Designers' Natural Descriptions of Interactive Behaviors

Sun Young Park
School of Design
Carnegie Mellon University
sunyoung@cmu.edu

Brad Myers, Amy J. Ko
Human Computer Interaction Institute
Carnegie Mellon University
bam@cs.cmu.edu, ajko@cs.cmu.edu

Abstract

While a designer's focus used to be the design of non-interactive elements such as graphics or animations, today's designers deal with various levels of interactivity such as mouse, keyboard and touch screen interaction. Unfortunately, it is challenging for designers to create these diverse interactions since most implementation tools such as Flash require the use of conventional programming languages and do not support the natural expressions used by designers. To better understand how designers think about interactive behaviors, we conducted a lab study where designers and programmers described various primitive and composite interactive behaviors using their own language. From this, we learned that there is significant commonality among designers in terms of the verbs, syntax, and structure when describing interactivity. These results can help guide the way to building more natural programming languages and environments for designers to facilitate the development of interactive behaviors.

1. Introduction

Most of the interactivity designed by interaction designers involves pointer input, graphical objects, and relationships between these over time [3]. Unfortunately, current commercial tools for interactive behaviors seem to be focused on two approaches: either the designer is given a very limited selection of behaviors to select from a menu (such as the roll-overs and page transitions in Dreamweaver), or else the designer is assumed to only work on the appearance, with the behavior being created by a programmer using a conventional programming language (this is the apparent workflow of Microsoft's Expression Blend). Unfortunately, it is challenging for designers to explore the diverse interactive behaviors that they want using either of these approaches.

Is there a way to make the programming easier for designers, while still supporting the expressive range that they desire? Part of answering this question is understanding how designers *describe* interactive behaviors. In this paper, we report a study investigating how designers express behaviors with words given a graphical prompt (see Figure 1). In addition, because our prior study [3] showed that communication with programmers is an important part of the process of designers' work, the new study compares the results from designers and programmers to see where their expressions for behaviors are the same and where they differ. The extent to which programmers and designers do not agree will help assess the applicability of our results on different developer populations.

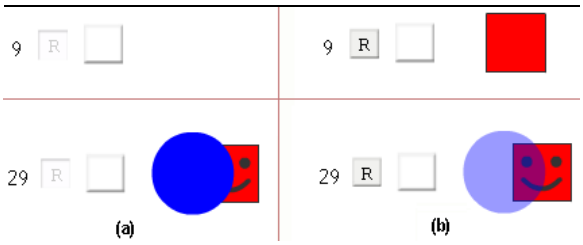

Figure 1: Two examples from our study of before the user clicks the button (a), and after (b). For #9, almost everyone used the same language: "the red box appears", but for #29, the language varied significantly ("fades", "becomes transparent", "opacity goes down", etc.).

2. Related Work

Studying people's use of natural language to inform the design of a programming language is not new. The psychology of programming literature [2] and previous studies have shown that this is possible and can make programming easier. For example, HANDS was successfully designed for kids programming [4] and Click! is a successful design for web developers. Davis [1] gathered a collection of numerous informal animations to study the primitive operations that people want to express in certain contexts, finding a number of basic operators for expressing complex animations. Vronay and Wang [7] considered the domain of *morphing* in animation, gathering people's descriptions of the

1-4244-2528-0/08/$25.00 ©2008 IEEE

shapes and transitions between a variety of morphing examples. Most recently, Tullio et al. [6] investigated people's descriptions of the behaviors of systems that rely on machine-learning algorithms. Most of these studies inspired novel domain-specific programming languages and authoring environments. We want to apply the same principles to discover what would be natural for interaction designers. Here, we define 'natural' to be that which designers would choose given their actual experience and preference.

3. Method

In this lab study, all participants saw the same screens in the same order. Before beginning the study, participants filled out a questionnaire that asked about basic background information. Next, they answered 56 questions that were presented in a web browser. The pages were implemented in Flash, and there were three parts consisting of five web pages total. Each part was preceded by explanations of how the buttons and question forms work and the format of the questions.

The instructions asked participants to describe all of the interactions, states, and feedback that occurred by typing into textboxes. They were told that they needed to be precise enough that a developer could implement the behavior solely from their description. Participants were told that there was no time limit, and there were no particular rules for what their answers should contain. However, they were not allowed to explain verbally or to draw pictures. The software collected all of the participant's edits (to capture revisions) as well as the final text for each item and timestamps. The textual prompts for each question were as brief as possible, to avoid influencing the participants' word choices.

Our study focused on graphical pointer-based interactive behaviors. These can be described by three aspects: the pointer actions that the user does, the visible responses to those actions on the screen, and the constraints on the causality and timing. We identified various ways that each aspect works in user interfaces, and designed a set of questions to see how designers would express them.

Part 1 focused on detailed interactions with mouse input, and explored how designers described the mouse buttons and movement across different interaction techniques. *Part 2* contained 43 questions across 3 pages and focused on describing the response of graphical objects. These questions covered the basic primitive properties of graphical objects such as size, shape, font, color, etc, (see Figure 1). Part 3 contained six questions and focused on causality and time. The questions consisted of two changing entities that had a certain relation in their behaviors. For example, the second object's color might depend on the first object's color, or the length of a bar might be the same as a number in a text box.

