SOFTWARE ART AND POLITICAL IMPLICATIONS IN ALG ORIT HM S

PAU DAVID ALSINA GONZALEZ

Before we invented computers to develop artificial intelligence applications and to try to use them as if they were our augmented brains that could help us to gain memory, calculability and evolve new creative abilities, somebody had to start thinking that our thought could be equivalent to calculus or that through logical calculus we could emulate the way we think. Many say that one of the firsts to believe in the power of the logical calculus and the possibility of its automation was Raimundus Lulius, a well known philosopher from Mallorca, now considered to be one of the finest historical personalities in Catalonia.

Raimundus Lulius, or Ramón Llull in catalan, (1235–1315) developed his "Ars Combinatoria", later Leibniz commented it in his "*Dissertatio de arte combinatoria*" (1666) and spread his thoughts throughout Europe making valuable contributions to logical calculus. Llull, who then learned the principles of algebra from his Arab assistant, asserted the calculability of problems and invented a mechanical way to solve them with the aid of a limited number of basic terms called "the alphabetum", as an instrument that could reach the truth. So in his symbolic games he established the assumption of analogy between the logical functions of the brain and a logical machine (as the one he designed, made with three concentric circles).

Since then many other experiences have conducted to the formalization of human thought and to the creation of symbolic machines as machines capable to simulate our thought. Now we have com-

puters crossing our culture and it sounds familiar to us to think about computer science as based on this double basis of both logical calculus and automation. As T Sales[1] says there are as much as 10 components of the "Ars" Lulian system, that could perfectly fit into Computer Science studies as for example: the idea of the solutions calculability of logical reasoning (explored by Leibniz), the idea of a alphabet of human thinking (mathematically interpreted by George Boole), the idea of a general method (heuristic and deductive) the idea of the logical analysis; the notion of a generative system,; the operation through diagrams; or even the graphs theories that order the "Ars" triangular figures.

So others have followed this path and developed automatic machines that evolved into what we now experience as computers; as we already know, key developments were Shannon's theory of communication, Wiener's cybernetics or Turing's Artificial intelligence thoughts that structured a theoretical corpus for the computer science studies. But, now, as computers (its hardware and software) are surrounding us and we keep on living in this continuous research on automation of mental processes, we can see how this research progressively constitutes a particular mental model that structures a range of possible actions within each particular software and hardware's functionalities. As it happens with computers, software assumes also a conception of communication, memory, cognition or intelligence confronted in dialogue with our own human capacities.

We are impelled to philosophical questions related to these conceptions hidden inside computers as they become a cultural artefact that affects us and influence the way we experience and explain the world we live in. As Mathew Fuller explains "software constructs ways of seeing, knowing and doing in the world that at once contain a model of that part of the world it ostensibly pertains to and that also shape it every time it is used."[2] So it's becoming true that machines are capable of truth as we let them conform our conceptions of the human by experiencing it as "natural", through their/our way of structuring actions experienced as

"natural", as if they could not be in another way. And as Inke Arns says too, "now that we are getting increasingly mediatized and digitized it becomes more and more important to be aware that code or software directly affects the virtual and actual spaces in which we are moving, communicating and living."[3]

We could then remember what Walter Benjamin wrote about the cinema back in 1936 and apply it to the software surrounding us. Benjamin said: "the cinema corresponds to deep modifications of the perceptive apparatus, modifications that now all passer-by lives at a private level in the traffic of a big metropolis, as any citizen of a contemporary state."[4] It would not be exaggerated to substitute "cinema" with "software" that rules what's behind the interfaces created to "solve" the human-computer interaction problem. And it would be wise to let us get deeper into computer's ontology of operations that will lead us to the algorithm structures.

Now we are becoming aware of research projects developing bio-inspired hardware devices, capable to merge software dynamism with hardware execution, as for example the POEtics project[5]. But commonly we still have to understand this separation between hardware (the machine itself) and software (the instructions of the machines that lent them dynamism). To accomplish a task, a computer must perform a detailed sequence of operations that are indicated by an algorithm in terms of finite list of instructions, expressed by a finite number of symbols.

