At the beginnning of the Spring semester 2020, I had established a clear research direction, directed at what role aesthetics have in the process of understanding source code. While the definition of aesthetics upon which this research relies is based on . The limitation of this starting point is justified mainly by the object of this study. By approaching source code as an object (or, rather, as multiplicity of objects, ``texts’’ written and read), rather than as a concept, I therefore put its graspable aspects in the foreground. While the social, cultural, intellectual and emotional components are still significant in the appreciation of beauty in source code, the comparative lack of close examination of code is written is the justification for such a definition of aesthetics.

What was yet to be defined, however, was the meaning of understanding''. The work conducted this semester has therefore focused on the gathering and examination of the corpus of source code texts, along with the accompanying explanations, justifications and overall meta-texts, in order to find out how are references tobeauty’’ and understanding'' made. Amongst the vast majority of the corpus elements, practicioners tend to present or discuss a piece of source code which they considerbeatiful’‘, ``aesthetically pleasing’’, and accompany this presentation with justifications about to make a piece of code beautiful and/or make a piece of code beautiful, and it is these discourses that are used to elaborate on what source code aesthetics look like , as well as what kind of role they play in the life of source code text. This process has also led to the constitution of an initial set of aesthetic properties that are repeatedly highlighted by a certain sub-set of practictioners. In effect, the group of those who write and read source code is far from being homogeneous, and can actually be grouped into at least three distinct categories: computer science, computational science and software development (Brian Hayes ). While additional sources establish their own distinctions, the multiplicity of contexts within which code is written leaves litte doubt. Leaving aside a thorough defintion of each of these, I’ve identified five main categories of individuals writing and reading source code, which I group under the umbrella term . These categories include: computer scientist, software engineer, hacker, amateur/student and artist. These categories intend to provide heuristics, rather than strict definitions, and each of these categories can overlap within one individual or group of individuals.

these understandings could be the interplay between theoretical, craft and literature

giving up the understanding in order to understand better (cognitive noise is related to the level of skill)

(in reference to the book pat found, where the author states that programming is treated as a management problem, and not a mathematical one: but in both cases, we’re still talking about source code (and to what extent is it always the ``same’’ source code?), a unique object/practice/paradigm which enables us to solve problems (and now, to what extent are these problems dependent on source code? or is it because source code (as computation) can solve theoretically any problem? and is that beautiful in itself?))

while it’s nice to have an overview of humanities at the beginning, it would be interesting to have reconciliation/reconsideration of those views by the end of this paper

relationship to art? question to the understanding of beauty?

apparent symmetry vs.~assymetry of reading vs.~writing

a problem of a lot of research is

information value, or information value?