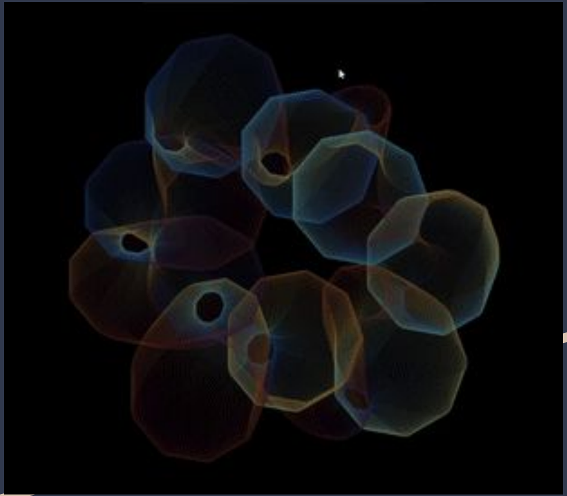
Understanding Creative Coding characteristics:

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

Candidate
Shantal - Marie Fabri Genskowsky

Supervisors
Luigi De Russis
Juan Pablo Saenz Moreno

What is creative coding



Waltz of the Circles by MiniPear

Application of computer programming where the goal is to create something expressive or artistic.

Results not necessarily predefined and rather based on discovery, variation and exploration that can produce unexpected results.

The growth in its popularity and applicability makes it a relevant and very interesting field to look at, analyze and learn more about.

Its existence also goes to show the capabilities of computers, beyond functional purposes.

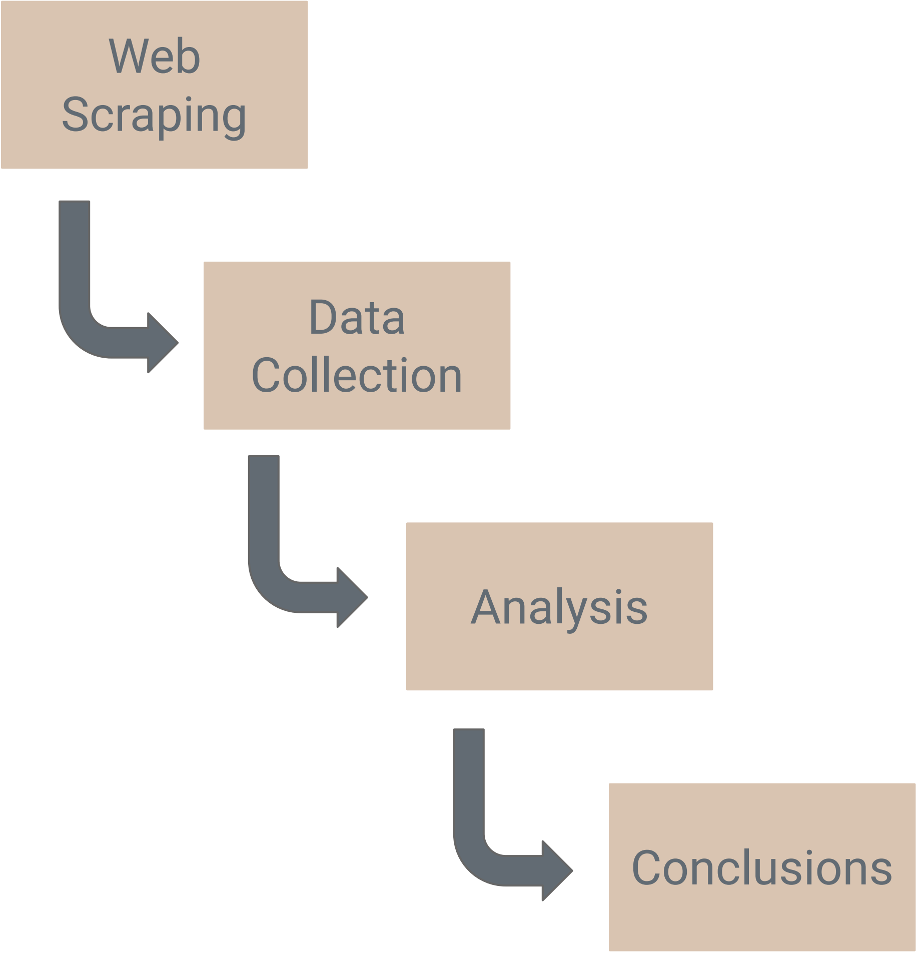
Goal

The goal of this thesis is to better understand the state of the art and the current way creative coding is being done, and then evaluate how well the creators of these projects are programming.

