GeForMTjs: A JavaScript Library Based on a Domain Specific Language for Multi-touch Gestures

Dietrich Kammer, Dana Henkens, and Rainer Groh

Fakultät Informatik
Professur Mediengestaltung
Technische Universität Dresden
01062 Dresden
dietrich.kammer@tu-dresden.de, dana.henkens@googlemail.com

Abstract. This paper presents GeForMTjs, a library which features an abstract way of representing multi-touch gestures. A domain specific language for multi-touch gestures, Gesture Formalization for Multi-touch (GeForMT), is adapted to the needs of web development. Web standards are addressed and mouse input is incorporated as well. A short overview of related work shows that a formal abstraction of multi-touch gestures is missing in the web context. A brief example illustrates the seven processing steps of the library.

Keywords: Gestures, Multi-touch, CSS, JavaScript, Web standards.

1 Introduction

Multi-touch interaction is currently almost ubiquitous with web browsers on mobile devices. Although a set of standard navigational gestures are used throughout these browsers, a great potential for more complex gestural interaction remains to be researched. The web events working group of the W3C is currently developing a standard to integrate multi-touch and pen input in web sites. The official recommendation is due in August 2012 [1]. However, few of the currently available web libraries assist the programmer in the definition of application specific multi-touch gestures.

This paper contributes an implementation of a domain specific language (DSL) for multi-touch gestures in JavaScript. It is more powerful than relying on fixed gesture events or raw touch data. The short, concise, and self-explanatory syntax is graspable for both developers and designers. Providing gesture definition and recognition in a library should help web programmers to design and test novel interaction concepts.

2 Related Work

Most web applications on mobile devices use standard gestures and seek to emulate native multi-touch concepts available on each platform. Examples are jQuery Mobile [2], the Dojo-plugin dojox.mobile [3], and Sencha Touch [4]. WKTouch [5] focuses object manipulation, where gestures and actions are implicitly assigned to objects. Jester [6] provides a library of common standard gestures. Representations of gestures on a higher abstraction level are investigated by researchers such as Kin et al. [7], Khandkar and Maurer [8], and Kammer et al. [9]. Currently, these approaches are not available on the web. Libraries such as Moousture [10] are rather limited in expressing complex multi-touch gestures. A greater freedom and ease to design gestures can result in better and more powerful multi-touch interfaces in the future.

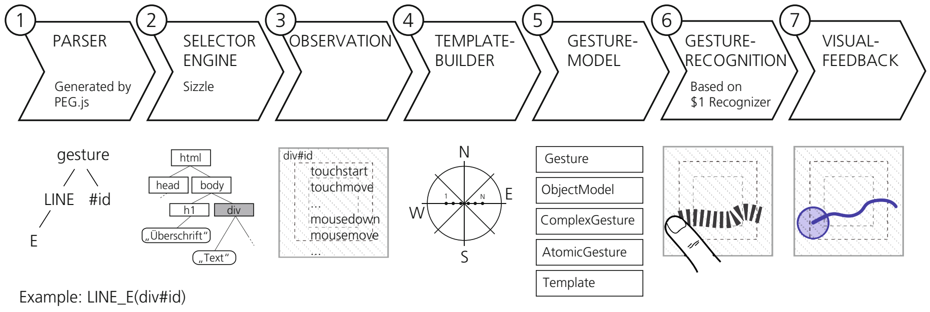


Fig. 1. Library components and processing steps of GeForMTjs

3 Gesture Library GeForMTjs

GeForMTjs relies on web standards to make touch data available in the web browser. Seven components are responsible for registering and processing gestures (see Fig. 1).

Step 1: The Gesture Formalization for Multi-touch (GeForMT) by [9] provides a DSL for multi-touch gestures defined by a context-free grammar. GeForMT features atomic gestures, which describe the form or path of a gesture and operators to describe the temporal progression of gesture strokes. Complex gestures are defined by combining atomic gestures. A GeForMT expression is validated by the Parser and split into syntactical units. PEG.js is used to generate a concrete parser implementation (http://pegjs.majda.cz/, last access: 05/09/2012), which is based on the parsing expression grammar formalism [11].

