SurfaceShop: Techniques for Complex Adjustments in Multi-Touch Computing

E. J. Kalafarski
Department of Computer Science
Brown University

Abstract

Though the commercial application of multi-touch is expanding rapidly, there exists tremendous opportunity for innovation and influencing the development of interaction techniques within this design space. We demonstrate a technique for executing complex and precise adjustments within the medium: rearrangeable "tokens" that exploit the simultaneous interactions, modeless underpinnings, and collaborative nature of the multi-touch environment. Users perform a two-fingered "pinch" gesture directly on these tokens to effect an adjustment in value, allowing for considerable combination and creativity by the user. We implement this technique in an application for image manipulation and evaluate it in a preliminary user trial, which demonstrates surprising discoverability and versatility of our technique and, by extension, the medium as a whole.

1 Background

Though the commercial application of multi-touch is becoming widespread—appearing in various implementations and sizes on cell phones, tablet PCs, surfaces, walls, and the nightly news—software development for the multi-touch environment remains in its infancy. There exists little standardization for the basic concepts and objects, such as one finds in a more traditional GUI—namely, the menus and pointers of the ubiquitous WIMP-style interface (windows, icons, menus, and pointers).

The number of multi-touch applications available commercially on various devices measures in the tens of thousands, but the functionality of these devices are narrow and largely organizational: multitudes of mapping, photo organizing, and game applications exist, but no real attempt has been made to transition a complex, industry-standard application, such as image manipulation or spreadsheets, to the multi-touch environment. Likewise, no serious attempt has been made to combine multi-touch with existing and proven, productive interaction techniques, such as those used with pen and keyboard. As a result, the benefits of multi-touch to existing application problems remains largely unexplored.

With this gap in mind, we focused on the question of what an industry-standard, complex application would look like in the multi-touch environment—and more importantly, what advantages unique to the environment it could exploit. Though we explored several complex applications, such as 3-D modeling and architectural diagramming, we gravitated towards image manipulation—a complex job defined considerably by the WIMP interface. Destructive tasks in image manipulation, such as the application of filters and the adjustment of image properties, are largely sequential, accomplished by a series of dialog boxes. The user is constantly switching between dozens of tools, and the user's hand rarely leaves the mouse.

We identify these inherent inefficiencies as areas in which image manipulation stands to potentially gain considerably from the introduction of new techniques. The tasks presented by image manipulation are numerous and complex, but it is nonetheless an application so commonly-used in industry as to have generated its own verb ("Photoshopping"). As a fundamentally creative endeavor, image manipulation is currently limited by a very narrow interaction paradigm.

2 Related Work

We build on a broad base of prior hardware [Han 2006] and software [Varcholik 2008] work in constructing a low-cost environment for developing multi-touch techniques. From this relatively consistent starting point, modest innovation within the design space has been accomplished in a relatively short timeframe. Techniques have been developed for physics-based manipulation of rigid objects [Microsoft 2008] and the deformation of fluid objects using several fingers at once [Moscovich 2006].

Methods which provide more information than simple contact location are also available. Frameworks regularly infer the direction of a user based on the properties of a contact [Microsoft 2008]. Various configurations and technologies are capable of extrapolating the location [Dohse et al. 2008] and even the owner [Dietz et al. 2001] of the hand initiating the contact, allowing the surface to distinguish between the contacts of different users. Advanced surfaces encourage collaboration by assigning a secondary device, like a laptop or a pen, to each contributor [Wigdor et al. 2009].

Building off of theories that more than one "mode" is necessary for most modern interfaces [Buxton 1985], further work attempted to translate states common to the mouse into multi-touch, leading to the *SimPress* technique [Benko 2007], in which the hover and click states are simulated by "rocking" the finger back and forth on the table, and the *Take-Off* technique, in which a "click" is not triggered until the user's finger leaves the surface.

