

Towards a Formalization of Multi-touch Gestures

*Dietrich Kammer, Jan Wojdziak, Mandy Keck,
Rainer Groh*
Technische Universität Dresden
01062, Dresden, Germany
dietrich.kammer@tu-dresden.de

Severin Taranko
queo media GmbH
Tharandter Str. 13
01159, Dresden, Germany
severin.taranko@queo-media.com

ABSTRACT

Multi-touch is a technology which offers new styles of interaction compared to traditional input devices like keyboard and mouse. Users can quickly manipulate objects or execute commands by means of their fingers and hands. Current multi-touch frameworks offer a set of standard gestures that are easy to use when developing an application. In contrast, defining new gestures requires a lot of work involving low-level recognition of touch data. To address this problem, we contribute a discussion of strategies towards a formalization of gestural interaction on multi-touch surfaces. A test environment is presented, showing the applicability and benefit within multi-touch frameworks.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms: Design, Human Factors, Standardization, Languages, Theory

Keywords: Multi-touch, Gestures, Interaction, Formalization, Standardization

INTRODUCTION

Multi-touch is an emerging technology with a great potential to enhance human computer interaction. Currently, the large interest by scientists, companies, and end-users is stirred by readily available hardware solutions like the Surface from Microsoft® or the iPhone and iPad from Apple. Users can directly manipulate virtual objects, draw simple shapes to create objects, and execute commands with their fingers. New and innovative applications are the key to leveraging this potential. Current multi-touch frameworks like MT4j [12], Surface SDK [24], Multi-touch Vista [6], Sparsh-UI [32], or .NET4 [26] implement a set of standard gestures. Apart from these traditional gestures for pointing, translation, rotating, and scaling, new ones can be implemented. Without an abstraction of gestures, this task imposes a great workload on the programmer. Presently, recognition and processing

of user interaction relies on raw touch data and further support by the framework is limited or does not exist [18]. A system of gesture building blocks could help developers to construct gestures with the desired properties. Different kinds of gestures need to be addressed and treated in a uniform way. In this contribution, formalization of gestural interaction is proposed as a way to achieve gesture building blocks for multi-touch frameworks.

We first consider related work, followed by the presentation of GeForMT (Gesture Formalization for Multi-touch). Our formalization is based on a rationale rooted in semiotics. An implementation is discussed as well as the outline of future work in the field of gesture formalization and multi-touch framework development.

RELATED WORK

This review is focused on efforts to formalize multi-touch gestures. Further research and studies upon which our formalization is based will be presented as needed in the next section.

De-facto standards for multi-touch like Windows Touch [28] or TUIO [19] lack a formal abstraction of high-level properties of performed gestures. Extensibility is thus complicated and forces programmers to work on raw touch data if provided standard gestures (e.g. WM_GESTURE events in Windows 7) are insufficient. Lao et al. recognize the importance to generalize and standardize touch interaction in order to support reusability of touch styles [23]. Their focus is on providing an understanding across user's intentions, touch actions, and mapping to system functionality. While this is in line with our work, the main concern is basic gestures for manipulation only. Their use of state-transition diagrams is computationally sound, but introduces an abstraction level separated from touch interaction. It is neither suitable to be used without transformation in a framework, nor does it provide a quick and intuitive way to describe gestures in code.

Labeled Deductive Systems (LDS) are proposed by Görg et al. as formalism for the abstract representation of gestures in order to support recognition in multi-touch applications [16]. We argue that the claim "developers can construct intuitive logics which closely model the problem domain" does not hold true for every developer. While computationally elegant, the definition of sound rules based on logic may impose a great challenge. This problem increases if designers are involved in the process of gesture

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ITS'10, November 7–10, 2010, Saarbrücken, Germany.

Copyright 2010 ACM 978-1-4503-0399-6/10/11...\$10.00.

definition [18]. In contrast to computer scientists, they typically lack training in formal logics. In addition, the authors acknowledge that their reference implementation suffers from some shortcomings. These include race-conditions and ambiguities regarding rule application.

The proposal for a Gesture Definition Markup language (GDML) builds on XML and seeks to describe the syntax and characteristics of gestures [25]. The document is available online and work on it seems to be discontinued. The advantage over LDS and state-transition diagrams is an improved human-readability. Our main critique is that the XML-related syntactic sugar hampers the overall readability of a gesture description. Such a description quickly becomes bloated.

From the observations of related work, we can conclude that gestures are typically informally described or expressed in data structures that depend on specific algorithms. Few efforts to formalize gestures can be identified. Our goal is both a human-readable and computationally sound specification. From the presented approaches, XML meets these requirements best, but introduces significant syntactic sugar.

In order to further discuss our approach to formalizing multi-touch gestures, we compare the results to neighboring disciplines that have solved similar problems in their fields. It is reminiscent of work being done in sketching. Bimber et al. introduce a textual description for a sketch-language defined by a simple BNF-grammar [3]. Similar to GeForMT, basic shapes like POINT, CIRCLE, LINE and FREEFORM can be combined. Hammond and Davis describe LADDER [17], which allows defining domain specific sketching languages with a syntax similar to a small programming language. Their approach shows how such a formally defined language can be transformed into concrete recognizer code. While sketching is not the main concern of multi-touch applications, this discussion shows that GeForMT has the potential to unify these concepts. The possible benefits are evaluated by Appert and Zhai in their study concerning cognitive benefits of stroke shortcuts over traditional keyboard shortcuts [1].

