Beyond Pinch and Flick: Enriching Mobile Gesture Interaction

Yang Li, *Google Research*

An open source toolkit lets developers easily create mobile gesture applications.

During the past two decades, human-computer interaction has been dominated by the WIMP (window, icon, menu, pointer) paradigm. Though powerful, the interaction based on this paradigm requires much of a user's attention and can be slow to perform and hard to learn.

WIMP-based interaction is particularly problematic and awkward on mobile phones, which have a small form factor and are primarily used on the go. Mobile phone users want to be able to continue focusing on real-world tasks at hand rather than have to carefully key in commands, tap buttons, and navigate through menus.

Touchscreen gestures can provide a quicker, more intuitive way to operate a mobile phone.

Gestures have always played an important role in human communication (A. Kendon, *Gesture: Visible Action as Utterance*, Cambridge Univ. Press, 2004). They allow us to express a variety of feelings and thoughts that might be difficult or time-consuming to do verbally. Gestures also serve as an important visual complement to verbal communication. A thumbs up or down, for example, is a common gesture for appraisal.

Touchscreen technology, a major driver for the rapid growth of smart phones, not only lets a user view rich content such as high-resolution graphics or movies, it also enables gesture interaction. A user makes a gesture by sliding a finger on the touchscreen, and the resulting path can express numerous operations.

For example, pinch and flick are two representative gestures used by touchscreen phones such as the Apple iPhone: A user can pinch two fingers to zoom into or out of, say, a map, or scroll a list vertically by rapidly flicking a finger.

However, existing mobile phones support only a handful of built-in gestures. Touchscreen gesture-based interaction is thus a rich area to explore.

GESTURE TOOLKIT

As in human communication, touchscreen gestures can vary according to cultures and application domains. They can also be unique for individual users based on their experiences and preferences. For example, a user might draw a star-shaped gesture to call his wife, while another might draw the letter W to do so.

To unleash the power and richness of touchscreen gestures, developers and end users must be able to create their own gestures. To this end, my team at Google has developed an open source toolkit (http://developer.android.com/intl/en/sdk/index.html) that makes it easy to create mobile gesture applications.

The gesture toolkit is designed to address two major challenges.

First, gesture-based interaction requires a different mechanism for event dispatching and processing than WIMP-dominated mobile interfaces. A gesture may consist of more than one stroke, and a stroke is generated from a sequence of touch movements. The toolkit encapsulates the low-level details of gesture composition and rendering, and simplifies the integration of gesture-based interactions with existing mobile user interfaces.

Second, gesture recognition that maps ambiguous shapes or sequences to precise, executable commands requires sophisticated techniques such as machine learning that are beyond the capacity of many application developers. The gesture toolkit packages and delivers these technologies in a well-defined and developer-friendly way.

The toolkit supports the seamless integration of gesture-based inter-

0018-9162/09/$26.00 © 2009 IEEE Published by the IEEE Computer Society

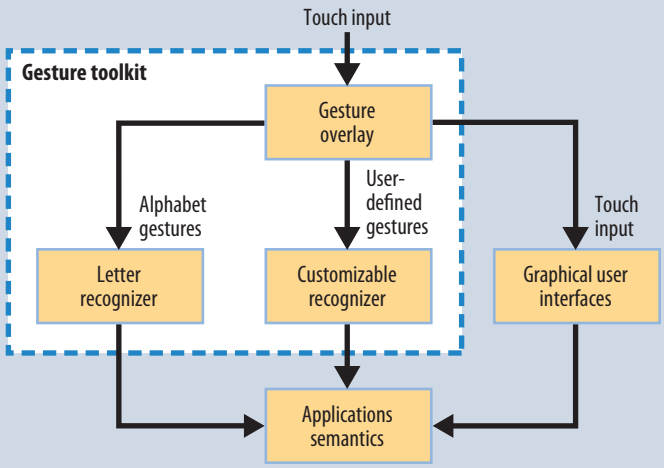

Figure 1. Gesture toolkit runtime architecture. The toolkit supports the seamless integration of gesture-based interaction and existing, GUI-oriented mobile interfaces and recognizes both built-in alphabet and user-defined arbitrary gestures.

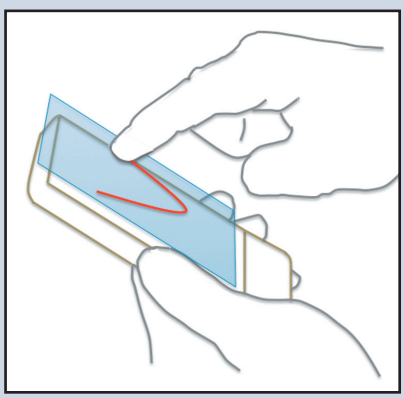

Figure 2. The gesture overlay, which collects touch events, wraps sequences of movements into gestures, and dispatches these to the application, is a transparent layer that a developer can stack on top of any interface widget.

action and existing, GUI-oriented mobile interfaces and recognizes both built-in alphabet and user-defined arbitrary gestures. Figure 1 shows the runtime architecture.

GESTURE OVERLAYS

The *gesture overlay* collects touch events, wraps sequences of movements into gestures, and dispatches these to the application. The overlay can be configured to accept single-stroke or multistroke gestures; currently, a timeout delimits multistroke gestures. A gesture has a set of properties, including a minimum bounding box and path length that developers can use to analyze its overall geometric features.

