PdParty: An iOS Computer Music Platform using libpd

Dan Wilcox
University of Denver
danomatika.com
danomatika@gmail.com

Abstract

This paper presents PdParty, an open-source iOS application for running Pure Data patches on Apple mobile devices using libpd. Directly inspired by Chris McCormick's PdDroidParty for Android and the original RjDj by Reality Jockey, PdParty takes a step further by supporting OSC (Open Sound Control), MIDI, & MiFi game controller input as well as implementing the native Pd GUI objects for a WYSIWYG patch to mobile device experience. Various scene types are supported including compatibility modes for PdDroidParty & RjDj and both patches and abstraction libraries can be managed via a built-in web server. Unlike the rise of the single-purpose audio application, PdParty is meant to provide a platform for general purpose digital signal processing via Pure Data patches.

Keywords

mobile, libpd, wearable computing

1 Introduction

Projects such as PD-Anywhere by Günter Geiger [1] have pioneered using Pure Data as a lingua franca for DSP between desktop & mobile platforms. The 2008 release of Reality Jockey's commercial RjDj application for iOS [2] built upon this work through the concept of user-friendly scenes with bundled content and included access to built-in smart phone sensor events such as multi-touch, accelerometers, and GPS. RjDj "songs" are, in fact, Pure Data patches running live on the users device, allowing for interactive & generative audio beyond simple playback. This approach allowed for both distribution of artist scenes to listeners as well as a platform for computer musicians fluent with Pure Data itself. In 2010, Peter Brinkmann, with help from the RjDj team, released the first version of libpd [3], an embeddable DSP library using the core of Pure Data itself, much to the benefit of the community.

2 Background

Figure 1: *robotcowboy* @ New Media Meeting in Norkörping SE 2009

robotcowboy is the author's ongoing human-computer wearable performance project. Focusing on the embodiment of computational sound, *robotcowboy* was originally built in 2006-2007 as an MS thesis project using an industrial wearable computer[1] running GNU/Linux & Pure Data, external stereo USB sound & MIDI interfaces, and various input devices including HID gamepads. Influenced by roadworthy analog gear, chief system requirements are mobility, plug-in-play, reliability, & low cost. Compositional approaches must include live input/generation and room for failure as opposed to overly sequenced output. [4]

The original *robotcowboy* system hardware was gigged often, went on a 2 month tour of the United States in 2008, and lasted until the 2011 Pd Convention in Weimar. Around this time, Apple released the iPad 2 which featured a dual core processor and, most importantly, supported USB audio & MIDI interfaces. Seeking an option for new system hardware, the author began on and off development of an iOS application that could perform all of the tasks required for a live *robotcowboy* performance: run patches, full duplex stereo audio, MIDI, HID game controller support, & Open Sound Control communication.

[1]Xybernaut MA-V 500 Mhz P3 256 MB RAM, $350 second hand on Ebay.com in 2006

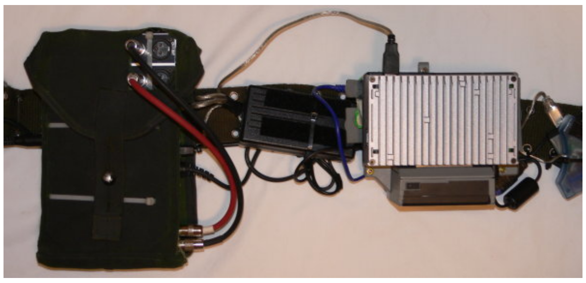


Figure 2: *robotcowboy* hardware 2007: Roland UA-25 audio interface, Xybernaut MA-V wearable computer, USB hub

3 PdParty

The development of what became PdParty began in 2011 as the "robotcowboy app" using the OpenFrameworks C++ creative coding toolkit. Over the course of a year, a set of OF add-on libraries were also developed and/or updated to provide required capbilities: ofxPd to wrap libpd [5], ofxLua as a scripting engine for visual interfaces, and ofxMidi to provide MIDI support. By the fall of 2012, the alpha prototype was working[2] but the author felt the scope of the project had far outgrown the core needs of *robotcowboy* and a slimmed-down approach was needed.

Around this time, Chris McCormick released PdDroidParty for Android which pioneered the concept of emulating native Pure Data GUIs in a mobile device app [6]. In early 2013, PdParty began as a native Objective-C port of PdDroidParty focused on usability as a general purpose platform for running Pure Data patches.

