VoodooFlash: Authoring across Physical and Digital Form

First Author Name (Blank if Blind Review)
Affiliation (Blank if Blind Review)
Address (Blank if Blind Review)
e-mail address (Blank if Blind Review)

Second Author Name (Blank if Blind Review)
Affiliation (Blank if Blind Review)
Address (Blank if Blind Review)
e-mail address (Blank if Blind Review)

ABSTRACT
Design tools that integrate hardware and software components facilitate product design work across aspects of physical form and user interaction, but at the cost of requiring designers to work with other than their accustomed programming tools. In this paper we introduce *VoodooFlash*, a tool designed to build on the widespread use of Flash while facilitating design work across physical and digital components. VoodooFlash extends the existing practice of authoring interactive applications in terms of arranging components on a virtual stage, and provides a physical stage on which controls can be arranged, linked to software components, and appropriated with other physical design materials.

Author Keywords
Product design, Physical interfaces, Authoring tools, Flash

INTRODUCTION
Product design processes are fast and fluid. Ideas are rapidly made tangible using paper and foam to create low-fidelity prototypes that are iteratively refined. But for interactive products, the design of behaviour and user interaction usually remains decoupled from the 'physical design' process, and is developed on desktop screens with 'flat' representations of the product and tools such as Adobe Flash to develop the interaction. Various tools have been emerging to allow designers to better couple physical design and interaction design, including techniques for hooking up 3D product models with software simulations via keyboard emulation [1,5], physical interface toolkits [2], and complete design environments that cover hardware and software aspects [3]. In contrast to these, we present an approach that integrates physical prototyping with Flash, a predominantly used environment in design practice.

Our design tool, *VoodooFlash*, is based on the Flash concept of a stage on which interactive components are arranged in the process of designing an interface. Alongside the graphical stage in Flash, we provide a physical stage on which designers can arrange controls, such as buttons, rotary knobs and sliders. The graphical and physical stage are closely coupled, with physical controls represented by virtual counterparts on the Flash stage, and with Flash programmed output visually overlaid on the physical stage.

Figure 1 shows the two stages side by side for design of a map navigation interface. The physical stage serves as an arena in which designers can work with paper, foam and other materials around the controls, and the graphical stage provides the environment in which controls can be associated with functionality and interactive behaviour. The two stages are kept tightly synchronized, to allow dynamics such as rapid change of the behaviour of a control (affected on the graphical stage) and immediate testing by manipulating the respective control (on the physical stage).

We have implemented our tool by integrating Flash with VoodooIO, a physical interface system that supports rapid and dynamic arrangement of controls on interactive substrate material [6]. The VoodooIO system is used for provision of the physical stage in our tool, and communicates with Flash via a small set of events to signal placement, manipulation or removal of physical controls.

The VoodooFlash tool has been evaluated in design sessions with Flash experts and with a group of industrial designers. After brief discussion of related work, and of the integration of Flash with VoodooIO, we report use experience and feedback from these studies.

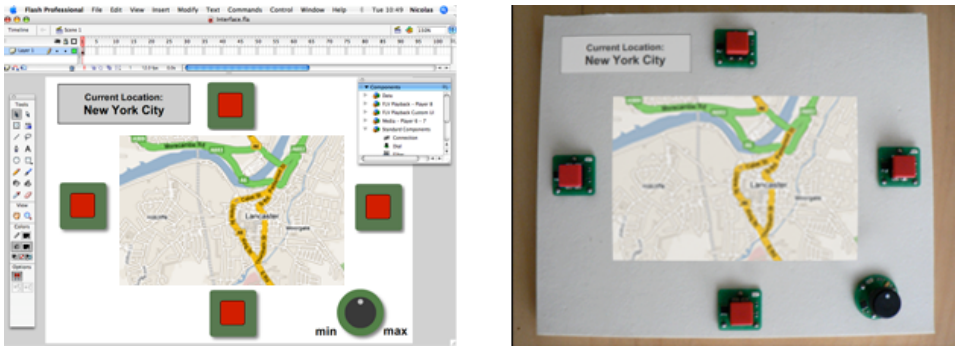

Figure 1. VoodooFlash provides a physical stage for interface prototyping(right) alongside the Adobe Flash's virtual stage (left).

