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 '14, September 23-26, 2014, Toronto, ON, Canada
ACM 978-1-4503-3004-6/14/09.
http://dx.doi.org/10.1145/2628363.2645699

Abstract

How can mobile technology create new models for audience participation in live performances? CoPerformance is a research project that aims to develop a set of plug-and-play participatory performance modules. These modules will allow designers to quickly build/test interactive experiences that utilize mobile devices. CoPerformance can be deployed via web browsers or native applications. The goal of this platform is to use existing frameworks to offer designers a powerful set of tools, templates, and scripts for interactive performances; and decrease the barriers for building participatory performances.

Author Keywords

Interaction design; platforms; real-time mobile; Node.js; Socket.IO; AngularJS; 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

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.

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]. We look to three influential projects for insights into user engagement, distributed real-time interaction, and spontaneous audience organization [9, 10, 11].

Until recently, building distributed mobile interactive performances has presented both a design and technical challenge. Existing frameworks require significant knowledge of server-side languages, and/or mobile development. The relatively recent introduction of Google's V8 Engine, HTML5, CSS3, and decreased hosting costs has changed the web considerably. Since theses changes, there has been an explosion in rich web-based applications being developed by novice developers. As web technologies progress, new frameworks offer increased utility and distributed connectivity. We are using these new frameworks to develop a series of boilerplate modules that will facilitate the design and build of distributed mobile participatory performances without the need for the expert level programming skills. In the following text we will cover our design rationale, technical documentation, and our work in progress with the musical group The Battle of Santiago. We look to three influential projects for insights into user engagement, distributed real-time interaction, and spontaneous audience organization [9, 10, 11].

Server-side JavaScript & AngularJS

Node.js represents a substantial contribution and shift in server-side development, allowing for the development of server-side JavaScript applications. 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]. AngularJS, is a MVW (Model View Whatever), which in short is a framework optimized for dynamic HTML that interacts between client and server fluidly [2]. We are using the above frameworks/packages to create CoPerformance.

Philosophy of Non-abstraction

CoPerformance avoids the addition of new semantic layers, and ensures that the code remains written in a syntax that is readable as conventional JavaScript. By building without abstraction, the platform remains open and familiar. Any techniques learned through the use of CoPerformance are transferrable to future projects beyond CoPerformance. Most importantly the platform will be community driven, while avoiding the pitfalls of managing proprietary knowledge and mediums.

Core Module, Sub-modules, & Communication

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

Acting as satellites to the Core Module, sub-modules extend the input/output from the Core Module to other applications, enabling them to emit and receive data within the CoPerformance platform. Without the feed

Figure 1. The tapper uses accelerometer and gyroscope data from each connected mobile device. Users tap the phone to the drumbeat of the audio performance. This gesture flashes the screen bright white on every detected tap, and sends the tapping rhythm to our main controller. The main controller renders each tap as a white band, revealing a video. Each new user connection subdivides the projected image playing behind the musicians.

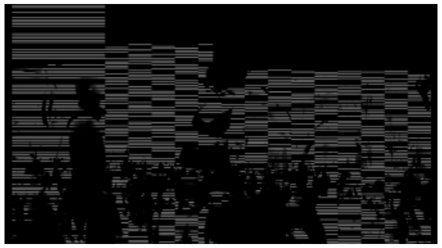
Figure 2. User feedback, as seen on the video projection behind the performers.

from the Core Module, these satellites would only perform predetermined routines. 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 transmit to any connected input/output modules.

We use four low-level communication standards for input/output can be easily received and/or transmitted from a wide range of sources: Socket.IO (TCP), UDP (extends to other UDP based libraries), MIDI (facilitates bidirectional communication to audio platforms), Serial (sensors, microcontrollers, and lighting systems).

UI/UX Modules

CoPerformance introduces a set of key UI/UX templates designed and built in AngularJS. These modules will be composed of interface elements, animations, and templates written to using the 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 speed up the prototyping of these applications and forego the need to write from scratch.

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.

Current Prototype & Future Work

Our prototype is within the testing phase, and will be tested at a live concert with 100 participants. The test will be conducted over the duration of five songs, using two styles of interactions, and three styles of passive visual feedback. The interactions will be: Tapping, and Shaking. TouchDesigner will transmit passive visuals, videos, and drumbeats to all connected mobile devices. Users are cued to interact via notifications on their mobile devices, and/or callouts from the performers. Users become a form of embodied input of the music—whereby tapping, clapping, shaking, and gyrating can become inputs into large-scale visualizations that occur with the musical performance.

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.

[8] [9] Levin, G., et al. (2002). Dialtones (A Telesymphony). Swiss National Exposition '02. Romandy, Switzerland.

[9] [10] Longford, M., Shea, G., King, R. (2011). Tentacles 1.0. MoMA Exhibition: Talk to Me. New York, NY, USA.

[11] Carpenter, L. (1991). Loren Carpenter Experiment. In SIGGRAPH '91. Las Vegas, NV, USA: ACM.