Gesticulation is another interesting area of research when discussing multi-touch gestures. The Behavior Markup Language (BML) [22] is a standardization effort that captures a user's gaze, speech, and gestures. Due to the complexity of these different modalities, XML is used. CoGesT is yet another attempt to standardize gesture annotation systems and proposes the use of gesture feature vectors [15]. They are formally defined using a Backus Naur Form (BNF) and can be transferred into XML for interchange purposes. These developments show that transformation of GeForMT which is defined via a context-free grammar (EBNF) into an XML structure is feasible for interoperability with other modalities, such as speech or gesticulation in three-dimensional space.

In the next section, GeForMT is described, which uses a textual description of gestures without the syntactic sugar that XML introduces.

GEFORMT: GESTURE FORMALIZATION FOR MULTI-TOUCH

As a rationale for the formalization of multi-touch gestures, we chose semiotics, a technique utilized in linguistics. This is an approach already suggested for Human Computer Interaction by researchers like Nespoulous [30]. Semiotics describes all phenomena associated with the production and interpretation of signs and symbols [11,31]. Syntactics describes these symbols and their combination, semantics addresses meaning and pragmatics is concerned with the user's mental model. Table 1 summarizes the three aspects of semiotics, applies them to multi-touch interaction, and relates them to methods of formalization and implementation.

	Syntactics	Semantics	Pragmatics
Scope	Symbols	Meaning	Interpretation
Formalization	Formal grammars	Ontologies	User model
Implementation	Recognition engines	Application commands	Metaphors with feedback and feed-forward
Multi-touch	Performed gesture	Semantic dimensions	Intuitive interface concepts

Table 1: Overview of semiotics for multi-touch gestures

In this section, we describe the GeForMT approach to the formalization of multi-touch gestures and relate it to relevant studies and research concerning gestures. At this stage of the research, the focus is on syntactics. Semantics and pragmatics are covered more briefly, but enough to give a complete overview of the rationale behind GeForMT. The following description does not include implementation specific details but is focused on high-level properties. The section "Implementation" elaborates on more detailed requirements of software and hardware, in order to effectively make use of the GeForMT approach.

Syntactics

In order to formalize the syntax of multi-touch interaction, a comprehensive understanding of the possible signs that can be produced by a user is necessary. Technical resources and standards like the TUIO protocol [19] or Windows 7 gesture events [28] give insight into the data that is tracked and recorded on multi-touch devices. To establish an adequate grammar adapted to users, this technical information has to be related to empirical findings by researchers like Wobbrock [37].

Although many different sensor technologies and devices are available on the market, the basic information supplied for multi-touch interaction is coordinates on a two-dimensional surface. They are grouped into continuous

contacts. A contact is continuous if it does not change in size and is not removed. To be able to describe these contacts, GeForMT uses so called *pose functions*. They consist of a prefix, describing the nature of the contact. For instance, fingers or hands are abbreviated to “F” and “H”, respectively. Introducing new prefixes, e.g. “B” for a generic blob, is possible and results in an extension of the modular GeForMT syntax. The prefix is followed by braces which encapsulate the trail that is produced by the contact (cp. Table 2).

Language element	Purpose	Example
Pose Function	Describes the shape of the blob that is being tracked	2F(...), 1H(...)
Atomic Gestures	Describes the movement of the tracked blob	MOVE, POINT, DEPOINT, HOLD, LINE, CIRCLE, SEMICIRCLE
Composition Operators	Describes the temporal progression of a gesture, e.g. parallel, successive, or asynchronous execution	Comma, asterisk, plus-sign
Focus	Specifies the focus of a gesture, i.e. objects, a set of objects, or application	LINE(o), LINE(s), LINE(a)
Area Constraints	Describes the relative movement of atomic gestures, e.g. overlap or convergence	CROSS[...], SYNC[...], JOIN[...], SPREAD[...]

Table 2: Language elements of GeForMT

In GeForMT, these trails are abstracted to *atomic gestures*, which are centered on a compass rose (cf. Figure 1). They describe simple shapes like lines, semicircles, and circles, which can be combined to more complex entities. Moreover, freeforms can be specified (MOVE) as well as short contacts leaving no trail at all (POINT, HOLD). The special DEPOINT gesture is used to describe a short lifting of a finger or hand, as part of one gesture. Depending on the atomic gesture, constraints can be specified by adding an underscore. For instance, the direction vector of a line can be restricted (e.g. LINE_NORTH). In this case, the system must be able to report or interpret directions with respect to the user. If necessary, a *focus* for an atomic gesture can be added as well. A placeholder for an object (o), the application (a) or a set of objects (s) is appended in braces. The focus can be a list of items and objects accept indices to distinguish between them. Sets contain an arbitrary number of objects.

Since multi-touch allows parallel interaction, simultaneous contacts and their respective trails have to be addressed. This is achieved by GeForMT’s *composition operators*. An asterisk (multiplication sign) specifies synchronous movement, a plus-sign (addition sign) asynchronous movement. If no parallel but successive movements are

needed, a comma is used as delimiter. A simple sequence of atomic gestures is defined in this way. Finally, *area constraints* can be used to specify relative movement of atomic gestures. Similar to pose functions, a prefix is used to define the relation between atomic gestures. For instance, overlap is indicated by the CROSS prefix. This allows the definition of a cross, consisting of atomic line gestures. SYNC requires two lines to be drawn parallel to one another, JOIN describes a converging motion, and SPREAD denotes that two atomic gestures depart from each other.

