PdParty

An iOS Computer Music Platform using libpd

Dan Wilcox

University of Denver

November 18th, 2016

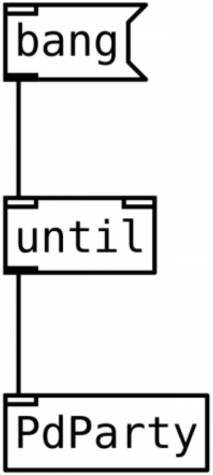


Figure 1: PdParty logo

Background

2003: PD-Anywhere by Günter Geiger

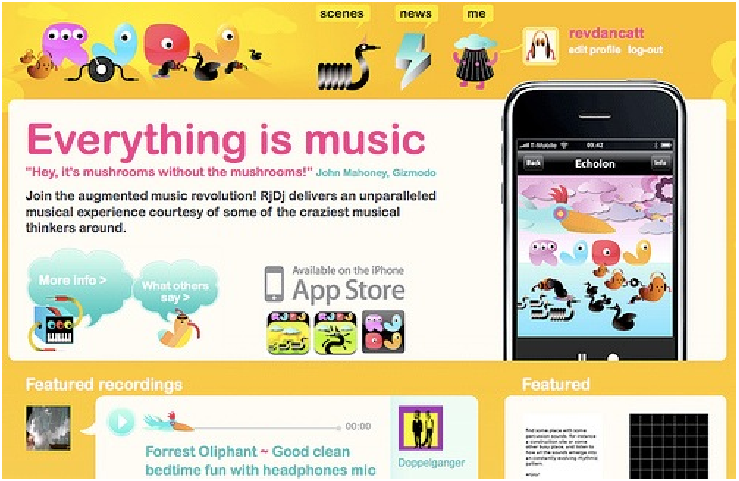
Pure Data for DSP on early mobile devices:

- PDA
- PocketPC
- iPod
- Nokia meamo
- embedded: gumstix, triton, etc

Custom version of the Pure Data core

Integer-only math: support for hardware without floating point

2008: RjDj by Reality Jockey



Everything is music

"Hey, it's mushrooms without the mushrooms!" John Mahoney, Gizmodo

Join the augmented music revolution! RjDj delivers an unparalleled musical experience courtesy of some of the craziest musical thinkers around.