These instructions could be integrated at the logical level of the machine through machine code, which is really difficult to do by programming tasks with just strings of 0 and 1. So that is why there are programming languages that make it easy to write good algorithm for a computing system. There are different levels of representation inside the computers: there is the physical system (a structure of integrated circuits), the logical system (the interpretation of high/low voltage as 1/0), the abstract system (patterns of 0/1may represent alphanumeric characters, commands may stand for whole sets of logical instructions), and conceptual system (software applications programmed with programming lan-

guages). So software is written using programming languages that in the end must result interpretable by the machine code at the logical system level which must correspond to physical patterns interpretable by the physical system at the physical level .[6]

So the algorithm analyzes problems and finds a method to solve them, establish a definite list of operations to do and its order of execution. Therefore the objective of an algorithm is to synthesize a task, calculus or mechanism before it is transcribed into the computer. So then the programming languages codify the algorithms in a computer understandable language, but as you know there are different programming languages and most of them are high level, which means that are languages closest to human understanding (C, PASCAL, PROLOG, BASIC, FORTRAN, LISP), as opposed to the low level languages that are closer to machine understanding (assembly languages). Computer instructions are then vital as computer's operations are invoked by internal instructions that are unintentionally triggered by external inputs.

Each language is designed for different uses, and makes possible some actions better than the others. This is related with the "non transparency of the code" idea, that we could draw a parallelism with the "whorf-sapir" hypothesis that says that human thinking is determined by the code of natural language: the speakers of different natural languages perceive and think about the world differently [7]. So there are programming languages for business, for artificial intelligence for kids, for commercial applications or even maybe could be programming languages for art making. As Casey Reas noticed in Ars Electronica, when Computer programs execute they are dynamic processes rather than static texts on the screen. Core expressions of software — including dynamic form, gesture, behaviour, simulation, self-organization, and adaptation — emerge from these processes.

Each programming language has a material with unique affordances and constraints, each has differences and distinct aesthetic gestalts. And from an attempt to build a new programming language for new creative uses capable of unique visual expressions

emerged the "Processing" experience,[8] which makes us think about the relation between art and the possibility of artistic programming languages to be created. Those new programming languages make possible easier programming tasks in order to accomplish brighter visualisations, but we could also think that maybe what is behind the creation of a programming language for artistic creation is the assumption of a conception of art as the consecution of beauty, a conception of art from back in the XVIII century before Schiller and Lessing's thoughts, or even as manual capacities as we could find them in the old greek word "techne," that referred to the creativity of applied arts, far from the notion of art as creation directly involved with social changes, right into its search for modernity.

As Cramer says a conception of art as only what is tactile, audible and visible, take us back to the romanticist philosophy and the privileging of aisthesis (perception) over poeisis (construction)[9]. So once Software art collides with computer culture and art culture both at the same time, different uses of the term art appear living together. In computer culture there is this notion of art as artisanship closer to Donald E. Knuth's "Art of computer programming" and the beauty of the code, which is totally different from the one being used in contemporary art circles. As Andreas Broeckman says, we might need a strong notion of what constitutes art, which could be defined as he explains as "art as about the transgression of boundaries, about making familiar experiences strange, about dramatising what pretends to be innocent, and about exploring the virtualities, the potentialities of technologies and human relationships."[10]

Therefore it would be meaningful what Christa Sommerer says about the properties of a digital artist, quality that lies in the sensitivity to create new visions and explore new tools and structures that support these visions and finally present us with content and experiences that transcend time and material by touching deeper emotional qualities that are not readily explained through code or

numbers alone .[11] But of course in order to do that a deeper knowledge of computer internal hardware architecture, its resources and infrastructure could help to become less dependent on pre-defined limitations. The key point is to evaluate technical possibilities of software and hardware and explore new technical and intellectual ideas, not the other way as an instrumental version of spiritual compensations for technological brutality of the everyday, as Geert Lovink wrote.[12]

Anyway, throughout all digital art scene it seems to be a constant mixture and confusion between art and applied arts, and maybe some will claim this distinction may seem irrelevant nowadays, but the objectives of each one of the conceptions of art involves quite different political implications and approximations to technology, one as a constructive critique the other one many times as a playful use of technology considered as an affirmation without any kind of social critique. So this distinction matters because as the different main categories of software (operating systems, utilities and applications) with their pre-defined algorithms structure and pre-configure our relationship with the information, they also establish a range of possibilities of thought, perception and knowledge within the life-with-computers in the so called "Digital revolution".

This is why a computer's ontology and its deconstruction through art and philosophical practice are needed in order to evoke the "essence" hidden inside cultural reflections on software, their control structures living behind. Software art allow us to do critical reflection of software and its cultural impact, and encourage us to create new algorithms, new sources codes, new programs with new results that could make us able to experience other relationships, new possibilities to manage data, experience reality and explain the world to ourselves.