- Understand and characterize how it is that creators are coding, what programming languages they are using, how they are structuring their code, find common patterns.

- Evaluate measurements like lines of code (LOC), amount of files per project, complexity of projects as a whole and of functions, variation of functions, and other maintainability and complexity indexes.

Process and Steps



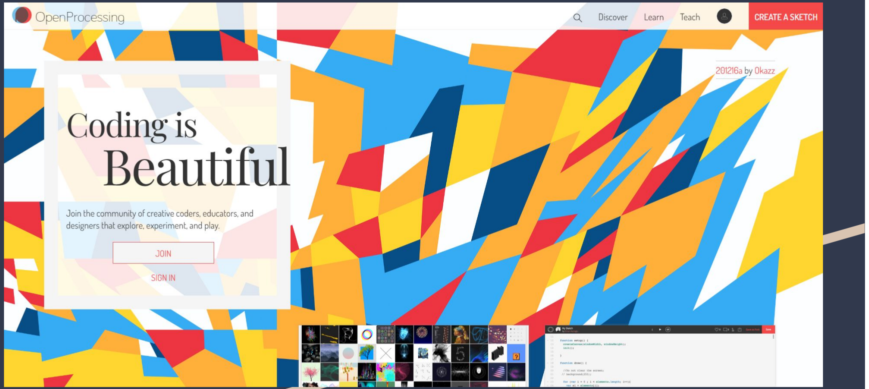
Web Scraping

Data Collection

Analysis

Conclusions

OpenProcessing



Coding is Beautiful

Join the community of creative coders, educators, and designers that explore, experiment, and play.

Main page of OpenProcessing

Creators publish their creative coding projects online, many in open source platforms.

One of the main platforms is the website OpenProcessing (https://openprocessing.org/), that hosts over one million projects.

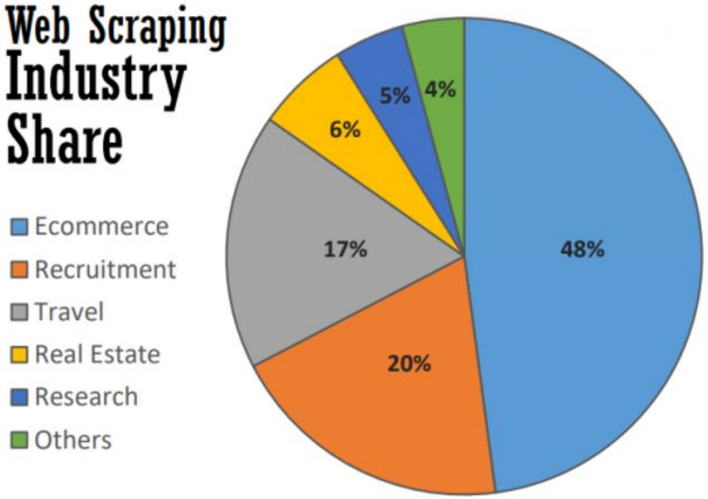
Encourages collaboration within the community. Projects can be shared, downloaded, liked, commented and forked to create the base for a new project to build on.

Web Scraping

Practice of extracting or “scraping” data from the web. The term usually refers to the use of automation tools to collect the data.

Form of gathering and copying data from the web, into a local or central database where it can later be retrieved from or analyzed.

Many applications, like news and content scraping, research, price comparison, etc. and many sectors that use it, such as real estate, travel agency, e-commerce, among others.



Scraping in this thesis

The scraping for this thesis was done over OpenProcessing, with a script to simulate a person navigating the website. This script was written in python with the help of the library Selenium.

