Pen and Paper Techniques for Physical Customisation of Tabletop Interfaces

Florian Block[1], Carl Gutwin[2], Michael Haller[3], Hans Gellersen[1] and Mark Billinghurst[4]
[1]*Lancaster University,* [2]*University of Saskatchewan,*
[3]*Upper Austria University of Applied Sciences,* [4]*HitLab NZ*
{block, hwg}@comp.lancs.ac.uk, gutwin@cs.usask.ca,
haller@fh-hagenberg.at, mark.billinghurst@hitlabnz.org

Abstract

An advantage of physical interfaces over graphical widgets is that they bring controls closer to hand. VoodooSketch is a system that supports dynamic customisation of tabletop interfaces with physical controls that users can arrange on palettes. The system employs pen and paper techniques to achieve two novel capabilities: first, users are able to sketch controls that are immediately operational for pen interaction; second, users can label the controls with a handwritten name that identifies their function and binds the control to an application. This paper presents the results of an empirical evaluation of the VoodooSketch interface customisation techniques. The main findings of the study are: that users are able to easily create sketched controls; that they can use them as effectively as traditional input devices; that handwritten labelling is more efficient for control mapping than conventional screen-based methods; and that the sketched controls improve user performance and reduce error rates.

1. Introduction

Physical controls have an advantage over graphical interface widgets in that they do not compete for screen space, cannot become hidden under other windows, and are more directly accessible to the user. Previous work has demonstrated the feasibility of customising graphical interfaces with physical controls that users can bind at run time to conventional applications [4]. Lightweight physical customisation allows users to create shortcuts to frequently-used controls, bring tools and functions closer to hand for a given task, and facilitate more efficient and precise input [17].

Customisation with physical controls can be particularly useful for tabletop interaction. Tabletop interfaces involve larger graphical surfaces so control widgets conventionally arranged in toolbars can be difficult to reach, especially in the context of multi-user interaction. A common strategy is to replicate graphical toolbars that users can position and re-orient for round-the-table interaction [14]. VoodooSketch presents an alternative strategy through the idea of physical customisation. As previously reported, the VoodooSketch system allows users to dynamically create and arrange controls on physical palettes that can be conveniently positioned and moved over the larger graphical surface [2]. Users can adapt physical palettes of arbitrary shape and size, 'plug in' physical control devices, draw controls for pen-based interaction, and label adapted controls with a name that identifies their function and creates a corresponding application binding (illustrated in Figure 1). The result is an interface that it is very flexible – both in terms of *ad hoc* tailoring of functions for direct physical access, and also in terms of mobility and dynamic arrangement of physical palettes over the tabletop.

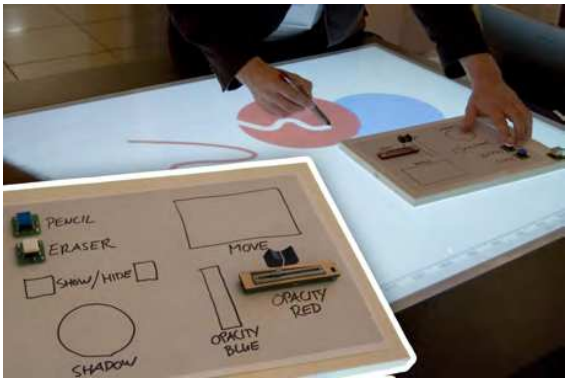

Figure 1. A tabletop interface augmented with physical palettes on which controls can be created and bound on the fly. The close-up shows an ad hoc assembly of controls for a Photoshop task

978-1-4244-2898-4/08/$25.00 © 2008 IEEE

This paper investigates in more depth the digital pen and paper customization techniques that are part of VoodooSketch. We report on a set of user studies that analyse three issues: (1) how intuitive and easy to learn are sketching and handwritten labelling of controls for interface adaptation; (2) how efficient labelling is for control mapping in comparison with on-screen methods; and (3) how well users perform with sketched controls. The studies provide empirical evidence that the techniques are usable and effective. They show in particular that pen and paper controls, even in a rough sketched form, can be used as effectively as traditional control devices, and that handwritten labels are not only a fast method for mapping but also help reduce the error rate with adapted controls.