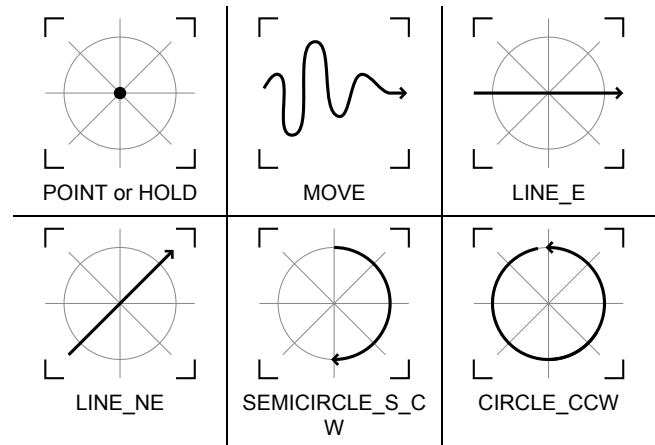


Figure 1: Examples of atomic gestures centered on a compass rose (POINT, HOLD, MOVE, and variations of LINE, SEMICIRCLE, and CIRCLE)

For a better understanding of GeForMT, the language elements (cp. Table 2) are related to the user study conducted by Wobbrock [37] (cp. Table 3). Eliciting gestures from non-technical users has identified three main categories concerning the form of a gesture.

Freeman’s extension of Wobbrock’s Taxonomy of Surface Gestures (form category)		GeForMT
Registration Pose	Single Finger	Pose functions
	Multi Finger	Pose functions if in close proximity, else composition operators
	Single Shape	Pose functions
	Multi Shape	Pose functions if in close proximity, else composition operators
Continuation Pose	Static	Atomic gestures (POINT, HOLD)
	Dynamic	Composition operators
Movement	No Path	Atomic gestures (POINT, HOLD)
	Path	Atomic gestures (LINE, CIRCLE, SEMICIRCLE)

Table 3: Application of GeForMT language on Freeman’s extension of Wobbrock’s taxonomy [13]

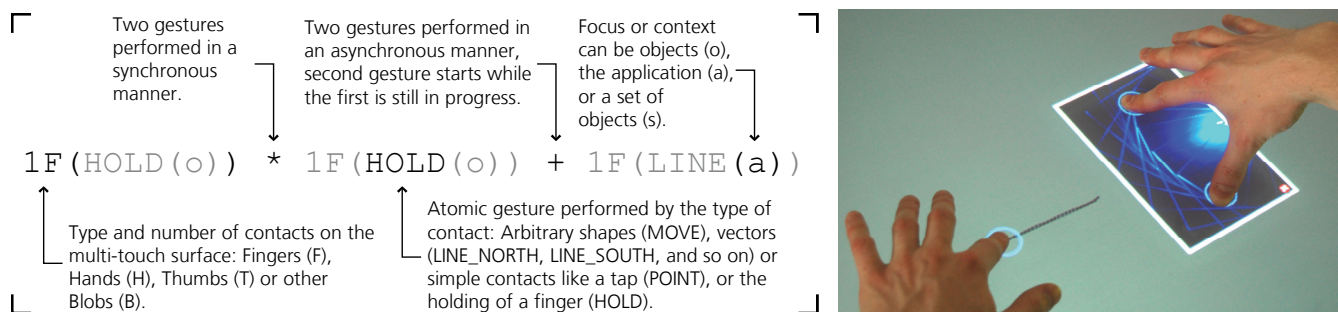


Figure 2: Complex gesture (spin) described by GeForMT syntax (left) and resulting gesture (right)

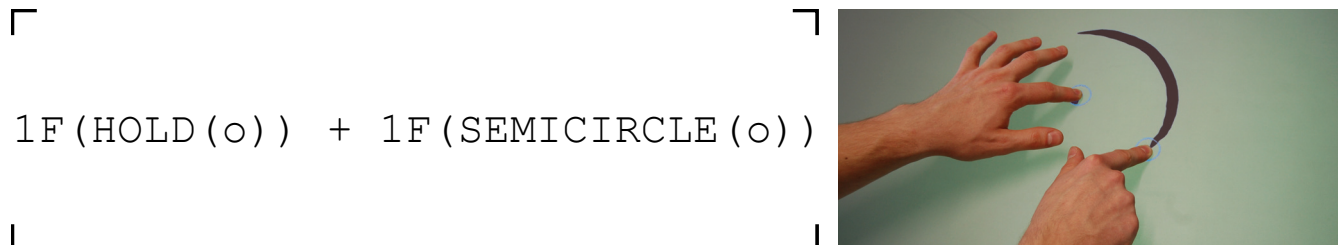


Figure 3: Rotate gesture described by GeForMT (left) and resulting gesture (right)

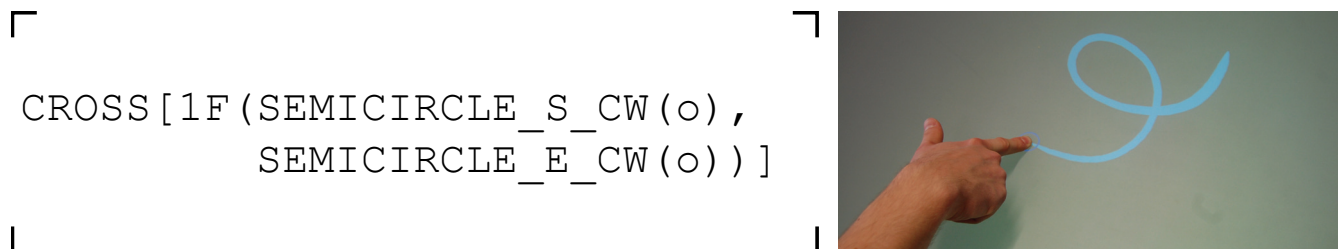


Figure 4: Curlicue gesture described by GeForMT (left) and resulting gesture (right)

The registration pose describes the shape of the initial contact on the multi-touch surface. One or more fingers or arbitrary shapes produced by hands are possible. To this end, GeForMT offers its pose functions. For instance, $2F(\dots)$ is used to denote two fingers touching the surface.

