd like the machine to do, and so generating a significant output. This formulation is a hybrid territory where pure language, explicit dynamic structures and simple to complex formulas collide. Loops, cycles that run depending on value-driven decisions are outputting computed meanings. Words and numbers, meaningfully sequenced are directing the formation of a text, a drawing, a picture, a sound, a movie, or a combination of all the above, with the programmer acting as an open scriptwriter and the user acting as a temporary director and spectator at the same time. These two actors (the programmer and the user) have an invisible and time-delayed relationship that is defined through the programming code, and the same code embodies the many adapted and twisted senses mutating the natural language. This is the territory where historically "software art" steps in. Playing with language and its power to generate impressive output thanks to its ability to use a readable formal language, that is potentially generating infinite sense (as the natural language does).

1. SOFTWARE ART, USING FORMAL LANGUAGES AS ART

Software art ancestors have been retrieved in the seventies, among the artists ascribed to the conceptual and performance art movements. And that now sounds quite plain to see, as they were art movements dealing with language at their core. Particularly "Draw a straight line and follow it" is one of the celebrated conceptual artworks by the composer La Monte Young. It's a small masterpiece as it can be seen as a music score, a piece of visual art, a poetic text, a performance. [1] "This piece can be called a seminal piece of software art because its instruction is formal" Florian Cramer and Ulrike Gabriel claimed in their seminal "Software Art" essay in 2001. [2] Technically it can be defined as a loop, more

precisely an infinite loop that generates a proper half-line, with a fixed origin and a straight direction with no end. It's a concept expressed in natural language, but it perfectly describes the structure of a typical computer programming loop (do something infinitely or until something else happens). Its endlessness has an intrinsic dynamic that is a peculiar software characteristic: dynamically designing a process and enabling the dynamics precisely generated by the software itself. The code becomes than a script infinitely variable as the natural language is. And code becomes a pure linguistic performance, in this perspective. In recent years artist Mary-Anne Breeze has epitomized this approach. She created the "mezangelle", a language composed with hybrid words (conceptually close to the portmanteau words invented by Lewis Carroll). [3] Moreover in mezangelle the words are not only condensed but they also recombine language, stacking multiple layers of meanings into single phrases. This is accomplished hybridizing formal code and informal speech into a condensed textual space (like, for example in "[vec]Tor[n]Space_[di]Stancing"). She cut forms, conventions, phonetic spelling, abbreviations and slang used in the internet culture at large and grafts them onto regular words in a still readable way. These alterations don't follow fixed syntactic or grammar rules but more a very coherent "style" that is instantly recognizable. The poetry and the discourse she composes are an ever changing morphological synthesis of different formal languages into a variable one that embodies the tremendous potential and dynamism of them combined in a different way every time she writes a new text. That's why the initial dilemma of how to categorize art made with software (involving telematic networks or not) has been generally solved sorting it as "performance," that meant to preserve its essential dynamism. In fact considering software as a performance means to acknowledge its linguistic properties, including having a beginning, (at least potentially) an end, and a process that goes on between the two. Moreover it means also to definitively recognize the strategic role of the instructional code and its literary gesture, involving scripting entities and events that form the process. The result is always formal, being it natural language, code or a hybrid one, but it's nevertheless written in a universal and unambiguous language.

2. DANCING CODE

Software can be then defined, with no doubts, as a dynamic process. It can describe an infinite type of processes in a formal language making the computer calculating the output, basing on some input. Now let's consider the dance practice as a process. Especially popular type of dance is often quite formalized through a recognizable code that anybody can follow. There are a few notation systems for describing dance, but none of them are acknowledged as a major standard, and universally acknowledged. The best known are the Labanotation and the Benesh Notation [4] and they use abstract symbols to specify the position of body parts, their direction and the speed of movements. Every notation system has developed its own array of symbols and syntax to generate a shared formal language to express the "set of postural and motion rules to define how the execution of the movement is to be applied." [5] The finite sequences of these symbols are describing an animated sequence of a body performing a choreography with its own embedded narrative. Here the space, where the body is moving, has to be precisely described and the used codes are rendering the space's different peculiarities in a symbolic way. Nevertheless the whole sequence of a notated choreography is a code that functions in a very similar way a software program does. It needs the body as the input and it generates an output of a whole animation of movements in space. Dance is then animating the body through a code and it expresses the smoothness of transition between the start and the end of an event, with all the dynamics in between. Seen through an anthropological perspective, the "code" of dancing practices (feet and body positions, and the sequence of movements) has been historically spread through an oral tradition that in the 20th century has eventually become viral. The description of entities and their movements has been assumed to be learnt by heart (almost in its literal sense), to be then stored and transmitted

eventually with variations. It has been a social process with dynamics similar to what FLOSS (free/libre/open source software) programming is nowadays: acquiring a code, using it, eventually modifying it and vastly sharing it. So it'd not be any accident if computer programming, FLOSS, dancing and social aspects would collide in some ways.

