Medusa: A Proximity-Aware Multi-touch Tabletop

Michelle Annett [1,2], Tovi Grossman [1], Daniel Wigdor [3], George Fitzmaurice [1]

[1] User Interface Group
Autodesk Research
{firstname.lastname}@autodesk.com

[2] Department of Computing Science
University of Alberta
mkannett@ualberta.ca

[3] Department of Computer Science
University of Toronto
dwigdor@dgp.toronto.edu

ABSTRACT

We present Medusa, a proximity-aware multi-touch tabletop. Medusa uses 138 inexpensive proximity sensors to: detect a user's presence and location, determine body and arm locations, distinguish between the right and left arms, and map touch point to specific users and specific hands. Our tracking algorithms and hardware designs are described. Exploring this unique design, we develop and report on a collection of interactions enabled by Medusa in support of multi-user collaborative design, specifically within the context of Proxi-Sketch, a multi-user UI prototyping tool. We discuss design issues, system implementation, limitations, and generalizable concepts throughout the paper.

ACM Classification: H5.2 Information interfaces and presentation: Input, User Interfaces.

General terms: Design, Human Factors

Keywords: tabletop, context-aware, gestures, touch, bimanual, multi-touch, proxemics, proximity, hover

INTRODUCTION

Multi-touch tabletop displays, which combine large display and input surfaces on a horizontal plane, offer opportunities for new applications and interfaces. Advances in technology have made such devices the focus of considerable research [22] and have given rise to numerous prototype and commercial platforms. Multi-touch tabletops offer numerous potential benefits, such as the ability to support a more "natural" user experience [36], and allow for casual and collaborative interactions [25].

Despite these benefits, many tabletops still have several limitations. In particular, the majority of touch-based technologies are only capable of sensing touch interaction *on* the display surface. As a direct result, tabletop user interfaces cannot rely on a "tracking state" which is one of three essential input states of traditional GUI's [7]. Furthermore, by limiting their sensing to touch, tabletop devices have no inherent knowledge of where users are situated which users are interacting, or the number of users (if any) that are present. Given the clear relevance of tabletops to collaborative environments, these can be real limitations, and indeed, many research projects have developed software solutions that provide tabletops with this additional information. For example, a tabletop display may attempt to deduce a user's location based on the orientation and footprint of the user's touch points [8,32].

While software solutions are a worthwhile approach, there are also new technologies that allow tabletop devices to sense not only contact with a tabletop's surface, but also above and around the tabletop. For example, depth cameras have enabled interactions between and around display devices [35]. Similarly, the use of proximity sensors has enabled new gestures above and around a device [19].

As these sensing technologies have only recently become commercially available, little research has explored how their additional input channels can be leveraged to augment and improve multi-touch tabletops. We see the integration of tabletop and proximity sensing technologies as opening a range of possibilities to enhance existing multi-touch and collaborative interactions, and enable new ones. In particular, context-aware and proxemic interactions can be enabled [2,29], and existing challenges, such as mapping touch points to users, and users to locations, can be addressed.

Our work provides two primary contributions in this space. First, we present an integrated hardware solution, *Medusa*, which allows a traditional multi-touch tabletop to sense a user around its perimeter, as well as the user's interaction above the surface. Medusa is a Microsoft Surface that has been instrumented with 138 proximity sensors. These proximity sensors enable the Surface to sense the users around it, as well as the hands and arms above its display (Figure 1). Not only are these sensors inexpensive and simple to configure, but also they enable an integrated

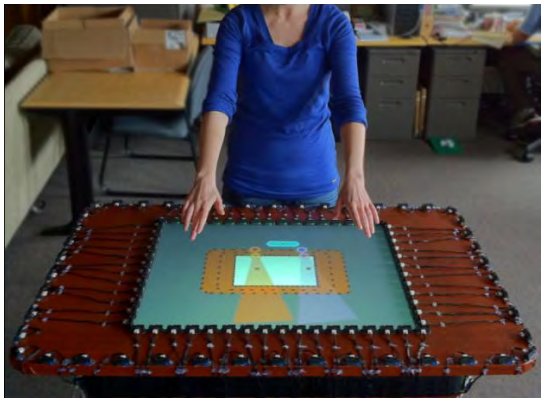

Figure 1: A user waving her arms above Medusa, a proximity-aware multi-touch tabletop.

UIST '11,October 16-19, 2011, Santa Barbara, CA, USA.

hardware solution, without requiring any markers, cameras, or other sensing devices external to the display platform itself. Our second contribution is a thorough exploration of the design space that results from the use of these additional information channels. As Medusa has an awareness of users' locations, it can for example, identify touch points by user, and disambiguate between touches made with left or right hands. Our exploration of new user interface paradigms and interaction techniques is facilitated by the development of Proxi-Sketch, a UI prototyping application.

RELATED WORK

In this section, we review previous efforts at sensing gestures around touch devices, responding to user location, and alternative sensing technologies.

Sensing Gestures around Touch Devices

There is a long line of research examining the use of explicit mid-air gestures, in particular, in VR research (we refer the reader to two surveys [13,23]). Freehand gestures have also been used as a means to interact with miniature devices [15,20,26]. While such work has relevance to Medusa, the focus of the present work is on the augmentation of traditional 2D touch device interactions with additional sensing channels, rather than on the development of replacement 3D gestures. Significant work, which relies on detecting the hover of a stylus, has also been done. Vogel provides a thorough review of this research [30], which we omit due to the reliance on a tracked object.

Adding a Tracking State for Touch

In modern user interface parlance, *hover* has come to describe the tracking state of an input device, whether or not that state is achieved by physically hovering over anything [7]. Multiple projects have introduced a *tracking* state to a touch device without actually detecting physical hover. These include differentiation of posture of the touching finger [4,10,11] through differentiation based on the number or choice of fingers [21], through the use of physical proxies [32], through sensing the height of the hand above the device while touching with a finger [24], and through EMG-sensed differentiation of posture [3]. While relevant to the current work, the ability to detect physical hover can provide a broader range of capabilities than the straightforward addition of a tracking state.