Separately, the continuing development of the physics-based desktop interface *BumpTop* [Agarawala et al. 2006] illustrates the utility of interface elements that respond to force and momentum in a manner that users are familiar with in the real world. Common operations such as stacking objects, putting them into piles, or throwing one object at another are inherently familiar to the user but take on new meaning when applied to operations of the computer desktop, such as grouping objects semantically or placing a file in a folder. We intend to build off of these successes by leveraging the inherent physicality of multi-touch to build intuitive associations for the user without cluttering the interface.

3 Design

We targeted what we identified as deficiencies or inefficiencies in current image manipulation applications. Based on our own criteria, these needed to be issues that were inherent to the WIMP paradigm itself, and not simply problems that were correctable with alteration to the point-and-click software. By nature, the mouse pointer is only at a single point at a giving time, and can thus only target one object or modify one value at once, short of premeditative grouping or batching. Current WIMP interfaces thus have a built-in serialization of operations, in which a user must complete one incremental operation before he or she can begin another. The combined effects of two operations can only be seen by alternating between them. For that matter, the windowing system of modern GUIs lead most operations to affect one target at a time—generally, the content of the window that currently has focus.

Our intent was then to 1) disassociate adjustment operations from the *serialized* concept of dialog boxes, 2) disassociate value-based properties from the concept of a slider with a *single point of contact*, and 3) disassociate functions from the *stationary* concept of a slider or input box.

Along the same vein, we also targeted the collaborative nature of multi-touch. In existing WIMP image manipulation environments, one image or window necessarily has focus at any given time. In a collaborative environment, however, it is entirely feasible that two users

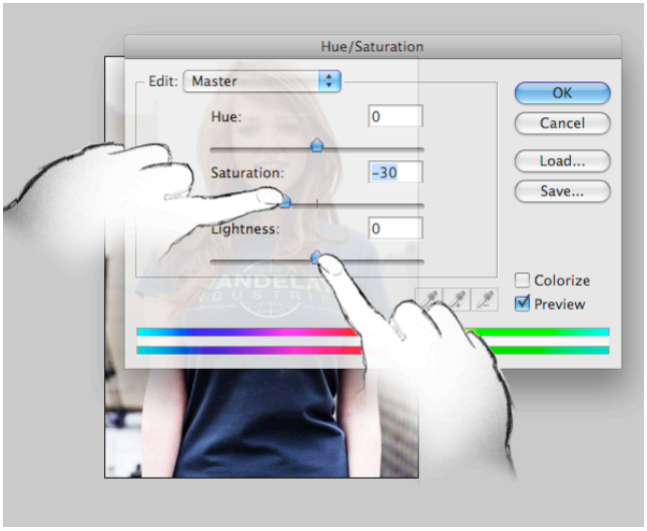

Figure 1: Storyboard envisioning multiple adjustments simultaneously

on opposite sides of the surface may be interested in using different tools or adjusting different images without having to wait for the other to finish with the image that currently has "focus." We sought the elimination of this *modality*.

To these ends, we developed the concept of a token, a physical representation for a specific property of an image, such as brightness, contrast, or a filter such as Sharpen. The token has no geographic limitations, as a WIMP slider fixed to a traditional dialog box does. Rather, these tokens are rearrangeable to the liking of the user, allowing him to create his own geographic associations. Further, we exploit the new mobility of these properties to eliminate modality. Instead of an image or window that has focus above the others, a token at any given time affects the image *closest* to it. A visual cue illustrates this proximity with a nonintrusive line from the token to its closest image.

The property assigned to a token is adjusted by the user performing a two-finger gesture on top of the token—the distance between the user's finger directly sets the absolute value of the property. No other restrictions are placed on the token: the token can be moved during adjustment, stacked on top of other tokens, flicked to the other side of the surface, etc. Tokens that are not currently needed can be dropped into a radial menu until the user decided to drag them out again. Our intent was to explore the possibilities available to the user when the physical representation of these properties could be arranged, organized, and used simultaneously within the multi-touch design space.

