Pd-Multitouch : Interacting with pure data patches in multitouch mode

Jean-Yves GRATIUS, *jyg@gumo.fr*

https://github.com/jyg/multitouch

https://github.com/jyg/mob

Summary

This article presents the **pd-multitouch** and **pd-mob** projects, sets of abstractions allowing, thanks to current external libraries, to use Pure Data in multitouch mode, on fixed or mobile platform.

Key words

Pure Data. Multitouch. Hid. Linux. MobMuPlat.

Introduction

Pure Data is a graphical programming software for real-time music and multimedia creation, belonging to the patcher[1] category

Its communication with the external environment can be done through different interfaces, digital audio, midi, network, video, keyboard / mouse, graphics tablets, joysticks, or any other input / output devices[2]. On mobile devices, it is also possible to use the integrated physical sensors (accelerometer, gyroscope, GPS, compass ...).

Such devices have popularized the use of multitouch touch screens since 2010, beyond museum installations.

The interest and relevance of multitouch interfaces for music computing applications is illustrated in particular by the number of mobile music applications that have since emerged, taking full advantage of the possibilities offered by the touch screens of phones, tablets and other mobile devices.

What about Pure Data and Multitouch in 2020 ?

First, we will review the existing solutions that allow direct or indirect interaction with Pure Data via a multitouch interface, on fixed or mobile platforms. We'll also discuss integrated solutions that use third-party music software. Then, we will present the pd-multitouch project by explaining the principle of its operation, as well as the hardware and software prerequisites to use it. In addition, the **pd-mob** project, which offers enhanced interface modules that can be used in patches and that are multitouch compatible, will also be introduced. This project helps exporting patches to the MobMuPlat mobile platform. Finally, we will discuss the future prospects for these projects, such as supporting other O.S. than Linux.

1.1 Multitouch & PureData via external interfaces

a) Hardware multitouch controller

Before homogeneous hardware solutions appeared, interaction with Pure Data in multitouch mode passed through specific external interfaces, communicating with the patch via a network protocol, for example OSC.

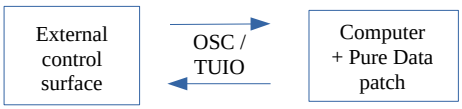


Fig. 1 - multitouch via OSC

For example, the Lemur, a hardware interface specifically dedicated to musical applications, was marketed in 2004 by the company JazzMutant, as a multitouch modular controller with a screen. Further, it upgraded in software form (application for iOS or Android).

Another protocol, Tuio, was introduced during the development of RéacTable[3] synthesizer and RéacTiVision[4] module. It is a layer added above the OSC protocol.

This protocol enabled the development of multitouch solutions using consumer sensors (Wiimote, infrared webcams, Kinect, etc.) associated with Pure Data by means of a server program generating Tuio events[5].

1 Puckette M."The Patcher", Proceedings of the International Computer Music Conference1988, p. 420-429, Cologne, Allemagne, 1988.

2 For example, see the malinette.info

3 http://reactable.com/

4 http://reactivision.sourceforge.net/

5 See the tuio.org site which presents different software bridges between hardware and client software.

b) Mobile device interface

With the rise of smartphones and tablets from the 2010s, a number of mobile applications have appeared, exploiting the multitouch functionalities of these devices to offer tactile interfaces allowing to communicate with a target program on another computer (Pure Data or any other music production software). We can mention one of them, TouchOSC[6].

c) Interface from web browser

Multitouch screens then appeared on some laptops, and O.S. like Windows, Linux or Chrome have started to support applications that take advantage of those features (including some web browsers).

Since 2014, with the Open-Stage-Control project, developed by Jean-Emmanuel Doucet[7], it's possible to load a GUI interface from a web browser connected to a local server which, in turn, communicates with the client program via OSC. Multitouch is supported on mobile platforms, as well as on Windows and Linux OS, as long as the used browser supports multitouch.

This solution, like the previous ones, is based on a client-server architecture. In the case of use with Pure Data, browser-based interface will communicate with the patch, via a server. The interface is modular and editable, the OSC communication is bidirectional, and it is possible to synchronize several instances on the same network, which is interesting in the context of a configuration with several interfaces / users. The project is very active and the developer very responsive.

1.2 Multitouch & PureData

Versions of Pure Data for mobile devices have existed for several years, naturally allowing access to the multitouch functionality of these devices, but the patches targeted for those platforms have first to be written and developed on a fixed platform, with the standard version of Pure Data and a set of tools / externals needed to adapt them. Those versions include MobMuPlat, PdParty, PdDroidParty, Pof, Ofelia, as well as various mobile applications derived from the libpd library.

Some of these applications are capable of directly rendering the interface as it appears in the original patch inside Pure Data (e.g. *slider*, *selector*, *number*, *toggle* objects and others…).

Others require to completely redefine the interface, either with an external editor (ex : MobMuPlatEditor), or inside a graphic rendering window and a specific library (Pof, Gem, Ofelia). In this case, even if the library supports the management of multitouch events, even if the computer used has a multitouch screen and even if the OS manages multitouch events (case of certain Linux or Windows distributions), multitouch interaction mode is not necessarily functional[8].

