Program Visualization Tool
for Educational Code Analysis
Natalie Beams
University of Oklahoma, Norman, OK
nataliebeams@gmail.com

Program Visualization Tool for Educational Code Analysis

1 Introduction

Botball, an educational robotics program [2], introduces students from hundreds of teams around the world to computer programming. KISS-C, developed by the KISS Institute for Practical Robotics (KIPR), is a programming environment used for Botball competitions (and, to a lesser extent, for anyone desiring a simple, cross-platform, free environment for writing C programs). [3] Since the mission of KIPR and Botball is promoting hands-on learning, tools to enhance the KISS-C environment are being developed, with the dual aims of both aiding new programmers and furthering the capabilities of more experienced programmers. The current tool in development is a "Program Visualizer" tool.

The Program Visualizer is meant to give users a functional visual representation of their code, allowing them to "see" their program. The visual representation will look much like a flowchart, showing the logical path the program will follow when executed. This flowchart will be composed of items representing the major logical control points of the program: `if-else` statements; `switch` statements; `for`, `while`, and `do-while` loops; and function calls. It will not just be a simple representation, however; the Program Visualizer is meant to be an interactive tool, allowing users to select an item on the graphical representation and view and edit its corresponding code. The Program Visualizer will also contain an orderly "tree" list of the items which shows parent-child relationships for items that are nested inside one another.

Whether used to increase understanding of code written by someone else or to view (and perhaps find flaws in) one's own code, the Program Visualizer tool will provide another instructional resource for programmers using KISS-C.

2 Overview of the Tool

The current concept for the Program Visualizer has three main components: the graphics window, the program browser, and the text editor for the currently selected item.

The graphics window displays the actual visualization of the program, which will be represented as a flowchart. The graphics window is also interactive; clicking on an object in the flowchart will make it the currently selected item in the program browser.

The program browser contains a "tree" of all the key items found in the program, such as `if-else` blocks, `switch` statements, loops, and function calls. The tree will appear in a format similar to the familiar file browsers used to find and open files on modern computers. This should make the browser intuitive for users to navigate and understand.

The text editor displays all of the code for the item currently selected in the program browser. For example, if the item for a particular `for` loop is selected, then all of the code that is inside that loop is displayed. The text editor is also a way to communicate back to the source file in the main KISS-C editor window; changes made to an item's text can be incorporated into the source program at the user's command.

3 Development of the Tool

The principles used in making design decisions are as follows:

1. Above all, the Program Visualizer should be informative, helpful, and easy to use.
2. The Program Visualizer should be sufficiently interactive and give users options for viewing and modifying their code from within the Visualizer window.
3. The Program Visualizer should not require the code to be compiled before being visualized; this allows the Visualizer to be a visual debugging aid in the case of syntax errors.
4. The Program Visualizer should be able to be incorporated into the existing KISS-C framework as simply and easily as possible.
5. Use of additional external software should be avoided when possible, especially if it would significantly increase the amount of memory required to install and run KISS-C (this also stems from the previous item).

3.1 Methods for Implementation

KISS-C is written in C++ and utilizes Qt, Nokia's "cross-platform application and UI framework." [1] In accordance with guidelines 3, 4, and 5 mentioned above, it was decided that the advantages of previously developed scanner/lexer/parser tools were not great enough to warrant the addition of new external software. Instead, Qt's powerful `QRegExp` class is being implemented to create an entirely new scanner. (This also gives the added benefit of being able to completely customize

the code for the purposes of KISS-C.) The `QRegExp` class allows the user to define "regular expressions" for text patterns, search for these expressions in a block of text, and perform various other operations if the text is found. [4] Regular expressions are more powerful than simple text-matching functions because they can account for variable repetitions of characters; this means that one can allow, for example, zero or more spaces between the open parenthesis of a function call and the declaration of any parameters.

