Task-Based Focus and AdHoc-Focus-Territory—Novel Concepts for Shared Interactive Surfaces

Mirko Fetter, David Bimamisa, Tom Gross
Human-Computer Interaction Group
University of Bamberg
96045 Bamberg
<firstname.lastname>(at)uni-bamberg.de

Abstract
Shared Interactive Surfaces allow co-located users to collaboratively work on a task. As current technology often is not able to distinguish between different users, there is a potential for concurrent conflicting actions of multiple users, leading to unwanted results and accordingly frustration. With our concepts for *Task-Based Focus* and *AdHoc-Focus-Territory* we provide light-weight solutions - integrated in our toolkit TUIOFX - for designers of multi-user, multi-touch applications. Our solution helps to overcome some of the problems of *anonymous touch input*, without an immediate need for more heavy-weight mechanisms like user identification.

Keywords
Shared Interactive Surfaces; Multi-User; CSCW

ACM Classification Keywords
H.5.2 [Information Interfaces and Presentation]: User Interfaces —Input devices and strategies

General Terms
Human Factors

Introduction
Shared Interactive Surfaces (SIS) provide new means for collaborative work of co-located users in the form of

large tabletops or interactive walls. Conceptually, they advance the idea of Single Display Groupware (SDG) [18, p. 286], which "enable co-present users to collaborate via a shared computer with a single shared display and simultaneous use of multiple input devices". However, they also share some of the challenges such systems brought with them, like possible conflicts that arise when users "attempt simultaneous incompatible actions" [18, p. 290].

While a SDG typically has several mice attached (i.e. one for each user) a SIS gives each user the possibility to interact directly with the surface via multi-touch input. Thereby one severe limitation of most current off-the-shelf technology is the restriction that—while they can detect touch points, they cannot keep track of the users who instantiated these touches. In the following we refer to this as *anonymous touch input.* This introduces new challenges for users, but foremost for application developers and their interaction design. Among these challenges are access control, the provision of personalised data and interfaces, and independent input sequences [15]. To deal with this, current solutions often restrict the user freedom (e.g., fixed territories) or require additional hardware (e.g., user identification, user distinction).

In this paper, we propose two new concepts for multi-user, multi-touch Widget-Toolkits, namely *Task-Based Focus* and *AdHoc-Focus-Territory,* which enable application developers to design multi-user applications that reduce the number of input conflicts without restricting the freedom of the users. In the following we provide a thorough understanding of the current challenges and the related work, and give a detailed explanation of our concepts and hints on their implementation.

Background

In single-user systems or mouse-based multi-user systems like SDG each input event is delivered on a dedicated channel registered to one user. Currently available technologies for detecting touch on shared interactive surfaces (e.g., Frustrated Total Internal Reflection (FTIR), Diffused Illumination (DI), In-Cell Touch) often do not allow to link input events to their originator. Such *anonymous touch input* introduces a variety of challenges for the developers of multi-user applications, because it is impossible to determine if a series of inputs originates form one or multiple users.

Input Sequences and Focus

One specific problem is that the predominant interaction style in UIs is based on sequences of input events from single users. However, when multiple users simultaneously interact with a system though *anonymous touch input*, the sequentiality of events for one user cannot be obtained. For example, to select something from a drop-down menu, a user would click to expand the menu, eventually scroll through the list of items, and then select one—which automatically collapses the menu. This mechanism relays on the premise of a single *focus*, assuming there is only one element receiving input at a time [1]. To continue the example: If instead of selecting an item, the user would decide to interact with a different UI element, the pop-down menu would loose the focus (i.e. a *blur* event) and also collapse. This *blur* mechanism is also used in other cases. Imagine the user made a rectangle selection of multiple icons in a file explorer. Starting a new selection or just clicking in the background causes a *blur* event and removes the previously made selection. Single-user multi-touch devices like smartphones or tablets use the *focus* and *blur* events on text fields and other controls to respectively show or hide the virtual keyboard. This mechanisms of *click-to-focus* and *blur* are elementary to many UIs—and hence our mental models—and so define our expectations on how such controls work.

