Channel pressure to poly-AT conversion scripts
User's Manual (sort of)

Gabriel Pasquali (gabpas64 _at_ hotmail.com)

This document should be included in the docs subfolder of the midipressure_to_polyAT folder.

The midipressure_to_polyAT folder contains three simple JSFZ scripts: midipressuretopoly, midipressuretopolylag and midipressuretopolylagdel

Installation

In a host like Cantabile (or others) the script must be loaded into ReaJS (which is part of the ReaPlugs free package: https://www.reaper.fm/reaplugs/) and the MIDI output from your controller has to pass through ReaJS before going to the VST plugin, which in turn should be able to use poly-AT.

To make ReaJS *see* the scripts, the files midipressuretopoly, midipressuretopolylag and midipressuretopolylagdel must be placed into the midi folder of ReaJS (for instance path_to_your_installed_vst_plugins\ReaPlugs\JS\Effects\midi).
Then use the "load" button in ReaJS to load the scripts (e.g. Load->midi->midipressuretopoly).
WARNING: Do not place other files (this doc, for instance) in the same folder as the script (or in any folder where ReaJS looks for scripts, otherwise ReaJS will try to interpret it as a script, which of course it is not ;-)).

In Reaper, the installation is a bit different (many thanks to cpaolo at Cantabile Forum for checking that it works):

- select menu "Options->Show REAPER resource path in explorer/finder";
- open the Effects folder in explorer/finder and copy the file "midipressuretopolylag" into the MIDI folder;
- refresh the plugin browser in Reaper

midipressuretopoly

midipressuretopoly is a little script which converts "Channel Pressure" data (i.e. monophonic aftertouch, which applies to all notes played) to Polyphonic After Touch messages (which contain info about which note they must be applied to). The note to which aftertouch is applied is the last note played. The script should receive at its input MIDI messages from a controller able to generate channel (monophonic) pressure messages.The output of the script should be connected to the MIDI input of a device (e.g. VST plugin) able to handle polyphonic aftertouch messages.

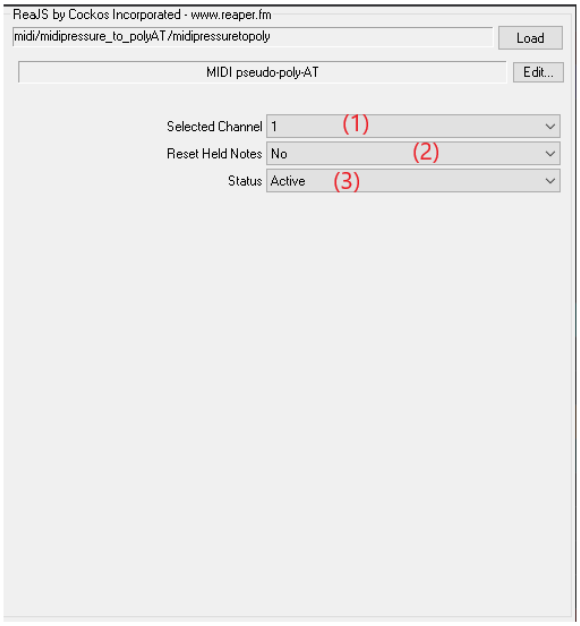
This idea of pseudo-poly-AT based on the last note played has already been implemented in the past (e.g. Memory Moon's ME-80, Cherry Audio's GX-80 VST plugin). It's the sort of Columbus' egg one would have liked to think about first...but I am glad anyway that someone did think about it.

In short, that's what the script does:

- if a NOTE ON is received, the note is "latched", a poly-AT=0 is sent for that note (as a reset), the NOTE ON message is forwarded to the midi output (i.e. to your instrument VST); before all that, however, if the “Reset Held Notes” option has been selected, the poly-AT is set to zero for the previously latched note which as a consequences loses the modulation;
- if a NOTE OFF is received, a poly-AT value of zero 0 is sent for that note; moreover, if the released note is the one latched for poly-AT, we reset the latched value (it is set to -1); the NOTE OFF is also forwarded to the midi output;
- if a CHANNEL PRESSURE message is received and the latched note is not minus one (i.e. it is a legit note value), a poly-AT message is sent for that note (with the value of the pressure taken from the received message) instead of the channel pressure message (which is not transmitted); this last point is of course, THE point of all of this;
- all other incoming midi messages are forwarded as they are to the output.

