Latest developments with Pd-L2Ork and its Development Branch Purr-Data

Ivica Ico Bukvic
Virginia Tech SOPA ICAT
DISIS L2Ork
Blacksburg, VA, USA 24061
ico@vt.edu

Jonathan Wilkes
jon.w.wilkes@gmail.com

Albert Gräf
Johannes
Gutenberg-University Mainz
IKM, Music-Informatics
Research Group
Mainz, Germany, 55122
aggraef@gmail.com

Abstract

In the following paper authors and co-maintainers will present Pd-L2Ork and Purr-Data forks of Pure-Data and Pd-extended, the motivation, and unique features. Introduced in 2010 Pd-L2Ork as a Linux-centric variant offers focus on improved usability, including SVG-enabled canvas, and a growing set of built-in objects and externals designed to lower the learning curve, facilitate rapid prototyping, and offer adaptable granularity. Purr-Data is a recent development branch of Pd-L2Ork focusing on a complete GUI rewrite and support for Windows and OSX. It leverages Node-Webkit and its unique affordances, enabling Pure-Data patches to utilize highly efficient canvas and CSS.

Keywords

Pd-L2Ork, Purr-Data, fork, usability, L2Ork

1 Introduction

Pure-Data, also known as Pd, [19] is arguably one of the most widespread audio and multimedia dataflow programming languages. Pd's history is deeply intertwined with that of its commercial counterpart, Cycling 74's Max [20]. A particular strength shared by the two platforms is in their modularized approach that empowers third party developers to extend software's functionality without having to deal with the underlying engine. Perhaps the most profound impact of Pd is in its completely free and open source model that has enabled it to thrive in a number of environments otherwise inaccessible to its commercial counterpart. Examples include custom in-house solutions for entertainment software (e.g. EaPd [16]), Unity3D [7] and smartphone integration via libPD [8], an embeddable library (e.g. RjDj [17], PdDroidParty [18], and Mobmuplat [15]), and other embedded platforms, such as Raspberry Pi [10].

Pd's author Miller Puckette has spearheaded a steady development pace with the primary motivation being iterative improvement while preserving the backwards compatibility and ability to run even oldest of patches in a rapidly growing library of community-generated creative code. Over the past two decades, Pd underwent several growth spurts. The earliest resulted in a rapid expansion of the program's functionality with a large number of supporting libraries, a number of which have delved in extending Pd's core operation. Its unprecedented popularity has challenged the evolving internal API with external library solutions leveraging calls and functionalities that were not in and of themselves finalized or deemed public. With Max's improved usability and a growing user base, the Pd community sought to complement Pure-Data's compelling core functionality with a level of polish that would lower the initial learning curve and improve user experience. In 2002 the community introduced the earliest builds of Pd-extended [4], the longest running Pd variant that was abandoned in 2013 due to lack of contributors to the project. Pd-extended is not the only Pd variant. There were other ambitious attempts, like pd-devel, Nova, and DesireData [5] that were born out of desire for a more nimble development, fundamental changes to the engine, and improved usability.

In 2009 the community led by Hans Christoph Steiner worked with Miller Puckette on refactoring the GUI code that had become overly convoluted and difficult to develop further. The rewrite was an opportunity for a convergence between Pd and Pd-extended. The effort was by and large successful and perhaps in part served as a catalyst for abandoning Pd-extended. Puckette's work on Pd continues to be instrumental in fostering creativ-

ity and curiosity across generations, and as the library of works relying on Pd grows, so does the importance of conservation and ensuring that Pd continues to support even the oldest of patches. This is where Puckette's ongoing stewardship and brilliance is indispensable. The inevitable side-effect of the increasingly conservationist focus of the core Pd is that any new addition has to be carefully thought out in order to account for all the idiosyncrasies of the past versions and ensure there is a minimal chance of a regression. This vastly limits the development pace. As a result, in recent years Pd has seen a resurgence in forks that aim to sidestep usability issues through alternative approaches, including embeddable solutions (e.g. libPd) and custom front ends.

