Low-Fidelity Prototyping of Gesture-based Applications

Ali Hosseini-Khayat **Teddy Seyed** **Chris Burns** **Frank Maurer**

Computer Science Department
University of Calgary
Calgary, AB Canada
{hosseisa, aseyed, ccburns, fmauer}@ucalgary.ca

ABSTRACT
Touch-based devices are becoming increasingly common in the consumer electronics space. Support for prototyping touch-based interfaces is currently limited. In this paper, we present a tool we developed in order to bridge the gap between user interface prototyping and touch-based interfaces.

Keywords
User interface, Prototyping, Touch, Tabletops

INTRODUCTION
Prototyping serves an important role in detecting and addressing usability issues in a user interface. Usability testing of prototypes detects usability issues that can be fixed. This leads to both improved and more intuitive interfaces for the user.

Low fidelity prototyping is a fast and cost effective approach that fits well with short iterations typically found in agile software development processes. Low-fidelity prototypes are typically sketches with pen and paper but also include digital sketches. Tools such as Microsoft's SketchFlow, allow for teams and developers to rapidly create a digital low-fidelity prototype for a design and test it with a user and gather feedback. These tools address prototyping of applications that utilize the traditional input paradigm of the keyboard and mouse.

Recently however, multi-touch and surface computing environments have become increasingly common in several areas such as education, retail and medicine. The affordances these technologies provide is a critical reason for their success, as collaboration and data visualization can be better facilitated on a large surface or multi-touch display. Additionally, the gestures that multi-touch can provide opportunities for developers and designers to create intuitive and innovative interfaces.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
EICS'11, June 13–16, 2011, Pisa, Italy.
Copyright 2011 ACM 978-1-4503-0670-6/11/06...$10.00.

Undoubtedly, these touch and gesture based technologies mark a shift in the input paradigm and consequently, creating tools that support the design of applications utilizing them is vital. Currently, toolsets for digital low fidelity prototyping are limited to the keyboard and mouse paradigm, with limited or no support for multi-touch or gesture-based applications.

In this paper, we present our work on a low-fidelity prototyping tool that addresses the aforementioned limitations and is particularly targeted towards gesture-based interactions.

RELATED WORK
Khandkar has done work in the area of gesture definitions and processing for touch-based interactions. His framework, Gesture Toolkit, provides a complete gesture recognition system along with a domain specific language (DSL) for defining gestures [**1**] [**2**] [**3**]. This system has several touch input providers supporting a large number of platforms including the SMART Table, Windows Touch devices and even the Nintendo Wii. Gesture Toolkit provides a strong basis for cross-platform touch detection and gesture recognition [**4**].

Derboven et al. performed a study comparing the use of low-fidelity versus high-fidelity prototyping for designing multi-user, multi-touch interfaces [**5**]. They report that low-fi prototypes can be a valuable tool for designing user interactions for multi-touch tabletops. Rick et al. performed a study of low-fi prototyping of tabletop applications for children and concluded that low-fi prototyping is a useful approach, although they recommend adapting desktop applications to digital surfaces [**6**]. We believe this approach results in less effective user interfaces for tabletops.

DTFlash [**7**], is an early attempt to develop a rapid prototyping tool for multitouch devices. Built for use with Multimedia Flash applications, the touch system is delivered using the DiamondTouch SDK. The system allows developers to rapidly create prototype applications using a graphics-oriented Flash authoring tool. This results in hi-fi prototype applications rather than low-fi prototypes. Example applications created with the tool can be interacted with online via Flash-enabled browsers. A drawback to this system is its reliance on the DiamondTouch SDK.

Low-Fidelity Testing Tool Support

Tool support for interaction designers is somewhat limited. Some tools, such as Microsoft Expression Blend – SketchFlow [**8**], are explicitly developed for that purpose. Other tools, such as Microsoft Visio [**9**] and PowerPoint [**10**], are simply adapted for interaction design work and have a much more general function.

