CodeGraph

Project Repository

Hannah Swan
hswan@cs.utah.edu
u0939840

October 27, 2017

1 Background and Motivation

The motivation behind this project is selfish, though somewhat motivated by research. In computer science research, as well as many industrial software, the code become increasingly complex. Even well documented programs can be difficult to get started with, or to pick back up and work on several months later. While all the fine level details are exposed in the code files, the overall picture of how all the parts fit together can be difficult to figure out. It's also difficult to trace where any data may flow, what functions may affect it, or what other data it might affect.

2 Project Objectives

I would like to explore some ways to better visualize code. In particular, I'd like to start with looking at functions and variables. For each function/variable, what other functions/variables may it affect, or affects it? Where could the data change? Where does the data flow through the program. These kinds of questions would help in the process of learning new code, remembering old code, better understanding the code, and debugging.

3 Data and Data Processing

I'm planning on using a simple C++ CPU ray tracer as my raw data. I may do some preprocessing on it myself to get it in a "nicer" form to work in. In particular, I'll make a list of all the functions, including the calling functions, and possibly the call order for each calling function. I'd also like to include a list of variables sent to the function. Additionally, I'll keep a list of all variables for each function, and a list of all variables that may affect it's value. For now, I'll treat classes simply as data variables and functions.

4 Design

Figures 1, 2, 3, and 4 shows design ideas.

