The Mobility is the Message: the Development and Uses of MobMuPlat.

Daniel Iglesia
Iglesia Intermedia; Google, Inc.
California, USA
daniel.iglesia@gmail.com

Abstract

This paper documents the development of MobMuPlat, a software suite for running Pure Data on mobile devices. It chronicles initial ensemble-oriented goals, pre- and post-launch development, and a few common user issues. It also lists a range of projects which use MobMuPlat, highlighting how the reduction of technological barriers yields new and unanticipated results.

Keywords

Pure Data, libpd, mobile, ensemble.

1 Definition

MobMuPlat[1] is a platform, built on libpd[2,3], for running Pure Data (Pd) patches on mobile devices. No text coding is needed. In addition to running a patch, MobMuPlat handles the user interface, networking, interaction with mobile hardware and services (sensors, GPS), and communication with external devices such as MIDI and HID devices. The GUI can be the "native" Pd GUI, or a custom interface with additional features and widgets suited to mobile (e.g. swipable pages, multi-touch). With a goal of development and distribution on any OS, MobMuPlat is four pieces of software: an iOS app, and Android app, an OSX editor application, and a cross-platform (Java Swing) editor application. In addition, multiple networking protocols are implemented for robust group communication, reflecting MobMuPlat's original focus on ensemble performance.

2 Origins

MobMuPlat sprang from a specific mix of interests and frustrations. From 2010 to 2013, I was a co-leader of the Princeton Laptop Orchestra (PLOrk)[4,5], a group comprised of rotating classes of students. I was also a member of its sibling ensemble Sideband[6], comprised of a more stable group of graduate student, staff, and faculty. Due to the fertile and wide-ranging interests of both groups' composers, both groups aim to support a spectrum of hardware and software. The range of audio applications, drivers, and controller hardware would often lead to technical issues, consuming time in long rehearsals. Performers each had a multi-channel hemispherical speaker and subwoofer, combined with all the controllers and cabling required for various works. The sheer amount of physical equipment added logistical issues and long setup times.

My goal was an ensemble performance model to counteract these vexations. It would be self-contained, hand-held, mobile, battery-powered, with minimal software or hardware conflicts, minimal technical learning curve, and no cables to lay down ahead of time. An electronics ensemble should have the mobility of an acoustic ensemble: the ability to walk onstage with minimal setup and perform as self-contained individual sound sources. Rather than sitting, tethered to laptop screens, the hardware should get out of the way, allowing motion and "heads-up" ensemble interaction.

As someone personally invested in electronic ensemble repertoire, I also hoped for an ensemble model requiring a significantly smaller budget, so that the model could exist beyond well-heeled music departments. In both musical and educational contexts, providing each student with a laptop is often infeasible, and can be an unnecessary technical distraction from an musical or educational goal. This lower technical and financial barrier also meant acoustic ensembles could more easily perform pieces with electronics.

Composers for PLOrk and Sideband predominantly use audio software which are unavailable on mobile operating systems (e.g. Max, Supercollider, ChucK); no one wants learn to write native mobile applications for a single work. The fortunate appearance of libpd, however, meant that composers could still develop in a familiar graphical audio environment, and deploy their work to

mobile devices. Graphical environments also allow fast prototyping and tinkering in a way that lower-level text coding does not, making Pd & libpd (which run on nearly anything) a good intersection between composer-friendly and mobile-OS-friendly.

Unlike solo electronics performance, which usually demands complex control of multiple streams of audio, works for LOrks are often fairly technically and instrumentally simple, with a single synth or a sampler and a few parameters exposed to each player. Richness or complexity was not required from any single player, but arose from the combined behavior of the group. Mobile devices fit this model, in which complex GUIs, inter-app routing, DAWs, or other laptop-only qualities were not necessary. In fact, the homogeneity and limitations of mobile operating systems (in which applications have limited power to influence the overall state of the OS) is a benefit, cutting down on incompatibility and driver issues. It also helps that everyone already has one in their pocket.

The missing pieces were:

1. A graphical user interface to control the patch, particularly one that was oriented towards mobile interaction (multi-touch control, swiping through pages of content), specifiable and editable as text. This would be a set of widgets (sliders, knobs, toggles, XY sliders, etc) defined in JSON and rendered by native app code.

2. Leveraging the mobile device's sensors and hardware connectivity (tilt, compass, GPS, MIDI and USB controllers) as control input. These would be routed between the native app layer and the user's patch.