4 Implementation

4.1 In-house "Home Brew" Surface

From proven low-cost methods [Han 2005] we set about constructing a multi-touch surface based on *frustrated total internal reflection* (FTIR), a system using infrared light and image processing. After extensive trial and error, we arrived at a configuration sensitive to very modest contacts on the surface, yet discriminating-enough to filter out and ignore ambient light, marks left from previous contact events, and artifacts created by the projector.

A 3/8"-thick piece of clear acrylic acts as our surface. The edges of this sheet is polished clear to allow optical transmittance: a series of 12 infrared LEDs placed at intervals around the edges and pointed into the plane of the sheet "edge-lights" the surface, creating the condition called total internal reflection. A finger placed on the top of the surface *frustrates* this internally-reflected light,

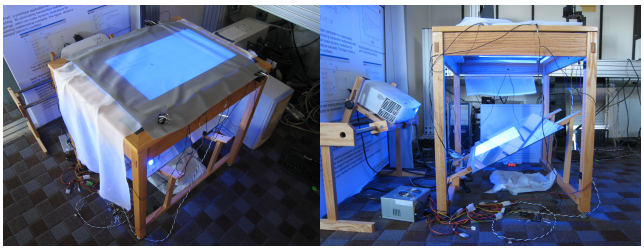

Figure 2: Overhead (left) and side views of FTIR multi-touch setup. Note the use of a mirror to shorten the effective throw of the projector.

creating a bright spot on the underside of the acrylic that can be captured by camera.

A sheet of grey Rosco laid above the acrylic provides a screen for the necessary rear-projection creating the surface image. This, however, prevents direct contact between fingers and the acrylic; a layer of silicone poured and set directly on the acrylic allows the internally-reflected light to be frustrated with *pressure* instead of direct contact. A translucent piece of fabric interface above this layer prevents the Rosco from sticking to the silicone.

We use an inexpensive Microsoft LifeCam VX-6000 to capture the image. Using a mirror to increase the distance between the camera and the underside of the acrylic, even the camera's low field of vision was able to capture the entire acrylic. What the camera sees is thus a black-and-white image of the underside of the acrylic—totally dark under most circumstances, except when a finger contact on the opposite side of the acrylic frustrates the internally-reflected light, creating a bright spot on the underside that was seen by the camera.

Much was learned from this construction on the strengths and weaknesses of FTIR, the same technology employed by the Microsoft Surface. Contact recognizability was likely to vary across the surface, especially in environments with harsh lighting—it was thus generally undesirable to fix interface elements in specific locations on the interface, as it may, for some users, be in a "dead zone" specific to their ambient conditions. Furthermore, given the "pressure-sensitive" construction of the device, it was uncomfortable for the user's finger when required to maintain contact with an element for more than several seconds. This deficiency did not carry over to the Microsoft Surface per se, but the unfortunate side effect was an effective reminder not to force the user to "babysit" a token or interface element when it was not necessary.

4.2 Bespoke Multi-Touch Framework

For development on our constructed surface, we employed the Bespoke Multi-Touch Framework [Varcholik 2008], an XNA-based framework authored by a Ph.D. candidate at the University of Central Florida. The framework provides low-level blob detection and processing of the captured image, offering several classes to the developer containing the blob characteristics.

With this framework, we began development of the application using Microsoft's XNA tool set—technologies intended for rapid game development and management—and were able to quickly implement an interface that responded with elementary physics to contact events. Image manipulations were implemented with XNA's pixel shaders, and a limited number of simultaneous adjustments proved effective. This success belies the difficulty of several crucial tasks: developing reliable physics for elements of the interface; re-implementing traditional image manipulations that would be non-destructive, as to allow real-time combination with other effects; and the development of a gesture-recognition framework and accompanying gesture library. The necessity of this underlying architecture forced us to explore other options.

4.3 Microsoft Surface and Microsoft Surface SDK

The availability of the Microsoft Surface, with its accompanying software development kit, allowed us to begin to focus instead on higher-level interaction issues. The Windows Presentation Foundation "flavor" of the SDK had several advantages built-in, including new and traditional Windows controls and more-sophisticated physics, allowing all objects to exhibit momentum, deflection, etc. With this legwork provided, it took only about two weeks to recreate five months of low-level development on our home-brew option. Basic image manipulation was accomplished powerfully and with very

Figure 3: Microsoft Surface

low-latency by exploiting pixel shaders executed on the Surface's GPU.

