CoPerformance: A rapid prototyping platform for developing interactive artist-audience performances with mobile devices

Bohdan Anderson, Technical Director
ALSO Collective
408-196 Spadina Ave.
Toronto, ON M5T2C2 CA
bohdan@alsocollective.com

Patricio Davila, Principal Investigator
OCAD University
100 McCaul St.
Toronto, ON M5T1W1 CA
pdavila@faculty.ocad.ca

Symon Oliver, Creative Director
ALSO Collective
408-196 Spadina Ave.
Toronto, ON M5T2C2 CA
symon@alsocollective.com

MobileHCI 2013, Aug 27 – 30, 2013, Munich, Germany.
ACM XXX-X-XXXX-XXXX-X/XX/XX.

Abstract
How can mobile technology create more ways for audiences to participate in live stage performances? CoPerformance is a research project that aims to design and develop a set of participatory performance modules that allow for this kind of engagement. These modules will be stand-alone, plug-and-play boilerplate templates and scripts that will allow designers to quickly build and test interactive experiences that utilize mobile devices. CoPerformance can be deployed via mobile browsers or native applications. The goal of this platform is to use existing code—without further abstracting what is already useful—and offer designers and artists a powerful set of tools for interactive performances; and decrease the barrier for entry into this domain.

Author Keywords
Interaction design; platforms; real-time mobile; Node.js; Socket.IO; AngularJS; rapid prototyping; performance

ACM Classification Keywords
D.2.2 Design Tools and Techniques; D.2.10 Design, J.5 Arts and Humanities; K.8.m Miscellaneous

Introduction

Enhancing live music performances has been an ongoing concern for music producers and with the advent of mobile devices the opportunities for participation at live events have increased substantially. Many audiences have become accustomed to recording performances, sending messages during performances or using the built-in light on mobile phones to create spontaneous crowd light shows. Over the past decade HCI researchers have endeavoured to understand how this interaction might work and how it may be enhanced through the use of sensors, wireless networks and mobile computers [4, 5, 6, 7, 8].

Until recently building, distributed mobile interactive performances have presented both a design and technical challenge. Additionally, the frameworks in place to facilitate such works required significant knowledge and skills in server-side languages, and mobile application development. In the past four years, the architecture of the web has changed considerably with the introduction of Google's V8 Engine, HTML5, CSS3, and decreased hosting costs. These changes have lowered the barrier to entry for designing and building rich web applications. As web technologies progress, new platforms and frameworks offer increased utility and distributed connectivity. We are using these new platforms and frameworks in order to develop a series of boilerplate modules that will facilitate the design and build of distributed mobile interactive performances without the need for the deep programming skills often required. In the following text we will cover the design rationale and philosophy of our platform, the technical documentation and feasibility, and our work in progress working with the musical group The Battle of Santiago.

Server-side JavaScript and AngularJS

Built from Google's V8 Engine, Node.js represents a substantial contribution and shift in server-side development allowing for the development of server-side JavaScript applications. Most importantly, Node.js is scalable, event-driven, lightweight, and efficient for real-time network applications across distributed devices [1]. Socket.IO—a Node Package—allows for real-time bidirectional communication between agents (client/server) [3]. The next important advance was the development of AngularJS, a MVW (Model View Whatever), which in short is a framework optimized for dynamic HTML that interacts between client browser and server fluidly [2]. We utilize these three recent frameworks and packages in order to create our Core Module, and a set of boilerplate UI/UX templates. We developed CoPerformance with a philosophy of non-abstraction.

Philosophy of Non-abstraction

The ultimate goal of this project is to further democratize the tools for designing and building engaging interactive performance projects. Although we intend to provide a plug-and-play style architecture, we believe that it is necessary to consider three essential qualities of this platform: ease of use, intuitive and educational, readable and familiar.

We avoid the addition of new semantic layers, and ensure that the code remains written in a syntax that is

Web Application	Native Application
Gyroscope	Gyroscope
Accelerometer	Accelerometer
Speaker	Speaker
Touch	Touch
Unreliable	Camera
Unreliable	Compass
n/a	Flash / Light
n/a	Vibrotactile Feedback
n/a	Notifications
Unreliable	Microphone

Table 1. Shows the availability of input and output features based of the deployment method.

readable as conventional JavaScript. By undertaking an abstraction free development process this platform allows users—new or advanced—to use familiar tools already in use within Node.js, Socket.IO, or AngularJS. This ensures that all techniques that are learned are transferrable to future projects beyond CoPerformance. This approach ensures that CoPerformance remains open and community driven, while avoiding the proliferation of proprietary knowledge and mediums.