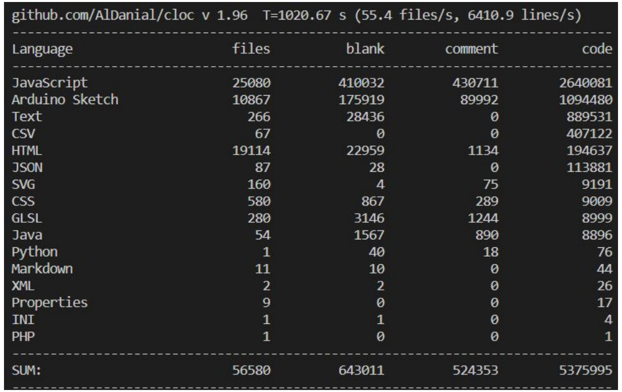
Needed to obtain the source code of almost 30000 creative coding projects, divided in hearted and created subgroups.

Executed in two phases, link collection and downloading of projects.

Data collection with Count Lines of Code (CLoC): files and languages

Command line program that takes files, directory or archive names as inputs, on which it then counts blank lines, comment lines and physical lines of code.

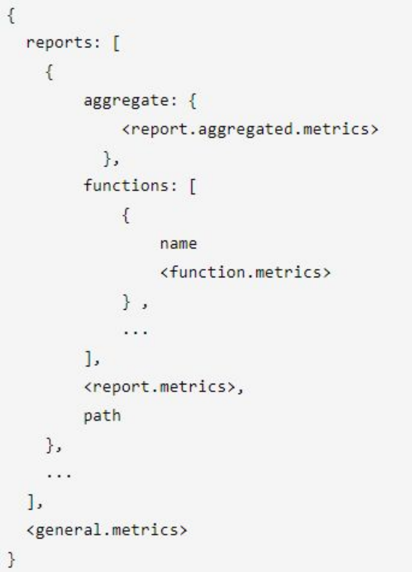
Categorizes files into languages/types, giving information for each one. This was used to get an aggregated overview of what creators are using (languages and other files) and to recognize broad patterns among sketches.



```
github.com/AlDanial/cloc v 1.96  T=1020.67 s (55.4 files/s, 6410.9 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
JavaScript                   25080         410032         430711        2640081
Arduino Sketch               10867         175919          89992        1094480
Text                           266          28436              0         889531
CSV                             67              0              0         407122
HTML                         19114          22959           1134         194637
JSON                            87             28              0         113881
SVG                            160              4             75           9191
CSS                            580            867            289           9009
GLSL                           280           3146           1244           8999
Java                            54           1567            890           8896
Python                           1             40             18             76
Markdown                        11             10              0             44
XML                              2              2              0             26
Properties                       9              0              0             17
INI                              1              1              0              4
PHP                              1              0              0              1
-------------------------------------------------------------------------------
SUM:                         56580         643011         524353        5375995
-------------------------------------------------------------------------------
```

Example output of CLoC

Data collection with Complexity Report (CR): various metrics



```
{
  reports: [
    {
        aggregate: {
            <report.aggregated.metrics>
          },
        functions: [
            {
                name
                <function.metrics>
            } ,
            ...
        ],
        <report.metrics>,
        path
    },
    ...
  ],
  <general.metrics>
}
```

Format of CR report output

Node.js based tool that reads files and produces as output a tree-like report for files and its functions.

The report includes metrics like lines of code, number of parameters, cyclomatic complexity, cyclomatic complexity density, halstead metrics and maintainability index, for both files and functions when applicable.

Used to analyze the JavaScript files present in the sketches.

The produced results were further analyzed, grouped, categorized and worked on with python and jupyter notebook.

