Understanding Creative Coding characteristics

a large-scale scraping and analysis of open-source projects

Thesis Summary

Shantal Fabri

July 2023

Introduction

Creative coding is an application of computer programming where the goal is to create something expressive or artistic. Through the use of software, code and computational processes, it aims to create results that are not necessarily predefined and rather based on discovery, variation and exploration that can sometimes produce unexpected results.

The growth in its popularity and applicability makes creative coding a relevant and very interesting field to look at, analyze and learn more about. And, moreover, its existence also goes to show the capabilities of computers that go beyond functional purposes.

Nowadays, creators publish and share their creative coding projects online. A lot of creators publish and share their creative coding projects on open source platforms. One of the main platforms where this is done is *OpenProcessing*, a website that hosts over one million projects and invites creative coders, educators and designers to explore, experiment and play, and where creators can share their projects as open source and collaborate with the community. Each project can be shared, downloaded, liked and commented on by other users of the website, who can also fork the projects to further work on them or add their own twists to what other creators have done.

The goal of this thesis is to better understand the state of the art and the current way creative coding is being done. The objective is to understand and characterize what it is that creators are doing and how they are creating, what programming languages they are using, how they are structuring their code and projects, see if there are common patterns between different creative coding projects, what these patterns are and how many share them, etc. After this analysis, the goal is to evaluate how well these creators are programming based on measurements like lines of code (LOC), lines of comments, amount of files per project, complexity of projects as a whole and of functions, parameters used in functions, variation of functions, and other maintainability and complexity indexes.

A quantitative analysis was conducted over a set of thirty thousand projects, publicly available on *OpenProcessing*, whose source codes were obtained by performing web scraping. Static code analyses were conducted on the projects and the whole data set to provide insights on the various source code metrics mentioned above.

Results discussion

There are two main themes on which some conclusive observations can be made after the analysis was carried out and its results presented. These two being; the general patterns found in creative coding projects, with a more detailed description of a typical JavaScript based project, and observations in relation to metrics.

In relation to the general patterns and layout of the source code of creative coding projects, the first thing to mention is the fact that there are two types of projects. Sketches are either based and developed on the Processing language (file extension being pde), also referred to as Arduino Sketch, or they are based and developed on the language JavaScript (file extension being js), in conjunction with HTML.

In terms of the use of other languages or different media files other than JavaScript, HTML or Processing, to complement the creation of the sketch, it was found that the absolute majority of projects don't have any of these files present, with only about a 3% of total files belonging to these. It was observed that hearted sketches have a slightly higher percentage of these files in comparison to created sketches, with the difference being about only 0.5%.

When it comes to Processing based projects, a higher percentage of these files is found among the subgroup of sketches denominated as hearted, with about 5% more files found than in the subgroup of created sketches. In general, considering all subgroups, about 20% of all existing files correspond to Arduino Sketch files. When considering the total number of sketches, about one third of sketches were found to be Processing

based. This information is only an approximation based on the information that Processing based projects contain a single pde file, and that JavaScript based files tend to contain only one HTML file.

The previously stated observation, and the fact of there being no tools for Arduino Sketch static code analysis, lead to only carrying a further analysis for JavaScript based projects. The following observations are made having only files from this language in mind.

If the typical 'skeleton' of a JavaScript based project wants to be defined, the first thing to mention is the fact that they are composed of at least one main JavaScript file, commonly named 'mySketch.js'. Some sketches contain more than only this JavaScript file, these extra files corresponding to either definition of functions then used by the main file or different semi-individual components to the sketch.

In all cases, the JavaScript file(s) is accompanied by one HTML file, that is in charge of embedding the script that defines the sketch written usually in the 'mySketch.js' file, into the web browser. Sometimes, the HTML will embed content from multiple JavaScrip files if the sketch's components are defined separately in different JavaScript files.

When it comes to the main file, 'mySketch.js', it is usually conformed by the setup and draw functions, sometimes exclusively or sometimes including other functions defined by the library p5.js, or other user defined functions.

In the case of hearted sketches, when compared with the created subgroup of sketches, a higher percentage of functions per file and per project was observed. They also presented a higher rate of files per project on average. Both of these observations can implicate a higher modularization, both of components of sketches and functions in general, and also possible a higher complexity of the sketches developed.

In relation to different metrics analyzed, it was observed that among all subgroups of sketches, the statistics and values obtained showed to be very similar. In the case of the hearted subgroup of sketches, they showed, on average, slightly higher values on metrics related to complexity, which goes along with the results and conclusions mentioned above.

When it came to the values of the metrics themselves, all the data showed on average what would be considered as "good and acceptable ranges". With the exception of a few records that 'escaped' the tendencies of all other records, for complexity related metrics, the values were on average low, and in the case of maintainability indexes sketches showed values tending to the higher segment of the range. All this implies that creators in general have code that is maintainable, readable, manageable and not too complex.

Conclusion

After the research done, observing the state of the art and the results obtained and presented, JavaScript, along with the library p5.js, seems to be a language and way of developing creative coding projects that is gaining popularity over the Processing language among creators. The library in question is a very accessible way for people with no expertise in programming to be able to code and experiment with creative coding.

It is important to notice, that even though at source code level, these projects seem to have a very similar structure and characteristics, the results are very dynamic and diverse, and many of them involve the interaction of the user to actually create a meaningful result. For this reason, it is very hard to discern and actually identify what makes a good sketch good by just looking at the code alone. In general, it's hard to obtain many concrete conclusions without also analyzing visually what the result of these projects is.

The only differentiation in use was the division of sketches into hearted sketches and created or recent sketches. When using this as a factor to determine 'good' sketches, it seems that better, more modularized but at the same time more complex code, seems to lead to more appealing sketches that are more liked by the community.