First publ. in: Proceedings of the Workshop on the Challenges of Engineering Multimodal Interaction: Methods, Tools, Evaluation 2009

Visual Design of Multimodal Interaction - Bridging the Gap between Interaction Designers and Developers

Werner A. König, Roman Rädle, and Harald Reiterer
HCI-Group, University of Konstanz
Box-D73, Universitätsstraße 10, 78457 Konstanz
{koenigw, raedler, reiterer}@inf.uni-konstanz.de

ABSTRACT

In contrast to the pioneers of multimodal interaction e.g. Richard Bolt in the late seventies, today's researchers can benefit of a wide variety of existing interaction techniques, devices and frameworks. Although these tools are available, the usage of them is still a great challenge particularly in terms of usability. A major issue results from the trade-off between the functionality of the system and the simplicity of use. We introduce a novel visual user interface concept which is especially designed to ease the design and development of post-WIMP user interfaces including multimodal interaction. It provides an integrated design environment for our interaction library "Squidy" based on high-level visual data flow programming combined with zoomable user interface concepts. The user interface offers a simple visual language and a collection of ready-to-use devices, filters and interaction techniques. We specifically address the trade-off between functionality and simplicity by utilizing the concept of semantic zooming which enables dynamic access to more advanced functionality on demand. Thus, developers as well as interaction designers are able to adjust the complexity of the Squidy user interface to their current need and knowledge.

VISUAL INTERACTION DESIGN WITH SQUIDY

In the last years diverse frameworks evolved which offer great opportunities to implement and realize novel interaction techniques in the field of multimodal interaction and post-WIMP user interfaces [8]. These frameworks such as ICON Input Configurator [1], its successor MaggLite [2], the OpenInterface Framework [7], MAX/MSP [5] and vvvv [9] provide high flexibility and great functionality, but interaction designers with less or no programming experience are mostly overstrained by the complexity of these tools and their user interfaces. However interaction designers have to iteratively test, improve and evaluate their techniques especially when doing research.

We address this issue with our interaction library "Squidy" which unifies various device toolkits and frameworks in a common library and provides an integrated user interface for visual dataflow management as well as device and data filter configuration. Squidy thereby hides the complexity of the technical implementation from the user by providing a simple visual language and a collection of ready-to-use devices, filters and interaction techniques. This facilitates rapid prototyping and fast iterations during the process of design and development. However, if more functionality and profound customizations are required, the visual user interface reveals advanced information and operations on demand by using the concept of semantic zooming. Thus, users are able to adjust the complexity of the user interface to their current need and knowledge (ease of learning).

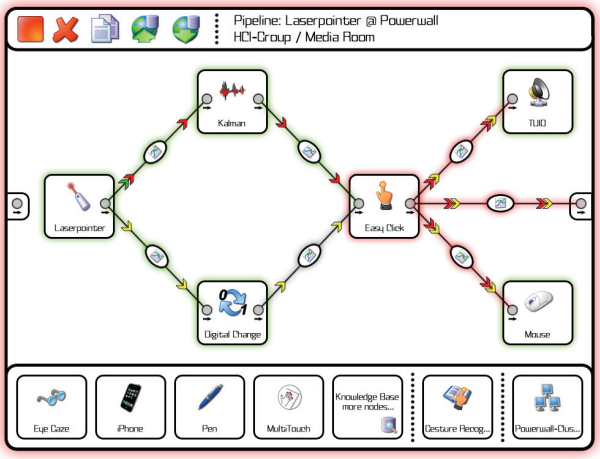

Figure 1: View of a simple pipeline in Squidy. The pipeline receives position, button, and inertial data from a laser pointer, applies a Kalman filter, a filter for change recognition and a filter for selection improvement and finally emulates a standard mouse to interact with conventional applications. At the same time the data is sent via TUIO to listening applications. The pipeline-specific functions and breadcrumb navigation are positioned on top. The zoomable knowledge base with a selection of recommended input devices, filters, and output devices are located at the bottom.

The basic concept which enables the visual definition of the dataflow between the input and output is based on a pipe-and-filter concept (Figure 1). This offers a very simple but powerful visual language to design the interaction logic. The user thereby selects the input device or hardware prototype of choice as "source", connects it successively with filter nodes for data processing such as compensation of hand tremor or gesture recognition and routes the refined data to the "sink". This can be an output device such as a vibrating motor for tactile stimulation or LEDs for visual feedback. Squidy also provides a mouse emulator as output node to offer the possibility to control standard WIMP-applications with unconventional input devices. Multipoint applications (e.g. for multi-touch surfaces or multi-user

environments) and remote control are supported by an output node which transmits the interaction data as TUIO messages over the network. TUIO is a widely used protocol for multipoint interaction based on OSC. The internal dataflow between the nodes in Squidy consists of a stream of single or multiple grouped data objects of well-defined data types based on the primitive virtual devices introduced by Wallace [10]. In contrast to the low-level approaches used in related work, such abstracting and routing of higher-level objects has the advantage that not every single variable has to be routed and completely understood by the user. The nodes transmit, change, delete data objects, or generate additional ones (e.g. if a gesture was recognized).

Semantic Zooming

According to the assumption that navigation in information spaces is best supported by tapping into our natural spatial and geographic ways of thinking [6] we use a zoomable user interface concept to navigate inside the Squidy visual user interface. When zooming into a node, additional information and corresponding functionalities appear, depending on the real estate available (semantic zooming). Thus, the user is able to gradually define the level of detail (complexity) according to the current need for information and functionality.