Despite the convenience of the SDK's built-in controls and physics, moderate extension of its capabilities was required. WPF supports the detection of events directly on interface elements, but our token concept required recognition of a second contact within the neighborhood of the token as well. The recognition of this second contact necessitated the introduction of several "invisible" interface elements on which contacts could be captured. We look forward to the inclusion of robust abstract gesture recognition and a larger gesture library, as has developed throughout the pen computing community, in future iterations of the Surface software.

Figure 4: Rearrangeable tokens implemented in "SurfaceShop." A user adjusts an attribute with a "pinch" gesture on top of its token, the value governed by the distance between his two fingers.

4.4 Influences

We obviously exploit the insights of physics-based systems such as BumpTop [Agarawala 2006] in our implementation of WPF-based tokens and image canvases that react intuitively to pushes, slides, and flicks. Though physics-based desktops are not exclusively a multi-touch conceit, the prominent role of proximity in the function of our technique means we can build intuitively off of behavior the user expects in the objects he manipulates. We very simply exploit this property of which a user has a pre-existing grasp—the distance between two objects.

We also leverage the Surface's byte tag recognition framework to implement an elementary airbrush mode, controlled with a physical icon (phicon) that is placed on the Surface. A phicon in the shape of an inkwell locally affects the mode of its surrounding images; a contact on any of the affected images, instead of the standard WPF manipulations of translate, scale, etc., spreads paint on the image with shape and pressure governed by the contact finger. This concept of a function-specific phicon is obviously inspired by previous work on the concept, such as seminal work conducted at MIT [Ishii 1997], but with a concept of strict localization added—the phicon explicitly affects the mode of the elements in close proximity to it.

5 Evaluation

With particular interest in the usability of our techniques, we performed a pilot user trial. 11 participants over a five-day period (April 8 – April 12, 2009) were observed using our application and responded to a verbal interview afterwards.

Though all participants were Computer Science graduate students, they represented a cross-section of several crucial characteristics. Roughly half (6 out of 11) had experience with Photoshop, GIMP, or comparable image manipulation applications, while the rest did not. Just over half (7 out 11) had some experience with multi-touch devices (e.g., an iPhone or a Blackberry Storm).

5.1 Trial design

Participants were presented with the SurfaceShop environment with three open images and asked to complete three tasks of increasing complexity:

- Adjust a single property (e.g., the brightness) of one of the images
- Adjust two properties of an image simultaneously (e.g., the brightness and the contrast) and examine combinations of the effects

- Apply a filter (e.g., blurring) to all three images as a batch operation

Furthermore, they completed each of these tasks three times, each in an environment presenting an interaction style with varying degree of modification from a standard WIMP interface. The first style mimicked a WIMP interface as closely as possible, presenting the user with a slider for each property or filter. The slider was manipulated with a single finger dragging along it (as a point-and-click slider would be), movable to the extent a dialog box might be, and—as in modern programs such as Photoshop and GIMP—the image with *focus*, or currently "on top" of the others, was the image affected. As in contemporary WIMP interfaces, users brought the focus to a window with a single tap.