According to Wobbrock, the continuation of this pose can be either static or dynamic. Static means that the shape remains the same throughout the gesture. The continuation of the pose is dynamic if the shape changes. For example, the gesture starts with one finger and ends with two on the surface. GeForMT’s composition operators address the dynamic continuation poses and the atomic gestures POINT and HOLD static ones.

Finally, the movement category introduces a distinction in path and no path. If there is movement involved in the gesture, a path is recorded. This path can be specified by atomic gestures (cf. Figure 1) and their combination. Atomic gestures are freeforms, lines, semicircles, and circles. If a direction or rotation is important, these atomic gestures can be constrained. Should no movement be required, e.g. holding a finger down on the surface, POINT and HOLD are used. Again, the language elements offered by GeForMT are summarized as a reference in Table 2.

Interestingly enough, in Freeman’s extension of the form category of Wobbrock’s Taxonomy of Surface Gestures (cp. Table 3), examples are already specified in a formula-like manner (such as “1-finger down + tap other”) [13]. Our goal is to describe gestures in such a human-readable manner, but with a formally defined grammar.

The resulting textual description is similar to the informal description of gesture production. The Extended Backus Naur Form (EBNF) in Figure 5 defines the syntax of GeForMT. Consider the instruction: “Use two fingers to draw a circle on an object”. In our formalization the string $2F(\text{CIRCLE}(o))$ would result.

Suppose that this gesture needs to be constrained to a semicircle drawn clockwise in the eastern quadrant of the compass rose (cp. Figure 1). In the resulting description, the atomic gesture is substituted (semicircle) and constrained by defining the endpoint on the compass rose (south) and the rotation (clockwise) – note that abbreviations are used to keep the resulting string short: $2F(\text{SEMICIRCLE_S_CW}(o))$.

Figure 2 shows a complex spin gesture and its description using GeForMT. This gesture consists of three contacts with the interaction surface. Two fingers are placed

simultaneously on an object, both defining the focus of the gesture and an axis for spinning. While these simultaneous HOLD gestures are still in progress, a third finger draws a line on the application, triggering the spinning event.

Another manipulation gesture (rotate) is shown in Figure 4. The presented expression imposes a certain style to rotate an object. First, a HOLD is performed by one finger, followed by the drawing of a semicircle by another finger. It is conceivable and should be supported to express alternatives for rotation, e.g. both fingers drawing a semicircle. Figure 4 illustrates another type of gesture, a stroke gesture. The curlicue is performed by drawing two overlapping semicircles. It is required that the gesture starts in the top right corner. Again, alternatives are conceivable, e.g. to start in the bottom right corner.

1: complex	::=	gesture (operator gesture)* relation['gesture (operator gesture)*']
2: gesture	::=	function('atomfocus(','atomfocus')*) atomfocus(','atomfocus)*
3: relation	::=	'CROSS' 'JOIN' 'SYNC' 'SPREAD'
4: atomfocus	::=	atom atom('focuslist')
5: focuslist	::=	focus(','focus)*
6: focus	::=	'a' 's' 'o'[integer]
7: function	::=	[integer]'F' [integer]'H' [integer]'B'
8: operator	::=	'*' '+' ','
9: atom	::=	identifier vector['_direction'] shape['_direction']['_rotation']
10: identifier	::=	'MOVE' 'POINT' 'DEPOINT' 'HOLD'
11: vector	::=	'LINE'
12: shape	::=	'CIRCLE' 'SEMICIRCLE'
13: direction	::=	'NORTH' 'N' 'NORTHEAST' 'NE' 'EAST' 'E' 'SOUTHEAST' 'SE' 'SOUTH' 'S' 'SOUTHWEST' 'SW' 'WEST' 'W' 'NORTHWEST' 'NW'
14: rotation	::=	'CLOCKWISE' 'CW' 'COUNTERCLOCKWISE' 'CCW'
15: integer	::=	('1'-'9') ('0'-'9')*

Figure 5: Extended Backus Naur Form (EBNF) grammar to define the Syntax of GeForMT

A flick gesture that requires three fingers would be simply defined by $3F(LINE(o))$. The speed of the atomic LINE gesture that is necessary for flicking is not explicitly expressed and has to be evaluated later in the application. This limitation is further discussed in the section “Discussion”.

An online appendix of more GeForMT expressions can be found at <http://vi-c.de/geformt/>. The gesture sets supplied by the Sparsh-UI framework [32], mostly focused on manipulation and online gestures, as well as the Microsoft® Application Gestures, mostly stroke gestures, are formalized using GeForMT.

In this section, the formalization of the syntax of multi-touch gestures has been discussed and addressed by the GeForMT approach. The textual description is kept simple

and is easy to extend. For instance, new basic shapes can be added to atomic gestures, as well as constraints. The next section shows how the current syntax addresses aspects concerning the Semantics of surface gestures.

Semantics

The formalization we propose must be able to address all intentions that users want to express through surface gestures. Classical taxonomies describing the semantic dimensions of gesticulation make some distinctions that are noteworthy when discussing multi-touch gestures (cp. Table 4).

Cassell [5] describes Iconic gestures as using “some features of the action or event being described”. In contrast, Metaphoric gestures represent a concept without physical form. Deictics spatialize or locate aspects of the discourse. Beat gestures are “movements that do not change in form with the content of the accompanying speech”. Beat gestures do not occur in multi-touch without speech input and can be disregarded for the scope of this paper. Karam and Schraefel’s classification of gesture based human computer interaction identifies five gesture styles [21]. Again, Deictic gestures identify or locate an object. The term gesticulation encompasses “hand movements within the context of the user’s speech” and is disregarded due to the scope of this paper.

