Taxonomy and Overview of Multi-touch Frameworks: Architecture, Scope and Features

Dietrich Kammer, Mandy Keck
Technische Universität Dresden
dietrich.kammer@tu-dresden.de

Georg Freitag, Markus Wacker
Hochschule für Technik und Wirtschaft Dresden
freitag@htw-dresden.de

ABSTRACT

Multi-touch hardware has only recently entered the mainstream of information technology. Interaction designers, product developers and users have been influenced by this development. In order to build multi-touch applications, programmers have created various reusable frameworks. To obtain an overview of this diversity, we present a list of criteria to classify, evaluate, and select multi-touch frameworks. Our main contribution consists in a taxonomy that we have elaborated out of a vast list of existing projects, of which we present nine freely available ones here. To promote the development of multi-touch frameworks, our list of criteria reveals four main areas for future work.

Keywords

Software engineering, frameworks, multi-touch, natural user interface, gesture-based interfaces

INTRODUCTION

A multitude of frameworks related to multi-touch has emerged in the last years, most of which are still under development and might not prove stable. In addition, considerable restrictions with regard to flexibility exist when designing multi-touch enabled applications. By examining current architectures, it is possible to observe recurring solutions that have proven to be effective. Moreover, unique approaches can be identified and evaluated. This research was our starting point for a top-down approach towards a taxonomy of multi-touch frameworks. With the help of our scheme, new requirements arising from a designer's point of view can be easily integrated. This classification leads to a common understanding and an easy comparison of new emerging frameworks.

We consider software libraries that support the programmer in implementing fully functional multi-touch applications. The frameworks presented offer diverse components and tools and have reached a certain level of maturity. While outdated projects have been disregarded, unique approaches are covered, even if they have received little attention until now.

LIST OF CRITERIA

The classification of multi-touch frameworks stem from the different capabilities of supported hardware devices and concrete use cases that are being addressed. Across our list of criteria (Figure 1), frameworks can have different focuses and offer programming support of varying degrees.

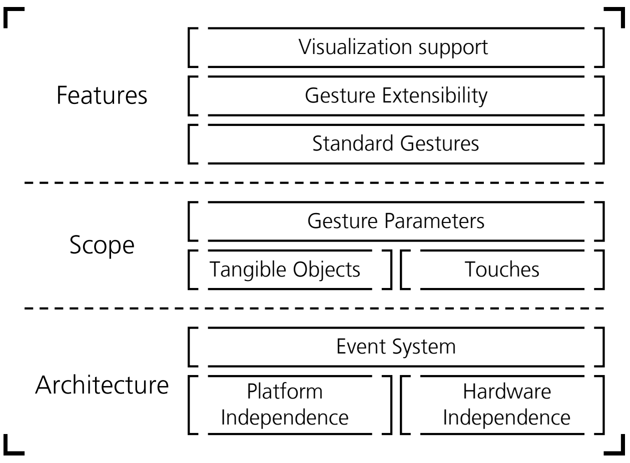

Figure 1: Criteria for multi-touch frameworks.

The lower level of the section *Architecture* in our list of criteria is concerned with the applicability of frameworks. Independence of specific hardware setups or operating systems is a key characteristic for distributing applications across multiple platforms and devices. We consider the event system as the bridge between the client application and the framework. The modality of information exchange is a key concern to the programmer.

One main scope of a framework is the recognition of touches and tangible objects. While some frameworks implement both, they are mostly focused on one aspect. Since the concern of this contribution is multi-touch, purely single touch frameworks (like iGesture [5]) are not considered. Looking at frameworks from a programmer's perspective, there are various parameters that can be extracted from touch-interaction. Likewise, there are different degrees of freedom when dealing with tangible objects. Either specific fiducial markers need to be applied, or arbitrary shapes can be recognized.

