

# What is the relationship between CREATIVE programming and creative writing ?

## 1 Introduction: general position

The main idea I will develop is that creative programming is mainly not a question of control but rather a question of semiotic representation. I will analyse the relationship between making representation inside programming and making representation in written. I suppose that the program is not only a digital written but also a performative text: the totality of the representation it contains can only be grasped when the program runs. It is why I don't say that programming would consist to create lines of code but rather to create lines of code and an observable transient event that only exists while running. So programming consists to create both a permanent representation that lies in lines of code of the source program and data (that both constitute the *texte-auteur*<sup>1</sup>) and a transient representation that lies in the transient observable. In fact, the global representation that programming creates is a complex think that do not restricts to these both representations: it contains another representation that is inscribed anywhere and that lies in the comparison between these both. I will name "codobs" the complete representation resulting of the association of these three representations. Let us note that the addressed person is not the same for these different representations. He is, today, mainly the author for the *texte-auteur*, the reader for the representation that lies in the transient observable<sup>2</sup> and a new role I names "metareader" for the comparison. Let us note that these representations are not legible in an other space than the space they appear : the representation that lies inside the program is not legible on screen, the representation that appears on screen is not legible inside the program and the third representation is only legible by an action of comparison<sup>3</sup>.

So, I will make a distinction between coding and programming. Coding is the local level of programming that only consists in writing lines of code.

There are both a difference and a relationship between programming in programmed digital poetry and creative writing if one considers what arrives at local and global level. A creative writing consists at a deconstruction at the local syntactic level, the level of the sentence by instance, and a specific reconstruction at the global level of the text itself. It is a complex system in which construction and deconstruction occur together. Local level of programming is coding. One cannot make a syntactic deconstruction in coding that would be similar to the textual deconstruction. At this level, coding is completely constrained by the language of programming. It is a "prison" space. So, the notion of reconstruction cannot be used. But the global level of the whole program gives an other issue in which creativity rushes: a specific disjunction between two different parts. Both couples : deconstruction/reconstruction on one hand and constrain/disjunction on the other give the same result: this transformation produces a semiotic text that is out of the common use.

---

<sup>1</sup> The text-author is not the digital files the machine manipulates but the understandable form they take for a human subject : program source in the authoring software, data in a media form (for example, picture, not jpeg)

<sup>2</sup> this representation is the *texte-à-voir*

<sup>3</sup> cf the divers examples I gave.

Creative text is not a common text and creative programming differs from usual programming project.

## **2 First question : what is creative writing practice ?**

In general : I don't know

In poetry : the history of contemporary poetry seems to show that creativity in poetry consists to transform natural language into another language that seems to be "strange and familiar" in order to add a plus-value

So, my definition is : in creative poetry the language is the matter of the language.

What does it mean ?

The language appears two times in this definition, it appends at 2 levels.

### **2. 1 First level : the language is matter**

It means that language is deconstructed. Its properties become material. And what are properties of the language? its linguistic nature, id est syntax and vocabulary; syntagmatic and paradigmatic dimensions. So, at this level, linguistic nature becomes a material: There is no vocabulary in creative writing except your own vocabulary. There is no grammar in creative writing except your own grammar.

### **2. 2 Second level : matter of the language**

It means that language is reconstructed as semiotic forms: new syntagmatic and paradigmatic dimensions appear that define this language as language. It is the main difference, for me, between creative writing and use of language as a matter in non writing works. Naturally, the semiotic reconstruction is not the semiotic of the "common" language that was deconstructed. What is the plus-value ? I think that the most elegant manner to talk about it is to use the Abraham Moles's concept of "supersign". A supersign is a sign with a new dimension that Abraham Moles names the aesthetic dimension of the sign. For him, this dimension applies at the same level as the significant : it is inscribed in the plan of expression, in the unique signifier code. This aesthetic level is evident in concrete poetry but it also exists, in an other manner, in all poetical genres.

### **2. 3 The language "is"**

First and second levels apply together in simultaneity. They constitute together a complex system in the sense of Edgar Morin: the whole is more than the summary of both and each part contains the whole. One don't have, first, a deconstruction followed by, second, a reconstruction, but deconstruction and reconstruction run together in a whole.

So, the condition of the existence of such an object lies in "running", in the performative dimension of natural language pointed out by John Austin in his book " how to do thinks with words" and by Peirce in his concept of interpreter<sup>4</sup>.

---

<sup>4</sup> I am not sure of the English term. The French term is "interprétant"

### 3 Can programming work on this manner ?

Can programming be programmed matter of the program? Or can the program be matter of the program? Can it be both a deconstructive and reconstructive performative object?

#### 3.1 The question of performativity

The physical condition of creative writing lies in its performative dimension. A computer program has also a performative dimension: its execution while running. This dimension is its “raison d’être” But are programming and writing performativity equivalent ?

