CONTROL: SOFTWARE FOR END-USER INTERFACE PROGRAMMING AND INTERACTIVE PERFORMANCE

Charles Roberts

University of California at Santa Barbara
Media Arts and Technology Program
charlie@charlie-roberts.com

ABSTRACT

The author presents Control, a software application for mobile devices enabling users to define custom graphical user interfaces for transmitting both OSC and MIDI. Unlike other similar mobile applications, interfaces are defined using web standards such as HTML, CSS and JavaScript. The widgets that come predefined with Control can be extended by users via JavaScript; scripting can also be used by users to define new widgets from scratch. The ability to add user defined scripting provides flexibility, dynamism and sophistication to interfaces that is absent in other mobile interface applications.

Control is free to download from the Apple App Store and will also be freely available from the Android Market by the time this paper is published. Devices running Control can be automatically discovered on a network; once this occurs servers can "push" interfaces to them that are ready for immediate use. Bi-directional communication with Control also allows servers to update widget values and dynamically restructure interfaces on the fly. These features make it ideal for audience interaction pieces and installation art as participants can download the software for free, be pushed a custom interface and quickly begin interacting with a work. This paper will describe the advantages and disadvantages of Control over other interface applications for mobile devices.

1. INTRODUCTION

1.1. Motivation

As mobile smart devices proliferate, so do applications enabling musicians and artists to use these devices to control external software. These applications draw heavily from the JazzMutant Lemur [1], a dedicated stationary multitouch surface that empowered users to define their own interfaces for sending Open Sound Control (OSC)[5] messages. Unfortunately, the mobile applications that followed the Lemur did not imitate the feature that made it more than just a collection of virtual sliders and buttons: end-user scripting. By giving users the ability to write custom scripts that could be attached to widgets the Lemur enabled dynamic interfaces that would otherwise have been impossible. A primary motivation of Control is the desire to program interfaces on mobile devices that could match the complexity of those present on the Lemur while reaping the benefits of portable, wireless interaction that mobile devices provide.

[1]http://www.jazzmutant.com

Other important motivations include easing the design of audience interaction pieces and enabling the ability to send contextual interfaces to users inside of Virtual Reality Environments (VREs). The author researches interactive control of VREs inside of the Allosphere[1]: a three story, spherical, immersive instrument that can accommodate up to thirty users simultaneously. Providing Allosphere researchers the ability to present users contextual, individualized interfaces was another major design consideration when creating Control.

1.2. Terminology

The *Application Interface* is the interface users see when they first launch the Control application. The application interface is used to set preferences, choose destinations and protocols for sending control information, and to download and launch user interfaces. *User Interfaces* are individual interfaces written in JavaScript Object Notation (JSON) with custom JavaScript for specialized scripting. Some user interfaces are included with Control while others can be downloaded from the internet or are created by users.

Widgets are graphical elements presented in user interfaces that (for the most part) generate data in response to touch events. *Sensors* are hardware components in devices whose signals can be transmitted using OSC and MIDI by Control.

Clients are instances of Control running on mobile devices. *Servers* are software applications running on computers receive data from instances of Control. Bi-directional communication is also possible between clients and servers.

2. RELATED WORK

For the sake of brevity related work is defined here as multitouch control surfaces geared towards use in the arts. Now discontinued, the JazzMutant Lemur basically invented this market in 2005. Although a significant portion of the Lemur focused on providing virtual versions of physical widgets that are standard in musical interfaces (sliders, knobs, buttons etc.) it also offered a number of interesting affordances to users that have yet to be duplicated in mobile device software. The Lemur came with a variety of non-traditional interface elements for users to employ, each of which had its own physics engine attached to it. A single interface (even a single widget) could output to multiple OSC destinations. It was easy for users to add scripts in order to execute simple behaviors like outputting arbitrary OSC messages or constraining the output of one widget based on the current state of another. One of the main design goals of Control is to match and exceed the scriptability of the Lemur and to eventually feature a library of widgets as diverse as the Lemur's. That said, Control reaps the significant advantages of running on wireless mobile devices possessing an array of sensors that the touch dependent, wired, stationary Lemur never possessed. The scripting possibilities of Control also allow for widgets to be rearranged or reconfigured on the fly in response to OSC or MIDI messages and for the creation of custom widgets.