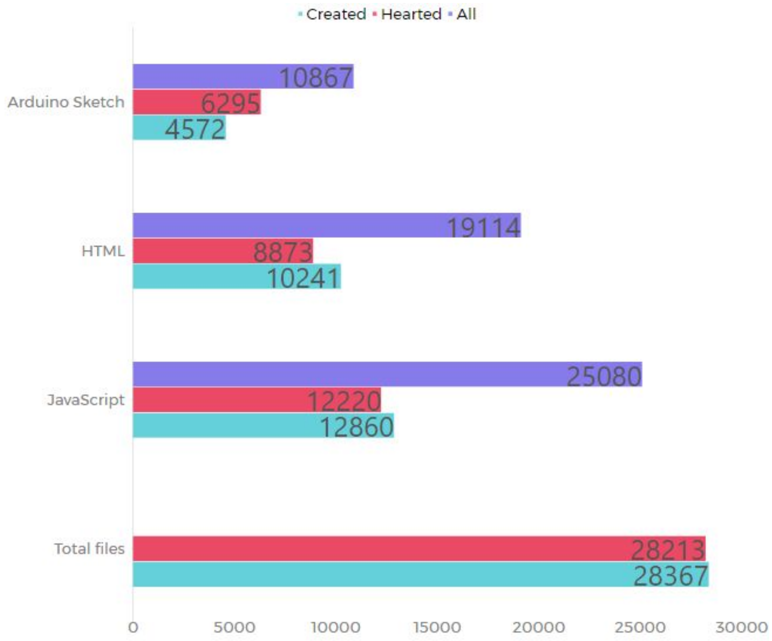
Analysis of CLoC results

JavaScript, HTML and Arduino Sketch (Processing), together about 97% of all files.

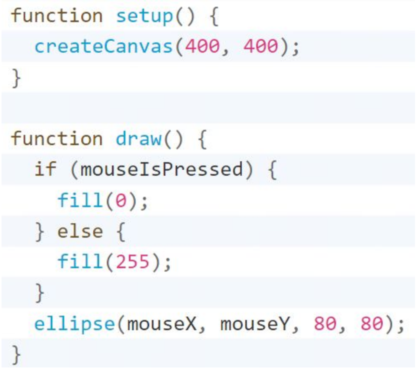
JavaScript files about 44% of all files, HTML about 34% and Arduino Sketch about 20%.

JS based sketches come paired with an HTML file that renders the sketch, the percentages are similar. Using this as an estimate of JS based sketches, about 1/3 of sketches are Processing based, while about 2/3 are JavaScript based.

For hearted sketches, higher counts and averages of lines of code and comments. And slightly higher use of external files.

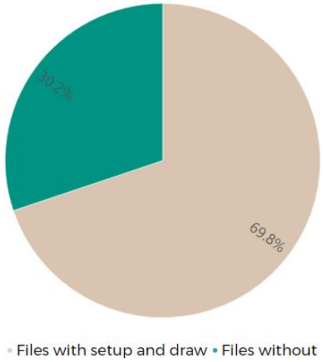


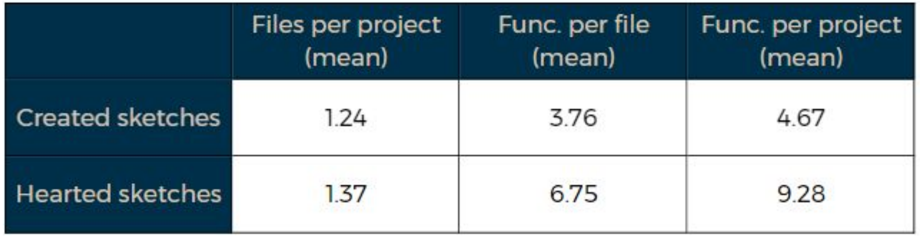
Analysis of CR results: functions and files



```
function setup() {
  createCanvas(400, 400);
}

function draw() {
  if (mouseIsPressed) {
    fill(0);
  } else {
    fill(255);
  }
  ellipse(mouseX, mouseY, 80, 80);
}
```





	Files per project (mean)	Func. per file (mean)	Func. per project (mean)
Created sketches	1.24	3.76	4.67
Hearted sketches	1.37	6.75	9.28

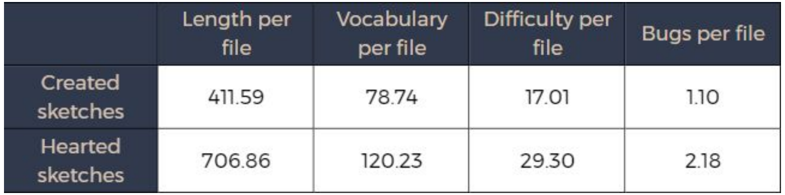
JavaScript based sketches formed by at least one JS file, commonly named mySketch.js.

