Designers' Natural Descriptions of Interactive Behaviors

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

Brad Myers, Andrew 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 instantiate this diverse interaction since most implementation tools such as Flash require the use of conventional programming languages and do not support the natural expressions used by designers. Many studies have shown that specifying interactive behaviors is a barrier for designers. To better understand how designers think about interactive behaviors, we conducted a lab study where designers and programmers described using their own language various primitive and composite interactive behaviors. 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

Designers wish to create innovative interactive behaviors. In our previous survey [10], when asked to describe what they wanted to create, designers listed complex interactive behaviors such as "Dynamic layout based on user preference," "An animated 'lens effect' list UI," and "Multi-dimensional selections that impact the display of other controls and data." 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 roll-overs and page transitions in Dreamweaver, and the menu of 19 behaviors in the upcoming Thermo product from Adobe [1]), 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).

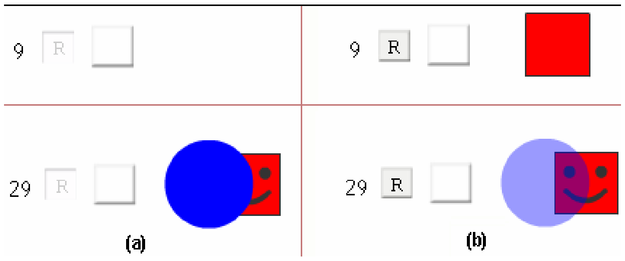


Figure 1: Two examples from our study of before the user clicks the button (a), and after (b). For #9, almost everyone used the same language: "the red box appears", but for #29, the language varied significantly ("fades", "becomes transparent", "opacity goes down", etc.).

Unfortunately, it is challenging for designers to explore the diverse interactive behaviors that they want using either of these approaches. On the one hand, the interactive tools are limited to very conventional behaviors, which our survey [10] suggests will not be flexible enough for the types of behaviors that designers want to implement. On the other hand, designers find it difficult to use today's programming languages to program behaviors.

Is there a way to make the programming easier for designers? When designing programming languages and tools, one has to make certain design choices. In the past, those design choices have often been made for backward compatibility—for example, Flash's ActionScript is based on JavaScript, with a syntax that looks like Java, which looks like C. What if we instead made design decisions based on how people think about their domain of expertise? Is it possible to design programming languages and tools that are closer to the way designers naturally think? The psychology of programming literature [5, 9] and previous studies [11, 13] have shown that this is possible and can make programming easier: for example, HANDS was successfully designed for kids programming [12]and Click! is a successful design for web developers [15]. We want to apply the same principles to discover what would be natural for designers.

In this paper, we describe a study investigating how designers describe interactive behaviors with words. In addition, because our prior study [10] showed that communication with programmers is an important part of the process of designers' work, the new study compares the results of 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.

Our study showed the participants many different primitive and composite interactive behaviors (see Figure 1) and asked them to describe what happens in their own words, as if they were trying to instruct someone else to implement the behavior. The participants were given three groups of questions: *input that causes a reaction* questions, where they had to describe how the mouse and mouse buttons cause a behavior to operate (e.g., various responses to roll-over and mouse button press); *response* questions, where they were shown a set of simple behaviors and had to describe each behavior (as shown in Figure 1); and *causality and time* questions, where they were given animated interactions and had to describe the relationship between the objects. They answered all of these questions with textual descriptions typed into text boxes on the computer (see Figure 2).

The results suggest that there are some noticeable commonalities in designers' descriptions. They often used the same verbs and other vocabulary to refer to simple behaviors and actions, and used similar structures when describing an input and response. Designers consider objects from a user-centric perspective while programmers take a computer-centric view. The study also uncovered different uses of modifiers in the ways participants described time and properties, and uses of physical metaphors to specify complex parameters. We also found that designers consider object constancy to be important—they preferred to describe the changes as one object that morphs, rather than having two different objects where one object disappears and the other appears.

These observations help to explain some of the underlying reasons why current tools and languages are so difficult for designers to use, and suggest that refining the vocabulary and rules of expression in future tools might improve the accessibility and usability of programming languages for designers. In the rest of this paper, we will detail our study process, findings, and implications for improving future programming environments for designers.

