Simplified Expressive Mobile Development with NexusUI, NexusUp and NexusDrop

Ben Taylor
Louisiana State University
1065 Digital Media Center
Baton Rouge, Louisiana
btayl61@lsu.edu

Jesse Allison
Louisiana State University
1065 Digital Media Center
Baton Rouge, Louisiana
jtallison@lsu.edu

Will Conlin
Louisiana State University
1065 Digital Media Center
Baton Rouge, Louisiana
wconli1@lsu.edu

Yemin Oh
Louisiana State University
1065 Digital Media Center
Baton Rouge, Louisiana
yoh1@lsu.edu

Danny Holmes
Louisiana State University
1065 Digital Media Center
Baton Rouge, Louisiana
dholm13@lsu.edu

ABSTRACT

Developing for mobile and multimodal platforms is more important now than ever, as smartphones and tablets proliferate and mobile device orchestras become commonplace. We detail NexusUI, a JavaScript framework that enables rapid prototyping and development of expressive multitouch electronic instrument interfaces within a browser.

Extensions of this project assist in easily creating dynamic user interfaces. NexusUI contains several novel encapsulations of creative interface objects, each accessible with one line of code. NexusUp assists in one-button duplication of Max interfaces into mobile-friendly web pages that transmit to Max automatically via Open Sound Control [14]. NexusDrop enables drag-and-drop interface building and allows users to save and recall interfaces.

Finally, we provide an overview of several projects made with NexusUI, including mobile instruments, art installations, sound diffusion tools, and iOS games, and describe Nexus' possibilities as an architecture for our future Mobile App Orchestra.

Keywords

mobile music, web interface, multimodal, NIME, Web Audio API, OSC, mobile games, mobile apps, websockets

1. INTRODUCTION

Mobile music performance is a growing community [7], driven by development platforms for musical apps [5, 6] as well as commercial controllers like TouchOSC[1] and Mira[2].

The web browser is established as a desirable platform for single-build, multi-platform NIMEs that are compatible across desktop and mobile devices, and that can be easily distributed [13]. Advantages for building browser-based interfaces have been enumerated [8, 9], and several projects have explored the terrain of desktop-to-mobile interface libraries, including massMobile for distributed performance [13] and Interface.js[3] for Gibber [9].

While the aforementioned toolkits should be explored, we present NexusUI as an alternative that expands on the simplicity of TouchOSC but that retains the depth and flexibility of a mobile interface development platform. [2] Benefits of NexusUI including ease-of-code, novel and non-standard musical interface elements, and two paradigms for graphically creating interfaces – NexusUp and NexusDrop – which allow for rapid prototyping and development of interfaces by non-programmers.

As Atau Tanaka notes, the nature of an instrument is both self-sufficient and open-ended [10]. Nexus continues the reimagining of what open-ended could be: open to gesture and touch, but also open to the audience, to other phones, to the World Wide Web and its data.

Whether a musician turns to mobile instrument-making for portability, gesturability, distributability, or accessibility, NexusUI and its extensions attempt to remove the barriers for that musician to develop her mobile NIME.

2. NEXUS USER INTERFACE

In *NEXUS: Collaborative Performance for the Masses, Handling Instrument Interface Distribution through the Web*, we exposited a distributed performance platform using web applications, called Nexus. [3] The user interface portion of that project, NexusUI [1], has become a robust and easy-to-use toolkit for quickly building flexible multimodal interfaces on mobile devices.

As a UI toolkit, NexusUI can simplify development of many existing digital instrument forms, including iOS apps using libPD [4], remote gestural controllers of Max patches with mobile devices, and self-contained Web Audio projects in the browser.

2.1 Implementation

At its core, NexusUI is a JavaScript library of interfaces drawn on HTML5 <canvas> elements. A central JavaScript framework, nexusUI.js, provides interface initialization, unified design templates, transmission protocols, multimodal event listening, web interaction, and a library of common functions that can be drawn upon within individual user interface objects.

NexusUI makes use of the jQuery framework[4], which should

[1] `http://hexler.net/software/touchosc`
[2] `http://cycling74.com/products/mira`
[3] `http://www.charlie-roberts.com/interface/`
[4] `http://www.jquery.com/`

NIME'14, June 30 – July 03, 2014, Goldsmiths, University of London, UK.