Physical Hover Detection

The detection of physical hover has been achieved through vision techniques, using either above-device cameras (as with Smart's DVIT device, (smart.com/dvit), and LucidTouch [33]), or beneath-surface cameras [16, 27]. Sensing of induced capacitance has also been used to detect hands above a device [24]. While these approaches can extend the capabilities of touch devices by supporting in-air gestures [16], the physical mounting locations limit the interaction volume. To address this, Wilson and Benko proposed the use of multiple depth-cameras in a LightSpace [35]. As with the earlier approaches, the interaction volume was limited to the viewing area of the cameras. By mounting three rings of proximity sensors directly onto the table, Medusa ensures that the table itself does not occlude the sensors' view of the surrounding area.

Responding to User Location

Medusa is an interactive tabletop able to detect, track, and differentiate between multiple users. Previously, the DiamondSpin project demonstrated the use of user differentiation to augment interaction [25] by relying on capacitive coupling to provide user-differentiation for each contact [9]. While robust for the differentiation of users, user location is based on an assumption of static positions of receiver pads, and requires the user to remain in contact with their pad while interacting with the device. Vision techniques applied to touch input have demonstrated the ability to distinguish between contacts made with the right or left hand [8], but have not provided a mechanism to label hands from multiple users, cannot sense users before they touch the display, nor sense orientation information for each contact.

Multiple projects have demonstrated promising interaction techniques given user tracking, such as in public ambient displays [29], and proxemic interactions [2]. Both of these projected relied on the use of commercial motion trackers to observe fiducial markers worn by the user. Multitoe provided user tracking, but relied on an instrumented floor [1]. Medusa builds on the interactions provided by these systems, while contributing a commercially viable method for achieving user tracking. Also relevant to Medusa is the Range project, which explored the use of proximity sensing of implicit interactions [18]. While our focus is the enabling of explicit interactions, this work has informed some of our designs.

Sensing Technologies

Three rings of proximity sensors provide Medusa's additional information channels. Earlier projects have used these devices to enhance sensing. In Hoverflow, an array of proximity sensors enables the detection of 'coarse gestures' above a worn device [19]. Similarly, SideSight used two arrays of comparable sensors, pointed outwards from the sides of a mobile phone, to detect gestures made on the surface on which the phone was resting [6].

Furthermore, Tanase et al. used an array of 12 proximity sensors to coarsely detect the presence of users around a tabletop [28]. Unlike Tanase et al.'s implementation, Medusa has a much higher spatial resolution, is able to sense user presence around and above a tabletop, and fully explores the design space surrounding proximity-aware multitouch tabletops.

Summary

While there have been a large number of projects that have informed the design of Medusa, little to no work has previously examined how the information about users around a device can be used in tandem with information about gestures above and touching the device. Further, no device has previously been able to sense the various user parameters without additional external sensors. Finally, the previous explorations have focused on explicit gestures that engage the external sensors, rather than using the sensors to implicitly enhance existing surface-constrained interactions.

HARDWARE IMPLEMENTATION

Medusa is an augmented Microsoft Surface multi-touch tabletop, elevated to a height of 86 cm. To sense a user's body, arm, and hands, 138 low-cost, IR-based proximity

sensors are affixed to the top and side panels of the Surface. A combination of Sharp 2Y0A21 long-range (10-80 cm) and 2D120X short-range (4-30 cm) sensors are used.

The proximity sensors are arranged in three rings: an outward ring, an outer ring, and an inner ring (Figure 2). Although the sensors in our prototype protrude from the Surface, in the future they could be embed directly into the bezel or base of any multi-touch tabletop or display. The sensors were connected to the Microsoft Surface via 18 Phidget Interface 8/8/8 Kits. No additional data acquisition devices or hardware upgrades were added to the Surface.

The outward facing ring is composed of 34 long-range sensors, spaced 3.3 cm apart, and mounted at the top of each side of the table. Because the sensors point outwards, they create a horizontal sensing plane that projects 80 cm from the side panels of the Surface, along all sides. When a user walks up to, or near the Surface, their legs engage the sensing plane. Forty-six long-range sensors, spaced 3.3 cm apart and pointed upwards, make up the outer ring of sensors, creating a vertical sensing plane wrapped around the perimeter of the tabletop. Long-range sensors were used to allow users of various heights to interact with the system. Fifty-eight short-range sensors, spaced 0.8 cm apart, are located around the perimeter of the Surface's touch area. Similar to the outer ring, these sensors point upwards to form an inner vertical sensing plane. Shorter-range sensors were used to provide greater sensing resolution. When a user reaches towards the touch area of the tabletop, their arm first engages the outer and then inner sensing planes.

The three rings of sensors allow Medusa to track users, arms and hands, and to identify which *user* and which *hand* generated each touch event detected by the Surface.

SENSING

Multiple proximity sensors have been used in combination before [6,28,31], but to our knowledge, not to the extent of our 138-sensor implementation. As such, special attention is paid to the aggregation, processing, and filtering of the data. Furthermore, reconstructing body, arm, and hand positions is non-trivial. In particular, the sensing resolution is magnitudes lower than what a 3D depth camera provides [35]. Here, we describe the techniques we used to translate the simple proximity data from of our three sensor rings into the 3D location of users, arms, and hands.

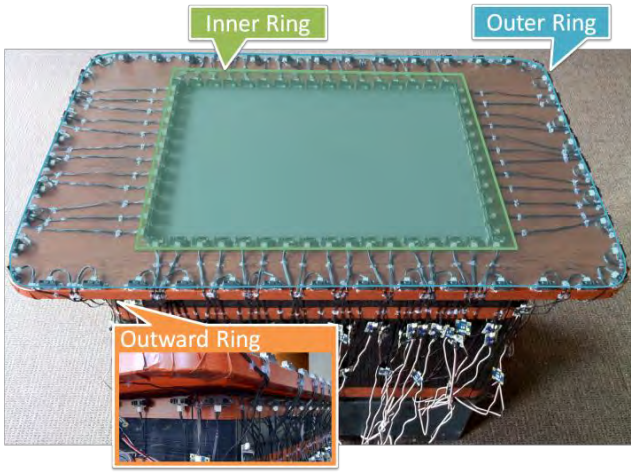

Figure 2: Medusa's sensors are arranged in three rings. An outward-facing ring of 34 sensors is mounted beneath the lip. Two upward facing rings atop the table are made-up of 46 sensors (on the outer) and 58 sensors (on the inner).

