SketchSpace: Designing Interactive Behaviors with Passive Materials

David Holman
Human Media Lab
Queen's University
Kingston, ON
holman@cs.queensu.ca

Hrvoje Benko
Microsoft Research
One Microsoft Way
Redmond, Wa
benko@microsoft.com

Abstract
This paper presents *SketchSpace*, a system that allows designers to interactively sketch [3] device's interactive behaviors by imbuing digital functionality to passive materials. SketchSpace requires no augmentation of the device itself, but instead it uses a depth-sensing Kinect camera to simulate the use of hardware sensors by using depth information to infer an object's three-dimensional position, motion, proximity, shape, deformations, and touch events on its surface. A designer can map these inputs to desktop applications in real-time and thus experiment with different interactions. We showcase how SketchSpace can be used to prototype two devices: from tilt sensitive mice to bendable displays. In general, we discuss how this simplifies the process of generating an interactive device sketch and supports rapid exploration of design solutions.

ACM Classification Keywords
H.5.2 [Information interfaces and presentation]: User Interfaces. – Graphics User Interfaces

General Terms
Design, Human Factors

Keywords
Sketching user interfaces, rapid prototyping, depth sensing camera.

Introduction
During the design process, industrial prototypes are critical artifacts for innovation and collaboration. Of the hundreds of prototypes a designer sculpts, few will have interactive functionality. *Bread boarding,* as designers refer to it, integrates hardware components into a prototype to roughly approximate its final interactive behavior. Although this approach is common in practice, it is labor-intensive, impacts ergonomics, and requires technical knowledge of embedded hard devices.

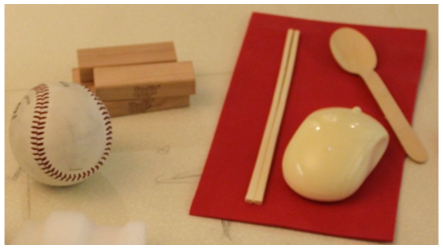


Figure 1. A range of materials a designer could work with using SketchSpace. This includes everyday objects (wood blocks, chopsticks, spoon, baseball), deformable surfaces (Mousepad), and plastic device prototypes (mouse).

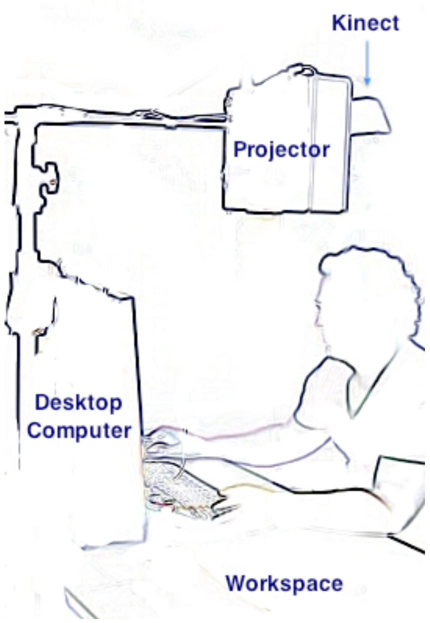


Figure 2. A Kinect camera hovers over the designer's workspace. Its depth data is used to the track the designer's manipulations of a physical prototype.

What if a designer could explore the interactive behavior of a device prior to having a working prototype? In this way, we envision an interactive sketching tool that is as seamless as working with common design materials, such as tape or foam core, to express a physical a design. Achieving this *interactively* helps a designer experience a device *hands-on* and, consequently, triggers the serendipitous realizations arrived at with a concrete artifact [7].

With this in mind, we designed *SketchSpace*, a lightweight environment that adds interactive behavior to passive physical objects (see Figure 1). A designer places an object in her workspace and works with a prototype as if it *actually had* physical sensors. All input sensing in SketchSpace is achieved using Microsoft's Kinect's camera (see Figure 2). The depth data allows SketchSpace to observe a broad set of inputs, ranging from touch events on the object's surface, orientation, position, motion, proximity, and, among others, shape deformations. Although these inputs are often recognized using hardware sensors, a designer does not have to find, attach, and work with physical sensors. Using the depth data to infer *virtual sensors* leaves the designer free to ideate, explore, and quickly make interactive sketches.