However, this single *focus* and *blur* model cannot be maintained in a multi-user environment with *anonymous touch input*. When it is unclear whether the same or a different user produced two subsequent inputs, the intention becomes ambiguous. If we rethink our previous examples, it is unclear what should happen when a second touch occurs on a second drop-down menu, while the first one is still open. Is it the same user, and should the old one be closed? Is it a different user, and both should be open? In the same way it is ambiguous if for a selected text input ideally a new virtual keyboard should be displayed for a second user, or if it is the same user, just proceeding with filling out multiple text fields. And for the selection of multiple icons, every interaction by a second user with the file explorer would remove the selection, if the single focus model of most widget toolkits would be maintained. In the following we look at what solutions to this problem so far were conceived in the related work.

Related Work

While one of the first commercial available shared interactive surfaces, the MERL DiamondTouch table [2], was able to distinguish input sequences from different users, based on capacitive coupling and receiver mats on the users' chairs, the approach was a little limited due to the fixed positions of the users. Since then, the now predominant optical approaches lost the ability to identify - or at least distinguish between - different users. In the following we highlight different approaches to deal with this problem.

User Identification

One approach to overcome this limitation is the idea to superimpose some form of *user identification* that can instantaneously link individual anonymous touch input to users. For example the IR Ring [14] is worn on the user's hand and emits a pulsed light pattern that can be tracked by a FTIR table. Nearby touches then are assumed to be originated by the person wearing this IR Ring. Marquardt et al. [7] used gloves tagged with fiducial markers that are detectable by the Microsoft Surface table. There are many other approaches, including the use of objects like a mobile phone as a proxy (MobiZone [12]), the assignment of touches to users based on the position of their shoes (Bootstrapper [13]), or the use of electromagnetic sensing at the users wrist to detect interaction with an interactive tabletop (EMSense [6]). In all cases, additional hardware and highly customised software is needed. Further these approaches also lead to very specialised setups that often come with a high configuration effort.

Territories

Another approach is the use of territories [11,17]. The basic idea is to design applications in a way that they allocate a fixed screen estate—a personal territory—to a single user. In these personal territories it can be fairly assumed that only one single user generates all touch input. Some approaches even combine the ideas of territories with those of user identification, like for example Bodylenses [5] or IdLenses [16]. They either use the body or hand shape to distinguish users and generate adhoc territories linked to this shape. However, territories often limit the freedom of users in the intensive parallelised interaction style that SIS originally promise [4].

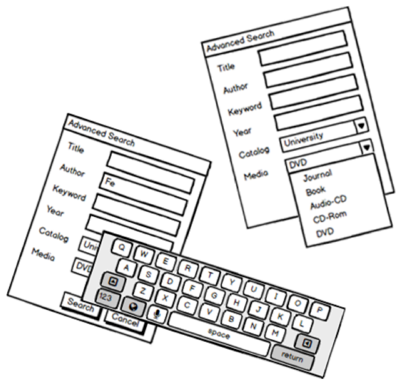

Figure 1. Two instances of a search dialog, each with a task-based focus, allowing optimal parallel input by two users.

Toolkits and Other Approaches
Most existing widget-toolkits for SIS only provide rudimentary support for multiple users. For example, the popular Surface SDK allows only one single virtual keyboard as it inherits WPF's single-focus input restriction [8]. The `TextEditWidget` of the Cornerstone SDK [9] provides an attribute `groupName` that allows assigning text fields to a group that share one virtual keyboard, which at least gives the developers some freedom. Specific challenges, like the problem of multiple selections for multiple users, are also addressed by some research, but in a more isolated manner (e.g., in [10,12]) and without the integration into a toolkit.