Most prototyping is done using pens and paper, which has the advantage of being low cost and requires no training to use. A prototype can be generated quickly and changed easily, simply erase part of the sketch and redraw it as desired. Since paper prototypes have no emotional attachment, as they are cheap and easy to create, they can be modified extensively even during a test session. The physical nature of paper sketches becomes a disadvantage, however, since they cannot be shared easily with users in distributed teams. Users also sometimes have difficulty viewing the paper sketches as the future application they are supposed to represent.

SketchFlow, a tool developed by Microsoft, allows users to create low-fi and medium-fi prototypes. It provides a suite of UI elements that have the appearance of being hand drawn. Designers can use button-clicks, dragging and dropping and other interactions to transition between states in a prototype, creating a flow that simulates a real application. As the prototype is developed the fidelity level can be increased and eventually even moved into production development. This particular emphasis on reuse of components is unique to SketchFlow. Using the SketchFlow player users can execute the prototype at a remote location.

WOzPro [**11**] captures freehand drawings rather than using defined widgets. It also supports templating, pages can be based off a single master page and updates to the master page will propagate out to all the template pages. It does not, however, provide any interaction support but rather displays the prototypes through a simple slideshow. Since it does not provide any interaction, its utility as a usability evaluation tool is limited.

Another set of design tools use widgets, dragged and dropped into a canvas, as a mechanism for prototype creation. SILK [**12**] [**13**], DENIM [**14**], Serena Prototype Composer [**15**], Visio [**9**] and Balsamiq Mockups [**16**] are all among this set. While these products support interaction by test users, they limit the designer to an existing set of widgets. Serena Prototype Composer provides a more hi-fi prototyping feature while there rest use widgets explicitly designed to appear hand drawn or rough.

Another tool, often used for prototyping interfaces, is Microsoft PowerPoint. Prototypes can be easily created as a package of slides and widely distributed because of PowerPoint's wide install base. However, designing interactive prototypes using PowerPoint can be a time consuming process given the intended use of the application.

Tools for prototyping mobile device interfaces have emerged recently, such as Briefs [**17**] which runs live on the iPhone and iPhone Mockup [**18**] which runs through a browser. These can execute functionality and accept interaction. Briefs allows users to defined prototypes using images and then host them for others to view via a "briefcast". iPhone Mockup relies on a web hosted solution rather than a native application. Mockups are created using a web application from a series of defined UI widgets and then distributed and hosted via the application. Both applications, however, are specifically targeted at developing iPhone applications and do not facilitate designing a more general prototype.

To the best of our knowledge, tool support for low-fidelity of prototyping touch-based user interfaces is non-existent at the time of writing.

ACTIVESTORY TOUCH

Touch-based devices and interactions are gaining increasing momentum in the consumer space, with touchscreen smartphones, tablets, and surfaces. However, prototyping and tool support for prototyping touch-based interfaces have not evolved at the same pace. In fact, existing tools lack support for touch-based interactions entirely. Given the importance of usability, particularly in the context of touch-based applications, we attempted to address this issue.

Motivation

Previously, ActiveStory Enhanced was developed to provide tool support for low-fidelity prototyping and usability testing [**19**]. The tool was built with standard desktop applications with mouse-based interactions in mind. Designers sketch pages of a prototype such that each page represents a state of the user interface. Once the various pages of the prototype are sketched, the designer adds interaction to the pages by clicking and dragging over areas of the page that should be hotspots. Upon creation of a hotspot, the designer specifies which page of the prototype the hotspot should navigate to when clicked by test users. After the prototype has been designed, it can be deployed to a web server where test users can interact with and attempt to complete tasks specified by the designer. Usability data, such as mouse movements, time spent on pages and comments, are automatically collected by the web testing tool as users interact with the prototype. Designers can then analyze the collected usability data through the web-based reports component.

