MT4j – An open source platform for multi-touch software development

Fraunhofer Institute for Industrial Engineering (IAO) Stuttgart, Germany

Uwe Laufs

Christopher Ruff

Anette Weisbecker

Multi-touch has been one of the hot topics within the areas of human computer interaction and computer science for the last few years. Although there is a lot of activity in this area, the development of multi-touch applications still suffers from barriers and missing support on the software side.

This article describes current challenges in multi-touch software engineering and the open source platform Multi-Touch for Java (MT4j). MT4j is designed for rapid development of graphically rich applications on a variety of common PC hardware and operating systems. The platform has a special focus on reducing existing barriers with regard to multi-touch software development and on making multi-touch software development easier and more efficient. Detailed information about the MT4j platform can be found at http://www.MT4j.org.

Keywords: multi-touch, application development, software engineering, frameworks, software architecture, gesture control

1 Introduction

Multi-touch technology has received considerable attention in recent years. Even though the technology has existed since the early 80's, the current hype regarding the multi-touch interaction started no earlier than 2005, when an article about low-cost construction of multi-touch screens [Han 05] made multi-touch technology more popular and available for a bigger community. In 2006, a video [Han 06] showing multi-touch interaction experiments and potentials of the technology was hyped across the internet and also received a lot of attention from the scientific community.

In the meantime, products with multi-touch capabilities are available, for instance Microsoft's interactive table Surface PC [Microsoft 2009]. It is also likely that Microsoft's support for multi-touch in Windows 7 will further increase the relevance of the technology [Spath et al. 2009].

Although currently there is a lot of activity in research, industry and in the open source community, there are still barriers on the technical side regarding the development of multi-touch applications in practice.

Multi-touch for Java (MT4j) is an open source multi-touch application development platform which is released under the GPL license and can be freely used by anyone. The platform provides high level functionality and aims at providing a toolkit for easier and faster development of multi-touch applications.

Section 2 of this article describes existing challenges and requirements regarding required technical support that exists in multi-touch software development. Section 3 gives a detailed overview about how the MT4j platform addresses these requirements and describes the architecture and the design of MT4j. Section 4 shows two applications built with MT4j and in section 5 we conclude this article with a short summary and the further prospects in this area.

2 Challenges and requirements addressed by MT4j

Multi-touch applications running on standard PC hardware differ from traditional PC applications in many ways. Using multi-touch for the control of PC applications means that many existing applications have to be modified or to be completely re-implemented because of the existing differences in user interface interaction and also because of barriers regarding technical aspects on the software engineering side. [Rahimi 2008].

In order to reduce implementation efforts during multi-touch software development, there are several requirements and challenges which can be addressed by a development platform in a generic and reusable way.

2.1 Multiple input motions

Traditional user interfaces commonly use input devices like a mouse and a keyboard. Many of these applications are based on a windowing system and make use of a windowing toolkit. Existing windowing toolkits already provide mechanisms for handling user interaction and provide a wide range of UI components and widgets which can be composed to complex user interfaces. Unlike a mouse, a multi-touch screen can provide more than one input motion at the same time. A mouse can provide additional events (wheel, buttons) instead, which cannot be directly produced using a multi-touch screen. Having multiple input motions causes problems regarding the use of windowing toolkits because these toolkits normally support a single pointer device, only.

2.2 Multi-point gestures

Most multi-touch applications intensively use multi-point gestures like zooming the screen content or rotating objects with two fingers. Multi-point gestures are gestures that include more than one input motion. The realisation

of such gestures also requires the capability to process multiple input motions on the software side. Because different applications may require additional gestures or use gestures differently than expected, gesture recognition and gesture processing has to be flexible and extensible.

2.3 Combination with additional input devices

Multi-touch technology is often combined with different additional input devices. For example, interactive tables like Microsoft's Surface PC or the Reactable [Reactable 2009] also process input from optical tracking systems like the reacTIVision engine [Bencina et al. 2005]. Other systems combine multi-touch for example with digital pens [Ruff et al. 2008], [Leitner et al. 2009] or for example provide foot interaction functionality [Pakkanen and Raisamo 2004]. During software development, it can also be an advantage to be able to use traditional input devices like mice. This means that development and basic testing can take place using standard PC technology without multi-touch input devices. Also, multi-point applications beyond multi-touch (e.g. multi-point applications with multiple mice) can be supported this way. Having multiple different input devices means that there has to be a way to use a wide range of different input devices simultaneously. Also, it seems necessary to be able to extend input device support with additional devices and provide an extensible mechanism to realize this.