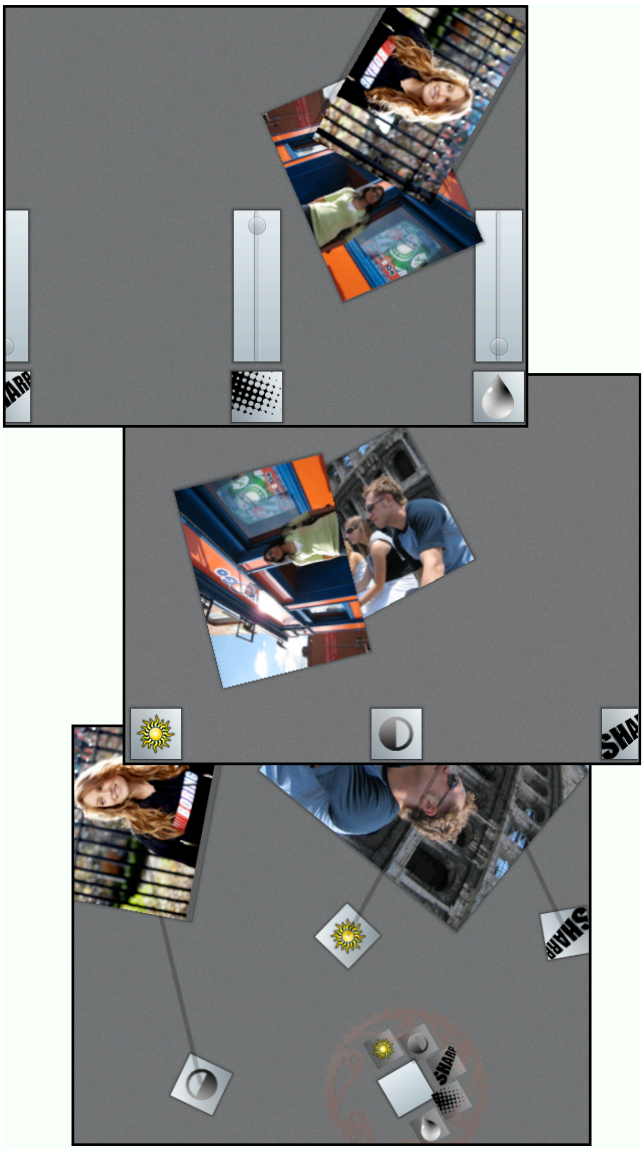

Figure 5: Details of interaction styles; from top: sliders, stationary tokens, and rearrangeable tokens

The second style added only the gesture element of our concepts: although the mechanism for each filter remained stationary, sliders were replaced with tokens whose values were adjusted with our two finger technique. As before, the image with focus was the one affected.

The third and final style was a full implementation of our concepts. Tokens were fully movable and rearrangeable, and were created by the user "tearing" them off a movable radial menu. The adjustment of a token no longer affected an image having focus, but rather the closest image.

The three interaction styles were presented in random order to each participant. The behavior of each interaction style was briefly described verbally to each participant, but no visual demonstration was provided. Participants were observed accomplishing the tasks in each of the three interaction styles, and then asked which style they preferred in regards to completing the tasks as a whole. They were additionally asked to comment on which style they found most intuitive, most precise, and most efficient for batch operations.

We also solicited feedback on our elementary airbrush tool, triggered by placing a physical icon (phicon) on the Surface in proximity of the image to be airbrushed.

5.2 Observations

Six out of the 11 participants selected the rearrangeable tokens as their preferred interaction style for accomplishing the tasks, commonly citing the ease of selecting the target image. (One user "liked the idea of taking a [token] and moving it close to a picture" to perform an operation; another enjoyed "doing it fast [by] just by moving stuff around.") Tellingly, of the three users who said they preferred sliders instead, all gave the same reason: they appreciated a visual cue as to the range of the available values ("You could see the range explicitly"). One mentioned, unprompted, that the addition of this visual cue to the tokens would have aided him considerably. Two participants selected no preference.

All users were successful at the individual tasks, though they executed the adjustments in wildly varying ways. In the adjustment of a property's value, several users made use of the index and middle finger to define a value, instead of the more popular approach of using the index finger and thumb. This was seen both during the adjustment of a single property and the adjustment of more than one

property simultaneously. Other alternatives discovered included:

- One user used his two index fingers for a single adjustment operation, calling it "more exact" than using two fingers from the same hand.
- Two users moved each token directly on top of the image to be adjusted, instead of simply close to it.
- One user moved two tokens close to each other and used a third finger to adjust their values simultaneously, essentially linking their values together.
- One user determined that he did not need to physically "slide" to the new value, but instead could jump to the new value by tapping the token with his fingers held rigidly at a predefined distance.

