Tiquid - Creating Continuous Transitions for Multi-Touch Interactions

Georg Freitag, Michael Wegner, Michael Tränkner, Markus Wacker
Fakultät Informatik und Mathematik, HTW Dresden
Friedrich-List-Platz 1, 01069 Dresden, Germany
freitag@htw-dresden.de, wegner@htw-dresden.de,
michael.traenkner@htw-dresden.de, wacker@informatik.htw-dresden.de

ABSTRACT
Designing the *look and feel* of multi-touch applications is a challenging task, especially in the early sketching phase when it is imperative to quickly generate as many ideas as possible. There are numerous techniques and tools to conceptualize the appearance of a user interface but few solutions for rapidly creating and evaluating interaction ideas. To solve this problem, we propose a concept which enables designers to quickly create simple transitions that are controlled using continuous gestures. As proof of concept we implemented Tiquid, a multi-touch tool designed for sketching interactions during the very early stages of interface development.

Author Keywords
multi-touch, sketch, prototype, gesture, transition, look and feel

ACM Classification Keywords
D2.2 Design Tools and Techniques: User Interfaces, D.2.6 Programming Environments: Interactive environments, H.5.2 User Interfaces: Interaction styles, Prototyping

INTRODUCTION
Even though, natural user interfaces (NUI) such as multi-touch applications are commonly used and widely adopted in our everyday lives, the development of such interfaces is still challenging [9,12]. Since a common standard like the WIMP approach for GUI design is not available, NUI-designers must rely on general guidelines, design patterns, and best practice approaches [6]. This situation is further complicated by the lack of appropriate interface prototyping tools. While tools for creating the *look*, the visual aspects of an interface, are numerous, widely-adopted and well-researched [2,5], possibilities to explore the *feel* or "dynamics of the interaction" [1] are scarce. However, especially in the early stages of interaction design, it is important to have a suitable approach for creating the *feel* of an application in order to be able to analyze its usability [18].

Our concept bridges this gap by providing designers with a sketching approach that allows for the development of visual transitions as feedback to standard multi-touch gestures. Designers can define an abstract interface and easily set different states for the components using key frames. When a gesture is performed its progress is used for interpolating between the states, showing a continuous, user-controlled transition.

Tiquid, our implementation of the concept, is optimized for the quick and cheap creation of simple, disposable interaction ideas. This makes it particularly suitable for designing *feel* during the very early stages of interface development. Inspired by the works of [14] and [17], we give immediate feedback within the tool so designers can create the transitions exactly the way they want them to *feel*. Since Tiquid is a multi-touch application there is no need to switch between input methods or even devices, everything can be done in place without compilation or changing modes. The user interface design reflects this simplicity. Tiquid is based on the multi-touch prototyping framework Liquid (see Related Work).

In the next section we give a more in depth analysis of the problem after which we provide an overview of existing approaches. Next we present our own solution and discuss its advantages. The closing section will contain the conclusion and an outlook for future work.

MOTIVATION
Interface Prototyping is an important process when designing user interfaces that *look and feel* right. However, the analyses of [4,11] show that prototyping tools often neglect the *feel* aspect. This fact is confirmed by the observations of [2,5], who interviewed designers and reviewed their work process to identify shortcomings. This gap is particularly big in the beginning of the design process when *look* can be easily approximated, e.g. by using quickly scribbled sketches, whereas modelling *feel* is almost impossible. This means interaction design has to start at a much later stage, reducing the time it can be

NordiCHI '14, October 26 - 30 2014, Helsinki, Finland

ACM 978-1-4503-2542-4/14/10□$15.00.
http://dx.doi.org/10.1145/2639189.2639208

worked on and possibly limiting choices through the already established *look*.

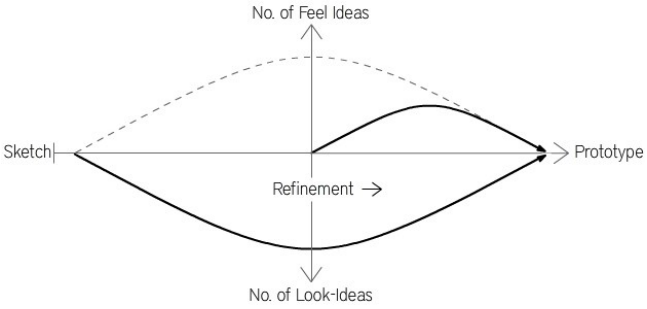

Figure 1. Adapted model of the design funnel, corrected for the imbalance between *look and feel* during the design process

Figure 1 illustrates this imbalance using a modified version of Laseau's design funnel as interpreted by Buxton [1]. Starting from a design problem or "starting point", ideas are quickly created and collected. Buxton uses the term *sketch* in this context to describe quick, inexpensive, minimalistic, disposable design suggestions. Sketches are not limited to pictures scribbled onto paper; material is not important. After enough ideas have been collected they need to be filtered and refined through design decisions and experience. In the later phase of this process, the remaining ideas are prototyped, meaning much more time is invested to allow for more informed design decisions. In the end, a specific solution or "focal point" is reached.