To create the program browser, Qt's `QTreeWidget` class has been utilized. The graphics window and all of the objects in it are handled with reimplementations of Qt's `QGraphicsScene`, `QGraphicsItem`, and `QGraphicsViewer` classes. All classes created for the Program Visualizer are contained in a separate library and loaded into KISS-C through the user-selected target (either his computer or a Botball device).

The overall visualization process is as follows:

1. Send entire source file text to the Program Visualizer.
2. Scan the file for functions and create a new object of the FunctionItem class (a member of the Program Visualizer library) for each function.
3. For each function, scan the text to find matches for regular expressions defining `if` statements, `switch` statements, loops, and function calls. (Each match is tested to make sure that it is not currently commented out.)
4. Take the earliest match and find its corresponding code.
5. Create a FlowchartItem object for the item that appears earliest in the code and add it to the function's list of such items.
6. Recursively search the text of this item for any new items, which will have the current item as a "parent."
7. Move the beginning search point to this location in the original file and repeat until the end of the file is reached.

3.2 Preliminary Tool Development and Testing

In order to demonstrate the layout of the preliminary version of the Program Visualizer, consider the simple test program on the next page:

```
int main()
{

    int x = 1;
    int y = 1;

    for(x = 1; x <= 20; x++)
    {
        printf("\nx = %d\n", x);

        while( y <=10)
        {
            y = double_value(y);
            printf("y = %d\n", y);
        }
        y = 1;
    }
}
```

where the `double_value` function is defined as

```
int double_value(int y)
{
    y = y*2;
    return y;
}
```

The initial layout for the Program Visualizer can be seen in Fig. 3 on the next page, along with partial output for the example program given above. (The graphics window is still in development at the time of writing, so output for this window has been suppressed in this example.)

The first step in the visualization process is finding all functions in the program and their definitions. In the case of the example program, the Program Visualizer finds two functions: `main` and `double_value`. The user can then freely choose which function is displayed in the Visualizer window, as shown in Fig. 1.

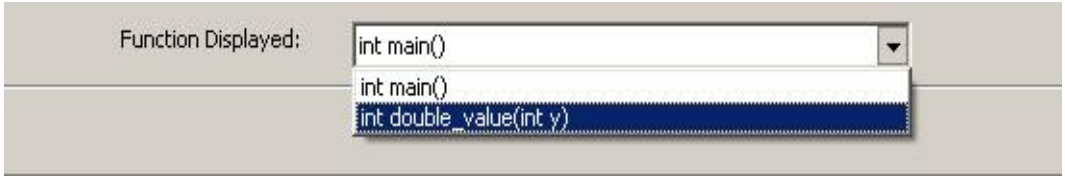

Figure 1: Selecting a function to view.

Within the `main` function, the Program Visualizer tool has found the `for` loop and its associated text. When that loop item is selected, as shown in Fig. 3, the loop's code appears in the text window at the bottom of the Visualizer window. After finding an item, the Visualizer uses a recursive algorithm to search the text of every item found for more keywords. The Visualizer then finds that there is a `while` loop nested inside the `for` loop, and this `while` loop is treated as a "child" of the `for` loop. This results in the clear hierarchical structure seen in the program browser window in Fig. 2. The finished Program Visualizer will also find the call to the `double_value` function and create an item for it; however, this capability had not yet been implemented at the time of this writing.

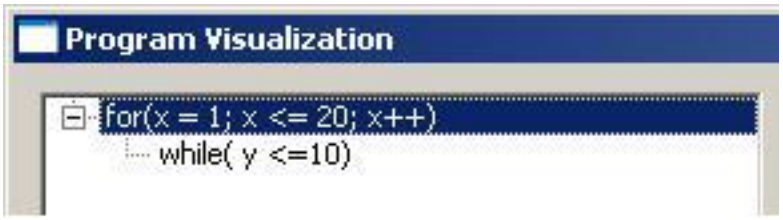

Figure 2: Nested items cause the Program Visualizer to create a hierarchical tree in the browser window.

