Multi-Touch im Browser eines Tabletop-PCs
Frameworks, Einsatzmöglichkeiten und Einschränkungen

Martin Grubinger

Diplomarbeit Jahrgang 2011/2012
Masterstudiengang Digitale Medientechnologien, FH St. Pölten

martin.grubinger@gmail.com

Zusammenfassung

Die Verwendung von Multi-Touch als Eingabemethode hat in den letzten Jahren stark zugenommen. Die Entwicklung von Software für Multi-Touch-Tabletop-PCs wird derzeit vorwiegend mit traditionellen Softwareentwicklungswerkzeugen realisiert, woraus eine hohe Plattform- und Geräteabhängigkeit resultiert. Die Technologien im Umkreis von HTML5 und CSS3 ermöglichen die Ausführung umfangreicher Webanwendungen und Spiele im Webbrowser ohne zusätzliche Plug-ins. Zudem unterstützen immer mehr Browser Zugriff auf Touchinformationen des Systems. Aufgrund dieser Entwicklungen wird in dieser Arbeit untersucht, ob der Browser des Tabletop-PCs eine Plattform für ansprechende, gestenbasierte Multi-Touch-Applikationen darstellt und als Alternative zu Multi-Touch-fähiger Desktopsoftware betrachtet werden kann.

Um diese Frage zu beantworten, wurden die Fähigkeiten moderner Webbrowser zur Verarbeitung von Multi-Touch-Events untersucht und unterschiedliche Frameworks evaluiert, die bei der Entwicklung derartiger Anwendungen und webbasierten Animationen unterstützen. Unter Verwendung des Frameworks jQMultiTouch, das browserspezifische Touchevent-APIs vereinheitlicht und Gestenerkennung durchführt, wurde ein Prototyp eines einfachen Multi-Touch-Spiels entwickelt. Anhand eines Benutzertests dieses Prototyps wurde die Verwendbarkeit von gestenbasierten Multi-Touch-Applikationen auf Basis von Webtechnologien evaluiert.

Im Test dieses Prototyps konnte die generelle Verwendbarkeit gezeigt werden, jedoch wurden Einschränkungen hinsichtlich der Performance sowie der Anzahl der unterstützten Berührungspunkte festgestellt. Aufgrund dieser Einschränkungen sind Browser derzeit nur für die Ausführung von einfachen gestenbasierten Multi-Touch-Anwendungen auf Tabletop-Systemen geeignet. Für

Multi-User-Anwendungen mit hohen Grafikanforderungen und komplexen Gesten sind Webbrowser (noch) nicht geeignet.

1 Einleitung

Tap, Swipe, Drag – die Steuerung unserer Mobiltelefone und Tablet- oder Notebook-Computer mittels Multi-Touch-Gesten ist in den letzten Jahren beinahe zur Selbstverständlichkeit geworden. Mit Produkten wie Microsofts PixelSense[1] wird diese Art der Interaktion auch für großflächige, in Tischen integrierte Bildschirme verwendet. Eine locker durchgeführte Geste genügt, um Inhalte zu vergrößern, das nächste Foto anzusehen oder durch lange Listen zu scrollen.

Gleichzeitig werden immer komplexere Anwendungen für die Ausführung im Browser entwickelt. Die Technologien hinter diesem Trend sowie die Webbrowser sind ständig im Wandel begriffen und werden mit sehr kurzen Releaseabständen weiterentwickelt. Im Umfeld der Schlagworte HTML5 und CSS3 entstehen Technologien, die es ermöglichen, Applikationen für das Web zu entwickeln, die noch vor wenigen Jahren ohne die Verwendung zusätzlicher Plug-ins undenkbar gewesen wären (Pilgrim, 2010). Die Vielzahl von Anwendungen, die im den Chrome Web Store[2] für die Ausführung im Browser Google Chrome verfügbar sind, verdeutlichen diesen Trend.