This activity of quickly and inexpensively generating numerous possible designs is referred to as *sketching* [3]. Buxton distinguishes it from prototyping and details how a sketch's intentional roughness evokes useful discussion and leads to better design solutions. Unlike other design disciplines that operate primarily in a spatial domain, *interactive* sketching introduces a temporal dimension that complicates representing even simple interactions. It is challenging —yet absolutely necessary —for an interactive sketching tool to quickly enumerate the design options from which to choose. To do this, Buxton argues such a tool should enable a designer to make quick, inexpensive, and disposable sketches. SketchSpace explores exactly this area of *sketching* interactive behaviors by using only physical materials and a depth camera.

Related Work

Previous work such as d.tools [5], VoodoIO [13], the Calder Toolkit [9], and BOXES [6] augment physical prototypes with hardware sensors and provide high-level toolkits that abstractly represent interaction to a designer. This promotes ease of use and drastically simplifies the work needed to rapidly prototype hardware interfaces. It does not, however, minimize the physical challenge of seamlessly integrating sensors into a prototype. The time needed to find, fasten, and fit numerous sensors is a barrier to quick and disposable sketching [3]. Although iStuff Mobile [2] addresses this problem using a single integrated sensor that senses acceleration, tilt, and rotation, it assumes a rigid mobile form factor that cannot sense touch. SketchSpace, instead, removes the need for most hardware sensors and uses the depth data to generate virtual ones. This, unlike previous examples, permits physical prototypes to be swapped without refitting sensors to a new device and further supports rapid design exploration.

DisplayObjects [1], unlike physical prototyping toolkits, relies on motion capture to rapidly prototype foam mockups that are rendered with projected content. Similar to SketchSpace, this minimizes the use of physical hardware sensors and requires only infrared tracking markers on the mockup. Unlike SketchSpace,

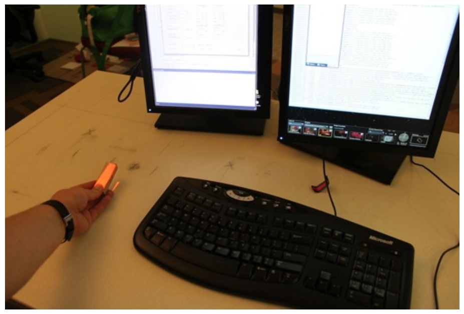
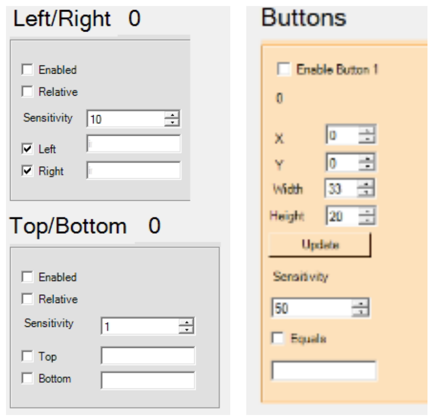


Figure 3. A red highlight indicates that SketchSpace is tracking the block of wood and its virtual sensors are active.



Left/Right 0

☐ Enabled
☐ Relative
Sensitivity 10
☑ Left
☑ Right

Top/Bottom 0

☐ Enabled
☐ Relative
Sensitivity 1
☐ Top
☐ Bottom

Buttons

☐ Enable Button 1
0
X 0
Y 0
Width 33
Height 20
Update
Sensitivity
50
☐ Equals

(a) (b)

Figure 4. A subset of the Mapping Panel.

(a) *The controls for the virtual tilt sensor*. The designer enables a sensor, sets it to a relative or absolute value, specifies its sensitivity, and designates the key presses to be raised when it is triggered. Similar controls for the other virtual sensors are available.

(b) *The control to add a virtual button*. The designer adjusts its position, size, and sensitivity. Up to two buttons are supported.

DisplayObjects's motion tracking algorithms requires the designer to build a virtual model of the mockup that is input to the tracking engine. After this step, the designer is free to explore aesthetic design by dragging projected content from a physical palette and manually placing it on the mockup. Although button presses are supported, complex input behaviors that express changes in a position or shape are not mapped to interactions.

Design Concept

SketchSpace is motivated by observations of industrial and interaction designers working at Microsoft Hardware group. We interviewed two designers as they worked through the design of novel computer input devices (e.g., mice, remotes, or game controllers). First, we observed the lead industrial designer as he sculpted different form explorations for a novel controller. We also interviewed the designer responsible for the interaction techniques this controller would exhibit and discussed the current practice of designing interactions.