Using C# and the Phidget SDK, the signals from each sensor are sampled at 63 Hz and filtered using a median filter with a window size of 17 (for the outward-facing sensors) or 11 (for the upward facing sensors) to remove noise. Different window sizes were used because it is important to have a steady body position (rather than a responsive, more error-prone position), while responsiveness is required for position of the arms. To remove any noise that results from sporadically firing sensors, a uniform-weighted low pass filter with a window size of eight is applied to the result of the median filter.

To speed up processing and prevent the misidentification of arms in touch events, Medusa processes the sensors using a cascading logic approach. On each frame, the sensors in the outward ring are processed. If engagement with sensors in the outward-facing ring is detected, the sensors located in the outer upper-facing ring along the same side of the table are processed. If there is then a disruption in the outer ring, only those sensors in the inner ring, which are located along the same side, are processed. Using this approach prevents unnecessary processing while also preventing users from interfering with the sensing of other users (e.g., reaching over to their side, reaching across them, etc.).

Body Tracking Algorithm

Medusa uses the sensors' known physical locations and reported depth values to determine each user's body position around the Surface. Medusa first looks at all of the sensors in the outward ring to determine which ones are firing. If groups of consecutive sensors that are firing, they are placed into a 'sensor chain'.

Once all firing sensors have been identified, each sensor chain is analyzed. Medusa first discards all chains with a length less than 2 and then computes a Gaussian weighted average of each chain's sensor position and proximity values. This weighted value is used as an estimate of the user's body position (Figure 3). The distances between each body found in the current frame is compared to each previously computed body position. Bodies are mapped to the closest positions in the previous frames. If a new body is more than 50 cm from any previous body position, it is marked as a new body. Body positions are smoothed using a uniform low pass filter with a window size of 13. If a previously detected body has not been updated for 500 ms then that body is no longer considered present. Using this technique, we can robustly track two people along each long side of the table and one person along each short side of the table.

Arm Tracking

Similar to the body tracking method, Medusa looks at the outer and inner ring of sensors to find chains of sensors that

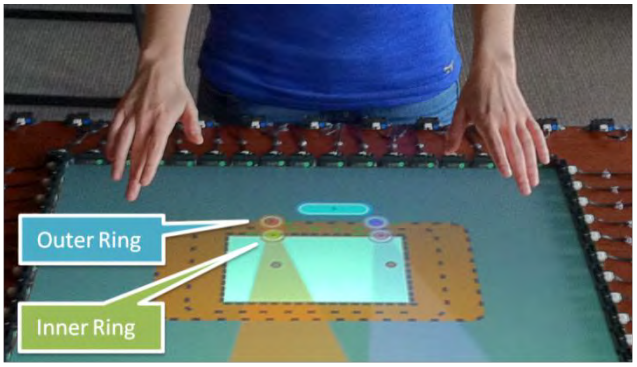

Figure 3: Visualization of tracking output. The location of the user's body is represented by the blue paddle. Right and left arm locations (with respect to the outer ring) are represented by orange and purple circles, respectively. The locations of the user's arms within the inner ring are represented by yellow and pink circles, respectively. The orange and purple cones represent the 'arm projections' of the user.

are firing. Unlike the body tracking, it is possible to disrupt only one sensor with the arm, so all chains are considered.

To find if an arm is present above the outer ring, a weighted average of the positions of each sensor in the chain is computed. Arms are matched across frames with the closest previously detected arm. If the current arm position is more than 20 cm from any previous arm, it is marked as a new arm, and right/left is determined. Arm position is smoothed using a low pass filter with a window size of 13. If an arm position has not been matched or updated for 500 ms, we then consider it as being removed.

To label an arm as right or left, each arm is compared to all body positions that are on the same side of the Surface, and is assigned to the closest one. If the arm is located to the right of the assigned body, it is classified as the right arm, else, the left. The side of the body an arm belongs to is always assigned when the outer ring of sensors is engaged.

The same heuristics are used to track arms above the inner ring of sensors: chains of disrupting sensors are found, and a weighted average of the sensor position is computed and matched against previous arm positions. When a new arm is identified above the inner ring, it is associated with the closest arm position currently sensed above the outer ring, and inherits the right/left arm classification.

When an arm engages both rings of sensors, the reported depth values, along with their known positions, can be used to provide a rough estimate of the horizontal and vertical arm angles (Figure 3).

Mapping Touch Points to Users and Hands

To map the touch points detected by the Surface to a specific user and hand, Medusa considers the orientation of the arms that are currently being tracked. For each arm that is engaging both the inner and outer rings, Medusa computes an 'arm projection', which is the vector that extends from the 3D line segment between the arm's estimated positions above the inner and outer rings. When a new touch point is sensed, Medusa determines the minimum distance that exists between the touch point and each of the arm projections. If the minimum distance to the closest projection is less than 15 cm, the touch point is matched to that arm and user. If no projection passes within that threshold, Medusa determines which arm is engaging the inner ring closest to the touch point. If two or more arms are within 20 cm of the touch point (20 cm), Medusa matches the touch point based on the minimum distance between the touch point and the body positions of the users, and associates the touch point with the arm of the user to which it is closest.

Tracking Accuracy

While we have not formally evaluated the tracking accuracy or robustness of Medusa, our initial experiences are quite encouraging, although there are some problem cases where contact points or arms are incorrectly tagged. In our discussion section, we outline some of the limitations of both our tracking heuristics and of the sensors themselves. Tracking is certainly sufficiently robust to enable our exploration and implementation of the new interaction techniques that are unique to Medusa's enhanced sensing capabilities. This exploration takes place within the context of Proxi-Sketch, our UI layout application.

PROXI-SKETCH

Application Scope and Motivation

To explore the new types of interactions that our additional sensing affords, Proxi-Sketch was created. Proxi-Sketch is an application that enables users to create and edit prototypes of graphical user interfaces (Figure 4), inspired by desktop applications such as Balsamiq (Balsamiq.com). This seemed like a reasonable application domain that would provide us with a sandbox for exploring numerous functions and features. Our contribution is not focused on the application domain itself, but rather how Medusa's sensing information can improve multi-user, multi-touch tabletop applications in general.