3.1 Focus

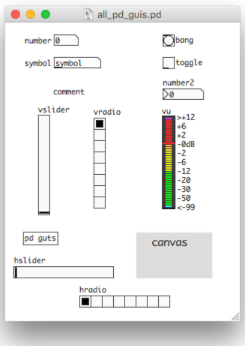


Figure 3: Demo patch in Pure Data on macOS

PdParty is focused on the easy deployment & playback of Pure Data patches on iOS devices. To that effort, great care has been taken to accurately emulate *all* aspects of the built-in GUI objects: number, symbol, comment, number2, bang, toggle, sliders, radios, vumeter, and canvas. This enables a WYSIWYG patch UI experience between desktop & mobile usage as opposed to requiring custom tools and/or programming.

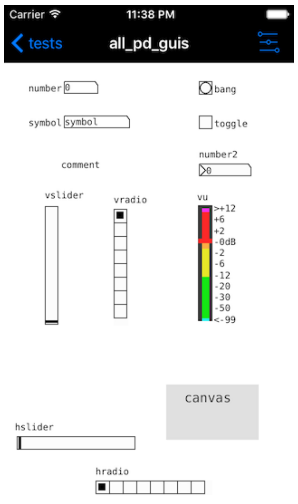


Figure 4: Demo patch in PdParty on iPhone

Like Pure Data itself, PdParty is meant as a general purpose platform for new expression and attempts to stick with Pd idioms as much as possible. Usage should be straight forward and "plug and play."

3.2 Features

The main features of PdParty include a libpd core, native GUI object emulation, scene types, onscreen controls, sensor events, game controller support, MIDI, OSC network communication, and a built-in web server. PdParty is a universal app which runs on both iPhone and iPad with appropriate interfaces and is released as open-source on GitHub.

3.2.1 libpd

PdParty is built around libpd, a wrapper library for the Pure Data vanilla DSP core with an included Objective-C AudioUnit. Patches created in vanilla will work directly in libpd, including those using the `extra` externals including `[expr~]`, `[sigmund~]`, etc. Additionally, the following externals are also included to provide access to directory information & midi files: ggee `[getdir]`, `[stripdir]`, and mrpeach `[midifile]`. As iOS does not allow dynamic library loading, all externals are compiled into the application itself.

[2]"robotcowboy app" alpha demo video: https://vimeo.com/52557228

3.2.2 GUI Emulation

PdParty emulates the Pure Data built-in GUI objects via CoreGraphics drawing routines in native Objective-C. When loading a patch or scene, the main patch is parsed, supported objects are identified by object name, and those with send/receive names are created and added to the main patch view. When interacting with the patch, control messages are intercepted using each object's send and/or receive names via libpd. GUI objects without send/receive names are ignored. Screen orientation is interpreted based on the aspect ratio of the patch canvas itself and emulated GUI object placement is scaled to approximate the original patch on desktop.

Patching a UI for PdParty follows the Model-View-Controller design pattern with GUI objects acting as both view & controller elements which communicate with the core logic of the patch via sends/receives. This approach was adapted from PdDroidParty whose custom GUI objects are also emulated: display, knob, loadsave, menubang, numberbox, ribbon, taplist, touch, and wordbutton.

3.2.3 Scene Types

Beyond plain patches, PdParty supports running "scenes" which are folders with a specific layout that are treated as a single entity for encapsulation and have certain attributes. RjDj scene folders end with ".rj", contain a _main.pd patch, and an optional thumbnail, background image, and Info.plist metadata file. PdDroidParty scene folders contain a droidparty_main.pd patch and optional an background image and .ttf font file. Native PdParty scene folders contain a _main.pd patch and an optional thumbnail and info.json metadata file.

The scene type specifies supported attributes such as required sensors and preferred samplerate. RjDj scenes are locked to portrait on iPhone, touch events are normalized to 0-320, additional sensors are accessed via rj sensor abstractions, and a 22050 Hz samplerate is used. PdDroidParty scenes are locked to landscape, do not require touch or accelrerometer events, and additional sensors are accessed via the `[droidsystem]` object. PdParty scenes infer orientation from patch aspect ratio, normalize touch events to 0-1, and support all sensor types.