2.4 Performance requirements

Performance issues are also a relevant point regarding the development of multi-touch applications. Multi-touch applications often have a special focus on user experience and graphically rich user interfaces. Also, unlike mostly static user interfaces, the usage of gestures like the zoom gesture on fullscreen content requires a complete redraw of the whole screen. In such cases, common optimisation strategies, like updating only changed parts of the user interface can often not be applied. To achieve the required frame rates for smooth and direct user interface interaction, in many cases, there is a necessity for high graphics rendering performance. In other areas of software development, high performance graphics rendering requirements (for example 3D CAD tool or game development) are addressed by using standards like OpenGL [OpenGL 2009] which provide high graphics performance based on hardware accelerated rendering.

3 MT4j platform architecture and design

The functionality of the MT4j platform architecture is subdivided into different layers communicating through events sent from one layer to the next. The emphasis on input layers represents the importance of a flexible input architecture. The different layers are described in the following text.

3.1 Input hardware and hardware abstraction

By using a hardware abstraction layer, MT4j can support various input hardware with only minimal adjustments in the input hardware abstraction layer. In this abstraction layer, the different raw input data is converted into uni-

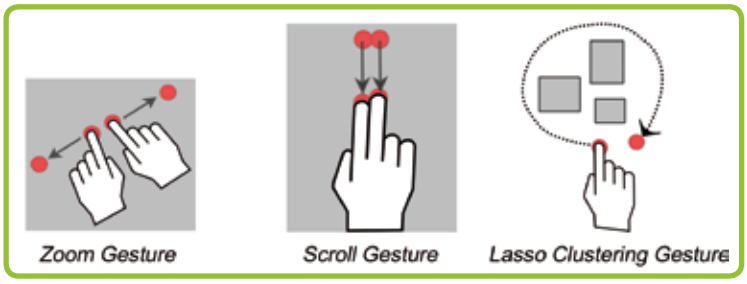


Figure 1:
Common multi-touch gestures for background interaction

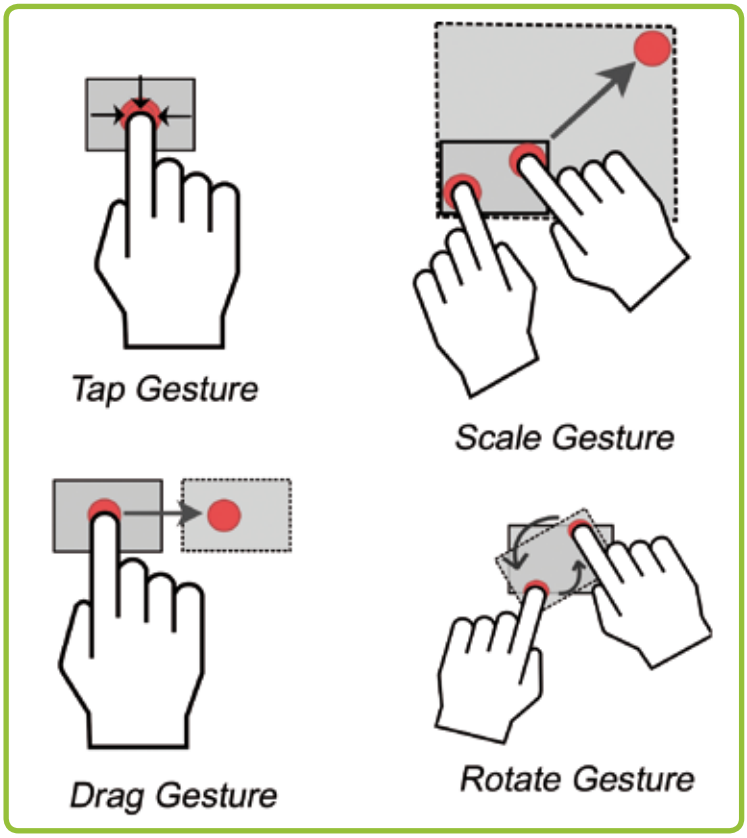
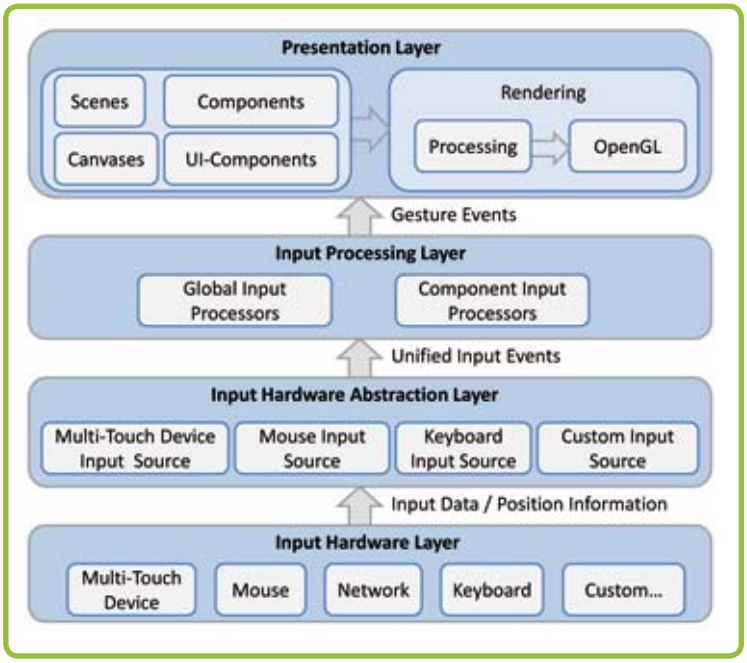


Figure 2:
Common multi-touch gestures for object interaction

Figure 3:
MT4j architecture overview



fied input events. The only step to be taken in order to support a new type of input hardware is to extend the abstract super class of all the input sources and add the functionality specific to this type of input. By providing this easy input extensibility, MT4j lends itself to many different applications and especially to areas where new kinds of human computer interaction methods and techniques are explored and applied.