3. Network communication (OSC, and one-to-many protocols that improved reliability over pure multicast). Connectivity (addresses, ports) would be managed natively, and with messages routed to/from the the user's patch.

Once the above were complete, and after initial testing by colleagues, MobMuPlat was publicly released on iOS in January 2013.

3 Post-launch development

The development of MobMuPlat was influenced first my own interests, then by the interests of my personal circle of composers and performers. After launch, many other voices appeared, including early users versed in Pd, users of similar software, users in new performance configurations, and eventually longer-term users with deeper needs and questions. This section tallies MobMuPlat's continued development after initial launch.

Some missing features were apparent, and added shortly after launch. Composers working with dynamic graphical scores had little recourse until an "LCD" object was added, with simplified syntax inspired by Max's LCD object. Complex audio control signals were not expressible/drawable until the "table" object was added, with the ability to directly interact with Pd tables. For users looking to connect mobile to laptop, direct unicast to a user-specified IP and port was added. By popular request, I added Audiobus support, allowing MobMuPlat to be a node in a larger app-to-app audio graph.

Up to this point, MobMuPlat was only on iOS; the Android version was released in late 2014. Though it had the benefit of opening MobMuPlat to many, many more users, my primary motivation was far more specific. The GameTrak tether controller has a unique ability to move electronics performance into physical/gestural space; it had become a staple of electronic ensemble repertoire by this point[7]. HID devices remain publicly unsupported on iOS, while Android, a more open platform, supports them natively. As an initial goal of using mobile devices was to get performers away from a laptop screen, towards physical heads-up performance, then supporting tether input was a high priority. Prime pieces of PLOrk and Sideband repertoire (e.g. Jascha Narveson's *In Line*[8]) were now portable to mobile.

Narveson also designed and wrote LANdini[9], a networking protocol which added guaranteed message ordering and delivery on top of UDP networking. LANdini was an early feature of MobMuPlat, for the same reasons as tether support: to port existing LOrk repertoire to mobile. Its use on mobile devices, however, presented a limitation: power. LANdini was written for plugged-in laptops, and so heavy CPU and messaging load was not a problem. The same load on a mobile device would yield fast battery drain (along with a notable heat). Though LANdini remains supported on MobMuPlat, a leaner counterpart was needed. I wrote the Ping & Connect protocol in response,

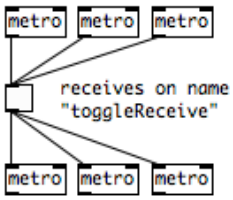
with a simple and unoriginal pattern. Like LANdini, each client pings out their existence (with an optional player index, the better to refer to one another programmatically) and everyone creates direct connections to the pings they receive. This model is a common one, and was baked into the earliest ChucK pieces for PLOrk (e.g. Clix [10]). Though message receipt is not guaranteed, a good router will drop few packets over direct socket connections[11], and the performance and battery usage is improved.

After MobMuPlat had matured this far, I revisited its overall learning curve. Creating a custom interface shouldn't always be necessary, and so I explored adding "native" Pd patch rendering. A user's patch ought to be openable directly, displaying its toggles, sliders, comments, etc. Fortunately, work by Chris McCormick (PdDroidParty for Android[12]) and Dan Wilcox (PdParty for iOS[13]), provided existing examples and reusable code. Their pattern was to have users notate desired Pd GUI widgets with send/receives, communicating their state to the rest of the patch; this works well with libpd's structure, in which the native app layer communicates via send/receive with the audio patch. I, however, endeavored to use patches without this modification, so that users could drop in their patches as-is. This turned out to be non-trivial. A brief overview of the strategy follows.

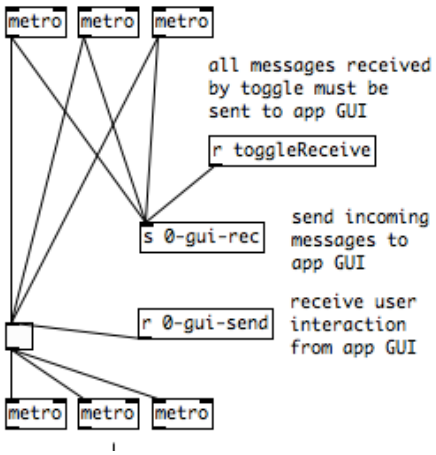
3.1 Native GUI processing

On selecting a patch to open, the patch (as text) is processed into two separate patches. The first is rendered as the app GUI, with auto-generated send/receive names for each widget. The second is the patch actually running within libpd, in which each widget is now surrounded with send/receive objects. All messages flowing to the patch widget are sent to the app GUI to render its visible counterpart; all messages flowing from the app GUI (in response to user interaction), are sent back into the patch widget. I attempt to illustrate this below.

