# Development Tools for Interactive Behaviors

Stephen Oney

Carnegie Mellon University
Human-Computer Interaction Institute
5000 Forbes Ave. Pittsburgh, PA, USA 15232
soney@cs.cmu.edu

**Abstract.** This research uses participatory design workshops and user-centered design with trained interaction designers to guide the development of a new programming language and environment for creating interactive applications. Interactive behaviors, which define the operation of an interactive application, are often difficult for interaction designers to program because many interaction designers do not have formal programming training and many features of interactive behaviors make the task of programming them distinct from, and often more challenging than, other programming tasks. This research aims to create a programming language and environment that is tailored to the needs of interaction designers and that alleviates the problems that make programming interactive behaviors difficult.

**Keywords:** end-user programming, interaction design

## 1 Introduction

Rogers, Sharp, & Preece define interaction design as "designing interactive products to support the way people communicate and interact in their every day and working lives." [1] Interaction designers are often tasked with designing novel and complex interactive software as part of their job. The medium of software presents a unique challenge for interaction designers who are interested in writing interactive applications. Unlike other designers, who work with their materials in a studio or workshop, interaction designers are not able to engage in meaningful reflection-in-action [2] (which means to evaluate and generate ideas while in the process of creating) when designing interactive software. In addition, the threshold of programming knowledge required to programmatically create new interactive behaviors is prohibitively high for many interaction designers, which often forces interaction designers to rely on professional developers to program the interactive behaviors they design.

In my research, I am interested in investigating ways to enable and encourage interaction designers themselves to design and *develop* interactive software. This means accounting for not only the needs of interaction designers, but also analyzing the features of interactive behaviors that make them difficult to program in traditional programming languages and exploring ways to lower the threshold for creating interactive software. This is a form of End-User Development since interaction designers are authoring code, but they are not professional programmers.

## 2 Background

A recent survey of interaction designers shows that while interaction designers find it more difficult to prototype and implement the *feel* of an interactive application than the *look* [3]. Further, 78% of the participants in this survey indicated that designing interactive behaviors requires collaborating with a developer. Designers often communicate a design to a developer through annotated design sketches and storyboards, but they indicated that communication breakdowns are frequent [3].

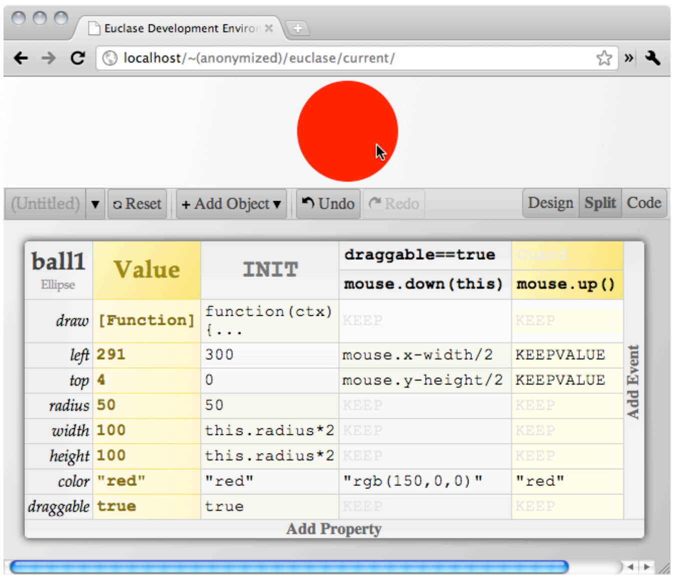
Even putting aside the possibility of communication breakdowns and the cost of having to collaborate with professional developers, relying on another team member to prototype in implement their interactive behaviors reduces their potential for reflection-in-action and to iterate on and evaluate their design. So why do not more interaction designers learn to program? In a different survey, they pointed to the high learning curve, time consumption, the difficulty of creating novel interfaces, problems with generated code, and toolset limitations as weaknesses of various programming languages and environments [4]. Additionally, from a software engineering perspective, the task of writing interactive applications presents a unique set of challenges [5], as I will outline in the next section.

## 3 Research Approach

This research includes participatory design workshops conducted with interaction designers to gain insight into design requirements, the design of the language & environment, and evaluation & iteration through evaluative user studies of environment prototypes.

In the participatory design workshops, conducted with fourteen interaction designers and programmers with at least two years of professional experience, and described in detail in [6], designers indicated the need to better evaluate their designs, the importance of examples for exploration and communication, and of programming tools that can keep track of design rationale.