BACKGROUND / RELATED WORK

Designers have responded to the increased embedding of computing in products with techniques for coupling hard and soft representations of a product in the design process. The Buck method is based on tethering a functional product model to a PC for user testing of physical controls in conjunction with interface software [5], and the IE system uses micro-switches embedded in foam-core models and keyboard emulation to facilitate a physical-interactive experience within hours of an initial design sketch [1]. The same principle, in more rudimentary form and focussed on cardboard prototypes, has been adopted in the BOXES system [4]. In common with these approaches, our tool facilitates linkage between physical model and software parts of a design, but on the physical side with a richer set of controls (beyond switches and touch sensors), and on the software side focused on extension of a design environment already in widespread use in design practice.

A variety of toolkits and design environments have emerged for development of physical interactive systems. Phidgets, for instance, provide physical interface building blocks analogous to widgets in graphical user interfaces [2]. The system was initially targeted at making hardware more accessible by GUI programmers but also provides a Flash API. VoodooFlash likewise supports Flash development of the behaviour of physical controls but provides a much tighter integration by giving physical controls an explicit representation as Flash components.

The representation of physical devices within the design environment is a feature our tool shares with d.tools [3]. The d.tools system provides an integrated design environment that supports 'plug and draw' integration of physical devices and statechart-based editing of interactive behaviour. However while the system is more open-ended in terms of hardware that can be integrated it does not provide support for existing authoring environments. Specifically the lack of support for Flash developers has been reported as a distinct shortcoming [3].

The VoodooIO system, on which our Flash extension is based, is similar to Phidgets in providing a range of physical controls but in addition emphasizes malleability of physical interfaces [6]. VoodooIO does this by providing a substrate material on which controls can be dynamically added, arranged, manipulated and removed. This substrate material effectively serves as a network bus to which controls can be connected effortlessly, wirelessly and faster than via a standard USB connection (faster both in terms of user interaction and network discovery).

INTEGRATION OF FLASH AND VOODOOIO

The VoodooIO system is available as a TCP service to which clients can connect to set properties of physical controls and to monitor interaction (i.e. adding, manipulation, removal of a controls on a substrate). For integration of the system in Flash, we have built a connection manager and event dispatcher that, transparently

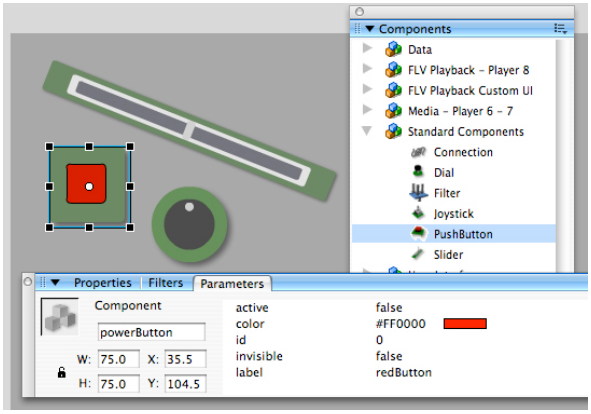

Figure 2: Physical controls are integrated as standard components in the Flash development environment.

for the user, handle communication, parsing and event dispatching between Flash and VoodooIO.

From the user's perspective, VoodooIO is integrated in the form of reusable Flash components. These include:

- a connection component through which a VoodooIO service can be selected (connections to multiple services are possible, facilitating programming of distributed physical interfaces),
- a component each for the available VoodooIO control types (each with predefined event handlers for the three core VoodooIO events: *added, manipulated, removed*),
- a filter component that allows filtering of events (for example to filter events from a particular service if multiple connections are made).

Users interact with these components as they do with any other Flash component, using the standard mechanisms for instantiation of a component, arrangement on or off the stage, and setting of properties and parameters through graphically inspectable panels (cf. Figure 2).

Physical controls and corresponding components can be brought onto their respective stages independently of each other, in no prescribed order. Associations can at any time be made and changed, either by using the unique ID that each VoodooIO control has built-in, or by using a name that may be pre-programmed for a control, or interactively assigned.

USE EXPERIENCE

We conducted two studies for evaluation of VoodooFlash with external users, one in Munich with two Flash experts, and one in Delft with a larger group, primarily from an academic industrial design background.

Munich experience

The study in Munich was organised as a half-day expert evaluation to which we invited two professional Flash developers from a local start-up. Both participants had a background in computing and several years of professional work experience in Flash application development.

