Sensyble Workshop 2010 manuscript No.
(will be inserted by the editor)

TUIO AS3: A Multi-Touch and Tangible User Interface Rapid Prototyping Toolkit for Tabletop Interaction

Johannes Luderschmidt[1], Immanuel Bauer[2], Nadia Haubner[1], Simon Lehmann[1], Ralf Dörner[1], Ulrich Schwanecke[1]

[1] RheinMain University of Applied Sciences, Wiesbaden Rüsselsheim Geisenheim, Department of Design, Computer Science and Media, Kurt-Schumacher-Ring 18, 65197 Wiesbaden, Germany
[2] Vienna University of Technology, Institute for Design & Assessment of Technology, Human Computer Interaction Group, Favoritenstrae 9-11/3, 1040 Wien, Austria

Received: date / Revised version: date

Abstract Multi-touch and tangible input paradigms provide new opportunities for post-WIMP (Windows, Icons, Menus, Pointer) user interfaces. The development of such novel interfaces challenges designers and software developers alike. TUIO AS3 addresses these challenges by providing a toolkit for the rapid development of multi-touch and tangible user interface (TUI) interaction. The TUIO AS3 toolkit comprises two basic functionalities. Firstly, it offers a sophisticated multi-touch and TUI interaction API. Secondly, to foster the development process, TUIO AS3 is capable to simulate complex multi-touch and TUI interaction via mouse and keyboard. In terms of the interaction API, TUIO AS3 provides a wrapper for the network-based open source protocol TUIO in Adobe Flash's programming language Actionscript 3 (AS3). TUIO AS3 allows to enhance user interface (UI) elements developed in AS3 with interactivity for gestural and tangible interaction. The interaction APIs support two kinds of interaction: On one hand, multi-finger/multi-hand controls for dragging, rotation and scaling can be added to any UI element to enhance it with standard multi-touch functionality. On the other hand, complex gestures can be defined with a simple grammar and tracked within TUIO AS3. In terms of simulation, TUIO AS3 offers means to simulate multi-touch and TUI interaction without the need for an additional simulator application. Aspects like multi-finger flicking, multi-finger rotation and complex gestures are supported via keyboard and mouse interaction. Tangibles can be added, manipulated and removed. TUIO AS3 has been used and matured in several projects.

1 Introduction

Tabletop computing UIs that employ multi-touch and tangible user interaction differ from WIMP interfaces in several aspects. Firstly, the interaction with the interface can be performed with multiple fingers and real-life objects called tangibles in a TUI instead of using a mouse pointer. Secondly, by using multiple fingers, a user can perform gestures and multiple users can simultaneously interact with the UI. Thirdly, as multiple users can be dispersed around the tabletop, the UI should allow content orientation towards users.

These interaction aspects have been considered widely in literature (for instance in [14] and [11]) and various multi-touch interaction software frameworks allow for these aspects (for instance [3] and [4]). These frameworks support a rapid prototyping approach that allows to quickly create design prototypes. Such design prototypes can then be evaluated in user tests rather than having to interpret designs based on descriptions. This is a promising approach for the creation of tabletop interfaces, where complex aspects like multi-touch gestural interaction or collaboration of multiple users must be evaluated that seem difficult to test on a purely conceptual basis. Additionally, as tabletop interfaces are not as common as WIMP interfaces, fewer best-practice approaches for interface components and user interaction exist. Thus, for the design of interaction and UIs for tabletop interfaces the development of prototypes are crucial.

However, the rapid prototyping tabletop frameworks introduced in section 2 suffer from several shortcomings. Firstly, there is no combined framework that enables prototyping of hybrid multi-touch and tangible user interfaces. Secondly, multi-touch gesture interaction often makes use of 'standard' gestures like scale, pinch or rotate gesture to enlarge, shrink or rotate objects. Although, a user should be allowed to use as many fingers or even hands to perform such gestures, most frameworks constrain users to perform these gestures with two fingers only. Hence, a more general approach for these standard gestures is necessary. Additionally, the design and use of complex gestures for special purposes like a *one finger down one finger move* gesture should be alleviated for developers. Thirdly, the actual software development process for tabletop interfaces is not considered entirely: For instance debugging of interaction code is difficult to achieve on actual tabletop setups as debuggers are not designed to work with tabletop setups. Thus, the development of tabletop interfaces is usually carried out with specialized tools on a standard desktop computer [8]. Therefore, it is necessary to provide means for a developer to simulate multi-touch and tangible interaction on a mouse and keyboard workplace. Section 2 introduces simulation applications and in-application simulation for multi-touch and tangible interaction via mouse and keyboard that are integrated into existing frameworks. However, these tools neither support TUI interaction nor the simultaneous manipulation of multiple fingers in different directions that is necessary to test whole-hand and other complex gestures like rotation performed with multiple fingers.