In addition to architectural concerns and the scope of a framework, specific features need to be addressed. Support of standard gestures is an important factor when choosing a multi-touch framework. We consider online gestures, which can also be regarded as direct manipulations like scaling and rotating. In contrast, offline gestures are usually processed after the interaction is finished; e. g. a circle is drawn to activate a context menu. A similar distinction in gestures and manipulations is made by [4]. Gesture extensibility is another important aspect. Most frameworks provide the programmer with direct access to the raw touch data. Moreover, further support is conceivable by providing gesture recognition facilities or by means of a gesture abstraction [10]. By identifying properties of gestures, e. g. motion vectors, speed, or pressure, programmers can create new gestures with more ease, or combine existing standard ones. Finally, frameworks can be judged by their visualization support. Depending on the application context, 2D or 3D graphics should be available. In any case, the availability of standard widgets that are multi-touch enabled is crucial. Again, the extension of these widgets should be flexible and straightforward. This gives a greater freedom to the designers and developers of novel multi-touch interfaces.

REVIEW OF FRAMEWORKS

The review of existing frameworks is organized according to the overall architecture, the scope, and the provided features as discussed in the previous section. We seek to identify categories for each criterion and discuss the properties of each category. In this comparison we consider *Sparsh-UI* [14] and *MT4j* [3] which are realized in Java, as well as the *Surface SDK* [12] and *Breezemultitouch* [13] as Windows Presentation Foundation (WPF) frameworks. *Miria* [1] is based on WPF as well, but is focused on web development with Silverlight. In addition, *GestureWorks* [7] is based on Flash, *PyMT* [6] on Python, *Grafiti* [14] is written in C# and *libTISCH* [2] in C++.

Architecture

Across the available frameworks, commonalities with respect to the underlying software architecture can be identified.

Platform independence

Most frameworks support various hardware setups by working with the TUIO protocol. It is a network protocol based on UDP which supplies information about touch events and tangible objects [9]. The notable exception is the Surface SDK which supports Windows 7 multi-touch events and specific drivers for Windows Vista. Both, TUIO and Windows 7 events, are natively supported by Miria and PyMT. Sparsh-UI, MT4j, libTISCH, and GestureWorks offer the possibility to implement appropriate drivers for arbitrary hardware devices and inputs.

Independence of operating systems can be achieved using cross-platform runtime environments like Java, Flash, or Silverlight. MT4j and Sparsh-UI rely on Java's virtual machines, whereas GestureWorks and Miria, respectively, are based on the last two. Available for all major operating systems is a platform specific interpreter for Python which runs applications based on PyMT. Surface SDK and Breezemultitouch target the Microsoft® Windows platform by relying on WPF. Grafiti and libTISCH are realized in C# and C++, respectively. Thus, applications can be cross-compiled to provide executable code for different platforms.

Programmers are usually constrained to the programming language used by the framework. Exceptions are Sparsh-UI and libTISCH. Due to the particularities of their event system, applications can be implemented in the programming language of choice.

Event system

Most frameworks rely on the event paradigm supported by their host language. Events are being raised by gesture recognizers and processed by the registered listener methods. Exceptions are Sparsh-UI and Libtisch: Sparsh-UI provides a gesture server, which achieves a loose coupling between gesture processing and the application. Events are transported via a network protocol, which makes interfaces to various clients possible. Clients can be implemented in any programming language that allows network communication. Similarly, libTISCH provides a gesture recognition layer which communicates via a network protocol as well.

Another distinction can be made with regard to the information contained in a gesture event. PyMT, Breezemultitouch, Miria, and MT4j route touch data to components of the application, where gesture recognition takes place. Sparsh-UI, libTISCH, GestureWorks, and Grafiti process this data within the framework and aim to provide information about performed gestures only. Thus, gesture recognition is either centralized or decentralized. In the centralized scenario, a gesture registry is provided and an abstraction of the user interface (UI) is required. The registry queries the application about its visual components in order to associate gestures with concrete elements of the UI. Only the position and dimension of these elements are important for the abstraction.

Scope

The support for tangible objects and touch information differs across the frameworks and depends on the architecture decisions described above.

Tangible objects

As discussed in the section *Architecture*, most frameworks employ the TUIO protocol, which inherently supports tangible objects. Their ID, position, and rotation are reported to applications. However, among the frameworks in discussion, only Grafiti and Surface SDK actually focus tangible objects. Mostly, tangibles are recognized based on fiducial markers (tags) that are attached on a flat area of the object. In addition, the Surface SDK is the first framework to show progress towards the recognition of arbitrary objects without the use of markers. Moreover, this SDK can identify a large number of distinct objects, as it supports two variants of markers. Breezemultitouch is completely oblivious of tangibles.

