

Ian Bellomy
January 5, 2009
ITGM 705
Andrew Hieronymi

"Shapes" Stage 2 : Paper Prototype

Overall I found my paper prototype experience rather enlightening. The interaction and communication with the user really helped crystalize how they see what's in front of them and what I need to focus on. In addition to learning things about my design I now understand how running a paper prototype is a skill in of itself. I look forward to using this process in the future.

With my design there were problems of course. Beyond the user's confusion, the nature of my project created several hurdles. Most user actions in my application cause piecemeal updates to a number of parts of the interface. Besides simply taking extra time to respond to a user (which isn't really a problem), I would sometimes forget to update things and have to interrupt the user to finish adjustments. Separate from this, the small flurry of activity would sometimes confuse the tester as to what had happened. Contrary to ?'s experience one tester reported having missed some changes to the screen amidst my activity. They claimed that "...had it been a real screen I probably would have seen it." In that instance however, I think they may have been actively avoiding that part of the interface (I'll talk about this later).

There were the good problems too, numerous ones. From even a conceptual point of view there were weaknesses. The idea of combining programming and vector graphics ends up being like combining programming and fishing. As a completely symbolic representation, programming has no general relationship to vector art, and proved highly difficult to insert in a meaningful way. After testing I have doubts about some of my original assumptions. (Programming would be more accessible if it were introduced in the context of something familiar for instance).

One positive I found was that the shape based/ vector image making was instantly accessible. I'm fairly certain thought that the nature of the paper prototype, with the shapes being literal shapes cut out from paper, made this exceptionally easier to understand than it would be on a screen. To check this I'll really need to do a version that uses markers and a whiteboard for the page. On the other hand, it might be best to go with it and explicitly use a paper metaphor in a screen version.

One particularly interesting behavior I noticed was that users ignore interface elements that they don't understand. On top of that, the amount of stress they feel using the system seems to relate to how much of it they're trying to ignore. In one case with an early version, a user was instructed to change the color of a shape. While hunting for a way of accomplishing this, they explicitly asked to close the very panel containing the button to perform this task. The panel was an editable list of the shape's properties, including color. "This box, can I just close it? I just don't want to look at it."

Way more interesting than the problems was the different mental models users came up with. My initial (and vague) goal was to introduce a user to programming. My approach was to show a user the abstractions (commands, shape information) along side the visuals. If only for the behavior I just mentioned this proved to be highly inadequate. Through iterations I continued to remove as much from the interface as possible. Though this still didn't help with getting users to manipulate tools abstractly. When it came to tasks that involved creating tools or most testers needed specific instructions. While most had limited problems with the shape list and using it to modify the visuals, the idea that the tools were entities to be created and modified seemed exceptionally obtuse. However, their inexperience with the concepts gave them some interesting points of view.

The command box that listed the code versions of the actions the user performed was one the few elements to remain as the interface got pared down. Even the fine artist was comfortable with it, asking what was going on with it they replied "Oh, I just figured it was showing me what I've done so I could go back and change things." Not only were they comfortable with it, but they went on to describe how it worked in their mind in, with it being notably non-linearly editable. They figured they they'd be able to reconstruct their image by rearranging their actions. While I'd thought about this before, I never would have guessed the idea would be that intuitive to someone.

Another user was prompted to make a tool that would perform all the actions they had taken so far. The designed method was to click "new tool" , and drag the contents of the history into the code box. Their approach was to try and save the file as a tool via the file menu. It seems so obvious, I don't know why I never would have thought of it.

In addition to coming up with novel mental models, I found it interesting how users would combine pieces of their knowledge to perform tasks. They seemed to bounce back and forth between trying new things and trying to create processes out of things they knew. If they could construct an approach out of steps they knew, they would. If it didn't work, they would try a button or two they might not have looked at. If that didn't work they'd try another combination of familiar things they knew. And so on. In one while looking to edit a shape's color (while trying very hard to block out the scary list of shape properties) they went to the edit menu. Seeing that the option wasn't listed, they clicked on the shape and then edit menu. Which brings me to another general observation.

The importance of menus for beginners can not be stressed enough. *Every* user hunted through them when confronted with a task they were unfamiliar with, often going directly to where the content *should* have been had I known better. Originally I thought showing the same list of commands in two places in the menu *and* having them directly accessible on the screen was a little redundant. Seems that redundancy is very much a friend.

The interaction with the users was more than just informative. Typically with interactive design, specifically web design, a designers slaves away behind a computer only to send the work into the ether. If they're lucky, there may be recognition in terms of awards, but I'm not sure it's nearly as fulfilling as watching a user interact with the work. Getting to this point early not only helps the project's quality, but makes it more enjoyable to work on.