Likewise, users discovered a myriad of methods for performing a batch operation on several images. Several users went with the ostensibly "obvious" method of moving a token close to all the images in succession, performing the value adjustment each time. But other users found a variety of successful alternatives:

- One user held the token stationary with her dominant hand, using her non-dominant hand to instead move the images into proximity of the token, one-by-one, to perform the adjustment.
- One user set the token to the desired value and dragged it as such, with both fingers, close to each image, affecting all in rapid succession.
- Two users created additional copies of the required token, one for each of the three images to be manipulated. The first of these users tried to use six fingers to make the three adjustments simultaneously, in an attempt to synchronize the adjustment amounts.
- The second user creating three copies of this token was able to use his two fingers held rigidly at a predefined distance and tap on each token in rapid succession, effecting the batch adjustment extremely quickly.

Eight out of 11 participants responded positively to using the phicon to toggle a section of the application into airbrush mode, expressing confidence in becoming "fluent in it" given time and calling it "intuitive," "super easy," and "a great idea." Notably, seven of the 11 participants forgot to move the phicon off the Surface when asked to resize the image before continuing to paint (resulting in accidentally painting on the image with their "resize" gesture), although almost all expressed confidence in not repeating that mistake once they "got used to it." One user noted the utility of simply tipping over the phicon to leave airbrush mode, and one simply moved the phicon out of proximity of the image in question.

5.3 Discussion

Although most users quickly singled out the most traditional interaction style, the sliders, as the most intuitive, they uniformly identified the reason for this as their familiarity with the device. One user summed it up thusly: "I've seen sliders before. They're pretty standard." We infer that multi-touch interaction and gestures more advanced than simply dragging around objects are likely to encounter a learning curve similar to that of pen-based gestures. (Surprisingly, three users called rearrangeable tokens the most intuitive; all three identified themselves as previous users of multi-touch devices such as iPhones. According to one, "there's something natural about this movement of your fingers.")

A majority of users felt that the interaction style with physical sliders was the most precise, despite this style's complete lack of a numerical visual output. This reinforces our inference that displaying the visible range of available values was reassuring to users, creating a perceived precision where none actually existed. It should be trivial to add this same visible range to a token when a gesture is being executed on top of it.

Perhaps most exciting were the recurring and varying demonstrations of the versatility that our implementation engendered from user to user. While the operation of each function specifies certain interaction parameters—in the case of value adjustment, the manipulation of two contacts with at least one on top of a specific token—the flexibility of the medium and our exploitation of it led to an immensely interesting variety of techniques. Users were able to show the author techniques and functionality in his software that he had not previously envisioned.

Further along these lines, users began to infer functionality where none existed yet. More than one user attempted to perform batch operations to several images *simultaneously* by moving the images into an overlapping cluster before performing the adjustment nearby. (In the environment we presented them with, this merely affected the image that was truly closest to the token.) This not only suggests a new functionality to us, but it is an implicit validation of the concept of proximity as a selector. Users found the idea of moving a tool towards an image to modify it intuitive enough that they began to independently extend the metaphor to other concepts such as batch operations. Closeness of the manipulable objects on the Surface was shown to be a powerful visual cue.

5.4 Conclusions

Though the results of this informal trial are largely anecdotal, they are consistent enough that we can

confidently draw some basic conclusions about strengths and weaknesses. Users indicated that our technique of property adjustments disassociated from fixed form, location, and serialized order was effective and efficient. The solitary consistent complaint was the lack of visual cues that traditional sliders afforded. Unobtrusive visual cues at or surrounding the point of contact are likely to substantially resolve this. (Indeed, additional visual cues providing contact feedback have been added by Microsoft to the recent release of the Surface SDK Service Pack 1.)

It is extraordinarily encouraging to see that not only were users able to exploit the token scheme in different ways, but that all methods were successful. A design space in which the user is *allowed* to innovate with even simple constructs and tools is exciting and arguably an asset to the creativity of the user.