2. Related Work

The use of physical devices for interaction on tabletops has been demonstrated in a wide range of work, from tangible user interfaces to augmented reality surfaces [8, 3]. This research highlights the advantages of physical controls for tangible interaction with graphical simulations [16], for multi-user interaction around the table [3], and for precise user input on interactive surfaces [12]. The mapping of physical controls, however, is usually application-specific. The principal idea of VoodooSketch, in contrast, is to support end-user adaptation of generic physical controls, as a shortcut to tools and functions in tabletop applications.

Greenberg and Boyle were the first to discuss end-user customisation of conventional GUI applications with physical interfaces [4]. They introduced a 'widget picker' method for direct selection of a widget in the GUI, triggering a dialogue through which functions associated with the widget could be selected from a list and mapped to a physical device. VoodooSketch likewise supports spontaneous binding of physical controls that are immediately usable (without 're-booting' the application), but provides handwritten labelling as a different end-user technique for control mapping. This is a conceptual difference, as it focuses adaptation on definition of the function of a newly added control, rather than on selection of a proxy for a graphical control. This is also a practical difference, as the customisation occurs off-screen by direct annotation of a control rather than on-screen by selection from a menu. In this paper, we specifically examine whether direct annotation with a handwritten label is more efficient than on-screen control mapping.

Customisable physical interfaces have also been investigated in systems that focus on rapid prototyping across the physical and digital aspects of digital products, including d.tools [7] and VoodooIO [15,17]. These systems provide tools for dynamic assembly of the physical controls of an interactive product, and for their mapping to simulations of target functionality. However, control mapping in this context is for design rather than customisation, and is supported by code editors rather than by techniques for end-user adaptation at run time.

Sketching of interfaces has also been widely investigated, although the work has focused on enabling designers to produce rough sketches of an interface that at a later stage can be translated into an operational interface [9]. Sketching is generally considered to be the preferred preliminary capture process for designers because it provides a quick and easy way to externalise design ideas [13]. This has inspired much follow-on research into interactive sketching as a design tool, including the combination of sketching with physical prototyping for interactive product design [11]. In contrast to interface sketching for design, VoodooSketch introduced sketching as an end-user technique for on-the-fly creation of controls that are immediately usable for interaction. In this paper we investigate the ease of use of the technique as well as the usability of sketched controls for interaction.

Digital pen and paper techniques have also been studied more widely, from early visionary work on fluid transition between paper and digital media [18] to infrastructures for documents that can be manipulated in digital as well as in paper form [5]. Commercially available technologies such as Anoto now support capture and wireless transmission of handwriting in real time [1], and the synchronous interaction between pen, paper and computer can be used to create paper interfaces that control an application. This has spurred development of pen and paper interfaces, for instance for a digital scrapbook [19], and for annotation of digital documents using printouts as proxies [10]. Our work differs from this research in that we do not focus on leveraging printed material, but on facilitating customisable off-screen control of applications.

3. The VoodooSketch System

The VoodooSketch system was originally introduced in earlier work [2]. As context for the empirical studies we report below, we summarise the main concepts and components of the system.

Interface palettes and pen interaction. VoodooSketch augments an interactive tabletop with one or more interface palettes which are physically

Figure 2. A hybrid pen writes with real ink on the palette surface and acts as digital input on the augmented tabletop (which is covered with an ink-repellent layer)

separate from the underlying tabletop surface. Initially, the palettes are empty. A special pen can be used to draw on the palette surface with real ink and to digitally interact with the graphical tabletop surface (see Fig. 2, the augmented tabletop is based on [6]).

Interface Composition, Configuration and Usage. A user can instantiate physical controls and/or sketched controls on the palettes. As shown in Figure 3 (1a), physical controls are added by inserting plug-in devices from a toolkit (e.g., a slider) that become networked via conductive sheets embedded in the palette, while sketched controls are created by drawing an arbitrary closed shape (e.g., a rectangle, see Fig. 3, 1b). To give meaning to a new control, the user labels it by writing the name of a function. This process is identical for sketched and physical controls (see Fig. 3, 2a & 2b). The system detects the handwriting and finds the appropriate function in the function pool of the application that is active on the tabletop. Once a control is labelled, its input is linked to that function.