2 Motivation

Introduced in 2009 by Bukvic, Pd-L2Ork [9] started as a Pd-extended variant that builds on version 0.42.5 with a growing number of patches submitted for upstream adoption. The focus was on nimble development designed to cater to the specific needs of the Linux Laptop Orchestra (L2Ork), even if that meant suboptimal initial implementations that would be ironed out over time as the understanding of the overall code base improved and the target purpose was better understood through practice. A part of L2Ork's mission was educational outreach. Consequently, a majority of early additions to Pd-extended focused on the usability improvements, including graphical user interface and editor functions. While some improvements were incorporated into the Pd proper, a growing number of rejected patches began to build an increasing divide between the two code bases. As a result in 2010 Bukvic introduced a separately maintained Pd-extended variant named Pd-L2Ork after L2Ork for which it was originally designed.

2.1 Philosophy

Pd-L2Ork's philosophy grew out of its initial goals and the early development efforts. It is defined by a nimble development with focus on both major and iterative code changes whose first and foremost goal is to streamline behavior and improve usability and stability as quickly as possible even if that means doing so at the expense of backwards compatibility. Despite an ostensibly lax outlook on backwards compatibility, to date Pd-L2Ork remains fully backwards compatible, and has over time included the "-legacy" flag for changes that may be too disruptive to existing users, such as the repositioning of all iemgui objects to reflect a consistent position in respect to their x and y locations on the patch canvas. Another important aspect of this philosophy is releasing improvements early and often even if the actual fix may not be optimal. Early on, such an approach offered opportunities for better understanding of Pd's internal code while having working iterations in the hands of dozens of students of varying educational backgrounds and experience ensured quick vetting of the ensuing solutions. In an attempt to minimize overhead in configuring the programming environment, including installing supplemental libraries, and in part address the potential for binary incompatibility with Pd, Pd-L2Ork provides a single turnkey monolithic solution with all the libraries included in one package.

Over time, as Pd-L2Ork grew in its scope and visibility, it attracted other long-term co-developers, co-maintainers, and community contributors. In 2014 Wilkes joined Bukvic as a co-developer. The same year Mathieu Bouchard briefly joined the team and assisted in code refactoring, with particular focus on streamlining the Tcl-C socket-based communication protocol. The team was further complemented in 2015 by co-maintainer and packager Gräf. The increasingly team-based development became yet another pillar of the Pd-L2Ork's philosophy and the team continues to openly invite others to contribute in whatever capacity may be the most appropriate.

3 Implementation

Pd-L2Ork's code base is increasingly divergent with that of Pd. It consists of over 1,700 bug-fixes, additions, improvements, and backports to the Pd-extended code base which can be split into engine, usability, documentation, new and improved objects and libraries, and scaffolded learning and rapid prototyping. The list below offers highlights to some of the most obvious additions to each of the said categories.

3.1 Engine

Internal engine contributions have largely focused on implementing features and bug-fixes requested by past and existing Pd users. Some of these include patches that have never made it to the core Pd, such as the cord inspector (a.k.a.

magic glass), improved data type handling logic, and support for outlier cases that may otherwise result in crashes and unexpected behavior (e.g. sending a signal which during execution changes the number of sends it is trying to reach before it has reached them all). Additional checks were implemented for the Jack Audio Connection Kit (JACK) [12] audio backend to avoid hangs in case JACK freezes (e.g. when an external soundcard is disconnected without closing JACK). Default sample rate settings are provided for situations where Pd-L2Ork may run headless (without the Graphical User Interface or GUI), so that objects that depend on sample rate can properly initialize, thus removing the need for potentially unwieldy headless startup procedures. Messages with the argument $0 now automatically resolve such value to the patch instance, while the $@ argument can be used to pass the entire argument set inside a subpatch or an abstraction. [trigger] logic has been expanded to allow for static allocation of values, which alleviates the need for creating bang triggers that are fed into a message with a static value and thereby considerably streamlines the patching process.