Through art practice we could be located on the level of the source code, on the level of abstract algorithms or on the level of the result generated by a certain program code, as Rob Myers said on

the PD-List[13]. Each of the levels involves different conceptions of art practice and different possibilities of action. Of course we could look at software art as a sophisticated evolution of the "demo" scene, being at the risk of getting incorporated in the next version of Photoshop as a new filter, or the evolution of those fractal experimentations as an art and mathematics approximation to "the beauty of truth", or a brief social commentary to the main commercial software applications that are surrounding us.

But this deep political and aesthetical implications behind the selection and creation of algorithms and the software's functionalities, behind Google algorithms, military purpose software applications, users conceptions prefiguring software design or holistic user profile databases should aware us seriously and encourage us to take part in its creation, to understand its implications, to observe their development and to throw away its assumed innocence.

Algorithms are the core of programmes, but we can also find algorithms everywhere, algorithms in the washing machine, in a music score, in a book of cooking receipts. An algorithm is a defined set of instructions in order to solve a problem, so once its invented in order to accomplish a task then, supposedly, we do not need to understand its principles, to understand why it works, and we just have to follow its instructions because the intelligence required to do the task it's already codified in the algorithm.

Of course we say "supposedly" because algorithms, those finite list of instructions set in order to solve a problem, are not as "natural" as they seem to be once codified, and, for example, if we type in a keyword in Google's search bar we get a result that could also make us say that there could be different ways to execute this finite list of instructions, and there could be different variables that could help us to identify the key elements of a problem. Google's search engine defines the way in which millions of people find information online, so google's page rank algorithm structures the world on line. Therefore if we turn them upside down algorithms could be considered as tactical media tools, as

Amy Alexander pointed out,[14] structuring actions in another way, for another possible actions in order to solve "other problems" to be considered.

As Gilles Deleuze showed us in his philosophy of cinema,[15] which is also an ontology of the present, we have been trained to read movies and now we are totally able to understand the language of cinema as we have been constructing it through practice, with the meaning of a close shoot, a mid shoot, travelling, etc. As Deleuze drew a taxonomy of images that allowed us to construct meanings within the context of cinema, and reconstructed this careful eye and mind instruction history that expresses the way we have developed our particular way of seeing the world and experience it, now we could start to draw new taxonomies that would show us that we have also been progressively trained to read the operations of software and construct the meaning that allow us to understand and build knowledge through the "universal cut and paste options", or the myriad of filters and menus and options waiting to be happily selected.

But "learning to read images" has little or nothing to do with "learning to act with software" and we do need different traditions to help us understand this software culture getting over the media culture already shocking us. As Lev Manovich stated in the introduction of " the language of new media,"[16] he tried to build this ontology of new media as a study of the operations involved in its development, but finally he mixed up computer's ontology with media's ontology and got more involved in graphical user interfaces than in software and source code structuring actions. We have learned how to comment still images through art history tradition, we then have learned how to comment and read moving images through cinema studies and film theory, but as images are being digitalized (or "numerique" in French) and they can be calculated, and therefore generated, then we have also to learn from the computer science studies. From Media studies we move into software studies; to understand the logic of the new

media we need to turn to computer science and merge different approaches. But after all, is it the computer a media? Or better, is it just a media? Florian Cramer says that computers are not just a media because computers are not just in between sender and receiver but they are senders and receivers which themselves are capable of writing and reading, interpreting and composing messages within the limitations of the rule sets inscribed into them.[17]

So once the computer becomes more than a media, more than a distributor and an assistant and starts being used as a generator itself, everything changes and the software becomes meaningful as a key point in our technological culture. We could then search for the origins of art category that expresses this subject and then establish how the term Software art appeared in the digital art scene during the Transmediale 2001 festival in a panel named "Software Art", as it is usually quoted in many places. The questions raised there were quite significant and still are waiting for responses, as for example the question about the meaning of art within the software code programming, the relation between creativity and computers, the differentiation between conceptions of art and artisanship, etc... But those questions are not new and we can look back into the history of art and recall the computer artists, the algorists,[18] the Burnham's "Software" exhibition,[19] the early days of art and mathematics ,[20] the relation between computer art and conceptual art, even those analogue questions arisen from the poetry field as Cramer suggests.