Physical controls are directly manipulated by hand (see Fig. 3, 3a) and produce a defined value (e.g., scalar output for sliders), which is forwarded to the associated application function (e.g., opacity of an object). The sketched controls are operated with the pen, and show different behaviours depending on which function they are associated with. If the function is binary (e.g. "Save"), tapping the control with the pen invokes the action; if the function requires scalar input, the longer side of the shape is taken as an axis and the pen can be moved along this axis in a slider-like fashion. Pen interaction leaves permanent traces in the shape but this does not disrupt the operation of the underlying tracking technology (see Fig. 3, 3b).

4. Empirical Evaluations

We have conducted four user studies to evaluate the pen and paper techniques for interface customisation.

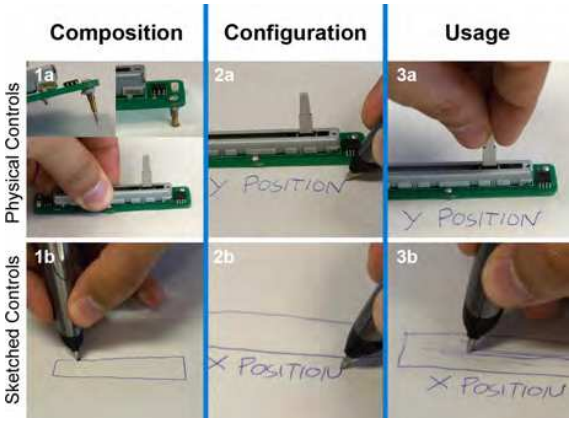

Figure 3. Composition, configuration and usage of controls on an interface palette

The first study tested the accuracy of the sketch-and-handwriting recognition engine; the second tested the time needed to associate controls with application functions, in comparison with list selection; the third tested the benefit of having labels beside the adapted controls; and the fourth assessed the performance of sketched controls in actual tasks, in comparison with physical controls and keyboard.

In all four studies, participants were recruited from a local university, and all had extensive experience with mouse-and-windows software. The studies used either Photoshop CS2 or custom-built C# applications, and were run on a Toshiba Qosmio notebook with a dual-core 1.83 GHz Centrino processor, 2GB RAM, and an nVidia 7600 display card. Two of the studies (second and fourth) used a 1024×768 top-projected table display. The table was 120cm×85cm, and was covered with Anoto paper. All interaction was carried out through an Anoto pen as a digital input device. The other studies used the standard laptop screen, at 1440×900 resolution.

4.1. Expt. 1: Recognition Accuracy

The goal of the first study was to determine how accurate the recognition engine was for component sketches and associated handwritten labels. Twelve participants who were new to the recognition system (five male, seven female) were each asked to draw three sets of ten component/label pairs, and the error rate was recorded. The users could draw arbitrary closed shapes but the label was different for each set and was prompted to the user via a monitor. An error occurred when the system did not detect a closed shape or when the system did not detect the prompted label (so one trial could produce two errors). In the first set

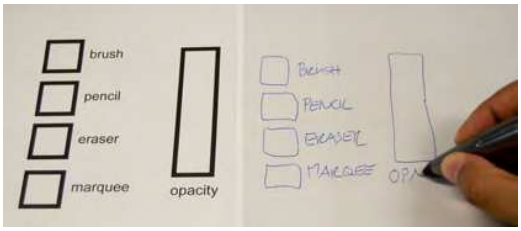

Figure 4. A user creates an interface after a template (same for physical controls).

of trials, the mean error rate in recognition was 0.18 (approximately one trial in six); this improved by the third and final set to 0.09 (less than one in ten). Although this level of accuracy is still lower than required for real-world use, the rapid improvement shows that people can quickly learn to use the recognizer. From our own use of the system, it is clear that this trend continues - our informal tests suggest that people can achieve nearly perfect recognition accuracy (~ 0.01) within about 5 hours of training (the power trendline over the recorded data falls below an error rate of 0.01 at about 1500 trials ~ 3000 minutes). Based on these results, and because we did not have time to train the system for each of the participants in the next studies, the experimenter controlled the recognition engine and ensured that the user's input was correctly interpreted. This allowed us to compare techniques without requiring extensive training; however it is important to note that participants worked normally with the system and were not aware of this experimenter control.