Touches

The TUIO protocol supplies IDs of touches and a history of coordinates. Additionally, Breezemultitouch and libTISCH calculate the velocity, and MT4j provides a direction vector of the touch. Depending on the hardware, some frameworks provide more information. For instance, libTISCH supplies the shape of a blob based on custom shadow tracking hardware. Similar to the Surface SDK, the orientation of a finger is provided as well.

By default, Sparsh-UI supplies parameters of gestures instead of touches. This is due to its centralized gesture recognition. In this case, provided parameters depend on the triggered gesture.

As a notable exception, Grafiti maintains target lists of touches. Elements of the UI and tangible objects that are close to a touch during its movement are collected in a target list. This can help the application programmer to process the impact of a performed gesture on all the relevant elements.

Features

Basic features of frameworks are the support of standard gestures as well as options of implementing new ones. In addition, the user is assisted to varying degrees in creating a UI.

Standard gestures

All frameworks support online processing of gestures. The classic ones like scaling, rotation, and translation of objects are possible. Only PyMT explicitly supports offline gestures. Other frameworks use the online processing facilities to make offline gestures possible. Simple gestures like Tap, Double Tap, or Flick are included in each framework. Grafiti and MT4j additionally implement a Lasso gesture to select multiple objects.

Gestures are mainly processed depending on components of the UI. In addition, Grafiti and MT4j introduce the notion of global gestures. They are relevant throughout the application and may take precedence over component based gestures. For instance, Grafiti provides an appropriate configuration variable to prioritize global gestures [14].

Gesture extensibility

In order to implement new gestures, interfaces are provided to establish a common infrastructure. New gestures result from extending these interfaces and implementing appropriate algorithms. The Surface SDK only provides access to raw touch data. Depending on the architecture, gestures are instantiated by the central gesture registry or by each component. Responsibility for analyzing input is thus delegated to the components or remains within the framework.

In contrast to other frameworks, libTISCH implements an abstraction of the properties of a gesture. As a result, it is possible to select the information which is included in a gesture event, for instance the velocity and amount of touches associated with this gesture.

Visualization support

Multi-touch frameworks must either provide a set of visual components that are extensible, or allow the creation and integration of new components. Miria and Surface SDK build upon the infrastructure of WPF and provide multi-touch enabled controls. In contrast, Breezemultitouch implements wrapper classes which route multi-touch input to existing WPF controls. GestureWorks extends basic Flash containers to process multi-touch input. MT4j and PyMT provide custom components based on OpenGL. Sparsh-UI, libTISCH, and Grafiti explicitly support custom widget libraries by means of their UI abstraction.

Visualization support for both 2D and 3D graphics is available through DirectX for WPF based frameworks, or OpenGL. Flash applications realized with GestureWorks rely on external libraries for 3D support.

DISCUSSION

In order to discuss the presented frameworks and their specific properties, we propose two points of view: the product developer and the interaction designer which have different requirements. With the help of the criteria list, the different approaches will be analyzed and matched to the appropriate solutions presented in the frameworks discussed.

Product developer

Product development teams require the rapid creation of stable applications. A solid foundation like WPF is recommended, as it has a strong reputation in industry. In addition, a large community is devoted to the development of WPF applications. Especially with a product line in mind, the portability of its large widget base across the Windows platform is an advantage. If the limitation to Windows is not acceptable, cross-platform solutions need to be considered.

We find the concept of a gesture server beneficial to product developers, although it has only been prototypically implemented by more research focused frameworks like Sparsh-UI and libTISCH. Gesture servers provide events to many clients, are exchangeable and developed independently of their clients. However, servers require a more complex architecture and introduce some communication delay.

Since gestures are encapsulated, feedback and feed-forward is not easily customizable by clients. In the future, functionality of the gesture server can migrate into the operating system, minimizing delays as well as providing a uniform interface.