The question of whether ActiveStory Enhanced would suffice for touch-based user interfaces gave rise to a pilot study that was conducted with two participants, with the main goal of evaluating the applicability of the tool to prototyping touch-based applications. One participant was a user experience (UX) professional and the other was a graduate student. The graduate student used the tool as part of an actual project and designed a prototype for the user interface of a tabletop media manager and player. An interview was conducted and feedback was elicited. The UX professional was asked to create a UI for a similar application and think aloud while doing so. This participant was asked to use the tool and design the prototype on a tabletop (a SMART table), in order to understand whether it makes sense to design a prototype for a tabletop application on a tabletop. This was a side-goal in addition to the main goal of the study. The researchers observed the participants' interaction with the tool and his attempt at creating a prototype for the application. This was followed-up by an interview regarding his experience prototyping a surface application with the tool.

Both participants indicated that they found it difficult to represent interactions such as dragging and dropping, scaling (pinch-to-zoom) and rotations. This was because hotspots always acted like buttons in the testing environment, i.e. once they were touched they would immediately navigate to the target page, regardless of the gesture used. The UX professional pointed out that prototyping on the tabletop with ActiveStory Enhanced was difficult at best. Some of the difficulties arose from the tablet-oriented nature of the tool, whereas other difficulties were caused by the hardware and the tabletop form factor. Difficulties with the tool included lack of support for specifying gestures other than a tap, the inability to run prototypes within the design environment (i.e. preview the prototype at design time) and the need to reach across the table to access the menus and palettes. Issues with the tabletop included lack of precision, washed-out colors and unresponsive touch input.

In an effort to bridge this gap, ActiveStory Enhanced [**19**] was extended to provide support for adding touch-based interaction and allowing test users to interact with the prototype on a touch device. The new tool, ActiveStory Touch, addresses some of the issues with prototyping touch-based interfaces that were discovered in ActiveStory Enhanced, mainly the lack of support for defining different gestures for prototype elements[1].

Usage Scenario

ActiveStory Touch allows UI designers to sketch prototypes using the designer portion of the tool and subsequently distributes them to test users for evaluation over the web. Prototypes can be sketched on a tablet, tabletop or desktop computer. The designer tool is shown in Figure 1 and the process workflow for ActiveStory Touch is presented in (Figure 5).

Once the various pages of the low-fidelity prototype have been sketched, the designer can specify which page to navigate to for a given gesture and a given part of a prototype page (Figure 2, Figure 3). For example, a designer can specify that page 2 should be navigated to when a UI element is tapped and page 3 should be navigated to when a user presses-and-holds.

Once the designer has completed sketching the prototype, it can be published in the form of a Silverlight web application, allowing test users to access it remotely (Figure 4). Test users can use a touch-based device to interact with the prototype as if it were an actual touch-based user interface for an application. In other words, a user with a touch device capable of running Silverlight can navigate to a web server where a prototype is hosted, view the prototype and perform gestures on it in the same way they would for a touch-enabled application on a touch-based device. Various usability data such as mouse movements, clicks and time spent on pages is automatically collected, however the usability data is not the focus of this paper. This rapid deployment and feedback loop allows designers to iterate through various designs quickly and determine how they would be used and whether they are usable. The tests with potential end-users allow identifying any usability flaws that may exist in the design.