Featured recordings

00:00

Forrest Oliphant ~ Good clean bedtime fun with headphones mic

Doppelganger

Featured

Figure 2: RjDj website, circa 2010

2008: RjDj by Reality Jockey

iOS app using custom wrapper for Pure Data core

User-friendly scenes with bundled content

Access to built-in smart phone sensor events

RjDj “songs” are live Pure Data patches

- Beyond simple playback
- Interactive & generative audio

Platform for both listeners and computer musicians

2010: libpd by Peter Brinkmann, et al

Library wrapper for the Pure Data core

Developed with experience from RjDj

Base platform for Pure Data as a portable, embeddable DSP library

2010: RjDj Sceneplayer by Peter Brinkmann

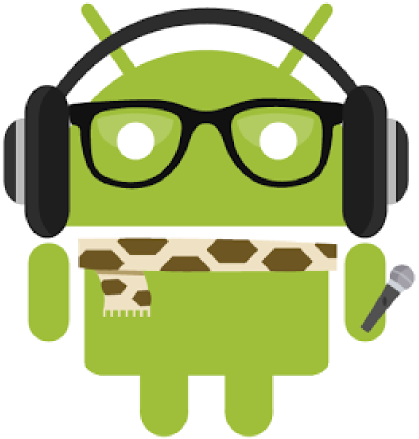


Figure 3: RjDj Sceneplayer logo

robotcowboy

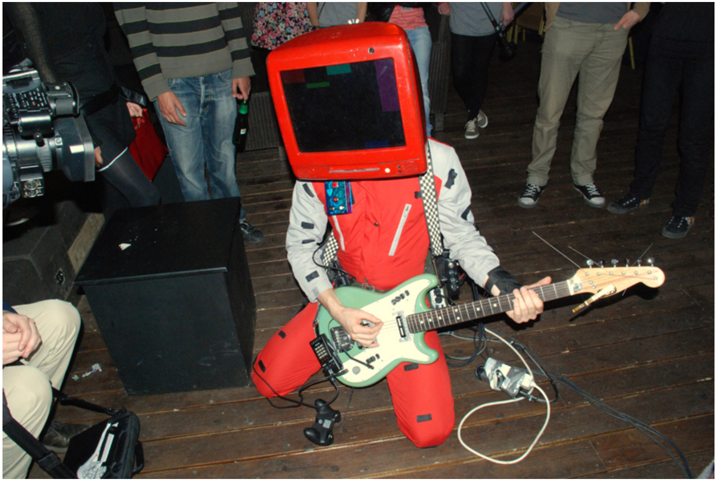


Figure 4: *robotcowboy* @ New Media Meeting in Norkörping SE 2009

robotcowboy

Human-computer wearable performance project

Embodiment of computational sound

Original 2006-2007 MS thesis project using:

- Industrial wearable computer: Xybernaut MA V (P3 500 Mhz 256 MB)
- GNU/Linux, Pure Data, & custom software
- External stereo USB sound & MIDI interfaces
- HID input: gamepads

Roadworthy focus on mobility, plug-in-play, reliability, & low cost

Compositional approach: life input/generation and room for failure

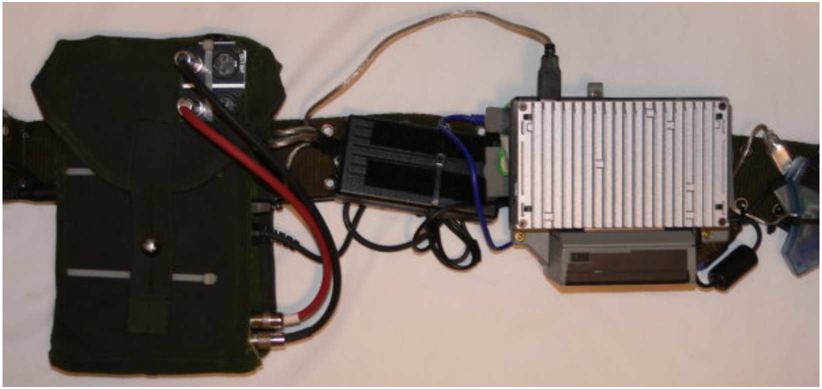


Figure 5: *robotcowboy* hardware 2007: Roland UA-25 audio interface, Xybernaut MA-V wearable computer, USB hub

PdParty

Focus

Main Focus of PdParty

Easy deployment and playback of Pure Data patches

Focus

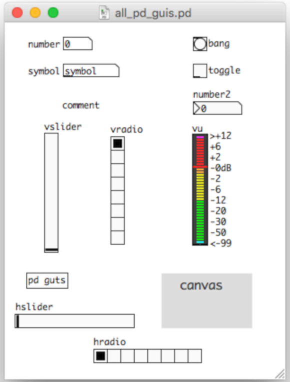


Figure 6: Demo patch in Pure Data on macOS

Focus

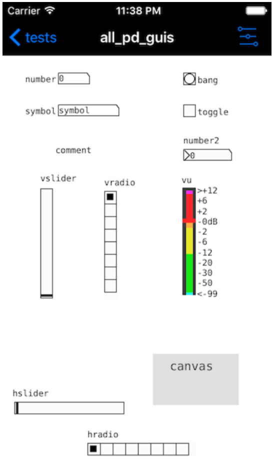


Figure 7: Demo patch in PdParty on iPhone

Focus

WYSIWYG patch UI experience between desktop & mobile usage

Accurate emulation of *all* aspects of the built-in GUI objects:

number, *symbol*, *comment*, *number2*, *bang*,
toggle, *sliders*, *radios*, *vumeter*, *canvas*

Focus

Like Pure Data itself, PdParty is meant as a general purpose platform

Attempts to stick with Pd idioms as much as possible

Usage should be straight forward and “plug and play”

Features

- libpd core
- Native GUI object emulation
- Scene types
- Onscreen controls
- Sensor events
- Game controller support
- MIDI
- OSC network communication
- Built-in web server

Features

Universal app: iPhone & iPad

Released as open source on Github

libpd

Built around libpd: wrapper for Pd vanilla DSP core

Uses libpd’s Obj-C wrapper and AudioUnit

libpd **is** vanilla -> patches created in vanilla will work directly in libpd/PdParty