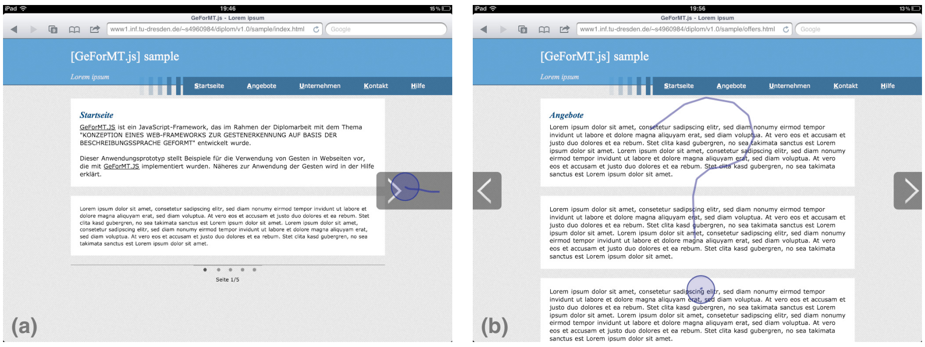
Step 2: The Selector Engine checks selectors contained in the gesture description for focus definition and returns corresponding nodes of the DOM tree. For the concrete implementation, Sizzle (http://sizzlejs.com/, last access: 05/09/2012) is used.

Step 3: The Observation module registers mouse and touch events for these elements. Mouse events are emulated as single touch gestures by dispatching appropriate touch events (cp. [12, 13]). To detect gesture input on content or structure elements (e.g. div) as well as on underlying parent elements of the DOM tree (e.g. html), the bubbling strategy of events is adopted. The programmer can define a contiguity interval to allow the specification of gestures that require the user to lift all fingers from the multi-touch display, e.g. a double-tap.

Step 4: The Template Builder converts formal parameters of the parsed expressions into a computable data structure containing ordered coordinates.

Step 5: Results of Parser and Transformation are stored in the Gesture Model.

Step 6: Based on these templates, Gesture Recognition is performed. Wobbrock et al.'s $1-Recognizer [14] has a good balance between recognition rate, memory



[GeForMT.js] sample

Lorem ipsum

Startseite

GeForMT.JS ist ein JavaScript-Framework, das im Rahmen der Diplomarbeit mit dem Thema "KONZEPTION EINES WEB-FRAMEWORKS ZUR GESTENERKENNUNG AUF BASIS DER BESCHREIBUNGSSPRACHE GEFORMT" entwickelt wurde.

Dieser Anwendungsprototyp stellt Beispiele für die Verwendung von Gesten in Webseiten vor, die mit GeForMT.JS implementiert wurden. Näheres zur Anwendung der Gesten wird in der Hilfe erklärt.

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

[GeForMT.js] sample

Lorem ipsum

Angebote

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Fig. 2. Sample web page using GeForMTjs

requirements, and tolerance and is suitable for a JavaScript implementation. A key advantage is the minimal effort used for feature extraction at runtime. Wobbrock et al.'s algorithm is adapted according to the specification of GeForMT including sequentially and simultaneously performed gesture paths, as well as gestures that are continuously recognized. In these cases, the steps of classification are processed in repetition.

