Visions for Euclase: Ideas for Supporting Creativity through Better Prototyping of Behaviors

Position paper for the ACM CHI 2009 Workshop on Computational Creativity Support

Stephen Oney, Brad Myers, and John Zimmerman
Human-Computer Interaction Institute
Carnegie Mellon University
Pittsburgh, PA 15213
{soney,bam,johnz}@cs.cmu.edu
http://www.cs.cmu.edu/~NatProg/

ABSTRACT

Our research is investigating how to allow designers and other creative professionals to easily prototype and create interactive computer applications and web sites. In this paper, we discuss several studies we have conducted to better understand the requirements of an environment to support the authoring of interactive behaviors by creative professionals. Then we summarize our proposal for a new environment that tries to address those requirements. This environment would include the ability to explore multiple designs, support for collaboration, and the use of metaphors that better support the creative process. Finally, this paper poses questions related to computational support of creativity.

INTRODUCTION

In this paper, we will focus on the Natural Programming group's preliminary work on Euclase, an environment that enables *designers* to create *interactive behaviors*. The term *designer* is used with regard to a user's intention (planning the look and feel) rather than profession – a designer might be an interaction designer or a professional programmer. We define *interactive behaviors* as ways that applications respond to users. Interactive behaviors are concerned with the *feel* of applications, rather than the *look*. Euclase stands for **E**nd **U**ser **C**entered **L**anguage, **A**PIs, **S**ystem, and **E**nvironment.

Unlike the look of an application, which can be modeled in Photoshop or on paper, the feel of an application usually must be modeled with complex programming. The few applications that address the feel at all, such as page transitions in Dreamweaver, are limited to a small fixed set of behaviors, which in turn limits the designer's creativity for what can be expressed.

In initial studies performed by members of our group, we found that the current tools for designing interactive behaviors are inadequate [6]. In addition, we found that designers use a common set of phrases to describe some simple behaviors, which might help form the foundation of the syntax for expressing these behaviors [8].

We are approaching the problem of creating a new environment by classifying the prototyping of interactive behaviors as a form of *end-user programming* (EUP) [10]. End-user programming refers to writing programs to support some other activity, rather than where the program itself is the user's job. Because EUP is defined as a goal, someone could be considered a professional programmer and an end-user programmer in different situations.

We believe that designers should be considered end-user programmers because their primary goal is to prototype designs – not to write the program itself. Their ability to explore and share possible designs is much more important than implementation details and minor performance optimizations.

Our main goals for Euclase are to allow for easy exploration and collaboration. We have proposed strategies for achieving this, and we believe our group's process will be beneficial to this endeavor.

ABOUT THE AUTHORS

Stephen Oney is beginning his research in the field of end-user programming as a first-year PhD student at Carnegie Mellon University. His previous research was in natural language processing and machine learning at MIT.

Brad Myers is the head of the Natural Programming group at Carnegie Mellon. He has led multiple projects pertaining to end-user programming and programming-by-example. In addition, he helped organize the NSF workshop on creativity in 2005.

John Zimmerman is an interaction designer with a joint appointment in the Human-Computer Interaction Institute and School of Design at Carnegie Mellon. His main research areas include the role of design in HCI research and mixed-initiative interfaces that leverage the strengths of humans and computers.

NATURAL PROGRAMMING

The overall goal of the Natural Programming group is to apply usability principles to programming environments and languages, so as to make programming a more natural and intuitive exercise [5]. One of the most important goals of the Natural Programming group is to enable EUP, so that the difficulty of any programming task may be proportional to the complexity of the task.

HANDS is an example of a tool designed especially for end-user programmers [7]. Created by John Pane, a former member of our group, HANDS allows children to express solutions to problems without advanced programming knowledge. By using the metaphor of cards as variables, built-in support for groups of objects, and other features, HANDS made end-user development more intuitive [7]. Many of the lessons from HANDS are relevant to the design for our new language and environment.