[1] Available at activestoryenhanced.codeplex.com

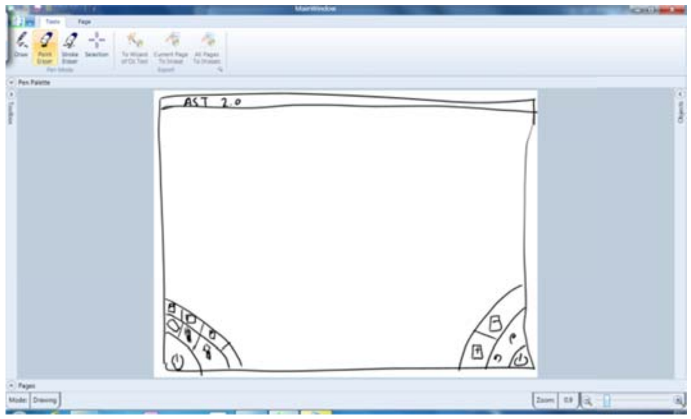


Figure 1. Designing a prototype using the ActiveStory Touch Designer Tool

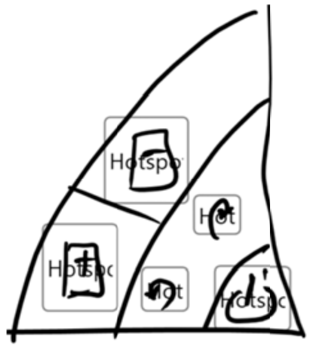
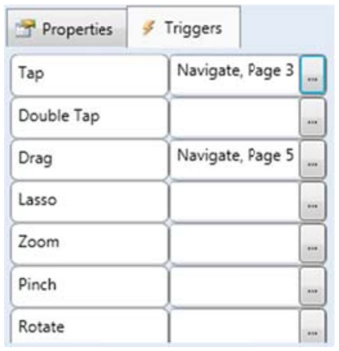


Figure 2. Adding hotspots to the sketch.



Properties | Triggers

Tap	Navigate, Page 3
Double Tap	
Drag	Navigate, Page 5
Lasso	
Zoom	
Pinch	
Rotate	

Figure 3. Specifying a gesture and behavior for hotspots.

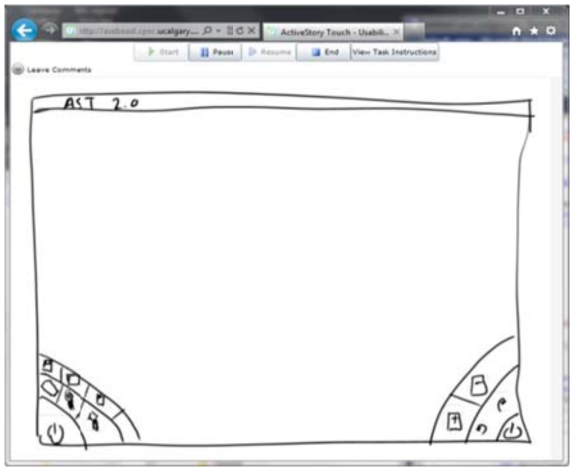


Figure 4. ActiveStory Touch Usability Testing tool, allows multi-touch input and gesture recognition.

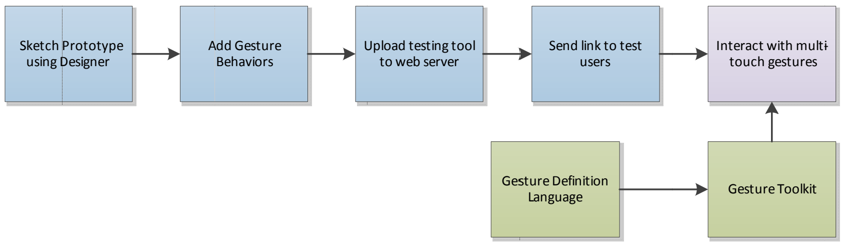

Sketch Prototype using Designer
Add Gesture Behaviors
Upload testing tool to web server
Send link to test users
Interact with multi-touch gestures
Gesture Definition Language
Gesture Toolkit

Figure 5. ActiveStory Touch prototype design and evaluation workflow.