Adobe Flash [1] is a platform for animated multimedia applications for the Internet (via the browser-based Flash Player) and for desktop computing (via Adobe AIR). Flash offers cross platform support, as runtime environments are available for free for the major platforms Windows, Mac OS and Linux. Flash development in Actionscript aims at two communities: On one hand, for software developers Actionscript 3 provides an object orientated programming model similar to programming languages like Java. On the other hand, for designers Adobe provides a Flash Creative Suite application that offers WYSIWYG authoring functionality to provide graphical means for the creation of Flash content. Tabletop UIs are typically graphically rich interfaces that are usually created by designers and/or software developers. Thus, the aspects that Flash is a platform for animated multimedia applications and that its tools are tailored for developers' and designers' needs alike seem to be an ideal combination for the creation of tabletop UIs. However, an interaction toolkit that allows to use multi-touch and tangible interactivity in Flash applications is necessary. Such a framework should as well support the work of designers as of developers.

In section 2 we introduce related work. Section 3 explains the multi-touch and tangible interaction rapid prototyping toolkit TUIO AS3. Section 4 presents example projects created with TUIO AS3 and discusses the results. Finally, in section 5 a conclusion is given.

2 Related Work

There are several commercial and open source rapid prototyping and application frameworks that support the creation of multi-touch and tangible user interfaces.

reacTIVision provides a tangible tabletop toolkit [6]. It comprises the actual reacTIVision application that allows to optically track different kinds of fiducial markers, which are attached to tangibles, and (multi-)touch input and sends this data to applications via the TUIO protocol [7]. However, reacTIVision does not address the creation of actual UIs. It rather provides the necessary technical foundations to build a tabletop setup that tracks tangibles and touch input. Additionally, there is a reacTIVision TUIO simulator application that allows to simulate touch and fiducial interaction via mouse and keyboard. However, simulated tabletop interaction is not carried out directly in the application that should be tested but in the external simulator. Thus, simulation cannot be performed spatially aware as a user can only guess where a touch or tangible interaction in the simulator is performed in the actual application.

PyMT is an open source cross platform multi-touch UI toolkit [4] written in Python. PyMT offers software developers to quickly create application prototypes based on multi-touch widgets. These widgets provide access to detailed touch input data and are used to recognize several two-finger gestures like drag, scale or rotate. Touch simulation can be carried out in the actual application. However, PyMT does so far not support TUIs and real multi-touch interaction can only be simulated by using two mice (and then only two fingers can be moved simultaneously).

MT4J (Multi-Touch for Java) resembles PyMT in many ways: It offers an open source, cross platform framework for the rapid development of multi-touch applications [3]. MT4J also provides UI widgets that offer access to detailed touch input data. The main difference to PyMT is that MT4J is written in Java.

TISCH (Tangible Interactive Surfaces for Collaboration between Humans) is a cross-platform, cross-device multitouch development framework developed by Florian Echtler [2]. Similar to PyMT and MT4J, it offers multitouch widgets that in the case of TISCH are based on OpenGL. Additionally, TISCH provides a reconfigurable, hardware-independent gesture recognition engine and support for widely used (for instance move, scale, rotate) as well as custom gestures. Applications for TISCH can be developed in C++. However, there are bindings for C#, Java, Python. TISCH has similar drawbacks as PyMT and MT4J.

Although, Adobe itself offers no TUI support so far for Flash, there is an Adobe AIR multi-touch API that provides access to the Windows 7 multi-touch capabilities [9]. The AIR multi-touch API provides a low level touch event model and a gesture event model. This API only works in Windows 7 and only with multi-touch hardware that supports Windows 7 touch capabilities. The gesture event model supports only gestures provided by Windows 7. Thus, if gestural interaction should be used with the AIR multi-touch API either own gestures have to be developed based on the low level API or the high level gestures of the multi-touch API can be used that however will not work with all Windows 7 multi-touch hardware.

GestureWorks [5] is a cross platform, commercial multi-touch framework for Adobe Flash and Flex. It features a gesture library with several built-in gestures. Gestures can be used in combination with other Flash and Flex UI widgets. Additionally to code-based development, GestureWorks supports the creation of multi-touch applications by designers in the Flash Creative Suite application. Thus, GestureWorks also aims for the creation of multi-touch applications by designers. The simulation tools for GestureWorks are similar to those of PyMT, MT4J and TISCH. Therefore, GestureWorks does not support the simulation of whole-hand gestures. Also, GestureWorks does not support the creation of TUIs.

3 TUIO AS3

TUIO AS3 has been developed for several reasons. Most importantly, no toolkit was found that supports rapid prototyping of multi-touch in combination with tangible user interfaces. TUIO AS3 is based on the Adobe Flash programming language Actionscript 3 (AS3). Accordingly, TUIO AS3 has two target communities: software developers and designers. TUIO AS3 focuses on interaction rather than to provide interaction widgets, as already rich media user interface widgets like image and video panels are available for the Flash platform that should be reusable in TUIO AS3. Hence, Flash and TUIO AS3 can be used in combination to create multimedia applications with multi-touch and TUI interactivity.