2. Related work

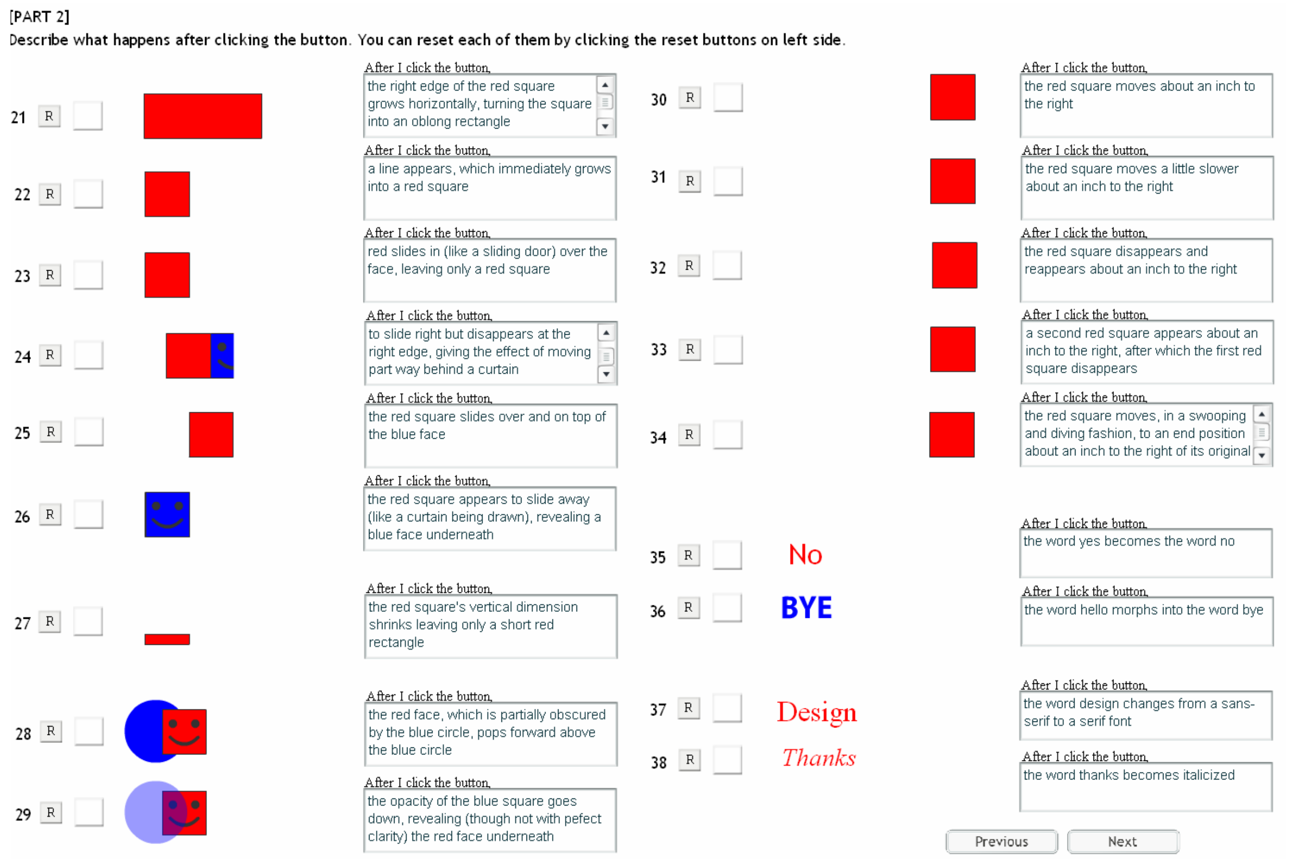
Studying people's use of natural language to inform the design of a programming language is not new. We performed an early study with this goal, focusing on understanding people's descriptions of the rules and behaviors in interactive games, as well as spreadsheet relationships [13]. Since then, other domains have been of interest. Rode [15] studied the domain of server-side web development using a similar methodology, identifying the conflict between the people's stateful expressions and HTTP's stateless nature. Davis [4] 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 [17] considered the domain of *morphing* in animation, gathering people's descriptions of the shapes and transitions between a variety of morphing examples. Most recently, Tullio et al. [16] 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.