Step 7: Visual feedback is provided and application specific event handlers are called for recognized gestures on their corresponding DOM elements. Gestures strokes and contacts are visualized on two separate overlaying canvas elements, which are excluded from event processing. Sequential gesture paths are considered in the feedback visualization as well. The gesture expression illustrated in Fig. 1 defines a line drawn to the east on a div element. The definition is embedded in JavaScript code and is registered with the API of GeForMTjs as follows:

```
GeForMT.addGesture ({
  identifier: "swipe",          // unique identifier
  expr: "LINE_E(div#id)",       // GeForMT expression
  online: true,                 // continuous/discrete recognition
  handler: function(e) { … } // gesture specific event handler
});
```

GeForMTjs can be seen in action in a test environment[1] demonstrating example gesture sets and a sample web page[2] (see Fig. 2), which substitutes access keys with stroke shortcuts to access menu entries. Browser functions can be accessed by gestures as well, for example browsing through the history or bookmarks. If a gesture cannot be recognized, a short information is displayed as a layer on top of the website, which indicates how to access the help page.

4 Conclusions and Future Work

The library presented in this paper is based on a DSL for multi-touch gestures. It complies with web standards to reap the benefits of platform-independent, web-based

[1] http://vi-c.de/geformtjs/testbench/

[2] http://vi-c.de/geformtjs/sample/

development. GeForMTjs supports different interaction techniques by generalizing mouse, touch, and pen input. Extensions like the browser plugin npTUIOClient and MagicTouch [12] working with the TUIO protocol [15] are considered as well for more hardware independence. However, further performance tests and web browser compliance must be tested and ensured. An important issue is the visualization of feedback and feed-forward, which reveals available gestures in an application. Another interesting possibility is the combination of GeForMT with a UI library. Combining GeForMT with other DSLs or adding extensions might make it feasible to address other modalities such as speech, spatial gestures, or video processing.

References

1. Brubeck, M., Moon, S., Schepers, D.: Touch Events version 1, `http://www.w3.org/TR/touch-events/` (last access: September 05, 2012)
2. jQuery: jQuery Mobile, `http://jquerymobile.com/` (last access: September 05, 2012)
3. Dojo: Dojo Mobile, `http://dojotoolkit.org/features/mobile` (last access: September 05, 2012)
4. Sencha: Mobile JavaScript Framework for HTML5 Web App Development | Sencha Touch, `http://www.sencha.com/products/touch` (last access: September 05, 2012)
5. Gibson, A.: WKTouch, `https://github.com/alexgibson/WKTouch` (last access: September 05, 2012)
6. Seaward, S.: Jester, `https://github.com/plainview/Jester` (last access: September 05, 2012)
7. Kin, K., Hartmann, B., DeRose, T., Agrawala, M.: Proton: Multitouch Gestures as Regular Expressions. ACM, Austin (to appear, 2012)
8. Khandkar, S., Maurer, F.: A Domain Specific Language to Define Gestures for Multi-Touch Applications. In: Rossi, M., Tolvanen, J.-P., Sprinkle, J., Und Kelly, S (hrsg.) Proceedings of the 10th Workshop on Domain-Specific Modeling (DSM 2010), Aalto University School of Economics, B-120, Aalto-Print (2010)
9. Kammer, D., Wojdziak, J., Keck, M., Groh, R., Taranko, S.: Towards a formalization of multi-touch gestures. In: ACM International Conference on Interactive Tabletops and Surfaces. S.49–S.58. ACM, New York (2010)
10. Sibt-e-Hassan, Z.: Moousture, `http://maxpert.github.com/moousture/` (last access: September 05, 2012)
11. Ford, B.: Parsing expression grammars. In: Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pp. S.111–122. ACM Press (2004)
12. Smus, B.: MagicTouch, `https://github.com/borismus/MagicTouch` (last access: September 05, 2012)
13. Carstensen, B.: Phantom Limb | Vodori Blog, `http://www.vodori.com/blog/phantom-limb.html`
14. Wobbrock, J.O., Wilson, A.D., Li, Y.: Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In: Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology, pp. S.159–S.168. ACM, New York (2007)
15. Kaltenbrunner, M., Bovermann, T., Bencina, R., Costanza, E.: TUIO: A Protocol for Table-Top Tangible User Interfaces. In: Gehalten auf der 6th International Workshop on Gesture in Human-Computer Interaction and Simulation, Vannes, France Mai 18 (2005)