4.2. Expt. 2: Creating and Mapping Controls

The goal of the second study was to compare the time required for different methods for creating new controls and mapping them to application functions. We tested three types of controls (paper sketches, VoodooIO physical controls, and keys on a standard keyboard), and two association mechanisms (handwritten annotation with labels and on-screen selection from a list). List selection was chosen as a reference mechanism, as it is common for control mapping in commercial software (e.g., Photoshop) and existing customisable physical interfaces [6, 25].

The study involved twelve participants (5 male, 7 female), and was carried out using Photoshop CS2 on the top-projected table (see description above).

4.2.1. Design. The study used a 3x2x2 factorial design with three factors:

- Interface (Sketch, VoodooIO, or Keyboard)
- Association Technique (Annotation or List)

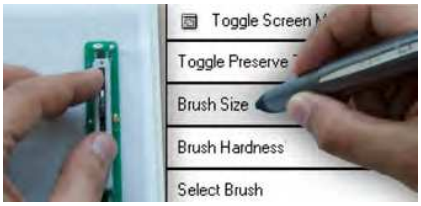

Figure 5. Alternative binding: a control is configured by selecting a function from an on-screen pop-up (same for sketched control)

- Trial (first or second trial with same component)

Since the combination of Annotations and Keyboard could not be tested, we set up the study as two separate designs: one that focused on the effects of Association technique, and one that tested the effects of Interface. The dependent measure in all cases was task completion time.

4.2.2. Procedure. Participants were introduced to the interface and the concept of adaptable interfaces, and then carried out a five-minute training session, in which they practiced associating basic commands in Photoshop. They then carried out test trials in each of the five conditions; their task was to construct a small interface from a template (Fig. 4). In two conditions, sketched and VoodooIO controls were configured with the demonstrated labelling mechanism (Fig. 3) and in another two by selecting from an on-screen list (Fig 5). The list pops up automatically once a new control is detected. The fifth condition was to associate keyboard shortcuts with Photoshop's standard key-binding dialogue. Before each different interface, participants were given a demonstration of that condition. Participants carried out each task twice in succession. The order of conditions was balanced using a Latin square design.

The target interfaces each contained four buttons and one linear control (slider or dial). Since there is no linear control on the keyboard, this was replaced with two buttons for increasing and decreasing the value of the appropriate function. The functions that participants had to associate were the same for the two trials within each condition, but were changed between conditions. In all of the List association conditions, participants used the standard Photoshop list of commands for the standard palettes (e.g., brush, pencil, eraser); this list contains 65 items. Items were chosen to require an equal amount of scrolling and were organized to match the order of the on-screen list, so that participants only had to scroll down to find the next command.

4.2.3. Results. *Effects of Association Technique*. This analysis included only the Sketch and VoodooIO interfaces, as these were the only two that allowed both Annotation and List association. Across all trials in these conditions, tasks were completed in approximately one minute (mean 54 seconds, s.d. 16 seconds). An ANOVA showed a significant main effect of Association Technique ($F_{1,11} = 19.9$, $p < 0.01$), with Annotation (mean 47.7s, s.d. 10.4s) faster than List (mean 61.8s, s.d. 17.5s) by more than 20%. There was also a significant interaction between Association Technique and Trial ($F_{1,11} = 10.4$, $p < 0.01$). From trial one to trial two, performance with List association improved more dramatically (71.0s to 52.6s) than it did with Annotation (51.2s to 44.2s). This was expected, as subjects learned the position of the list items and could find them more easily in the second trial; in contrast, writing the labels was nearly as fast in the first trial as in the second.