In designing the language and environment, I focused on five features of interactive behaviors that make creating interactive applications difficult. First, interactive behaviors are usually graphical in nature, and while it is relatively easy to declaratively specify the look of an application, imperatively writing a graphical application that controls how it operates is difficult. Second, interactive behaviors are often state-oriented; their behavior may be dependent on a combination of global and local states. Third, interactive behaviors are often constraint-heavy; conceptually, there are often constraints that update a view based on some underlying model, and there are constraints on the layout of elements in the view with respect to each other. Fourth, interactive behaviors are often event-based, as they react to user input.



| ball1 Ellipse | Value | INIT | draggable==true / mouse.down(this) | Guard / mouse.up() |
|---|---|---|---|---|
| *draw* | [Function] | function(ctx) {... | KEEP | KEEP |
| *left* | 291 | 300 | mouse.x-width/2 | KEEPVALUE |
| *top* | 4 | 0 | mouse.y-height/2 | KEEPVALUE |
| *radius* | 50 | 50 | KEEP | KEEP |
| *width* | 100 | this.radius*2 | KEEP | KEEP |
| *height* | 100 | this.radius*2 | KEEP | KEEP |
| *color* | "red" | "red" | "rgb(150,0,0)" | "red" |
| *draggable* | true | true | KEEP | KEEP |

**Fig. 1.** A representation of a draggable red circle. The top half of the window shows the "design" view, while the bottom half shows the "code" view. In the code view, attribute names are shown in the far left column. The current values of the attributes are shown in "Value" column. Initial values are shown in the `INIT` column. The subsequent two columns specify constraints that will hold in various states: the fourth column specifies constraints that will hold when the user is dragging the circle (to constrain the center of the circle to always be the mouse location), and the last column specifies constraints that will hold after the user stops dragging (`KEEPVALUE` keeps the current value but gets rid of the constraints that were in place when the user was dragging the circle.)

Finally, interactive behaviors are often integrated with animations, and coordinating the behavior with the animation is often a significant challenge.

## 4 Progress

The current iteration of a prototype of our language and environment is shown in Figure 1 above. Interactive behaviors are written declaratively; every object has a set of attributes that can be static values (`3`, `red`, etc.) or constraints (`this.x+foo.bar`, `max(a,b)`, etc.). Attributes of objects are represented by rows in the object. Events are represented as columns, with the values in a column specifying the constraints that will hold after that event occurs. Our prototype is implemented in client-side Javascript and provides immediate feedback after the user updates the code. An initial user test conducted with interaction designers showed promise; interaction designers took advantage of the immediate feedback that our prototype provided when they updated their code. They also were very willing to experiment in their code. In fact, some designers asked for more immediate feedback where the design view would update as they were typing code.

For future work, I plan on making additions to the prototype, including a timeline view for specifying and coordinating animations, a state-flow diagram view to illustrate states and transitions between states, and improving the usability of the language syntax through iterative usability evaluation. Another addition that I plan to make for the environment is to create an "open box" widget set. One of the strengths interaction designers see in tools like Adobe Flash Catalyst is the availability of widgets to help them get started [4]. However, widgets in such tools are usually inflexible, and cannot be customized. An open box widget set would include pre-provided widgets that encourage designers to extend and customize them.

## 5 Impact for End-User Development

This research will result in the creation of, and design recommendations for programming languages and environments for creating interactive behaviors. The two-dimensional representation that the current prototype uses is a unique contribution that may prove to be a simpler representation for interactive behaviors than the style of imperative code used by C-derived languages like Processing and OpenFrameworks[1]. Although previous research has focused on providing widgets, or programming-by-example tools to reduce the threshold of creating interactive applications, my research focuses on the underlying representation of interactive behaviors. While interaction designers have played a large part in the design of this environment, its usefulness will likely extend beyond interaction designers. I plan on releasing the development environment for general use over the web.

**Acknowledgments.** I thank my advisor, Brad Myers, for his continuing guidance and support. The author is also supported by the Ford and ARCS foundations. This research has been partially funded by NSF grants IIS-0757511 and CCF-0811610.

## References

1. Rogers, Y., Sharp, H., Preece, J.: Interaction Design: Beyond Human-Computer Interaction. John Wiley & Sons, Ltd., West Sussex, England (2007).
2. Schön, D.: The Reflective Practitioner: How professionals think in action. (1983).
3. Myers, B.A., Park, S.Y., Nakano, Y., Mueller, G., Ko, A.J.: How Designers Design and Program Interactive Behaviors. VL/HCC pp. 177-184 (2008).
4. Carter, A., Hundhausen, C.: How is User Interface Prototyping Really Done in Practice? A Survey of User Interface Designers. VL/HCC, pp. 207-211 (2010).
5. Letondal, C., Chatty, S., Phillips, G., Fabien, A., Conversy, S.: Usability requirements for interaction-oriented development tools. Psychology of Programming, (2010).
6. Ozenc, F.K., Kim, M., Zimmerman, J., Oney, S., Myers, B.: How to support designers in getting hold of the immaterial material of software. CHI, pp. 2513-2522, (2010).

[1] http://processing.org/ & http://www.openframeworks.cc/