Democratizing Computational Tools for Interaction Designers

Stephen Oney
Human-Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213 USA
soney@cs.cmu.edu

Abstract

I am creating a new programming language and editor that is aimed towards authoring interactive behaviors. This language is intended to allow more interaction designers to write their own interactive applications. This paper discusses the motivation, method, and design ideas for such a language.

1. Introduction

Interaction designers interested in working with the virtual medium of computer applications face significant challenges that often hinder their ability to express their desired behaviors. One of the barriers interaction designers face is the effort that it takes to learn a programming language to the point that they can implement fully functional applications or prototypes. Another challenge is that computer application programming does not allow for reflection in action, which is roughly the act of evaluating one's work before completion. Whereas in the course of working with physical materials, the designer is able to evaluate their work before completion, they cannot with traditional computer programming languages.

While many tools have been created to allow designers to create interactive software, including Adobe Catalyst, Microsoft Blend, and Processing, these tools still rely on a fixed set of widgets for ease of use. Although widgets can often provide a decent approximation of what an application designer wants, they often are not able to capture exactly what they want [2]. The application designer might, for example, want to make a slight tweak to a supplied widget, but this usually requires re-implementing the widget from scratch in a programming language like ActionScript, which is not a task that is possible for most interaction designers.

Put another way, most existing tools have placed an emphasis on parameterization, or providing widgets and guessing which parameters application designers might want to vary. However, in our research, we have found that designers often want complex behaviors that widget creators cannot predict and parameterize for. For this reason, I am focusing on creating a new programming language for interactive applications, rather than a new tool on top of an existing programming language, to allow more interaction designers to author interactive behaviors.

2. Previous and Related Work

I have helped develop two tools aimed at helping interaction designers: FireCrystal [3], which is aimed at allowing interaction designers to understand and duplicate interactive behaviors on the web; and Playbook, which is a tool that uses a simple language and programming-by-demonstration to help designers evaluate and compare early-stage interface prototypes.

Our research group has also done several studies on interaction designers and the needs of tools to support interaction design. One study showed that most interaction designers have to rely on programmers to prototype or implement their designs [2], which highlights the need for tools that enable more interaction designers to author their own behaviors. Another [4] investigates the requirements of an environment for authoring interactive behaviors.

3. Method

Having already compiled a list of needs of interaction designers in terms of features, we are now exploring language primitives. To go about this, we are looking at existing interactive applications with "complex" behaviors, where "complex" means difficult to implement in existing languages. We are examining complex behaviors, because we feel they represent samples of behaviors that designers are truly interested in, rather than a compromise between the designer's intention and the tools available to them. We are also examining the behaviors of standard widgets, like buttons and scrollbars, to make sure they have a representation that easy to understand. As mentioned in the introduction, we want to focus on making the language easy to understand, rather than providing a more complex language and simple widgets that can be parameterized, with the aim of allowing more interaction designers to author custom interactive behaviors.

One promising approach we have found has been to revisit the spreadsheet paradigm explored by environments like Forms/3 [1]. Spreadsheet and dataflow languages represent one way to improve the reflection in action component, because continuous evaluation allows consequences to be immediately seen. We want to augment this by reducing some of the known deficiencies of spreadsheets, like lack of visibility, and by making our programming environment multimodal, so that multiple views of a program can be used to manipulate the logic. For example, when designing program logic, a state machine might be the best view of the program; when defining program dependencies, a spreadsheet; when customizing animations, a timeline, etc.

Another important feature is that we want behaviors to be localized, and to minimize "spaghetti code". Not only does this make programs simpler and more modular, it also allows for easier exploration because designers can easy add, remove, or modify behaviors. After a set of language features is finalized, we plan on doing Wizard of Oz prototypes with interaction designers before implementing the language.

4. Conclusion

To enable more interaction designers to program interactive applications, we plan on creating a programming language and environment tailored to interactive application design.

5. References

1. Burnett, M., Atwood, J., Djang, R. W. *et al.* Forms/3: A first-order visual language to explore the boundaries of the spreadsheet paradigm. *J. Funct. Program.,* vol. 11, no. 2, (2001), 155-206.
2. Myers, B. A., Park, S. Y., Nakano, Y. *et al.* How Designers Design and Program Interactive Behaviors. *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, (2008), 177-184.
3. Oney, S., and Myers, B. FireCrystal: Understanding Interactive Behaviors in Dynamic Web Pages. *Visual Languages and Human-Centric Computing*, (2009 (to appear)),
4. Ozenc, F., Kim, M., Zimmerman, J. *et al.*, "How to Support Designers in Getting Hold of the Immaterial Material of Software," 2010.