In Figure 1, we depict the same design process. We plotted the amount of ideas for the *look* and *feel* on the negative and positive y-axis, respectively. Since many tools for the *look* of an application are available in terms of sketching as well as prototyping, the bottom half of the diagram shows that from the very beginning there is a wide variety of ideas that can be explored. On top, however, the design funnel for the *feel* is shortened due to lack of appropriate tools for sketching interactions early on. As can be seen from the dotted curve, much potential for innovative design is lost because of this.

RELATED WORK

Numerous research projects propose tools aimed at supporting the creation of user interfaces that *look and feel* right. We focus on related works that promise a universal approach to interaction design and analyse them in regards to their practicality for sketching interactions.

The line between sketching and prototyping tools is sometimes rather fuzzy, as in the case of ProtoActive [16] which features a storyboard-like sketching approach to create "draft prototypes". Designers can sketch different states of a user interface and link between them by defining gesture areas with assigned trigger gestures. However, only discrete changes between different states are possible.

Another research team created SILK [7], which combines the benefits of paper-based sketching with the conveniences of digital editing and basic interaction. This tool recognizes shapes that were drawn and replaces them with standard user interface components. Links between various sketches are created in a storyboard view that provides a structured overview of all the sketches.

Similar to SILK, the tool described by [8] analyzes digital sketches, recognizes shapes and transforms them into sets of components. The purely *look*-focused sketching tool furthermore allows for switching between different levels of fidelity but provides no way of creating transitions.

Sketchify [13] aims to incorporate interactive components such as sensors and state-of-the-art technology into interaction sketching. Defining transitions with this tool is possible by creating example animations that are triggered by gestures. However, we consider the lack of a truly continuous input, processing and output as a big disadvantage in an otherwise well-conceived approach.

With Monet, researchers tried to overcome "the fact that previous informal prototyping tools cannot prototype continuous interactions" [10]. The application allows for sketching parts of the interface, specifying different states for these components and thereby defining a continuous interaction space. Event-triggers can be setup up to enable navigation between states. It is an extensive tool that even allows for compound behaviour but is mainly targeted at mouse or pen input.

Liquid [3] is a multi-touch framework for prototyping multi-touch applications using visual programming. It enables the creation of elaborate user interfaces with an intricate, interconnecting structure and rich interactions. However, it is much too complex for quickly creating sketches. Tiquid was conceived as a supplement to Liquid to address this issue. Using Tiquid designers can create continuous interactions like in Monet or Liquid, but optimized for touch input and in a simpler way more suited for the early design process.

THE TOOL

The goal of Tiquid is to make creating and evaluating transitions as easy as sketching the *look* of an interface. Therefore, we designed the tool to be simple to use, sacrificing fidelity for the ability to rapidly sketch interactions. A transition in Tiquid behaves much like a classical computer animation, where attributes like position, rotation or transparency of objects are interpolated between key frames. However, unlike animations, we do not interpolate over time but instead over the progress of a gesture as it is being performed. Designers can create components, specify their start and end states, as well as define a gesture for the transition. Afterwards, they can directly test the result by performing the gesture. Like the blank piece of paper a designer draws a sketch on, our canvas contains only the elements belonging to the transition. No sidebars or similar controls are needed since all editing is done directly, using gestures and context menus.