We plan on leveraging our experience in creating development environments for EUP, as well as taking advantage of interdisciplinary collaborations with psychologists and professional designers when creating Euclase.

EUCLASE

Most of today's tools used by applications designers to prototype are more focused on designing the *look* of prototypes, rather than the *feel*. In programs like Flash and Dreamweaver, programming custom behaviors still requires advanced knowledge of languages such as JavaScript and ActionScript.

The few applications that allow for some prototyping of behaviors through façade tools and interface builders do this by giving the user a library of widgets (as in Microsoft Visual Basic). However, the creativity of users is limited by the selection of widgets. In addition, further customization of these widgets, if even possible, still requires advanced programming knowledge. A similar approach is taken by the new Adobe Flash Catalyst, which gives designers a library of predefined behaviors on which they can overlay graphics.

Because prototyping of interactive behaviors requires the use of advanced programming languages, the creation of these prototypes is limited to people with programming experience. Usually, the process seems to involve designers drawing up interface sketches, with rudimentary descriptions of the interactive behaviors, sending these sketches to a programmer, and having the programmer implement a computer representation of the design. In fact, the whole philosophy of the Microsoft Expressions environment seems to require this style. However, one of our studies found that most designers have trouble communicating their desired interactive behaviors to programmers [6].

Thus, the goal of the Euclase project is to enable non-programmers, especially interaction designers, to create and prototype interactive applications themselves. Put another way, we are seeking to eliminate the barriers to implementation that inhibit designers from actually creating the interfaces they are trying to design, and trying to encourage designers to create, prototype, and explore different designs themselves without the need for a professional programmer. We believe that this will allow designers to better translate their ideas into reality, and enhance their creativity.

PRELIMINARY STUDIES

In 2005, we helped organize an NSF workshop on creativity support tools, to better understand how development and design tools can foster creativity [11]. One of the results of this workshop was an enumeration of design principles that lead to innovation, including support for exploration, collaboration, and designing for designers.

This was confirmed by our study of over 200 interaction designers [6] that formed the start of the Euclase project. One of the main findings of this study was that designers overwhelmingly have trouble designing and communicating the behaviors they want – much more so than the appearance, which they frequently sketch or prototype in environments like Photoshop. In addition, many designers said they wanted to explore many possible designs, but are inhibited by the current tools.

In order to address the difficulties our study found in current prototyping environments, and provide the kinds of tools that the workshop found to foster creativity, we have developed the following goals for Euclase:

1. Support collaboration and better communication of interactive behaviors.
2. Allow for and encourage exploration by supporting side-by-side comparison, keeping track of artifacts from previous designs, and enabling easy use and integration of examples.
3. Lower the learning curve by using metaphors and a natural syntax.
4. Integrate debugging as a component of the interface, so that debugging may be part of the design process.

Some of our ideas on how to achieve these goals are summarized in the following sections.

IMPROVING COLLABORATION AND COMMUNICATION

As mentioned, one of the key conclusions of our study of the needs of designers is that it is difficult to communicate the feel of interactive applications. Thus, one of our goals for Euclase is to incorporate features for collaboration and communication directly into the interface.

While the tools used by developers and designers to communicate, such as versioning systems and e-mail, can be generalized to multiple languages and disciplines, they lack features specific to any language that would be beneficial for collaboration. In Euclase, we want to keep track of artifacts made by multiple users during the design process, and allow users to pick, extract, and incorporate features from older designs by any of the designers. This feature is further discussed in the "encouraging exploration" section.

Even when the designers themselves ultimately implement the behaviors they create, it is still crucial to give them the ability to annotate their prototype for use as design rationale for better understanding of the prototype later. Thus, one idea we have for improving communication of behaviors and their rationales is to allow for annotation on top of the user interface, rather than just on the code for implementation details. These annotations will allow multiple users to describe why design changes were made.

Incorporating these features directly into the interface of Euclase will lower the barrier to collaboration and communication between designers, and will likely increase participation and the volume of ideas generated during the design process.