The ReacTable[3] is a control surface for audio that does away with traditional paradigms of interaction associated with audio manipulation. Although originally not technically multitouch, the original ReacTable does track the position and rotation of multiple physical markers placed on its surface. In addition to using position and rotation of the markers it also uses the physical proximity of the markers to each other as a parameter for sound synthesis and processing. Originally a large physical table with a projected display, the ReacTable software now also runs on iOS devices, where multitouch allows users to manipulate virtual representations of the formerly physical markers. In the author's opinion the mobile ReacTable interface is extremely creative and unique; it is the author's hope that after further development users will be able to design interfaces of similar complexity and creativity using Control.

TouchOSC[2] is the most popular OSC interface program in Apple's App Store at the time of this writing and is also available for Android devices. One strong feature of TouchOSC is a companion desktop application that allows users to construct their own interfaces using a GUI instead of by writing markup. Unfortunately, there are many factors preventing the use of TouchOSC in interactive performances and installations. First, the application is not free for iOS devices. In the author's opinion it is unlikely that potential participants in an interactive work will be willing to pay money to download the software immediately before a performance. Even if participants were willing to pay there is no public API for pushing interfaces to mobile devices running TouchOSC; the application was clearly created with a different set of design goals from Control. Finally, there is no scripting capability to add behaviors to widgets. Simple behaviors such as having a button affect the position of a slider are not possible with TouchOSC and users are not able to define their own widgets.

[2]http://hexler.net/software/touchosc

Mrmr[3] is a project for iOS that has some similar design goals to Control. It features a protocol for pushing interfaces to clients and manipulating them using OSC. It also has auto-discovery via Bonjour so that clients running Mrmr on mobile devices can easily participate in interactive performances and installations using their own devices. However, Mrmr eschews the benefits of scripting in favor of using a deliberately simple application-specific protocol for describing interfaces. In contrast, Control is built using web standards such as JavaScript, JSON, CSS and HTML. These technologies provide much greater flexibility to users as well as greater potential for interoperability than application-specific syntaxes. As mentioned previously the end-user scripting available in Control provides capabilities that are critical for the design of complex interfaces.

3. CONTROL

3.1. Overview

Control currently runs on iOS devices and is built using open-source web technologies. It is capable of outputting both OSC and MIDI messages depending on the type of network destination users choose. In addition to data from the touchscreen Control can also read and transmit data from the Accelerometer, Gyroscope and Compass sensors present in devices. The application GUI consists of four simple tabs:

- Interfaces: a list of interfaces that the user can run on their device. There are also buttons to add new interfaces from the internet and to remove interfaces from the device.
- Destinations: a list of OSC / MIDI IP addresses and ports that Control can transmit information to.
- Preferences: miscellaneous preferences such as whether or not to stop the device screen from locking while the application is running
- Info: links to additional resources and credits

Users define custom interfaces in JSON; these interfaces can be downloaded over a standard TCP/IP connection from a web server or pushed to a device via OSC. In addition to using JSON to define interface structure users can also program custom behaviors for widgets using JavaScript.

Figure 1 shows both the main application interface and a user interface running the Game of Life.

3.2. Standardization and Web Technologies

Control is built using a variety of open-source web libraries and frameworks. The PhoneGap project [4] played a particularly significant role in the development of Control. PhoneGap presents user interfaces on mobile devices

