Ubi Displays: A Toolkit for the Rapid Creation of Interactive Projected Displays

John Hardy[1], Carl Ellis[2], Jason Alexander[2], Nigel Davies[2]
HighWire-DTC[1], School of Computing and Communications[2]
Lancaster University, UK
john@highwire-dtc.com, carl.ellis@comp.lancs.ac.uk, j.alexander@lancaster.ac.uk, nigel@comp.lancs.ac.uk

ABSTRACT

In this demonstration we present *Ubi Displays*: an open source toolkit designed to simplify and expedite the process of creating media-rich interactive projected displays. Our toolkit enables users of web technologies and commodity hardware to quickly create applications that previously would have taken substantial development effort by skilled programmers. By reducing the required skills, development time, and the costs of tools and hardware we hope to encourage people from a diverse set of communities (e.g. designers and hobbyists) to engage with pervasive displays and explore the different kinds of impact and interaction they can support. This demonstration covers the key features of *Ubi Displays* and highlights a selection of real-world case studies that illustrate how the toolkit can be used.

Categories and Subject Descriptors

H.5.2 [**User Interfaces**]: Input Devices and Strategies

General Terms

Design, Human Factors

Keywords

Pervasive displays, interactive, projection, multi-touch, multi-display, ubiquitous computing

1. INTRODUCTION

Over the last two decades, researchers have explored how mixing relevant interactive content with the physical world has the potential to create transformative interfaces [2,4,5,6] However, a key challenge in this space has always been that creating such interfaces is expensive. Implementations are typically bespoke and the lack of standardised development tools dictates that large amounts of skilled labour, time, and money must be invested in their creation [1]. These factors create barriers that prevent many people from engaging with the technology and thus iteratively evaluating and improving it as a community. This is particularly harmful in application driven / exploratory scenarios where a technical contribution is not the focus of the work and the expected results cannot always justify a large investment of technology or development time.

The *Ubi Displays* toolkit aims to change this by offering a toolkit that greatly simplifies and expedites the process of creating rich interactive projected displays.

This demonstration serves as an introduction to the Ubi Displays toolkit (http://code.google.com/p/ubidisplays) and showcases a selection of its features and capabilities. In the following sections we outline the key features of Ubi Displays and provide several real-world case studies that show how the toolkit has been applied or extended.

PerDis 2013, June 04 – 05 2013, Mountain View, CA

2. TOOLKIT FEATURES

Using commodity hardware (a PC combined with a standard projector and depth camera) users with web development skills are able to quickly create a wide variety of display installations that leverage their existing hardware, skillset and expectations of rich content (see Figure 1).

Users do not need to learn any new display programming languages and the basic operation of the toolkit's drag-drop interface can easily be communicated through a five minute tutorial video.

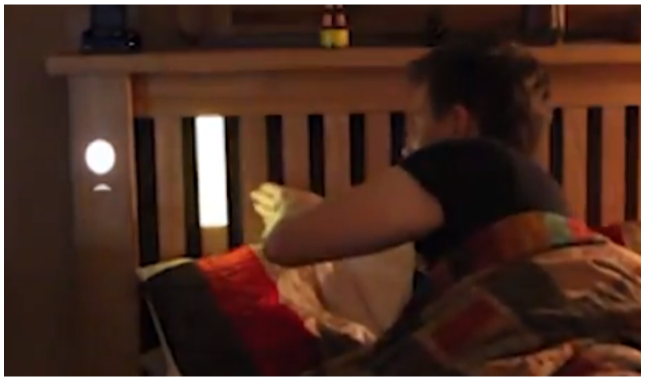
Figure 1: An interactive bed display; each wooden slat in the bed controls a light in the room. Related video: http://youtu.be/df1NO7MoAUY

In brief, the toolkit offers the following features:

Displays are sandboxed Chromium instances. Each display is effectively a web-browser tab and has a similar performance profile.

Web Standards enable rich content. Web developers can use the latest HTML5, CSS3 and JavaScript constructs to quickly build complex interactive displays. Limited support for Adobe Flash and WebGL (software rendering) is also included.

Multiple displays per projector are supported to maximize the utility of the projection hardware. The JavaScript hosted in each display is able to query and invoke logic on surrounding displays through a simple programming interface. A moderation mechanism is included to help prevent malicious code.

Physical responsive design enables display content to configure itself automatically based on the physical characteristics of the display surface it is being presented on (e.g. increasing button sizes on physically larger display surfaces). As with a normal responsive design, the extent of its application remains under the control of the web developer.

Displays are able to move around their environment within the view of the projector and depth camera. However, we also limit projection to areas pre-defined by the person responsible for the deployment so that they can maintain a degree of control over the content and aesthetic of the space.

Displays can spawn, delete and move other displays using JavaScript functions that interact with the underlying toolkit.