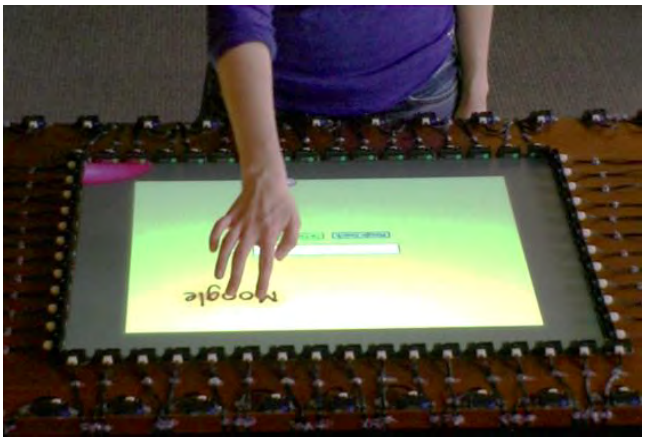

Figure 4. A user building a user interface prototype using Proxi-Sketch, running on Medusa.

While Medusa has the ability to sense explicit freehand 3D gestures, we focus instead on how implicit engagement of the sensors can be used to augment traditional 2D interac-

tions. We organize our discussion of Proxi-Sketch based on the properties sensed by Medusa:

- User Position Tracking
- Bimanual Input Distinction
- Pre-Touch Functionality
- Touch + Depth Gestures
- User Differentiation

User Position Tracking

An important aspect of Medusa is that it can detect when users are present, how far away they are from the table , and where around the table they are situated. In this section, we discuss how we leverage this information.

User Presence and Representation

When no users are present, an attract application is displayed. As with any input device that is proximity- or context-aware, it is important to convey to users that the system recognizes their presence and behaviour [29]. We indicate the identification of a user's presence by displaying a persistent visual representation of the user via a glowing orb (Figure 5). When a user first enters the *idle* state (by walking up to the tabletop), they will see a blurry, blue glowing orb (which represents an unknown user). The location of the orb moves with the user as they walk toward, around, and away from the tabletop. If a user chooses to move closer to the tabletop, their orb comes into focus and invites the user to login. If the user exits the *idle* state (by walking beyond sensing range), the orb disappears.

User Login/Logout

Users login to the system by touching the orb, rotating through a carousel of user ID's, and selecting their profile picture (Figure 5). Once logged-in, the blue glowing orb changes to a user-specific colour, displays the user's name, and minimizes to the corner of the screen, allowing immediate visual identification by all users without being intrusive or distracting. If the user moves to another side of the tabletop, the orb moves with them and becomes anchored in a corner of the new side. If the user walks away from the table at any time, the orb changes to gray and then gradually fade out, visualizing a timeout until the user is logged-out.

Once a user has logged-in, the orb acts as a personal gesture dashboard for context-specific gestures. One of the subtle benefits of Medusa is that after logging-in, a user's gesture dashboard remains associated with them, even as they walk around the table. There are five gestures that are supported by the dashboard: *logout, open file, save file, group*, and *do not disturb*. The logout gesture is a 3 finger tap, and allows users to manually logout, instead of automatically logging out by walking away. Opening and saving files is accomplished with a swipe of a finger to the left and right, respectively. Saved content is associated with the user's profile, and the open gesture evokes a carousel of documents associated with their account. The remaining gestures are discussed in the sections that follow.

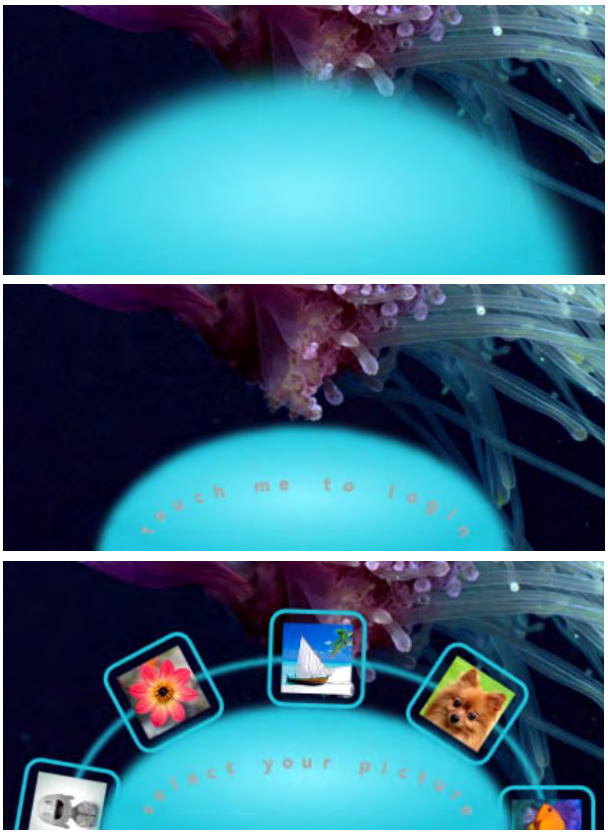

Figure 5: Initially, each user is represented by a blurry, glowing blue orb (Top). After moving closer to the tabletop, the orb comes into focus and encourages the user to login (Center). If a user taps on their orb, it will bring up a rotating carousel of available user accounts (Bottom).

Different Sides / Different Functions

Since Medusa provides knowledge pertaining to a users' location, Proxi-Sketch can assign specific modes, tools, or functionalities to different sides of the tabletop. To explore this possibility, we assigned different sides of the tabletop to different fidelities of the current prototype. By default, prototypes are rendered in a 'sketchy' style. If the user wishes to see their prototype from a fresh point of view, in a higher fidelity, they can walk over to an adjacent side of the tabletop (Figure 6). Not only will the visual style of the interface change, but also the canvas appropriately reorients to face the user's new location. When a user moves to another adjacent side of the tabletop, the interface switches back to the 'sketchy' style and reorient to face the user.

Standing at the Corner

When a user stands at one of the corners of the tabletop, the presence of the user's body is identified but not their arms, arm movements, or touch points. The corners of a tabletop are perfect locations for users who have restricted permissions but are allowed to observe the collaboration process. Within Proxi-Sketch, blue orbs appear for users who are standing at the corners of the Surface to indicate that they are recognized, but are not given the opportunity to login or manipulate any content.