From a visual perspective, the Tk-based [22] graphical engine has been replaced with TkPath [6] which offers SVG-enabled antialiased canvas. While the ensuing GUI is less efficient than Tcl, in part due to uncertain state of the TkPath development, the internal engine changes offer accelerated displacement and redraw of objects. A lot of effort went into streamlining graph-on-parent (GOP), including proper bounding box calculation and detection, optimizing redraw, and resolving drawing issues with embedded (GOP) patches. Considerable effort went towards implementing consistent behavior. Improvements also focused on sidestepping the limitations of the socket-based communication between the GUI and the engine, such as keyboard autorepeat detection. As a result, the [key] object can be instantiated with an optional argument that enables autorepeat filtering, while leaving the default object behavior backwards compatible.

Another substantial core engine overhaul pertains to consistent ordering of objects in the glist (a.k.a. canvas) stack. Its implementation has helped ensure that objects always honor the visual stacking order on top of each other, even after undo and redo actions. More importantly, it has paved way towards more advanced functionality including advanced editing techniques (see usability below), and a system-wide preset engine. The preset engine consists of two new objects [preset_hub] and [preset_node]. Nodes can be connected to various objects, including arrays, and can broadcast current state to its designated hub for storing and retrieval. Multiple hubs can be used with varying contexts (defined using the optional symbol argument). The state saving references each object depending on its location in the multidimensional stack with predictable object positions. The ensuing system is universal, unaffected by editing actions, abstraction- and instance-agnostic (e.g. using multiple instances of the same abstraction is automatically supported), and efficient, allowing for anything from recording individual states to real-time automation of multiple parameters through periodic snapshots.

Path detection and retrieval for patches, externals, and abstractions has been expanded to improve Pd-L2Ork's ability to locate necessary files. The file path finding logic has been enhanced with universal prefixes, like the @pd_extra and @pd_help that autoresolve to extra and doc folders.

Data structures were also enhanced with the addition of sprites and new ways to manipulate said data. As a result, Pd-L2Ork ships with a game/tutorial designed by Wilkes where the user can navigate the patch with a virtual character who can interact with various objects and by doing so learn the basics of Pd programming language.

3.2 Usability

On the surface Pd-L2Ork builds on Pd-extended's appearance improvements. Under the hood, with the canvas being drawn as a collection of SVG shapes, the entire ecosystem lends itself to a number of new opportunities. The most obvious involve antialiased display, advanced shapes (e.g. Bézier curves that are also used for drawing patch cords), support for image formats with alpha channel (e.g. PNGs), and advanced data structure drawing and manipulation using SVG-centric enhancements.

A majority of usability improvements focus on the editor. The consistent stacking order implemented in the engine has served as a foundation for the infinite undo, as well as to-front and -back stacking options that are accessible via the right-click context menu. Iemgui objects' positions have been updated to ensure consistent be-

havior, namely accurate reflection of x and y values and consistent autopatching object placement. In addition, their size, display font, and label position were made editable through the GUI with the properties dialog values reflecting GUI-edited changes on-the-fly. Their redrawing was optimized and GOP visibility conditions updated to factor in labels. GOP window size and position were similarly retrofitted to support GUI-based adjustments and many of the supporting dialogs were streamlined and improved, including iemgui properties window and the color picker. Iemgui labels natively support spaces and comments provide graceful handling of line breaks that are also saved in backwards-compatible Pd files. Pd-L2Ork has integrated the old autotips patch and improved upon its design. The "tidy up" feature has been redesigned to offer a two-step realignment of objects. The first key press aligns the objects on a single axis, while the second respaces them, so that they are equidistant from each other. Intelligent patching was implemented to provide four variants of automatically generating multiple patch cords based on user's selection, and to provide additional ways of creating multiple connections (e.g. SHIFT + mouse click). The canvas scrolling logic has been overhauled to minimize the use of scrollbars, provide minimal visual footprint, and ensure most of the patch is always visible. Pd-L2Ork supports drag and drop, as well as preliminary support for pasting script code snippets directly onto the canvas. Its documentation browser has been enhanced to support xapian-enabled searching by keywords, as well as annotated navigation of folders and supporting libraries. To minimize confusion in running multiple concurrent instances, Pd-L2Ork introduces the "-unique" startup flag which is by default disabled and whose purpose is to explicitly specify that a new instance should be spawned. This is particularly useful when opening multiple patches from a file browser, thus preventing redundant spawning of multiple instances every time a new file is opened. The copy and paste engine has been overhauled to improve buffer sharing across multiple applications. The entire graphics engine has been made themeable and its settings are by default saved with the rest of the configuration files.