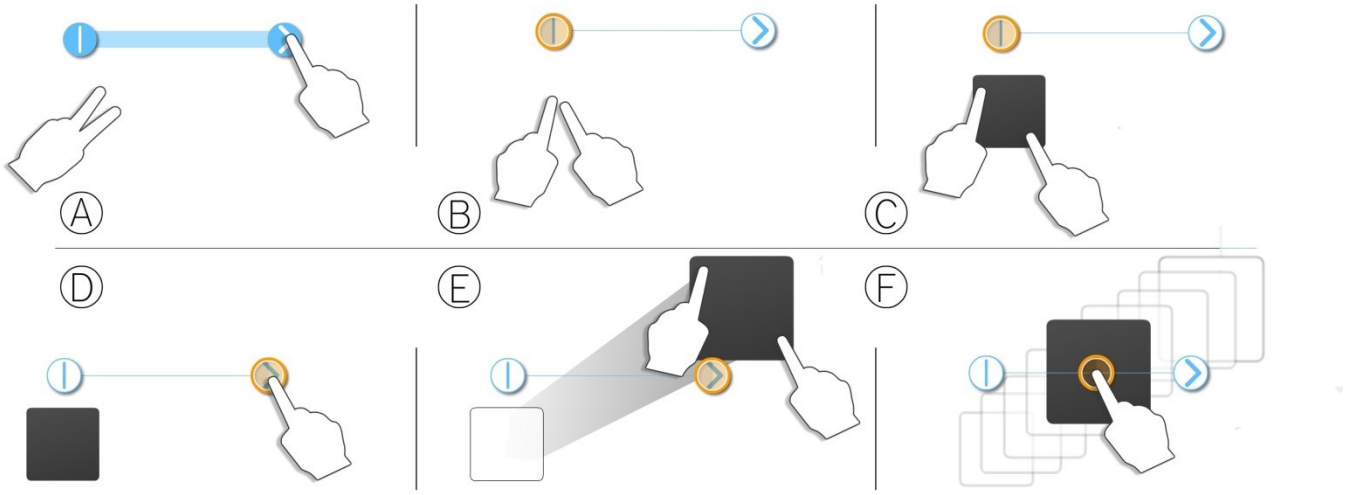

Figure 2. Steps in the creation of a transition using Tiquid. A) Define a gesture using quasimode (holding two fingers down); B and C) Creation of a new component at the starting key frame using a spread gesture; D) Navigation to end key frame of the gesture; E) Setting a new state for the component by scaling it up; F) Testing the transition by executing the gesture

Components

Consistent with our low-fidelity sketching approach, the available components for modelling a user interface are limited to basic shapes. They can be created by performing a spread gesture on the background (see Figure 2B and C) and deleted using a pinch gesture. Basic manipulations like positioning, resizing, and rotating can be performed directly on an object. Additional attributes like transparency, 3D rotation, and blur can be set using a menu that is accessed by tapping the component. To sketch the behavior of a component during a transition the user simply sets different attribute values for different key frames (see Figure 2E).

Gestures

Gesture definition is handled using a quasimode [15]. It can be entered by placing and holding two fingers on the canvas and is stopped as soon as the fingers leave the display (see Figure 2A). To avoid conflicts with component creation and manipulation gestures, the distance between the fingers has to be above a certain threshold and the fingers should not move. While the quasimode is active, designers can perform a gesture supported by Tiquid (see Figure 3) with their free hand. The gesture's position, length, and direction are recorded and visualized on the canvas with the help of a proxy. Now, the users can either manipulate the gesture by moving the proxy's start or end point or they can draw a new gesture. Once they are satisfied, they can leave the quasimode for gesture definition by lifting up their fingers. The proxy for the gesture remains on the canvas as a simple visualization.

Transition

After defining the gesture, it can be performed to control the transition (see Figure 2F). The start-position is not important, however, direction and length are taken into account to measure the progress of the transition. A cursor visualises the current progress directly on the proxy itself. Users can stop performing the gesture at any time and the progress will freeze there. To jump to a certain point in the progress of a gesture/transition it is also possible to drag the cursor directly (see Figure 2D).

If one of the components is manipulated, a key frame is inserted at the current point of progress. Initially, a transition only has two key frames (start and end) but additional ones can be added in this manner (see Figure 3). Different states for the components can be defined very quickly using this approach. Note that key frames act on each component. This makes it possible to create complex transitions that act on multiple components. Once a key frame has been defined, it is possible to jump there by tapping its representation on the gesture's proxy. To remove a key frame it is simply pulled out from the proxy.

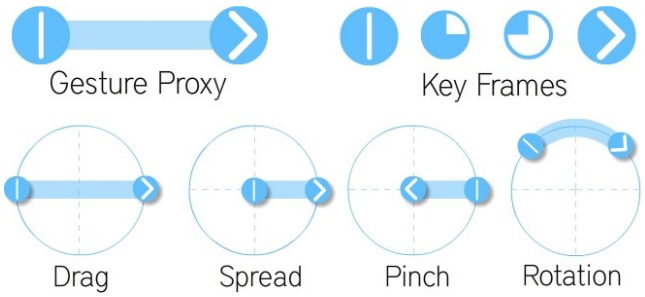

Figure 3. Top Left: Visualization of a drag gesture using a proxy. Top right: Key frame representations for different states of progress (start, one quarter, three quarters, end). Bottom: Gestures supported by Tiquid as depicted during gesture definition

Interface Concept/Summary

The main challenge for Tiquid's user interface was the visualization of the entire transition, including gesture, progress, and key frames without using a progress bar or a timeline, found in classical animation tools. We think that our solution, of visualizing all these features in the gesture

proxy, provides several benefits beyond uncluttering the interface. Unlike a fixed, horizontal progress bar direction and length of the gesture can be derived directly from the proxy. This design helps performing the gesture and enables direct feedback about gesture length and direction, even while still editing the transition. The gap between what users are doing and what they are seeing is minimal, allowing for sketching and testing at the same time. Additionally, the gesture can easily be performed in reverse to get to earlier points of the transition.