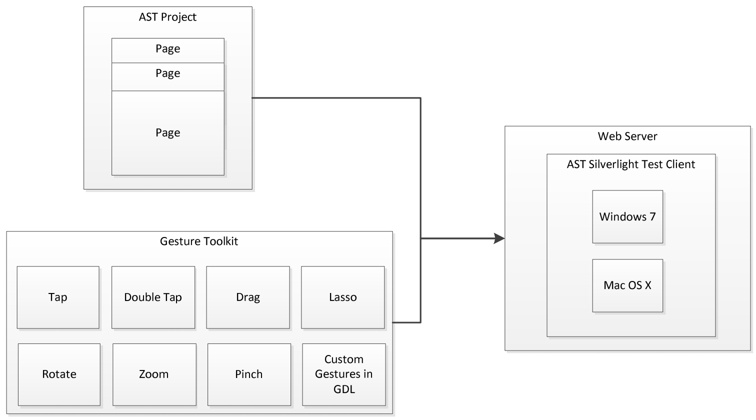


Figure 6. ActiveStory Touch Architecture

Design

ActiveStory Touch prototypes consist of multiple pages. Pages consist of ink strokes, imported images and prototype elements. Each page corresponds to a page of a prototype as it would have been sketched on paper, i.e. a given state of the user interface.

Prototype elements are currently limited to hotspots, i.e. a given area that is specified by the designer and reacts to clicks or touch interaction. Elements may be extended in the future to include support for mixed-fidelity prototyping with text boxes, combo boxes and other common widgets. Each element is associated with a set of gestures and target behaviors in the form of a dictionary. In other words, each element can react to one or more gestures and perform a different behavior for each, for example navigating to different pages based on the type of gesture, where Navigation is the behavior. While the possible behaviors are currently limited to navigating to different pages in the prototype, future work includes adding support for animations and other behaviors on the same page of the prototype.

ActiveStory Touch prototype projects are serialized in XML to allow easy reuse of the data. The same project file is then used as input for the Silverlight-based testing tool, allowing it to render a given page of the prototype. Once the page is rendered, the prototype elements are created and their corresponding gesture events are registered with Gesture Toolkit. Behaviors are implemented by listening to the gesture events and performing a common method, BehaviorInvoked. BehaviorInvoked is supplied with any parameters that are specific to the behavior, for example the target page for a Navigate behavior. This allows for a generic prototype player that can render any prototype created in the designer by simply changing the input XML project file.

System Implementation

ActiveStory Touch was developed in C# using the Microsoft Windows Presentation Foundation (WPF) for the desktop designer portion of the tool and Microsoft Silverlight for the web-based testing portion. Silverlight provides support for touch-based interactions through a web-browser across touch-enabled versions of Windows and Mac OS. In addition, Silverlight applications can be run out-of-browser, as if they are desktop applications. As a result, the usability testing component of ActiveStory Touch can be executed as if they were native multi-touch applications. Running a prototype in full screen mode, out-of-browser allows the designer to perform testing on a tabletop or wall-mounted surface. Gesture Toolkit [**4**] was used for gesture detection and handling in the testing tool. Using Gesture Toolkit, ActiveStory Touch provides support for detecting common gestures such as dragging, swiping, pinching, tapping, double-tapping and lassoing as well as being easily extensible through Gesture Toolkit to provide support for custom-defined gestures. For example, a designer could define a counter-clockwise circle gesture in Gesture Toolkit and subsequently specify, within ActiveStory Touch, that when a user performs that gesture the prototype player should show a state in which the “undo” action was performed. ActiveStory Touch’s use of Gesture Toolkit also provides the advantage of being cross-compatible between WPF and Silverlight, allowing for future extension of ActiveStory Touch to allow for a more flexible environment for tablet and tabletop design, perhaps providing a tabletop-based user interface for designers. An overview of the architecture is presented in Figure 6.

FUTURE WORK

Further work on ActiveStory Touch involves adding support for the collection of real-time usability data, further enhancing the development process of an application with automated feedback from non-collocated users.

ActiveStory Touch will also be further extended to provide better support for collaborative prototyping on a digital tabletop, including a non-directional touch-conducive UI.