Effects of Interface. This analysis used the List association data from each of the three Interface types. There was no significant main effect of Interface ($F_{2,22} = 1.3$, $p = 0.31$) (Sketch controls mean 59.1s, s.d. 16.3s; VooodooIO 64.5s, s.d. 18.5s; Keyboard 64.6s, s.d. 19.1s). This result indicates that the main difference between the conditions is caused by Association Technique and not by Interface.

Subjective Ratings. Participants rated each condition on a scale of 1 (worst) to 20 (best) at the end of each set of trials. Participants clearly preferred the Sketch+Annotation condition (mean rating 16.7, s.d. 2.8), over both Sketch+List (13.8, s.d. 4.6) and Keyboard+List (11.1, s.d. 4.42).

4.3. Expt. 3: Benefits of Labelling

In our system, handwritten annotations are used to associate controls to application functions, but they also serve as a label for the control. The goal of the third study was to determine whether visible labels help users remember the functions of adapted controls.

4.3.1 Design and Procedure. The study used a one-way factorial design with the single factor Labelling (Labels or None). Participants were asked to select particular named controls from a mocked-up interface, containing five controls; the controls were either labelled with their names (using handwritten labels similar to those used in Study 2) or not, depending on the condition.

The 12 participants of his experiment were the same as in Study 2. In both conditions (Labels and None), participants were given ten seconds to familiarise themselves with the layout of the controls and labels (if present). Component names were different in each condition, and were chosen randomly from the standard Photoshop palettes. The order of the conditions was equalized across the participant group.

Participants carried out 60 trials in each condition. In each trial, participants were given the name of an application function, and then had to click on the corresponding interface control. The system gave audio feedback on correct and incorrect actions, but did not continue to the next trial until the participant had selected the correct control. The two dependent measures were completion time and number of errors.

4.3.2 Results. We tested the effects of labelling on completion time, error rate, and subjective preference. As expected, having labels led to significant improvements in performance and preference.

Completion time. A one-way ANOVA showed a significant main effect of Labelling on completion time ($F_{1,11} = 12.93$, $p < 0.01$), with Labels (1.2 seconds, s.d. 0.18 seconds) faster than None (1.5s, s.d. 0.34s).

Error rate. There was also a significant main effect of Labelling on error rate ($F_{1,11} = 8.8$, $p < 0.05$), with Labels (mean error rate 0.0028s, s.d. 0.0065s) dramatically lower than None (mean 0.12s, s.d. 0.14s).

Subjective rating. Participants' qualitative rating of the conditions (on 20-point scales) clearly showed that they preferred the Labels condition (mean rating 15.0, s.d. 3.5) over None (mean 7.7, s.d. 4.8).

4.4. Expt. 4: Using Adapted Controls

The goal of the fourth study was to compare the different interface conditions in use – that is, how do the interfaces compare for actual tasks, once the controls are created and associated with application functions.

We tested the same three interfaces that were used in the second study: a paper-based sketched interface, annotated VoodooIO controls, and a wireless keyboard. The keyboard was included because keyboard shortcuts are the standard means for adapting interfaces in conventional applications. The participants in the study were the same people from Studies two and three, and the experiment was conducted in the same setting and with the same setup as Study two (Photoshop displayed on a top-projected table).

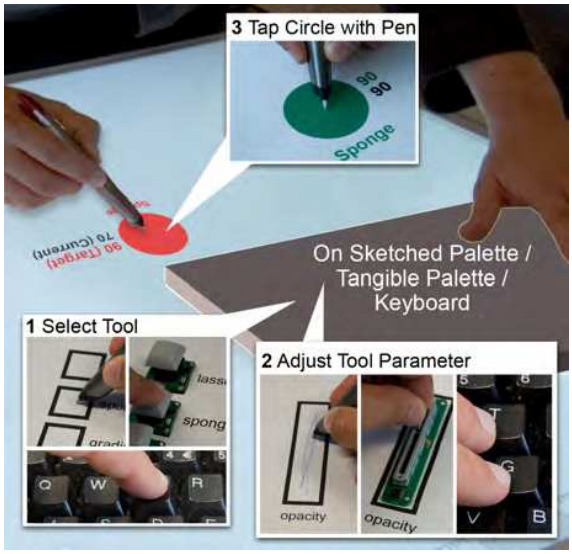

Figure 6. Study 4 trial: (1) the user selects a tool, (2) adjusts a tool parameter and (3) clicks on a target on the screen. This was done for sketched and tangible controls and a standard keyboard. On-Screen targets randomly pop up on different screen locations

4.4.1. Design and Procedure. The study used a one-way factorial design, with Interface as the single factor (Sketched Controls, VoodooIO, and Keyboard, as shown in Figure 6). Dependent measures were item completion time and error rate.