6 Moving Forward

Based on the conclusions above, there are some obvious avenues for immediate incremental improvement. It will be relatively trivial to combine the power of the rearrangeable tokens with the perceived "precision" advantage of the slider with the inclusion of logical but non-intrusive visual cues. For a user performing an adjustment with our two-finger gesture, a slider should appear *under and aligned with* the gesture itself, to provide the user with visual feedback and a representation of the range of available values. This will address the vast majority of user complaints without cluttering the interface or diminishing from the utility of the gesture itself—the user's perception will be that of adjusting an abstract slider of the user's own creation and control.

Moving forward, we continue to develop intuitive gestures for cropping and selection, as well as phicon functionality for color selection and "spot healing" tools. There is much merit to the consideration of a functional dichotomy between phicons and gestures, as WIMP interfaces are equate menu items and keyboard shortcuts. Gestures, although perhaps limited in expressiveness, are immediate and efficient for common or quick tasks that require little customization, as are keyboard shortcuts. When more precision and customization is needed, the user has the option of reaching for the more powerful phicon, analogous to accessing menus or toolbars for the appropriate specialized dialog. A given function could be accessible with both methods to different degrees: for example, a user could hold his fingers in a "frame" gesture to effect quick and rough cropping; for more precise cropping, a phicon could toggle the image into cropping mode, allowing a more detailed adjustment with the user's fingertips.

Further support for collaboration is necessary. As more robust gesture recognition becomes available, we envision gestures that may be assignable to individual users, in order to bring up a personalized menu or commonly-executed personal function.

We believe our technique is by no means limited to image manipulation. This trial is clearly a demonstration of a single potential technique within a large and emerging design space. Value adjustment and mode changes are not unique to image manipulation, and we envision the extensibility of this kind of value assignment to many other applications.

Acknowledgements

The author wishes to give acknowledgement and thanks for the advice, encouragement, and assistance of Andries van Dam, Andrew Forsberg, and Robert Zeleznik. Special praise is reserved for the help and hard work of Mark Oribello. Thanks go to Paul Oka of Microsoft for early discussions about multi-touch and photo manipulation.

References

Agarawala, Anand and Balakrishnan, Ravin. 2006. Keepin' it real: pushing the desktop metaphor with physics, piles and the pen. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*. 1283–1292.

Benko, H, Wilson, AD, Baudisch, P. 2007. Precise Selection Techniques for Multi-Touch Screens. US Patent App. 11/379,297.

Buxton, W. 1985. Issues and Techniques in Touch-Sensitive Tablet Input. In *Proc. ACM SIGGRAPH '85*. 215-224.

Dietz, P, and Leigh, D. 2001. DiamondTouch: a multi-user touch technology. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*. 219–226.

Dohse, K, Dohse, T, Still, J, Parkhurst, D. 2008. Enhancing Multi-user Interaction with Multi-touch Tabletop Displays Using Hand Tracking. In *2008 First International Conference on Advances in Computer-Human Interaction*. 297–302.

Han, J. 2005. Low-cost multi-touch sensing through frustrated total internal reflection. In *Proceedings of the 18th annual ACM symposium on User interface software and technology*. 115–118.

Ishii, H, Ullmer, B. 1997. Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 234–241.

Microsoft, Inc. 2008. Microsoft Surface SDK 1.0.

Microsoft, Inc. 2009. Microsoft Surface SDK SP1.

Moscovich, T. 2006. Multi-touch interaction. In *CHI '06 extended abstracts on Human factors in computing systems*. 1775–1778.

Varcholik, P. 2008. Bespoke Multi-Touch Framework Version 4.2.

Wigdor, D, Jiang, H, Forlines, C, Borkin, M, Shen, C. 2009. The WeSpace: The Design, Development, and Deployment of a Walk-Up and Share Multi-Surface Visual Collaboration System. In *Proceedings of the 27th international conference on Human-Computer Interaction*.