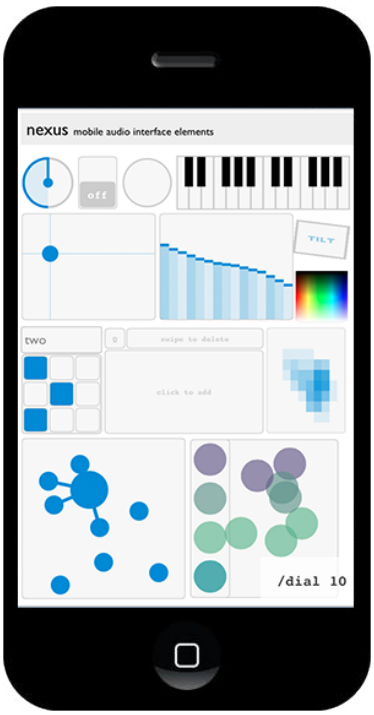

Figure 1: A NexusUI interface on a mobile phone.

be loaded prior to loading nexusUI.js. Linking to jquery.js and nexusUI.js constitutes a full initialization of NexusUI.

2.1.1 *Nexus Manager*

Upon loading NexusUI, the core manager is instantiated as the JavaScript object *nx*. It catalogs all Nexus objects on the page, controls a central animation timer, and provides the shared methods and properties for all objects in the library. For example, the styling of Nexus is centralized so that *nx.colorize()* can modify the color scheme of the entire UI, or *nx.lineWidth* can change the default thickness of drawn lines throughout the UI.

2.2 Interface Objects

NexusUI includes standard audio interaction modes such as **button**, **toggle**, **dial**, **slider**, **multislider**, 2-dimensional **position** slider, **matrix** with variable amplitudes for each matrix point, **number**, **message**, **keyboard**, **item select**, and **comment**.

Focusing specifically on mobile paradigms, a **multitouch** version of the 2-dimensional slider is included which offers five touchpoints that each output x/y data, resulting in continuous control over ten points of data. Also included is an accelerometer **tilt** object which functions on iOS and Android devices, and the Google Chrome browser. The tilt object accesses x and y tilt, and "z" vertical force.

Several novel, non-standard music interaction paradigms are also included. The **colors** interface provides RGB data from a chosen pixel of a colorwheel in response to touch. **Joints** outputs the connections of one node to any nearby node, and relative distance between each node. **Metroball** lets users add bouncing balls into the interface and control them with tilt motion, outputting data on a ball each time it collides with an edge. Finally, **pixels**, is a matrix that you can draw into (in draw mode) or use as a sequencer (in sequence mode).

As described in the Projects section below, camera input and audio input have been incorporated into upcoming NexusUI objects, using the video camera's incoming pixel matrix as a touch field, and letting incoming audio data be utilized as part of your Nexus interface through the web.

2.2.1 *Making an Interface*

To make the library as accessible as possible, creating a Nexus NIME has been kept as simple as possible. Interface objects are created by adding a custom *nx* attribute to any HTML5 <canvas> element to specify which NexusUI object to load. The HTML code for invoking a single NexusUI button is:

```
<canvas nx="button"></canvas>
```

This button, by default, is of an average size (100 pixels), is accessible in JavaScript with the variable *button1*, is both click and touch-compatible, and will transmit its interaction data as an AJAX request to the hosting server. These default settings can be customized upon initialization or updated at any point afterward.

The following five lines of code show the complete body of an HTML document with Nexus multitouch, tilt sensor, and toggle, resulting in ten points of continuous touch control, three axes of continuous motion control, and a toggle that could turn audio on and off.

```
<body>
   <canvas nx="multitouch"></canvas>
   <canvas nx="tilt"></canvas>
   <canvas nx="toggle"></canvas>
</body>
```

In developing Nexus, several methods are provided for creating objects, including JavaScript and HTML paradigms, however, the *nx* HTML attribute has been the method most commonly chosen technique by users.

2.2.2 *Configuring Objects*

NexusUI is designed with a *convention over configuration* approach, meaning objects are given logical and functional default settings wherever possible, but are also customizable to retain flexibility of development. This is evidenced above in the default OSC name assigned to the button, */button1*, which could be changed in JavaScript (.oscName) or in HTML with an ID attribute to be something more descriptive like */volume*:

```
<canvas nx="button" id="volume" style="width:200px">
</canvas>
```

This code would also overwrite the default width of the object, with the GUI adjusting automatically to its new size.