There are also a number of studies that explored the way people think when doing visual design and interaction design. Alibali et al. [2] found that designers externalize their mental models in drawings to relieve burdens on working memory and planning. Sketches are also way for designers to explore ideas, but they are often annotated with words to describe behavior [7]. Diagrams, which are laden with verbal descriptions, are a central part of industrial engineering [8]. Our recent investigation into the newer role of "interaction designer" has found these same trends, but also revealing the subtle interactions between the visual and interactive details in designers' process [10].

Even the most novel kinds of interaction design utilize conventions. For example, any interaction that uses a mouse or keyboard is likely to be described using the kinds of events produced by these input devices ("click", "mouse down", etc.). The same is true of conventional types of *controls*, such as buttons and menus. These controls may also have a variety of transitions between states, such as "rollover effects" or many of the kinds of slide transitions that appear in presentation software ("push", "wipe", "fade"). These conventional types of behaviors are bound to influence the verbal descriptions that we solicited in our study.

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


[PART 2]
Describe what happens after clicking the button. You can reset each of them by clicking the reset buttons on left side.

21 After I click the button, the right edge of the red square grows horizontally, turning the square into an oblong rectangle

22 After I click the button, a line appears, which immediately grows into a red square

23 After I click the button, red slides in (like a sliding door) over the face, leaving only a red square

24 After I click the button, to slide right but disappears at the right edge, giving the effect of moving part way behind a curtain

25 After I click the button, the red square slides over and on top of the blue face

26 After I click the button, the red square appears to slide away (like a curtain being drawn), revealing a blue face underneath

27 After I click the button, the red square's vertical dimension shrinks leaving only a short red rectangle

28 After I click the button, the red face, which is partially obscured by the blue circle, pops forward above the blue circle

29 After I click the button, the opacity of the blue square goes down, revealing (though not with pefect clarity) the red face underneath

30 After I click the button, the red square moves about an inch to the right

31 After I click the button, the red square moves a little slower about an inch to the right

32 After I click the button, the red square disappears and reappears about an inch to the right

33 After I click the button, a second red square appears about an inch to the right, after which the first red square disappears

34 After I click the button, the red square moves, in a swooping and diving fashion, to an end position about an inch to the right of its original

35 After I click the button, the word yes becomes the word no

36 After I click the button, the word hello morphs into the word bye

37 After I click the button, the word design changes from a sans-serif to a serif font

38 After I click the button, the word thanks becomes italicized

Figure 2: One screen from the study after being filled in by a participant. For each question, the participant clicks on the button, and then fills in the text field with a description of what happened.

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 (see Figure 2 as an example[*]). Each part was preceded by an instruction explaining how the buttons and question forms work and introducing the format of the questions.

The instructions asked participants to describe all of the interaction, states, and feedback that occurred by typing into the textboxes. They were told that they needed to be clear and precise enough in what they typed that a developer could reproduce the behavior. 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 (so we could see when they went back and revised answers) as well as the final text for each item and timestamps. As shown in Figure 2, the textual prompts for each question were as brief as possible, so as not to influence participants' word choice.

Each participant's answer was analyzed by the following procedure: for Part 1, we mainly evaluated the structure, voice and verbs that the participants used. For Part 2, we mainly evaluated the nouns, verbs, and parameters they used. For Part 3, we focused on the relationship 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.

After finishing the answers for all 56 questions, the participants filled in a final questionnaire that asked their opinions of the study and any final thoughts.

3.1. Specific questions

Part 1 focused on detailed interactions with mouse input. The first four questions had the same response—a number incrementing—but the interaction differed. For #1, the action happened immediately on mouse down. For #2, the button displayed roll-over behavior

[*] You can try out the study yourself at http://www.cs.cmu.edu/~NatProg/CMU-behavior-study.swf

(it became grey when the mouse was inside the button), and the action happened on mouse button *release*. For #3, the behavior was similar to #2, but there was different feedback when the mouse button was pressed down. #4 was similar to #3, but operated as a check box, so a check toggled inside the button. Questions #5 and #7 in Part 1 were examples of linear and 2D constrained dragging, and #6 was a color selection grid, with both roll-over and final feedback. Participants were asked to describe all of the states, feedback, and interaction that occurred for each button.