Concept

In the following we present the concepts for *Task-Based Focus* and *AdHoc-Focus-Territory,* helping to maintain input sequences of multiple users under the presence of *anonymous touch input.* The basic assumption is that either maintaining a single-focus model or removing the focus concept completely, both lead to unwanted and unexpected results for the users. An according solution would be a multi-foci model, which provides each user with a focus. In Table 1 we can see the number of foci needed for various setups is ideally based on the number of users (U). In Single User-Systems this number can be assumed to be one. In mouse based multi-user systems like SDG, the number of pointers (P) allows to derive the number of users. In multi-user systems with *anonymous touch input* it is impossible to come to an optimal number of *foci* and to continually assign the correct focus to the correct user.

Table 1. The number of foci (F) needed for various setups. While in most setups F is easily derivable, from the number of users (U) or indirectly from the number of pointers (P) for the SDG, the only clue in the SIS setup is an estimate of the simultaneously carried out tasks (T), which is equal or lower the number of users.

	Single User		*Multi-User*	
	Desktop Computer	*Multi-Touch Device*	*Single Display Groupware (SDG)*	*Shared Interactive Surface (SIS)*
No. of Users	1	1	$U \mid 2 \leq U \leq 10$	$U \mid 2 \leq U \leq 10$
No. of Pointers	1	$P \mid P \leq 10$	P=U*1	$P_1 + P_2 + \ldots + P_u \mid P_i \leq 10$
No. of Tasks	1	1	$T \mid T \leq U = P$	$T \mid T \leq U$
No. of Foci	**1**	**1**	**F=P=U**	**F=T**

Task-Based Focus
In our concept of *Task-Based Focus* we therefore assume that even though multiple users are interacting with a SIS synchronously on a common group task, this group interaction is still compromised of smaller individual tasks (i.e. subtasks) that are carried out independently by the users. Such individual tasks for example are entering an address, configuring an advanced search, or editing the properties of an item—all tasks that are consisting of filling out multiple text fields and other form elements. For all the UI elements that belong to such an individual task, we can assume that only one user simultaneously is interacting and therefore a single focus can be assigned to this group of elements.

Such a grouping we call a *focus area*. All elements in a focus area share one focus. Accordingly the behaviour of drop-down menus inside this area is as a user would expect it from a single-user system. Further, the elements are not affected by or affecting controls outside this focus area. In this sense also all text input elements in this focus area share one virtual keyboard, that is visible or not based on which element currently holds the focus in this focus area (cf. Figure 1).

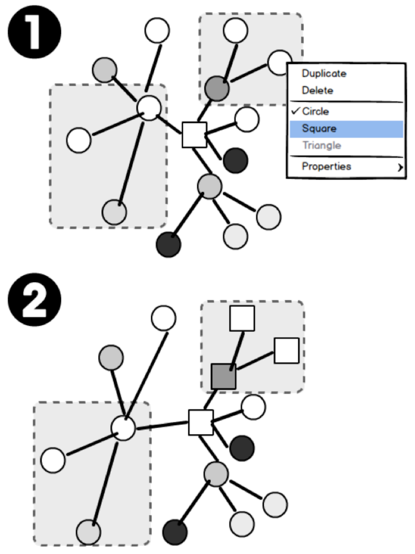

Figure 2. The AdHoc-Focus-Territories through multi-selection allow two users to work in parallel on a subset of a composed artefact. While one user changes the properties of some elements via a context menu (1), a second user moves three selected elements (2) by directly dragging them.

AdHoc-Focus-Territory

The *AdHoc-Focus-Territory* is a special case of the *Task-Based Focus*. In some cases, it is difficult to group a number of UI elements together to form a focus area. This for example happens in environments were operations are directly invoked on a selection of multiple objects. For example, a UML-Editor or CAD-programme, where users work on different parts of a bigger structure. In our concept of the *AdHoc-Focus-Territory* we therefore depart from the understanding that such a multi-selection leads to the adhoc generation of a transient *Task-Based Focus* area. If a user performs a selection (cf. Figure 2.) we instantly generate a focus area, in the size of the selection. In order to make this visible to the user, the rectangular marquee selection stays visible. All actions inside this rectangular are treated with a single focus. Any interaction outside of the rectangular has no effect on the selection and the selected items. This way, other users can also generate *AdHoc-Focus-Territories* in parallel. Inside the territory, the user now can manipulate the group of selected objects directly (e.g., move them) or via a context menu (e.g., delete them). Once finished with the task, the user touches the background inside the rectangular, resulting in a *blur* event instantly removing the territory. Similar mechanisms for selections of cells in a table or parts of a text by multiple users are conceivable.