There are very few controls in the midipressuretopoly window (see picture below):

(1) Select the channel (1-16) on which the plugin listens for incoming notes and channel pressure messages;
(2) If “Reset held notes” is on “Yes”, a null poly-AT message will be sent for the previously latched note when a new key is pressed (i.e. its poly-AT will be reset, even when the previous note is still pressed). Otherwise, the previous note will maintain its last poly-AT level;
(3) If status is “Active”, incoming MIDI messages are processed as explained above; otherwise, the script acts as a MIDI THRU.



midipressuretopolylagdel and midipressuretopolylag

These scripts are improved versions with respect to "midipressuretopoly". They are identical, except for the “Release delay” parameter, which is missing in midipressuretopolylag.

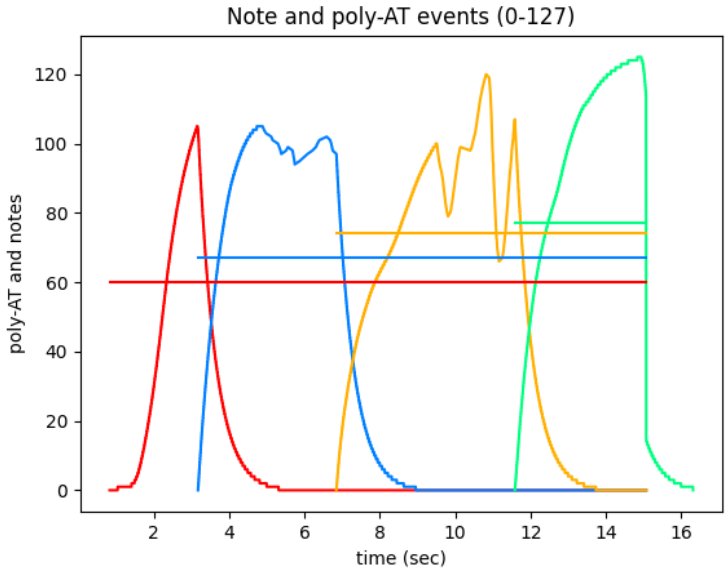
PRINCIPLE OF OPERATION:

- If a NOTE ON is received, the note is "latched", a poly-AT=0 is sent for that note (as a reset), the NOTE ON message is forwarded to the midi output (i.e. to your instrument VST); before all that, however, if the “Reset Held Notes” option has been selected, **the poly-AT for the previously latched note goes to zero either exponentially or linearly (according to the Exponential/Linear selector).** For exponential transitions, the time constant is set by the "Release" slider and the transition lasts about 4-5 times the set Release" value (*). For linear transitions, the total transition time is set by the Release slider. Therefore, at variance with midipressuretopoly, the poly-AT for the previously latched note is not set to zero immediately; even more important, the poly-AT value of the new note is not set immediately to the current value of CHANNEL PRESSURE. Instead, it goes up either exponentially or linearly (according to the Exponential/Linear selector). For exponential transitions, the time constant is set by the "Attack" slider and the transition lasts about 4-5 times the set value. For linear transitions, the total transition time is set by the "Attack" slider.

- If a NOTE OFF is received, a poly-AT=0 is sent for that note, unless the released note is the one latched for poly-AT: in the latter case, we reset the latched value (it is set to -1); the NOTE OFF is also forwarded to the midi output and **the poly-AT of the released note goes to zero according to the Release time value.**

- If a CHANNEL PRESSURE message is received and the latched note is not minus one (i.e. it is a legit note value), a poly-AT message is sent for that note (with the value of the pressure taken from the received message) instead of the channel pressure message (which is not transmitted); this last point is of course, THE point of all of this. However, if the note is in the “Attack” phase, the new pressure value will not be reached immediately: the attack phase will go on until the current pressure value is reached.

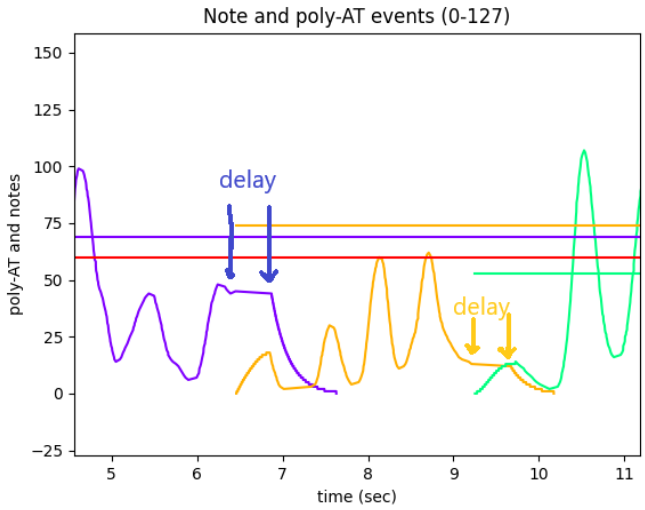
This behavior can be seen in the picture below: the horizontal lines represent the notes which have been played (a different color for each note, which is drawn at an ordinate corresponding to the MIDI value of the note itself). The curves (which follow the same color scheme) represent the poly-AT messages generated by the script. For instance, at about time=3 seconds, a new note is pressed (in blue) so that the poly-AT value of the first note (in red) starts to decrease (release phase) while the poly-AT of the new note increases (attack phase).