There were 43 questions across 3 pages in Part 2. This section focused on describing the response of the button, and we looked at what nouns, verbs, and parameters the participants used. The questions consisted of a variety of interactive primitive behaviors mostly using a red square (see Figure 2).

Finally, there were six questions in Part 3. This part 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 (a third example is shown in Figure 3). Participants had to describe the relationship between the two entities.

3.2. Participants

In addition to examining designers who are the target audience of our programming language, we were 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.

16 volunteers participated, 10 designers (interaction designers, information architects, web designers, graphic designers), and 6 programmers. All of designers had been exposed to Flash, 5 of them reported that they were skillful at Flash, and 3 of them had 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

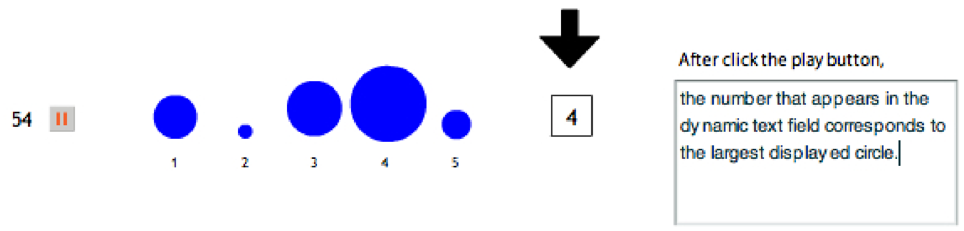
In analyzing participants' verbal descriptions, there were two types of analyses performed: first, there were several specific questions that we wanted to answer, particularly regarding differences between programmers and designers. Second, we explored the descriptions holistically, looking for patterns 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. From their descriptions, we found that designers preferred to describe one object which morphs, rather than using two objects with one fading or blending into the other. For example, we had four object movements—first, the object jumped from one position to another, in the next it slid smoothly, in the third it disappeared from the first place and appeared after a pause in the second place, and in the fourth, the second object appeared first (so two objects were showing), and then the first object disappeared after a pause. *All* participants described the first three as movement of a single object. It was only when we forced them to think about two objects in the fourth condition (because both objects were visible on the screen at the same time), that they described a second object ("another red box", "a copy of the red box"). Designers seemed to assume that the second square would automatically adopt the properties of the first (7 out of 10 said something like "a second red square"

This object constancy even persisted for changes to objects that are not supported in today's environments. For example, when a square slowly changed to be a circle, participants said the box "transforms", "changes" or "becomes" a circle. Similarly when we showed text changing (when "Hello" changes into "BYE"), participants described it as a single object "morphing", "changing" or "becoming" the new value.

In all of today's programming environments and graphical user interface (GUI) toolkits, some things about objects can be changed as properties (e.g.,



the number that appears in the dynamic text field corresponds to the largest displayed circle.

Figure 3: A question from Part 3, in which participants described the relationship between the changes to objects on the left and the changes to objects on the right. A play/pause button allowed participants to start and stop the changes, which occurred at a frequency of typically once a second.

`rect.color = red;`) and others are changed by calling a method (e.g., `rect.setRGB(0xFF0000)` in Flash). We did not find this distinction in participants' answers – they seemed to use the concepts interchangeably. For example, sometimes they wrote "…become italicized", while other times, "the font style changes into…" There was not even consistency within the same person across answers. There was a trend towards more use of methods for changing size, shape and opacity, when there were special words to describe the desired change, such as "increase", "grow", "zoom in/out", "stretch", "expand", "extend", "fade in/out", etc. Participants used properties more when the relevant words were adjectives rather than verbs, such as for colors: "the color of red button changes form red to blue", but often the parameter was unspecified since it is implied by the value: "The red square turns blue".

Another interesting pattern was the notion of the *origin* of objects. In terms of an object's size change, 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 pay attention to or mention the point. However, when the change happened from a different point, then they explicitly mentioned from where the object changed. This is different from how GUI toolkits work, which change size from a corner by default.

We also observed that designers have the contextual concept of defining the "Z" order of objects by using the terms "front" and "back." Programmers also exhibit this tendency, but they use simpler definitions such as "on the top." This is also connected with the concept of layers, which is used as one of the terms in graphic tools such as Photoshop or Illustrator. Three out of 10 designers interviewed defined the order of objects by using the term "layer."

