ARGOS: AN OPEN SOURCE APPLICATION FOR BUILDING MULTI-TOUCH MUSICAL INTERFACES

Dimitri Diakopoulos[1]

California Institute of the Arts[1]
Music Technology
ddiakopoulos@calarts.edu

Ajay Kapur[1,2]

New Zealand School of Music[2]
Sonic Arts
akapur@calarts.edu

ABSTRACT

Argos is a multi-touch graphical user interface builder aimed at applications in musical performance and sound synthesis. The interface builder lets users construct interfaces through a library of existing widgets (e.g. knobs, sliders, buttons), while providing access to an extensible, open-source toolkit for developing new widgets. Argos was designed with platform-independence in mind, enabling it to run on major operating systems in conjunction with many DIY and commercial multi-touch devices.

1. INTRODUCTION

Argos targets a specific need for an open-source, easy-to-use interface builder in the music community. Argos gives users the ability rapidly prototype complex graphical interfaces without writing a single line of code. Argos interfaces can communicate with external programs and programming languages via user-definable OpenSoundControl (OSC) messages. Interfaces built using Argos have been demonstrated in music performance and synthesis through software like ChucK, Reaktor, Max/MSP, SuperCollider, PD, and Ableton Live.

Multi-touch software often uses proprietary or custom libraries to assemble interfaces. One of the core aims of Argos is to provide a suite of C++ classes to facilitate the creation of innovative, experimental UI widgets. As a toolkit, Argos also enables developers to use these widgets in their own applications without the overhead of the interface builder itself.

This paper begins with an overview of related musical multi-touch and interface-oriented software in Section 2. Section 3 discusses underlying mechanics and features unique to Argos. Section 4 details some usage scenarios. We conclude in Section 5 with discussion about future Argos development.

2. RELATED WORK

Argos stems from a lineage of tabletop computing, multi-touch interaction, and unique graphical interfaces. This section gives an overview of the prior academic work and commercial implementations of user interface builders as they have influenced the development of Argos.

Some of the most well known work in tabletop music performance is MTG Barcelona through the evolution of their interface, the reacTable [5]. Along with the Audiopad [8] developed at the MIT Media Lab, these interfaces allow the control of built-in music synthesis parameters using custom-built GUIs. The Audicle [11], while not specifically multi-touch related, is an experimental collection of open-source networkable interfaces designed to transcend traditional GUI interaction for musical performance.

The JazzMutant Lemur[1] is a commercially available multi-touch device that integrates with its own proprietary GUI building software, the JazzEditor. The editor allows users to build custom interfaces and upload the layout to the Lemur. TouchOSC[2], a similar application, lets users construct interfaces for Apple's iPhone and iPad. The MMF (Max Multi-touch Framework)[3] for Max/MSP enables users to create touch-centric interfaces from the built-in Max widgets, but akin to the Lemur and TouchOSC, provides no facility for extending UI widgets or creating new ones.

Philip Davison and Jeff Han [1] have presented work that explains the difficulties and limitations of using proprietary devices and interfaces in the multi-touch domain. The Lemur, TouchOSC, and MMF all use OSC [12] to transmit GUI data to a host, but only MMF supports TUIO [6], a protocol designed to enable direct touch-tracking on many types of vision-based multi-touch hardware.

The SurfaceEditor [7] attempts to resolve the problem of creating an interface builder that works across TUIO-enabled multi-touch devices. The SurfaceEditor does let users create new controls via a Java-based plugin-architecture, but the proprietary licence does not allow core modifications to the builder itself.

We attempt to combine the best aspects of these works into a single package: an open platform for building both common and experimental user interfaces which can be used across a variety of multi-touch capable devices.

[1] http://www.jazzmutant.com
[2] http://hexler.net/software/touchosc
[3] http://www.mathieuchamagne.com/MMF/

3. IMPLEMENTATION

Argos is built from the ground up utilizing openFrameworks[4], a suite of freely-available C++ libraries designed to free developers from low-level, platform-specific implementation concerns. The application contains common features found across many interface builders in general, including the ability to move, resize, rotate, and visually theme controls. Additionally, Argos can save and load interfaces through XML files. Figure 1 provides a simple overview of the architecture of Argos.

