Software Engineering Abstractions for the Multi-Touch Revolution

Lode Hoste
Vrije Universiteit Brussel
Pleinlaan 2
1050, Elsene
lode.hoste@gmail.com

1. PROBLEM AND MOTIVATION

Multi-touch interfaces allow users to use multiple fingers to provide input to a graphical user interface. The idea of allowing users to touch and manipulate digital information with their hands has been subject of research for more than 25 years [5, 4]. Recently several of these research artifacts have found their way to industry, with examples like the iPhone and the Microsoft Surface. Mainstream programming languages do not offer support to deal with the complexity of these new devices. Unlike the evolution in the hardware technology, the complexity of these new devices has not yet been addressed by adequate software engineering abstractions.

Current multi-touch frameworks provide a narrow range of hardcoded functionality like *pinch*, *rotate* and *move* known as multi-touch gestures. There is however a substantial need to develop new and more gestures for domain specific applications. Multi-touch devices are inherently concurrent and provide a continuous stream of events. In many of these frameworks capturing these events to extract gestures is done by means of event handlers. Programming multi-touch devices with event handlers is cumbersome for a number of reasons.

First, programming with event handlers introduces a lot of overhead, even when implementing a simple two-finger gesture. This is because the programmer has to manually store and combine the events generated by several event handlers. Moreover, he has to manually garbage collect the stored events when they become uninteresting.

Secondly, current state of the art hardware does not provide the programmer with any information about which fingers are being used. The only information available in an event is an identifier and a position. In many multi-finger gestures it is crucial to map the identifiers to the correct fingers. Implementing this mapping manually complicates the event handling code significantly.

Thirdly, the programmer has to deal with conflicts between multiple gestures. For example, the well known *pinch* gesture only tracks the distance between two finger, which may conflict with the *rotate* gesture. This creates the need to specify some priority between gestures in order to resolve the conflicts when multiple gestures are detected. In current approaches this conflict handling code scattered over the detection of multiple gestures. Therefore, adding a new gesture requires deep knowledge about the existing gestures.

Finally, it is difficult in current approaches to maintain temporal invariants, for example when implementing the click and double click gestures. The programmer has to use a timer which takes care of triggering the click gesture after a period of time if no new events came in which triggered the double click gesture. This code is distributed over multiple event handlers, again complicating the event handling code significantly.

We claim that current multi-touch libraries providing simple event handlers are too low level, resulting in complex event handling code. As a result, we advocate the use of a rule language which allows programmers to derive useful patterns out of the events generated by the multi-touch device. The advantage of such an approach is that the programmer no longer needs to be concerned about how to derive gestures but only about describing the gesture.

2. UNIQUENESS OF OUR APPROACH

Our proposed solution finds its roots in Complex Event Processing (CEP) [8]. CEP deals with analyzing and correlating multiple events, with the goal of identifying meaningful events. Existing CEP implementations only focus on temporal reasoning with events. In order to cope with the needs of the multi-touch domain, spatial operators are also needed, for example in a *drag* gesture. Spatio-temporal data models have already been discussed in many research papers [7, 2]. They were however not intended for use in this domain. To the best of our knowledge our work is unique in applying high level language concepts for the development of multi-touch enabled applications.

We wrote a driver which converts low level signals to TUIO [3] events[1], a protocol supported by many multi-touch interfaces. These TUIO events are fed into a reasoning engine, as logical facts which are evaluated against a set of rules. Facts are stored as tuples in the following form:

```
(Factname attribute1, attribute2, ..., attributeN)
```

Next to these TUIO events, changes to the GUI are also fed to the reasoning engine. For example when moving a picture, the surface where the move gesture applies must also

[1]Our Stantum TUIO bridge implementation can be downloaded at http://github.com/Zillode/Stantum-TUIO-bridge

ICSE '10, May 2-8 2010, Cape Town, South Africa

be altered.

Figure 1 shows a *tap* (or click) gesture in our rule language. Our syntax is based on CLIPS [1]. The left-hand side of the rule (before the ⇒) contains a number of condition elements that are matched against the fact-base. Actions on the right-hand side are taken when the left-hand side is satisfied. Variables in a rule are denoted by a question mark.

```
(defrule myTap
  (declare (salience 100))
  (SetOf
    (TUIO_Event ?fingerID,_,_,Within: 300),
    ?eventList),
  ?eventList[0].state == BIRTH,
  (isListOf ?eventList[1..-2].state, MOVE),
  ?eventList[-1].state == DEATH,
  (veryNearPosition ?eventList)
  =>
  (assert (Tap ?fingerID, ?eventList[0].position)))
```

Figure 1: Tap gesture

When the rule *myTap* triggers, the fact *Tap* is asserted containing a specific fingerID and the position where the tap was performed. The matching is done by a logical rule using the facts from the multi-touch devices which have the following form:

```
(TUIO_Event fingerID, position, state, time)
```

The fingerID attribute is a unique identifier for one finger. The position is a pair of the X-Y coordinates. There are three possible states: *birth* (finger down), *move* (finger update) and *death* (finger up). The final attribute denotes the time of detection in milliseconds.