[3]http://mrmr.noisepages.com/
[4]http://www.phonegap.com

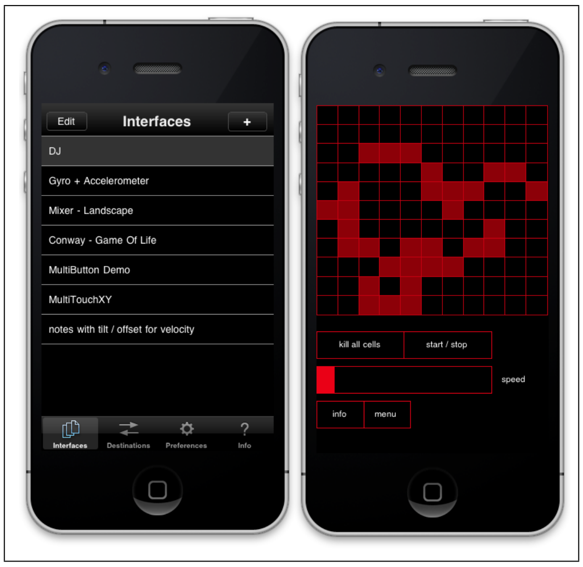

Figure 1. Screenshots of the application interface and a user interface that plays the Game of Life. The Game of Life interface uses button widgets as cells that can be toggled on and off by touch in addition to the rules of the game. The tick speed can be changed by a slider.

in a web browser and is designed to run on a wide variety of mobile operating systems. Unlike standard web application programming where the programmer does not have access to underlying hardware features of devices, PhoneGap provides a useful abstraction for accessing hardware features via JavaScript across all operating systems it supports. By building the GUI of the software using PhoneGap the process of porting the application to other operating systems is eased; the port of the GUI (both the application interface and user interfaces) is almost automatic to operating systems that employ a modern web rendering engine.

The application interface of Control is designed using HTML, CSS and JavaScript. User interfaces are primarily defined in JSON; when these interfaces are read by Control they are translated into HTML + CSS and rendered by the web engine. By basing user interfaces on web standards the barrier of entry to creating them is lowered for potential users with basic web programming experience. Web standards also open the possibility of Control being ported to run inside of web browsers running on personal computers and to other web-enabled devices. Finally, adhering to standards should help ensure that interfaces from Control will still be operable in the future.

3.3. Access to Sensors

One goal of Control is to permit access to as many of the sensors present on mobile devices as possible. Currently this list includes the accelerometer, the compass (magnetometer), and the gyroscope. The gyroscope in particular is an extremely useful sensor that allows users to measure rotational rate. While the accelerometer is successful at measuring sudden gestural movements coupling it with the gyroscope adds a great deal of resolution when measuring orientation. Users can select the rate that these sensors output at; this can help stop sensor data from flooding external software with MIDI or OSC messages or may be used to achieve interesting quantization effects. Although many mobile applications provide access to the accelerometer, the author believes that Control is the only mobile interface application that can also use the Gyroscope and Compass to generate OSC and MIDI messages.

The author is currently in collaborating to provide access to the microphone and video camera sensors. Both will have a variety of signal processing algorithms for users to to obtain data with. Early work has started on using the microphone to obtain finger velocity when striking the screen; this is useful, for example, to assign volume to sounds that are triggered percussively.

3.4. End-User Development

It is not enough to allow users to simply physically arrange pre-defined widgets on a screen. Control enables dynamic interfaces by letting widgets and sensors trigger JavaScript functions; these functions can also be triggered upon reception of MIDI and OSC messages. All sensors and widgets are instantiated as JavaScript objects; Control attempts to ensure that all sensors and widgets are modifiable by the user as much as possible.

There are many simple examples where scripting is necessary in musical interfaces. As one example, the DJ interface included in the Control download has a crossfader for fading between different sounds. To the left and the right of the crossfader are toggle buttons that tell the crossfader to jump to its minimum and maximum values. This is a extremely simple example of one widget controlling the behavior of another, but it is an example that the vast majority of dedicated mobile interface applications are not capable of. In applications without scripting support the toggle buttons would most likely send a control message to a server application but not change the crossfader position. This would make the crossfader position out of sync with the audio being generated by the server.

As a more complex example Control also comes with an interface that runs Conway's Game of Life. Although it is arguably interesting to listen to sonified cellular automata the real purpose of this interface was to show the type of complex, generative behavior that scriptable interfaces support. None of the logic for the Game of Life is included in the primary application code; it is all JavaScript that contained in an external interface file.

Although most widgets are defined in JavaScript files that are bundled with the Control application the scripting support is robust enough that users can define their own widgets. This includes the ability to define drawing routines, hit tests, and how widgets output OSC and MIDI messages.

