libpd

Past, Present, and Future of Embedding Pure Data

Peter Brinkmann   Dan Wilcox   Tal Kirshboim   Rich Eakin
Ryan Alexander

November 19th, 2016

Introduction

Complete Coincidence

- libpd began in July 2010 as a learning project
- Original goal: run Pd patches on Android devices
- Built on top of preliminary work by Naim Falandino, Peter Kirn, and Hans-Christoph Steiner
- iOS branch grew out of collaboration with RjDj
- Now used in many projects

Current State

libpd and Pd Vanilla

- libpd **is** Pd Vanilla without the GUI or audio backends
- Not a fork of Pd; Pd repo is included as a Git submodule
- The core of libpd is just a thin C wrapper around Pd, with an audio processing function and a message passing mechanism
- Bug fixes to the Pd core are discussed and submitted upstream (e.g. 64-bit related fixes)

Ecosystem

- Language bindings for Java, ObjC, C++, Python, C#
- Audio glue for Android and iOS
- Integration into creative coding environments (OpenFrameworks, Cinder, Processing)
- Integration into Unity game engine
- General purpose mobile apps (RjDj app, PdDroidParty by Chris McCormick, MobMuPlat by Daniel Iglesia, PdParty by Dan Wilcox)

Releases and Distribution

- Core libpd C API largely stable since 2010 (a lock-free queue for passing messages was added later)
- Semantic versioning added in 2014 including a Changelog
- Development workflow:
  - Master branch is largely stable
  - Stable releases are git tagged
  - Minor changes are generally submitted to the master branch
  - Major changes discussed and implemented in topic branches
- Git tagging versions allows for support by package management schemes
- Cocoapods library dependency management for iOS and macOS Xcode projects
- NuGet package management for C# by Thomas Mayer
- JCenter and CircleCI support for Android by Tal Kirshboim and Joe Bowbeer

Documentation

- High-level:
  - libpd website: `http://libpd.cc`
  - libpd book: “Making Musical Apps”
  - sample projects: pd-for-ios and pd-for-android on Github
- Low-level:
  - libpd wiki on Github: build settings, language APIs, working libpd in various software environments
- Supplementary information is community based:
  - mailing lists
  - forums
  - 3rd party text and video tutorials

C++ Wrapper

- Began in early 2011 as part of the ofxPd addon library for OpenFrameworks
- Later adapted for general use and integrated with libpd a year later
- Design influenced by Java wrapper:
  - PdBase: main class which wraps main libpd C API
  - PdReceiver: message receiver base class
  - PdMidiReceiver: MIDI receiver base class
  - PdPatch & PdList: convenience classes for instance identifier and variable list data
- Declared within `pd` namespace

C++ Wrapper

PdBase provides a C++ stream interface for message building and sending:

```
pd << StartMessage() << 1.23
   << "sent from a streamed list"
   << FinishList("fromCPP");
```

- Can be built with C++11 `std::mutex` for thread safety via compiler define
- Can be initialized to use libpd ringbuffer utility layer for messaging
- Currently uses protected PdContext singleton class and can be updated for multi-instance support without C++ API changes

ofxPd

- ofxPd is a C++ addon library for OpenFrameworks
- Audio, messages, and MIDI events can be passed between libpd and the OF run loop
- Implemented as an OF-specific subclass of the libpd C++ wrapper PdBase class which adds:
  - support for multiple message and MIDI receivers
  - routing messages from specific sources to dedicated receiver instances
  - small changes like changing MIDI channel range from 0-15 to 1-16 to match range used in Pd itself