A part of the usability improvements also included the K12 mode (a.k.a. module) [11] accessible via the "-k12" startup flag. This education-centric mode designed specifically for beginners and children offers its own improvements to the user interface, including a sidebar with object buttons that is split into two tabs or categories (control and sound), and a simplified menu and reduced set of options. The ensuing patches can be transported seamlessly between the K12 and default (advanced) modes allowing for Pd-L2Ork to "grow" in its complexity in sync with user's proficiency. We will discuss the K12 mode in greater detail in the Scaffolded Learning section below.

3.3 Documentation

Pd-L2Ork continues to build on the Pd-extended documentation efforts. This includes over 200+ new and updated help files, including the cyclone library documentation. In addition, the K12 library offers a comprehensive documentation of its own, including a growing number of example patches. All of the new help files provide supporting meta info contained within the META subpatch that is also used by the autotips, thereby enabling easier prototyping of abstractions and documentation.

3.4 New and Improved Objects, Abstractions, and Libraries

Apart from the core Pd objects and improvements described in the Engine section above, including [trigger], [preset_hub], [preset_node], and [key], Pd-L2Ork offers a growing number of revamped objects while also pruning redundant and unnecessary objects. By doing so, its long term aspirational goal is to co-locate all the objects in a single folder and thereby abandon the subfolder structure for external libraries altogether.

Special attention was given to supporting the Raspberry Pi (RPi) platform with a custom set of objects designed specifically to harness the full potential of the RPi GPIO and I2C interfaces, including [disis_gpio] and [disis_spi] [10]. The cyclone library has received new documentation and a growing number of bugfixes and improvements. The [coll] object, for example, now offers threaded file reading and writing to prevent potential sample drops when used in complex patches on a low power hardware. Ggee library's [image] has received a significant overhaul and became the catchall solution for image manipulation, including accelerated displacement, support for file formats with alpha channel (e.g. PNGs), and size reporting. In addition to the standard Pd-extended libraries, Pd-

L2Ork has reintroduced the flext library with [disis_munger~] and an upgraded version of the [fluid~] soundfont synth external. Other libraries include fftease, lyonpotpourri, and RTcmix~. Pd-L2Ork bundles advanced networking externals [disis_netsend] and [disis_receive], convenience externals, like [patch_name], and abstractions (e.g. K12 and a growing number of L2Ork-specific abstractions designed to foster rapid prototyping). A few libraries have been removed due to lack of support and/or GUI object implementations that utilize hardwired Tcl-specific workarounds.

Most interpreted languages have mechanisms to do introspection. Pd-L2Ork features a collection of "info" or introspection classes for retrieving the state of the program on a number of levels, from the running Pd instance to individual objects within patches. Four classes provide the basic functionality:

- [pdinfo] reflects the state of the running Pd instance, including dsp state, available/connected audio and midi devices, platform, executable directory, etc.
- [canvasinfo] is a symbolic receiver for the canvas, abstraction arguments, patch filename, list of current objects, etc. The object takes a numeric argument to query the state of parent or ancestor canvases.
- [classinfo] offers information about the currently loaded classes in the running instance. This includes creator argument types, as well as the various methods.
- [objectinfo] returns bounding box, class type, and size for a particular object on the canvas.

While the introspection provided by these classes is relatively rudimentary, it alleviates the need for a large number of external libraries that add missing core functionality. For example, Pd-L2Ork ships with several compiled externals whose purpose is to fetch the list of abstraction arguments. These externals all have different interfaces and are spread across various libraries. Having an interface for fetching arguments that behaves similarly to other introspection interfaces improves the usability of the system. Furthermore, opening up rudimentary introspection to the user increases the composability of Pd. Functionality that previously only existed inside the C code can now be pushed to an abstraction, or to a collection of abstractions. Because abstractions don't require compilation and are written in Pd, they are more accessible to a wider number of users to test and improve them.