For all questions, we analyzed what specific nouns, verbs, and parameterization the participants used. For Part 1, we also evaluated to what extent they accurately represented all the possible mouse button and movement states. For Part 2, we also evaluated the vocabulary and structure of the answers. For Part 3, we focused on the relationships among objects. Since this was an exploratory study, we did not try to evaluate statistical significance of any of the measures, and just looked for trends. In conducting evaluations, all three authors examined the data together and resolved the few disagreements in interpretation.

In addition to examining designers who are the target audience of our programming language, we were also interested in whether the results would generalize to programmers, who are often part of designers' teams. Therefore, we recruited both designers and programmers to participate in the study. Overall, 16 volunteers participated, including 10 designers (interaction designers, information architects, web designers, graphic designers) and 6 programmers. All of designers had used Flash with 5 of them reporting that they were skillful at Flash, and 3 of them having some experience with implementation (programming) as a part of their job. None of programmers had used Flash, but they had programmed as a part of their job and all mainly used Java and C++. The study took about 1.5 hours, and participants were paid for their time.

4. Results

In analyzing participants' textual descriptions, there were two types of analyses performed: first, there were several specific questions that we wanted to answer, particularly regarding differences between programmers and designers. Second, we explored the descriptions holistically, looking for patterns in the language used to describe the various examples in our study. This section describes results from these analyses.

4.1. Object Orientation

The notion of object constancy is important to designers. We found that when objects change shape or visibility, designers preferred to describe such changes as two objects. In all other changes to size, color, gradient and other properties, they only described a single object. For example, when an object jumped from one position to another, animated to a new position, or disappeared and reappeared in a new location, *all* participants described this as movement of a single object. However, when a second object appeared and afterwards, the first object disappeared, then they used wording showing they were thinking about two objects

("another red box", "a copy of the red box"). 7 designers out of 10 assumed that the second square would automatically adopt the properties of the first, using a phrase something like "a second red square".

In all of today's programming environments and graphical user interface (GUI) toolkits, some things about objects can be changed as properties (e.g., `rect.color = red;`) and some can be changed by calling a method (e.g., `rect.setRGB(0xFF0000)` in Flash). The participants in our study did not make such distinctions. Instead, they often neglected to even name the property or behavior, instead just referring to the new *value.* For instance, 8 out of 10 designers wrote something like the "square changed to blue." The other two designers and 5 of 6 programmers specified the property, as in "the square's *color* changed to blue."

Another interesting pattern was the notion of the *origin* of objects. Based on one question in which an object's size changed, 8 out of 10 participants considered the center to be the default position. When they were shown the size change of the object that gets smaller into the center point, they did not mention the point. However, when the change happened from a different point, then they explicitly mentioned from where the object changed (9 out of 10). This is different from how GUI toolkits work, which change size from a corner by default.

4.2. Naming and Metaphors

With regard to word choice, designers described some concepts with very similar words. All of the designers used "appears/disappears" (for #9 in Figure 1) and "fade in/out." Other concepts had a larger set of words used, such as: "extend", "expand", "increase", "grow", "enlarge", and "become larger" (See also #29 in Figure 1). Programmers, in contrast, used more varied language on all of the questions.

Designers used common names from design software such as Photoshop for property changes. For instance, they use names such as "gradient" (10 out of 10) "mask" (5 out of 10), and "wipe effect," "wipe transition" (3 out of 10). However, none of the programmers used these expressions to describe the same behaviors, and only one programmer used "gradient" as designers did. They used more verbose descriptions, such as "…get filled" or "appears and extends to the right." This difference shows that designer's experience with tools like Photoshop and PowerPoint influences their natural expression of behaviors.

When the participants did not know the name of a behavior, they would use metaphors and examples, as indicated by phrases such as "as if" and "like". For example, 9 out of 10 designers described a square rotating towards the viewer using metaphors: "As if the door opens up into you," "As if spinning," "Like an automobile," and "Like a flat piece of cardboard."

4.3. Modifiers

For the more complicated behaviors in our study, designers used modifiers on the common verbs to describe subtle differences in interactivity and motion. For example, modifiers described how an object moved or appeared, as in "appears by fading out," or "moves to the right." Participants also used modifiers for object changes that happen over time, such as "appears immediately" or "fading out slowly." Some participants used quite general modifiers ("gradually") and other provided precise numbers ("doubles in thickness"). Sometimes the numbers were modified with words like "about" to be less precise ("about 25%").

4.4. Relation between Entities

In Part 3 of our study, participants described relationships between entities. An earlier study of children's expressions showed a preponderance of event-based behaviors for user interfaces. In the present study, however, it was hard to separate whether designers found event-based expression or constraint-based expressions more natural. Many modern programming environments support both. For example, Flash supports event handlers for property changes in an event handler, but also dynamic values to tie the properties of two objects together automatically.