4.2. Naming and Metaphors

There were many observable commonalities in the terms that designers used in their descriptions in Part 2 of our study. Every designer used the same terms for some behaviors, including "appears/disappears" and "fade in/out". When describing other behaviors, the range of terms became broader, such as: "extend", "expand", "increase", "grow", "enlarge", "become larger", etc. Programmers used much more varied language, and it was difficult to find much commonality in the vocabulary used.

Designers used familiar names for property changes. For instance, they use names such as "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. They used more verbose descriptions, such as "…get filled" or "appears and extends to the right." This difference indicates that since designers are familiar with graphic tools such as Photoshop, PowerPoint, or Keynote, they use terms used in those tools when they see similar concepts in behaviors, whereas programmers are more likely to use terms from GUI toolkits.

However, when all participants did not know the name of a behavior, they tended to use metaphors and concrete examples rather than descriptions of the details of the behavior. The metaphors were usually signaled by a clear syntax, such as starting with "as if…" or "like…" For example, "As if the door opens up into you," "As if spinning," "As if falling backward," "Like an automobile," and "Like a flat piece of cardboard.".

4.3. Syntax

In addition to the structure of the participant's answers, we were also interested in the particular syntax they used (what word order and special characters). For example, when setting the value of parameters, almost all participants (9 out of 10 designers, and 6 out of 6 programmers) used "…of" as a syntax rather than using a possessive "…'s." For example, they used "the color of a red box" rather than "a red box's color."

The participants occasionally used quotation marks or parentheses in their descriptions. They sometimes (but not always) use single or double quotation marks when indicating some specific text (as in "sans serif "design" changes to serif") or when they want to emphasize specific behavior among the multiple behaviors occurring simultaneously. They also sometimes seemed to use quotation marks to signify metaphors. For instance, they described that "the square "opens"" or ""tips over"." They also used parentheses to add detailed information such as numerical values or related concepts. For instance, "move (no transition, no sliding)" or "become transparent (50%)." While designers use syntax in a casual way, programmers had more syntactic consistency. Most programmers used double quotation marks to indicate a specific text string. This shows that programmers' knowledge of coding conventions affected their descriptions.

4.4. Modifiers

As we described in the previous section, designers had a considerable commonality in terms of using terms to describe simple behaviors. In addition, we found that when the behaviors get complicated, they tended to use modifiers on those common verbs to describe subtle differences in interactivity and motion. Modifiers described how an object moved or appeared.

For example, “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.”

It was interesting that participants sometimes used general modifiers (“gradually”) and other times provided precise numbers (“doubles in thickness”). Sometimes the numbers were modified to be less precise (“about 25%”).

4.5. Relation between entities

In Part 3 of our study, participants had to describe the relationship between entities. In our earlier study of children’s expressions, we saw a preponderance of event-based behaviors [13]. However, in the present study, we found that it was hard to separate whether designers found event-based expression or constraint-based expressions more natural. Many modern programming environments support both kinds of specifications. For example, in Flash, you can use event handlers where you program a property change in an event handler. Alternatively, you can use dynamic values to tie the properties of two objects together, so the system maintains this constraint for you.

In our study, designers seemed to use a mixture of both kinds of expressions. However, if there was some time delay between the entities, designers tended to mention the time value, as in “…a second after the first one” or “…immediately after” (4 out of 10). Also, some designers with less interaction design experience (i.e., conventional graphic designers) tended to avoid 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”). However, this time delay did not affect the programmers’ expressions. Most of the programmers used constraint-based expressions and described the relationship very simply for all of the questions in Part 3.

We looked at the structure when designers used an event-based expression. They referred to things in reverse order such as “…that 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.” Note that this is consistent with our results of section 4.1 and of previous work [11] that showed that people prefer to express the main object first and then exceptions and modifiers afterwards. Likewise, while programmers hardly mentioned time values, many designers used time to emphasize that the relationship of entities is repeating.

There are some differences in the range of verbs used by designers for the properties that depended on the other object, although most used “change” of a property. However, they used a variety of terms to refer to the number that controls the property that changes: “correspond”, “correlate”, “reflect”, “change”, “display”, “refer”, “indicate”, “equal”, “show”, and “represent.”

Two of the questions required participants to refer to a set of objects. Consistent with our prior results [11], participants operated on the entire set without adding control structures (for example, “position of the largest circle”, or “the number of red triangles”).