With L2Ork's focus on Wiimote devices as the primary ensemble controllers, Pd-L2Ork has furthered the development of the libcwiid that is provided as a custom fork with the most comprehensive support for Wii devices on Linux, including interleaved mode and a more recent variant of the Wii Motion Plus. Similarly, TkPath included with Pd-L2Ork is a fork of what appears to be an abandoned library that includes several bug fixes and improvements.

3.5 Scaffolded Learning and Rapid Prototyping

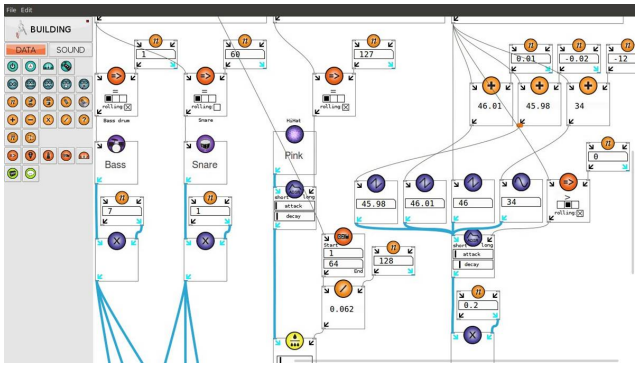

Figure 1: Pd-L2Ork K12 mode.

In 2011 Pd-L2Ork introduced a K12 learning module (Fig.) [11]. Embedded as a startup "-k12" option that modifies the user interface to address the needs of beginners and particularly young learners, the K12 module offers a collection of abstractions that encapsulate common functionality and thereby empowers users to quickly prototype an interactive sound-making instrument. Initially the external device support focused solely on the Wiimote family of controllers. Since its introduction it has been implemented in dozens of Maker-like experiences targeting various age groups [21]. As a result, its functionality was extended to support Arduino and eventually Raspberry Pi [10]. A further simplified K12 module variant was also used for a community-building 50 RPi installation titled Cloud [3] where community participants were given an opportunity to program the behavior of individual "cloudlets" in under an hour. In 2014, the K12 module was used for a gifted summer program where middle- and high-school students programmed and eventually performed in a Raspberry

Pi Orchestra. Most recently, we've begun experimenting with Pd-L2Ork and the K12 module as a rapid prototyping platform in robotics scenarios. This continues to be one of the primary thrust areas of both L2Ork and Pd-L2Ork.

3.6 Limitations

Apart from the ongoing need for further improving documentation and expanding its features, perhaps Pd-L2Ork's greatest limitation is its focus on the Linux platform. With its increased visibility, there was a growing community interest to support other platforms, namely Windows and OSX. The TkPath toolkit, however, lacks Windows support. As a result, there was a need for yet another GUI rewrite. This time its focus required a platform-agnostic solution.

4 Purr-Data Development Branch

The Tcl/Tk toolkit severely limits how usable Pd can be. Tk has no native notion of a hyperlink. It has no easy, deterministic way to separate autorepeat key presses from manual keypress. There is no easy way to use non-system fonts, no support for theming or native dialogs under GNU/Linux, and little default support for many of the common image formats.

When taken separately, each example above could be worked around given a few hours or days of clever playfulness. When taken as a whole, however, all the workarounds necessary to make Tk both responsive and usable across OSX, Windows, and GNU/Linux become too burdensome for a small or even medium-sized graphical free software community. Another, more modern toolkit is needed.

Tcl commands with Tk window strings are hard-coded into the C source files of Pd. This means that any port to a different toolkit must either replace those commands with an abstract interface, or write middleware that turns the hard-coded Tcl strings into abstract commands. Given the complexity of Tcl commands in both the core and external libraries, that middleware would essentially have to re-implement a large part of the Tcl interpreter. Consequently, Purr-Data opted for the former approach of directly implementing an abstract interface.

Adding to the porting difficulty is the fact that Pd has no formal specification, and its GUI interface follows no common design pattern for 2D graphics. For example, the GOP window appears at a glance as a viewport that clips to a specified bounding box. However, the bounding box itself behaves inconsistently–for built-in widgets like [hslider] or [bng] it clips (per widget, not per pixel), but for graphed arrays, data structure visualizations, and widget labels it does no clipping at all.