3.5. Network Support

All client instances of Control broadcast their IP address via Bonjour[5] so that they can be easily discovered by Bonjour/mDNS/Avahi enabled programs. When a server application is notified that a client exists the server can push an interface to the device using OSC. The server can also specify the client's destination IP address and port. The client is presented with a pop-up window letting them accept the new interface and destination or refuse it.

In addition to broadcasting their availability via Bonjour clients also search for Bonjour services that receive OSC or RTP MIDI [4]. Users can select which of these services they would like to send messages to in the Destinations tab. They can also manually add new IP address / port combinations inside the Destinations tab if they want to send messages to server applications that are not Bonjour enabled.

As mentioned previously servers can push new interfaces to clients at any time assuming the client approves this via a pop-up notification. This allows servers to present clients with different interfaces contextually. Use cases for presenting contextual interfaces include different movements of an interactive composition or different locations inside of a virtual reality environment. Servers can also send arbitrary JavaScript functions to clients to be executed. Each interface has access to a custom JavaScript callback for processing OSC messages allowing users to define their own OSC namespaces for handling messages. MIDI messages can be similarly processed via a custom callback.

4. USE IN PERFORMANCE

At the time of this writing Control has only been available for a few months; even so, it has already been used in a wide variety of settings. Users have submitted videos of projects that allow Control to variously manipulate lighting rigs, emulate non-traditional hardware interfaces like the Haken Continuum[6]and the Monome[7], and perform more traditional tasks like mixer emulation. Many of these videos are collected for viewing on the Control website[8].

The author has used Control in to perform live DJ sets, control a simulation of nano-particles attacking a cancer tumor, wirelessly mix rock bands in clubs and to perform in an electronic improvisation group. In the three months that Control has been available it has already been downloaded well over fifteen thousand times from the Apple App Store; the author is thus confident that there will be many more interesting use cases to report about in the coming months, especially with its introduction to the Android Market.

[5]http://www.apple.com/support/bonjour/
[6]http://www.hakenaudio.com/Continuum/
[7]http://www.monome.org/
[8]http://www.charlie-roberts.com/Control/

5. CONCLUSION

Control provides valuable features for creating user interfaces to control musical and artistic applications. In particular, the scripting support of Control provides flexibility in defining interfaces that no other mobile device software can match. Control takes advantage of the wide array of sensors in mobile devices; these sensors provide interactive affordances to interfaces that other similar applications do not take advantage of.

Control has many benefits for use in audience participation pieces and interactive installation art. First, it is free to download. Second, client instances can be autodiscovered when running on the same network as a server application. Finally, servers can push contextual interfaces and trigger JavaScript routines on clients so that interfaces can be dynamically manipulated.

Control is available as a free download for iOS devices from the Apple App Store and is open-source software under the MIT license[9]. It should also be available from the Android Market by the time of this publication. The author would like to thank Greg Shear and Matt Wright for their help with the initial design of Control, and Alex Norman and Matias Wilkman for their work porting Control to run on Android devices.

6. REFERENCES

[1] X. Amatriain, J. Kuchera-Morin, T. Hollerer, and S. Pope, "The AlloSphere: Immersive Multimedia for Scientific Discovery and Artistic Exploration (HTML)," *IEEE MultiMedia*, vol. 16, no. 2.

[2] S. Jordà, M. Kaltenbrunner, G. Geiger, and M. Alonso, "The reactable: a tangible tabletop musical instrument and collaborative workbench," in *ACM SIGGRAPH 2006 Sketches*, ser. SIGGRAPH '06. New York, NY, USA: ACM, 2006. [Online]. Available: http://doi.acm.org/10.1145/1179849.1179963

[3] J. Lazzaro and J. Wawrzynek, "A case for network musical performance," in *Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video*, ser. NOSSDAV '01. New York, NY, USA: ACM, 2001, pp. 157–166. [Online]. Available: http://doi.acm.org/10.1145/378344.378367

[4] M. Wright and A. Freed, "OpenSoundControl: A protocol for communication with sound synthesizers," in *Proceedings of the 1997 International Computer Music Conference*. International Computer Music Association, 1997.

[9]https://github.com/charlieroberts/Control