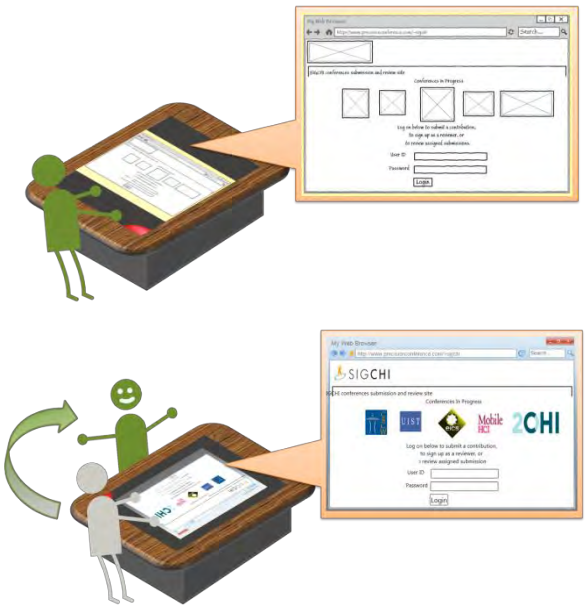

Figure 6: By default, a low-fidelity rendering of each UI component is generated (Top). Once a user walks to an adjacent side of the table, a high fidelity rendering of each component becomes visible (Bottom).

Bimanual Input Distinction

While bimanual input is commonly used in tabletop applications, Medusa is able to distinguish between the user's left and right hands. This distinction is particularly important, as interactions should support a non-symmetric division of labour [14]. This section describes the interactions we explored to leverage this information.

Content Creation and Removal

We use the notion of dominant and non-dominant hands to assign appropriate functionality for adding and removing content. Content is added by tapping on the canvas with the dominant hand. A marker coloured with the identified user's orb colour appears on the canvas. When touched, a hierarchal radial menu opens, oriented towards the user. This menu adds UI components, such as browser windows, media players, and scroll bars to the canvas.

If a user wishes to clear the canvas, she user can touch the canvas with their non-dominant hand, displaying a 'clear canvas' icon, oriented to that user. If the clear icon is touched, all user interface components are removed from the canvas. Assigning removal to the non-dominant hand reduces the risk of users accidentally removing content.

Translation, Rotation, Scaling

On a multi-touch tabletop, it can be difficult to perform controlled manipulations, because a single gesture typically activates rotation, scale, and translation all at the same time. Proxi-Sketch addresses this problem by taking advantage of hand identification. To translate a component, the user can touch the component with one finger and drag it. If a user wishes to rotate the component, they can touch the component with two fingers from the *same* hand. If a user wishes to scale a component, the user touches it with one finger from *each* hand. This differentiation allows for increased precision through transform isolation.

Manipulations with Groups of Content

Proxi-Sketch uses bimanual input distinction to quickly differentiate manipulation gestures as applying to the whole group, or to an individual item.

If a user wishes to manipulate a whole group of components, they can do so by touching any component in the group with their non-dominant hand. Conversely, if the user wishes to manipulate only a single component within a group, thus leaving the rest of the group in place, they touch the desired component with their dominant hand. All of the components belonging to the group become highlighted (to ensure that the user knows that the component is part of a group), but the desired component is the only one that will be manipulated. Our design rationale when making this bimanual separation is in keeping with research in bimanual interaction, which advocates assigning coarser actions (moving all the components in a whole) to the non-dominant hand [14].

Content is grouped by tapping one finger on the gesture dashboard, enabling 'group mode'. Once in this mode, any components touched by the user become highlighted with the user's orb colour. After all desired components are selected, the user can again tap the dashboard with one finger to create the group and exit the mode.

Pre-Touch Functionality

In addition to sensing the proxemic state of users, Medusa can also sense the proxemic states of a user's hands. In particular, Medusa knows if a hand has crossed the bezel or display area, but not come into contact with the display area. This allows us to explore *pre-touch* functionality, discussed below.

Global Gesture Guide

Similar to many other multi-touch tabletop applications, there are a number of gestures that can be performed within Proxi-Sketch. Previous efforts have used on-demand guides to illustrate available gestures [12]. An open problem has been how to display this guide without having to reserve a "help" gesture. Medusa provides an obvious mechanism: guides are displayed when the user's hand hovers over the table, demonstrating hesitation. A global gesture guide (Figure 7) is shown when the user's hand hovers over the bezel of the table. We chose this interaction as it implicitly indicates that the user intends to interact with the system, but may be unsure how to proceed. The guide remains anchored near the user, in the center of the screen, until the user removes their arm or reaches towards the touch area.

Dashboard Gesture Guide

Similar to the global gesture guide, each user has an on-demand, gesture dashboard guide available (Figure 8). The guide becomes visible when the user's hand hovers above their orb, and disappears whenever the user touches their orb or moves their hand away. This extends the notion of traditional gesture guides, allowing users to view context-specific gestures.

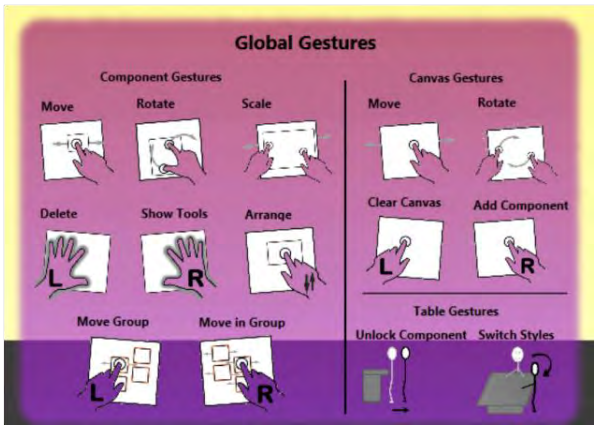

Figure 7: Global Gesture Guide. This guide appears whenever a user's hand dwells over the outer ring of sensors. It assists users in remembering which gestures are permissible in Proxi-Sketch.

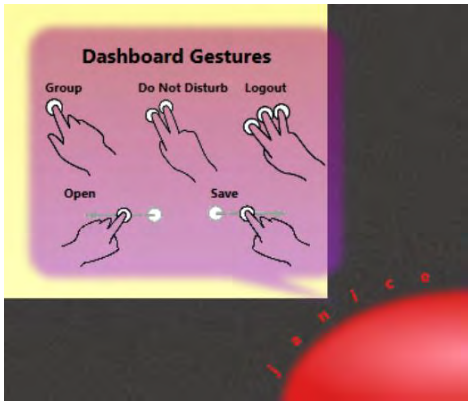

Figure 8: Dashboard Gesture Guide. This guide illustrates gestures that can be performed on an orb. It appears when a user's hand dwells over their orb.