ENCOURAGING EXPLORATION

As concluded by the NSF workshop on creativity, allowing for exploration of multiple design possibilities is one way to encourage creativity [11]. There are three main ways we plan on encouraging this sort of exploration: enabling easy integration of examples, allowing for side-by-side comparisons, and keeping track of multiple versions of designs.

Because there are many common behaviors which designers will likely find useful in prototyping their applications, we plan on providing examples of interactive behaviors in Euclase, and making it easy to capture examples found on the Internet. Examples of interactive behaviors can serve three roles for designers. First, examples can provide a wealth of inspiration for new design ideas. Second, using such example behaviors would provide pre-built components from which designers could capture interesting behaviors, and third, examples with their corresponding source code make it easier for designers to learn how to achieve desired effects.

In the past, most research on allowing for integration of found examples has been focused on advanced programmers, enabling them to look at and learn from code snippets or example code. One example is our Mica tool, which helps programmers find example code in Java [12]. Another such tool is EG, which is also focused on Java code and allows developers to incorporate, customize, and test example code [1]. However, there is no equivalent for prototyping behaviors, or even other types of EUP.

End users interested in creating an original website are frequently influenced by ideas for interactions and layouts found at other websites. However, when they see a layout or a behavior that they like and want to use in their website, they have no way of extracting it out without extensive knowledge of multiple web languages - HTML, CSS, and JavaScript. Thus, we plan on investigating ways to enable users to extract behaviors from other websites, and customize them for use in their own creations.

Another crucial aspect in enabling exploration is to allow the comparison of multiple possible designs, so that the merits and faults of explored designs may be highlighted. Because we are focused on *interactive* behaviors, one possibility is to allow users to interact with multiple behaviors by having side-by-side views of how different revisions of interfaces would handle a particular user input.

Finally, one of the main problems that designers have with exploration in existing prototyping frameworks is the difficulty in reversing unsuccessful modifications. While most editors for languages like JavaScript have basic undo/redo functionality, it is very difficult to keep track of what each undo action does. In addition, undoing intermediate steps can result in non-functional code. Thus, exploration is discouraged for fear that a working version of the prototype might be lost. In Euclase, we want to automatically save working prototype states, and make representations of different working versions of the prototype plain to see for easy access, and to let the user know what state each undo action would return them to. Because each version will be an incremental change, we could use a pictorial or interactive representation of how behaviors incrementally changed over time, and create an undo timeline.

SYNTAX AND REPRESENTATION

In looking for possible natural representations of interactive behaviors, we conducted an additional study which showed designers some behaviors, and asked them to explain the behavior in plain English [8]. We learned that there is little variation in their descriptions for the behaviors that were shown in the study. We plan on using the results of this study, and others, to guide the creation of a behavior prototyping language for Euclase.

In addition, previous projects like HANDS have looked into what is the most natural representation for common programming tasks, such as performing an operation on a collection of objects. Also, Gamut looked into ways to infer conditional expressions in a programming-by-demonstration system [3].

We plan to leverage our studies and past experiences to create an intuitive and easy to learn syntax for Euclase. In addition, we plan to use other metaphors to help visualize program behavior. For example, we found in our survey of designers that storyboards are a popular way to describe behaviors. Thus, we will consider metaphors such as showing behaviors on storyboards, with symbols representing particular kinds of movement.

DEBUGGING

Because exploratory design is largely experimentation, we believe it will be beneficial to integrate debugging into the development process. This would support “debugging into existence” [9] – using debugging information to guide development. This is difficult in languages such as JavaScript, where debugging can only be done while testing the application. Instead, we want to incorporate iterative execution, so that users’ actions take effect immediately. This model has worked well for languages such as Smalltalk [9].

We also want to address the debugging needs of users who have undesired behaviors in their prototypes. We plan to investigate the needs of users while they are debugging, and center our debugging strategy on the users' needs. One approach would use Andrew Ko's Whyline [2]. This would allow designers to ask "why" and "why not" questions about their prototypes; a debugging technique that led to significant gains in effectiveness of debugging in ALICE and Java [2].