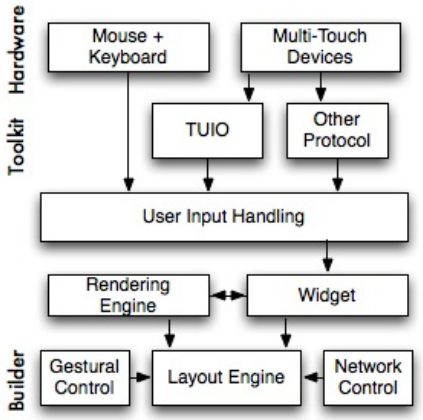

Figure 1. Architecture of Argos

The development of Argos has been split into two separate codebases: a desktop application with the interface builder and performance mode combined, and a mobile-friendly version which only runs the performance mode. The viewer is currently under development to compile against the iPhone & iPad SDK, while versions for recently announced multi-touch tablets including the HP Slate[5], Notion Ink Adam[6], Asus EEE Tablet[7], Fusion Garage JooJoo[8], and Neofonie WePad[9] are planned.

3.1. User Interaction

In the desktop version, the interface builder presents two modes to the user: edit and perform. When editing is activated via a toggle button, a widget editor and browser appear on the edges of the application window. Edit mode is primarily designed for traditional mouse/keyboard use, as the exact layout and configuration of widgets requires more pointing precision than currently available on smaller multi-touch devices. Once a user is finished with the layout, the edit button may be toggled again to prevent accidental editing.

[4] http://www.openframeworks.cc
[5] http://www.hp.com/
[6] http://www.notionink.com/
[7] http://www.asus.com
[8] https://www.thejoojoo.com/
[9] http://www.wepad.mobi/en

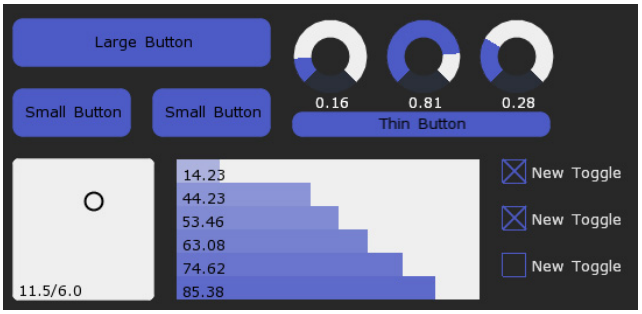

Figure 2. Example interface with buttons, X/Y pad, sliders, toggles and knobs.

From the browser panel, the user is able to drag and drop widgets onto the screen where they snap to a fixed grid. The editor panel binds to a single focused widget, where the user is presented options to fine-tune width, height, x-position, y-position, label, min/max value, network/OSC parameters, and aspects of visual appearance. Widgets are preset with a high-contrast color scheme to facilitate readability on projector-based tabletop surfaces.

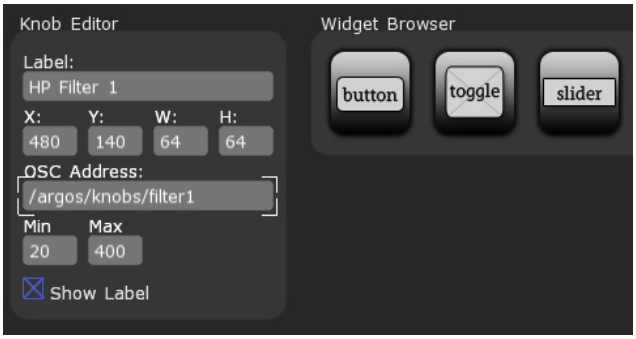

Figure 3. UI showing the editor and part of the widget browser.

3.2. Widget Toolkit

At the sourcecode level, the interface builder can be separated from the underlying library of widgets so that developers can build C++ applications with native multi-touch controls. Although other frameworks such as PyMT [3] and libnui[10] facilitate the development of entire multi-touch applications, we primarily are concerned with forming a sharable, reusable library for rapid UI prototyping.