Both designers confirmed that working with a functional device is increasingly important in their work. As the interaction designer stated: "design fails when certain parts aren't evaluated." The omission of a functioning input device increases the likelihood of failure. Although tools like Knörig's Fritzing [8] can make bread boarding easier, introducing a functional prototype into the designer's workflow can still be time consuming. If an idea is too difficult to be explored or cannot be represented, it will be avoided. Overall, the designers wanted a sketching tool that is lightweight, easy to use, and allows them to work together when iterating over an interactive design.

SketchSpace System

Guided by these observations, we designed SketchSpace to allow designers to more seamlessly sketch interactive devices. The system positions a depth sensing camera and projector above a designer's workspace (see Figure 2). The depth data is used to track an object and make inferences about its input behavior. The projector renders perspective correct feedback to indicate that a passive object has interactive behavior (see Figure 3). While manipulating an object, a designer monitors its input behavior on SketchSpace's Mapping Panel (see Figure 4). From these input values, the designer generates simple rules that trigger interactive actions on a Desktop application. For example, bending the corner of scrap paper can be quickly mapped to trigger a page back in a web browser as originally envisioned by Gallant et al. [4]. Using SketchSpace in this way, the designer iterates a device's interactive potential.

Virtual Sensors

SketchSpace is capable of reporting the changes in several object states in a form of a virtual sensor stream. Our selection of virtual sensors was motivated by the most common actions that our observed designers wanted to prototype when designing a new device. We have prototyped five virtual sensors that serve as the basic elements for building interactions in SketchSpace:

1. *Grasp*. When a designer picks up an object, SketchSpace understands the object is being grasped and how much of it is covered.

2. *Touch*. Designers can touch a single point on a flat or curved surface. We limit this gesture to

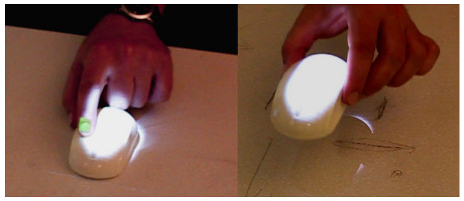


Figure 5. A designer adds a virtual button to the mouse prototype and then explores tilt-based interactions.

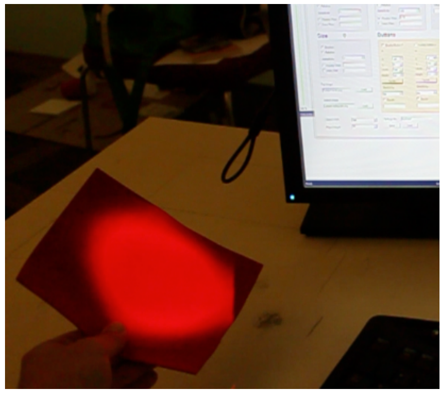


Figure 6. An everyday flexible material, such as a Mousepad, is used to quickly explore the combination of bending and tilting interactions. Note that the projected color hue can change with the tilt of the device.

single touch to simplify the complexity of distinguishing natural grasps versus multiple touch points.

3. *Position*. This sensor expresses the three-dimensional position of an object as a designer manipulates it. In general, it represents spatial information such as position in three dimensions, tilt around its horizontal and vertical axis, and rotation around its z-axis.

4. *Proximity*. This sensor expresses the spatial relationship between two objects and how near or far from each other. In Figure 6, or example, the distance between the designer's hand and flexible material is known. In general, this gesture enables new types of proximity-based interaction to be explored.

5. *Deformation*. A designer reforms an object's surface geometry dynamically. A piece of paper, for example, can be reshaped to express a convex or concave bend.

Mapping Gestures to Interface Actions

Each of these sensor streams must be mapped to specific actions in the designer's workflow. The Mapping Panel provides an overview of input behavior and, for each virtual sensor, shows its value, mapping rules, and the corresponding action the rule triggers (see Figure 4). We currently map all our actions to keyboard and mouse events, which make it possible to easily control most desktop applications. Like BOXES [6], the designer specifies the trigger condition and the key or mouse event that is sent to the active desktop application. Although specifying these standardized key encodings [10] introduces some overheard, it allows the designer to control any application on the fly without using any specialized programming interface.