As the name states, the communication with the hardware is based on TUIO [7], which is usually supported by open source tracking applications like Touchlib [10]. Additionally, Windows 7 multi-touch capabilities [9] can be used instead of TUIO.

TUIO AS3 provides a low level event model and callback system for multi-touch and tangible interaction. However, TUIO AS3 offers high-level multi-touch and tangible interaction means in order to rapidly develop prototypes. In terms of multi-touch interaction, TUIO AS3 supports as well standard interaction like dragging, rotation, scaling and flicking as complex, customized gesture interaction. In order that interaction can be performed with more than two fingers, TUIO AS3 allows the usage of whole hand interaction for standard interaction. For tangible interaction, callbacks can be registered for different tangibles in order to react when a certain tangible is put on the tabletop.

For development purposes, TUIO AS3 supports a sophisticated simulation of multi-touch and tangible input with a mouse.

3.1 TUIO AS3 Architecture

As can be seen in Figure 1, TUIO AS3 is based on multiple tiers. Interaction on a tabletop system is recognized

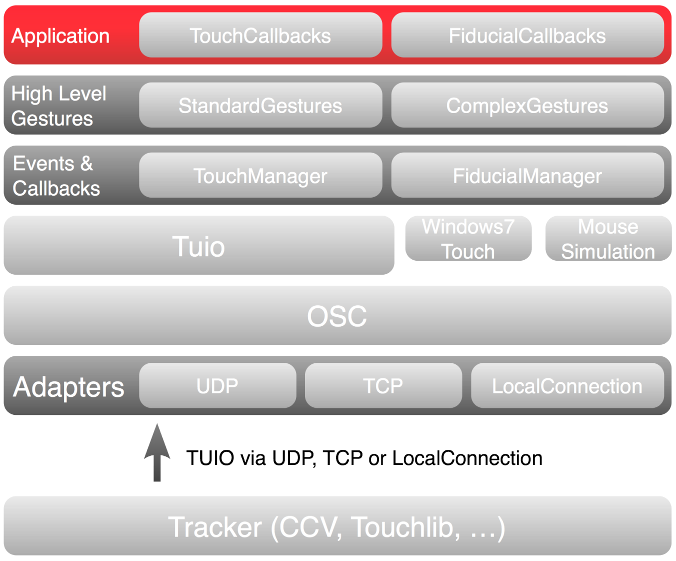

Fig. 1 TUIO AS3 architecture

by the system's *Tracker* and sent to the TUIO AS3 based application via TUIO. With TUIO AS3, TUIO connections can be physically established via one of three protocol adapters: *UDP*, *TCP* or the Flash specific exchange format *LocalConnection*. TUIO messages are based on OSC [12]. Hence, a TUIO connection is set up via *OSC* and the network protocol adapter of choice with the tracker. TUIO messages sent via this connection are handled by the *OSC* tier and forwarded to the *Tuio* tier that feeds these messages to the *Events & Callbacks* tier. Depending on how the message is a touch or a fiducial message, an appropriate event will be created by *TouchManager* or *FiducialManager* and dispatched to the actual application or to the *High Level Gestures* tier. Instead of using a TUIO connection, TUIO messages can be simulated via mouse with the *MouseSimulation* module or via Windows 7 multi-touch capabilities with the *Windows7Touch* module. However, in TUIO AS3 *Windows7Touch* is implemented as TUIO message provider which is transparent to the developer. Thus, an application based on TUIO AS3 can be used with each TUIO message provider, each Windows 7 system based on touch hardware and with mouse interaction. However, Windows 7 does not support TUI interaction and mouse simulation cannot provide the directness of touch interaction and real multi-touch interaction with a mouse is not possible.

From a development point of view, working with multi-touch differs significantly from working with mouse events, as there can be multiple touches simultaneously. Hence, a multi-touch application cannot simply listen on some kind of touch down event and afterwards on a touch move event, as both events may belong to different touches. If a touch down event is performed on a UI element the element must remember the touch event id in order to figure out if subsequent touch events like

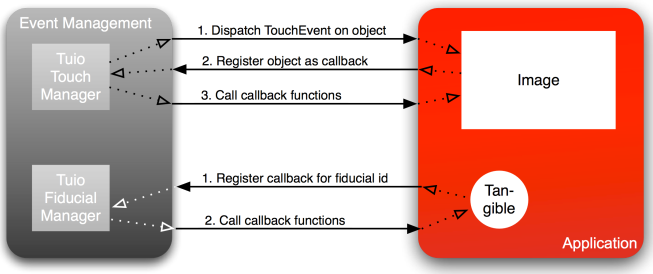
Event Management
Tuio Touch Manager
Tuio Fiducial Manager
1. Dispatch TouchEvent on object
2. Register object as callback
3. Call callback functions
1. Register callback for fiducial id
2. Call callback functions
Image
Tan-gible
Application

