The Breadth–Depth Dichotomy: Opportunities and Crises in Expanding Sensing Capabilities

A simple touch is not simple. What we think of as "touch" actually includes a variety of object-sensing technologies and an even wider variety of information that can be detected about the sensed objects. This wide range of capabilities forces developers to choose between designing once for a "lowest-common-denominator" platform (breadth) or significantly redesigning their software for each hardware capability (depth). This dichotomy threatens the future of touch computing as a platform for innovation.

by Daniel Wigdor

THERE is no such thing as "touch." It is a vast category of sensors varying from fingertip location to object tracking. This wide range of sensors forces developers to choose between designing once for a "lowest-common denominator" platform (breadth) or significantly redesigning their software for each hardware capability (depth). This dichotomy threatens the future of touch computing as a platform for innovation.

The number of simultaneous touch points that a technology must detect in order to be termed "multi-touch" is not yet an agreed upon standard in the touch industry. While this is an interesting issue, a more pressing one for content developers is the guaranteed minimum number of contact points a hardware device will support if it is running a given platform. In the case of Android, it is one. In Windows Phone 7, it is four and in iOS 4, it is five. If a design team sets out to create a touch-based application, should they design it for a single touch point so that it will work on all of these platforms? Or should they design a different version of their application for each platform?

The pressure for software to be highly tailored to a device comes from designing for a good user experience. The more software is tailored to a given form factor, the better the experience can be, and the more highly differentiated the software will be. The pressure for homogenization comes from what might seem to be good business sense. The less tailored a piece of software is, the easier it is to distribute across multiple platforms.

How these choices are already being made on the iOS platform is illustrative. The author selected two-dozen iOS applications at random and compared their iPhone versions with their "designed for the iPad" versions. None of the applications had significant differences in the design for the two devices, despite a larger screen and the ability to give two-handed input on the iPad. The business arguments seem to be winning out; the iPhone was successful and the iPad is (so far) unproven, so content producers hedge their bets by releasing content designed for the iPhone with only minor changes (if any) for the iPad. If the availability of content to a new platform is oxygen, then the iPad might be having trouble breathing.

The Breadth–Depth Dichotomy

This phenomenon is not unique to touch computing. It is common to witness an explosion of variations of that technology with each laying claim to some unique property that supposedly makes it superior to the alternatives, especially in the early days of a technology's penetration into the marketplace. Direct current worked better with batteries, but alternating current could more easily be converted to different voltages. Wax-based phonographs could more accurately record sound, but flat gramophone records could be more easily mass-produced. Projective-capacitive touch screens can be better tuned to detect the first moment of a touch, but vision-based touch screens can identify objects and shapes.

The dangers of the breadth–depth dichotomy, whether to design a less robust platform that works across a broad range of applications or a more powerful one that works only in a particular and deep niche, are found in instances where technology differences require content producers to make

***Daniel Wigdor** is an Assistant Professor of computer science at the University of Toronto. He was the user-experience architect of the Microsoft Surface and is the author of the book, "Brave NUI World: Designing Natural User Interfaces for Touch and Gesture." He can be reached at dwigdor@dgp.toronto.edu or +1-416-978-7777.*

0362-0972/03/2011-0??$1.00 + .00 © SID 2011

choices that fundamentally affect content. Consider the choice faced by filmmakers regarding the framing decision. Should they frame their movies for the aspect ratio of standard television (1.33:1 = 4:3), widescreen television (1.78:1 = 16:9), or movie theaters (1.85:1 or 2.40:1) or should they produce three different films, each designed for a different venue? One solution for content intended mainly for television is to frame by using the 14:9 "action-safe area," a compromise between 16:9 and 4:3. For content intended for all three venues, the aspect ratios are so different that framing for the 14:9 action-safe area results in a significant compromise in the movie cinematographers' ability to express their vision. This is the heart of the breadth–depth dichotomy, where content producers are forced to choose between developing a single, watered-down design for multiple platforms (breadth), thus accessing more markets for a lesser cost or taking full advantage of the available technology to produce an ideal experience for each platform (depth).