Displays are transient and can be made to disappear when not needed. The can then re-appear in response to some internal or external trigger (i.e. a person walked into a certain area).

Displays are agnostic of interaction modality. Each display is able to request and handle multiple forms of interaction without affecting the other displays. For example, a display placed on a floor could react to foot position, whereas one on a wall could react to touch or gesture.

Accurate multi-touch detection when the sensor is positioned less than 1.2m from the interaction surface. This uses a point cloud based multi-touch detection algorithm described and profiled by Hardy et al [1].

Built on an extensible codebase. It is easy for native programmers to add new features either by editing the program code or invoking external processes from the JavaScript and capturing the output. Technical web developers can use browser features such as WebSockets to enable external connectivity.

Simple user interface for deploying displays. The toolkit has a simple visual process for selecting which part of a physical space can be used as a display area. Deploying content is also a ‘drag-drop’ process with fully supported save and load functions which help automate the start-up of unattended installations.

Easily achievable hardware requirements. Readily available commodity hardware helps to promote the use of *Ubi Displays* in cost-sensitive and rapid-prototyping situations.

Online community support helps to promote a growing community of *Ubi Displays* users (over 700 downloads worldwide at the time of writing) which are able to offer community support.

3. CASE STUDIES

The following sub-sections are real-world examples of how *Ubi Displays* has been applied to create a variety of display types. Each has been selected as the toolkit was in some way extended in order to achieve it.

Big Bang Fair Particle Accelerator Simulator

Ubi Displays was used to help create a ‘Particle Accelerator Simulator’ game for a large national science fair. Robustness was a key requirement as the fair had 60,000 visitors over the course of 4 days. A WebGL display formed the centerpiece of a large metal table. WebSockets were used in order to communicate with a USB BV4626 general purpose IO board. This enabled the visualization of the projected display to control external devices via relays and receive input from large physical buttons. Related video: http://youtu.be/IqtZC9sewuk

Object Detection Support

In collaboration with researchers from the University of Stuttgart, *Ubi Displays* has been combined with an implementation of the BRISK feature recognition algorithm [3]. This gave displays the ability to determined which (if any) pre-determined objects are placed in an area visible by a web camera. The integration extended the native *Ubi Displays* code by allowing it to invoke external processes (in this case an object detection binary implemented in C++) and stream data to and from these processes via JavaScript. Although a security risk, this extension could be useful in other contexts.

Interactive Table Alternative

One of the most popular requests we receive through the open source project is the ability to use *Ubi Displays* as a cheap interactive table. Typically, these requests are motivated by one of four reasons: (1) they have a pre-existing native application they want to use, (2) they lack experience programming JavaScript, (3) they want to use a full Windows desktop environment, or (4) they cannot afford the cost of a large damage-resistant multi-touch table display. To cater for these groups, we have extended a pre-release version with two additional modes: *Desktop mode* which forwards multi-touch events from a display and injects them into a full Windows desktop, and *display mode* which enables users to select desktop regions and then re-project this content onto another surface with multi-touch support.

4. ACKNOWLEDGMENTS

Thanks go to Markus Funk at the University of Stuttgart who implemented the object detection algorithm. This demonstration was conducted as part of the post-disciplinary HighWire programme at Lancaster University. HighWire is funded by the Digital Economy Programme: a Research Councils UK cross council initiative led by the EPSRC and contributed to be AHRC, ESRC and MRC. We also acknowledge the financial support of the Future and Emerging Technologies (FET) programme within the 7th Framework programme for Research of the European Commission, under FET-Open grant number: 244011 (PD-NET).

5. REFERENCES

[1] Hardy, J. and Alexander, J. Toolkit Support for Interactive Projected Displays. In *Mobile and Ubiquitious Multimedia (MUM)* (2012), ACM.

[2] Hardy, J., Bull, C., Kotonya, G., and Whittle, J. Digitally annexing desk space for software development: NIER track. In *Proceeding of the 33rd international conference on Software engineering* (2011), ACM, 812--815.

[3] Leutenegger, S., Chli, M., and Siegwart, R. Y. BRISK: Binary robust invariant scalable keypoints. In *Computer Vision (ICCV)* (2011), IEEE, 2548-2555.

[4] Pinhanez, C. The Everywhere Displays Projector: A Device to Create Ubiquitous Graphical Interfaces. In *Proceedings of the 3rd international conference on Ubiquitous Computing (Ubicomp '01)* (2001), ACM, 315-331.

[5] Raskar, R., van Baar, J., Beardsley, P., Willwacher, T., Rao, S., and Forlines, C. iLamps: Geometrically Aware and Self-Configuring Projectors. In *ACM Trans. Graph.* (2003), 809-818.

[6] Underkoffler, J., Ullmer, B., and Ishii, H. Emancipated pixels: real-world graphics in the luminous room. In *SIGGRAPH'99* (1999), 385-392.