MT4j comes with a set of implemented input providers including mouse, keyboard and multi-touch input protocols such as WM_TOUCH (the native interface for multi-touch hardware in Windows 7) and the TUIO protocol [Kaltenbrunner et. al. 2005] which has gained the status of a standard protocol especially among open source multi-touch solutions. Additionally, MT4j supports the use of multiple mice input on windows platforms, which facilitates testing of multi-touch functionality even without multi-touch capable hardware available. All of these input sources can be used synchronously and in combination without the risk of non-deterministic behaviour.

3.2 Input Processing

The aspects of processing, analyzing and interpreting user input are very important for a platform focused on multi-modal input. In MT4j, the input processing occurs at two different stages in the input event flow. The first stage is the global input processing stage where a number of input processors can be registered which subsequently listen directly to the input of the various input sources. This stage is used when all input has to be processed. It also allows modification of user input before it is passed up to the next layers. For example, every newly created scene in MT4j (see 3.3) automatically registers a global input processor which checks if there is a component at the position of the input and sets that component as the target of that input event.

The second input processing stage is located at the component level. It allows processing of input that was targeted at one component only. Here, multi-touch gestures like the rotate and scale gestures can be found. These component input processors can be registered modularly with any component allowing for a pluggable behaviour changeable at runtime. As the field of multi-touch interaction and gestures is currently an active field of research, it is understood that both global and component input processors have to be extensible and customizable. If the criteria for a multi-touch gesture are met, the input processor fires a gesture event carrying the information about the recognized gesture to the corresponding component which passes the event on to its gesture listeners.

The action taken when a gesture event is received is determined by the attached gesture listeners which can modify the components behaviour or appearance. For easy usage, the platform already includes default implementations of such listeners to react to the included gestures. The use of the observer pattern [Gamma 95] in this way is a familiar concept to most Java programmers and is widely used in Java windowing toolkits.