Included externals:

- vanilla extra: `[expr~]`, `[sigmund]`, etc
- ggee: `[getdir]`, `[stripdir]`
- mrpeach: `[midifile]`

Note: *Apple does not allow dynamic object loading on iOS*

GUI Emulation

Pd vanilla objects are recreated in Obj-C using:

- AppKit input events: touchDown, touchMoved, etc
- CoreGraphics drawing routines: fill/stroke, line, rectangle, circle

GUI Emulation

When PdParty loads a scene or patch:

- main patch is parsed separately
- supported objects identified by name: ie. “tgl”
- objects with send / receive names are added to patch view
- screen orientation interpreted from patch canvas aspect ratio
- object placement is scaled to approximate original patch position

Gui Emulation

Patching a UI for PdParty follows Model-View-Controller pattern:

- core patch logic: model
- GUI objects: view & controller

Communication via send / receive names

Gui Emulation

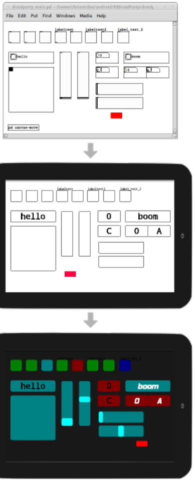


Figure 8: PdDroidParty GUI concept

Gui Emulation

Overall approach adapted from PdDroidParty: patch loading & emulation

PdDroidParty object support:

display, *knob*, *loadsave*, *menubang*, *numberbox*,
ribbon, *taplist*, *touch*, *wordbutton*

Scene Types

Plain *.pd Pd patches (of course)

Scene: a folder with a specific layout that is treated as an encapsulated bundle

Scene folder types:

- RjDj: end with ".rj", contain a "_main.pd" patch, optional metadata "Info.plist" file & background/thumbnail images
- PdDroidParty: contain main "droidparty_main.pd" and optional background image & font files
- PdParty: contain main "_main.pd" patch and optional metadata "info.json" file and thumbnail image

Scene Types

Scene types specify attributes and sensor access

- RjDj: locked to portrait on iPhone, touch events normalized to 0-320, additional sensors accessed via rj sensor abstractions, 22.5k sample rate
- PdDroidParty: locked to landscape, no touch or accelerometer events, additional sensors accessed via the `[droidsystem]` object
- PdParty: infer orientation from patch aspect ratio, normalize touch events to 0-1, support all sensor types

Onscreen Controls

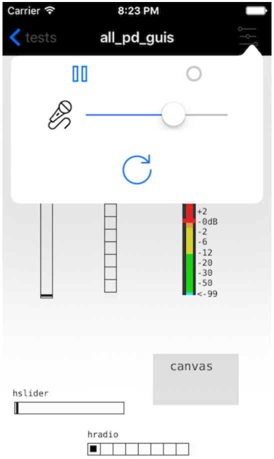


Figure 9: Onscreen control popover on iPhone

Onscreen Controls

Inspired by original RjDj app

Appear either in a popover or on-scene view for RjDj scenes

Controls:

- DSP play/pause
- record
- microphone level
- scene restart
- open the console view (optional)

Patches must use the rjlib `[soundinput]`/`[soundoutput]` abstraction wrappers for `[adc]`/`[dac]` to enable microphone input level and record controls

Sensor and Control Events

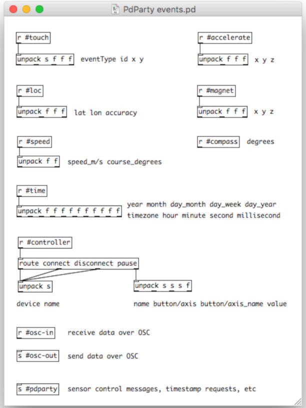


Figure 10: PdParty event receivers

Sensor and Control Events

Supported sensors:

- touch screen
- accelerometer
- gyroscope
- magnetometer
- GPS
- compass

Sensor and Control Events

Sensor events are sent to special receive names starting with a ‘#’: *#touch*, *#accelerate*, *#gyro*, *#magnet*, *#loc*, *#compass*

Some sensors are enabled/disabled based on the scene type, ie. RjDj scenes always receive *#touch* & *#accelerate*