Both concepts provide the designers of applications new means to handle input from multiple anonymous users, without being too restrictive. Our rationale was to stay close to the mental models users already have from single-user interfaces. This way the users often won't even recognise, or at least quickly pickup these new concepts. Of course, it requires an interaction designer that has a clear concept of the users' tasks.

Implementation in TUIOFX

Both concepts are integrated in and tested with the TUIOFX Toolkit [3], a JavaFX toolkit for developing multi-user, multi-touch applications.

The *Task-Based Focus* is implemented in form of a simple boolean property (`focusArea`) which can be assigned by the developer to any JavaFX `Node`. Any children of this node will then belong to the same *focus area,* and share a single *Task-Based Focus*. JavaFX holds all UI elements (i.e. nodes) in a tree-like structure (i.e. the scene graph). Accordingly each element can easily infer to which *focus area* it belongs. In order to make the UI elements behave according to the *Task-Based Focus*, we adapted the behaviour of all standard widgets through extensive skinning. Firstly, this required making all widgets agnostic to Java's internal focus model—which is single-focus. On an individual widget level, all `PopupWindow`-based controls (`ComboBox`, `ChoiceBox`) for example, should only collapse when a touch event occurs within the focus area associated to that `PopupWindow`. To achieve that, we removed the standard closing behaviour and added an event filter. The filter is listening for all events routed through the respective focus area's root node. Only when a new touch event is received on any of the other child nodes, the menu is closed. Furthermore, the text input controls (`TextField` and `TextArea`) are changed in order to support parallel writing. By extending `TextFieldSkin` and `TextAreaSkin` we are able to receive user text inputs from attached TUIOFX virtual keyboards, even though they lose Java's internal focus. Additionally, hiding and displaying the keyboard is based on the same event filter model like for the `PopupWindow`. Finally, also the visual styling is adapted, that when a text field looses Java's single in-

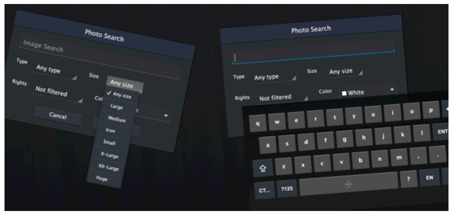

Figure 3. Two dialogs with task-based focus allowing one user to select from a drop-down menu while a second user can enter text in a second form.

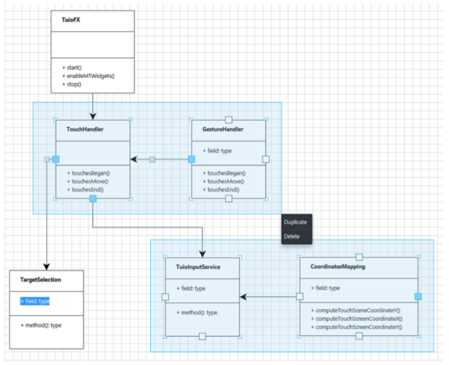

Figure 4. Two lightblue AdHoc-Focus-Territories (top one with a open context menu) allow two users to work with multiple selected items in parallel, while a third user at the same time can alter the text of an individual item (blue highlighted text) in this collaborative UML Editor.

ternal focus, it still shows a caret and is highlighted, as long it has the focus of the focus area. All these changes are made under the hood of TUIOFX, that a JavaFX developer can use the standard JavaFX API and only needs to assign the `focusArea`–property as needed.