3.2.4 Onscreen Controls

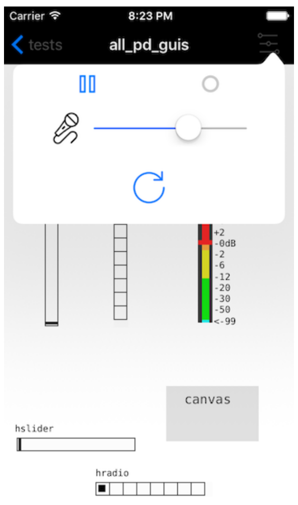
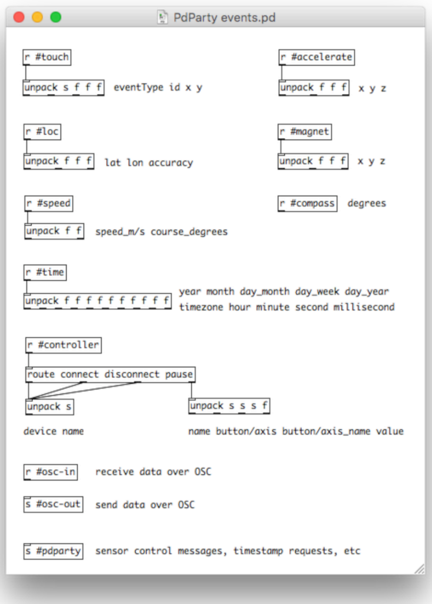


Figure 5: Onscreen control popover on iPhone

Inspired by the original RjDj app, onscreen controls appear either on the scene view itself for RjDj scenes or via a popover view controller for plain patches and all other scene types. Controls are DSP play/pause, record, microphone input level, scene restart, and an optional button to open a console view for debugging. As with RjDj, patches should use the rjlib `[soundinput]`/`[soundoutput]` abstraction wrappers for `[adc]`/`[dac]` which are required to enable the microphone input level and live recording controls.

3.2.5 Sensor and Control Events



```
PdParty events.pd

[r #touch]                           [r #accelerate]
[unpack s f f f]  eventType id x y   [unpack f f f]  x y z

[r #loc]                             [r #magnet]
[unpack f f f]  lat lon accuracy     [unpack f f f]  x y z

[r #speed]                           [r #compass]  degrees
[unpack f f]  speed_m/s course_degrees

[r #time]
[unpack f f f f f f f f f f]  year month day_month day_week day_year
                              timezone hour minute second millisecond

[r #controller]
[route connect disconnect pause]
[unpack s]                    [unpack s s s f]
device name                   name button/axis button/axis_name value

[r #osc-in]    receive data over OSC

[s #osc-out]   send data over OSC

[s #pdparty]   sensor control messages, timestamp requests, etc
```

Figure 6: PdParty event receivers

PdParty provides access to the touch screen, accelerometer, gryoscope, magnetometer, GPS antenna, and built-in compass on an iOS device via events to special receive names starting with a '#'. The `#touch`, `#accelerate`, and `#loc` events match those used by RjDj. Scene types that require specific sensors and events will enable them by default, RjDj scenes for example always receive `#touch` and `#accelerometer` events.

Since some sensors use additional resources when enabled, they must be turned on by the patch or scene that uses them by sending a control message to the `#pdparty` send name, ie:

```
#pdparty loc 1 ; enable gps loc events
#pdparty loc accuracy 10m ; accuracy
```

Additionally, PdParty provides access to timestamp generation sent to the `#timestamp` receiver, manual record cueing, and opening a local or online URL through messages sent to #pdparty.

Furthermore, `[key]` events work with an external bluetooth or USB keyboard[^key]. `[keyup]` and `[keyname]`, however, are not supported as there is currently no official way to intercept raw key events in iOS.

3.2.6 Game Controllers

Compatible iOS MiFi game controllers can be read in PdParty and are hot-pluggable. Controller events are sent to the `\#controller` receive name and iOS supports up to 4 simultaneous controllers.

3.2.7 MIDI

MIDI is supported on iOS via either USB or over Wifi using Network MIDI with a computer running macOS. PdParty detects hot-plugged devices and automatically enables sending and receiving MIDI messages. All Pure Data MIDI objects are supported (`[notein]`, `[ctlout]`, etc).

3.2.8 OSC

PdParty sends and receives OSC (Open Sound Control) messages internally between the libpd instance and a built-in OSC server using liblo, an open-source C library for the OSC protocol. Messages can be received in Pd patches using the `#osc-in` receive name and sent to the `#osc-out` send name. Message parsing and formatting are provided through the `[oscparse]` and `[oscformat]` objects which are part of Pure Data vanilla versions 0.46+.