The participant's task was to carry out a set of interface actions using an interface similar to what was built in Study Two (again the context was a drawing application such as Photoshop). Each task involved selecting a particular tool, adjusting the linear control to match a particular value shown on the display, and clicking on a screen region with the Anoto pen. An example of the task is shown in Figure 6. Participants carried out 60 trials in each condition. The on-screen targets for each trial moved to a new random location in the workspace. The order of the conditions was balanced using a Latin square. The task was designed to reflect some aspects of real-world use, where users quickly switch between tools or between colours, and also adjust a parameter such as brush size, while interacting with different objects in the workspace.

The Sketch and VoodooIO interfaces worked as described above. The Keyboard interface mapped six keys (Q, W, E, R, T, and G) to each of the different functions (T and G adjusted the linear control up and down). The keys on the keyboard were not labelled with their function names; however, participants were told which keys were active (and were told that the mapping corresponded with what they had built in Study 1).

4.4.2. Results. We tested the effects of Interface on completion time, error rate, and subjective rating.

Completion time. A one-way ANOVA showed a significant main effect of Interface on completion time ($F_{1,22} = 4.1$, $p = 0.03$). However, none of the post-hoc pairwise tests showed significant differences between conditions (using a Bonferroni correction). The trend was that VoodooIO was slowest (mean 6.9 seconds, s.d. 1.0 seconds), then Keyboard (6.0s, s.d. 1.7s), and Sketched Controls fastest (5.9s, s.d. 0.88s).

One reason for the slower performance of VoodooIO might be its higher latency (200ms) and minor hardware issues (e.g., sometimes buttons are pressed without firing the event). These problems are typical for this type of tangible user interface. However, it was interesting that the Sketched interface performed similarly to the Keyboard. The keyboard has an advantage in that it can easily be used in a bi-manual fashion (dominant hand holding the pen, non-dominant hand using the keys), but also has the disadvantage of being hard to move to the local region of the activity.

Error rate. There was also a significant main effect of Interface on error rate ($F_{2,22} = 16.0$, $p < 0.01$). The rates for both the Sketch interface (mean error rate 0.029, s.d. 0.10) and the VoodooIO (mean 0.022, s.d. 0.026) were far lower than for the Keyboard (mean 0.69, s. d. 0.56). These differences are significant ($p<0.01$). These error rates imply approximately one error in 40 selections for Sketched and VoodooIO, but more than one error in every two selections for the Keyboard.

Subjective rating. Participants clearly preferred the Sketched interface (mean rating 17.7, s.d. 1.7) over VoodooIO (mean 12.7, s.d. 4.4), and both of these over the Keyboard interface (mean 9.4, s.d. 4.3). Participants commented on the hardware problems with the VoodooIO controls, but nevertheless preferred this interface over the standard keyboard solution.

4.5 Summary

We can summarize the key results from these experiments as follows:

- *Expt. 1:* Users are easily able to sketch interface controls that were recognized by our system.

- *Expt. 2:* Handwritten annotations perform better than list input methods, and users prefer a sketch and annotation interface over VoodooIO or keyboard input.
- *Expt. 3:* Visible labels enable users to complete tasks faster with fewer errors than unlabelled controls.
- *Expt. 4:* In a realistic task user preferred a sketch interface compared to VoodooIO or keyboard input and were able to complete the task as fast as with a keyboard and with less error.