Cassell [5]	Karam and Schraefel [21]	Wobbrock [37]	George and Blake [14]	GeForMT
Iconic	Semaphores	Symbolic	Gestures	Iconic
Metaphoric		Abstract		Lexicalic
			Metaphorical	
Deictics	Deictic	Physical	Manipulations	Deictic
	Manipulation			Manipulative
				Navigational

Table 4: Semantic dimensions of gestures

Manipulations occur when a “tight relationship between the actual movements of the gesturing hand/arm” applies to an entity. Semaphores require a stylized dictionary and Sign Language (SL) is used for conversational style interfaces, requiring individual signs with grammatical structures. Since the focus is on surface gestures only, SL is disregarded as well.

The semantic dimensions of multi-touch gestures as a highly specialized form of gesticulation have been classified by Wobbrock [37], and George and Blake [14]. George and Blake identify only two categories: manipulations and gestures.

Manipulations are defined similar to Karam and Schraefel’s classification and Wobbrock labels them Physical. Every other gesture that has no direct mapping to the modification of an object is simply called “gesture” by

George and Blake. In this category, Wobbrock makes a distinction in symbolic (gesture depicts a symbol visually), metaphorical (gesture indicates a metaphor) and abstract (gesture-referent mapping is arbitrary).

We have identified several shortcomings of existing approaches concerning surface gestures. In our taxonomy, the distinction in metaphorical and abstract gestures is not considered since every gesture on a two-dimensional surface must be abstracted due to its mapping to a two-dimensional, planar surface. In addition, metaphors are culturally defined and will have a different or no meaning depending on the user's background. As a result, gestures referring to high-level commands will need to be learned by the user. We use the term "Lexicalic" and distinguish Iconic gestures, which visually depict a symbol. In the case of multi-touch, Iconic gestures can be used to draw pictures that are used "as-is" within a painting application. A combination of atomic gestures (freeforms and simple geometrical shapes) can be used to specify Lexicalic and Iconic gestures with GeForMT. George and Blake's Manipulations are further split into Deictic, Manipulative, and Navigational gestures. Only Navigational gestures are introduced as a new concept for surface gestures. This serves to differentiate between the manipulation of the application state and an object within the application. GeForMT uses the focus of atomic gestures to address this distinction.

Category	Language elements	Gesture type
Lexicalic	Combination of atomic gestures	Offline
Iconic	Combination of atomic gestures	Online
Deictic	Focus suffix (o) and atomic gestures (POINT, HOLD)	Offline
Manipulative	Focus suffix (o) and atomic gesture and their combination	Online
Navigational	Focus suffix (a) and atomic gesture and their combination	Online

Table 5: Using GeForMT to express different semantics of gestures.

Table 5 summarizes how the different semantic dimensions are covered by language elements of GeForMT. A distinction in offline and online processing must be made in the implemented framework. Online gestures are evaluated and processed while they are being performed. This means that the state of the application or affected object changes continuously. Offline gestures include symbols to create new objects or shortcuts to execute application commands. Their effect is evaluated after they are completely performed. The state of an online gesture needs to be reported as long as it is in progress. Different parameters that change while the gesture is being performed must be continuously extracted. Fine-grained information for each atomic gesture has to be available within the framework. This is discussed in more depth in section "Implementation".

As shown in Table 1, ontologies are used to formalize meaning in a computer system. This is an approach well established in the semantic web [7,24]. Currently, GeForMT does not require formalized semantics from the application developer. It is conceivable that formally defining the meaning of a command leads to more flexible and reusable mappings of syntax descriptions using GeForMT to commands across multiple applications.

Pragmatics

The user model can be associated with pragmatics. The user's knowledge of gesture production and general emotions involved in the interaction with multi-touch hardware and interfaces is essential to this element. With the help of suitable metaphors at the conceptual stage, intuitive interface concepts can be conceived for multi-touch applications. Different aspects of pragmatics in HCI have been explored at depth elsewhere [4]. GeForMT is suited to establish gesture lexicons (gesticons) which can be used to create uniform feedback and feed-forward as shown by [36] and [2], respectively. These mechanisms can substantially aid the user when dealing with multi-touch applications.

Since pragmatics is the most complex aspect of semiotics, a more detailed discussion is out of the scope of this paper and is subject of future work. Another concern for pragmatics is ergonomic properties of gestures. Once the formalization of syntactics is complete, these can be researched further, e.g. by governmental bodies or non-governmental organizations responsible for standards development like the NSF International, The Public Health and Safety Company™.

This section has established the foundations of GeForMT. We next present first steps towards the implementation of these concepts.

IMPLEMENTATION

In order to efficiently implement GeForMT, certain requirements must be met. These general considerations are followed by the description of a first prototype using GeForMT.

General Considerations

So far, GeForMT is described as a high level abstraction of gesture properties. However, an implementation must supply more fine-grained details about atomic gestures, e.g. the speed of a LINE gesture or its exact coordinates on the interaction surface. Moreover, the implementation has to accommodate the facilities provided by GeForMT. For instance, reference orientations must be clarified for constrained gestures like LINE_NORTH, e.g. with respect to the user, objects, or the table. To this end, finger orientation [35] or application specific coordinate systems can be leveraged. Multi-touch systems that are wall-mounted or tilted already provide an unambiguous orientation.

