Definition of topic and outline of work

The current development of the software studies field, and particularly of the critical code studies sub-field opens up an opportunity for this research to build upon existing examinations of source code. Since the existence of beautiful source code has been established, and while techniques of close-reading are being elaborated by scholars of literary and media studies, there is a possibility to further explore the nature and role of aesthetics and poetics within source code. This document outlines my precise approach on the topic, as well as the research protocol to study said topic.

First, there is a need to delimitate the type of literary approach. I would focus specifically on *aesthetics*, rather than on *poetics*. As literary theorist Gerard Genette approaches the concept of literature through both *diction* and *fiction*, I will start from those two complementary, yet separate approaches to the literary artistic.

The difference I would like to draw here is **(1)** between the scales of representation, and **(2)** between *formal representation*, encompassing the concrete signs which make up a work, and *conceptual representation*, the over-arching structures along which a work is composed and then communicated. Following the definition of Jacques Rancière, I understand aesthetics as a physical manifestation which can be grasped by the senses. Aesthetics belong to the *sensible*, whether "the movement of a light, the brush a fabric, the splash of a color" (Ranciere, 2019). These sensible moments, through their deliberate laying out by the (potential) author(s), act as symbols by virtue of being received by an audience. *Aesthetics* as delimited here are the myriad of symbols, linguistic or non-linguistic, which facilitate the communication of a given message of a given nature (e.g. a statement, an emotion, a question, etc.).

As the counterpoint to Genette's *diction*, *fiction* operates on a higher level, and is an organizing force which shapes the work as a whole into a certain configuration, and thus affords the audience of the work the possibility it to apprehend from a particular angle. Fiction here is understood as a proposal for a conceptual representation, a certain way of framing, akin to the difference between comedy and tragedy, to take the Ancient Greek example. Poetics, then, moves toward *what* is being said, while aesthetics skews towards about *how* it is being said. In that sense, it is the process of representing the world, the process of world-making, as Goodman presents it. As a global act of making, poetics operate on a macro-level, and apprehend the subject of the work (be it itself, or something external to it) in a totality. It is an operating process upon that subject, one which transforms its manifestation holistically. If poetics is operating, then aesthetics is being, an existing feature of a work waiting to be received, interpreted through active engagement.

In the light of this distinction, it is those specific formal aspects, these aesthetics,

that I intend to examine in source code, while acknowledging that it is impossible, and perhaps even counter-productive, to completely ignore the poetic part of source code.

So what, indeed, is the role of aesthetics within those specific texts, those programmed texts that have not been considered artworks, and will perhaps never enter a certain field of art production and reception (i.e. galleries, museums, auction houses)? The hypothesis from which I will be starting is that *aesthetic features play a particular role in the need to communicate, and to understand.* A brief overview of existing literature and discourses around the beauty of code always seem to point back to a necessity to be understood —elegance, clarity, simplicity, usability are all values which seem to be central to the definitions of beautiful code by its readers and writers.

The concept of understanding as used here deserves a little more explicit attention, particularly as it exists in source code, stretched between human understanding and machine understanding.

As a phenomena happening between a source and a reception, human understanding implies a general grasp of the causes, consequences of a particular state of things, and the ability to act upon it immediately and to transpose this grasp, or the broad patterns identified, to a situation at different places, moments, and/or with different actors. Conversely, computer understanding focuses more narrowly on the ability to interpret messages and turn them into actions (e.g. the computer understands the code if it does what the programmer wanted the computer to do, through the process of transcribing that want, that intent, into code).

And yet, while humans might be under the impression that they understand something, communicating this understanding proves to be somewhat complicated, an issue that is at the core of the field of linguistics. This understanding is wrapped in codes, contexts and customs, some implicit, and some explicit. Communication between humans therefore somehow seems to always elude *total* understanding. Computers, on the opposite, understand very clearly what is being communicated to them through the medium of programming languages, and will effectively act upon it. Whether the result of that action matches the expectation of the programmer or not, a certain message has still been understood. It is when the computing machine has to expand upon that understanding in order to apply it to a different situation, that the process breaks down. A new message has to be communicated (i.e. a program rewritten) in order to create a new understanding.