5. Discussion

The study results show that users are able to quickly understand the pen and paper techniques we have introduced, and to use them effectively for creation and mapping of controls. Simple closed shapes have proven to be easily understandable by both the user and recognition system, while still being powerful and flexible. This allows users to draw symbolic control areas (e.g. an up-arrow for the "up" function) or to balance the size of controls according to the need for frequency of access (larger controls are easier to target) or resolution (larger sliders give higher resolution). The handwriting recognition also performed very reliably, which can be attributed to the fact that the recognizer can limit its vocabulary to the set of application functions. This is a specific advantage of handwriting recognition in the context of interface configuration.

Labelling proves to be more efficient than list selection for control mapping, independent of whether the mapped control is a physical or paper-based device. Users also prefer labelling over list selection for either type of control, with a more pronounced preference when the control itself is paper-based. These results are based on an experimental design in which subjects were asked to create their interfaces from a given template. However they could have been asked to sketch any arbitrary interface that was meaningful to them. Further studies will need to be made to determine if this could have produced different results. Custom sketch interfaces may require more trial and error work before an optimal design is arrived at.

One important assumption is the fact that users are able to remember the appropriate function names. The current algorithm for parsing the handwritten labels relies on the user writing down the exact function name. In case the user cannot express a function name (e.g. because she fails to remember the exact name) several possibilities exist. The most obvious solution is to offer a fall-back, such as an on-screen list from which the user can select the function and learn its name (the list could be invoked, for example, if the user labels a control with '?'). Another method is to extend the parsing of the function names in a more sophisticated way. For instance, words written close to a control could be treated like a search query. Functions could then be found without an exact match (e.g. "transparency" would give a hit for "opacity"), corrected for mistakes or slightly wrong expressions and behaviour could even be refined by adding more words to the label (e.g. if "opacity" alone does not change the opacity of the layer with the name "background", adding "background" next to opacity would set the function to control the "background" layer opacity).

Two more conceptual advantages of labelling can be mentioned. First, assuming that users have learned the function names, labelling is likely to be faster where there are large numbers of functions, since retrieving a name from memory is faster than finding it in a list. Second, labelling does not rely on the user being at a specific location around the table; in contrast, the on-screen list pops up at a fixed position, requiring a fixed user location and orientation to interact with the list. Labelling on localized palettes works inherently for multiple users and allows mobile creation and configuration from arbitrary positions and perspectives around the table.

Labelling was further shown to have the benefit of improving the usability of adapted controls: increasing efficiency of their use as well as reducing the number of errors. Users also preferred labelled controls over unlabelled ones: this would be expected for neatly printed labels but, importantly, shows user acceptance of handwritten labels in the interface.

The final study investigated the usability of adapted controls in realistic tasks, and the results show that users are able to use them as effectively as traditional input devices. In particular, users perform as well with sketched controls as with physical controls, in terms of both efficiency and error rate. Notably, we observed a subjective preference for sketched controls.

Originally, we designed the system to support physical controls because of their conceptual benefits (e.g., tactile feedback and two-handed interaction). Our studies show that pen and paper alone could allow for sufficient interface customization and would also deliver good performance with few errors. This is a particularly interesting result since paper-only palettes are cheap and easy to add to existing tabletop systems. Paper-palettes can be operated completely wirelessly and easily shared by participants. In comparison to interacting with graphical controls, paper controls also have the advantage of providing very high input resolution, and can be easily moved around the work-

surface (or off entirely) to eliminate occlusion with other on-screen elements.

The main benefit of labelling and sketching of controls is the natural, spontaneous, and fast manner in which they allow users to add off-screen control to applications. Since the footprint of pen-and-paper labels and sketches is larger than that of on-screen controls, these methods are most suited for customisation with a moderate number of controls. The speed at which controls can be created, mapped, spatially arranged, and removed makes the approach particularly useful for throwaway adaptations: the dynamic assembly of controls to be close to hand for a limited duration, in support of changing tasks and interaction focus.

6. Conclusion

This paper's contribution is the evaluation of two digital pen and paper methods that support on-the-fly customisation of interfaces. Our experiences in applying these methods in an augmented tabletop and the results of our empirical studies show that VoodooSketch provides both practical usability and concrete performance advantages, including efficient control mapping and effective use of dynamically-mapped controls. A more general conclusion is that digital pen and paper lend themselves well to lightweight and dynamic customisation of interfaces. This is important since digital pen and paper are low cost, mobile, and intuitively usable.