Technical Documentation and Feasibility

Within this section we will outline the following Core Module currently being developed. Through the use of Node.js we have built a standalone core application that is device and operating system agnostic. We have chosen four key low-level communication standards for inputs and outputs: Socket.IO (TCP), UDP (which can extend to other UDP based libraries), MIDI (communication to and from musical instruments and mixers), Serial (for sensor inputs and physical outputs such as lighting systems and microcontrollers). Due to the low-level communication standards that are implemented into this platform input/output can be easily received or transmitted from a wide range of languages and applications. In our prototype we are using TouchDesigner from Derivative to visualize user input or augment video based off user input and input from musicians. However, the Core Module is capable of sending data to other frameworks or languages such as Processing, MaxMSP, Pure Data, Ableton, and any other framework that we write a module for. These sub-modules extend the input/output from our Core Module to other applications, enabling them to emit and receive data within the CoPerformance platform.

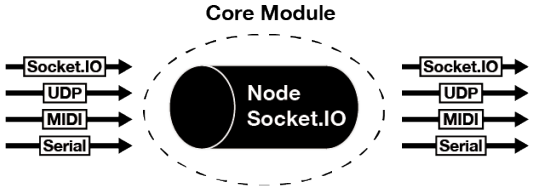

Figure 1: Illustrating the use of our Node.js core application and the hot swappable inputs and outputs to and from Socket.IO (TCP), UDP, MIDI, and Serial.

Depending on the intentions of the user, the platform can be easily deployed as a traditional Web Application, or as a Phone Gap App that allows for access to native features of mobile devices. The distinction is represented in Table 1.

CoPerformance: Core Module

The Core Module is a Node.js server utilizing the Socket.IO package, enabling upwards of 250,000 concurrent bidirectional connections. The main role of the Core Module is to send/receive data from input modules and rebroadcasted to all connected input/output modules.

CoPerformance: Sub-modules

Sub-modules can be thought of as input or output modules acting as satellites to the Core Module. Without the feed from the Core Module, these satellites would only perform predetermined routines and send user specified data. Modules cast for output, receive data from the Core Module, triggering changes in visuals, lighting, or audio. Modules for input rely heavily on the data they receive from the Core Module. These

modules send and receive data to the Core Module, which in turn transmits these inputs to any connected output modules.

CoPerformance: Design and UI/UX Modules

CoPerformance will introduce a set of key UI/UX templates designed and built in AngularJS. These modules will be composed of interface elements, animations, and designs written to consider modern, and best practices of web development. Users are able to customize down to the finest details, or build their own custom interface without the use of these templates. The templates ensure that users are able to get prototypes up and going without the need to write a frontend from scratch.

Current Prototype

At present, our prototype is using Socket.IO and MIDI deployed as a Native Application via Phone Gap, in conjunction with TouchDesigner to deliver an interactive performance. Users are cued via notifications on their mobile devices to use their phones as a form of embodied input, whereby tapping, clapping, shaking, and gyrating become inputs into large-scale visualizations and lighting cues that occur with the musical performance.

Future Work

We would like this work to continue and become widely available to artists working with a variety of scales in audience and participation. To this end we intend on releasing our work as open source using public repositories for code as well as case studies or descriptions of best practices.

References

[1] Node.js. (n.d.). Retrieved June 6, 2014, from http://nodejs.org/

[2] AngularJS — Superheroic JavaScript MVW Framework. (n.d.). Retrieved June 6, 2014, from https://angularjs.org/

[3] Socket.IO. (n.d.). Retrieved June 6, 2014, from http://Socket.IO/

[4] Barkhuus, L., & Jørgensen, T. (2008). Engaging the crowd: Studies of audience-performer interaction. In CHI '08 Extended Abstracts on Human Factors in Computing Systems (pp. 2925–2930). New York, NY, USA: ACM.

[5] Bongers, B. (2000). Physical interfaces in the electronic arts: Interaction theory and interfacing techniques for real-time performance. In Trends in Gestural Control of Music (pp. 41–70).

[6] Brown, B., O'Hara, K., Kindberg, T., & Williams, A. (2009). Crowd computer interaction. In CHI'09 Extended Abstracts on Human Factors in Computing Systems (pp. 4755–4758). New York, NY, USA: ACM.

[7] Freeman, J. (2005). Large audience participation, technology, and orchestral performance. In Proceedings of the 2005 International Computer Music Conference (pp. 757–760).

[8] Ulyate, R., & Bianciardi, D. (2001). The interactive dance club: Avoiding chaos in a multi participant environment. In Proceedings of the 2001 Conference on New Inter- faces for Musical Expression (pp. 1–3). Singapore, Singapore: National University of Singapore.