Understanding in source code is therefore stretched between this wide, general and loose appreciation of the human mind and this narrow, specific, highly-efficient functioning of the computing machine. I postulate that *aesthetics in source code is a set of literary features through which is negotiated this dual nature of understanding between humans and machines, and that a closer examination of these will shed a new light on the functioning of aesthetics as both a cognitive*

and artistic device.

In order to pursue this work, I intend to proceed as follows. After having completed a literature review on code studies, and gathering a corpus of source code which is both considered "technical" (e.g. the Linux kernel) and "artistic" (e.g. the `{code poems}` anthology), I now intend to identify the principal components of the discourses taking place around the beauty of source code, through technical textbooks, online discussions, prologues to published works and existing critical literature. These components (which will feature elegance, simplicity, cleanliness, clarity, among others) will then need to be further defined in order to compose a set of aesthetic standards, a framework through which the beauty of source code can be evaluated. At this point, it will become necessary to examine the specificity of the languages in which a given text is written. If fiction can be seen as relatively language-independent, diction, the minuteness of a comma, of a bracket or of an upper-case is, to a large extent, enforced by the language in which the program is being written, and might therefore reflect some aesthetic standards differently than in other programming languages. Finally, having constructed a framework through which the beauty of source code can be methodically approached, I intend to apply this framework to existing source code texts to see how such an interpretation might fare, and how productive such an approach might be to understand texts (in both programming and, perhaps, human languages) better. A revised version of these aesthetic definitions will then be elaborated on the basis of these case studies.

A tentative outline of the work could be as such:

1. **The stakes of source code as understandable text**

This part will lay out the limits of my object of study, starting from a literature review on specific code studies, and an overview of the current practices of writing and reading code. From there, I highlight the issue of *understanding* between human and machines through the medium of source code, as an ambivalent notion which necessitates a symbolic interface. Finally, this section will conclude with a justification for the corpus of texts gathered.

- Source code as an object of study (literature review)
- The reading and writing of source code (the practice of programmers -what and how do they read and write?)
- The problem of understanding in humans and machines
- The delimitation of the corpus (from functional, to hobbyist, to artistic source code)

2. **The aesthetics of source code**

This part will be dedicated to the analysis and development of the place of clarity in the aesthetics of source code. Having gathered and analyzed the discourses around what makes source code beautiful, I will locate the place of clarity within a broader panel of aesthetic standards. This investigation of clarity will happen

first in the light of literary studies and aesthetic philosophy, and secondly in the light of computer science and computer philosophy. This work of defining will conclude in the description of a set of aesthetic features which relate in different ways to clarity in source code.

- The discourses of beauty around source code
- Clarity as an aesthetic and literary concept (human-human communication)
- Clarity as a programming concept (human-machine communication)
- A set of aesthetic features for clarity in source code

3. **The linguistic influence of source code**

This set of aesthetic features, composed from both empirical observations and theoretical explorations, will be re-examined at the light of the influence of programming languages. After having elaborated a concept of clarity, I will see if and how programming languages modulate this concept. Programming languages will be investigated both as semantic systems, but also as socio-economic system. By seeing how discourses on clarity vary from programming language to programming language community, it will be possible to identify such a linguistic influence and integrate it into the on-going construction of the concept of clarity.

- The programming language object (programming languages for themselves)
- The programming language ecosystems
- The impact of programming languages on the set of aesthetic features in source code

4. **An application to existing texts**

Finally, this constructed set of aesthetic features related to clarity will be re-examined through case studies of existing source code. The choice of these case studies will attempt to be as broad and representative as possible (both commercial, functional, amateur or artistic), with the aim of using diverse approaches in order to further qualify the aesthetic approaches defined heretofore.

- Case study A
- Case study B
- Case study C
- Conclusion and revision

5. **Conclusion**