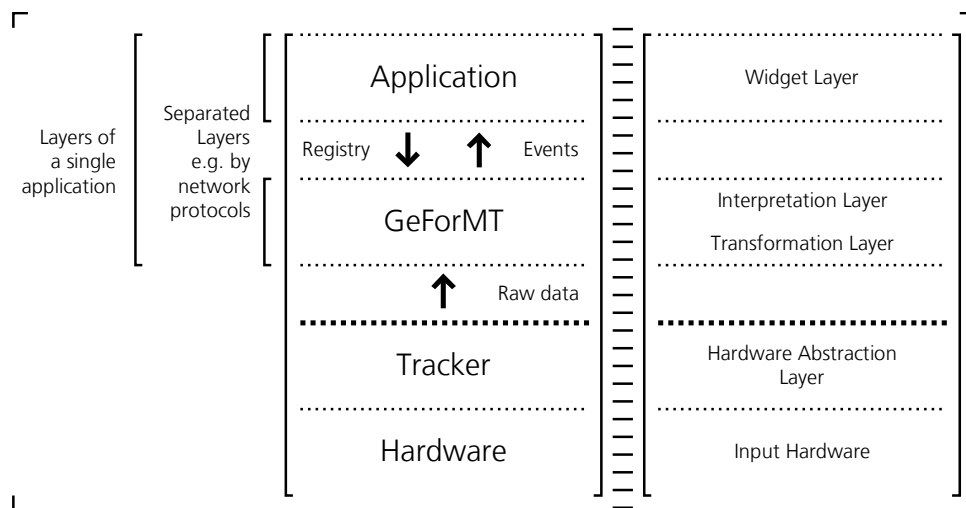


Figure 6: General architecture using GeForMT (left), compared to Echtler's multi-touch software architecture (right) [10]

Eliciting the focus of a gesture is another challenge. As Figure 2 shows, a gesture can be performed on an object to set a focus. It is also conceivable that the target area of each object is extended around its geometry. This would allow gestures to be performed nearby an object, which is essential when using tangible objects within a multi-touch application. For instance, the Graffiti Framework introduces a targeting process, which results in target lists for gestures [8]. It contains the IDs of tangible objects that are close to the performed gesture. In this case, a certain radius around the object accounts for a gesture targeting it.

As described earlier, online gestures have to be tracked differently compared to offline gestures. Current frameworks are mostly focused on either of the two types or introduce a specific tracking state or attribute to distinguish online or offline processing of a gesture. The same applies to GeForMT. Again, Table 5 relates the semantic dimensions of gestures to the language elements of GeForMT with regard to online or offline gesture types.

Figure 6 shows a software architecture using GeForMT. In analogy to [10], a layer providing transformation and interpretation has to be developed which works on the raw data provided by hardware. On top of this layer, the application must be able to register gestures in GeForMT syntax and receive events if they are recognized. The framework should be able to provide very detailed data if necessary. For instance, specific touch coordinates and blob-sizes might be of interest to application developers.

The layers in discussion can be separated using a gesture server approach with network protocols as suggested by [10] and [32]. Alternatively, layers of a single application would result in tighter coupling of gesture recognition and processing, allowing faster prototyping [20]. The formal specification of GeForMT offers further potential to automatically create optimized recognizer code for specific applications or existing frameworks. In addition, reasoning over gesture descriptions can reveal conflicts and

ambiguities between gestures used within an application. The feasibility of these prospects is subject of future research.

GeForMT Test Environment

A first prototype using GeForMT is a test environment for multi-touch gestures (cp. Figure 7). Touch data provided by the TUIO protocol [19] is parsed and transformed into the syntax of GeForMT. At this stage, recognition strategies for sketch understanding from [33] are applied.

The test environment allows the production of gestures, which are translated into their corresponding GeForMT syntax. As visual feedback, a filled circle indicates the starting point and a non-filled one indicates the endpoint of a tracked blob. Additionally, the TUIO output of the performed gesture is displayed for debugging purposes.

So far, the test environment is focused on vector based gestures and does not implement composition operators or online processing facilities. The performance of the current recognizer needs to be optimized as well. Particularly, calculation speed and accuracy of the generated expressions are lacking refinement. The results and experiences from the first working prototype are currently being incorporated into a more general architecture for multi-touch frameworks.

DISCUSSION

GeForMT is a gesture abstraction and thus its granularity is kept low. Especially constraints with a high granularity regarding timing and dynamics such as speed, acceleration, or precise geometric properties are not part of the language itself. At the stage of implementation, this information has to be available in order to correctly process a gesture (cp. section "Implementation"). Once it becomes apparent that certain constraints need to be moved to the stage of gesture definition, language extensions need to be considered. In this case, the trade-off between expressiveness and complexity of the GeForMT language has to be evaluated.

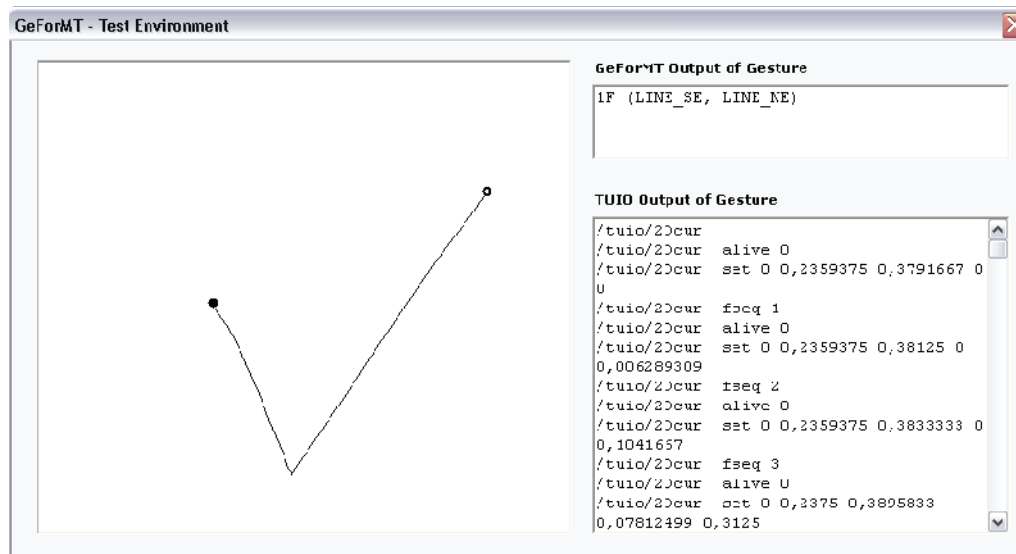


Figure 7: The GeForMT test environment

Some facilities of GeForMT like area braces (cp. Table 2) to describe the divergence of atomic gestures may be intricate to process and to evaluate. Especially if more than two atomic gestures need to diverge from another. These pitfalls require suitable interpretations and solutions in the implemented framework.