4.6. State transitions

In Part 1, we performed within subject comparisons on a set of questions that ask respondents to describe the states, feedback, and interactions of different levels of complexity for responses to the mouse. In these comparisons, we found that the more complex the behavior, the less accurately designers described the states. This difference was clearer in designers with less implementation ability. Table 1 indicates that all designers accurately described every state of first two simple behaviors. However, from the interactions that had four or more states, the number of designers who gave accurate descriptions was drastically reduced.

Number of states (complexity of button behaviors)	**1**	**2**	**4**	**8**
Number of correct answers from designers (out of 10)	10	10	4	0

Table 1. The accuracy in describing the states in button behaviors

Programmers had an easier time describing complex interactions. While designers tried to explain the response and feedback by describing the appearance, programmers tended to mention the function. For example, for #4 that included a checkmark and both roll-over and separate mouse press behaviors, many programmers described the behavior by referring to “a check-box” to imply its function. They also used the term “toggle” to refer to the interaction. However, designers often missed several states by focusing on the appearance of feedback, not necessarily its complete function. This is consistent with what we have seen on commercial web sites, where often the implementations will do the wrong thing if, for example, the user presses down inside a custom-built button, moves outside and then releases.

We also looked at the vocabulary used to refer to the states of the interactions. Whereas programmers mostly use the term “mouse over,” the designers used various words for this, such as “cursor over,” “mouse over,” “roll over,” “hover over,” and “mouse in.”

Another significant finding in Part 1 is the difference between designers and programmers in terms of the order in which interactions are described. Designers describe the order in which a user uses a button. For instance, they started with “When a user moves a mouse over the button…” This suggests that designers have a user-centric perspective and seemed to focus on what the user does. However, programmers described these same behaviors by first defining the function and then talking about the actions done with the mouse. For example, “Implement the button which on clicking increments count starting from num 1, … when the button is pressed down…” This suggests that programmers have a computer-object-centric point of view, the opposite of the designers.

Another conceptual difference between designers and programmers is their approach to the problems. Almost all of the programmers (five out of six) mentioned the initial value, saying, for example, “increments count starting from no 1,” or “The number begins in 1.” However, none of designers mentioned the starting value explicitly. Few of the designers (3 out of 10) referred to numbers such as “1 is added to the number” but the majority described it in general terms, such as “the numbers appear and count.”

This part suggests that designers and programmers have different approaches due to their different backgrounds. Designers are more focused on describing the input-feedback relation and its look in a general way, rather than describing the behavior in a specific order like in a programming language.

5. Discussion and Implications for Future

Many of 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 lots of expressive primitives. For example, it might be useful to include many of the PowerPoint and Keynote transitions and object animations, but making them polymorphic so that they can be used for object transformations. This should apply to allow morphing of all properties of an object, including its shape.

Furthermore, the expression of the changes should be allowed either as methods or as properties obtaining new values (e.g. the “visible” property might take a number instead of a Boolean to support fading out, the X and Y positions might be set to a path instead of to an integer, etc.). As in HANDS [12], 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 the functioning of Alice, in which all properties can change over time [14], but unlike Alice, allowing such changes to be fine-tuned through parameters. 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 like another object).

Most participants (designers a bit more than programmers) sometimes described the modifiers for the changes as a metaphor. The idea of using metaphors has been an accepted practice for graphic tools [3][6], and current tools such as PowerPoint and Keynote have many functions, such as transition behaviors, which use metaphors. However, these kinds of metaphors have not been used as part of a programming language. Possibly, physical metaphors such as an underlying parameterized physics engine might be included (as is available in game engines). It might provide movements that conform to various real-world situations, such as gravity, bouncing, etc. Just as interesting would be 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 [10].

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

6. Threats to Validity

The main issue with the generalizability of the results is the small sample size and the informal analysis techniques. We feel comfortable using our recommendations and observations to guide further studies and language designs, but there is certainly no guarantee that an environment that takes advantage of them will be better. Clearly, any system created based on these findings will need to be evaluated.

Since this was a limited lab study, the results cannot fully cover the designers' pure and natural language, 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 would 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. However, this is the same as the target audience for our future tools. Another concern is that all participants were students at Carnegie Mellon University, so they might have similar experience and instruction. However, they have quite varied backgrounds before coming to Carnegie Mellon, which hopefully mitigates this concern.