If enabled, all PdParty events can be streamed over OSC including Pd prints, eg. `#touch` events are sent to the `/pdparty/touch` address. This allows development and debugging of patches and scenes in desktop Pure Data using events streamed from PdParty running on a mobile device.

3.2.9 Browser

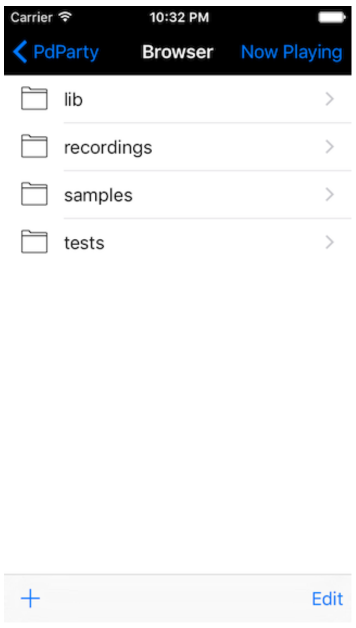


Figure 7: PdParty browser on iPhone

Patches and scenes are managed in PdParty via a standard "drill-down" file browser which displays files and folders. Additionally, common editing controls are supported including delete, rename, move, and copy. Selecting a patch or scene will open it in a patch view. Also .pd patch files and .zip archives are affiliated with PdParty and can be copied and opened from other applications including Mail and DropBox.

3.2.10 Web Server

Patches and scenes can be loaded onto PdParty either using iTunes File Sharing through iTunes or over a local network using the built-in WebDAV web server. The server allows for full access to the PdParty Documents folder and can be enabled from the start screen which also displays the server IP and .local address.

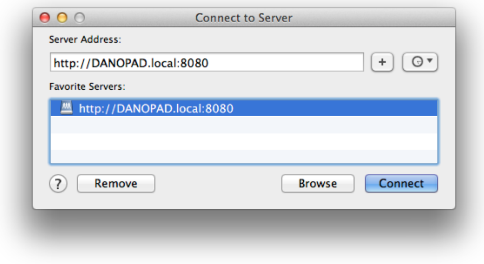


Figure 8: Connecting to the PdParty WebDAV server in macOS Finder

A connection can be made using a file transfer program such as FileZilla or Cyberduck as well as from operating system file managers that support WebDAV including macOS Finder and Gnome Nautilus. Working in this manner allows for live, direct access to the patch files on the device from desktop Pure Data.

3.2.11 Lib Folder

PdParty ships with a default "lib" folder which contains PdParty's required abstractions, allowing for the bundled patches to be upgraded or replaced by the user. Similarly, any subfolders are automatically added to the libpd search path when opening a patch or scene, so it can be used as a central place for abstraction libraries. If the folder or any required abstractions are missing, PdParty falls back to its own internal copy.

3.2.12 App Settings

Important PdParty application settings control behavior, OSC event forwarding, audio latency, and default folder copying. PdParty can be allowed to run in the background and configured to disable the lock screen from appearing. OSC event types can be individually enabled for automatic event forwarding if the OSC server is running. The audio latency can be chosen automatically or set manually by buffer size (64-2048). Last, the contents of the lib, samples, and tests folders can be recursively overwritten by their default files.

3.3 User Guide & Composer Pack

Usage and patching information is detailed in the online PdParty User Guide which includes notes on all event send & receive formats as well as OSC addresses. A composer pack is also available via .zip file which includes notes, scene type templates, and OSC communication patches for desktop Pure Data.

3.4 Development Timeline

The first major alpha version, 0.3.0, was finished in March 2013 and featured native emulation of all built-in Pure Data GUI objects, a patch browser, MIDI support, OSC communication, and a web server for on-device patch management. In September 2013, 0.4.0 alpha was released to testers using the TestFlight framework and included a settings interface, on-screen controls, RjDj & PdDroidParty scene type support, and PdDroidParty custom UI emulation. At the time of writing, PdParty is at version 0.5.6-beta and includes a user guide, composer pack, UI icons, demo scenes, full iOS sensor event support, game controller support, and various bug fixes & improvements.