SetOf selects the list of TUIO events grouped by fingerID within 100 milliseconds and binds it to the variable *eventList*. The list is ordered by ascending time, *list[0]* being the oldest event. Negative indexes are used relative to the end of the list, meaning *list[-1]* being the youngest event.

The next three lines in the rule specify respectively, that the first element in the list should be a *birth* event, in between the first and the last there should only be *move* events and the last element should be a *death* event. Finally the last line in the rule uses the *veryNearPosition* spatial operator, forcing all events to be very near to each other. This constraints the *tap* event to be performed in a certain range since a tap should not happen with a lot of movement.

We introduced a priority concept in order to provide the user with easy accessible event consumption. Events that are matched by a rule with a higher *salience*, will not be available to the other rules. When two rules have the same priority, they share the ownership of the event.

Evaluating these rules for every event that enters the fact-base, even for a modest rule set, would be far too slow. Rather than reevaluating the entire fact-base against all the rules, we use an optimization technique called RETE which caches intermediate results. Therefore, it is relatively efficiently to perform the permutation of fingerID's. Our approach is unique to combine the concepts of CEP with an optimized RETE engine for the multi-touch environment. The result is an efficient high level language to describe multi-touch gestures.

3. RELATED WORK

Multi-touch libraries can be roughly divided into two approaches. A first approach, adopted by many of the industry platforms, such as the iPhone and Windows 7 SDK, is under the form of a low level library that allows one to capture the events sent by the multi-touch device. In many of these libraries capturing these events is done by creating event handlers. In these systems the programmer has to: manually store and combine the events generated by several event handlers, deal with conflicts between multiple gestures, keep track of temporal invariants, and find out which identifier maps to which finger. All this makes that programming with such approaches is cumbersome.

A second approach adopted by gesture frameworks is to make use of template matching [6]. However in these frameworks the detection is limited to single point and stroke gestures. When template-based recognition gets extended with multiple points, it will still be complex to recognise collaborative gestures because of the variability of positions.

4. CONTRIBUTIONS

The complexity of human-computer interaction will greatly increase with the rise of multi-touch interfaces. In order to deal with this evolution, we need adequate software engineering abstractions. We presented a first step in that direction in the form of a domain-specific language supporting spatio-temporal operators. This makes it possible to describe gestures using simple constructs. It allows programmers to easily deal with multiple fingers, hands and persons. We removed the complexity to manually store and combine the events generated by several event handlers and the problems that flow forth from this approach. We have implemented a fully functional proof of concept prototype[2], and we are currently optimizing it for speed. First experiments with our prototype have revealed that our approach scales well for traditional gestures as well as for some more exotic gestures that we have implemented.

5. REFERENCES

[1] J. Giarratano, G. Riley. Clips Manuals, version 6.0. Johnson Space Center, NASA, 1993.

[2] C. Holzmann. Rule-based reasoning about qualitative spatiotemporal relations. In *MPAC 07: Proc. 5th Intl. workshop*, pages 49–54, NY, USA, 2007. ACM.

[3] M. Kaltenbrunner, T. Bovermann, R. Bencina, and E. Costanza. TUIO: A protocol for table-top tangible user interfaces. In *GW 2005: Proc. 6th Intl workshop*. Citeseer, 2005.

[4] S. Lee, W. Buxton, and K. Smith. A multi-touch three dimensional touch-sensitive tablet. *ACM SIGCHI Bulletin*, 16(4):21–25, 1985.

[5] N. Mehta. A flexible machine interface. 1982.

[6] D. Rubine. Specifying gestures by example. *ACM SIGGRAPH Computer Graphics*, 25(4):337, 1991.

[7] S. Schwiderski-Grosche and K. Moody. The spatec composite event language for spatio-temporal reasoning in mobile systems. In *DEBS 09: Proc. of the 3rd ACM Intl. CONF.*, pages 1–12, NY, USA, 2009. ACM.

[8] E. Wu, Y. Diao, and S. Rizvi. High-performance complex event processing over streams. In *Proc. 2006 ACM SIGMOD Intl. CONF.*, page 418. ACM, 2006.

[2]A short movie of our working prototype can be downloaded from http://www.youtube.com/watch?v=wCSppBNNPEI