Many objects have configurable interaction modes. For example, Nexus multitouch can operate in normal (continuous) mode, or matrix (discrete or "snap-to-grid" motion) mode. A Nexus button has three modes of interaction:

- **impulse** transmits 1 on touch, nothing on release
- **toggle** transmits 1 on touch, 0 on release
- **node** transmits [1,X,Y] on touch/move, [0,X,Y] on release

Animation and physics are configurable on enabled objects including position and dial, synced to a central *nx* animation timing pulse.

2.3 Transmission Protocols

Configurability extends to the interface's transmission of interactions. Transmission protocols of AJAX, iOS, Android, and local JavaScript callback are available.

By default, all NexusUI objects send AJAX requests back to the hosting server to be handled. Examples are provided in Ruby on Rails, node.js, and a default PHP relay script which passes the data from any incoming requests to localhost as User Datagram Protocol (UDP) OSC messages on port 7475. For this default transmission to function, the page must be hosted and accessed by a server as opposed to simply viewing the file in a browser. This can be easily achieved by using the built in Apache server in OS X or installing a server like XAMPP[5] or MAMP[6] as a turnkey solution.

NexusUI objects can be set to send locally to a JavaScript callback function, passing the interaction data as an array that can be used to affect any JavaScript or DOM element on the page, including Web Audio or to control other Nexus elements.

The final two transmission protocol modes are geared towards embedding nexusUI interfaces into apps built on the iOS and Android platforms. The iOS version uses custom URL schemes to pass control data to the app. The navigation controller hosting the webView based UI implements the *shouldStartLoadWithRequest* and captures any page requests headed to *nexus://*. In Android, this approach results in a delayed interaction, however one can directly tie JavaScript functions in the UI to Java functions in the app which allows for fast UI interactions.

Nexus objects' transmission destinations can be set collectively or individually, so that some objects could transmit via AJAX, another could send to an app, and still others could send to a JS callback function. Here is a JavaScript example of the default transmission protocol being set to iOS transmission, albeit with button1 being dealt with locally to trigger the newNote() function within the browser.

```
nx.onload {
  nx.sendsTo("ios")
  button1.sendsTo("js")
  button1.response = function(data) {
    if (data == 1) {
      newNote(data)
    }
  }
}
```

2.4 Extensibility

The library of NexusUI objects is highly centralized, with all interface objects prototyped from a common template that automatically manages gesture data for each interface object, including multitouch positions, delta motion, and default transmission protocols. This template provides easy maintenance of the library, and makes it straightforward to extend with new objects. Creation of new objects is also simplified by utilizing the shared methods available in the nexusUI.js framework.

The general structure of a NexusUI object is:

1. Declare the object constructor, with arguments of target canvas ID and object index:

   ```
   function dial(target, uiIndex) { ... }
   ```

2. Define self and setup the template:

   ```
   var self = this
   this.uiIndex = uiIndex
   this.defaultSize = { width: 100, height: 100 }
   getTemplate(self, target)
   ```

3. Define attributes unique to the object

4. Initialize any needed parameters

5. Create .draw() function

6. Create click/touch, move, and release functions to handle unique aspects of interaction for this object.

7. Add animation or any other code unique to the UI.

2.5 Advantages of NexusUI

NexusUI takes advantage of the pervasiveness of HTML5 and web enabled mobile devices for cross-platform distributability via URL or QR code, and can be embedded into developer applications. Our convention over configuration approach makes NexusUI easy to get started with while retaining flexibility to customize more complex interfaces.

We add several interfaces to the NexusUI library that have no equivalents in standard UI libraries. Nexus is open-source and extensible for development of more novel objects using the <canvas> drawing surface, a feature that is less accessible in many development platforms.

An additional advantage of NexusUI is its accessibility to non-programmers via NexusUp and NexusDrop described below.

3. INTERFACE BUILDERS

NexusUI's interface building assistants, NexusUp and NexusDrop, extend access to the NexusUI platform and enable rapid prototyping of control interfaces for music, games, robotics, or any other project that can use OSC.

3.1 NexusUp

NexusUp is a one-click method for turning existing Max patches into distributable mobile interfaces with built-in OSC communication. NexusUp duplicates a Max presentation-mode interface as an HTML web interface built with compatible NexusUI objects (Figure 2).