For this reason, Purr-Data uses a GUI toolkit called nw.js that allows the Pd canvas to be drawn and manipulated using the HTML5 API. Since the HTML5 API is a widely documented and used, there is an enormous pre-existing knowledge and developer base for it. Furthermore, the code to display Pd patches in HTML5 will work in any modern GUI toolkit that has a webview widget.

4.1 Pd Canvas as SVG

Purr-Data implements a Pd canvas as an SVG document. SVG was chosen because it is a mature, widely-used 2D API. Unlike HTML5 canvas, larger canvas sizes have little to no performance impact on the responsiveness of the graphics. Since Pd users sometimes employ unusually large logical canvas sizes, this responsiveness makes SVG a better choice for drawing a Pd canvas than the HTML5 canvas API.

4.2 Leveraging the SVG Spec to Improve Pd Data Structures

Purr Data includes a large number of improvements to data structure visualization. In order to achieve this, a small subset of the SVG specification was used.

Inheriting from a pre-existing standards-based 2D API has several advantages over an ad-hoc approach. First, if implemented consistently, the extant documentation can be used to test and teach the system. Second, it is not necessary to immediately understand all the design choices of the entire specification in order to implement parts of it. Since those parts have been used and tested in a variety of mature applications, it makes it easier to avoid design mistakes that might otherwise be likely for someone who isn't a 2D graphics expert. Finally, there is less risk of a standards-based API becoming abandoned than a more esoteric API.

To improve data structure visualizations, several [draw] commands were added which map to the basic shape/object types in SVG: there is [draw circle], [draw ellipse], [draw rect], [draw line], [draw polyline], [draw polygon], [draw path], [draw image], and [draw group]. [draw text] has not been implemented yet. Each has

a number of methods which map directly to SVG graphical attributes. Methods were also added for Document Object Model (DOM) events to trigger notifications to the outlet of each object. Unfortunately, Pd has no easy way to put key/value pairs in messages, so the outlet gives a message with positional arguments.

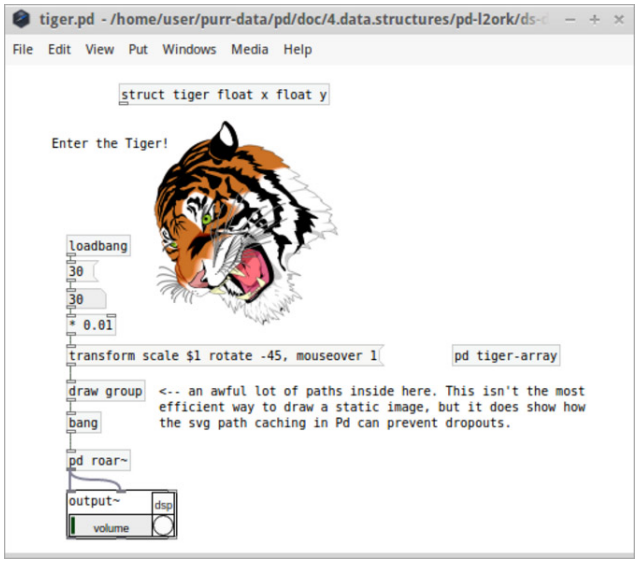

Figure 2: Purr-Data patch with a complex interactive SVG data structure.

The screenshot in Fig.1 shows the famous "SVG tiger" drawn from a few hundred paths found inside the [draw group] object. Even though the drawing is complex, Purr-Data caches the bounding box for the tiger object to prevent the hit-testing from causing dropouts. One can mouse over the tiger and trigger real-time audio synthesis.