Figure 4 shows the event flow from the input hardware abstraction layer to the input processing layer and finally to the presentation layer (from left to right). First, input events are produced by the underlying hardware. An input source listens to the events produced by the hardware (e.g. a multi-touch screen that sends motion events via the TUIO protocol). Based on these events, unified input events are produced and passed to the input processors on the input processing layer. These events are used to provide input specific functionality (e.g. a cursor is shown at every position where a multi-touch screen is currently touched) by global input processors. Component input processors translate the input events to higher level gesture events. For example, a rotate processor recognizes when a user rotates at least two fingers on top of an object and produces a rotate gesture event. Finally, the target components (user interface components on the presentation layer) listen to higher level gesture events and react by performing the desired action (e.g. a rotate event is used by the default

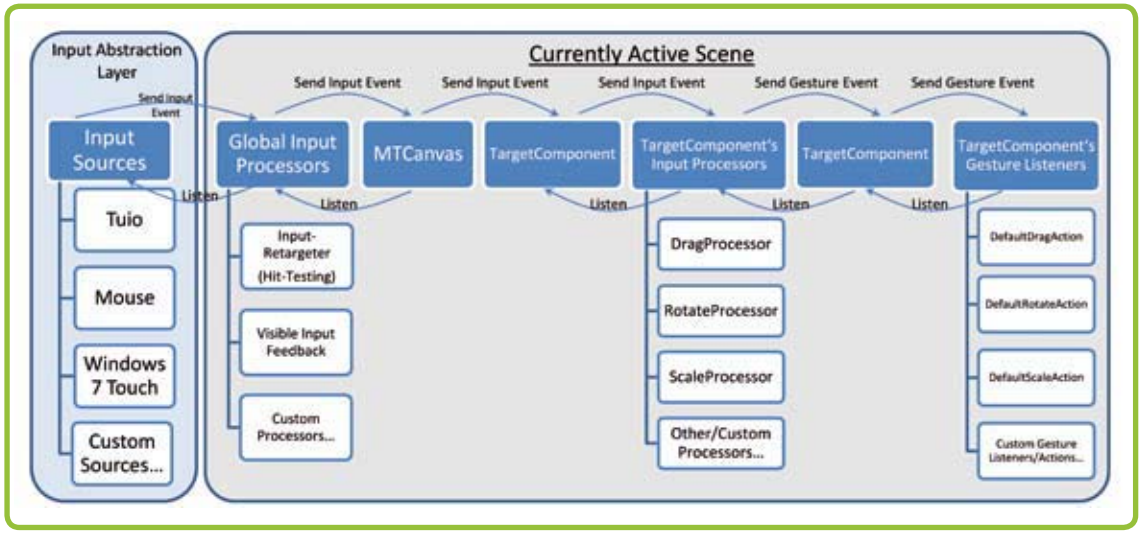


Figure 4:
Input event propagation in MT4j

rotate action to rotate the target object in the user interface).

3.3 Presentation

The presentation layer is a vital part of any application using a graphical user interface. Multi-touch applications are often created for specialised use cases that differ a lot from everyday office applications. Often, one of the requirements is the creation of an exciting and eye-catching user experience. This implicates, that the user interface elements provided by conventional GUI toolkits are often not appropriate. Instead, MT4j provides a flexible way to create customizable and media rich user interfaces. For this purpose, MT4j contains graphical components ranging from graphical primitives to more complex user interface widgets and is inherently designed to allow development of 2D and 3D applications.

In order to organize different aspects of an MT4j application, the concept of “scenes” was introduced. Scenes encapsulate and cleanly separate the input processing and presentation of one aspect of an application from another. An example of using scenes would be a game that has a menu scene and a game play scene which contain different business logic and interfaces. To navigate between scenes, a scene change can be triggered.

Figure 5:
Screenshot: Rendering scalable vector graphics with MT4j

Figure 6:
Real time 3D rendering with MT4j on a multi-touch notebook and on a large multi-touch screen

The root component of every scene in MT4j is the canvas component. It acts as the link between the global input processing layer and the presentation layer. All input events pass through the canvas component, which then further propagates events to their destinations. It also contains methods for checking which components are located at a specified screen position and it is responsible for recursively drawing the canvas with all its child elements.