Many of those questions remind us the early questions brought up by computer art pioneers practices decades ago, when the computer was in the previous graphical user interface period waiting to spread all over the world as a personal computer ready to establish the office metaphor as the only one capable of allowing direct communication with the machine. Now we are so highly influenced by this metaphor that it's really hard to think behind the GUI itself. We are so embedded within their premises that we cannot see them at all as premises, as if they were as natural as the laws of physics and not changeable at all.

The early computer artists knew they had to get deep into the code in order to do their algorisms, they had no other chance and no Photoshop available for them. The early Digital Aesthetics theoreticians had a big problem trying to fit them into an art category, but now all culture passes through computer and its software. We look at the world and build knowledge through windows, databases, browsers, etc... Software is embedded in social practices because there is a cultural dimension of culture as a heterogeneous social field in which software gets built and used, in which it operates and in which it gets developed, as Mathew Fuller argues.[21] The difference between those early "algorists" as Roman Verotsko or Ken Musgrave or "computer artists" as Harold Cohen which are focused on developing programs to generate their work is that in software art software itself is the artwork and what is important is the process generated, not the results of the computer and the output generated as happens with the works executed on paper with plotters and so on.

So once processuality is embedding art, and we are
getting over the image culture to get back to this brand new
text culture where the execution expresses those dynamic
processes evolving from their algorithms structuring its actions,
we need to think about the deep relation between ontology,
politics and aesthetics within those algorithms that lead
us to software art practices, and the importance to
open those black boxes behind the surface
effects of images and audio
surrounding
us
.

[1] Sales, Ton. (1998) "La informàtica moderna, hereva intellectual directa del pensament de Llull", *Studia Lulliana* 38 , pp.51–61.

[2] Fuller, Matthew.(2003) Behind the Blip. New York: Autonomedia

[3] Arns, Inke.(2004) Read_me, run_me, execute_me: some notes about software art. Lecture at Kuda, Novi Sad.

[4] Benjamin, Walter. (1987) Discursos Ininterrumpidos I. Madrid: Taurus

[5] For example the goal of POEtic project is the development of a novel digital electronic circuit, a flexible computational substrate or artificial tissue, capable of integrating the three biological models of self-organization: phylogenesis (P), ontogenesis (O), and epigenesist (E). This tissue will be the essential substrate for the creation of POE-based machines, capable of evolution, growth, self-repair, self-replication, and learning. http://www.poetictissue.org/

[6] Floridi, Luciano. (1999) Philosophy of Computing. New York: Routledge

[7] Manovich, Lev. (2000) The Language of New Media Cambridge: MIT Press p.64

[8] Reas, Casey. (2003) Ars Electronica Catalogue. Programming Media. Hatje Cantz Verlag

[9] Cramer, Florian.(2002) read_me 1.2 catalogue: Concepts. Notations. Software. Art.

[10] Broeckman, Andreas.(2003) On Software, Art and Culture. Nettime Mailing List (25/09/03)

[11] Sommerer Christa. Mignonneau, Laurent (2003) Ars Electronica Catalogue: From the Poesy of Programming to Research as an Art Form. Hatje Cantz Verlag

[12] Lovink, Geert. (2002) Dark Fiber. Tracking Critical Internet Culture. Boston: MIT Press.

[13] quoted in Arns, Inke.(2004) Read_me, run_me, execute_me: some notes about software art. Lecture at Kuda, Novi Sad.

[14] Goriunova, Olga. Shulgin, Alexei. Quickview on Software Art. Interview with Amy Alexander, Florian Cramer, Matthew Fuller, etc..

[15] Deleuze, Gilles.(1983) L'image-mouvement.Cinema I. Paris : Les Editions de Minuit.
Deleuze, Gilles. (1985) L'image-temps.Cinema 2. Paris : Les Editions de Minuit.

[16] Manovich, Lev.(2001)The Language of New Media. Cambridge: MIT Press

[17] Cramer, Florian.(2002) read_me 1.2 catalogue: Concepts. Notations. Software. Art. *http://userpage.fu-berlin.de/~cantsin*

[18] More information about the algorists at http://www.verostko.com/algorist.html

[19] In 1970 Jack Burnham organized and exhibition named "Software — Information Technology: Its New Meaning for Art" , which took place at the Jewish Museum in New York;

[20] Emmer, Michele (1993) The Visual Mind. Cambridge: MIT Press

[21] Fuller, Matthew.(2003) Behind the Blip. New York: Autonomedia