5 Observed Impact

Pd-L2Ork has seen international adoption among other laptop ensembles, in curricula, and a growing number of outreach initiatives, including Maker camps, workshops, gifted programs. In 2010, L2Ork has formed a long-term partnership with a regional Boys & Girls Club of Southwestern Virginia where a number of initiatives of this kind have taken place. It has also helped start over half-dozen other similar initiatives across North and South Americas. The partnership with the regional Boys & Girls Club also served as the foundation for the initial exploratory study of the K12 module. Pd-L2Ork has been instrumental in ensuring L2Ork's recognition. In 2014 L2Ork was named as one of the top eight research projects at Virginia Tech [1] and in 2015 as one of the top six national transdisciplinary exemplars among research institutions [2]. With the prospect of cross-platform support, Pd-L2Ork is increasingly seen by some as Pd-extended's spiritual successor.
Since 2012, Pd-L2Ork is also being used extensively for teaching computer music courses at the Computer Music Research Group of the Johannes Gutenberg University (JGU) in Germany. The main reasons for switching from the vanilla and extended flavors of Pd initially were the GUI improvements (especially the ability to configure GUI objects and graph-on-parent patches in a graphical way) and the infinite undo capability, which make Pd-L2Ork much easier to use for beginners. On the other hand, one obstacle with Pd-L2Ork was that it required Linux, which hampered adoption by students (who, at least at the JGU, are often using Mac and Windows systems as their daily platform). But this is about to change now with the advent of Purr-Data.

5.1 Availability

Because of Pd-L2Ork's addons and its comprehensive set of bundled externals, the software has a lot of dependencies and a fairly complicated (and time-consuming) build process. So, while the software can be built straight from the source (which is best done using the included tar_em_up.sh build script located in the l2ork_addons subfolder), it is much easier to use one of the available binary packages:

- Virginia Tech's official Pd-L2Ork packages are available at http://l2ork.music.vt.edu/main/make-your-own-l2ork/software/, and
- Jonathan Wilkes' Purr-Data packages can be found at https://git.purrdata.net/jwilkes/purr-data-binaries/tree/master.

In addition, at the JGU we have created a Debian source package which can be used to build packages for all recent Ubuntu releases on both 32 and 64 bit architectures, along with a Makefile which fully automates the build process and also makes it easy to roll your own packages from current git sources. This build system is available in the debuild subfolder of the Pd-L2Ork and Purr-Data git repositories. Corresponding binary packages can be found on Launchpad. As of summer 2016,

these are also linked to from the Pd-L2Ork website as the default Linux packages. In a similar vein, we also support Arch Linux and its derivatives by means of corresponding package builds in the Arch User Repositories, as well as a binary package repository hosted on Bitbucket. More information about these, including pointers to the Ubuntu PPAs, binary Pacman repositories and bug trackers for reporting packaging issues, can be found at http://l2orkubuntu.bitbucket.org/ and http://l2orkaur.bitbucket.org/, respectively. In the future, we also plan to improve support for recent OSX versions by creating a similar MacPorts package.

The above repositories also contain Pd-L2Ork versions of JGU's Faust and Pure externals that allow for running signal processors written in Grame's Faust and JGU's Pure programming languages [14] [13]. These enable you to program your own Pd externals in a high-level, functional programming language. Faust is tailored for creating audio effect and instrument plug-ins, while Pure is a general purpose language geared more towards advanced symbolic processing of control messages. Both are well-integrated in the Pd environment and support an interactive (live-coding) development style, since Faust and Pure modules can be reloaded dynamically while the patch keeps running. Please note that these extensions require an installation of the Pure runtime environment, which is readily available through a separate Ubuntu PPA and the Arch User Repositories; detailed instructions can be found under the links provided above.

6 Future Work

Following the stable release of the cross-platform version tentatively scheduled for the winter 2016 (Purr-Data is currently in beta), Pd-L2Ork's roadmap includes continued improvement of program's usability, consolidation of externals and libraries, and the pursuit of the remaining bugs. In addition, the team would like to work on designing libPd-L2Ork and streamlining the Pd-L2Ork to libPd-L2Ork pipeline, as well as continue to develop frameworks for the embedded scenarios and recently established thrust areas, such as RPi and robotics.

With the imminent expansion onto other platforms, Pd-L2Ork's key challenge is ensuring sustainable growth. This can be achieved through fostering greater community participation in its development and maintenance, as well as in securing necessary resources. If interested in contributing to the project regardless of the aspirational capacity please do not hesitate to contact us.