3.1.1 Max bpatcher

NexusUp is encapsulated as a bpatcher that scans a Max patch's presentation mode layout and generates a mirror HTML document using Nexus objects. The Nexus page then sends AJAX messages to a PHP script which forwards the messages via OSC into the patch. The NexusUp bpatcher contains a udpreceive connected to a JavaScript file in Max which updates each object in the patch to new values received from the web interface.

3.1.2 Support

Currently, NexusUp supports many core Max interface objects, including number, slider, dial, button, toggle, message, comment, multislider, and kslider. Some Max interfaces that have no equivalent are translated into nexusUI objects that function in a similar fashion, so that a filtergraph in Max is translated to a 2-D x/y position touch interface, and a gain or live.gain in Max becomes a standard slider in Nexus. Objects like comment and message duplicate their text values from the Max interface.

[5] http://www.apachefriends.org
[6] http://www.mamp.info

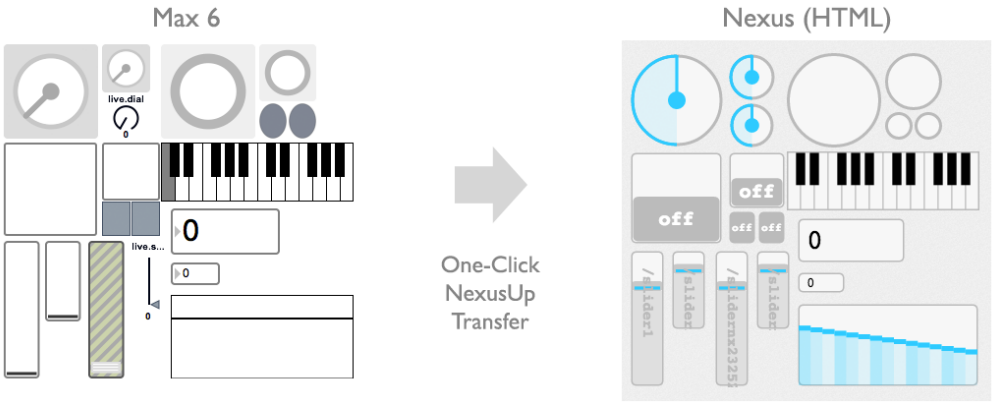

Figure 2: Example NexusUp transfer from Max 6 to a Nexus HTML page.

3.1.3 Setup

To enable NexusUp, a user adds the NexusUI library to the directory of their Max patch and then places the whole project within a web server that has PHP enabled. Once UI objects are added to the presentation mode of the patch, the NexusUp button can be pressed which scans the page and creates an HTML file alongside the patch. This HTML file can then be loaded in any browser to control the patch.

3.1.4 Goals

Different modalities for creating and using NexusUIs are useful for different situations. NexusUp was built to simplify Max integration with NexusUI, make an option for Max users that avoids the extra step of writing HTML code, and to invite Max users to quickly create a UI for mobile performance. Another approach to quickly creating user interfaces is NexusDrop.

3.2 NexusDrop

NexusDrop, a drag-and-drop interface for adding and removing Nexus objects from a webpage, makes NexusUI available to non-programmers and aims to make creating web based interfaces a flexible and seamless process. This has advantages over NexusUp, because users can add and delete objects on-the-fly, and are not restricted to pre-existing Max interfaces. It is also an opportunity for audiences to experiment with creating their own interfaces in a distributed performance setting.

3.2.1 Implementation

NexusDrop uses an edit/perform mode paradigm similar to Max and Pure Data. In edit mode, the webpage provides a list of Nexus objects, and users can click and drag any of those objects onto their interface page, in any number and arrangement. Figure 3 shows NexusDrop in edit mode, with a toggle, tilt, and slider created by click-and-drag. By default, these objects have OSC names */objectN* and send their data to 127.0.0.1 on port 7475, just like NexusUI. The OSC names and destination IP and port can be customized in real-time on the webpage in edit mode. Each object can send to a unique IP, if desired, opening up custom on-the-fly network rerouting within a mobile ensemble an accessible practice.

3.2.2 Reusing Interfaces

Once an interface is made, it can be saved locally using HTML5 localstorage, or uploaded to our central NexusDrop UI database, accessible for use by anyone on the web. An interface could also be exported as an HTML file, to be edited to include Web Audio. NexusDrop is in development to have built-in Web Audio API integration.