In MT4j, graphical user interfaces are based on a hierarchic component structure which allows the composition of user interface components in a tree structure. Even complex composite components can interact with user gestures as a whole. This means that for example a rotate gesture can be applied to a component including all children within its component hierarchy.

There are two general types of components in MT4j. Invisible

components provide basic functionality like the composition of components. Visible components can directly interact with the user input. Included visible components are primitive shape components (e.g. rectangle, polygon, ellipse and line) as well as a set of more complex user interface components. More complex user interface components are often based on primitive shapes and provide functionality like image rendering with support for common image file formats, rendering of scalable vector graphics [SVG 2009] or rendering of 3D models. Also, components like buttons or vector based text rendering components are included in MT4j.

The component based structure of user interfaces in MT4j is extensible. When creating a custom user interface component, a lot of functionality can be reused from available components. Custom components can be built upon already available functionality (e.g. hit detection, gesture processing) by composing primitive shapes and available user interface components to more complex components. An example of such a higher level component is the multi-touch keyboard, which is composed of a wide range of MT4j user interface components.

For rendering of MT4j components, the processing [Processing 2009] toolkit is used. Processing is an open source Java toolkit aimed at the creation of data visualizations, interactions and computational art and has a very active community. It provides an easy syntax for accessing drawing and visualization functionality and contains many useful utilities. By using a rendering abstraction layer, it is possible to choose between different renderers. Software- and hardware accelerated renderers are available [JOGL 2009]. A disadvantage of the rendering abstraction is that not all available hardware accelerated functionality and features are exposed by the rendering API, which can prevent the use of optimization techniques like Vertex Buffer Objects or display lists. The underlying OpenGL context can also be accessed and used directly when necessary. When the OpenGL renderer is chosen, most MT4j components use this direct access for speeding up rendering, especially of complex components or 3D models by orders of magnitude. This allows for a very good performance of MT4j applications on newer systems, but also allows falling back to software rendering if run on older hardware.

Figure 7:
Screenshot: MT4j Flickr photo search showcase

Figure 8:
Screenshot: MT4j map navigation showcase

4 Application showcases

The following showcases give an example of how the functionality provided by MT4j can be used to realize multi-touch applications. The applications described in this section (and several others) are available at MT4j.org.

4.1 Flickr photo application showcase

The Flickr photo application is a multi-user multi-touch application. It uses the MT4j keyboard component to allow the users to type in photo search terms. The keyboard is extended by a custom flickr button which triggers a photo search. When the button is pressed, the entered search term is passed over to the integrated Flickr interface [Flickrj 2010] which returns the best matches to the application and starts to download and display the photos. For image sorting and manipulation, MT4j's default gesture processors and gesture actions are used to make photos movable, rotatable, scalable and to provide

zoom functionality for the whole screen content.

4.2 Map navigation showcase

The Map navigation application provides deep-zoom functionality for digital maps. The maps are provided by different external image data providers (e.g. Open Street Maps) via the internet. Additionally, geo-tagged photos from Flickr.com can are visualized on the map an can be opened. Images can be moved and manipulated analogous to the Flickr photo search application using the default MT4j gesture processors and gesture actions.

5 Conclusion and Future Prospects

In this paper we described current challenges in multi-touch software development and showed how the open source platform MT4j addresses these problems. The main focus of this article were architecture and design of the MT4j platform. In two showcases, we demonstrated examples of applications built with MT4j. During the development of the showcases, a lot of functionality provided by MT4j could be reused and implementation efforts were reduced. Currently, we use MT4j as platform for the development of multi-touch applications with a focus on multi-user business applications. This includes applications for collaborative planning and modelling. Additional components developed during these projects will be part of future MT4j releases.

Since the open source release in august 2009, the positive feedback from the community and more than 350.000 hits on MT4j.org until January 2010 show that there is a high demand for a multi-touch application development platform based on Java programming language. For the further development, it is intended to include contributions from the open source community to provide a comprehensive toolkit including a wide range of high level user interface components.