The *AdHoc-Focus-Territory* is more difficult to generalise and therefore so far is only partially integrated in the TUIOFX toolkit. As selection mechanisms for selecting multiple items often are very application specific (think of lasso selection, rectangular selection, selection of part of a text or table cells, etc.), it is almost impossible to provide valuable default behaviours on a toolkit-level. We up to now started implementing the *AdHoc-Focus-Territory* concept for some assorted widgets like `TextArea` and *`TableView`.*

To verify these concepts, we developed several demo applications. In Figure 3 you can see two instances of the same search dialog, each with a *Task-Based Focus* assigned from a collaborative moodboard application. We see that, while the dialog on the left has an open drop-down menu the dialog on the right has the focus on a text field and an open virtual keyboard. The interaction in each dialog resembles the expectations of a user from a single user interface, as the controls inside each dialog affect each other but not between two dialogs. Figure 4 shows a part of a collaborative UML- Editor that relies on the *AdHoc-Focus-Territory*. We can see that two users have made a selection, and now both are able to interact with the items inside the territories, while the rest of the model stays unaffected. A third user can even work on another part of the model, and directly rename an object. To remove the territory, the user simply needs to click in the background inside the blue rectangular visualising the territory.

Conclusion

In this paper we presented the concepts for *Task-Based Focus* and *AdHoc-Focus-Territory.* While our approach does not address problems like access control or personalised data, it aims to tackle a small subset of the problems related to *anonymous touch input*. In these cases it gives the developer of multi-user, multi-touch applications new means for their application design for of-the-shelf hardware and without the need for more heavy-weight approaches like user identification. Further, by building on mental models users already have from single user systems, we aim to improve the user experience.

In future work, extensive studies need to prove if these concepts work fluently. That is, if users easily understand the interaction principles, if they help to minimise conflicts among users, and if they are widely applicable in different scenarios. Therefore, we are interested in feedback from developers using both concepts with TUIOFX, to deepen our understanding of where these concepts work, and where they might be challenging.

Acknowledgements

We thank the members of the Cooperative Media Lab.

References

[1] Hrvoje Benko, Shahram Izadi, Andrew D Wilson, Xiang Cao, Dan Rosenfeld and Ken Hinckley. Design and Evaluation of Interaction Models for Multi-touch Mice. In *Proceedings of Graphics Interface 2010 - GI 2010* (May 31 - Jun 2, Ottawa, Canada). Canadian Information Processing Society, Toronto, Canada, 2010. pp. 253-260.

[2] Paul Dietz and Darren Leigh. Diamondtouch: A Multi-User Touch Technology. In *Proceedings of the 14th Annual ACM Symposium on User Interface*

Software and Technology - UIST 2001 (Nov. 11-14, Orlando, FL, USA). ACM Press, New York, NY, USA, 2001. pp. 219-226.

[3] Mirko Fetter and David Bimamisa. TUIOFX—Toolkit Support for the Development of JavaFX Applications for Interactive Tabletops. *In Proceedings of the 15th IFIP TC.13 International Conference on Human-Computer Interaction - INTERACT 2015* (Sept. 14-18, Bamberg, Germany). Springer, Heidelberg, Germany, 2015. pp. 476-479.

[4] Mirko Fetter, Tom Gross and Maxi Hucke. Supporting Social Protocols in Tabletop Interaction through Visual Cues. In *Proceedings of the Thirteenth IFIP TC.13 International Conference on Human-Computer Interaction - INTERACT 2011* (Sept. 5-9, Lisbon, Portugal). Springer, Heidelberg, 2011. pp. 435-442.

[5] Ulrike Kister, Patrick Reipschlaeger, Fabrice Matulic and Raimund Dachselt. BodyLenses: Embodied Magic Lenses and Personal Territories for Wall Displays. In *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces - ITS 2015* (Nov. 15-18, Madeira, Portugal). ACM Press, New York, NY, USA, 2015. pp. 117-126.