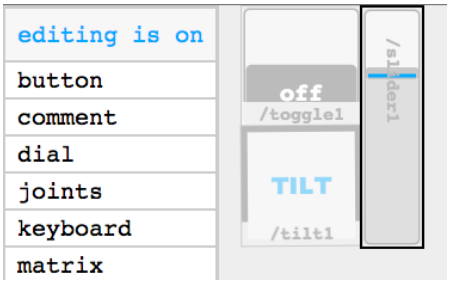

Figure 3: NexusDrop menu of draggable items (cropped) with three created objects and their default OSC names.

4. PROJECTS WITH NEXUSUI

NexusUI has been used in a diverse array of projects in our music and art community, including mobile apps, Web Audio projects, electronic chamber ensemble compositions, games, sound diffusion tools, and sound art installations.

4.1 Mobile Apps

4.1.1 nxPetals

nxPetals is an iPad app using one multitouch object and one tilt sensor. The multitouch object is in matrox mode (10 x 10) and is used to choose up to five frequencies for a Pure Data synthesizer, while the tilt sensor is used as a volume pedal (y-axis) and panner (x-axis). The instrument plays sweeping, slow-attack chords that can be moved in space. *nxPetals* is a standalone iOS app using libPD for audio generation. Nexus communicates with libPD via the browser's URL.

4.1.2 nxChimes

nxChimes, another app using lipPD, has no visible interface and is explored solely with motion. The instrument is played by moving through the three-axes of accelerometer space and triggering sounds when crossing certain thresholds. With a hidden tilt sensor and no indication of what points trigger sound, *nxChimes* is intended to be an interpretation of John Cage's *Inlets*, exploring an invisible and uncontrollable sound source, triggering music unknowingly.

4.2 Web Audio

4.2.1 SawCircle

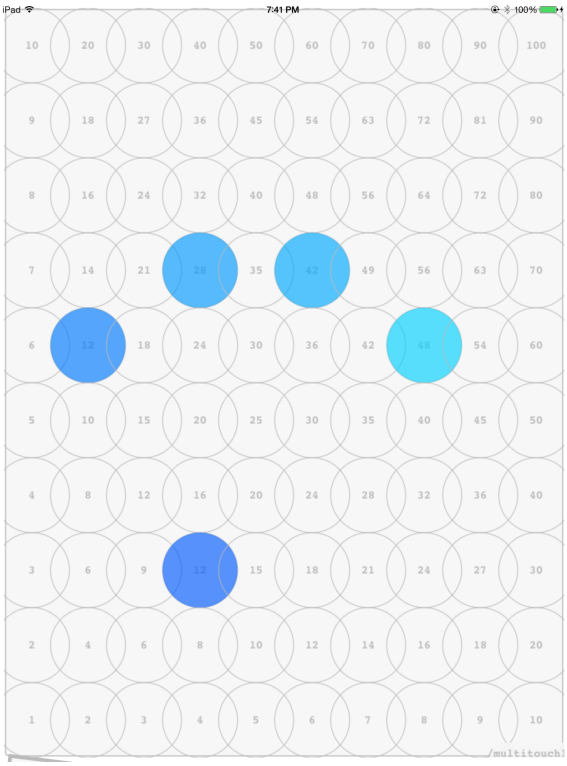

Figure 4: *nxPetals* shows the multitouch object in matrix mode, with numbers denoting overtones

SawCircle is a distributed performance by Trey Duplantis using Web Audio in mobile devices. A QR code is distributed to an audience, most recently a Mardi Gras parade, which loads a NexusUI button interface on their phones. The button triggers polyrhythmic sawtooth waves in just-intonation intervals and irrational tuplets. The composition reimagines a drum circle as a distributed performance of harmonic sawtooth pulses, using a traditional drum circle as a reference point to foster creativity, a goal of distributed performance [11].

4.3 Laptop/Mobile Ensemble

4.3.1 VOX

Vox is an electronic chamber piece for two vocal and two tablet performers by William Conlin. The tablets send OSC messages to laptops from NexusUI, controlling four filters and four delays in Max. Tablet performers use both discrete and continuous control of live audio processing of the vocal performers.