In the case of input technologies, it is easy to see that variation can also affect content. This is because the content itself – the game, application, and even software platform – is fundamentally different depending on which sensing capabilities are targeted and leveraged. As will be shown, more than a dozen technologies all claim the ambiguous category of "touch," and content producers are already being forced to make choices between compromising content or missing certain markets.

Nintendo and the Dichotomy

To understand the dangers of the breadth–depth dichotomy in input devices, it is worth considering Nintendo's history of innovation with technologies. Nintendo's Wii has been a wild success, in no small part due to the deep game designs created to take advantage of its innovative input technology. The success of the Wii may lead one to forget several failed bets the company made in user-interface technology. One such bet was the Power Glove.

The Power Glove was worn by the user, and its position was tracked in three dimensions, as well as its roll, pitch, and yaw and the degree to which each finger was "curled." It could, in a technical sense, detect many of the sorts of gestures now made popular by the Microsoft Kinect gaming device. Although there were issues with the technology, that is not where the device failed. Instead, the source of the problem was that the makers of the glove emphasized breadth over depth by enabling the glove to control old games designed for the Nintendo controller. For the vast majority of the users, the Power Glove was simply emulating the controller they previously used to play their games. Designers of these experiences were retroactively disempowered, in that they had no opportunity to design, build, or test their game designs for use with the Power Glove. Reviewers and gamers alike agreed that the experience was terrible, and the glove was a business failure. However, it is striking that the Wii's success followed an almost identical technological path as the Power Glove, *i.e.*, leveraging cutting-edge sensors to provide game makers with additional input channels. This newfound success with innovative technology is in no small part due to Nintendo's decision to emphasize depth over breadth; rather than enable the control of content designed for a different input device, Wii games are designed for this new platform.

Nintendo had the luxury of emphasizing depth over breadth in the Wii in part because of its business relationships with content producers and a history as a platform producer. For hardware companies, depth can be an unaffordable luxury. Instead, they typically call on platform makers such as Microsoft and Google to create technologies that enable developers to strike a balance between breadth and depth. Unfortunately, in the case of touch computing, this call has gone largely unanswered.

To understand the scale of this problem, it is important to realize that touch technologies are already coming to be differentiated by far more than the number of simultaneous touch points they are able to detect. By relying on research systems as a guide, it can be predicted that at least seven types of capabilities will soon come to define touch technologies. The number of touch points sensed is just one of these seven.

A Taxonomy of Sensing Capabilities

To understand how deeply the breadth–depth dichotomy affects touch computing, it is essential to understand that touch itself is already fragmented in terms of the capabilities of each touch technology and that the fragmentation will only increase as more promising technologies make the transition from the research lab to consumer device. As shown in Fig. 1, these capabilities can be categorized into two high-level areas: (a) *sensed objects* (the types of objects which can be sensed), and (b) *sensed information* (the details which can be detected about the sensed objects).

Sensed Objects

The types of objects that can be sensed and the nature of the sensing have perhaps the most immediate effect on the user experience. The taxonomy defines three types of object sensing: *touch*, *stylus*, and *imagery*.

Touch: The number of touch points that a technology can detect is critical in differentiating a user experience. A technology might detect only a single touch (*single touch*) or it might provide sufficient contacts to sense a single user giving gestural input (*single-user gestural*) or multiple users giving gestural input simultaneously (*collocated gestural*). The related differences between a single-touch device, a single-user device that can accept multi-finger gestures, and multiple users on a large display are substantial.

