Graphical Illustration of Code Structure

Wenhao Su* wenhaos3
Yichen Yang† yy18
Zhen Yu‡ zheny2

1 Project Goal

In this project, we will implement an information visualization application using D3.js or VTK.js. This application will be used to visualize the structure of a program and show the relationship between different functions in the program. It will help the developers to visualize how the program works and understand the relationship between different functions more easily. In addition, it will provide the user with the basic information about the program such as the percentage of comments, the number of functions and the number of classes. With this visualization application, one can visualize different kinds of composition in the program and the whole map of relationship between each parts. In addition, one can click the specific parts to learn more about what s/he want to know. In this way, the application will help the users to quickly understand the program.

2 Required Materials and Technologies

2.1 Code Pre-processing

We will use existed library for code pre-processing to parse the code from plain text to some data structure that can be utilized during data visualization. The following are two planning

- Using AST library in Python to parse users' python program into the abstract syntax tree. This would need a back-end program to respond to the front-end request.
- Using Acorn.js that parse Javascript program. This program can be integrated into the front-end Javascript code so that only the browser is need to run the whole project.

2.2 Front-end Interface

For the front-end visualization we will use React framework to build the html web page. We will also use D3.js as the visualization tool.

3 Proposed Deliverables and Rubric

Deliverable	Grade
Pre-processing	4
basic information	2
function and class relationship	4

Table 1: The importance of each deliverable

3.1 Code Pre-processing Module

The first part of the final deliverable is supposed to be a code pre-processing module, which may be written in Python, Javascript or C++, and finally integrated into the front-end graphic interface.

The ideal user story is that: a user opens our application, which is typically a website page, and there should be an interface for him to upload the project files that he wants to visualize. The project may be written in any kind of language, and there might be multiple files. Then, our pre-processing module will analysis the uploaded files and extract the information that we're interested in, such as the plymorphism/inheritance status of classes and function call relationship.

For simplification, the first step of this project may only consider a specific programming language and one or two source code files.

3.2 Visualization

The second part is the front-end visualization of our extracted information from the code. Upon the analysis of the code is done, we decide to display the result in the following two parts.

3.2.1 Basic Information

The basic information part is an overall graphic snapshot of the code, which may be a pie chat of the code composition: how many percentages are classes, functions, comments and testing codes. The following graph is an example of the expected result. When users hover on the specific pieces on the pie chart, information such as detailed lines of code and position in the file are supposed to be displayed.

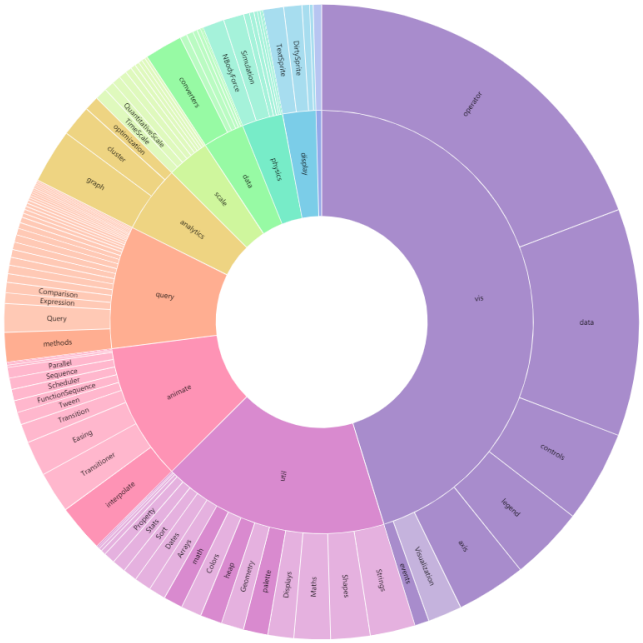

Figure 1: An example pie chart for basic information, cited from https://observablehq.com/@d3/zoomable-sunburst.

*e-mail: wenhaos3@illinois.edu
†e-mail: yy18@illinois.edu
‡e-mail: zheny2@illinois.edu

3.2.2 Function and Class Relationship

Apart from basic statistical data for the code, the user will also have the opportunity to have a view on the relationship between classes and functions. As is shown in Figure 2, we plan to build lines/edges for the cases when a function calls another. In terms of classes, there could be different lines showing inheritance or metamorphism features.

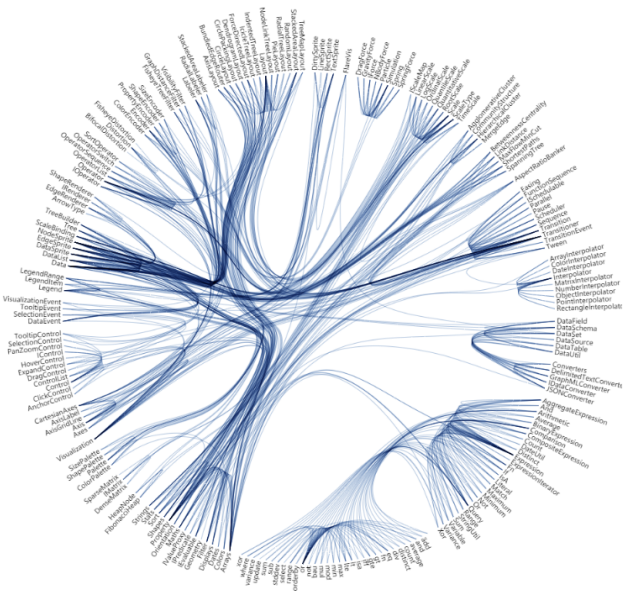

Figure 2: An example function and class relationship graph, cited from https://observablehq.com/@d3/hierarchical-edge-bundling.