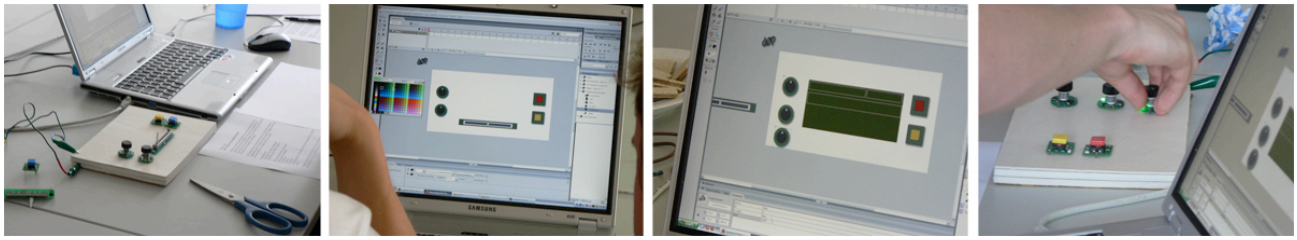

Figure 3: Participants in our study at Munich initially moved from paper design and physical interface layout to work within Flash (images on the left), and at later points introduced first in Flash and to then reflect and test them on the physical stage (right).

We started with giving our developer-users a 15-minute introduction to the VoodooFlash environment, using an example design case. They were then given the task to design a user interface for an Internet radio, with a design brief describing the requested functionality (selection of preset stations grouped by categories, etc.), the available resources (a set of buttons, knobs, sliders, etc.) and the design focus (functional interface design, abstracting from issues such as data formats and storage). Our two users were given two hours to jointly work on their task. This was followed by an interactive session, in which they explained their design and were challenged to carry out a change in the interface on-the-fly (simulating change requests customers might have in a design session). Finally, we invited and collected feedback on the design experience.

Figure 3 shows a series of photos taken during the design session, and indicates, from left to right, how the two developers progressed in general with their task. Initially they sketched a crude design on paper and then laid the interface out physically with VoodooIO controls. They then switched their attention to design on the Flash stage, instantiating, linking and arranging the corresponding virtual controls and associating them with functionality programmed in ActionScript 2.0. At a later stage, changes to the interface configuration were first carried out in the Flash environment, and then reflected on the physical stage.

Throughout the design session, the two developers interacted intensely, in continuous joint reflection over their task (very much exemplifying the reflective prototyping practice that Hartmann et al. discuss in [3]). One of the two developers tended to keep control over mouse and keyboard for work within Flash, and specifically for ActionScript programming. His design partner would simultaneously work with the controls on the physical stage, to generate life input to the script as it evolved, and to continuously try and test the effect of additions and changes in functionality.

The two developers were able to complete their task in the given time, including iterations for refinement (e.g. fine-tuning the response to knob rotation to the specific value range generated by the device). They did not require help other than support we had integrated in the tool environment (documentation of the VoodooIO API), and they were able to completely abstract from VoodooIO technical detail (e.g., they did not have to understand how controls are detected and networked, and at some point during the session one of our users suggested to his partner to *"stick the two [knobs] further apart"*, suspecting they might interfere with each other when to close together). After completion of the task, the two developers were challenged to replace a slider they had selected for volume control with a rotary knob. This only required them to physically replace the devices, to instantiate, name and bind a virtual knob, and finally in ActionScript to replace the name they had used for the slider with the knob's name, all of which was achieved in less than a minute.

In the final feedback session our developer-users reported that working with VoodooFlash was *"identical in terms of programming"* to routine Flash development, *"all you had to know in addition was the VoodooIO events but there are only four anyway"*. They also speculated that development with separate physical input devices was *"probably faster because you don't have to go through menus [to trigger actions]"*, and also because mouse and keyboard focus always remained on the programming task, while the VoodooIO extension served for testing. The users also noted the fun factor of the system, and of being able to immediately see the effect of what you do.

Finally we prompted our developer-users to suggest improvements to our tool. Among others this resulted in consideration of how displays could be integrated with the physical stage instead of projection. This led to the idea of cut-outs in the physical stage to insert a display or to look through to a display underneath. As the VoodooIO material can be cut too any shape without comprising its functionality, it was easy to try this out (cf. Figure 4).

Figure 4. A cut-out in the physical stage for a display area

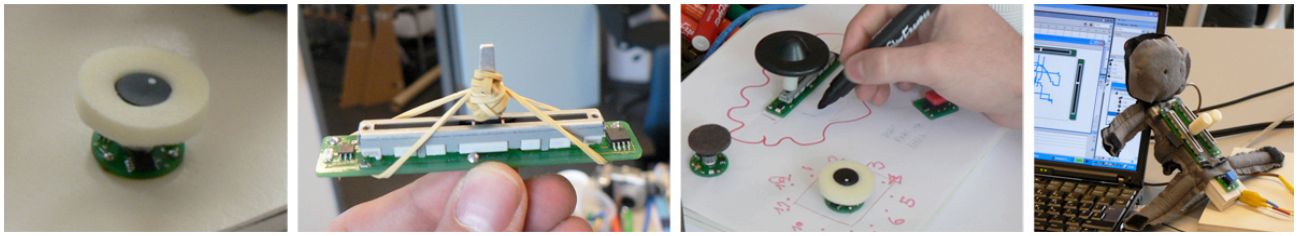

Figure 5: Participants in the the Delft workshop explored appropriation of the physical components provided by VoodooFlash.