ofxPd

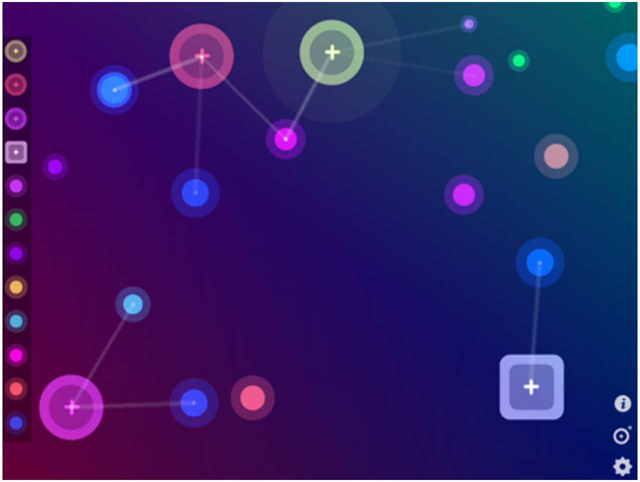
Figure 1: NodeBeat scene on iPad

ofxPd

App examples (tiny selection):

- *Nodebeat*: node-based visual music app for iOS, Android, macOS, and Windows by Seth Sandler, Justin Windle, and Laurence Miller
- *NinjaJamm*: chopper-looper by Ninja Jamm for Android and iOS with high quality curated sample packs
- *Scrapple*: interactive installation by media artist Golan Levin which uses computer vision to interpret the shapes of physical objects on a projected surface as a visual score

Cinder and libpd

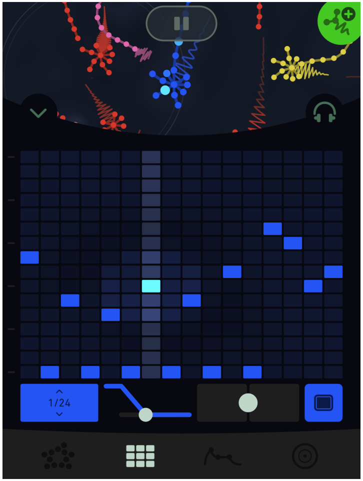

Figure 2: Seaquence on iPad

Cinder and libpd

- Cinder-PureDataNode is an addon for the Cinder creative coding library
- Cinder has its own modular audio processing graph and libpd can be used as a node
- Allows for OpenGL for visuals, platform-specific file decoders and encoders, and low level DSP tools such as sample-rate conversion

Cinder and libpd

- *Seaquence*: iOS app built using Cinder and Cinder-PureDataNode which a stereo spatialized 10 voice, 50 note polyphonic subtractive synthesis engine
- Development iteration time was very short thanks to the ability to build and test the audio engine in Pure Data
- Input and collaboration with the synthesis community also benefited from using Pure Data

Future Development

Concurrency and Realtime Safety

- Good practices for audio development
  - No locks
  - Don't allocate memory on audio thread
- But...
  - Locks are often hard to avoid when working with legacy code
  - Popular operating systems are not real-time systems, i.e., glitches are always possible
  - Not black and white, better to think in terms of "mean time between glitches"
  - So, the question of whether to use locks is just another tradeoff
- Then again...
  - Let's reduce the need for locking, e.g. by revising the Pd symbol table
  - That'll also help with multi-instance support

libpd as a Core for Alternate GUIs

- Create alternate editing GUIs on top of libpd
- Enable different flavors of Pd sharing the same upstream development source
- libpd with GUI messaging would facilitate the creation of esoteric editing environments (e.g. ncurses-based ASCII GUI, patch edition on mobile devices)
- Jonathan Wilkes's GUI messaging specification for Purr Data provides a roadmap

Conclusion

Acknowledgments

- Many contributors...
  - Chris McCormick, Dan Wilcox, Hans-Christoph Steiner, Joe Bowbeer, Martin Roth and RjDj, Miller Puckette, Naim Falandino, Peter Brinkmann, Peter Kirn, Richard Eakin, Richard Lawler, Ryan Alexander, Tal Kirshboim, Tebjan Halm, Thomas Mayer...
- ... but we still need help!
  - Windows developers
  - Release engineers, especially for Processing library and Link external
- Finally, many thanks to everybody who's building awesome stuff with libpd!