To test the transition freely without the risk of manipulating components or the gesture visualization, it is possible to fade-out the editing layer, which removes all clutter like gesture proxy or component menus. All that remains are the components themselves and by performing the gesture anywhere on screen the sketched transition can be previewed in a more presentable manner.

CONCLUSION AND FUTURE WORK

We introduced a concept for sketching continuous transitions in the early stages of interaction design. It allows for creating basic interface components and defining transitions that are controlled by gestures. This way, ideas for interactions can be tested quickly. We designed Tiquid, so that the results would fulfil Buxton's attributes for a sketch: it supports a workflow that is quick in terms of effort, inexpensive in cost, minimal in detail, clear in vocabulary, and focused on a single aspect. We brought this approach into the context of interaction design and focused on gesture interactions for touch-devices in regards to the *feel* and continuity of transitions.

We are currently planning an evaluation to proof our concept. Furthermore, our approach is amenable to numerous extensions. For example, storyboarding could be included to link individual transitions. Hereby, a more complex process with multiple screens and interactions could be sketched and tested.

REFERENCES

1. Buxton, B. Sketching User Experiences: Getting the Design Right and the Right Design. (2007).
2. Carter, A.S. and Hundhausen, C.D. How is User Interface Prototyping Really Done in Practice? A Survey of User Interface Designers. *2010 IEEE Symp. VL/HCC*, (2010), 207–211.
3. Freitag, G., Kammer, D., Tränkner, M., Wacker, M., and Groh, R. Liquid : Library for Interactive User Interface Development. *Mensch & Computer 2011*, Oldenbourg Verlag (2011), 202–210.
4. Freitag, G., Wegner, M., Tränkner, M., and Wacker, M. Look without Feel-A Basal Gap in the Multi-Touch Prototyping Process. *Mensch & Computer 2013*, Oldenbourg Verlag (2013), 159–168.
5. Grigoreanu, V., Fernandez, R., Inkpen, K., and Robertson, G. What designers want: Needs of interactive application designers. *2009 IEEE Symp VL/HCC*, (2009), 139–146.
6. Hesselmann, T., Boll, S., and Heuten, W. SCIVA. *Proceedings of the 3rd ACM SIGCHI symposium on Engineering interactive computing systems - EICS '11*, ACM Press (2011), 191.
7. James, A. and Brad, A. Sketching Interfaces : Toward More Human. (2001).
8. Kieffer, S., Coyette, A., and Vanderdonckt, J. User Interface Design by Sketching : A Complexity Analysis of Widget Representations. *Proc. EICS'10*, (2010), 57–66.
9. Lee, J.C. In search of a natural gesture. *XRDS: Crossroads, The ACM Magazine for Students 16*, 4 (2010), 9.
10. Li, Y. and Landay, J.A. Informal prototyping of continuous graphical interactions by demonstration. *Proc. UIST '05*, ACM Press (2005), 221–230.
11. Myers, B., Park, S.Y., Nakano, Y., Mueller, G., and Ko, A. How designers design and program interactive behaviors. *2008 IEEE Symp. VL/HCC*, (2008), 177–184.
12. Norman, D.A. Natural User Interfaces Are Not Natural. (2010), 6–10.
13. Obrenovic, Ž. and Martens, J.-B. Sketching interactive systems with sketchify. *ACM Transactions on Computer-Human Interaction 18*, 1 (2011), 1–38.
14. Quevedo-Fernández, J. and Martens, J. idAnimate: A General-Purpose Animation Sketching Tool for Multi-Touch Devices. *CONTENT 2013, The Fifth ...*, (2013), 38–47.
15. Raskin, J. *The Humane Interface: New Directions for Designing Interactive Systems*. Addison-Wesley Professional, 2000.
16. De Souza Alcantara, T., Ferreira, J., and Maurer, F. Interactive prototyping of tabletop and surface applications. *Proc. EICS '13*, ACM Press (2013), 229–238.
17. Victor, B. A worry dream. http://worrydream.com/#!/InventingOnPrinciple.
18. Wobbrock, J.O., Morris, M.R., and Wilson, A.D. User-defined gestures for surface computing. *Proceedings of the 27th international conference on Human factors in computing systems - CHI 09*, ACM Press (2009), 1083.