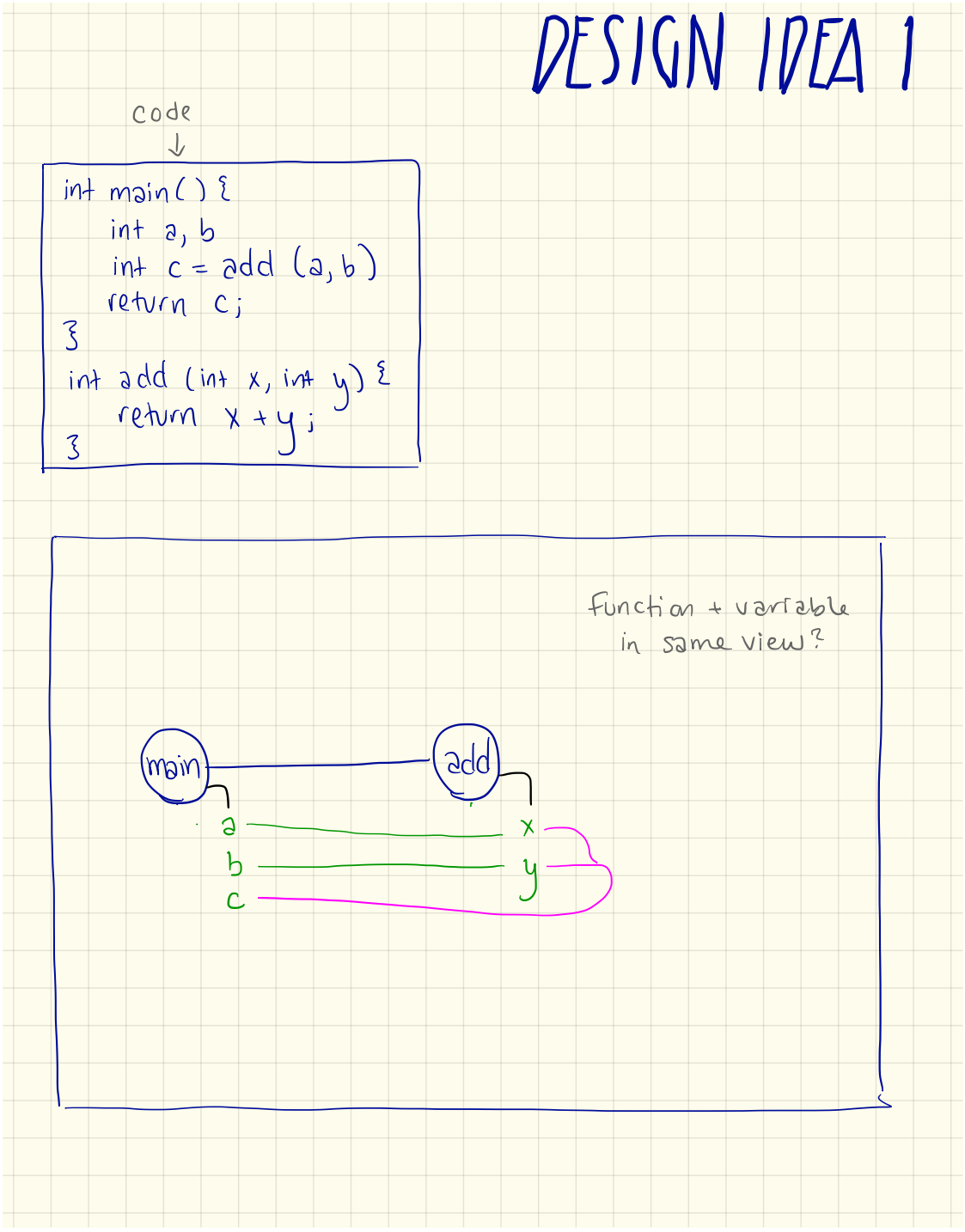

Figure 1: Design 1 Idea

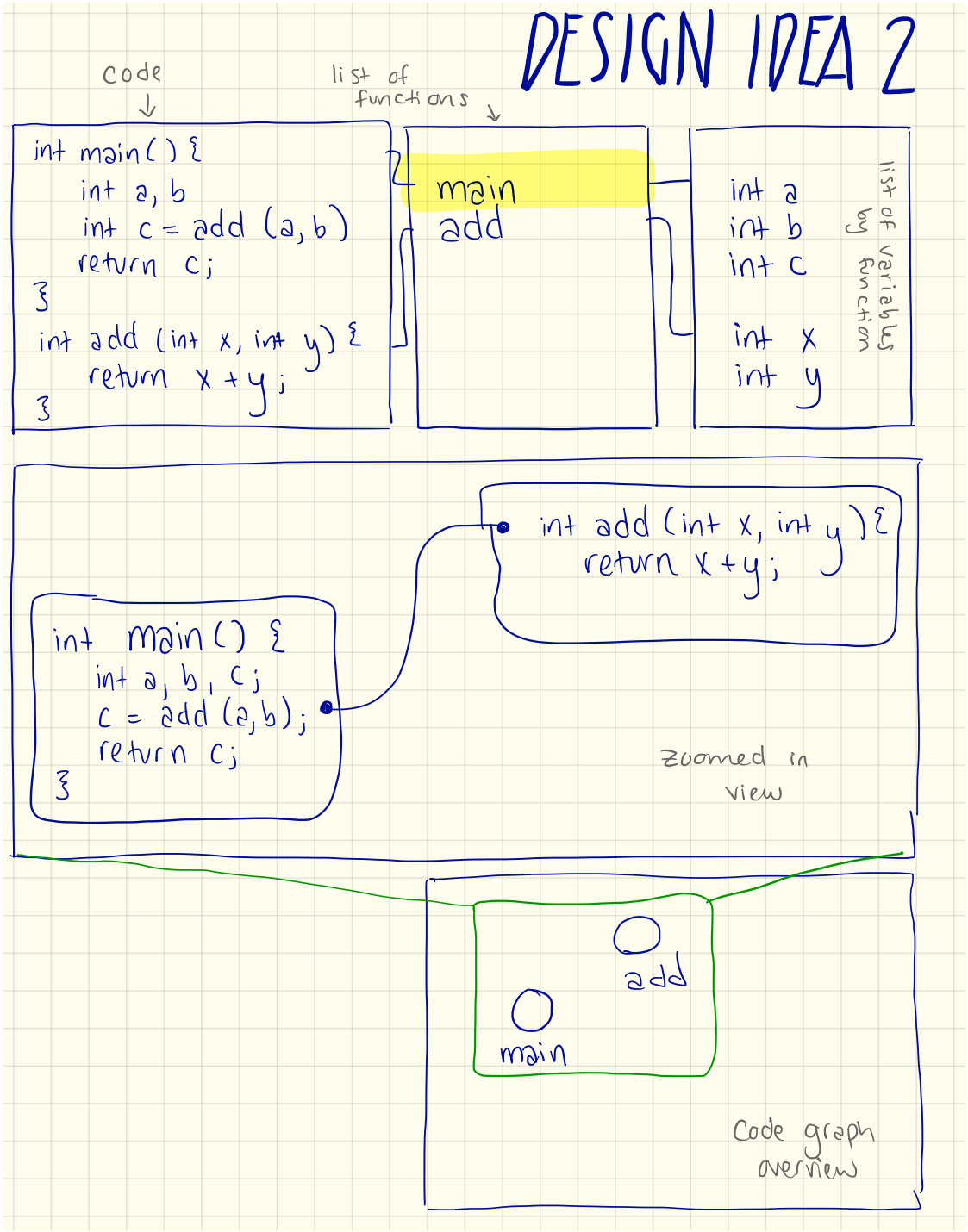

Figure 2: Design 2 Idea

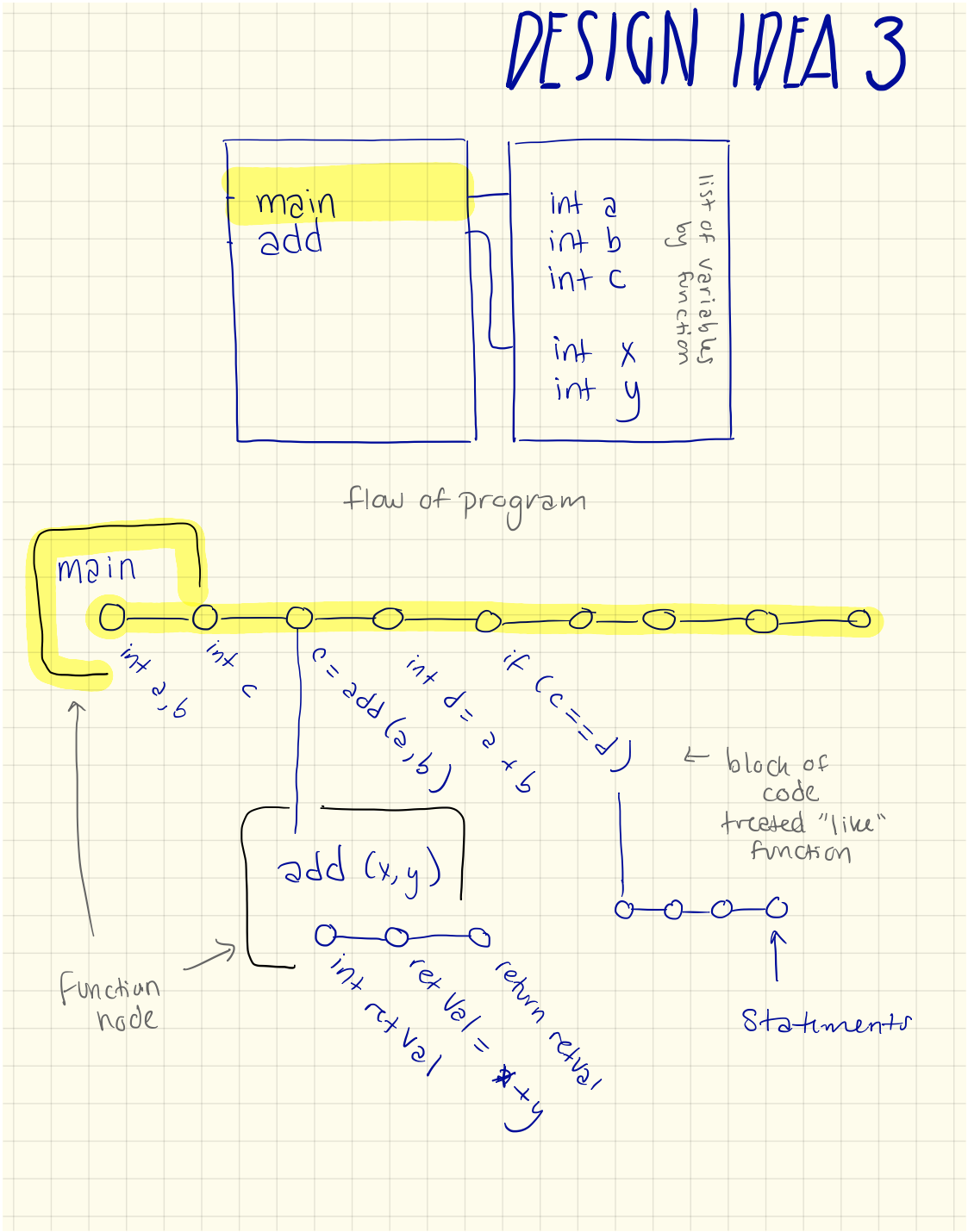

Figure 3: Design 3 Idea

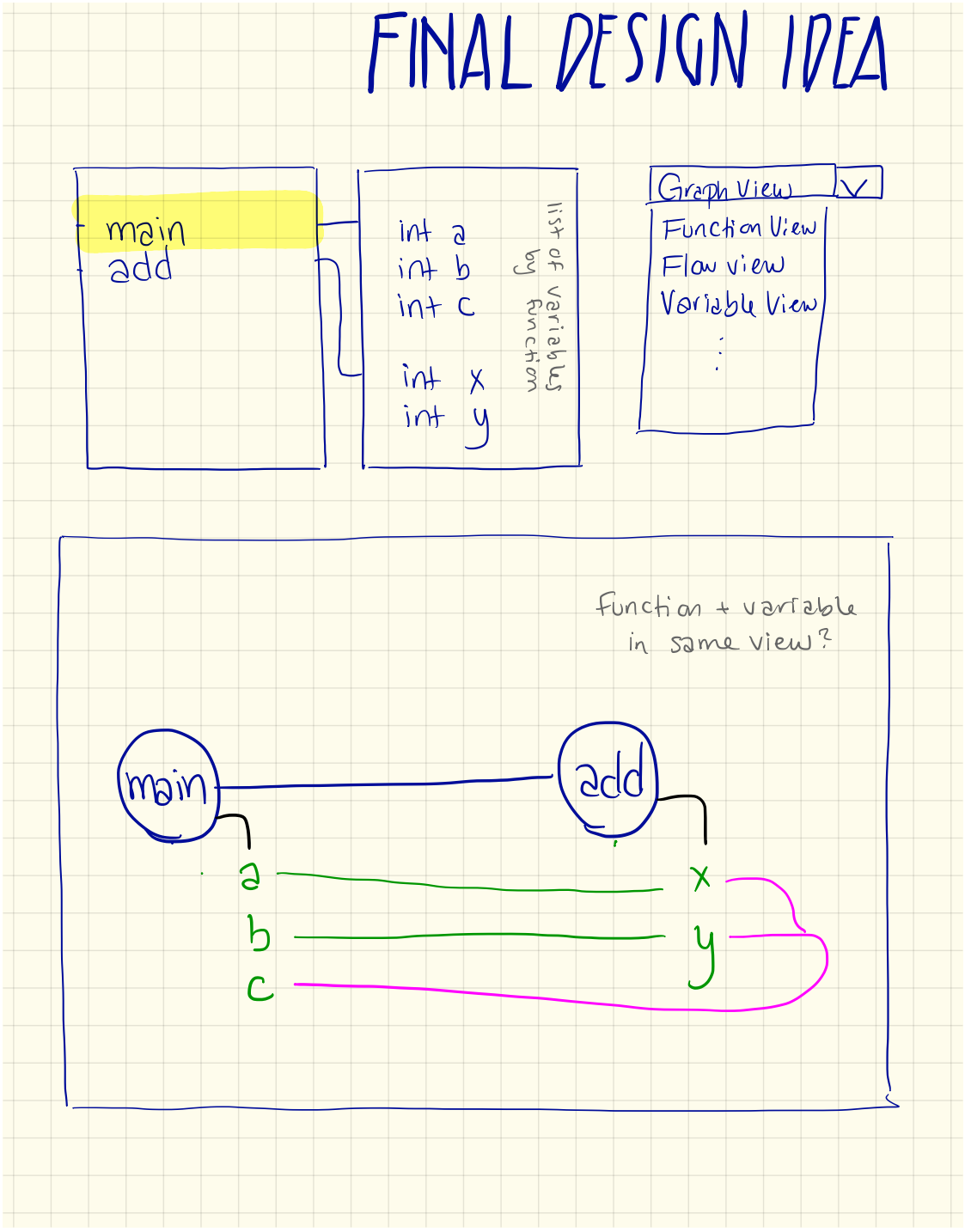

Figure 4: Final Design Idea

5 Features

5.1 Must have

At the very least, I'll need to have a graph (most likely a tree) showing, essentially, the call hierarchy. The interaction will work with some kind of list of functions and/or variables, should show the path(s) of all functions/variables it is dependent on, is dependent on it, and/or both.

5.2 Optional

I'd like to have multiple graph views. One for functions showing the call hierarchy, another for flow of the program (where each node would have an enter and exit connection). Similarly, I'd like a graph or multiple graphs where the nodes are variables, showing dependencies between variables with the program or a function.

I'm also considering including several lists/text boxes with interactivity between then. For example, at the very least, I'll need a list of all the functions and some list of variables. These could be connected so that when a function is connected, the variable list is sorted or filtered by variables within or affected by that function, and vice versa. It might also be helpful to include a text box with the original code for a selected function, and possibly highlights all calls and declarations of the functions or selected variables.

Ideally, the graph views would have some sort of level of detail processing for zooming in and out, with the most zoomed in view showing the code, and the most zoomed out view simply showing one node for `main()`.

Another thing I'd like to consider is including a class view, or a view that includes classes. I imagine this will become clearer as the must-haves become reality. In fact, I believe at least one of these optional features will be a must-have, though it's hard to know which it will be at this point.

6 (Rough) Project Schedule

6.1 Week One

Data processing and start building and displaying tree/graph (simple call hierarchy)

6.2 Week Two

Add list of variables and interactivity with the graph (highlighting relevant paths with relevant colors)

6.3 Week Three

Alternate graph view: functions as nodes, but rearranged to show data flow/flow of the program vs. call hierarchy

6.4 Week Four

Alternate graph view: variable graph (probably by function)

6.5 Week Five

I'll pick the most needed optional feature(s) for the final week