Fig. 2 TUIO AS3 callback system

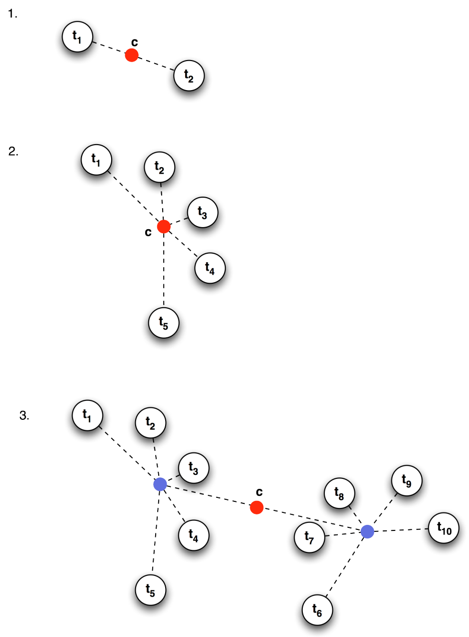

Fig. 3 Standard Gesture Interactivity

move or up belong to the same event. To simplify touch event handling, an UI element can register a callback for a touch id (*TouchCallbacks* in Figure 1, illustrated in more detail in Figure 2 in the upper half) after the initial touch down happened. Every time, the touch will be moved or eventually removed the appropriate callback function will be notified. It will be shown later in this paper that this simplification goes not far enough in terms of multi-touch interaction as the handling of multiple touches can be a tedious task for developers.

For tangible interaction, a similar approach is possible: An UI element can listen on a fiducial event and react appropriately. Additionally, TUIO AS3 supports a global callback system in which to every type of tangible callback handlers can be registered (*FiducialCallbacks* in Figure 1, illustrated in more detail in Figure 2 in the lower half). As soon as a certain tangible is put, moved, rotated or removed, the appropriate callback function will be notified.

3.2 Multi-Touch Interaction

Complementary to the event and callback system, TUIO AS3 provides a high-level interaction API that allows to enhance UI elements with standard multi-touch interactivity like dragging, rotation and scaling. Additionally, it is possible to define and track custom defined gestures like *two finger taps* or *one finger down one finger move* gestures. These gestures can be defined with a simple gesture grammar. Both kinds of interactivity are introduced in the following.

3.2.1 Standard Gesture Interactivity Drag, scale and rotate gestures are used frequently for gestural interaction with multi-touch UIs. TUIO AS3 provides simple means to provide developers and designers to enhance UI elements with this interaction metaphors.

Usually, in multi-touch gesture frameworks a drag gesture is performed by putting exactly one finger on top of an UI element and moving the finger around. Scale and rotate gestures are usually implemented as two finger gestures: For a rotate gesture two fingers on top of an UI element are turned clockwise and counter-clockwise and for a scale gesture two fingers are moved towards each other (for shrinking) or away from each other (for enlargement). However, this kind of implementation does not support interaction that is semantically related: For instance a user could use whole-hand gestures and put all fingers of left and right hand on top of an image to move the image around or to modify the image's scale. To make standard interaction as flexible as possible, TUIO AS3 makes use of the barycenter of all touches that prevail on an UI element. Flash is frame-based and the barycenter $\mathbf{c}^j$ of all touches in frame j is calculated as follows where $\mathbf{t}_i^j \in \mathbb{R}^2$ are the n^j touches in the frame j:

$$\mathbf{c}^j = \frac{1}{n^j}\sum_{i=1}^{n^j}\mathbf{t}_i^j$$

Dragging (or translation) $\mathbf{v}^j$ in the frame j can be calculated as the distance of $\mathbf{c}^j$ between successive frames $j-1$ and j:

$$\mathbf{v}^j = \mathbf{c}^j - \mathbf{c}^{j-1}$$

Scaling $s^j \in \mathbb{R}$ in frame j is calculated relatively between successive frames $j-1$ and j in relation to the average length of the vectors $\mathbf{u}_i^j = \mathbf{t}_i^j - \mathbf{c}^j$

$$s^j = \frac{\tilde{s}^j}{\tilde{s}^{j-1}}$$

where

$$\tilde{s}^j = \frac{1}{n^j}\sum_{i=1}^{n^j}\|\mathbf{u}_i^j\|$$

Accordingly, rotation $\alpha^j \in [0, 2\pi)$ can be identified by calculating the difference of the angle between $(1,0)^T$

and the vector $\mathbf{m}^j = (x^j, y^j)^T$ in two successive frames $j-1$ and j:

$$\alpha^j = \tilde{\alpha}^j - \tilde{\alpha}^{j-1}$$

where

$$\tilde{\alpha}^j = \arccos\left(\frac{x^j}{\|\mathbf{m}^j\|}\right)$$

and

$$\mathbf{m}^j = \frac{1}{n^j}\sum_{i=2}^{n^j}(\mathbf{t}_i^j - \mathbf{t}_1^j)$$