4 robotcowboy with PdParty

With PdParty, the author now has a stable low latency mobile/wearable platform with a touchscreen, accelerometer, WiFi networking, and USB MIDI/audio. Here is a belt-based wearable setup using an iPhone, Camera Connection Kit, powered USB hub, Roland Edirol UA-25 USB audio interface, and a Behringer direct box (the latter two are built in the case on the left):

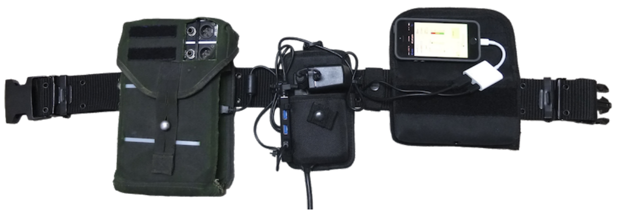


Figure 9: Prototype *robotcowboy* belt with iPhone 2016

5 Future

Although PdParty is largely feature complete, new developments are always possible since "software is never finished."

5.1 Multiple Patch Views

Currently, PdParty's patch view only displays GUI objects loaded from the main scene patch. It may be useful to be able to display multiple GUI patches in either separate tabs or from within a temporary modal patch view. One possible use case could be to open a mixer view from a running patch.

5.2 Link

Ableton Link[3] is a cross-device protocol for tempo synchronization which was released as open-source for iOS and desktop computers in 2016. Although not a new concept, Ableton's clout as a music software company will most probably push Link's adoption on many music environments and platforms in the future. PdParty could integrate Peter Brinkmann's abl_link~ external to send and receive Link messages within patches. [7]

5.3 AudioBus

AudioBus[4] is an iOS library for routing audio between multiple apps running on the same device. As PdParty is a general purpose Pure Data DSP platform, it is a natural fit as a node within the overall ecosystem of iOS audio applications. AudioBus support was not one of PdParty's main requirements but could be added in the future.

5.4 libpdparty

PdDroidParty can both run scenes as well as be used for creating new Android applications. The core of PdParty (libpd, GUI emulation, event handling, etc) could be similarly spun off as a separate Objective-C library for use when creating custom PdParty applications. Notedly, libpd-based MobMuPlat (Mobile Music Platform) by Daniel Iglesia uses the PdParty GUI emulation classes on iOS [8].

5.5 Patch Editing

PdParty is focused on the running of Pure Data patches and scenes but does not have the capability to edit them. If libpd adds a standard API for communication with the Pure Data GUI, an editing UI could be added for mobile patch creation. Some degree of interaction design and research, however, will be required for adapting a desktop UI idiom to mobile devices.

6 Conclusion

After years of on and off development, the author is happy to finally release PdParty to the Pure Data community. It is hoped PdParty will be a useful tool for musicians seeking alternate performance paradigms on an embedded device they already own. With a growing libpd-based mobile ecosystem, the future of computer music is in your pocket.

[3]https://www.ableton.com/en/link
[4]https://audiob.us

Links

https://github.com/danomatika/PdParty

Acknowledgments

The development of ofxPd was supported by the CMU Frank-Ratchye Studio for Creative Inquiry and director Golan Levin. PdParty is directly influenced by Reality Jockey's RjDj and Chris McCormick's PdDroidParty. Frank Barknecht & Joe White provided insight into the RjDJ scene format. Thanks to Miller Puckette and the Pure Data community for Pd itself.

References

[1] G. Geiger, "PDa: Real Time Signal Processing and Sound Generation on Handheld Devices," in *International computer music conference*, 2003.

[2] P. Kirn, Create Digital Music, Oct-2008 [Online]. Available: http://cdm.link/2008/10/rjdj-responsive-interactive-music-on-iphone-now-available-free-3

[3] P. Brinkmann, P. Kirn, R. Lawler, C. McCormick, M. Roth, and H.-C. Steiner, "Embedding Pure Data with libpd," in *Pure Data Convention Weimar 2011*, 2011.

[4] D. Wilcox, "robotcowboy: A One Man Band Musical Cyborg," Master's thesis, Chalmers University of Technology, 2007.

[5] D. Wilcox, "ofxPd: a Pure Data addon for OpenFrameworks using libpd." [Online]. Available: https://github.com/danomatika/ofxPd

[6] C. McCormick, "DroidParty." [Online]. Available: http://www.droidparty.net

[7] "Ableton Link integration for Pd." [Online]. Available: https://github.com/libpd/pd-for-ios/

tree/master/abl_link

[8] D. Iglesia, “MobMuPlat.” [Online]. Available: http://danieliglesia.com/mobmuplat