Currently, GeForMT may not be able to express certain special gesture types. These include gestures that require a temporal timeout and fluid movements which are repeated over a period of time. However, for productive instead of entertaining environments, these gestures have been argued to cause unwanted fatigue [38]. Another aspect is the composition of complex gestures using already defined complex gestures. This is not yet part of the language for the sake of simplicity and comprehensibility (cp. the EBNF in Figure 5), but adding iterative elements to GeForMT is feasible.

Furthermore, GeForMT is limited to single-user gestures at this point. The extension to multi-user settings is yet another challenge for the formalization of gestures. Morris et al. propose a taxonomy for cooperative gestures along different axes [29]. Symmetric and parallel gestures might be expressed by GeForMT's composition operators, but proxemic distance of users and additivity of gestures are concerns specific to cooperative settings. Identity-Awareness and number of users involved in a gesture could be specified by extending GeForMT with user IDs. Supporting user IDs requires hardware which is able to assign multi-touch interaction to individual users. This can be achieved by special interaction zones for each user, fingerprint scanning, or additional sensors and tracking [9,34].

CONCLUSIONS AND FUTURE WORK

This paper presented a new approach to ease the definition of multi-touch gestures for developers. This is achieved by the proposed Gesture Formalization for Multi-Touch

(GeForMT). In the future, developers will be able to both specify gestures by example, using a test environment (cp. Figure 7), or using a short textual description. GeForMT is simple and quick to learn due to its closeness to natural language and use of a very basic mathematical structure. At this stage of the research process, the tentativeness of the presented approach has to be acknowledged. The realization of the appropriate software structures is still subject to further research. In addition, the proposed description of gestures has the potential to bridge the gap between designers and software developers. Interaction designers can quickly define intended gestures within design tools. Software development environments can exchange or translate these descriptions. This could be an extension to existing tools like Microsoft's® Expression Blend and Visual Studio.

The implementation of a multi-touch framework based on GeForMT must be further researched. In addition, suitable parsers and tools for code generation might be used to integrate GeForMT into existing frameworks. For instance, the internal abstraction in features, regions, and gestures proposed in [10] might be unified with the GeForMT approach to specifying gestures.

At this point of research, expert knowledge and relevant research is cumulated to provide a conclusive framework in order to implement tools which enable an efficient user study. Such a user study has to address the actual human-readability of a gesture expression. Particularly, correct reasoning about GeForMT's composition operators has to be assured. In addition, the writability of a GeForMT expression is another important concern which needs to be validated. To enhance the workflow between designers and programmers, macro-recording as shown by the GeForMT test environment has to be compared to gesture descriptions written from scratch.

ACKNOWLEDGMENTS

This work has been supported by the European Union and the Free State Saxony through the European Regional Development Fund (ERDF). The research presented in this article has been conducted in cooperation with queo GmbH from Dresden, Germany. Thanks are due to Christin Henzen, Benjamin Bach, and Ingmar S. Franke for their invaluable comments and support for this research.

REFERENCES

1. Appert, C. and Zhai, S. Using strokes as command shortcuts: cognitive benefits and toolkit support. In Proceedings of the 27th international Conference on Human Factors in Computing Systems (Boston, MA, USA, April 04 - 09, 2009). CHI '09. ACM, New York, NY, 2289-2298.
2. Bau, O. & Mackay, W. E. OctoPocus: A Dynamic Guide for Learning Gesture-Based Command Sets. UIST 2008, October 19-22, 2008, Monterey, California, USA.
3. Bimber, O., Encarnacao, L. M., Stork, A. A multi-layered architecture for sketch-based interaction within virtual environments, *Computers & Graphics*, Volume 24, Issue 6, December 2000, Pages 851-867, ISSN 0097-8493.
4. Buxton, W. Lexical and Pragmatic Considerations of Input Structures. *Computer Graphics*, 17 (1), 1983, 31-37.
5. Cassell, J. A Framework For Gesture Generation and Interpretation. *Computer Vision in Human-Machine Interaction*. pp. 191-215. Cambridge University Press. 1998.
6. Codeplex. Multi-Touch Vista. Website. Last Update: 11/20/09, Last Checked: 5/5/2010, Available Online: <http://www.codeplex.com/MultiTouchVista>.
7. Davies, J., van Harmelen, F., Fensel, D. Towards the Semantic Web: Ontology-driven Knowledge Management. John Wiley & Sons Inc., New York, NY, USA, 2002.
8. 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/>
9. Dietz, P. and Leigh, D. DiamondTouch: A Multi-User Touch Technology. UIST 2001, 219-226.
10. 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.
11. Eco, U. A Theory of Semiotics. Indiana University Press, USA, 1979.
12. Fraunhofer-Institute for Industrial Engineering IAO. Multi Touch for Java. Website. Retrieved: 2010-10-15. Available at: <http://www.mt4j.org/>
13. Freeman, D., Benko, H., Morris, M. R., and Wigdor, D. ShadowGuides: Visualizations for In-Situ Learning of Multi-Touch and Whole-Hand Gestures. In Proceedings of ACM Interactive Tabletops and Surfaces (ITS '09). pp. 165-172, 2009.
14. George, R. & Blake, J. Objects, Containers, Gestures, and Manipulations: Universal Foundational Metaphors of Natural User Interfaces. CHI 2010, April 10-15, 2010, Atlanta, Georgia, USA.
15. Gibbon, D., Gut, U., Hell, B., Looks, K., Thies, A., Trippel, T. A Computational Model of Arm Gestures in Conversation. Proceedings of Eurospeech, 2003, 813-816.
16. Görg, M. T., Cebulla, M., Garzon, S. R., A Framework for Abstract Representation and Recognition of Gestures in Multi-touch Applications, *International Conference on Advances in Computer-Human Interaction*, pp. 143-147, 2010 Third International Conference on Advances in Computer-Human Interactions, 2010.
17. Hammond, T. and Davis, R. 2007. LADDER, a sketching language for user interface developers. In ACM SIGGRAPH 2007 Courses (San Diego, California, August 05 - 09, 2007). SIGGRAPH '07. ACM, New York, NY, 35.
18. Kaindl, G. Towards a flexible software framework for multi-touch application design. *Engineering Patterns for Multi-Touch Interfaces 2010. A workshop of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. June 20, 2010.
19. Kaltenbrunner, M. TUIO 2.0 Protocol Specification (Draft). January 18th, 2010. Website. Retrieved: 2010-05-28 2010-04-30. Available at: <http://www.tuio.org/?tuio20>
20. Kammer, D., Freitag, G., Keck, M., Wacker, M. Taxonomy and Overview of Multi-touch Frameworks: Architecture, Scope, and Features. *Engineering Patterns for Multi-Touch Interfaces 2010. A workshop of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. June 20, 2010.
21. Karam, M., Schraefel, M. C. A taxonomy of Gestures in Human Computer Interaction. *ACM Transactions on Computer-Human Interactions* (submitted), 2005.
22. Kopp, S., Krenn, B., Marsella, S., Marshall, A. N., Pelachaud C., Pirker, H., Thórisson, K. R., Vilhjálmsón, H. Towards a Common Framework for