The Argos source reveals a number of templates designed to simplify the process of developing new widgets. Widgets have direct access to touch data directly associated with them (for example, the location of a finger within a widget). This flexibility leverages the power of multi-touch for handling and interpreting complex user-

[10] http://www.libnui.net/

input cases. Using this system, widgets can be used in combination with different libraries and engines for physics, particle systems, and fluid dynamics. In the musical domain, the inclusion of these libraries enables the creation of widgets that produce unique musical data based on their touch input.

3.3. Platform Compatibility

By using a number of cross-platform libraries wrapped by openFrameworks, Argos was able to assert full operating-system independence across Apple OSX, Microsoft Windows, and many flavors of Linux.

Multi-touch support is maintained across many devices through the use of platform-agnostic touch tracking. While Argos is pre-configured to receive TUIO data from a vision-tracking system such as CCV[11], it can be configured to receive touch events from any platform that provides an API to its internal touch events, such as the Apple SDK. The independence of this system ensures that Argos may be able to use future multi-touch devices as they become available in the commercial sector.

3.4. Mobile Networking

A current developmental feature is the capability to quickly share common interfaces across multiple multi-touch devices. While large tabletop surfaces intrinsically mediate multi-user collaboration, it is hard to envision collaborative interfaces as commercial trends currently favor smaller devices.

In the mobile version, a key element is the ability to load layouts from a remote source in near real-time. This feature is similar to aspects of the Mrmr[12] project which aims to provide a standardized methodology of pushing graphical interfaces to mobile devices for interactive art and media installations. The ability to share and load interfaces on-the-fly has a great potential to create compelling multi-user and collaborative environments on smaller tablet-format devices.

3.5. Gesture Recognition

In the non-mobile version, Argos incorporates the Sparsh-UI [9] engine for multi-touch gesture recognition. The goal of gesture integration is not the control of widgets themselves, but to promote the possibility of using a tabletop surface to build interfaces as well as use them. A few example gestures include the use of a “swipe” to switch layouts, the use of a “scratch” to indicate an undo, or a “triple-tap” to indicate switching between editing and performing modes.

4. DISCUSSION

Argos has been tested in a number of scenarios dealing with live music performance and audio synthesis. The Bricktable [4] was used for usability testing on a 50” tabletop surface.

4.1. Ableton Live

Inspired by MonoTouchLive[13], a static interface designed for single-touch screens, we designed an experiment for users to control a live electronic music set through Ableton Live. Since Live does not natively support OSC, we used OSCulator[14], an OSC to MIDI converter. Live is particularly suited to interfaces built in Argos as the result of an easy-to-use MIDI assignment mode. We asked users to build suitable interfaces and prompted questions about design choices during the process.

Initial responses indicated a high degree of usability between layout design and OSC address assignment. Some participants noted that complex systems like step-sequencing and BPM displays would positively impact the ‘performability’ of their interfaces, suggesting that two-way communication between Argos and host program/language would be a practical feature.

4.2. ChucK

Argos was also tested with the ChucK [10] audio programming language to determine convenient uses as a teaching tool. An informal survey revealed that Argos worked well as a front-end for patches made in ChucK, but suffered from clutter due to an overabundance of widgets. Observationally, this clutter problem impacts other musical applications that allow rapid GUI prototyping as well, such as Reaktor. While a more formal study is needed, this problem signifies a potential need for an automatic layout mode that might assist users during the initial design of an interface.

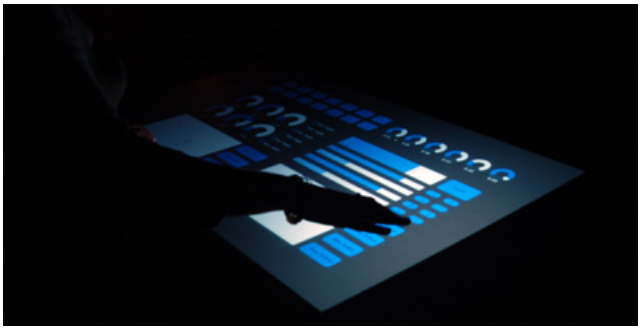

Figure 4. A user interacting with Argos on the Bricktable.