The base of the sketches is the functions setup and draw, defined by the p5.js library, with almost 70% of files analyzed containing both functions.

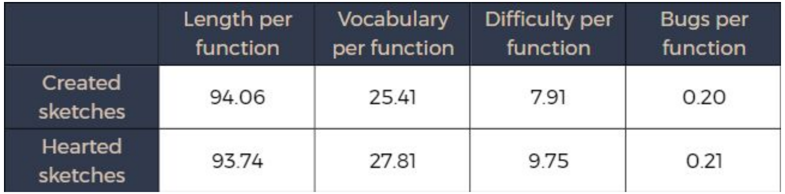
In general, functions take no parameters.

Higher amount of functions per file and per project was observed in hearted sketches. And a higher rate of files per project on average on hearted sketches.

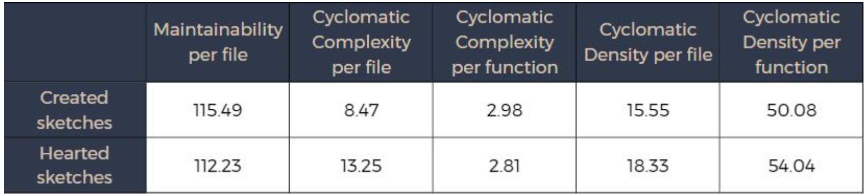
Analysis of CR results: metrics



	Length per file	Vocabulary per file	Difficulty per file	Bugs per file
Created sketches	411.59	78.74	17.01	1.10
Hearted sketches	706.86	120.23	29.30	2.18



	Length per function	Vocabulary per function	Difficulty per function	Bugs per function
Created sketches	94.06	25.41	7.91	0.20
Hearted sketches	93.74	27.81	9.75	0.21



	Maintainability per file	Cyclomatic Complexity per file	Cyclomatic Complexity per function	Cyclomatic Density per file	Cyclomatic Density per function
Created sketches	115.49	8.47	2.98	15.55	50.08
Hearted sketches	112.23	13.25	2.81	18.33	54.04

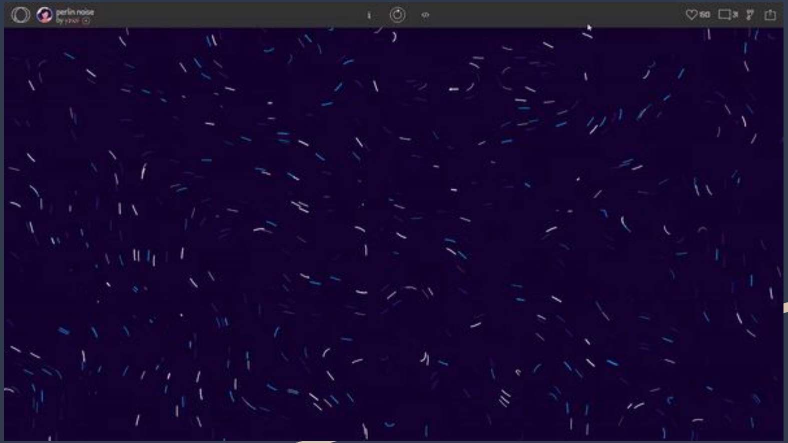
Statistics and values similar across both subgroups, especially at function level, with hearted sketches showing on average slightly higher values on metrics related to complexity.

The data showed on average what is considered as good and acceptable ranges.

Complexity related metrics showed low values, whereas maintainability metrics showed a tendency for high values, relative to this metrics range:]-inf, 171]

Creators in general have code that is maintainable, readable, manageable and not too complex.

Conclusions



perlin noise by yasai

JavaScript with the library p5.js seems to be the leading way over Processing.

At source code level, these projects seem to have a very similar structure and characteristics, but the results are very dynamic and diverse, many requiring user interaction.

Hard to discern and identify what makes a good sketch good.

Using hearted/created division to determine “good” sketches, better, more modularized but also more complex code, leads to more appealing sketches.

Further opportunity of research

Include more data about popularity and visual results of sketches rather than just the source code and division into hearted and created sketches.

Analyze Processing based projects.

Research on the use of external files (not JavaScript, HTML or Processing). What they contribute to sketches and why they aren't currently used.