Here's an initial patch state defined by the user:



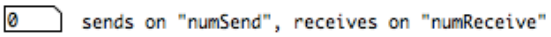
And here's what is generated internally for the patch. `"0-gui-rec"` and `"0-gui-send"` are the handles with which the app GUI widget communicates:



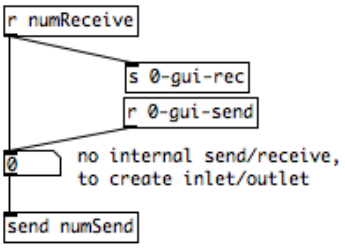
Note that there is no pass-through at the app GUI layer, e.g. something going to `[s 0-gui-rec]` will not trigger an output in `[r 0-gui-send]`.

There are still, however, some complications. Atom widgets (e.g. float number boxes) remove their inlet/outlet if the widget uses a send/receive name. This complicates the ability to connect to/from it. So in this case, the processing logic modifies the float box to not have internal send/receive names, restoring the inlet/outlet, and then manually generate send/receive objects to mimic the intended behavior.

Here's an initial state:



And the post-processing state:



4 Issues and future work

This section covers a few common questions asked by users, combined with some possibilities for future development.

The largest set of questions involves 1) the use of Pd-extended and external objects, and 2) the ability to generate a standalone app for an app store. To some, MobMuPlat's ease of use implies that related tasks are also easy; they are

not. Both these tasks are doable, but involve working with native mobile source code and IDEs, at which point most users understandably lose interest. (The whole point of MobMuPlat, after all, was to avoid needing to do that.) Regarding standalone apps, I believe that unless one wishes to monetize an app (which is difficult without a slick custom interface), or one is paranoid about people looking at one's patch, that installing MobMuPlat (or PdParty, PdDroidParty, etc) and sharing the patch is usually much simpler (and more open). The inability to create and monetize a standalone app also allows the community to avoid the ethical complaints, occasionally made on Pd forums, about open source and personal/commercial gain.

However, the questions regarding sharing patches coincides with a related set of questions regarding mass distribution. A composer/performer may wish to distribute a large-group work (e.g. a sound walk or audience-participation piece) and hope to do so in as few steps as possible. Right now, there is no faster way than installing MobMuPlat, and then downloading the patch and interface files onto the device, and then returning to MobMuPlat to select them. Future work may attempt a method of "pushing" patches to connected devices (though this has the additional hurdle of requiring the group to connect to a local router); this may help for small collaborative groups, but I doubt it will solve the large-group problem.

While I consider development of MobMuPlat to be approaching feature completion, there's still a few more open considerations for the future. Watch and other wearable support is of interest, since it allows even more heads-up musical interaction. MobMuPlat has Android Wear support (for specifying watch widgets that communicate with the mobile device) for some time; I personally use this to control a mobile device while using tether controllers. However I've not yet publicized this feature, nor investigated parity with Apple watches, nor implemented editor support. Additionally, my interest in networking protocols will likely continue, and there's constant discussion of new protocols; if any gain traction with a large community I may consider integration. (At the time of this article, there's some chatter about Ableton's new standard. No promises.)

5 Uses

MobMuPlat was originally conceived with specific uses in mind, but the primary personal benefit of MobMuPlat's adoption has been the surprising ways in which it has been used. Some projects continue common uses of Pd (e.g. solo performance, acoustic instrument augmentation, installation), while some highlight new use as ensemble instrument or embedded device, and many highlight uses completely specific to the nature of mobile devices (e.g. toys, running). This section contains a sample of the ways MobMuPlat has been used.

- **Mobility-based ensembles**: Though mobile-device-based ensembles have existed for some time[14,15], some new ensembles specifically use mobile devices in order to roam around public space. These include an outgrowth of PLOrk[16] (with hand-made controllers), and El Smartphone Ensemble in Colombia[17,18]. I discuss my own mobile ensemble in the following section.