Interaction designer

We consider an interaction designer with a strong focus on prototyping. The freedom to create visuals can be hampered by a fixed set of controls. This is the case with WPF, although it allows customization through design tools. Frameworks like PyMT and MT4j offer a greater freedom.

cross-platform single platform	☼ ○	Architecture				Scope			Features		
integrated library gesture server	<> ><	TUIO	Win7	Device Adapter	Gesture Events	Tangibles	Touch Params	Gesture Params	Standard Gestures	Gesture Extensibility	Visualization support
MT4j	<☼>	✓	-	✓	decentral	supported	✓	✓	online	super class	custom widgets
SparshUI	>☼<	✓	-	✓	central	supported	-	✓	online	super class	-
Surface SDK	<○>	-	✓	-	decentral	focus	✓	✓	online	raw data	WPF multitouch controls
Breezemultitouch	<○>	✓	-	-	decentral	not supported	✓	-	online	wrapper class	WPF multitouch controls
Miria	<☼>	✓	✓	✓	decentral	not supported	✓	✓	online	raw data and recognition support	WPF multitouch controls
Grafiti	<☼>	✓	-	-	central	focus	✓	✓	online	super class	-
libTISCH	>☼<	✓	-	✓	central	supported	-	✓	online	super class	custom widgets
PyMT	<☼>	✓	✓	-	decentral	supported	✓	-	online offline	raw data and recognition support	custom widgets
GestureWorks	<☼>	✓	-	-	central	not supported	-	✓	online	super class	custom widgets

Table 1: Comparison of frameworks according to the list of criteria

The decentralized approach to gesture recognition found in most frameworks allows complete freedom to design new gestures. This is due to the full control of touch events. At the stage of prototyping, the necessary information contained in a gesture event is not fixed. The effort to define and implement specific gesture events can be avoided.

While full control over touch events is beneficial, the detail of their parameters should not be excessive. As a common base, all frameworks in discussion provide a history of the coordinates of a touch contact along with an ID. Some frameworks add direction or velocity to the touch event. The Surface SDK collects data on the blob-size and orientation of a finger as well. We argue that recognition support should be available to interaction designers, but made optional. Instead of pushing full information to clients, it should be possible to pull more information about current touch events from the framework. The notion of gesture building blocks introduced in libTISCH shows how application programmers can select important parameters of a gesture for the appropriate event.

CONCLUSIONS AND FUTURE WORK

The list of criteria presented in this contribution has been created from observing current frameworks and analyzing the requirements of software developers. Table 1 gives an overview of the considered frameworks by means of our criteria. For future work, we identify four main focuses.

Offline gestures

Only PyMT has an explicit support for offline gestures. Ongoing manipulations are the main concern of other multi-touch frameworks. Amazingly enough, older frameworks like iGesture [5] that are focused on single-touch environments, offer an extensive support for offline gestures. The definition, storage, and recognition of new gestures are aided by the graphical iGesture tool.

Gesture extensibility

Gesture extensibility is another important concern. A common abstraction for multi-touch gestures could ease the definition of new gestures [11]. So far, developers are limited to standard gestures and processing of raw input data. In addition, many frameworks offer interfaces that can be extended to create and register new gestures. Recognition support is offered by Grafiti with a target list for each touch. Alternatively, PyMT includes tools to extract basic geometric features. These approaches are first steps towards a comprehensive gesture recognition support.

Integration of devices

We see a potential in combining different input devices by finding a way to unify their inputs. For instance, libTISCH and MT4j introduce unified input events. In this context, TUIO as the de-facto standard for touch devices has to be addressed. It is supported by all open source frameworks we considered. On the other hand, Windows 7 gesture events constitute an industry standard which has to be dealt with as well. It has to be noted that the new TUIO 2.0 standard is intended to cover a greater array of devices and interactive surfaces [8]. One existing project that aims to integrate various devices is the Squidy library [10].

Gesture server

As mentioned in the discussion, the idea of gesture servers can be implemented as a service of the operating system. Consistent visualizations of feedback and feed-forward throughout all applications are one potential of this approach [16]. More importantly, performance benefits and stable applications are to be anticipated. As discussed previously, detailed information about touches should be made available by the intended service when needed.

ACKNOWLEDGMENTS

This work has been partly supported by the European Union and the Free State Saxony through the European Regional Development Fund (ERDF) and the Ministerium für Wissenschaft und Kunst through the project Gestenbasierte Interfaces. We thank the students Dana Henkens and Stefan Wagner who participated in this research.

BIOGRAPHIES

Dietrich Kammer is a research associate and doctoral candidate at the Technische Universität Dresden. His interests include component oriented software engineering and computer graphics. Currently his work is centered on multi-touch technology. A key interest is the formalization of gestural interaction for use in appropriate frameworks.