Usage Scenario

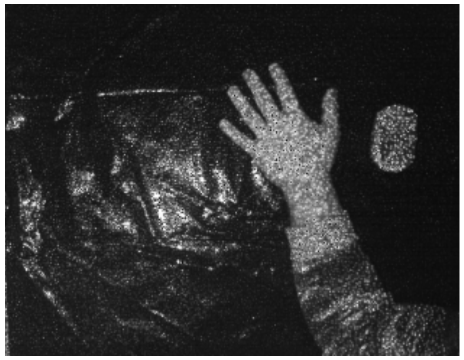
To illustrate how designers work with SketchSpace, we present the exploration of a tilt-based mouse device and an interactive sketch of a bendable display.

Sketching Mouse Interactions

With a non-functioning prototype of a new tilt-enabled mouse in hand, the lead industrial and interaction designers use SketchSpace to quickly try out new interactions. Using the Mapping Panel, the designers map its 2D position to the desktop cursor and place a virtual button on its surface (see Figure 5). To get a feel for the tilt interactions, they decide to map the mouse's virtual tilt sensor to horizontal panning in Microsoft's WorldWide Telescope. As they tilt the mouse, they monitor the sensor value in the Mapping Panel. Iteratively, they tweak the sensitivity to indicate when the mouse is sufficiently tilted. To pan left, the designers type "LEFT" in the mapping text field to indicate a left key press should be raised when this rule is triggered. This process is repeated until tilting of the mouse pan Microsoft's WorldWide Telescope in every direction.

Sketching a Bendable Display

SketchSpace also simplifies sketching interactions for emerging technologies. Gummi [12] presents a bendable hardware prototype that simulates the use of a flexible credit card sized display. To explore similar interactions, a designer finds a flexible material and manipulates its while monitoring the Mapping Panel's virtual bend sensor (see Figure 6). The designer uses


(a)

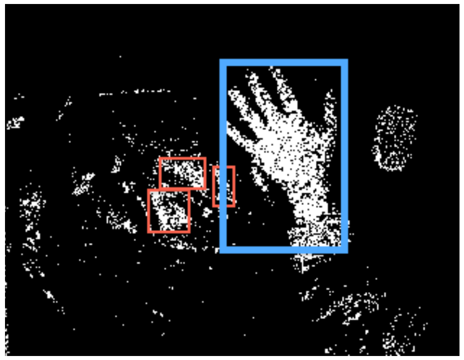

(b)

Figure 7. Skin densely reflects the infrared light emitted by the Kinect. We threshold the raw infrared image (a) and locate the hand (b). Although smaller candidates are sometimes found (red boxes), they are easily ignored.

Microsoft's WorldWide Telescope and bends the material to zoom in and out. Using the tilt operations from the last scenario, the designer navigates galaxies by combining bending and tilting interactions.

Implementation

SketchSpace's is a modified version of LightSpace [14]. It is implemented in C# in a Windows 7 environment and uses LightSpace's libraries to acquire and process depth data. Our single camera and projector setup is sampled at 50 Hz and calibrated in a single 3D coordinate system [14]. A grayscale image represents the sampled depth data and computer vision techniques use this image to identify and track up to two objects at a time. Once identified, an object's surface geometry is used to infer input behaviors and generate virtual sensor streams for each object.

From Input Behaviors to Virtual Sensors

To sense an object's input behaviors, the system thresholds the depth image and performs connected component labeling [14]. The position of a sufficiently large component indicates a potential object. From the component's bounding box, we extract the object's depth data and use this to generate a 3D surface mesh. Using the mesh, we calculate an object's volume, 3D position, and infer its rotation around its local Z-axis. A tracking engine compares these values to an index of previous values and generates a candidate match.

After an object is matched, we calculate its tilt using an object's vertical and horizontal major axis [14]. At this point, the object's three-dimensional position, rotation, and tilt are known. After this, curvature is computed by calculating the second derivative on the mesh using its surface normals. The result is a grayscale image that maps degree of curvature to pixel brightness.

Touch

Sensing touch requires (1) segmenting the designer's hand (see Figure 7) and (2) knowing when a fingertip deflects a surface. In SketchSpace's setup, the closely positioned Kinect floods the workspace with infrared light. Skin is a bright reflector of infrared light and is easily thresholded.

To sense touch, we sense a predictable pattern in the curvature at the boundary between a fingertip and object. After a dwell time of 500 ms, a touch event is raised. To reduce false positives, we confirm the fingertip's presence by indexing it in the segmented hand image. Unlike Wilson's use of depth sensing to sense touch [15], we do not require a static background model of the touch surface. Relaxing this requirement allows us to sense touch on moving objects. However, we currently limit our approach to a single touch point as natural grasping gestures interfere with multi-touch sensing.