- Multimodal Generation: The Behavior Markup Language. J. Gratch et al. (Eds.): IVA 2006, LNAI 4133, 2006, pp. 205 – 217, Springer-Verlag Berlin Heidelberg.
23. 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 (Cambridge, United Kingdom, September 01 - 05, 2009). British Computer Society, Swinton, UK, 440-446.
 24. Lopez, V., Uren, V., Sabou, M., Motta, E. Cross Ontology Query Answering on the Semantic Web: An Initial Evaluation. K-CAP 2009, September 1-4, 2009, Redondo Beach, California, USA.
 25. Meyer, A. S. Gesture Recognition. Website. Retrieved: 2010-10-15. Available at: [http://wiki.nuigroup.com/Gesture Recognition](http://wiki.nuigroup.com/Gesture%20Recognition), Nov. 2009.
 26. Microsoft® .NET Framework Developer Center. Website. Retrieved: 2010-10-15. Available at: <http://msdn.microsoft.com/en-us/netframework/>
 27. Microsoft® Developer Center. Microsoft® Surface SDK. Website. Retrieved: 2010-10-15. Available at: <http://msdn.microsoft.com/en-us/library/ee804845.aspx>
 28. Microsoft® Developer Center. Windows Touch. Website. Retrieved: 2010-10-15. Available at: <http://msdn.microsoft.com/en-us/library/dd562197.aspx>
 29. Morris, M. R., Huang, A., Paepcke, A., Winograd, T. Cooperative Gestures: Multi-user Gestural Interactions for Co-located Groupware. Proceedings of the ACM CHI Conference on Human Factors in Computing Systems, 2006, pp. 1201-1210, ACM Press.
 30. Nespoulous, J.-L., Perron, P., Lecours, A. R. Biological Foundations of Gestures: Motor and Semiotic Aspects. Lawrence Erlbaum Associates, Hillsdale, MJ, 1986.
 31. Posner, R. & Debus, S. (Hrsg.). 30 Jahre Zeitschrift für Semiotik 1979 – 2008. Stauffenburg Verlag.
 32. 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.
 33. Sezgin, T. M., Stahovich, T., and Davis, R. Sketch based interfaces: early processing for sketch understanding. In Proceedings of the 2001 Workshop on Perceptive User interfaces (Orlando, Florida, November 15 - 16, 2001). PUI '01, vol. 15. ACM, New York, NY, 1-8.
 34. Walther-Franks, B., Schwarten, L., Teichert, J., Krause, M., Herrlich, M., User Detection for a Multi-touch Table via Proximity Sensors, IEEE Tabletops and Interactive Surfaces 2008.
 35. Wang, F., Cao, X., Ren, X., Irani, P., Detecting and Leveraging Finger Orientation for Interaction with Direct-Touch Surfaces. UIST'09, October 4–7, 2009, Victoria, British Columbia, Canada.
 36. Wigdor, D., Williams, S., Cronin, M., Levy, R., White, K., Mazeev, M., Benko, H. Ripples: Utilizing Per-Contact Visualizations to Improve User Interaction with Touch Displays. UIST 2009, October 4-7, Victoria, British Columbia, Canada.
 37. Wobbrock, J. O., Morris, M. R., Wilson, A. D. User-Defined Gestures for Surface Computing. Proceedings of CHI 2009, 1083-1092. April 4-9, Boston Massachusetts, USA.
 38. Yee, W., Potential Limitations of Multi-touch Gesture Vocabulary: Differentiation, Adoption, Fatigue. Human-Computer Interaction. Novel Interaction Methods and Techniques, Lecture Notes in Computer Science, 2009, Springer Berlin / Heidelberg, pp. 291-300.