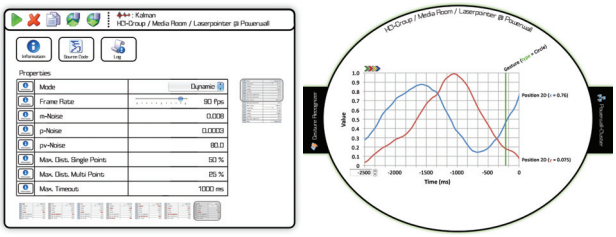

Figure 2: The left picture (a) shows the revealed parameters of a zoomed node. This view allows direct manipulation of the node parameters and thus an interactive adjustment of values at run-time. Dataflow visualization is placed at the right (b). It allows fast access to the current dataflow between two nodes. Users benefit from the insight into the interaction dataflow.

Interactive Configuration & Evaluation

In contrast to existing frameworks the user does not have to leave the visual interface and switch to additional applications and programming environments in order to get additional information, to change properties, or to generate, change or just access the source code of device drivers and filters. In Squidy, zooming into a node reveals all parameters and enables the user to interactively adjust the values at run-time (Figure 2a). The changes take place immediately and thus neither requires a restart nor a recompilation of the source code. This is especially beneficial for empirically testing different suitable parameters (e.g. adjusting the noise levels of a Kalman filter) because of the possibility to directly compare these settings without introducing any (e.g. temporal) side effects. This process of interactive configuration and evaluation is much needed during the design of multimodal user interfaces especially when using uncommon interaction techniques and user interfaces. Squidy therefore facilitates rapid development iterations.

Details on demand

Further beyond the access and manipulation of parameters, Squidy provides illustrated information about the functionality, usage and context of the node directly embedded in the node. By zooming into the information view marked by a white "i" on a blue background (Figure 2a) the information is shown without losing the context of the node. This information view (Figure 3a) may contain code documentation (e.g. automatically generated by javadoc), user-generated content (e.g. from online resources such as wikipedia.org or the Squidy-Wiki) or specifically assembled documentation such as a product specification consisting of textual descriptions, images and videos.

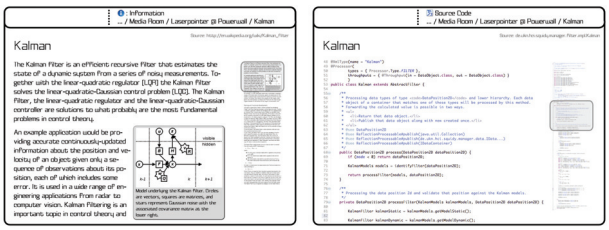

Figure 3: The information view (a) shown on the left illustrates the function of a corresponding node and furthermore provides additional information about its correct usage. On the right, the source code view (b) enables users to access all details and to apply changes to nodes programmatically.

Embedded Code and on-the-fly compilation

The user has the ability to even access the source code (Figure 3b) of the node by semantic zooming. Thus, code changes can be made directly inside the design environment. Assistants such as syntax highlighting or code completion support the user even further. If the user zooms out, the code will be compiled and integrated on the fly, again without the need of a system restart. Users may even add new input and output devices or filters by adding an empty node and augmenting it with applicable code. In order to share the new node with the community the user can publish it into the knowledge base. The design rationale is not to replace the classical development environments such as Microsoft Visual Studio or Eclipse, but rather to integrate some of their functionality directly into Squidy. Thereby, we provide a unified but easy to use design environment which seamlessly integrates the most relevant tools and functionalities for the visual design and interactive development of post-WIMP user interfaces and multimodal interaction.

CONCLUSION

The trade-off between functionality and simplicity is at least as old as the idea of visual programming. In previous systems the agreement concerning this trade-off was made in advance of their usage. The design rational behind

Squidy and its visual user interface concept is the idea to give the decision to the current user. Thus, we address the trade-off by a dynamic, user-controlled approach. This concept also bridges the gap between different users and their knowledge and tasks. Interaction designers may choose and combine input devices and interaction techniques at a high abstraction level whereas developers may zoom in and modify technical details programmatically. Squidy therefore provides a unified design environment bridging the gap between diverse techniques and users. You may find some more information about Squidy in [3] and [4].

REFERENCES

1. Dragicevic, P., Fekete, J-D. Input Device Selection and Interaction Configuration with ICON. In *Proc. IHM-HCI 2001*, Springer Verlag (2001), 543-558.
2. Huot, S., Dumas, C., Dragicevic, P., Fekete, J., and Hégron, G. The MaggLite post-WIMP toolkit: draw it, connect it and run it. In *Proc. UIST '04*, ACM Press (2004), 257-266.
3. Jetter, H-C., König, W. A., Reiterer, H. Understanding and Designing Surface Computing with ZOIL and Squidy. *CHI'09 Workshop - Multitouch and Surface Computing*, Boston (MA), USA, 2009.
4. König, W. A., Rädle, R., Reiterer, H. Squidy: A Zoomable Design Environment for Natural User Interfaces. In *Proc. CHI '09 Ext. Abstracts*, ACM Press (2009), 4561-4566.
5. Max/MSP/Jitter, Cycling '74. http://www.cycling74.com/
6. Perlin, K. and Fox, D. Pad: an alternative approach to the computer interface. In *Proc. SIGGRAPH '93*, ACM Press (1993), 57-64.
7. Serrano, M., Nigay, L., Lawson, J. L., Ramsay, A., Murray-Smith, R., and Denef, S. The OpenInterface Framework: A tool for multimodal interaction. In *Proc. CHI '08 Ext. Abstracts*, ACM Press (2008), 3501-3506.
8. Van Dam, A. Post-WIMP user interfaces. *Commun. ACM 40*, 2 (1997), 63-67.
9. vvvv: a multipurpose toolkit, vvvv group. http://vvvv.org/tiki-index.php
10. Wallace, V. L. The semantics of graphic input devices. In *Proc. SIGGRAPH'76*, ACM Press (1976), 61–65.