Delft experience

The study in Delft was organized as a workshop with an interdisciplinary project team of about 15, primarily composed of academic staff and students from industrial design departments, with some but not expert knowledge of Flash. The team was given a 30 minute introductory presentation of our system, and then split into a 'red' and a 'blue' group, both given the task to develop a version of the classic Etch-a-Sketch toy for which they were given 2 knobs, 2 sliders and 2 buttons as resource. The groups were to first develop their own version of Etch-a-Sketch in order to sketch a trace on a projected display with separate controls for X- and Y-axis of the cursor, and invited to then more freely experiment with the system, and to try and interfere with each others design (facilitated by exposing VoodooFlash events over a shared network). The workshop was concluded with a general feedback session.

In contrast to our experience in Munich, the groups struggled more with their initial task, as the participants were not as proficient in using ActionScript. The 'blue' group though quickly got into a more explorative mode, mapping controls in intricate ways to functions such as changing line thickness for etching, so to confuse the 'red' group as to how their Etch-a-Sketch version worked. As programming in ActionScript was taken over by 1-2 individuals in each group, others began to explore how the small set of controls they were given could be physically appropriated. Figure 5 shows some examples resulting from this, from left to right:

- A rotary knob 'dressed up' to modify look and feel.
- A slider customized with rubber-band to be self-centering and usable for rate control input (as opposed to absolute control).
- Pen and paper used on the physical stage to label and decorate the physical interface.
- A 'voodoo doll' constructed around a strip of VoodooIO substrate, two sliders and a button, for remote controlling (and hi-jacking) the visual display of the other group.

In the feedback session, participants welcomed the combination of physical prototyping with programming in Flash, as Flash had been adopted as the first language for design student education in two of the Universities represented in the workshop. Apart from this, feedback was more concerned with the physical sub-system of the VoodooFlash tool. Most workshop participants had experience with using Phidgets in product design classes, and in comparison saw in particular the wire-free assembly of VoodooIO devices as a significant advantage. Their other concern was ease of physical appropriation, and for example how more specialised sensors and transducers could be made to work with VoodooFlash.

CONCLUSION

VoodooFlash achieves a very seamless extension of an existing and widely used authoring environment for work across physical and interaction aspects in product design and prototyping. The two design exercises with external users reported in this paper indicate a very good fit of our tool with existing design practices. On the interaction design side, the tool extends the widely used Flash authoring environment in a manner that is intuitive and effective in hiding technical detail of integrating a physical interface system. And on the physical design side, the physical stage proves to be effective in supporting fast and fluid assembly of controls (by virtue of VoodooIO) and in facilitating appropriation with other physical design material such as paper, foam, textiles and rubber-band.

REFERENCES

1. Gill, S. Developing Information Appliance Design Tools for Designers. *Personal and Ubiquitous Computing*, Vol. 7, Springer-Verlag 2003
2. Greenberg, S. and Fitchett, C.. Phidgets: easy development of physical interfaces through physical widgets. *Proc. UIST '01,* ACM Press (2001), 209–218.
3. Hartmann, B., Klemmer, S., Bernstein, Abdulla, L., Burr, B., Robinson-Mosher, A. and Gee, J. Reflective Physical Prototyping through Integrated Design, Test and Analysis. *Proc. UIST 2006*, ACM Press, pp. 299-308.
4. Hudson, S. and Mankoff, J. Rapid Construction of Functioning Physical Interfaces from Cardboard, Thumbtacks, Tin Foil and Masking Tape. *Proc. UIST 2006*, ACM Press, pp. 289-298.
5. Pering, C. 2002. Interaction design prototyping of communicator devices: towards meeting the hardware-software challenge. *interactions 9, 6* (Nov. 2002), 36-46.
6. Villar, N., Gilleade, K.M., Ramduny-Ellis, D., Gellersen, H. The VoodooIO Gaming Kit: A real-time adaptable gaming controller. *Prof. ACM ACE 2006 (Advances in Computer Entrertainment Technology)*, Hollywood, USA, June 2006, ACM Press.