Vox was fully composed in TouchOSC before the NexusUI interface was developed. Upon changing the tablet interface to NexusUI there was no need to change the nature of the composition. The composer found that NexusOSC had several strengths: as a web-based interface it was cross-tablet compatible; NexusUI's networking infrastructure reduced the setup time for performance; and scripting allowed for easy and fast custom styling of instructions. He noted that it was easy to make the new interface almost identical to the previous TouchOSC interface, and more flexible to customize once constructed.

4.3.2 Imogen

A similar process to *Vox* was done with a composition *Imogen* by Lindsey Hartman, a granular remix of Imogen Heap's song *Hide and Seek* performed by a laptop orchestra. Originally using four MIDI controllers for four performers, the interface was ported to an iPad interface of NexusUI sliders and buttons.

The composer found advantages of NexusUI to be its reliability compared to MIDI controllers whose drivers sometimes faulted, and its flexibility, notably her ability to rearrange the Nexus sliders and buttons with HTML to fit the composition (for example, aligning 4 sliders next to each other to be played by 4 fingers of one hand). However, the composer ultimately chose to stick with MIDI controllers because she found the iPad interface vulnerable to accidental touches and wrong notes. We are researching solutions to this critique, including laser-cutting tangible overlays for some Nexus interfaces.

4.4 Game Design

4.4.1 MoonBall

Nexus was used in the artistic game design of an iOS app MoonBall, which incites musical composition through rule-based game dynamics. Moonball is built on the Nexus interface engine for its interaction, animation, and built-in methods, and uses Nexus' transmission protocol to send data to libPD inside the app.

4.5 Spatialization and Ambisonics

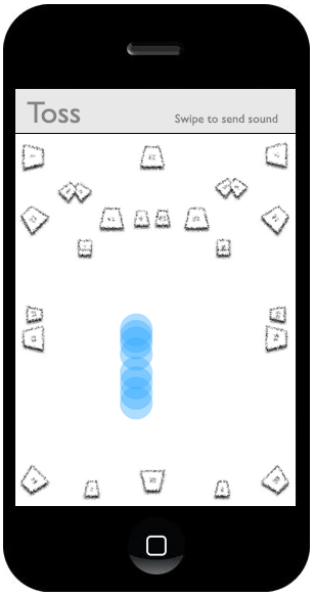

Figure 5: Sending sound around a speaker system in *Environmental Variables:Construction*

4.5.1 Environmental Variables:Construction

NexusUI is being explored as a performative spatialization tool for massive loudspeaker orchestras and sound theaters. Jesse Allison's *Environmental Variables:Construction*, uses a distributed mobile interface to allow an audience to send sound directionally to nearby speakers, enabled by the NexusUI interface sending OSC to a Chuck audio server. The piece was performed collectively with an audience of over 50 members.

4.6 Installation

4.6.1 Humming Mississippi

The sound art installation Humming Mississippi uses NexusUI to integrate web audience participation with the physical artwork. In the piece, transducers resonate planks of wood milled to the depths of a section of the Mississippi River

(Figure 6). A nearby QR code lets viewers load a custom NexusUI interface that simultaneously creates spectrum-based sounds in Web Audio and actuates the transducers in the physical wood sculpture.

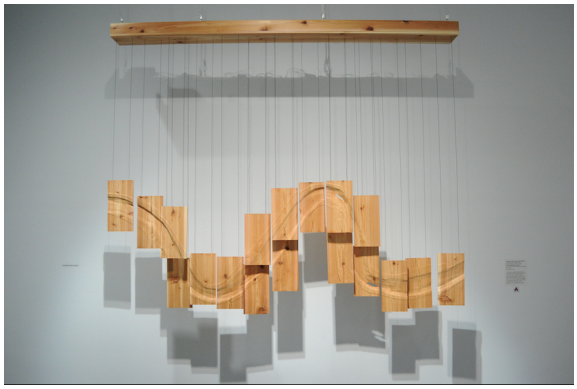

Figure 6: Humming Mississippi installed at the Shaw Center for the Arts, 2013

4.6.2 Graham-O-Phone

Finally, NexusUI also experiments with new HTML5 getUserMedia as a control interface, accessible by the upcoming NexusUI **camera** object. This object will make a live video stream accessible for various modes of data, including its color data or as an audio sequencer. The camera object has already been used for an online video synth, called the Graham-O-Phone, which turns video input into an audio sequencer. It allows a performer to zoom in on a region of space, pixelate it, and use the color data of those pixels in sequence as data to control audio made with Gibberish.