Diese Entwicklungen legen den Schluss nahe, dass auch immer komplexere Multi-Touch-Applikationen direkt im Browser ohne zusätzliche Plug-ins oder Runtimes ausführbar sind. Derzeit werden Multi-Touch-Anwendungen mit Gestenunterstützung meist als Desktopapplikationen auf Basis von traditionellen Softwareentwicklungswerkzeugen und Programmiersprachen wie C++, C# oder ActionScript umgesetzt und sind daher als eigenständige Anwendung auszuführen. Desktopapplikationen für mehrere Plattformen zu entwickeln, ist mit hohem Aufwand und hohen Abhängigkeiten zu anderen Softwarebibliotheken verbunden.

Multi-Touch-Anwendungen mit Webtechnologien zu realisieren, bietet einige Vorteile gegenüber Desktopanwendungen. Webapplikationen besitzen durch die zugrunde liegenden Technologien eine hohe Geräte- und Plattformunabhängigkeit sowie einfache Verbreitungsmöglichkeiten. Eine äußerst

1 http://www.pixelsense.com

2 https://chrome.google.com/webstore

aktive, globale Entwickler/innen-Community ist an der Weiterentwicklung von Webtechnologien und moderner Browser interessiert. Browserhersteller integrieren in kurzen Release-Abständen neue Technologien im Umfeld von HTML5 und verbessern die Leistungsfähigkeit für die Darstellung von aufwendigen Grafiken und Animationen. Eine Reihe von Werkzeugen vereinfacht zudem die Erstellung von Animationen auf Basis von Webtechnologien.

Anhand der Umsetzung eines Prototypen für ein Puzzlespiel wurden die aktuellen Möglichkeiten zur Erstellung von gestenbasierenden Multi-Touch-Applikationen auf Basis von Webtechnologien untersucht und in einem Test mit vier Testpersonen auf ihre Verwendbarkeit überprüft.

2 Multi-Touch im Browser

Die wichtigste Voraussetzung für Multi-Touch-Applikationen auf Basis von HTML und JavaScript ist die Möglichkeit, auf Touchinformationen im Browser über Events zugreifen zu können. Das W3C[3] erkannte die Notwendigkeit, diese Multi-Touch-Events zu standardisieren, und veröffentlichte im Dezember 2011 eine Spezifikation für Touchevents[4] mit Candidate-Recommendation-Status. Folgende Events vom Objekttyp TouchEvent sind verfügbar: Das Event touchstart wird gesendet, wenn ein Finger den Bildschirm berührt; touchmove, wenn der Finger bewegt wird, und touchend, wenn der Finger die Oberfläche verlässt. Zusätzlich ist das Event touchcancel vorgesehen, welches ausgelöst wird, wenn das System die Verfolgung einer Berührung abbricht. Ein Beispiel für dieses Event ist, wenn ein Finger aus dem Multi-Touch-fähigen Bereich ohne Unterbrechung in die grafische Benutzerschnittstelle des Browsers bewegt wird (Brubeck, Moon & Schepers, 2011; Flanagan, 2006, S. 456).

Mozilla führte im März 2011 mit der Version 4 seines Browsers Firefox ein API für die Verwendung von Touchevents unter Windows 7 ein (Mozilla, 2011). Zu diesem Zeitpunkt war die Spezifikation des W3C noch nicht finalisiert. Aus diesem Grund entwickelte Mozilla ein eigenes, proprietäres API

3 W3C: World Wide Web Consortium

4 Anmerkung: Der Autor unterscheidet hier zwischen zwei Schreibweisen: *Touchevents* in der deutschen Schreibweise und *Touch Event* in Eigennamen wie „W3C Touch Events“.

zur Verwendung von (Multi-) Touch-Informationen in Webdokumenten. Das Mozilla-Touch-Event-API ist ähnlich konzipiert wie das vom W3C empfohlene. Es unterscheidet sich jedoch in einigen Punkten. Die von Mozilla verwendeten Events verwenden andere Namen und den Präfix `Moz` anstelle der vom W3C spezifizierten Namen. Für das Event `touchcancel` der W3C-Spezifikation existiert in dem Mozilla-Touch-Event-API kein Pendant. Das Attribut für die Identifikationsnummer einer Berührung wird bei Mozilla mit `streamId` und beim W3C `identifier` bezeichnet (Brubeck, 2012; Brubeck u. a., 2011).