RESEARCH AND WORKSHOP QUESTIONS

Research into Euclase has generated many interesting questions, some of which are applicable to the field of computational support of creativity as a whole. One question deals with the possibly conflicting values of creativity and consistency. In designing applications, for example, designers and developers value creativity. From the point of view of a user, however, using original designs might be difficult to learn compared to the reuse of older designs. This is still a problem in disciplines outside of interface design – even in musical composition, a certain level of consistency seems to be deemed positive, as evidenced by common use of techniques like counterpoint. Thus we ask: what techniques would allow support of creativity while maintaining the appropriate level of consistency?

Another relevant research question is the role of examples in creativity. Examples simultaneously serve as a source of inspiration and a rough how-to guide for creative professionals. When using an example, the relevant parts of the example must be isolated. Thus another research question is: how we can isolate the relevant parts of examples so that they might be more useful for designers?

CONCLUSION

We have started creating a development environment that will allow designers prototype interactive behaviors without professional programming knowledge to. We believe that by incorporating collaboration, exploration, a natural syntax, and easy debugging into our environment, we can allow designers to be more creative in their design than the current generation of tools.

While we believe our technical background is suited to designing the features of this environment, we recognize that the challenge of encouraging creativity is a multidisciplinary one. Thus, we are approaching it through a collaboration among computer science, design, and psychology.

ACKNOWLEDGEMENTS

This work would not be possible without the help of Bonnie John, Andrew Ko, Sun Young Park, Yoko Nakano, and Greg Mueller. We would also like to thank the ARCS Foundation. Adobe has provided wonderful financial support and advice. This research was also supported by the NSF under Grant No. IIS-0757511. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the National Science Foundation.

REFERENCES

1. Edwards, J. "Example Centric Programming." *OOPSLA 04 Companion Guide*, Oct 2004, pp. 124
2. Ko, A. and Myers, B. "Designing the Whyline: A Debugging Interface for Asking Questions about Program Failures." *CHI 2004*, pp. 151-158
3. McDaniel, R. "Building Whole Applications Using Only Programming-by-Demonstration." *Ph.D. Thesis, Carnegie Mellon University, Computer Science Department.* 1999
4. Myers, B., Ko, A., and Burnett, M. "Invited Research Overview: End-User Programming" *Extended Abstracts*, CHI 2006 pp. 75-80
5. Myers, B., Pane, J., and Ko, A. "Natural Programming Languages and Environments" *Communications of the ACM*. Sep. 2004, pp. 47-52
6. Myers, B., Park, S., Nakano, Y., Mueller, G., and Ko, A. "How Designers Design and Program Interactive Behaviors," *IEEE Symposium on Visual Languages and Human-Centric Computing*. Sep 2008, pp. 177-184
7. Pane, J. "A Programming System for Children that is Designed for Usability." *Ph.D. Thesis, Carnegie Mellon University, Computer Science Department*. 2002
8. Park, S., Myers, B., and Ko, A. "Designers' Natural Descriptions of Interactive Behaviors" *IEEE Symposium on Visual Languages and Human-Centric Computing.* Sep 2008, pp. 185-188
9. Rosson, M. and Carroll, J. "Active Programming Strategies in Reuse." *EECOOP 1993 – Object Oriented Programming.* pp. 4-20
10. Scaffidi, C., Shaw, M., and Myers, B. "Estimating the Numbers of End Users and End User Programmers," in *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05)*. 20-24 September, 2005. Dallas, Texas: pp. 207-214.
11. Shneiderman, B., Fischer, G., Czerwinski, M., Resnick, M., Myers, B., et al. "Creativity Support Tools: Report from a U.S. National Science Foundation Sponsored Workshop," *International Journal of Human-Computer Interaction* 2006, pp. 61-77
12. Stylos, J. and Myers, B. "Mica: A Web-Search Tool for Finding API Components and Examples." *IEEE Symposium on Visual Languages and Human-Centric Computing* Sep 2006, pp. 5-7; 195-202