One characteristic of participants' descriptions that *did* differ was in how participants dealt with delay. In one question, 6 out of 10 designers used constraints and events, whereas 4 out of 10 mentioned the time value, as in "…a second after the first one" or "…immediately after." Also, designers with less interaction design experience (i.e., conventional graphic designers) avoided using constraints expression and used event-based expressions if there was a time delay (e.g. "The right box changes colors immediately after the left box"). Such time delays did not affect the programmers' expressions; in the same question, 4 out of 6 programmers used constraint-based expressions.

When designers did use an event-based verbal structure, they referred to things in reverse order such as "…B happens after A" rather than "after A, then B happens…", whereas the latter is the way you would have to express it in all event languages today. For instance, "The box on the right is changing color a fraction of a second after the first one," "The square on the right changes color to match the square on the left, after a slight delay." This is consistent with the results in section 4.1 and of previous work, showing that people prefer to express the main behavior first and then exceptions and modifiers afterwards. Likewise,

while only 1 out of 6 programmers mentioned time values, 6 out of 10 designers used time to emphasize that the relationship of entities repeated.

5. Discussion

The study results suggest new kinds of language features. For example, the object constancy and object property results suggest a new form of object-oriented programming, which blurs the line between data and behavior. Objects should be highly malleable, allowing moving, growing, morphing, and manipulation by expressive primitives. For example, it might be useful to include many of the PowerPoint and Keynote transitions and object animations, but make them polymorphic so that they can be used for any object transformation. This should allow morphing of all properties of an object, including its shape.

Furthermore, the expression of the changes should be allowed either as methods (set-x) or as properties obtaining new values (x=). As in HANDS [4], the target of the operation could be set of objects instead of a single object, for example to move or count a set of objects without requiring the creation of extra data structures. Changes to objects should be allowed to occur immediately or slowly (e.g. fade-out should be similar to becoming invisible). This is similar to Alice, in which properties can change over time [5], but also allowing such changes to be parameterized. For example, a movement could be modified to have a specified path, or a color change could be modified to be a gradient. Given that designers wanted new objects to be similar to existing objects, allowing a modifier to reference existing objects might be natural (e.g., to change color to be the same as another object).

Most participants used metaphors to describe behavior. The idea of using metaphors has been an accepted practice for graphical tools, but not in programming languages. Physical metaphors such as an underlying physics engine might be included (as is available in game engines), to help make objects move similar to real-world situations involving gravity, bouncing, and other behaviors. It would be interesting to investigate language mechanisms for "breaking" these rules of physics (defying gravity, etc.) to achieve some of the subtle effects desired in by participants in our prior study [3].

Although our study did not reveal a strong tendency towards event or constraint based language, our results do suggest that the only perceived difference between the two is whether there is a delay between a change and its effects. This suggests the need for a more flexible language constructs that allows the expression of relationships that occur on a variety of time scales.

There are several limitations to our study. First is the small sample size and the informal analysis techniques. The results also cannot fully cover the designers' language usage, since the study environment was fixed and participants were asked to type into small text boxes. In many cases, the most natural way for the designers to express these behaviors might instead be to draw pictures or create animations like those we presented to them. All of the designers in our study had some exposure to interactive programs like Flash, which may have biased their answers.

Despite these limitations, our study does provide some guidance for designing new programming languages and interactive tools for expressing interactive behaviors. We plan to use these results to guide the design our future tool, and expect that the results reported here will produce a system that is easier to learn and use than previous ones, and hope that these results will be useful to others, and will inspire similar studies to guide their designs.

Acknowledgements

This research was supported in part by a grant from Adobe, Inc., and in part 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] Davis, R.C. and Landay, J.A. "Informal Animation Sketching: Requirements and Design," in *AAAI 2004 Fall Symposium on Making Pen-Based Interaction Intelligent and Natural.* October 21-24, 2004. pp. 42-48.

[2] Green, T.R.G. and Petre, M., "Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework." *Journal of Visual Languages and Computing*, 1996. **7**(2): pp. 131-174.

[3] Myers, B., Park, S.Y., Nakano, Y., Mueller, G., and Ko, A. "How Designers Design and Program Interactive Behaviors," *VL/HCC' 2008.* To appear.

[4] Pane, J.F. and Myers, B.A. "The Impact of Human-Centered Features on the Usability of a Programming System for Children," in *Extended Abstracts for CHI'2002.* Apr 1-6, 2002. Minneapolis, MN: pp. 684-685.

[5] Pausch, R., *et al.*, "Alice: A Rapid Prototyping System for 3D Graphics." *IEEE Computer Graphics and Applications*, 1995. **15**(3): pp. 8-11. May.

[6] Tullio, J., Dey, A.K., Chalecki, J., and Fogarty, J. "How IT works: a field study of non-technical users interacting with an intelligent system," in *CHI'2007*. San Jose, CA: pp. 31-40.

[7] Vronay, D. and Wang, S. "Designing a compelling user interface for morphing," in *CHI'2004.* April 24 - 29, 2004. Vienna, Austria: pp. 143-149.