Mozilla hat die Arbeit an diesem API mit Firefox 6 eingestellt und den Status auf überholt (*deprecated*) gesetzt. In der Android-Version von Firefox werden ab dieser Version die vom W3C spezifizierten Touchevents unterstützt (Brubeck, 2012). Die zum Zeitpunkt der Erstellung dieser Arbeit aktuelle Firefox-Version 13.0.1 sowie der aktuelle Nightly Build 16.0a1 für Windows 7 unterstützen diese Events jedoch noch nicht.

Microsoft setzt mit der neuen Version seines Browsers Internet Explorer 10 verstärkt auf Multi-Touch und Gestenunterstützung[5]. Dafür hat Microsoft ein proprietäres API entwickelt, das Interaktionen des Benutzers über ein Eingabegerät abstrahiert und zu sogenannten Pointern zusammenfasst. Je nach verwendetem Eingabegerät entspricht ein Pointer entweder einer Fingerberührung, einem Eingabestift oder einfach einem Mauszeiger (Rossi, 2011).

Die Verwendung dieses APIs unterscheidet sich allerdings maßgeblich von den bereits beschriebenen sowohl in der Bezeichnung der Events als auch der verfügbaren Attribute.

Frameworks wie Hummer.js[6] (basierend auf Hammer.js[7]) und jQMultiTouch[8] gleichen diese Unterschiede der Touchevent-APIs aus. Zudem bieten beide Gestenerkennungsmechanismen in unterschiedlicher Komplexität. Während Hummer.js nur die Gesten Rotate, Scale, Tap und Doubletap erkennt, ermöglicht es jQMultiTouch auch, neben den vordefinierten Gesten eigene Gesten zu definieren. Dadurch kann das Framework für eine Vielzahl

5 Derzeit ist Internet Explorer 10 nur in einer Preview-Version mit Windows 8 Release Preview erhältlich.

6 https://bitbucket.org/mgrubinger/hummer

7 http://eightmedia.github.com/hammer.js/

8 http://dev.globis.ethz.ch/jqmultitouch/

von unterschiedlichen Gesten-Interfaces verwendet werden (Eight Media, o.J.; Nebeling & Norrie, 2012).

3 Umsetzung eines Prototyps

Ausgehend von einer an der Fachhochschule St. Pölten umgesetzten Flash-Applikation, wurde ein Teil ebendieser unter Verwendung von Webtechnologien und dem Framework jQMultiTouch für den Browser adaptiert. Der Prototyp stellt ein einfaches Puzzlespiel dar, das von bis zu vier Personen gleichzeitig mit Multi-Touch-Gesten bedient werden kann (siehe Abb. 1). Ist das Puzzle gelöst, wird eine kurze Animation dargestellt, um die Spieler/innen über das erreichte Spielende zu informieren. Alle Spiel- und Interface-Elemente wurden mit HTML definiert; die visuelle Erscheinung wurde mit CSS festgelegt. Die Interaktionsmechanismen wurden mit JavaScript unter Verwendung von jQuery[9] und jQMultiTouch implementiert. Zudem musste jQMultiTouch um einige Funktionen erweitert werden, um den es für die Bedürfnisse dieses Prototyps anzupassen. Die Animation am Ende des Spiels wurde mit dem Animationsframework GreenSock Animation Platform[10] entwickelt.