1.3 Multitouch & other music software

There are multimedia creation-oriented software that manage multitouch natively. These include, without being exhaustive, Usine Hollyhock[9], Bitwig Studio[10], Sonar[11], Kivy[12] and Processing[13] (the latter two being more to be considered as multimedia programming environments).

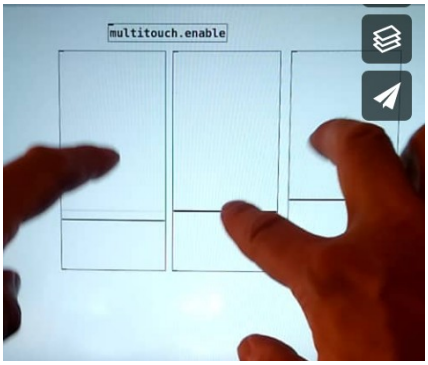
Some audio plugins are also multitouch, but we couldnot check them.[14].

2.1 Pd-multitouch : direct multitouch support in Pd

The solution proposed here consists in making *multitouch - compatible* any existing Pure Data patch, by adding the [multitouch] object to it. [multitouch] abstraction listens to touch events generated by the touch screen interface and converts them into messages which, inside the Pure-Data patch, interact with all graphical interface elements (GUI) it contains.

6 https://hexler.net/products/touchosc
7 https://openstagecontrol.ammd.net/

8 as of today (March 2020)
9 www.brainmodular.com
10 www.bitwig.com
11 www.cakewalk.com
12 kivy.org Kivy is a free and open source library for Python, useful for creating multitouch touch applications with a natural user interface (wikipedia source).
13 https://processing.org/
14 We refer to some threads on the net, for example : https://www.kvraudio.com/forum/viewtopic.php?t=454288



a) Operation

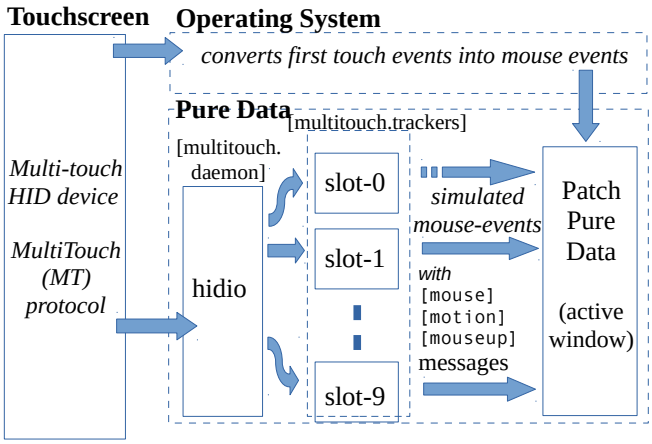


Fig. 2 - architecture of pd-multitouch module

It was not possible to change the code of the tcl / tk GUI to add support for multitouch events. We therefore opted for the direct interception, from within Pure Data, of multitouch events generated by the touch device thanks to the external hidio[15]. These events are then converted by the [multitouch] abstraction into series of virtual "mouse" events. For each touch contact established on the screen, – assigned to a specific “slot” whenever it appears –, a burst of three messages is sent to the active patch window : `mouse` (mouse click), `motion` (mouse movement with offset corresponding to finger movement), then mouseup (mouse release).

This process is repeated for each iteration, i.e. each time the device returns a new multitouch event frame for all of the slots[16].

The handling of events related to slot-0 is a little different, as the O.S. is likely to automatically generate mouse events for the first contact with the touch screen. Thus, [multitouch] shouldn’t produce duplicate events for this slot.

15 This is a patched version of hidio external, available at the following address https://github.com/jyg/hidio.

16 According to MT-protocol, (https://www.kernel.org/doc/Documentation/input/multi-touch-protocol.txt), each slot number is assigned to a single touch contact, from the appearance to the disappearance of the latter.

b) Configuration

- O.S. : Linux
- PureData > 0.50
- externals : hidio, iemguts, hcs/screensize

The system on which we developed the pd-multitouch library was a Lenovo Thinkpad T440s laptop with multitouch screen and Linux operating system (18.04.4 LTS).

Adapting to a Windows system would require a Windows version of the *hidio* external. So far, our attempts to intercept events generated by the multitouch device before they are captured by the active window have failed.

It is important to deactivate the O.S. multitouch gestures (*pinch*, *3-finger tap*, etc.) so that it does not send inappropriate messages (scrolling, zooming, etc.) to the Pure Data patch. On the other hand, the emulation of the mouse in monotouch mode is necessary for the calibration and the proper functioning of [multitouch][17].

c) Installation

- Check that required externals are installed : iemguts, hidio (https://github.com/jyg/hidio).

- Check that hidio is working properly when you open the hidio-help patch[18].

