The Design of an Array Visualization
Tyler Robison
University of Washington
trobison@cs.washington.edu

Abstract
We present the visualization of an array for a visual programming language, designed with the qualities of transparency, clarity, visual appeal and efficiency in mind. These qualities are essential to making the array useful to our target users, people without significant programming experience or computer-science background. We then implemented the visualization in the visual programming language known as the Data Factory, and ran a preliminary experiment to help us evaluate the design.

1. Introduction
Visual programming languages offer an interesting alternative to traditional text-based programming, with the programmer expressing the program in a graphical format instead of typing in code. This approach allows users to program in a less syntactically rigid and perhaps more appealing way. As such, visual programming languages have the potential to offer a more accessible method of programming. They can therefore be attractive to those lacking formal training and extensive experience in traditional programming languages; these are our target users.

In order to make visual programming languages more powerful and useful to novice programmers, there must to be equivalent versions of many common programming constructs, such as loops or arrays. This work focuses on the latter. We proceed by laying out 4 qualities we believe are important to the usefulness of such an array visualization to non-programmers, and then describe the visualization we have designed to fulfill these. Finally, we conclude with the results of a preliminary experiment conducted around the array device.

Array visualizations have been implemented before but here we focus on visualizations that are transparent with regard to process, are easy to understand and are visually appealing to the user; we formalize these qualities later. One instance of an existing visualization can be found in LabView [1], but the view of the data is driven by the user, so the program’s operations on the array are not always visible to the user. Visualizations of lists in general are seen in many different places but our usage differs from most of these; here we consider an array viewed and manipulated not only by the user, but by the program as well. How do we draw an array write operation to the user’s attention? How do we balance giving the user control over the display with showing the user what the system is doing? Our visualization is designed to address such concerns.

2. Formalization
Before going any further it is important that we formalize our use of the word ‘array’. By array we refer to an arbitrarily sized list where each index can have values read from and written to it. These two operations, ‘read’ and ‘write’, are the only ones considered, and the arrays we are concerned with here are one-dimensional.

3. The Relevant Qualities
Our design is intended to be not only useful but also attractive to people with little or no programming experience, and so it is essential that it be easy to use, and that no important information is hidden from the user. Making sure that all important data and operations are shown up-front helps ensure that the user has a complete view of the array, avoiding the confusion that might result if only a narrow view were given. Furthermore, we would like the device to be both interesting and appealing, and to make efficient use of space; to be successful, the array must be as inviting as is possible. We formalize these goals below in our four qualities.

1) Transparency: The idea that nothing is concealed from the user. All data is accessible to the user, and all operations performed on the array by the program are brought to the user’s attention.
2) Clarity: The idea that the visible information is easy to learn and understand.
3) Visual Appeal: The idea that the visualization is both interesting and attractive to the user.
4) Efficiency: The idea that the array is spatially compact.

These qualities have a strong emphasis on making sure that the user can see and understand what is
going on in the array. Note that here we do not consider qualities such as speed or power. While important for many users and applications, here we are more concerned with making the device simple, appealing and accessible to people without programming backgrounds.

4. The Visualization

With these qualities in mind we constructed our array visualization, shown in Figure 1. One of the most conspicuous features is that the most recently read or written index is displayed with a certain size, and its neighbors are shown increasingly smaller as they get further away. We use this both to draw attention to the index and to conserve space, helping to provide transparency (as the user’s attention is drawn to the index being used), clarity and spatial efficiency.

There is, however, more to an array visualization in a visual language than what can be shown in a static image. The array is one component in a program, and so the display must be updated to reflect new data being written to the array, and read operations from the array. The user must be notified when a read or write occurs. Such transparency of process was one of our most important design goals. To ensure this transparency, whenever a read or write occurs the array undergoes an animated transition to the target index. An abrupt transition could be very confusing (especially if reads and writes were performed in rapid succession), and so we use a smooth transition from the current index to the destination. As an added benefit, a smooth animation is more visually appealing than an abrupt transition.

5. The Experiment & Conclusions

In order to evaluate our design, we implemented our visualization in the visual programming language known as the Data Factory [2], and ran a preliminary experiment with a small number of subjects. In this experiment we had the subjects inspect a series of Data Factory programs. Some of these programs used our visualization while the others used a ‘black-box’ array. This black-box array was functionally the same as our array (it used the same inputs and outputs, and had identical operations), but had no visualization. Apart from watching the inputs and outputs of the device, there was no way to determine the state of the array.

The comments gathered from the subjects and the observations of the subjects as they interacted with the program were the most valuable results of the experiment. In general the subjects found the visualization to be helpful; the one exception was a person with vision problems who found the animation to be too distracting and difficult to follow. Overall, the experiment suggested that the visualization is useful, and that it fulfills the qualities listed above.

One limitation of the experiment was that it looked at our visualization versus the black-box, and as such it did not evaluate the individual components of the design, but rather the design as a whole. In the future we hope to investigate the design in more detail, with experiments selected to test different aspects of the design.

6. References