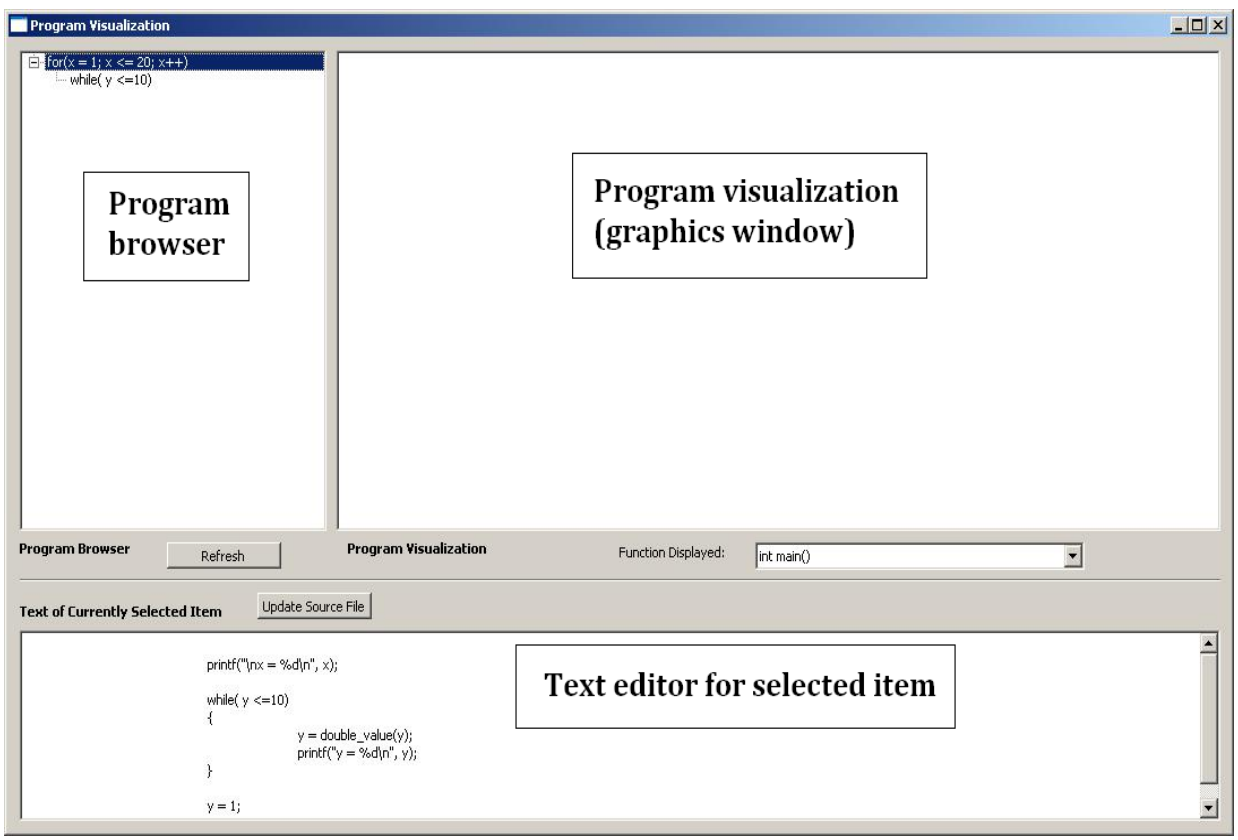

Figure 3: Initial layout for Program Visualization window.

4 Future Work

Further work will be completed to create and improve the capabilities of the Program Visualizer tool. A beta version of the tool will be presented to KISS-C users for testing. Feedback from these trials will be utilized to modify and improve the Program Visualizer tool and increase its usefulness for future programmers using KISS-C.

References

[1] Nokia Corporation. Products. http://qt.nokia.com/products, 2010.

[2] KIPR. Botball robotics education. http://www.botball.org, 2009.

[3] KIPR. KISS-C. http://www.botball.org/kiss, 2010.

[4] Trolltech. Qt 4.2: Qt Reference Documentation (Open Source Edition). http://doc.trolltech.com/4.2/index.html, 2007.