3. PROGRAMMING ANIMATION IS A WRITING (MOVING) PROCESS

So can we consider "computer animation" (the process of programming the movement of different objects on the screen) a form of choreography? Probably yes. Programming animation on a screen is definitely similar to notating or coding a dance choreography. Nonetheless in computer animation the human body is abstracted into any kind of programmable forms, and the space is a virtual three-dimensional one, visualized in the two-dimensional screen. Computer animation can be then considered as an "abstracted dance", since the principles of movement in space remain the same, generally including the physical law we obey to in physical reality. But it retains some specific characteristics of "dance", for example being based on harmonic movements that are expressed through a timeline and strategically positioned in the virtual space, with a peculiar narrative. Beyond that, programming computer animation is an activity that implies some computer programming skills, or being able to describe processes in a computer language code. If historically it has been made in programming code that was very close to the machine logic, one of its major popular shifts has been accomplished in the nineties with the Adobe Shockwave [6] platform. Making animations in Shockwave (with the historical characteristic of being viewable for the first time within any web page through a standard plug-in) involved learning the script proprietary programming language called Lingo. [7] Despite that, a substantial wave of animations was produced in the first wave of the web, and for cd-rom supports as well. In the years two thousands the awareness of the FLOSS (free/libre/open source software) community produced new types of platforms, including openFrameworks, [8] founded by Zachary Lieberman, [9] that were able to move this coding practice to another level of social interaction.
Lieberman noted that he spent his childhood in a printshop. That made him aware that a printing machine, indeed a powerful machine to produce content on a medium, is not something that can be easily owned by a single person, because is too expensive and definitively too heavy. It has to be eventually shared, capitalizing, both socially and economically, on the small community that can be formed around it. He learned as a kid that sharing content producing platforms and skills was the key to improve knowledge and produce beautiful products.
Lieberman engagement with computers and code led him to program animations, and teaching how to do that, in a peculiar way. He considers it a social process, more than a mere technical or educational one. That is evident in his personal research developed working with magicians, in order to help them integrate digital technologies in their public performances. Magic has different distinctive characteristics. It involves deception, because it involves distracting our attention and senses from what is being manipulated. There are also strict rules as, for example, that visual tricks are usually never unveiled, except among the practitioners. But magic involves fascination, and especially being in the mood to let ourselves be fascinated. Still magic is also something experienced in public, and it's an emotional experience. Once shared, it can be a terrific medium of communication for its perceptual involvement, and somehow it's already exploited in communication to wake up the spectator’s attention. But again it is a language skill. And it means to share a common language. Once the people involved are comfortable with a language and a grammar they can communicate, collaborate, share and build systems together. Somehow that almost literally means to bring life to code. If anthropologically building machines that seem to be "live" means to us to construct something with its own moving autonomy, for our senses building a system that contains autonomous entities that follow their "code"

to act, is one of the closest situations where to consider that system as a "live" one. Generating this kind of artificial life is close to our instinct that pushes us to build sophisticated machines that resemble our behaviors. Programming code becomes then the esperanto for building autonomous systems that can rely on the beauty and "magic" of animated objects, unifying a small community around its efforts of creation and simulation of "systems."

4. PROGRAMMING LANGUAGES CAN BE SOCIAL INTERFACES

Language is our most used social interface, but under the proper condition, a programming language can be a social interface too. In the case of openFrameworks, the code, developed under proper FLOSS conditions, becomes social for various reasons. First it relies on an active community that supports it and guarantees its technical update and management. Moreover, when dealing with animation, it deals with an activity that can be "socially" processed as dance historically has been done. Finally quoting Lieberman "we're moving away from objects, we're building systems." The dynamic environments built in open platforms like openFrameworks are shared and constitute micro-worlds that include the internal relationship, as small virtual communities. The process of building them can reflect the social communities outside these environments and reflect them both in the functioning models applied and in building and sharing them through the developing community. There are interesting social consequences for these practices. One is how language (any hybrid one can make by computer code and natural language) can build dynamic social systems and compelling animations beyond enabling a simple communication. And another one, even more important, is the crucial awareness that a temporary or fixed community can build a system, sharing it and using it, or improving it at will. Designing systems collaboratively can then change the communication we usually use. Staying free under these conditions we can become a multitude of test beds on how we can change ourselves and our cultural neighborhood.

References and Notes:

1. *Wikipedia, "La Monte Young," http://en.wikipedia.org/wiki/La_Monte_Young (accessed September 2011).*
2. *Florian Cramer and Ulrike Gabriel, "Software Art," August 15, 2001, http://www.netzliteratur.net/cramer/software_art_-_transmediale.html (accessed September 2011).*
3. *Wikipedia, "Mezangelle," http://en.wikipedia.org/wiki/Mezangelle (accessed September 2011).*
4. *Royce James Neagle, "Emotion by Motion: Expression Simulation in Virtual Ballet," (PhD Dissertation, The University of Leeds School of Computing, 2005), 7.*
5. *Royce James Neagle, "Emotion by Motion: Expression Simulation in Virtual Ballet", (PhD Dissertation, The University of Leeds School of Computing, 2005), 6.*
6. *Wikipedia, "Adobe Shockwave," http://en.wikipedia.org/wiki/Adobe_Shockwave (accessed September 2011).*
7. *Wikipedia, "Lingo Programming Language," http://en.wikipedia.org/wiki/Lingo_programming_language (accessed September 2011).*
8. *openFrameworks, http://www.openframeworks.cc/ (accessed September 2011).*
9. *thesystemis, http://thesystemis.com/ (accessed September 2011).*