Georg Freitag is a research associate and doctoral candidate at the University of Applied Sciences in Dresden. His interests are novel human-computer interfaces, interaction design and information visualization. At present, the focus of his work is on programming environments and frameworks for multi-touch application development.

Mandy Keck is a research assistant at the Technische Universität Dresden. Her interests are information visualization and the design of tangible user interfaces. The focus of her work is currently on interaction concepts for multi-touch devices and the evaluation of appropriate visualizations.

Markus Wacker is professor for computer graphics at the University of Applied Sciences in Dresden. His research interests are VR/AR applications, multi-touch development, and gesture based interfaces.

REFERENCES

1. Codeplex Open Source Community. MIRIA SDK – Multitouch, Gestures and Multipurpose Input. Retrieved: 2010-04-30. Available at: http://miria.codeplex.com/
2. Echtler, F., and Klinker, G. A multitouch software architecture. In *Proceedings of the 5th Nordic Conference on Human-Computer interaction: Building Bridges* (Lund, Sweden, October 20 - 22, 2008). NordiCHI '08, vol. 358. ACM, New York, NY, 463-466.
3. Fraunhofer-Institute for Industrial Engineering IAO. Multi Touch for Java. *Website*. Retrieved: 2010-04-30. Available at: http://www.mt4j.org/
4. George, R., and Blake, J. Objects, Containers, Gestures, and Manipulations: Universal Foundational Metaphors of Natural User Interfaces. *CHI 2010*, April 10-15, 2010, Atlanta, Georgia, USA.
5. Global Information Systems Group. iGesture: A General Gesture Recognition Framework, *ETH Zurich*, 2007. Retrieved: 2010-04-30. Available at: http://www.igesture.org/
6. Hansen, T.E., Hourcade, J.P., Virbel, M., Patali, S., and Serra, T. PyMT: A Post-WIMP Multi-Touch User Interface Toolkit. Proceedings of Tabletop 2009.
7. Ideum. GestureWorks – true Multitouch for Flash. *Website*. Retrieved: 2010-04-30. Available at: http://gestureworks.com/
8. Kaltenbrunner, M. TUIO 2.0 Protocol Specification (Draft). January 18th, 2010. *Website*. Retrieved: 2010-04-30. Available at: http://www.tuio.org/?tuio20
9. Kaltenbrunner, M., and Bovermann, T. & Bencina, R. & Costanza, E. TUIO - A Protocol for Table Based Tangible User Interfaces. *Proceedings of the 6th International Workshop on Gesture in Human-Computer Interaction and Simulation* (GW 2005), Vannes (France)
10. König, A. W., Rädle, R., Reiterer, H. Interactive Design of Multimodal User Interfaces Reducing technical and visual complexity. *Journal on Multimodal User Interfaces*. Springer Berlin / Heidelberg, 2010
11. Lao, S., Heng, X., Zhang, G., Ling, Y., and Wang, P. A gestural interaction design model for multi-touch displays. In *Proceedings of the 2009 British Computer Society Conference on Human-Computer interaction*. British Computer Society Conference on Human-Computer Interaction. British Computer Society, Swinton, UK, 440-446.
12. Microsoft Developer Center. Microsoft Surface SDK. *Website*. Retrieved: 2010-04-30. Available at: http://msdn.microsoft.com/en-us/library/ee804845.aspx
13. Mindstorm Inc. Breezemultitouch – Multi-touch framework for WPF 3.5. *Website*. Retrieved: 2010-04-30. Available at: http://code.google.com/p/breezemultitouch/
14. De Nardi, A. Gesture Recognition mAnagement Framework for Interactive Tabletop Interfaces. *Diploma thesis at University of Pisa*. December 2008. Available at: http://grafitiproject.wordpress.com/
15. Ramanahally, P., Gilbert, S., Niedzielski, T., Velázquez, D., and Anagnost, C. Sparsh UI: A Multi-Touch Framework for Collaboration and Modular Gesture Recognition. *Proceedings of the World Conference on Innovative VR*. 2009
16. Wigdor, D., Williams, S., Cronin, M., Levy, R., White, K., Mazeev, M., and Benko, H. Ripples: Utilizing Per-Contact Visualizations to Improve User Interaction with Touch Displays. *UIST 2009*, October 4-7, Victoria, British Columbia, Canada.