[11] http://ccv.nuigroup.com/
[12] http://poly.share.dj/wiki/index.php/Mrmr
[13] http://www.monotouchlive.com/
[14] http://www.osculator.net/

5. SUMMARY AND FUTURE WORK

We have demonstrated an application and toolkit designed to facilitate the rapid creation of graphical interfaces for music and media applications. Motivated by the absence of a complete system for interactively prototyping expressive software interfaces, Argos stands as an application and toolkit that leverages the multi-touch interaction paradigm to empower musical users and developers.

In the future we plan to integrate ability to bind physical controls & fiducials to the surface of Argos tabletop-based interfaces using a method similar to the one presented by Fiebrink et. al. in [2].

In addition to continuing code-level optimizations, we plan to conduct extensive usability and accessibility tests beyond our preliminary evaluations, especially in the musical pedagogy and interface design domains. Work to extend the widget library with physics-based controls, menus, and other experimental controls is ongoing. Based on previous evaluation, two-way OSC communication is currently being implemented.

A stable pre-release version of Argos is available as C++ source and compiled binary on a Google Code SVN located at http://code.google.com/p/ofxargos/

6. ACKNOWLEGEMENTS

This application was originally a product of the 2009 Google Summer of Code program. Many thanks to Seth Sandler for his helpful comments & ideas about multi-touch interaction design.

7. REFERENCES

[1] Davidson, P. and Han, J. *Synthesis and Control on Large Scale Multi-Touch Sensing Displays*. in *Proceedings of the International Conference on New Interfaces for Musical Expression*. 2006. Paris, France.

[2] Fiebrink, R., et al. *Dynamic Mapping of Physical Controls for Tabletop Groupware*. in *Human Factors in Computing Systems*. 2009. Boston, MA.

[3] Hansen, T.E., et al. *PyMT: a post-WIMP multi-touch user interface toolkit*. in *International Conference On Interactive Tabletops And Surface*. 2009. Alberta, Canada.

[4] Hochenbaum, J. and Vallis, O. *Bricktabke: A Musical Tangible Multi-Touch Interface*. in *Proceedings of the Berlin Open Conference*. 2009. Berlin, Germany.

[5] Jorda, S., et al. *The reacTable*. in *Proceedings of the International Computer Music Conference*. 2004. Barcelona, Spain.

[6] Kaltenbrunner, M., et al. *TUIO: A Protocol for Table-Top Tangible User Interfaces*. in *Proceedings of the International Workshop on Gesture in Human-Computer Interaction and Simulation* 2005. Berder Island, France.

[7] Kellum, G. and Crevoisier, A. *A Flexible Mapping Editor for Multi-Touch Musical Instruments*. in *Proceedings of the International Conference on New Interfaces for Musical Expression*. 2009. Pittsburgh, PA.

[8] Patten, J., et al. *Interaction Techniques for Musical Performance with Tabletop Tangible Interfaces*. in *Proceedings of the Conference on Advances in Computer Entertainment Technology*. 2006. Hollywood, California.

[9] Ramanahally, P., et al. *Sparsh-UI: A Multi-Touch Framework for Collaboration and Modular Gesture Recognition*. in *Proceedings of the World Conference on Innovative VR*. 2008. Brussels, Belgium.

[10] Wang, G. and Cook, P. *ChucK: A Concurrent, On-the-fly, Audio Programming Language*. in *Proceedings of the International Computer Music Conference*. 2003. Singapore.

[11] Wang, G., et al. *Building Collaborative Graphical Interfaces in the Audicle* in *Proceedings of the International Conference on New Interfaces for Musical Expression*. 2006. Paris, France.

[12] Wright, M. and Freed, A. *Open Sound Control: A New Protocol for Communicating with Sound Synthesizers*. in *Proceedings of the International Computer Music Conference*. 1997. Thessaloniki, Greece.