Figure 3 shows the flexibility of the standard interaction model: The first example illustrates two touches $\mathbf{t_1}$ and $\mathbf{t_2}$ that perform a standard two-finger gesture with the barycenter $\mathbf{c}$ between them. If both fingers are placed on top of an UI element, a dragging gesture can be performed by moving both fingers at once in the same direction. Scale and rotate gestures can be carried out with the usual two-finger gestures. The second example shows whole-hand interaction: By placing a whole hand on an UI element, the element can be dragged by moving the hand, scaled by spreading the fingers and rotated by turning the hand. The third example demonstrates two-handed interaction: Additionally to the barycenter between the hands, the barycenters of the single hands have been drawn for illustration purposes. Similar to two-finger interaction, dragging can be performed by moving both hands at once, scaling by moving both hands away from or towards each other and rotation by moving both hands in a circular fashion. However, other gestures are conceivable to perform such gestures. For instance a user could rotate an UI element by turning the index and middle finger of both hands on the element.

From a development point of view, standard gesture interactivity can be used with every UI element that uses the `TouchControl` provided by TUIO AS3. To a `TouchControl` another control can be added. For standard gesture interactivity a `MultifingerControl` must be added:

```
var touchControl:TouchControl =
      new TouchControl(this);
touchControl
      .addControl(new MultifingerControl());
```

If TUIO AS3 would be based on a purely event driven model (for instance if drag, scale and rotate events would be globally dispatched on UI elements) TUIO AS3 would need to calculate every potential kind of gestural interaction for every combination of touches in the application for every UI element. Hence, a `TouchControl` of each UI element administrates a touch list that can be used by other controls like the `MultifingerControl`. As interaction is only calculated locally, the necessity to globally calculate potential interaction is circumvented.

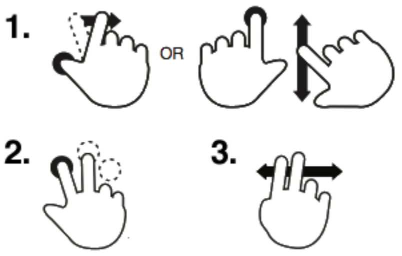

Fig. 4 Example for Complex Custom Gesture Interactivity [13]

3.2.2 Complex Custom Gesture Interactivity Gestures that should be tracked globally in an application can be registered with a `GestureManager` in TUIO AS3. Figure 4 exemplifies a few complex gestures: Number 1 shows a *one finger down one finger move* gesture, Number 2 a *two finger tap* and Number 3 a *two finger swipe* gesture.

A simple grammar allows to define own gestures: All that needs to be done is to declare the order of events for instance for a *three finger move* gesture:

1. Event: TOUCH_MOVE, containerAlias:'A'
2. Event: TOUCH_MOVE, containerAlias:'B'
3. Event: TOUCH_MOVE, containerAlias:'C'
4. Event: TOUCH_UP, containerAlias:'A', die:true
5. Event: TOUCH_UP, containerAlias:'B', die:true
6. Event: TOUCH_UP, containerAlias:'C', die:true

To perform a *three finger move* gesture, the first three rules must be satisfied: As soon as three different touches are moved simultaneously, they will be labeled with the alias A, B and C and the *three finger move* gesture starts. The gesture ends when any of the three touches A, B or C is lifted.

To alleviate the task to create own gestures, abstract base gesture classes are provided that allow to base gestures upon: For instance, a *two finger move* gesture is provided on which a two finger rotate or scale gesture could be based upon.

3.2.3 Naïve Physics Naïve physics properties can be added to an UI element by adding `FlickControl` to the element's `TouchControl`:

```
touchControl
      .addControl(new FlickControl());
```

`FlickControl` uses an UI element's momentum as a physical behavior causing an element to keep on floating after it has been released. The momentum depends on the element's velocity at the moment of release.

3.3 TUI Interaction

For tangible interaction, two approaches can be employed in TUIO AS3: On one hand, an UI element can listen on fiducial events and can react appropriately on them.