Just-in-Time Widgets

Pre-touch functionality can also be combined with bimanual input distinction. Similar to the addition and removal of canvas content, the actions required to delete or edit a specific user interface components have been divided bimanually. If users wish to edit the colour, text, or other properties of a specific component, they can hover their dominant hand above it, thus causing a coloured marker to appear (Figure 9). This type of "just-in-time" widget is made possible by our 3D arm location sensing. Once touched, a radial marking menu displays available options (Figure 9).

The marking menu is located such that it is not occluded by the user's arm, is oriented towards the user, and its options are located in a convenient position for the dominant hand to access. The selection of an element in this marking menu causes a dialog to appear, allowing the user to make their desired modifications.

If a user wishes to remove a component from the canvas, they hover their non-dominant hand above the desired component. This results in a red 'X' icon being placed on the component's left side (Figure 10). Touching the 'X' icon deletes the component.

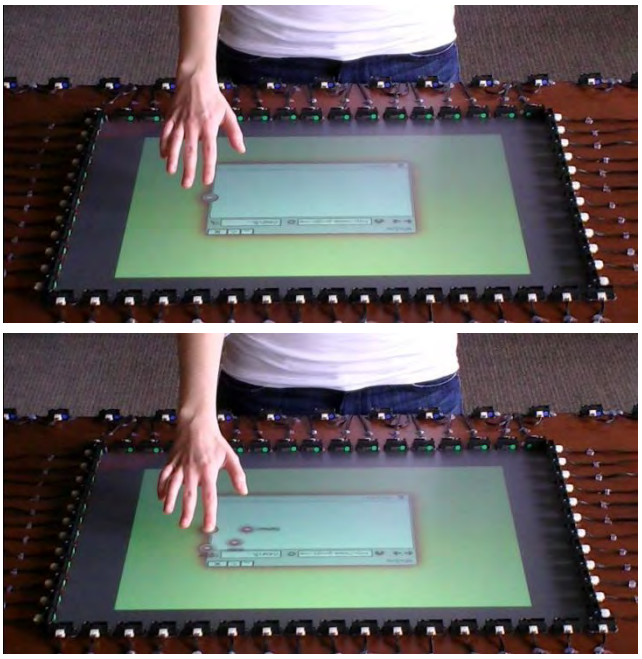

Figure 9: When the right arm is moved over a component, a marker appears below the hand. (Top). Touching this marker displays a component-specific marking menu (Bottom).

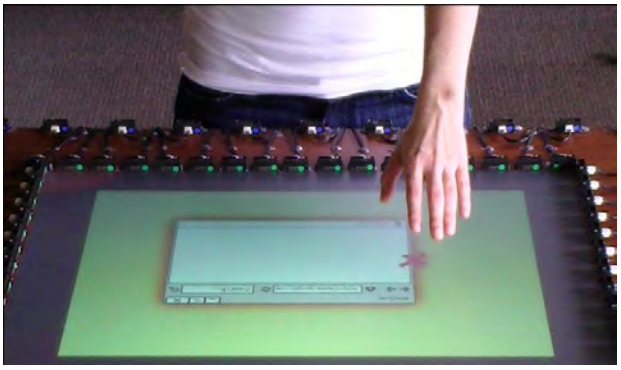

Figure 10: Moving one's left arm over a component brings a red 'X' icon into view. Touching this 'X' icon deletes the component from the canvas.

These just-in-time widgets allow users to preview possible actions with specific application components, and keep the working area clear of user interface clutter.

Touch + Depth Gestures

An important feature when producing UI prototypes is the ability to change the z-order of components of the UI. We developed a new type of gesture that Medusa enables, which *combines touch and 3D sensing* to control item arrangement. To send a component back in *z*, behind all other components, the user touches the component and moves their arm/elbow towards the Surface. To bring a component to the front, the user can touch the component and pick it up (i.e., move their arm upwards, away from the Surface). This new method of arranging components enables users to employ a direct manipulation metaphor to perform a task that is often obfuscated by icons or hidden menus.

User Differentiation

Issues in sharing and territoriality in tabletop collaboration have long been studied [22]. As Medusa provides information regarding user location and touch identity, new elements of facilitated collaboration are possible.

Content Control

In multi-user scenarios, control of content on a multi-touch tabletop has been a long-standing problem, especially when a system is uncertain who is interacting with content [22].

If a user walks up to the tabletop and does not login, all of their interactions with the tabletop are automatically blocked. This is only possible because Medusa associates touch points to users. This allows casual observers to point to content, without accidently changing it.

When multiple users are logged in, Proxi-Sketch makes use of its user-identification to assign ownership of components to users. To prevent users from accessing content from another user or interfering with someone else's work, Proxi-Sketch employs a 'one component per user, one user per component' rule. With this rule, each user has control over one component at a time, and at most one user can own each component at a time. Each 'owned' component is highlighted with its owner's colour and cannot be manipulated, edited, or deleted by anyone but the owner.

To take control of a component, a user simply touches it. As soon as a user begins working with a new component, they automatically give up control of their previously owned component. If a user wishes to give up control of a component, they can take a small step away from the Surface. The user's orb colour changes to grey and the component is free for other users to manipulate.

This approach to content control does not rely on explicit interaction with the tabletop, but rather uses a subtle and intuitive interaction to manage ownership. While devices such as the DiamondTouch could enforce a similar ownership scheme, the identity of the owners would not be robust to movement around the table, nor could a step back from the table be sensed to cede control [9].

Do Not Disturb

Brignull and Rogers identified the *Honeypot Effect* of digital surfaces, which has been amply confirmed by the authors' own experiences. It is quite common when working with a multi-touch tabletop to be interrupted frequently, or bothered by strangers or coworkers who are curious about what the surface is, want to know what is being worked on, and even start touching the surface without invitation [5]. As Medusa is able to identify when users are approaching, would-be users can now be *discouraged* from interacting with the Surface by enacting a new table mode: *Do Not Disturb* (DND), shown in Figure 11.