7. Conclusions

Based on the results of this and other similar studies [11, 13], it seems clear that this kind of investigation can reveal interesting insights into how a target audience thinks about programming concepts. When a new language or environment is being designed, having such knowledge can only be helpful as one of the considerations, so we recommend this methodology as a precursor to future domain-specific designs.

The current study of what is natural for designers when expressing interactive behaviors provides more scientific insight for choosing among design alternatives in tools for designers. We plan to use these results to guide the design of a new programming language and system for designers and expect that the results reported here will produce a language that is easier to learn and use than other languages. In addition, the result reported here can be used by developers to assist in the design of other kinds of tools.

Acknowledgements

Thanks to Adobe for financial support and advice for this research.

References

[1] Adobe Systems Incorporated, "Product Codename: 'Thermo'," 2008. http://labs.adobe.com/wiki/index.php/Thermo.

[2] Alibali, W., Bassok, M., Solomon, K.O., Syc, S.E., and Goldin-Meadow, S., "Illuminating mental representation through speech and gesture." *Psychological Science*, 1999. **10**: pp. 327–333.

[3] Carroll, J.M., Mack, R.L., and Kellogg, W.A. "Interface Metaphors and User Interface Design," in *Handbook of Human-Computer Interaction.* 1988. Elsevier Science Publishers B.V. (North Holland). pp. 67-85.

[4] 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.

[5] 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.

[6] Halasz, F. and Moran, T.P. "Analogy Considered Harmful," in *Conference on Human Factors in Computing Systems.* 1982. Gaithersburg, MD: pp. 383-386.

[7] Heiser, J., Tversky, B., and Silverman, M., "Visual and spatial reasoning in design," in *Sketches for and from collaboration*, 2004, pp. 69 – 78.

[8] Henderson, K., *On Line and On Paper: Visual Representations, Visual Culture, and Computer Graphics in Design Engineering.* 1998, The MIT Press.

[9] Hoc, J.-M., Green, T.R.G., Samurçay, R., and Gilmore, D.J., eds. *Psychology of Programming.* 1990, Academic Press: London.

[10] Myers, B., Park, S.Y., Nakano, Y., Mueller, G., and Ko, A. "How Designers Design and Program Interactive Behaviors," in *Submitted for Publication.* 2008.

[11] Pane, J.F. and Myers, B.A. "Tabular and Textual Methods for Selecting Objects from a Group," in *Proceedings of VL 2000: IEEE International Symposium on Visual Languages.* September 10-13, 2000. Seattle, WA: IEEE Computer Society. pp. 157-164.

[12] Pane, J.F. and Myers, B.A. "The Impact of Human-Centered Features on the Usability of a Programming System for Children," in *CHI.* Apr 1-6, 2002. Minneapolis, MN: pp. 684-685. http://www-2.cs.cmu.edu/~pane/research.html. Extended Abstracts for CHI'2002.

[13] Pane, J.F., Ratanamahatana, C.A., and Myers, B.A., "Studying the Language and Structure in Non-Programmers' Solutions to Programming Problems." *International Journal of Human-Computer Studies*, February, 2001. **54**(2): pp. 237-264. http://www.cs.cmu.edu/~pane/IJHCS.html.

[14] Pausch, R., Burnette, T., Capehart, A.C., Conway, M., Cosgrove, D., DeLine, R., Durbin, J., Gossweiler, R., Koga, S., and White, J., "Alice: A Rapid Prototyping System for 3D Graphics." *IEEE Computer Graphics and Applications*, 1995. **15**(3): pp. 8-11. May.

[15] Rode, J. and Rosson, M.B. "Programming at Runtime: Requirements and paradigms for nonprogrammer web application development," in *IEEE Symposium on Human-Centric Computing Languages and Environments.* 2003.

[16] 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 '07: Proceedings of the SIGCHI conference on Human factors in computing systems.* 2007. San Jose, CA: pp. 31-40.

[17] Vronay, D. and Wang, S. "Designing a compelling user interface for morphing," in *CHI'2004: SIGCHI Conference on Human Factors in Computing Systems.* April 24 - 29, 2004. Vienna, Austria: pp. 143-149.