Abb. 1 Startansicht des entwickelten Puzzlespiels

9 http://jquery.com/

10 http://www.greensock.com/v12/

Im Test des Prototyps mit vier Testpersonen wurde diese Implementierung in den Browser Mozilla Firefox 14 und Google Chrome 20 getestet und mit der bereits entwickelten Flash-Applikation verglichen. Dabei konnten Einschränkungen hinsichtlich Performance in beiden Browser festgestellt werden, vor allem bei der gleichzeitigen Interaktion mehrerer Benutzer/innen. Zudem ist die Anzahl der simultanen Touchpoints in der derzeitigen Version von Google Chrome (20) auf sieben beschränkt. Im Vergleich mit der Flash-Applikation wirkte die Interaktion mit der Browservariante auf die BenutzerInnen weniger flüssig.

4 Conclusio

In dieser Arbeit wurden Möglichkeiten untersucht, Multi-Touch-Applikationen für die Ausführung im Browser eines Tabletop-PCs zu entwickeln. Der Webbrowser als Plattform für Multi-Touch-Anwendungen bietet Vorteile gegenüber Desktopsoftware, da Webapplikationen eine hohe Geräte- und Plattformunabhängigkeit aufweisen, einfache Distribution ermöglicht und die Fähigkeiten der Webbrowser rapide verbessert werden.

Die Unterstützung von Touchevents in Desktopbrowser ist derzeit noch nicht sehr weit fortgeschritten. Während Mozilla Firefox bereits mit Version 3.6 (aktuell: 14) ein proprietäres API für die Verwendung von Touchevents einführte, bietet Google Chrome erst seit der aktuellen Version 20 experimentelle Unterstützung von Touchevents über ein vom W3C standardisiertes API. Microsoft kündigte für den in Windows 8 enthaltenen Internet Explorer 10 wiederum ein eigenes API zur Verwendung von Toucheingabe an.

Unsere Untersuchungen in dieser Arbeit zeigen, dass der Browser für einfache, gestenbasierte und animationsunterstützte Applikationen bereits jetzt eine Alternative zu Desktopanwendungen darstellt. Als größte Hindernisse für die Erstellung komplexerer Anwendungen konnte derzeit die Leistungsfähigkeit der Browser und die daraus resultierende mangelhafte User Experience identifiziert werden. Aber auch diese Einschränkung wird durch zukünftige Verbesserungen der Browser von geringerer Bedeutung sein.

Literaturverzeichnis

Brubeck, M. (2012). Touch events (Mozilla experimental) – MDN. Abgerufen Juli 2, 2012, von https://developer.mozilla.org/en/DOM/Touch_events_(Mozilla_experimental)

Brubeck, M., Moon, S. & Schepers, D. (2011). Touch Events version 1. Abgerufen Juni 29, 2012, von www.w3.org/TR/touch-events/

Eight Media (Hg.) (o. J.). Hammer.js – A javascript library for multi touch gestures. Abgerufen Juli 5, 2012, von http://eightmedia.github.com/hammer.js/

Flanagan, D. (2006). *JavaScript : the definitive guide*. Sebastopol, CA: O'Reilly.

Mozilla Foundation (Hg.) (2011). Mozilla Firefox 4 Release Notes. Abgerufen Juli 3, 2012, von www.mozilla.org/en-US/firefox/4.0/releasenotes/

Nebeling, M. & Norrie, M. C. (2012). jQMultiTouch: Lightweight Toolkit and Development Framework for Multi-touch/Multi-device Web Interfaces. Gehalten auf der *ACM EICS'12*, Copenhagen, Denmark.

Pilgrim, M. (2010). *HTML5 : up and running*. Sebastopol, CA: O'Reilly.

Rossi, J. (2011). Touch Input for IE10 and Metro style Apps. *IEBlog – MSDN Blogs*. Abgerufen Juni 4, 2012, von http://blogs.msdn.com/b/ie/archive/2011/09/20/touch-input-for-ie10-and-metro-style-apps.aspx