7 Acknowledgments

The authors would like to thank the original Pd author Miller Puckette, numerous community members who have complemented the Pd ecosystem with their own creativity and contributions, including Hans Christoph Steiner and Mathieu Bouchard. We would also like to thank the L2Ork sponsors and stakeholders without whose support Pd-L2Ork would have never been possible nor sustainable.

References

[1] 8 Awesome Research Projects At Virginia Tech, One Of The Top R&D Institutions In The U.S. http://dcist.com/2014/11/8_awesome_research_projects_at_virg.php.

[2] a2ru Selects Transdisciplinary Exemplars for the 2015 a2ru National Conference. http://a2ru.org/knowledgebase/a2ru-selects-transdisciplinary-exemplars-selected-for-the-2015-a2ru-national-conference/.

[3] Cloud | Ivica Ico Bukvic. http://ico.bukvic.net/main/cloud/.

[4] [PD-announce] MacOSX installers for pd 0.36 and pd 0.36 extended (CVS).

[5] pd forks WAS : Keyboard shortcuts for "nudge", "done editing". http://permalink.gmane.org/gmane.comp.multimedia.puredata.general/79646.

[6] TkPath. http://tclbitprint.sourceforge.net/.

[7] Unity - Game Engine. https://unity3d.com.

[8] P. Brinkmann, P. Kirn, R. Lawler, C. McCormick, M. Roth, and H.-C. Steiner. Embedding pure data with libpd. In *Proceedings of the Pure Data Convention*, volume 291. Citeseer, 2011.

[9] I. Bukvic, T. Martin, E. Standley, and M. Matthews. Introducing L2ork: Linux Laptop Orchestra. In *Interfaces*, pages 170–173, 2010.

[10] I. I. Bukvic. Pd-L2ork Raspberry Pi Toolkit as a Comprehensive Arduino Alternative in K-12 and Production Scenarios. In *NIME*, pages 163–166, 2014.

[11] I. I. Bukvic, L. Baum, B. Layman, and K. Woodard. Granular Learning Objects for Instrument Design and Collaborative Performance in K-12 Education. In *New Interfaces for Music Expression*, pages 344–346, Ann Arbor, Michigan, 2012.

[12] P. Davis and T. Hohn. Jack audio connection kit. In *Proc. Linux Audio Conference, LAC*, volume 3, pages 245–256, 2003.

[13] A. Gräf. Signal Processing in the Pure Programming Language. In *Proceedings of the 7th International Linux Audio Conference*, pages 137–144, Parma, 2009. Casa della Musica.

[14] A. Gräf. Pd-faust: An integrated environment for running Faust objects in Pd. In *Proceedings of the 10th International Linux Audio Conference*, pages 101–109, Stanford University, California, US, 2012. CCRMA.

[15] D. Iglesia. MobMuPlat (iOS application). *Iglesia Intermedia*, 2013.

[16] K. Jolly. Usage of pd in spore and darkspore. In *PureData Convention*, 2011.

[17] J. Kincaid. RjDj Generates An Awesome, Trippy Soundtrack For Your Life. http://social.techcrunch.com/2008/10/13/rjdj-generates-an-awesome-trippy-soundtrack-for-your-life/.

[18] C. McCormick, K. Muddu, and A. Rousseau. PdDroidParty-Pure Data patches on Android devices. *Retrieved January*, 21, 2014.

[19] M. Puckette. Pure data: another integrated computer music environment. *In Proceedings, International Computer Music Conference*, pages 37–41, 1996.

[20] M. Puckette. Max at seventeen. *Computer Music Journal*, 26(4):31–43, 2002.

[21] B. Sawyer, J. Forsyth, T. O'Connor, B. Bortz, T. Finn, L. Baum, I. I. Bukvic, B. Knapp, and D. Webster. Form, function and performances in a musical instrument MAKErs camp. In *Proceeding of the 44th ACM technical symposium on Computer science education*, SIGCSE '13, pages 669–674, New York, NY, USA, 2013. ACM.

[22] B. B. Welch. *Practical programming in Tcl and Tk*, volume 3. Prentice Hall Upper Saddle River, 1995.