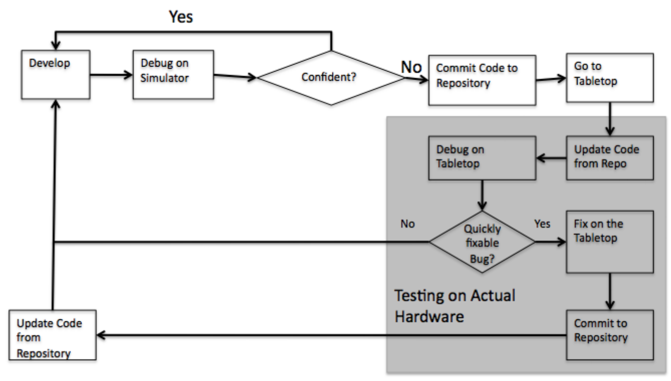

Fig. 5 Development workflow for multi-touch software [8]

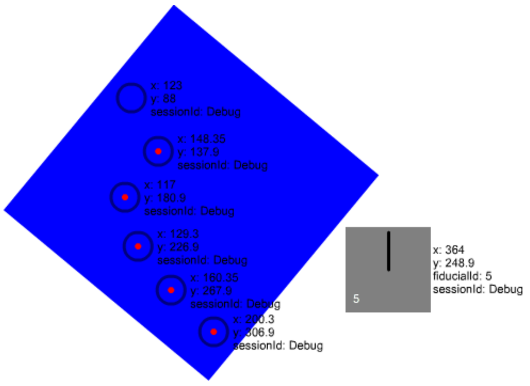

Fig. 6 Screenshot of simulation information in a TUIO AS3 based application

This means that if any tangible has been placed on top of an UI element, a `FiducialEvent` will be dispatched onto the UI element. A `FiducialEvent` contains the fiducial id (a fiducial id stands for a certain type of tangible), the position and the rotation of the tangible. On the other hand, TUIO AS3 provides a global callback approach: If an application always wants to react on a certain tangible in the same way, it can register a callback for that tangible. The TUI callback system is illustrated in Figure 2 in the lower half.

3.4 Development

Khandkar et al [8] illustrate the development workflow of tabletop applications as illustrated in Figure 5: The actual development of the application is carried out on a desktop computer where the results are tested and debugged with the help of a simulator application. As soon as a developer is confident that the results of the development are working, the code/application is transferred to the tabletop system where the application is tested and debugged. Remarkably, only small development changes are performed on the tabletop system. For bigger changes the results are transferred back to the desktop computer.

3.4.1 Tools Applications for TUIO AS3 can be developed with common Flash tools like Adobe Flash Builder or Adobe Flash CS5.

3.4.2 Simulation As testing is mainly carried out on a workspace computer, it is important to offer simulation of touch and tangible interaction as complete and intuitive as possible that is tailored to mouse and keyboard interaction. To make interaction as direct as possible, TUIO AS3 provides in-app simulation. This means that an external simulator application is not needed.

For single touch simulation, users can simply click and drag the mouse in the application. Figure 6 shows a simulated touch represented by a grey circle with additional textual debug information in the upper left. Multi-touch simulation can be achieved via shift-clicking: If the *shift* key on the keyboard is pressed while the mouse is clicked, the touch will stay in the application even after the mouse has been released making the touch permanent. Afterwards a permanent touch can be dragged around. In order to move multiple touches simultaneously, touches can be grouped by pressing the *ctrl* key while clicking on them. A grouped touch is marked with a red dot. Figure 6 illustrates five grouped touches that represent a whole-hand gesture. If one touch of a group is dragged, the whole group of touches will be moved altogether. However, this approach only allows the synchronized movement of all touches in one direction. In order to move touches in different directions for instance to simulate whole-hand interaction like scaling or rotation, additional keys can be used: If the *s* key is pressed on the keyboard while moving a group of touches, the touches will be moved perpendicular to the barycenter of the touches causing a scale gesture. If the *r* key is pressed the touches will be rotated around the barycenter causing a rotation gesture. To simulate the testing of a flicking gesture after real multi-touch interaction, pressing the *space* key while moving a group of touches will release all touches at once.

To simulate TUI interaction, a tangible can be added to an application by right clicking with the mouse and choosing a fiducial id from the context menu. A tangible is represented by a rectangle (see Figure 6 on the right). After clicking the right mouse button, tangibles can be added to the application by choosing the appropriate fiducial id from the context menu. A TUI object can be dragged around. By pressing *r* while dragging around a tangible, it will be rotated around its center. *Shift clicking* a tangible removes it.

As TUIO AS3 also supports Windows 7 touch capabilities, simulation can also be carried out on multi-touch hardware that supports these capabilities. Meanwhile, a broad spectrum of Windows 7 multi-touch hardware is available.