Sensor and Control Events

Some sensors use extra resources & can be enabled via a control message to #pdparty receive name:

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

Sensor and Control Events

Additional control messages via `#pdparty`:

- timestamp generation sent to the `#timestamp` receiver
- manual record cueing
- open a local or online URL

`[key]` events work via external USB / Bluetooth keyboards

`[keyup]` and `[keyname]` do *not* work, no way to grab raw iOS key events

Game Controllers

iOS MiFi game controllers are supported and can be hot-plugged

Controller events are sent to the `#controller` receive name

When plugged in, controller index LEDs are set matching the name of the controller: ie. controller “gc1” is LED 1

iOS supports up to 4 simultaneous controllers

MIDI

MIDI I/O on iOS is supported for USB-compliant MIDI devices

Also works over Wifi using Network MIDI with a computer running macOS

Devices can be hot-plugged and are automatically connected

All Pure Data MIDI objects are supported: `[notein]`, `[ctlout]`, etc

OSC

OSC (Open Sound Control) is supported via a built-in server using the liblo C library

OSC messages are passed between the libpd instance and the liblo server via the `#osc-in` and `#osc-out` send/receive names

Parsing and formatting are provided by the Pd vanilla `[oscparse]` and `[oscformat]` objects

OSC

PdParty events can be forwarded over OSC:

`#touch` events are sent to the `/pdparty/touch` OSC address

Browser

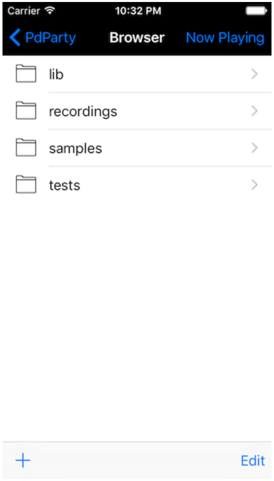


Figure 11: PdParty browser on iPhone

Browsers

Patches & scenes are managed via a standard iOS “drill-down” file browser

Common editing controls are provided: delete, rename, move, copy

Selecting a scene or patch opens it in a patch view

The “.pd” and “.zip” file types are associated with PdParty and can be copied or opened from other applications such as Mail or DropBox

Web Server

Patches & scenes can be loaded onto PdParty using iTunes File Sharing

PdParty includes a built-in WebDAV server which provides full access to the app Documents folder

Enabled from start screen and displays server IP and .local MDNS address

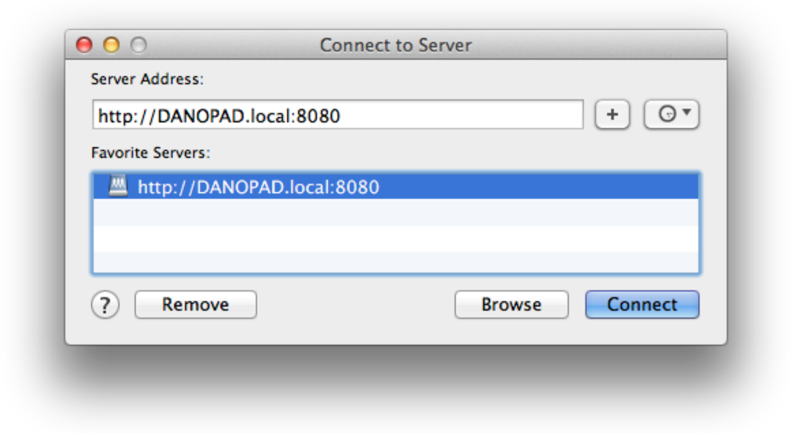


Figure 12: Connecting to the PdParty WebDAV server in macOS Finder

Web Server

Connect to the server with common FTP programs:

- FileZilla
- Cyberduck

Or via OS file managers that support WebDAV including:

- macOS Finder
- Gnome Nautilus

Allows for live, direct access to patching “on the device” from the desktop!