Discussion & Conclusion

While we designed SketchSpace with detailed feedback from the interaction and industrial designers, it is still a work in progress. We plan to deploy SketchSpace to the rapid prototyping shop in the Microsoft's Hardware group, and observe it in use when designing new designs. We would also like to more formally evaluate our prototype to gather feedback from designers about its efficacy as an interactive sketching tool.

Also, early informal feedback from designers indicated that they would like to render richer content on their prototypes. At present, SketchSpace only projects simple visual feedback onto the device to indicate tracking and virtual buttons; however, most of the feedback the user gets from seeing the actions performed in the desktop application. We would like to extend our work to allow for interactive content to be located on the device, much like the work of Nam et al. [11] and Akaoka and Vertegaal [1]. This would allow the designer to render images, video, and trigger complex visual feedback, and enable deeper design exploration of physical prototypes.

Finally, imbuing passive material with virtual sensors has the potential to infer more abstract forms of interaction. Analyzing the depth data in a designer's studio space could reveal, for example, whether someone has entered a room, how many people are present, and how many times someone has opened or closed their laptop. Designing interactions for this level of sensing is a rich area for future exploration.

References

[1] Akaoka, E. and Vertegaal, R. DisplayObject: Prototyping Functional Physical Interfaces on 3D Styrofoam, Paper, or Cardboard Models. In *Proc. of ACM TEI'10.* 49-56.

[2] Ballagas, R., Memon, F., Reiners, R., and Borchers, J. iStuff Mobile: Rapidly Prototyping New Mobile Phone Interfaces for Ubiquitous Computing. In *Proc. of ACM SIGCHI'07*. 1107-1116.

[3] Buxton, B. Sketching User Experiences: Getting the Design Right and the Right Design. Morgan Kaufmann, San Francisco, CA, 2007.

[4] Gallant, D., Seniuk, A., and Vertegaal, R. Towards More Paper-like Input: Flexible Input Devices for Foldable Interactions Styles. In *Proc. of ACM UIST'08.* 283-286.

[5] Hartmann, B., Klemmer, S. R., Bernstein, M., Abdulla, L., Burr, B., Robinson-Mosher, A., and Gee, J. 2006. Reflective physical prototyping through integrated design, test, and analysis. In *Proc. of ACM UIST '06*. 299-308.

[6] Hudson, S. and Mankoff, J. Rapid Construction of Functioning Physical Interfaces from Cardboard, Thumbtacks, Tin Foil and Masking Tape. In *Proc. of ACM UIST'06*. 289 – 298.

[7] Kirsh, D. and P. Maglio. On distinguishing epistemic from pragmatic action. *Cognitive Science* 18. pp. 513-49, 1994.

[8] Knörig, A., Wettach, R., and Cohen, J. Fritzing – A Tool for Advancing Electronic Prototyping for Designers. In *Proc. of ACM TEI'09.* 351-358.

[9] Lee, J.C., Avrahami1, D., Hudson, S.E., Forlizzi, J., Dietz, P.H., Leigh, D. The Calder Toolkit: Wired and Wireless Components for Rapidly Prototyping Interactive Devices. *In Proceedings of ACM DIS'04*. 167-175

[10] http://msdn.microsoft.com/en-us/library/system.windows.forms.sendkeys.send.aspx. Last accessed January 14, 2011.

[11] Nam, T., and Lee, W. Integrating Hardware and Software: Augmented Reality Based Prototyping Method for Digital Products. In *Extended Abstracts of ACM CHI'03.* 956 – 957.

[12] Schwesig, C., Poupyrev, I. and Mori, E. Gummi: A Bendable Computer. In *Proc. of ACM CHI;04*. 263-270.

[13] Villar, N. and Gellersen, H. A Malleable Control Structure for Softwired User Interfaces. In *Extended Abstracts. of ACM TEI'07*. 49-56.

[14] Wilson, A., and Benko, H. Combining Multiple Depth Cameras and Projectors for Interactions On, Above, and Between Surfaces. In *Proc. of ACM UIST'10.* 273-282.

[15] Wilson, A. Using a Depth Sensor as a Touch Sensor. In *Proc. of ACM ITS'10*. 69-72