Not at all. The natural language is not the program of a machine. It is itself programming and machine. One cannot separate something like software and hardware in the performativity of natural language. Natural language does not program anything else itself<sup>5</sup>. Umberto Eco claims this in his essay *Lector in fabula* by saying that the model reader and the model author are strategies inside the text.

But the principle of computing is the total separation between software and hardware. In programming, the performative level lets out of the software: the software only contains the logic of the computing performativity, not the performativity itself. It contains the set of rules that are necessary for performativity but performativity stays in the hardware. In this condition, one cannot deconstruct the language of the program. If one make this, the result is no more performative because the resulting code cannot be applied to the hardware. So, syntagmatic and paradigmatic levels of computing program cannot be changed. One only is able to change it in a metaphoric level or by using digital code like it would natural language used inside other natural language. But it is no more programming: In this case, the program changes its nature and losses its computational performativity.

When one want to preserve the software dimension of programming, one cannot deconstruct its rules of functioning. At the simple grammatical level, that is the level of coding, programming is a “prison space”.

#### 3.2 The difference in non-said

There are also big differences, always at the syntactical elementary level, between text and computing code. I do not speak here of digital code that only can be read by the machine, but of the coding in a high level programming environment. This level is significant for the author.

So, there are two differences between coding for computer and writing a text. The first difference concerns no-said. Each creative text contains a lot of non-said. This non-said is completed by the reader itself, by his knowledge, his experience, his imagination and is a part of the performative level of the text.

What happened with non-said in programming? Generally one considers that a computing program contains no non-said because it contains all rules that are necessary to run. First, it is no more true. At the high level of programming, that is the level at which the

---

<sup>5</sup> I do not take here into account the effect of the language on an addressed person because this effect is the same for a programmed work

author acts, many plugins, OS lines of code, etc... are not written by the author but used by the program while running. One can compare this situation with intertextuality: in intertextuality also an author can use parts of text he has not written. But does a poet use lines of text he cannot read ? When using deep code in library, you did not write them but you did not also read them. In fact, you are only a co-author of the digital code that runs.

And the non-said lies also in other properties of the computing code that has no equivalent at all in a text: the non-said lies in physical parameters that are used while running to achieve performativity. For example, you cannot program or know the duration of execution of a line of code. And this duration is beyond any control of the reader.

So, in the basic level of coding, programming is more like sculpture or cabinetwork: you must follow the “grain of wood”. In a certain way, you are dominated by the coding.

### **3. 3 The inscription of program in the situation of communication**

But in the high level of the whole program, one can say that programming is not like creative writing, it is a part of the creative text. I always talk here about digital poetry programming, not on general programming.

This result is a consequence of the second difference between a creative text and a program. Their inscriptions in the situation of communication that the work creates between the author and the reader are very different. This difference is due to the separation between software and hardware in order to achieve performativity. In programming art or poetry, one has 2 different texts : one inscribed in the program and the other on screen. I use here the semiotic notion of text and not only the linguistic notion. In semiotics, a text is a fabric of signs. The relationships between these two texts are complex. One, the code, is not the description of what happens on screen and the other is not the logical result of what is written in the program. I work at this moment in my works on these differences. I name “transient observable” what happens on screen because it is the transient and temporary observable state of the program while running. The representation on screen can be explained in classical terms of media, space, temporality. But, as its quality of text, the program carries in itself a mental representation of the world that does not appear on screen. This representation can be described in terms of exchanges, data, digital properties, structures, logical rules, physical model, measures etc... Even if it does not appear on screen, this representation of the world is a real part of the aesthetic project. Sometimes, the program resists to the reader exactly as makes a creative text. This resistance is due to the difference of representation of the world between what happens on screen and what is really in action in the running program. I invented the concept of “aesthetic of frustration” to describe this difference. In French digital poetry, aesthetics of frustration often appears in works. In fact, strategies of written that are similar to textual strategies Umberto Eco develops, lie in the couple programming + transient observable. The most imaginative part of the work, its innovative textual dimension can only be approached by comparing the logical description given by the program and the transient observable. In programming for a computer, in fact, creativity does not consist to describe the transient observable, but to introduce differences between what the program seems to make and what it is really making. This difference is possible because running is governed by logical rules and physical events when reading is governed by psychological processes, both in perception and in the creation of signification.

In conclusion, the program does not actualise while running what its is written inside it because of intervention of technical features, and the reader does not read what physically happens in the transient observable because of psychologic features. The creative programming I think about and I makes since fifteen years manages these both dimensions. It is why I prefer to speak of codobs. The codobs is a text. At this level, the program source is a real part of this text. At this level, vocabulary and syntax are no more vocabulary and syntax of the programming language itself but vocabulary consists in complete routines and syntax the way you assemble these routines, the structure of message they exchange<sup>6</sup>. At this level, you can make your own language and coding is truly a matter of the program, programming is the programmed matter of the program. Construction and deconstruction do not apply, they are replaced by disjunction between the program and the observable : between the texte-auteur and the texte-à-voir

---

<sup>6</sup> cf for example the “model of braids” I develops