- **Demonstration of scientific research**: MobMuPlat was used to demonstrate how the interaction of fireflies can be a model for synchronization within a group of devices[19,20].

- **Instrument design for solo performance systems**: Toby Hendrick (aka otem rellik) has built a suite of MobMuPlat-based instruments for his live rig[21].

- **Acoustic instrument extension**: For real-time acoustic processing, mobile phones are smaller and less distracting than laptops, and easier to deploy than embedded computers. Some examples include porting a processing chain for trombone performance (including AudioBus)[22] and an embedded processor for violin[23].

- **Concert works with audience interaction**: Celeste Oram recreated Vera Wyse Munro's 1940 experimental work *Skywave Symphony*, using audience's mobile devices in place of radios[24].

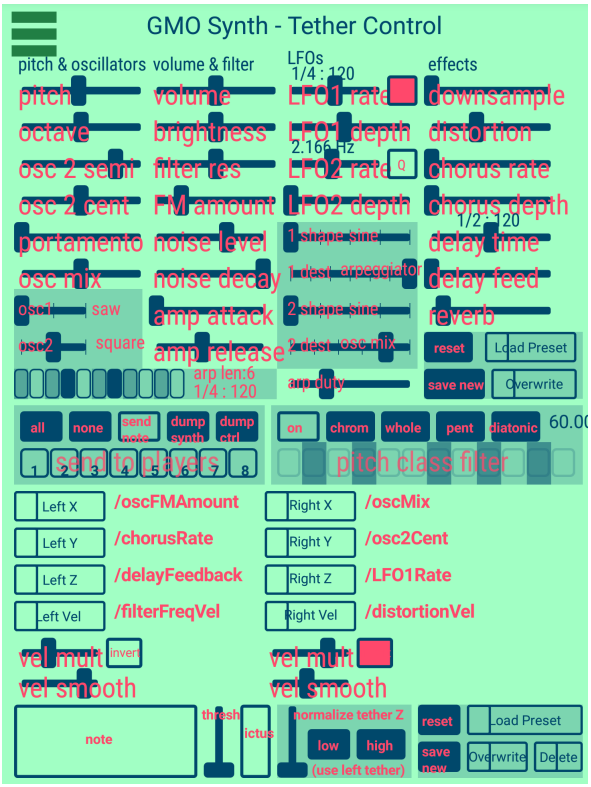
- **Sound mapping, sound walks, and personal surveillance**: Some projects focus on group experience in a specific location, including works for Seoul, Korea by Max Neupert[25] and Hailuoto, Finland by Antye Greie-Ripatti[26]. Others create conceptual works for personal, private experience, such as Oram's *Soft Sonic Surveillance*[27] and Beatrice Monastero's *Wandertroper*[28].

- **Toy development**: One of the most endearing and unexpected uses of MobMuPlat is Notori, a project to recycle mobile phones into toy components[29,30].

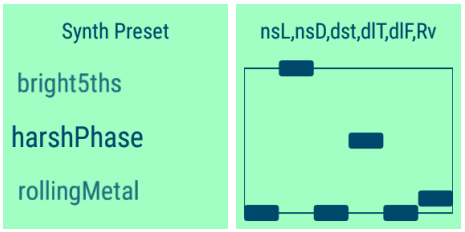
- **Research on running and music**: Multiple projects have used MobMuPlat to measure how music motivates running performance[31,32].

5.1 The Google Mobile Orchestra

The Google Mobile Orchestra (GMOrk), performs on tablets, with tether controllers, portable speakers, networking, and live video. It began in the LOrk model, with a repertoire of works, both adapted from the LOrk repertoire, and newly commissioned for mobile. The LOrk model, based on changing student membership with varying levels of musical experience, results in works often using fairly simple high-level control of a musical process, within a directed score. The members of GMOrk, however, felt that lower-level control, in a more self-directed improvisational context, would suit us better. While we retain the existing externally-composed repertoire, we have developed our own performance patches with many low-level parameters, custom real-time remapping to physical controllers, and heavy use of networking.



A page of a GMOrk interface, enabling control of, and tether mapping to, synth parameters.



Two pages of the accompanying watch GUI, for selecting presets and controlling a subset of synth parameters.