Lib Folder

Special folder in the main PdParty Documents folder: “lib”

Contains PdParty’s required abstractions which can be overriden or upgraded (as per GPL)

Subfolders are automatically added to PdParty search path: global location for abstraction libraries

PdParty falls back to internal “lib” copy if the main folder is missing

App Settings

Important settings are available in a settings view

App Behavior:

- run in the background
- disable lock screen

OSC Event Forwarding

Audio Latency:

- automatic
- manual buffer size: 64 - 2048

Copy Default Folders: lib, samples, tests

User Guide & Composer Pack

Online user guide: `http://danomatika.com/code/pdparty/guide`

Composer pack zip file which includes:

- notes
- scene type templates
- OSC communication patches for desktop

Development Timeline

- Mar 2013 **0.3.0**: first major alpha
- Sep 2013 **0.4.0**: initial beta on TestFlight framework
- Oct 2016 **0.5.6**: first release candidate
- Nov 2016 **1.0.0**: initial release on iOS App Store

robotcowboy with PdParty

With PdParty and iOS, robotcowboy now has:

- stable, low latency mobile/wearable platform
- touch screen
- accelerometer
- WiFi networking
- USB MIDI/audio

Initial wearable setup: iPhone, Camera Connection Kit (USB dongle), powered USB hub, Roland UA-25 USB audio interface, Behringer direct box

robotcowboy with PdParty

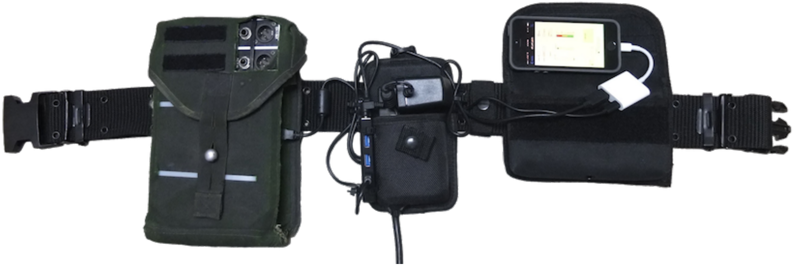


Figure 13: *robotcowboy* belt with iPhone 2016

Future

Never Finished

PdParty is currently feature complete… but software is “never finished.”

Multiple Patch Views

PdParty’s patch view only displays GUI objects loaded from single, main patch

Add ability to display other GUI patches in separate tabs or from within temporary modal patch view

Use case: open a pop up mixer view from a main patch

Link

As strong as its weakest link

Ableton Link: cross-device protcol for tempo synchronization

Released as open-source for iOS and desktop computers in 2016

Not a new concept, but pushed by Ableton’s considerable clout in the DAW scene

Add Peter Brinkmann’s `[abl_link~]` external to PdParty with a possible UI control view

AudioBus

AudioBus: iOS library for routing audio between multiple apps running on the same device

Perfect fit for PdParty as a “general purpose DSP” platform

Add AudioBus linking so PdParty can act as a node within iOS audio ecosystem

libpdparty

PdDroidParty is both an app for running scenes as well as a wrapper for creating self-contained apps

Core of PdParty (libpd, GUI emulation, event handling) could be spun off as a separate Obj-C library

Allow for creation of custom PdParty-based applications

Notedly: Daniel Iglesia's MoMuPlat (Mobile Music Platform) uses PdParty GUI emulation classes on iOS

Patch Editing

PdParty is focused on running Pure Data patches and scenes

No capability to create or edit patches

Could add editing controls and general canvas rendering if/when a GUI communication API is added to libpd

Note: will require research and effort in adapting desktop metaphors to mobile

Conclusion

It’s Alive!

It’s alive! It’s alive!

Finally released after years of on and off development

Hopefully PdParty will be a:

- useful tool for the Pd Community
- platform for alternate performance paradigms

Links

PdParty website: `http://danomatika.com/code/pdparty`

Github: `http://github.com/danomatika/PdParty`

Mobile Music Workshop

Want to know more?

12:10 - 13:30
Saturday Nov 19th
NYU, Room 320

PdDroidParty Chris McCormick (Android)

MobMuPlat Daniel Iglesia (iOS & Android)

PdParty Dan Wilcox (iOS)

The Future

With a growing libpd-based mobile ecosystem, the future of computer music is in your pocket.

Acknowledgments

Development of ofxPd and early libpd work 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 DroidParty

Frank Barknecht & Joe White provided insight into the RjDJ scene format

Thanks to Miller Puckette and the Pure Data community for Pd itself

Thank You