Stylus: Pen computing has long occupied the niche of design professionals. Although the transition of pen computing to the consumer space has been bumpy, the recent popularity of the tablet form factor and touch's weakness in content production point to its likely return. Some touch technologies cannot detect a stylus at all (*none*). Others can detect a stylus, but cannot differentiate it from touch input (*recognized*). The most promising technologies are able to make that differentiation, enabling a user to use their fingers to manipulate the user interface, then seamlessly switch to a pen to draw, take notes, or annotate content.[1]

Imagery: Capturing imagery of objects in contact with the surface has been demonstrated to enable new interaction methods. Very few commercial technologies are able to provide applications with photographic imagery of what is in contact with the display. The detection of *text and graphics* enables scenarios where users can easily scan content in real time (imagine sharing a magazine article with a remote collaborator just by holding it up to the screen or producing a sketch on paper and animating it with the computer). Finer imagery detection can detect *fingerprints*, which would enable a degree of instant customization and access to online identities, and also address mundane but important issues such as dramatically increasing the precision of touch input.[2]

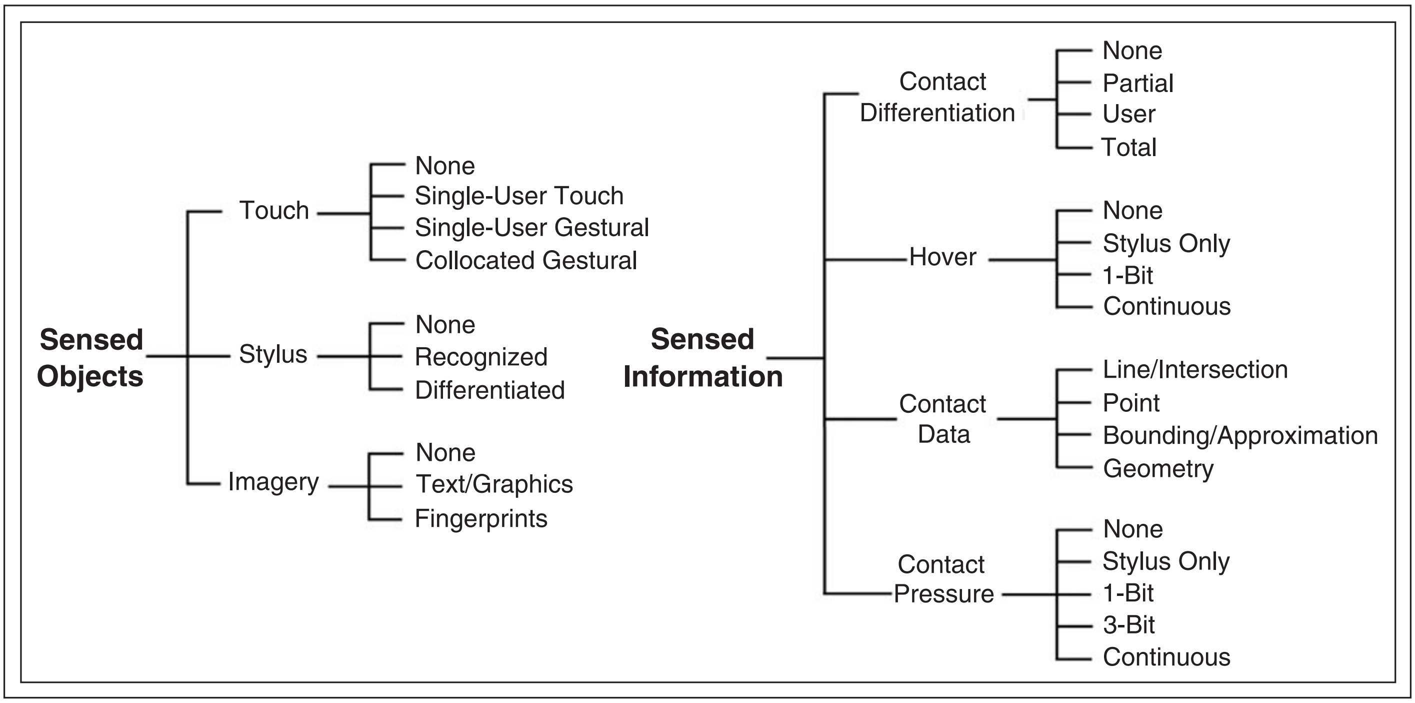

Fig. 1: *A taxonomy of sensing capabilities for touch computing helps show that the use of the generic term "touch" is clearly inadequate in the face of such a broad range of capabilities.*