6 References

[1] Bencina, R., Kaltenbrunner, M., Jorda, S.: Improved Topological Fiducial Tracking in the reacTIVision System, Computer Vision and Pattern Recognition - Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference, 2005

[2] Flickrj – Java interface to flickr API, http://flickrj.sourceforge.net, last visited: 10.02.2010

[3] Gamma, E., Helm, R., Johnson, R.E.: Design Patterns. Elements of Reusable Object-Oriented Software, Addison-Wesley, p. 293-305, 1995

[4] Han, J.Y.: Low-cost multi-touch sensing through frustrated total internal reflection. In: UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology, New York, NY, USA, ACM (2005) p. 115–118., 2005

[5] Han, J.Y.: Multi-Touch Interaction Experiments, Video at Youtube; http://www.youtube.com/watch?v=EiS-W9aeG0s; 2006

[6] JOGL Homepage –Java bindings for OpenGL https://jogl.dev.java.net/, last visited: 29.09.2009

[7] Kaltenbrunner, M., Bovermann, T., Bencina, R. Costanza, E. (2005). TUIO: A protocol for table-top tangible user interfaces. 6th International Gesture Workshop, Vannes 2005.

[8] Leitner, J., Powell, J., Brandl, P., Seifried, T., Haller, M., Dorray, B., To, P.: Flux: a tilting multi-touch and pen based surface: Proceedings of the 27th international conference extended abstracts on human factors in computing systems, 2009

[9] OpenGL - The Industry Standard for High Performance Graphics, http://www.opengl.org, last visited: 24.09.2009

[10] Pakkanen, T., and Raisamo, R.: Appropriateness of foot interaction for non-accurate spatial tasks. Conference on human factors in computing systems, pages 1123–1126., 2004

[11] Processing Homepage, http://www.processing.org, last visited: 29.09.2009

[12] Rahimi, M.: Gestenbasierte Computerinteraktion auf Basis von Multitouch-Technologie, Hamburg University of applied Sciences, 2008

[13] Reactable Homepage, http://www.reactable.com, last visited: 24.09.2009

[14] Ruff, C.; Laufs, U.; Korell, M.: Unterstützung von Problemlösungsprozessen mit Multi-Touch-Technologie und digitalen Stiften, Tagungsband: Stuttgarter Softwaretechnik Forum, 2008

[15] Scalable Vector Graphics, http://www.w3.org/Graphics/SVG, last visited: 24.09.2009

[16] Spath, D., Weisbecker, A. (eds.), Laufs, U.; Block, M., Link, J., Ardilio, A., Schuller, A., Bierkandt, J., Studie Multi-Touch - Technologie, Hard-/Software und deren Anwendungsszenarien, http://www.swm.iao.fhg.de/Publikationen/mt2009.jsp, 2009, last visited: 15.02.2010

[17] Microsoft Homepage http://www.microsoft.com/surface, last visited: 24.09.2009

Dipl.-Ing.(FH) Uwe Laufs M.Comp.Sc.
Uwe.laufs@iao.fraunhofer.de

Uwe Laufs studied Media and Computer Science in Stuttgart and studied Computer Science at FernUniversität Hagen. Now, he is a researcher at the Fraunhofer Institute for Industrial Engineering IAO in Stuttgart, Germany. Uwe Laufs has been involved in several research projects as well as projects directly funded by industrial customers. His areas of expertise and research interests include Software Engineering and Semantic Web Technologies as well as Software Management and IT Strategy.

Christopher Ruff B.Sc.
Christopher.Ruff@iao.fraunhofer.de

Christopher Ruff studied Media and Computer Science at Hochschule der Medien Stuttgart and is now working at the Fraunhofer Institute for Industrial Engineering IAO in Stuttgart, Germany. Christopher Ruff is lead developer of the MT4j platform. His areas of expertise and research interests include Software Engineering with a special focus on Multi-Touch Software Engineering.

Priv.-Doz. Dr.-Ing. habil. Anette Weisbecker
anette.weisbecker@iao.fraunhofer.de

Dr.-Ing. Anette Weisbecker is Assistant Director of the Fraunhofer Institute for Industrial Engineering IAO and heads the Competence Center Software Management, which supports companies by the design and implementation of complex software and services. Her focus of research is Software Management, Electronic Business and Grid Computing.