5. FUTURE DIRECTIONS

NexusUI holds significant opportunities for future development and use. The ease of developing and distributing mobile interfaces has encouraged us to develop the Louisiana Mobile App Orchestra (LMAO). NexusUI will be used to control audio generation, performative spatialization, and audience interaction. Notably, the NexusUI joints object has been used as an equal-power sound diffuser around a 72-channel sound theater, allowing live performance control of individual nodes of sound in space with multimodal input from mobile devices.

Key goals in the future development of NexusUI are to more rigorously take advantage of its networked nature, and to critically evaluate the possibilities of a browser-based interface, including the practice of in-browser performance. [12] There is a great opportunity to integrate NexusUI with Application Programming Interfaces (APIs) from other services to retrieve live data from social media, sports, games, or news, and to encapsulate those data streams in easy-to-create NexusUI objects. Imagine using a single line of code, or a NexusDrop draggable interface, to create an interface that could receive social media images (Instagram or Flickr), tweets, and text messages from your audience and to perform with touch, color, or text data from those sources on a touch device. We see Nexus as a burgeoning open-source platform for the development of new, unusual, networked touch-interface maps that break from the paradigm of sliders and buttons, and that can enable musicians to approach mobile devices more creatively.

6. ACKNOWLEDGMENTS

The authors would like to acknowledge the support of the Louisiana State University Center for Computation & Technology, Cultural Computing focus area, and the School of Music. We would also like to acknowledge the support of the Board of Regents for Mobile Initiatives.

7. REFERENCES

[1] Nexus User Interface - http://nexusosc.com.

[2] J. Allison. Nexus, https://github.com/jesseallison/nexus.

[3] J. Allison, Y. Oh, and B. Taylor. Nexus: Collaborative performance for the masses, handling instrument interface distribution through the web. In *Proceedings of the New Interfaces for Musical Expression conference*, 2013.

[4] P. Brinkmann, P. Kirn, R. Lawler, C. McCormick, M. Roth, and H.-C. Steiner. Embedding pure data with libpd. In *Proc Pure Data Convention 2011*, 2011.

[5] N. J. Bryan, J. Herrera, J. Oh, and G. Wang. Momu: A mobile music toolkit. In *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME), Sydney, Australia*, 2010.

[6] G. Essl and A. Müller. Designing mobile musical instruments and environments with urmus. In *New Interfaces for Musical Expression*, pages 76–81, 2010.

[7] L. Gaye, L. E. Holmquist, F. Behrendt, and A. Tanaka. Mobile music technology: report on an emerging community. In *Proceedings of the 2006 conference on New interfaces for musical expression*, NIME '06, pages 22–25, Paris, France, 2006. IRCAM - Centre Pompidou.

[8] T. Melamed and B. Clayton. A comparative evaluation of HTML5 as a pervasive media platform. In T. Phan, R. Montanari, and P. Zerfos, editors, *Mobile Computing, Applications, and Services*, volume 35 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 307–325. Springer Berlin Heidelberg, 2010.

[9] C. Roberts, G. Wakefield, and M. Wright. The web browser as synthesizer and interface. In *Proceedings of the New Interfaces for Musical Expression conference*, 2013.

[10] A. Tanaka. Mapping out instruments, affordances, and mobiles. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 15–18. New Interfaces for Musical Expression, 2010.

[11] A. Tanaka, N. Tokui, and A. Momeni. Facilitating collective musical creativity. In *Proceedings of the 13th annual ACM international conference on Multimedia*, MULTIMEDIA '05, pages 191–198, New York, NY, USA, 2005. ACM.

[12] B. Taylor and J. Allison. Plum st.: Live audiovisual storytelling with remote browsers. *Proceedings of the New Interfaces for Musical Expression conference*, 2013.

[13] F. J. G. S. Weitzner, N. and Y. Chen. massmobile – an audience participation framework. In *Proceedings of the New Interfaces for Musical Expression Conference*, 2012.

[14] M. Wright. Open sound control-a new protocol for communicationg with sound synthesizers. In *Proceedings of the 1997 International Computer Music Conference*, pages 101–104, 1997.