Sensed Information

While the types of objects that can be sensed are an obviously important technological differentiator, the details of those sensed objects are equally important. Technologies have been demonstrated that are capable of four types of information sensing: *contact differentiation*, *hover*, *contact data*, and *contact pressure* (see Fig. 1). Just as the ability to detect and differentiate a stylus enables a clearly different user experience, these capabilities also create a need for additional depth of design, thus further reinforcing the dichotomy of breadth and depth.

Contact Differentiation: Contact differentiation refers to the ability to differentiate between contacts that are produced by different parts of the body or by different users. Systems capable of *partial-user* differentiation can successfully identify whether two contacts are coming from the same or different hands. For multi-user systems, the ability to differentiate between *users* is an absolute necessity. Basic user-interface elements such as paint canvases do not work with multiple users unless differentiation is present (*e.g.*, it is impossible for two users to paint in two different colors at the same time without being able to tie each user's color selection to their ink strokes).[3] Finally, *total* differentiation allows designers to identify each contact by the body part making that contact.

Hover: If a system is able to detect the presence of a touch object above the display and detect when that object touches the surface of the display, it is said to support hover. Hover can improve touch accuracy and can enable previews of actions that will occur when the surface is touched by highlighting objects or displaying. In the keyboard and mouse world, hover is emulated as "mouseover," *i.e.*, moving the mouse without clicking. Some technologies can sense hover only for the stylus (*stylus only*), while others can detect hovering fingers or other objects as well. Two types of hover detection have been demonstrated: *1 bit*, in which the system can differentiate between a hovering and a touching finger, and *continuous*, which can provide the height of a finger above the display. The ability to hover can also enable increased precision, for example, by enlarging the area beneath a hovering finger, as has been described by Autodesk Research.[4]

Contact Data: Little attention has been paid in the touch world to the type of information that is known about each contact. The majority of touch technologies reduce the entire contact area to a single x-y coordinate (*point*). Some technologies are limited to detecting only the column and row location of touches (**line/intersection**), which produces ambiguities when more than one contact is touching the device. Others are able to approximate the size of a contact with a bounding rectangle or other shape (*bounding/approximation*). Finally, the most advanced technologies are able to provide full *geometry* of a contact area. Knowing this geometry can enable advanced gestural inputs such as the "Rock and Rails" language shown in Fig. 2, in which the user can change system modes by touching the device with different postures.[5]

Contact Pressure: Some technologies are able to detect the force with which the user is touching the display. This data can be used to differentiate input (*e.g.*, a light touch can preview a rectangle, while a forceful touch places it on the canvas), as well as to vary continuous input (*e.g.*, control the size of the brush in a paint application). There are five different types of pressure sensing. Projected-capacitive devices such as the iPad cannot detect pressure (*none*). Some others are able to detect

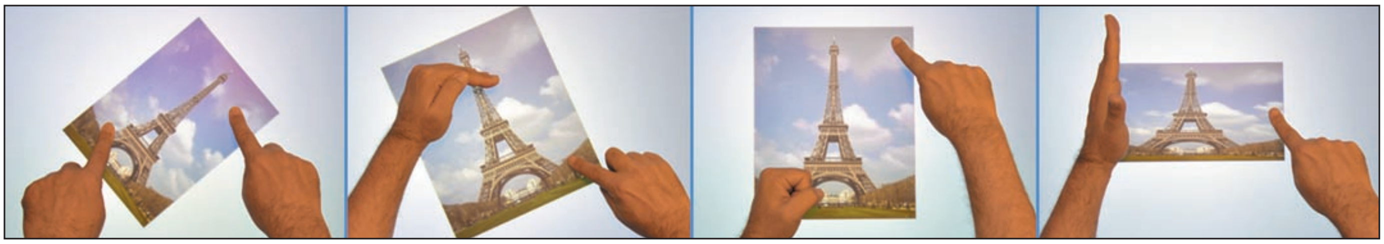

Fig. 2: Rock and Rails augments conventional direct-manipulation gestures (e.g., shown in the first photo) with independently recognized hand postures used to restrict manipulations conducted with the other hand (e.g., rotate, resize, and one-dimension scale, shown in the remaining three photos). This allows for fluid selection of degrees of freedom and thus rapid, high-precision manipulation of on-screen content.