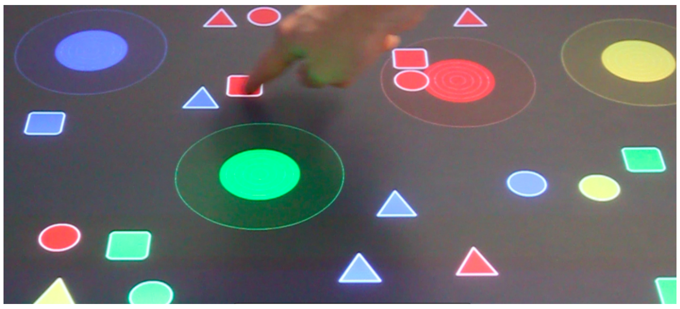

Fig. 7 Screenshot of the flicking testing application

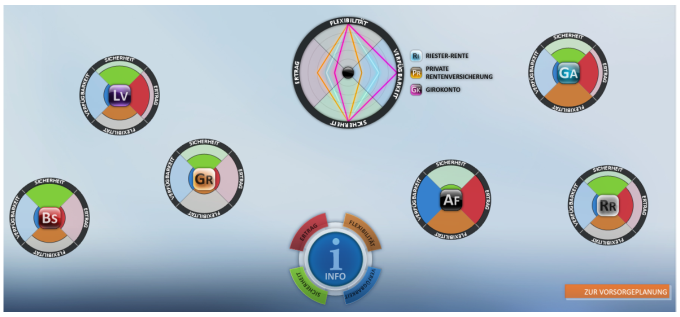

Fig. 8 Screenshot of the InfinityBank application

4 Examples and Discussion

4.1 Examples

Several projects have been developed based on TUIO AS3. Amongst others, an application that tests flicking behavior has been created (see Figure 7). This application uses the standard controls as well as the flicking control to provide the interaction that should be tested. Additionally, the application enhances TUIO AS3's naïve physics functionalities with a snapping behavior control. This means that a touchable object can be enhanced with the property to attract other touchable objects that enter the surrounding of the object. This example shows how the interaction API of TUIO AS3 can be used to use basic interactivity in addition with the creation of new interactivity.

The *InfinityBank* project is another example for a TUIO AS3 based application (see Figure 8). It can be used by a bank counselor to plan a customer's retirement provisions collaboratively with the customer itself. Amongst others, InfinityBank demonstrates how sophisticated visualization widgets can be used in combination with TUIO AS3's interaction API.

4.2 Discussion

TUIO AS3 offers a toolkit for rapid prototyping for tabletop interaction. Additionally to an architecture for the translation of TUIO messages into an event and callback system, TUIO AS3 offers a high level interaction API and simulation tools. Both will be discussed in the following.

4.2.1 Interaction The gestural interaction API for multi-touch supports two approaches: On one hand interaction with the standard gestures drag, scale, rotate and flick are supported. On the other hand, there is a complex gesture API that can be enhanced by a developer.

While both interaction models are necessary, only one interaction model can work with one UI element at a time. If for instance an UI element is draggable, scalable and rotatable via the standard interaction API, it is currently not possible to apply a complex gesture like a *three finger scroll* gesture to it. The element would interpret it as a dragging gesture because it would be sensed as a drag gesture by the standard interaction API that calculates any number of fingers as a drag, scale and rotate gesture in order to allow a user every gesture approach as standard interaction. In future implementations it is conceivable that the amount of fingers for the performing of standard gestures can be constrained by the developer in order to allow additional complex gestures on an UI element.

The TUI API allows a developer to use tangible interaction either with an event model or with a callback system. The event model might be more suitable for local interaction, as the event is directly dispatched onto the UI element underneath the tangible. Thus, tool tangibles are conceivable like a color correction tangible that can be placed and used directly on top of an image. The callback system could be applied for tabletop-wide interaction: A callback can register itself for a certain tangible when the application starts in order to provide a global reaction when the appropriate tangible is placed on the tabletop. This could be used for instance for a global UI element like a semantic magnet that attracts all red elements.

4.2.2 Development and Simulation Multi-touch simulation via mouse enhances existing approaches with new concepts like real simultaneous multi-touch interaction. Thus, also gestures like scaling and rotation are supported by allowing to move grouped touches in different directions at once by combining keyboard and mouse input.

However, currently only one group of touches is supported. If for example two-handed interaction should be supported, at least two groups of touches would be necessary that needed to be moved at once. The actual multi-touch input differs between mouse simulation and real multi-touch interaction: On a multi-touch setup touches are constantly moving a bit, no two gestures are performed exactly in the same way and gestures are not carried out as precise as with mouse and keyboard simulation. For instance with TUIO AS3's standard gesture implementation, a rotation gesture cannot be performed without additionally causing a weak scaling gesture and without causing a serious dragging gesture on a tabletop system. With the current simulation every gesture is simulated separately from each other. One approach could be to add some kind of fuzziness to the mouse simulation input. Alternatively, standard and complex gestures could be recorded on a multi-touch setup and replayed in the simulator for testing purposes. For instance, if developers wanted to test a rotation gesture on an UI element, they would choose *rotate gesture* from a menu and a rotate gesture from a set of different recorded rotate gestures would be performed on the element.