DND provides users who are currently interacting with the system with a method of discouraging others from approaching, without having to personally acknowledge them or be disrupted. A user working with Proxi-Sketch can evoke DND mode by tapping two fingers on their orb. Once in this mode, any potential user who walks near the tabletop is presented with a bright red 'prohibited' glowing orb (Figure 11). If seen from a distance, this red orb subtly but kindly informs any passersby or potential users that the individuals currently using the tabletop do not wish to be disturbed. If an uninvited user moves closer to the Surface, their orb will not shrink, nor will it let them login or touch any part of the canvas. Once users are ready to allow others to collaborate, they can tap two fingers on their orb to return the tabletop back to its normal state.

Figure 11: If the tabletop has been placed in the 'Do Not Disturb' mode, all logged out and potential users will be greeted with a 'prohibited' glowing red orb, encouraging them to walk away from the table and not bother those currently engaging with it.

Summary

Taken as a whole, Proxi-Sketch builds atop the sensing capabilities included in Medusa to provide a set of unique interactions. User position sensing is used to encourage (or discourage) use of the table, and to provide visual feedback of detected presence. Bimanual input distinction has been used to reduce the likelihood and cost of errors, and to provide improved precision for direct manipulation. Pre-touch functionality provides a new state for interaction, which we have used to explore a solution to gesture learning and to enable just-in-time widgets. Finally, we have demonstrated the utility of user and touch identification to enhance collaboration, and to provide mechanisms, both implicit and explicit, to manage content control and privacy. This has been accomplished while maintaining a viewpoint that Medusa's additional sensing is meant to *enhance* the touch sensing of the underlying Microsoft Surface, rather than to supplant or replace touch-based interaction.

DISCUSSION

We have presented Medusa, a novel system that uses proximity sensors to capture a rich set of user-proximity information, and Proxi-Sketch. In this section, we discuss other sensing solutions, the limitations of proximity sensors, and the generalizability of our interaction techniques.

Sensing: Proximity Sensors and Depth Cameras

The use of proximity sensors has a number of benefits. As proximity sensors are small, it would be plausible to integrate them into the bezels of touch displays, similar to their current usage in mobile phones. The sensors are also inex-

pensive and do not require any calibration. Most importantly, no external cameras or tracking markers are required, making Medusa a wholly integrated hardware solution.

The data we receive is of a much lower fidelity than what could be obtained from other sensing possibilities. For example, using a Microsoft Kinect depth camera could potentially provide full skeletal tracking, allowing a richer set of body-based gestures and techniques to be explored. However, it is unlikely that a single *integrated* Kinect device could replace the aggregate effect of our distributed sensing solution. Most likely, a Kinect device would need to be mounted externally, such as on the ceiling [35]. Associated technical challenges would thus include occlusions by users, possible interference between the IR emitters and receivers in the depth cameras and Surface devices, and the need to recalibrate each time the Surface is moved. Our solution could be embedded into existing hardware, overcomes issues of occlusion, and requires no calibration.

Another option would be the use of IR cameras mounted below the surface of the display, as demonstrated in SecondLight [16,17]. This however would only provide a restricted field of view, in comparison to the full proximity information our implementation enabled. Medusa is able to sense the location of users around the Surface, identify if the right or left hand is present, and detect which user each arm/hand belongs to. This information is also used to match touch points to specific users. Behind surface cameras may have knowledge of the position, orientation, and depth of the hand, but none of the additional user-based information is available for use. It is also important to note that our approach can be used to retrofit all existing tabletop systems (e.g., capacitive, resistive, FTIR, diffuse illumination, etc.), whereas many current and future multi-touch tabletops will not have a form factor that could support a beneath-surface camera and switching diffuser.

Limitations of Proximity Sensors

There are some limitations of the sensors that should be highlighted. To our surprise, there was little issue with sensors interfering with one another. However, we initially found that sensors would frequently report false positives, (ghost objects), due to IR reflections from reflective material in our lab (e.g. bicycles, exposed ceiling pipes). Once diagnosed, the problem was addressed using strategically placed mirrors, to ensure light was not reflected back towards the sensors. More advanced filtering may reduce or alleviate this issue.

We demonstrated that with some simple tracking algorithms, our arrangement of proximity sensors could track users, arms, hand locations, arm orientations, and could associated arm and touch points to users, as well as identify hands. Certain tracking information was accurate (such as detecting where users arms engaged the sensing planes), whereas other information was more coarse (such as estimating the 3-D location of a user's hand when hovering above the input area). While the tracking performance was robust enough to explore the interaction design space, there were some limitations. For example, if two users cross paths, our body tracking algorithm could mislabel users once they separate, or if two users' hands hover over the exact same area, and only one hand touches the Surface, the touch point may be associated with the wrong user.

Generalization of Interaction Techniques

While Medusa's hardware arrangement and tracking algorithms constitute a contribution of this work, many of the interaction techniques presented would also be suitable for devices built using other sensing modalities. In particular, a technical implementation using Kinect devices would be able to utilize all of the interaction techniques we have explored. An implementation using below-surface cameras would be able to utilize only those techniques that do not require knowledge of users or their locations – such as the hover-based, just-in-time widgets.

FUTURE WORK

Our work opens up new opportunities for future research, with respect to the tracking and the interaction designs.

The tracking algorithms we used were simple, but sufficient in most cases. Future work could formally evaluate the accuracy of our baseline sensing solution and potentially introduce more advanced techniques to match arms and touch points to users. For example, when matching touch points to arms, one could consider the timing correspondence between when the arm entered the touch area, and then the occurrence of the touch. Future work could also look at extending the tracking algorithms to support tabletops with different physical properties (i.e., height, number of sides, length of sides, etc.). More attention could also be given to potential sensing inaccuracies in the interaction design. Important or frequent system features should be mapped to gestures that have the highest accuracy rates, and error recovery techniques should be provided when misclassifications occur.

With respect to our design exploration, there are a number of interesting ideas that could be explored further. We introduced some new gestures enabled by Medusa, but there are many more possibilities. An implementation using a Kinect device may support new types of gestures and techniques that we did not explore. For example, full-skeletal tracking could support the detection of body postures for application control, such as leaning in and out to control zoom levels, or showing private information when other users turn their backs on the table.

New types of widgets could be explored using the sensing information. For example, widgets could become fatigue-aware, as the system could measure the amount of reaching a user was performing. Like previous explorations with the pen, widgets could also become occlusion-aware, based on the position of the user's hand and angle of their arm [30].