There are two broader considerations we take away from this work. The first concerns the use of handwritten labels as a simple but effective way of configuring a control. We have introduced this concept for mapping against a predefined set of functions within a given application setting; but in a more general approach, labels could also be used for finding and binding functions more generally, such as in pervasive networks. A second general principle that this research has brought out is the fluidity of interface customisation that can achieved (with plug-and-play controls as well as pen and paper): this can give rise to new interface styles for tabletop systems in which customisation and use are naturally intertwined.

10. References

[1] Anoto. http://www.anoto.com/.

[2] F. Block, M. Haller, H. Gellersen, C. Gutwin, and M. Billinghurst. VoodooSketch – Extending Interactive Surfaces with Adaptable Interface Palettes. In *Proc. Conf. on Tangible Embedded Interaction (TEI 2008)*, pp 55-58.

[3] M. Fjeld, F. Voorhorst, M. Bichsel, K. Lauche, M. Rauterberg, H. Krueger: Exploring Brick-Based Navigation and Composition in an Augmented Reality. In *Proc. HUC '99*, pp. 102-116

[4] S. Greenberg and M. Boyle. Customizable physical interfaces for interacting with conventional applications. In *Proc. UIST '02*, pp. 31–40.

[5] F. Guimbretière. Paper augmented digital documents. In *Proc. UIST '03*, pp. 51–60.

[6] M. Haller, P. Brandl, D. Leithinger, J. Leitner, S. T., and M. Billinghurst. Shared design space: Sketching ideas using digital pens and a large augmented tabletop setup. In *Proc. ICAT 2006*, pp. 948–959. LNCS 4282, Springer Verlag.

[7] B. Hartmann, S. R. Klemmer, M. Bernstein, L. Abdulla, B. Burr, A. Robinson-Mosher, and J. Gee. Reflective physical prototyping through integrated design, test, and analysis. In *Proc. UIST '06*, pp. 299–308.

[8] H. Ishii and B. Ullmer. Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proc. CHI '97*, pp. 234-241.

[9] J. A. Landay and B. A. Myers. Interactive sketching for the early stages of user interface design. *Proc. CHI '95*, pp. 43–50.

[10] C. Liao, F. Guimbretière, and K. Hinckley. Papiercraft: a command system for interactive paper. In *Proc. UIST '05*, pp. 241–244.

[11] T. J. Nam. Sketch-based rapid prototyping platform for hardware-software integrated interactive products. In *CHI '05 extended abstracts*, pp. 1689–1692.

[12] J. Patten, H. Ishii, J. Hines and G. Pangaro. Sensetable: a wireless object tracking platform for tangible user interfaces. In *Proc. CHI'01*, pp. 253-260.

[13] B. Plimmer and M. Apperley. Computer-aided sketching to capture preliminary design. In *AUIC '02: Proc. of the Third Australasian conf. on User interfaces*, pp. 9–12,

[14] C. Shen, F. D. Vernier, C. Forlines and M. Ringel. DiamondSpin: An Extensible Toolkit for Around-the-Table Interaction. In *Proc of CHI '04*, pp. 167-174.

[15] W. Spiessl, N. Villar, H. Gellersen, and A. Schmidt. Voodooflash: authoring across physical and digital form. In *TEI '07: Proc. Int. Conf. on Tangible and Embedded Interaction*, pp. 97–100, 2007, ACM Press.

[16] B. Ullmer and H. Ishii. The metaDESK: models and prototypes for tangible user interfaces. In *Proc. UIST '97*, pp. 223-232.

[17] N. Villar and H. Gellersen. A malleable control structure for softwired user interfaces. In *TEI '07: Proc. Int. Conf. on Tangible and Embedded Interaction*, pp 49–56, ACM Press.

[18] P. Wellner. Interacting with paper on the digitaldesk. *Commun. ACM*, 36(7):87–96, 1993.

[19] D. West, A. Quigley, and J. Kay. MEMENTO: a digital-physical scrapbook for memory sharing. *Personal and Ubiquitous Computing*, 11(4):313–328, 2007.