Tangible simulation via mouse has not been possible so far and allows to test basic tangible interaction like dragging and rotation of tangibles. Currently, the adding of tangibles is rather technical: To add a tangible to a running application, the appropriate fiducial id of the tangible must be chosen from a context menu. As different tracking systems allow to use a lot of fiducial ids (for instance reacTIVision supports up to 216 fiducials by default) choosing the appropriate id from the menu can take a while. Additionally, a developer has to know the fiducial id to every tangible of the application. As the callback system for tangible interaction is conceived as a system where each kind of tangible has its own callback class, it would make sense to list the callback names in the context menu to which fiducial ids are registered in the system in order to provide quick access to relevant tangibles with a meaningful name.

5 Conclusion

TUIO AS3 offers a toolkit for the rapid development of multi-touch and tangible user interface (TUIs) interaction in Adobe Flash. TUIO AS3's architecture supports TUIO messaging, the Windows 7 touch capabilities and mouse interaction. The toolkit comprises APIs for multi-touch and tangible interaction and multi-touch and tangible simulation via mouse and keyboard for development purposes. TUIO AS3's multi-touch API offers a low level event- and callback-based interaction API and a high level gestural interaction API that supports standard gestures like drag, scale, rotate and flicking and complex gestures like *three finger scroll* gestures or *one finger down one finger move* gestures. The standard gesture system calculates the appropriate interaction of an arbitrary amount of fingers on an UI element. Thus, additionally to common two-finger gestures, also whole-hand and two-hand interaction can be performed to drag, scale and rotate an UI element. The complex gesture system already supports several gestures and it can be enhanced with additional gestures by using a simple grammar or by enhancing existing gestures. The TUI API supports an event-based and a callback-based development of tangible interaction. Multi-touch and TUI input can be simulated with mouse and keyboard directly in a TUIO AS3 based application without the need for an additional simulator application. TUIO AS3 allows to simulate whole-hand rotation and scaling gestures by moving groups of touches simultaneously in different, appropriate directions. TUI simulation supports standard manipulation like dragging and rotation of tangibles.

References

1. Adobe Systems. Animation software, multimedia software — Adobe Flash Professional CS5. http://www.adobe.com/products/flash.
2. Echtler, F. Library for Tangible Interactive Surfaces for Collaboration between Humans. http://tisch.sourceforge.net/.
3. Fraunhofer-Institute for Industrial Engineering. MT4j - Multitouch for Java. http://www.mt4j.org.
4. Hansen, T. E., Hourcade, J. P., Virbel, M., Patali, S., and Serra, T. PyMT: a post-WIMP multi-touch user interface toolkit. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces* (New York, NY, USA, 2009), ITS '09, ACM, pp. 17–24.
5. Ideum. GestureWorks - Multitouch Authoring for Flash and Flex. http://gestureworks.com.
6. Kaltenbrunner, M., and Bencina, R. reacTIVision: a computer-vision framework for table-based tangible interaction. In *TEI '07: Proceedings of the 1st international conference on Tangible and embedded interaction* (New York, NY, USA, 2007), ACM, pp. 69–74.
7. Kaltenbrunner, M., Bovermann, T., Bencina, R., and Costanza, E. TUIO - A Protocol for Table Based Tangible User Interfaces. In *Proceedings of the 6th International Workshop on Gesture in Human-Computer Interaction and Simulation (GW 2005)* (2005), pp. 1–5.
8. Khandkar, S. H., Sohan, S. M., Sillito, J., and Maurer, F. Tool support for testing complex multi-touch gestures. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces* (2010).
9. Kiriaty, Y. MultiTouch Capabilities in Windows 7. http://msdn.microsoft.com/en-us/magazine/ee336016.aspx, 2009.
10. nuigroup. Touchlib - Home, October 2008.
11. Shen, C., Ryall, K., Forlines, C., Esenther, A., Vernier, F. D., Everitt, K., Wu, M., Wigdor, D., Morris, M. R., Hancock, M., and Tse, E. Informing the Design of Direct-Touch Tabletops. *IEEE Computer Graphics and Applications 26*, 5 (2006), 36–46.
12. Wright, M., Freed, A., and Momeni, A. OpenSound Control: state of the art 2003. In *NIME '03: Proceedings of the 2003 conference on New interfaces for musical expression* (Singapore, Singapore, 2003), National University of Singapore, pp. 153–160.
13. Wroblewski, L. Lukew - touch gesture reference guide. http://www.lukew.com/ff/entry.asp?1071, 2010.
14. Wu, M., and Balakrishnan, R. Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. In *UIST '03: Proceedings of the 16th annual ACM symposium on User interface software and technology* (New York, NY, USA, 2003), ACM, pp. 193–202.