Finally, there is a clear need to explore affordances and feedback to help users understand the parameters being sensed, and how to manipulate them, similar to earlier explorations with direct touch input [34].

REFERENCES

1. Augsten, T., Kaefer, K., Meusel, R., Fetzer, C., Kanitz, D., Stoff, T., Becker, T., Holz, C., and Baudisch, P. Multitoe: high-precision interaction with back-projected floors based on high-resolution multi-touch input. *UIST '10*, 209-218.
2. Ballendat, T., Marquardt, N., and Greenberg, S. Proxemic interaction: Designing for a proximity and orientation-aware environment. *ITS '10*, 121-130.
3. Benko, H., Saponas, T.S., Morris, D., and Tan, D. Enhancing input on and above the interactive surface with muscle sensing. *ITS '09*, 93-100.
4. Benko, H., Wilson, A.D., and Baudisch, P. Precise selection techniques for multi-touch screens. *CHI* '06, 1263-1272.
5. Brignull, H. and Rogers, Y. Enticing people to interact with large public displays in public spaces. *INTERACT '03*, 17-24.
6. Butler, A., Izadi, S., and Hodges, S. SideSight: multi-touch interaction around small devices. *UIST '08*, 201-204.
7. Buxton, W. A three-state model of graphical input. *IFIP TC13 '90*, 449-456.
8. Dang, C.T., Straud, M., and Andre, E. Hand distinction for multi-touch tabletop interaction. *ITS '*09, 101-108.
9. Dietz, P. and Leigh, D. DiamondTouch: a multi-user touch technology. *UIST '01*, 219-226.
10. Esenther, A. and Ryall, K. Fluid DTMouse: better mouse support for touch-based interactions. *AVI '06*, 112-115.
11. Forlines, C. and Shen, C. DTLens: Multi-user tabletop spatial data exploration. *UIST '05*, 119-122.
12. Freeman, D., Benko, H., Morris, M.R., and Wigdor, D. Shadow Guides: Visualizations for In-Situ Learning of Multi-Touch and Whole-Hand Gestures. *ITS '09*, 165-172.
13. Grossman, T. and Wigdor, D. Going Deeper: A Taxonomy of 3-D on the Tabletop. *ITS '07*, 137-144.
14. Guiard, Y. Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model. *Journal of Motor Behavior*, 1987, 9(4), pp. 486-517.
15. Harrison, C. and Hudson, S.E. Abracadabra: Wireless, High-Precision, and Unpowered Finger Input for Very Small Mobile Devices. *UIST '09*, 121-124.
16. Hilliges, O., Izadi, S., Wilson, A.D., Hodges, S., Garcia-Mendoza, A., and Butz, A. Interactions in the air: adding further depth to interactive tabletops. *UIST '09*, 139-148.
17. Izadi, S., Hodges, S., Taylor, S., Rosenfeld, D., Villar, N., Butler, A., and Westhues, J. Going Beyond the Display: A Surface Technology with an Electronically Switchable Diffuser. *UIST '08*, 268-278.
18. Ju, W., Lee, B.A., and Klemmer, S.R. Range: exploring implicit interaction through electronic whiteboard design. *CSCW '08*, 17-26.
19. Kratz, S. and Rohs, M. HoverFlow: expanding the design space of around-device interaction. *MobileHCI '09,* 1-8.
20. Loclair, C., Gustafson, S., and Baudisch, P. PinchWatch: A Wearable Device for One-Handed Microinteractions. *MobileHCI '10*.
21. Matejka, J., Grossman, T., Lo, J., and Fitzmaurice, G. The design and evaluation of multi-finger mouse emulation techniques. *CHI '09*, 1073-1082.
22. Mueller-Tomfelde, C. Tabletops - Horizontal Interactive Displays. Springer. ISBN: 978-1-84996-112-7.
23. Pierce, J. *Expanding the Interaction Lexicon for 3D Graphics.* Ph.D. Thesis, Carnegie Mellon University, 2001.
24. Rekimoto, J. SmartSkin: An Infrastructure for Freehand Manipulation on Interactive Surfaces. *CHI '02*, 113-120.
25. Shen, C., Vernier, F.D., Forliens, C., and Ringel M. DiamondSpin: an extensible toolkit for around-the-table interaction. *CHI '04*, 167-174.
26. Starner, T., Auzier, J., Ashbrook, D., and Gandy, M. The gesture pendant: A self-illuminating, wearable, infrared computer vision system for home automation control and medical monitoring. *ISWC '00*, 87-94.
27. Takeoka, Y., Miyaki, T., and Rekimoto, J. Z-touch: an infrastructure for 3D gesture interaction in the proximity of tabletop surfaces. *UIST '10*, 91-94.
28. Tanase, C.A, Vatavu, R.D., Pentiuc, S.G., and Graur, A. Detecting and Tracking Multiple Users in the Proximity of Interactive Tabletops. *Advances in Electrical and Computer Engineering*, 2010, 61-64.
29. Vogel, D. and Balakrishnan, R. Interactive public ambient displays: transitioning from implicit to explicit, public to personal, interaction with multiple users. *UIST '04*, 137-146.
30. Vogel, D. *Direct Pen Input and Hand Occlusion*. PhD. Thesis, University of Toronto, 2010.
31. Walther-Franks, B. Schwarten, L., Teichert, J., Krause, M., and Herrlich, M. User detection for a multi-touch table via proximity sensors. *ITS '08 Posters.*
32. Wang, F., Cao, X., Ren, X., and Irani, P. Detecting and Leveraging Finger Orientation for Interaction with Direct-Touch Surfaces. *UIST '09*, 23-32.
33. Wigdor, D., Forlines, C., Baudisch, P., Barnwell, J., and Shen, C. LucidTouch: a see-through mobile device. *UIST '07*, 269-278.
34. Wigdor, D., Williams, S., Cronin, M., Levy, R., White, K., Mazeev, M., and Benko, H. Ripples: utilizing per-contact visualizations to improve user interaction with touch displays. *UIST '09*, 3-12.
35. Wilson, A.D. and Benko, H. Combining multiple depth cameras and projectors for interactions on, above and between surfaces. *UIST '10*, 273-282.
36. Wilson, A.D., Izadi, S., Hilliges, O., Garcia-Mendoza, A., and Kirk, D. Bringing physics to the surface. *UIST '08*, 67-76.