As its name suggests, the gesture overlay, shown in Figure 2, is a transparent layer that a developer can stack on top of any interface widget—for example, a list or dial keypad. Users can thus gesture and still be able to manipulate the underlying interface widgets as usual, such as scrolling a list or tapping a button. This interaction style enables a sufficiently large gesturing area that leverages the entire mobile phone screen, keeps the application interface visible, and avoids the need to explicitly activate a gesture mode.

The gesture overlay moderates event dispatching of the entire interface. It employs a combination of techniques to disambiguate gesture input from regular touch input, such as the variation of a touch movement path, and renders touch input differently according to its confidence in discerning gestures versus regular touch input.

GESTURE RECOGNITION

Mobile gestures can be application- or user-dependent. For example, gestures for navigating a map might differ from those for controlling a music player. In addition, users might create unique gestures as shortcuts for invoking their favorite applications. To address these diverse needs, the toolkit provides one engine that recognizes the English alphabet and another that recognizes arbitrary developer- or user-defined gestures.

To create a gesture recognizer for the English alphabet, we developed Gesture Sampler, a data collection tool based on the Android phone. Gesture Sampler asks users to write each of the alphabet's 26 letters multiple times in random order to capture the natural variation of each user's writing style.

We asked a sample of users to download Gesture Sampler and install it on their Android phones. Participants could finish these data-collection tasks wherever and whenever they chose, with the data automatically uploaded to a server upon completion. This approach allowed us to collect more realistic data than in a laboratory setting.

Using the collected data, we trained a neural network classifier to recognize letters independent of specific writing sequences or the number of strokes. Although the *letter recognizer* only recognizes a fixed set of gestures, it does not require any training of developers or users.

In contrast to the letter recognizer, the recognizer for arbitrarily defined gestures must be able to quickly learn from examples given at development time or by an end user at runtime. The *customizable recognizer* thus uses a nearest-neighbor approach to match an unknown gesture against a library of known gesture examples. Each gesture in the library is associated with a meaningful name such as "search" or "copy." Multiple examples might share the same name to reflect a gesture's variation.

The gesture library can be configured to be sequence- or orientation-sensitive, with accordingly different preprocessing and classification methods. For example, when the

library is sequence-sensitive, drawing a circle clockwise or anticlockwise are different gestures; when it is orientation-sensitive, drawing left to right is different from drawing right to left.

Developers can easily add application-specific gestures through a gesture library without having to know any details about gesture recognition. They can also easily create an application that lets users customize their own gesture set (http://android-developers.blogspot.com/2009/10/gestures-on-android-16.html).

MOBILE GESTURE APPLICATIONS

With the gesture toolkit, developers can build various gesture-enhanced mobile applications.

One typical use of gestures is their serving as shortcuts to frequently accessed contents such as contacts, applications, or executable commands in general. For example, a user can associate a specific gesture with an application or data item and then bring it up by drawing the gesture, thereby avoiding the effort of searching for the item. Figure 3 illustrates a gesture map application created using the toolkit in which the user can, say, draw the letter h to quickly obtain driving directions from the current location to home.

In addition, gestures are expressive enough to convey various manipulation semantics in applications. For example, the shape of a gesture can be used to represent an operation, while the object on which the gesture is drawn can indicate the operand. A user could thus efficiently manipulate a target object such as a webpage, photo, or character in a game—say, draw a star shape on a photo to actually star it.

Developers can also create free-form note-taking or annotation applications based on gestures. For example, a user could compose a grocery-shopping list on his phone or annotate a picture shared by a friend and send it back along with the gesture comments. The gesture toolkit provides basic digital-ink processing capabilities to layout, render, and store gesture data.

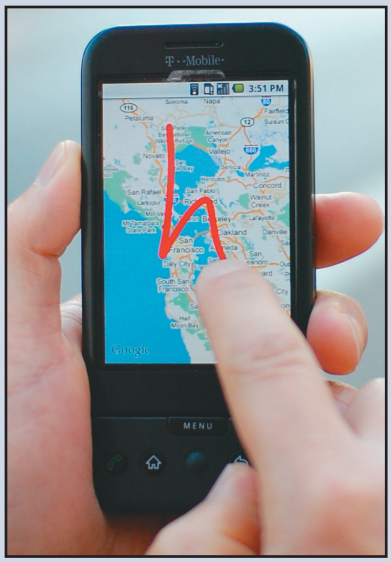

Figure 3. Example mobile gesture application. A user can quickly search for driving directions from his current location to home by simply drawing the letter h on his device's touchscreen.

There is great potential for gesture-based interaction on touchscreen mobile phones. The open source gesture toolkit enables developers to easily create a wide variety of mobile gesture applications, and provides gesture recognition components for both built-in alphabet and user-defined arbitrary gestures. We continue to improve the toolkit to provide support for even richer gesture-based capabilities on mobile devices.

Yang Li is a research scientist at Google Research specializing in human-computer interaction. Contact him at yangli@acm.org. *The author would like to thank the Android team for contributions to the Android gesture toolkit.*

Editor: Bill N. Schilit, Google; schilit@computer.org