The support of cross- platforms and multi-surface interactions and prototypes is also an area we plan to extend ActiveStory Touch, particularly to support development in areas where multi-surface environments are necessary.

Finally, studies will be conducted to evaluate the effectiveness and usefulness of the tool in supporting prototyping of touch-based interfaces.

REFERENCES

1 Khandkar, Shahedul Huq. *A Domain-Specific Language for Multi-Touch Gestures*. University of Calgary, Department of Computer Science, Calgary, December 2010.

2 Khandkar, Shahedul Huq and Maurer, Frank. A Domain Specific Language to Define Gestures for Multi-Touch Applications. (Reno/Tahoe, Nevada 2010), The 10th SPLASH Workshop on Domain-Specific Modeling.

3 Khandkar, Shahedul Huq and Maurer, Frank. A Language to Define Multi-Touch Interactions. (Saarbrucken, Germany 2010), The ACM International Conference on Interactive Tabletops and Surfaces.

4 Khandkar, Shahedul Huq, Sohan, SM, Sillito, Jonathon, and Maurer, Frank. Tool Support for Testing Complex Multi-Touch Gestures. (Saarbrücken, Germany 2010), The ACM International Conference on Interactive Tabletops and Surfaces.

5 Derboven, Jan, Roeck, Dries De, Verstraete, Mathijs, Geerts, David, Schneider-Barnes, Jan, and Luyten, Kris. Comparing user interaction with low and high fidelity prototypes of tabletop surfaces. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries (NordiCHI '10)* (New York, NY 2010).

6 Rick, Jochen, Francois, Phyllis, Fields, Bob, Fleck, Rowanne, Yuill, Nicola, and Carr, Amanda. Lo-fi prototyping to design interactive-tabletop applications for children. In *Proceedings of the 9th International Conference on Interaction Design and Children (IDC '10)* (New York, NY 2010), ACM.

7 Esenther, A. and Ryall, K. Fluid DTMouse: Better Mouse Support for Touch-Based Interactions. *Advanced Visual Interfaces (AVI)* (May 2006).

8 *Microsoft Expression Blend - SketchFlow - Product Overview*. http://www.microsoft.com/expression/products/Sketchflow_Overview.aspx.

9 *Microsoft Visio - Product Overview*. http://office.microsoft.com/en-us/visio/default.aspx.

10 *Microsoft PowerPoint - Product Overview*. http://office.microsoft.com/en-us/powerpoint/default.aspx.

11 *WoZ Pro Project Site*. http://www.eecs.wsu.edu/~veupl/soft/woz/.

12 Landay, James A. and Myers, Brad A. Sketching Storyboards to Illustrate Interface Behaviors. In *CHI '96 Conference Companion: Human Factors in Computing Systems, ACM* (New York, NY 1996), 193-194.

13 Landay, J.A. and Myers, B.A. Interactive Sketching for the Early Stages of User Interface Design. In *CHI, ACM Press* (Denver 1995), 43-50.

14 Lin, James, Newman, Mark W., Hong, Jason I., and Landay, James A. DENIM: An Informal Tool for Early Stage Web Site Design. In *Video poster in Extended Abstracts of Human Factors in Computing Systems: CHI 2001* (Seattle, WA March 31-April 5, 20), 205-206.

15 Site, Serena Protoype Composer Product. http://www.serena.com/products/prototype-composer/index.html.

16 *Balsamiq Mockups Product Site*. http://www.balsamiq.com/products/mockups.

17 *Breifs - A Cocoa Touch Framework for Live Wireframes*. http://giveabrief.com/.

18 *iPhone Mockup - Site*. http://iphonemockup.lkmc.ch/.

19 Hosseini-Khayat, Ali. *Distributed Wizard of Oz Usability Testing for Agile Teams*. University of Calgary, Calgary, 2010.

20 *ActiveStory Enhanced - CodePlex Project Site*. http://activestoryenhanced.codeplex.com/.