The basis of our networking allows any performer to change any parameter of any other performer. This can be useful to impart timbral or rhythmic unity when desired, and to inject challenging elements (e.g. "everyone send your commands to your left") which yield unanticipated results. This pattern is indebted to early network improvisers, particularly The Hub[33]. As every musical parameter is broadcast over the network, a central computer can then act as a recipient of this traffic, and re-renders the audio of all the players, using the audio to render real-time video for 3D glasses[34].

References

[1] `www.mobmuplat.com`

[2] `www.libpd.cc`

[3] Brinkmann, Peter, et al. "Embedding Pure Data with libpd." *Proceedings of the Pure Data Convention.* 2011. `link`

[4] `plork.princeton.edu`

[5] Trueman, Daniel, et al. "PLOrk: the Princeton laptop orchestra, year 1." *Proceedings of the International Computer Music Conference.* 2006. `link`

[6] `www.sidebandband.com`

[7] Freed, Adrian, et al. "Musical Applications and Design Techniques for the Gametrak Tethered Spatial Position Controller." *Proceedings of the 6th Sound and Music Computing Conference.* 2009. `link`

[8] `www.sidebandchronicles.com/jascha-narveson-in-line/`

[9] Narveson, Jascha, and Dan Trueman. "LANdini: a Networking Utility for Wireless LAN-based Laptop Ensembles." *Proceedings of the Sound and Music Computing Conference.* 2013. `link`

[10] Smallwood, Scott, et al. "Composing for Laptop Orchestra." *Computer Music Journal* 32.1, 2008. `link`

[11] Cerqueira, Mark. "Synchronization over Networks for Live Laptop Music Performance." Master's Thesis, Department of Computer Science, Princeton University. 2010. link

[12] http://droidparty.net/

[13] github.com/danomatika/PdParty

[14] Wang, Ge, Georg Essl, and Henri Penttinen. "Do Mobile Phones Dream of Electric Orchestras?" *Proceedings of the International Computer Music Conference.* 2008. link

[15] Oh, Jieun, et al. "Evolving The Mobile Phone Orchestra." *New Interfaces for Musical Expression.* 2010. link

[16] Snyder, Jeff, and Avneesh Sarwate. "The Mobile Device Marching Band." *Proceedings of the International Conference on New Interfaces for Musical Expression.* 2014. link

[17] sonologiacolombia.wordpress.com/lab/ensamble-de-smartphones/

[18] J. J. Arango and D. M. Giraldo, "The Smartphone Ensemble. Exploring Mobile Computer Mediation in Collaborative Musical Performance." *Proceedings of the International Conference on New Interfaces for Musical Expression.* 2016. link

[19] Nymoen, Kristian, Arjun Chandra, and Jim Tørresen. "Firefly with Me: Distributed Synchronization of Musical Agents." *Awareness Magazine.* 19 November 2013. link

[20] www.vimeo.com/67205605

[21] www.youtube.com/playlist?list=PLpP3G7pBTV-IiLO1oSdR7hI79sktGp3kb

[22] no-insects.blogspot.com/2014/10/meta-trombone-on-ios.html

[23] Overholt, Daniel, and Steven Gelineck. "Design & Evaluation of an Accessible Hybrid Violin Platform." *Proceedings of the International Conference on New Interfaces for Musical Expression.* 2014. link

[24] verawysemunro.nz/re-enacting-the-Skywave-Symphony

[25] choejeongeun.com/xD25-1-GPS-Audio-Walks

[26] medium.com/@poemproducer/when-you-listen-it-listens-back-4c4d99bea8f8

[27] workandthe.work/II-Soft-Sonic-Surveillance

[28] www.dawn.ul.ie/?/interactive-media/2014/BeatriceMonastero/

[29] www.yuichirock.com/notori/

[30] Katsumoto, Yuichiro, and Masa Inakage. "Notori: Reviving a Worn-Out Smartphone by Combining Traditional Wooden Toys with Mobile Apps." *SIGGRAPH Asia 2013 Emerging Technologies.* 2013.

[31] www.marcobasaglia.com/improving-running-motivation

[32] Forsberg, Joel. "A Mobile Application for Improving Running Performance Using Interactive Sonification." 2014.

[33] Gresham-Lancaster, Scot. "The Aesthetics and History of The Hub: The Effects of Changing Technology on Network Computer Music." *Leonardo Music Journal.* 1998. p39-44. link

[34] www.vimeo.com/178834011