[6] Gierad Laput, Chouchang Yang, Robert Xiao, Alanson Sample and Chris Harrison. EM-Sense: Touch Recognition of Uninstrumented, Electrical and Electromechanical Objects. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software and Technology - UIST 2015* (Oct. 8-11, Charlotte, NC, USA). ACM Press, New York, NY, USA, 2015. pp. 157-166.

[7] Nicolai Marquardt, Johannes Kiemer and Saul Greenberg. What Caused That Touch? Expressive Interaction with a Surface Through Fiduciary-tagged Gloves. In *Proceedings of the 2010 International Conference on Interactive Tabletops and Surfaces - ITS 2010* (Nov. 7-10, Saarbrücken, Germany). ACM Press, New York, NY, USA, 2010. pp. 139-142.

[8] Microsoft Corporation. The Microsoft Surface 2.0 SDK. *http://www.microsoft.com/en-us/pixelsense/ SoftwarePlatform.aspx,* 2014. (Last accessed: 5/6/2015).

[9] MultiTouch Ltd. Cornerstone SDK. *http://cornerstone.multitouch.fi/developer_guide,* 2015. (Last accessed: 08/01/2016).

[10] Chris North, Tim Dwyer, Bongshin Lee, Danyel Fisher, Petra Isenberg, George Robertson and Kori Inkpen. Understanding Multi-touch Manipulation for Surface Computing. In *Proceedings of the IFIP TC13 International Conference on Human-Computer Interaction - INTERACT 2009* (Aug. 24-28, Uppsala, Sweden). Spinger, Berlin/Heidelberg, Germany, 2009. pp. 236-249.

[11] David Pinelle, Mutasem Barjawi, Miguel Nacenta and Regan Mandryk. An Evaluation of Coordination Techniques for Protecting Objects and Territories in Tabletop Groupware. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2009* (Apr. 4-9, Boston, MA, USA). ACM Press, New York, NY, USA, 2009. pp. 2129-2138.

[12] Markus Rader, Clemens Holzmann, Enrico Rukzio and Julian Seifert. MobiZone: Personalized Interaction with Multiple Items on Interactive Surfaces. In *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia - MUM 2013* (Dec. 2-5, Lulea, Sweden). ACM Press, New York, NY, USA, 2013. pp. 8:1-8:10.

[13] Stephan Richter, Christian Holz and Patrick Baudisch. Bootstrapper: Recognizing Tabletop Users by their Shoes. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2012* (May 5-10, Austin, TX, USA). ACM Press, New York, NY, USA, 2012. pp. 1249-1252.

[14] Volker Roth, Philipp Schmidt and Benjamin Gueldenring. The IR Ring: Authenticating Users' Touches on a Multi-touch Display. In *Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology - UIST 2010* (Oct. 3-6, New York, NY, USA). ACM Press, New York, NY, USA, 2010. pp. 259-262.

[15] Dominik Schmidt. Instantaneous User Identification for Personalized Interaction on Shared Surfaces. Lulu Press, Inc., Raleigh, NC, USA, 2012.

[16] Dominik Schmidt, Ming Ki Chong and Hans W Gellersen. IdLenses: Dynamic Personal Areas on Shared Surfaces. In *Proceedings of the 2010 International Conference on Interactive Tabletops and Surfaces - ITS 2010* (Nov. 7-10, Saarbrücken, Germany). ACM Press, New York, NY, USA, 2010. pp. 131-134.

[17] Stacey D. Scott, M. Sheelagh T Carpendale and Kori M Inkpen. Territoriality in Collaborative Tabletop Workspaces. In *Proceedings of the 2004 ACM Conference on Computer-Supported Cooperative Work - CSCW 2004* (Nov. 6-10, Chicago, IL, USA). ACM Press, New York, NY, USA, 2004. pp. 294-303.

[18] Jason Stewart, Benjamin B. Bederson and Allison Druin. Single Display Groupware: A Model for Co-present Collaboration. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 1999* (May 15-20, Pittsburgh, PA, USA). ACM Press, New York, NY, USA, 1999. pp. 286-293.