the pressure only of an active stylus (*stylus only*). The remaining technologies can detect pressure with varying fidelity. Apple trackpads detect *1 bit* of pressure via a switch beneath the touch area, *3-bit* pressure is sufficient to enable the full range of human-discernable pressure levels, and continuous pressure can be used to create 3D-like experiences such as those in Perceptive Pixel's demonstration of easy stacking of objects in a scene by "pressing" them deeper.

Hardware and Software Support

This vast array of different capabilities, each laying ambiguous claim to the category of "touch," creates the challenge faced by content producers. Should they design their software so that it takes advantage of these capabilities, knowing that it will no longer function on some devices or should they de-feature their user experience to enable it to run across platforms? To facilitate this decision, it is important to know which touch-sensing technologies have been demonstrated to offer which sensing capabilities. Table 1 shows the various maximum capabilities that have been demonstrated using a given technology, usually in a research lab. It is worth noting that there is no commercial touch technology that is capable of everything shown in the table.

It is also worth noting that although individual technologies support various levels of sensing, the software platforms for which content producers most often develop their products are actually much more limited in their capabilities. Table 2 shows the levels of taxonomic properties that are supported by five common software platforms. Given this startling lack of support, one might conclude that breadth is the only option because none of the platforms presently supports much depth.

Table 1: Different sensing technologies appear with the levels of taxonomic properties they have been shown to support. Each technology is shown with an example product (in parentheses) that uses that technology. Note that these example products do not all meet the maximum capabilities that have been demonstrated in research labs.

	Analog Resistive **(ATM Machines)**	***Analog & Digital Multi-Touch Resistive*** **(NYU/TouchCo UnMouse Pad)**	***Direct Illumination*** **(Microsoft Surface, v1)**	***Frustrated Total Internal Reflection*** **(Perceptive Pixel Magic Wall)**	***Projected Capacitance*** **(iPad, iPhone)**	***Capacitive Coupling*** **(Circle Twelve DiamondTouch)**
Sensed Objects						
Touch	Single User Gestural	Collocated Gestural	Collocated Gestural	Collocated Gestural	Collocated Gestural	Collocated Gestural
Stylus	Recognized	Recognized	Differentiated	Recognized	Recognized	Differentiated
Imagery	None	None	Fingerprints	Fingerprints	None	None
Sensed Information						
Contact Differentiation	None	None	Partial	None	None	User
Hover	None	None	Continuous	None	None	None
Contact Data	Point	Geometry	Geometry	Geometry	Geometry	Geometry
Contact Pressure	None	Continuous	None	Continuous	None	None

Table 2: Five common software platforms are shown along with their capacity for conveying sensed information to applications.

	Windows 7	iOS 4	OSX	Android	Microsoft Surface, v1
Sensed Objects					
Touch	Collocated Gestural	Collocated Gestural	Collocated Gestural	Collocated Gestural	Collocated Gestural
Stylus	Differentiated	Recognized	Recognized	Recognized	Recognized
Imagery	None	None	None	None	Text/Graphics
Sensed Information					
Contact Differentiation	None	None	None	None	None
Hover	Stylus Only	None	None	None	None
Contact Data	Bounding/ Approximation	Point	Point	Bounding/ Approximation	Bounding/ Approximation
Touch Pressure	Stylus Only	None	None	Continuous	None

However, this would be short-sighted because the march of progress will undoubtedly increase the fidelity and range of information supported by these platforms.