- Open *multitouch-help.pd* patch.

- In the multitouch.settings.pd window that appears (fig. 3), follow the instructions for the three-step configuration. You’ll have to identify the touchscreen device in the list of system hid devices, to check the screen resolutions, and finally to adjust, if necessary, the display orientation settings (normal / inverted).

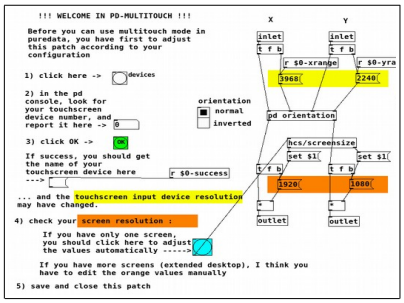


Fig. 3 – multitouch.settings window

17 [multitouch] compares hidio messages and mouse events received by the patch, and calculates the offsets in x and y, which depend on the position of the patch window on the screen.

18 If that doesn't work, it may be necessary to recompile the external. Open a terminal in the hidio folder, then type **./configure** and then **make**.

d) Project limitations

Native Pure Data GUI objects such as *sliders* or *selectors* work well in multitouch mode. This is less the case for *bng*, *number*, *number2* objects and especially for *toggle* and message boxes. Indeed, the bursts of events [`mouse` / `mouseup`] generated at each movement, even the smallest, of contact points, lead to untimely re-triggering of message boxes and *toggles*. Replace *toggles*, when possible, with 2-state *selectors*. *Bng* objects are more robust, due to the *interruption time* parameter they have.

In **mono-touch mode** or with the mouse, after clicking on a *slider*, you can control this *slider* even if the pointer moves outside its interaction area. In **multitouch mode**, *sliders* are operated as long as the contact on the screen remains within their interaction area, but as soon as the contact leaves this area, the association with the *slider* is lost. There is no exclusive capture of a touch by a specific slider.

The explanation is that bursts of mouse events (`mouse` / `mouseup`) are generated for all current contacts, and therefore the inclusion test is repeated, for each iteration, by all Pure Data native GUI objects[19].

That is why *number* and *number2* objects cannot be used in multitouch mode, because their interaction surface is too small. Changing graphs (arrays) is also possible in multitouch mode, but still requires some skill, as certain points in the table may be skipped during the finger movment…

Another limitation of the project is the impossibility to use several patches simultaneously in multitouch mode. Only the active patch window multitouch.

2.2 The pd-mob project

Complementary but independent of [multitouch], the [mob] collection of abstractions has two purposes :

a) Providing elaborate GUI objects

… in the form of abstractions, which allow new types of interactions in a patch: rotary buttons (*knob*), rotary dials (*jog*), popup menu, horizontal and vertical faders (*hfader* / *vfader*), xy pad (*xy_slider*), xy multitouch pad (*xy_multi*), graphic display area (*lcd*). In multitouch mode, most of these objects can "lock" on a slot to capture all its messages, which means that the association with a touch contact is maintained even if the contact moves outside the object's interaction area.

19 GUI objects that are released in pd-mob project work differently and allow exclusive capture of multitouch events.

.

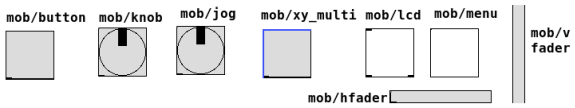


Fig. 4 – quelques objets GUI élaborés de la bibliothèque [mob]

b) Simplifying MobMuPlat export

… of Pure Data patches, with identical layout for the interface elements (fig. 5). You no longer need to use the third-party MobMuPlatEditor program to redesign the interface from scratch.

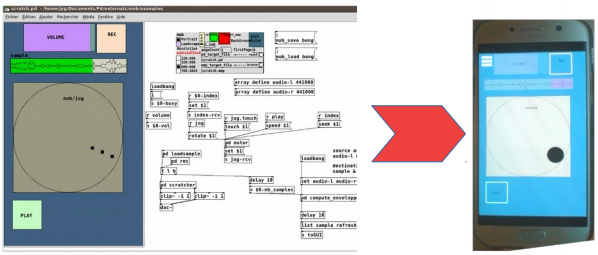


Fig. 5 – Fast export to MobMuPlat

This collection of abstractions requires a few externals to work : mrpeach / binfile, as well as iemguts and iemlib libraries.

2.3 Going further – To Do list

Regarding the pd-multitouch project :

- List the multitouch screen models compatible with the pd-multitouch project ;
- Consider a version for Windows and in particular MS-surface pro tablets…

Regarding the pd-mob project :

- Continue the development of other GUI objects and their export to MobMuPlat; ;
- Extend the interface export function to systems other than MobMuPlat, such as Pof, Open Stage Control.

Références

Documentation on Multitouch protocol (MT)

https://forums.opensuse.org/showthread.php/506695-Multitouch-on-ELAN-touchscreen

https://www.kernel.org/doc/Documentation/input/multi-touch-protocol.txt

Video demonstration

https://vimeo.com/292789