Overcoming the Dichotomy: An Unmet Challenge

Several attempts have been made to alleviate the pressures associated with the breadth–depth dichotomy. Three such solutions are as follows: (1) finding a lowest-common denominator, (2) separating content from presentation, and (3) providing a level of abstraction for sensing capabilities. Unfortunately, none of these approaches has yet met the challenge, but in their attempts we can see the promise of potential solutions.

The lowest-common-denominator approach is simple. By using it, a designer surveys the capabilities of all target platforms and designs the application such that it requires only the minimum available capability across those platforms. This is the approach that guarantees the greatest breadth, thus also guaranteeing the least depth of design. This is also the most common approach to the dichotomy – it is the Power Glove's approach, and the approach of most applications developed for iOS thus far. It is highly dangerous to innovation because it guarantees that new sensors and capabilities are always ignored in favor of the status quo.

The second approach is to attempt to separate content from presentation. A good example of this approach can be found in HTML, which allows content designers to have tight control over the presentation form while still maintaining an abstraction. This approach has enabled Web sites to easily produce content for both desktop and mobile-phone browsers. It has also enabled the differentiation of input type in order to produce different controls (*e.g.*, the iPhone's method for selecting objects from a list is distinctly different from the method of making the same selection on a desktop). While this enables some opportunity for depth, it fails to provide a solution to the dichotomy because it imposes an assumed interaction model on a device. For example, the iPhone's list selector is a great touch interface, but should users really ever be forced to choose from a linear list on a touch device? This approach provides insufficient granularity to fundamentally alter interaction for a given technology and context.

The third approach is executed at a lower level by providing an abstraction of input capabilities. This approach allows applications to query the capabilities of a particular device and deliver a modified version of the application based on the results. This is the approach taken by Java Platform Micro Edition (J2ME), a less-powerful alternative to Android. This is the solution that enables the greatest depth for every application because it forces the content producer to consider each capability, alone and in combination, and design the best possible solution in a given context. The limitations of this approach are immediately apparent because the number of possible combinations makes the development of truly deep applications prohibitively expensive. In the taxonomy shown in Fig. 1, there are seven categories, each with 3–5 levels. Designing a deep application for every combination in this space would require 11,520 different designs.

Conclusions

The breadth–depth dichotomy is a challenge that must be met as the sensing capability of touch devices explodes. The pressure of business to favor breadth over depth is a crushing force on innovation that has often extinguished the fires of exciting new technologies. The pressure of design in favor of depth is also clearly untenable – creating a perfect design for every sensing profile is prohibitively expensive. Finding a way forward is the only hope the touch-computing industry has of achieving its promise of fundamentally changing the way users interact with computers.

Acknowledgments

I would like to thank several of my former colleagues at Microsoft – Paul Dietz, Ken Hinckley, Sarah Williams, Hrvoje Benko, and Dennis Wixon – for their help in preparing this article. I also would like to thank Maureen Larkin, a Toronto film historian.

References

[1]K. Hinckley *et al.*, "Pen + touch = new tools," *Proceedings of the 23nd annual ACM Symposium on User Interface Software and Technology* (2010).

[2]C. Holz, and P. Baudisch, "The generalized perceived input point model and how to double touch accuracy by extracting fingerprints," *Proceedings of the 28th International Conference on Human Factors in Computing Systems* (2010).
[3]P. Dietz and D. Leigh, "DiamondTouch: A multi-user touch technology," *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology* (2001).
[4]X-D. Yang, T. Grossman, and G. Fitzmaurice, "TouchCuts and TouchZoom: Enhanced Target Selection for Touch Displays using Finger Proximity Sensing," *Proceedings of the 29th International Conference on Human Factors in Computing Systems* (2011) (to be published).
[5]D.Wigdor *et al.*, "Rock & Rails: Extending Multi-Touch Interactions with Shape Gestures to Enable Precise Spatial Manipulations," *Proceedings of the 29th International Conference on Human Factors in Computing Systems* (2011) (to be published). ■