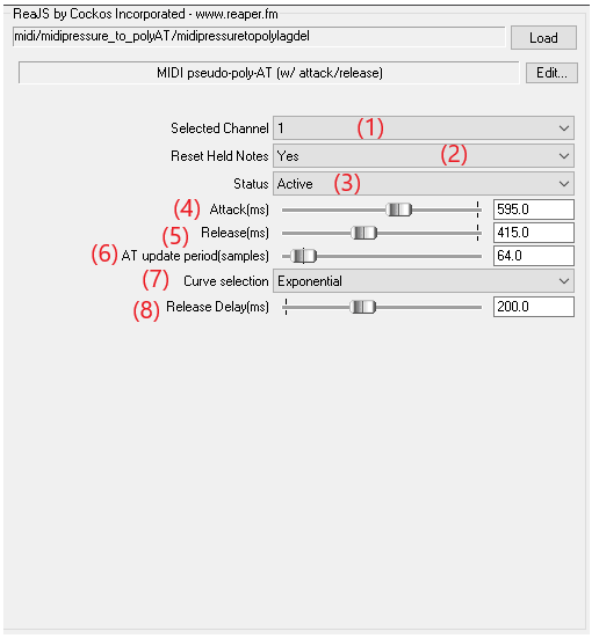
(*) The exponential transition is like that of a RC low pass filter (one pole). The Attack and Release values basically set the RC value of the simulated RC circuit.

Only in midipressuretopolylagdel:

This version of the script introduces a delay before the start of the release phase, so that the poly-AT value of the old note is kept (for a time interval equal to the delay) while the poly-AT of the new note increases. This feature allows for smoother transitions as shown in the picture.



Here is the control window of midipressuretopolylagdel:



(1) Select the channel (1-16) on which the plugin listens for incoming notes and channel pressure messages;

(2) If “Reset held notes” is on “Yes”, a null poly-AT message will be sent for the previously latched note when a new key is pressed (i.e. its poly-AT will be reset, even when the previous note is still pressed). Otherwise, the previous note will maintain its last poly-AT level;

(3) If status is “Active”, incoming MIDI messages are processed as explained above; otherwise, the script acts as a MIDI THRU;

(4) The “Attack” slider sets either the time constant (exponential attack) or the total transition time (linear attack) in milliseconds;

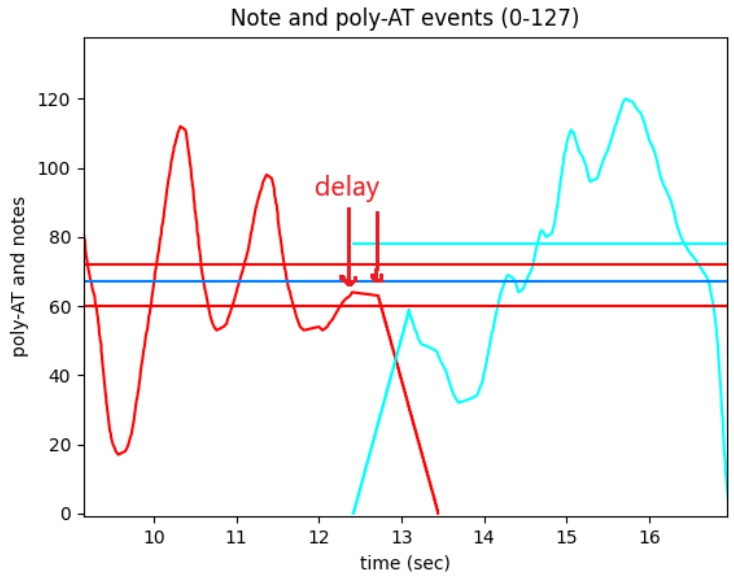
(5) The “Release” slider sets either the time constant (exponential release) or the total transition time (linear release) in milliseconds;

(6) The “AT update” slider sets the time interval (in samples) between poly-AT updates during the attack and release phases; the smaller the value, the greater the number of messages sent for a given attack or release time.

(7) Selects between exponential and linear attack/release

(8) A Release delay greater than zero introduces a delay before the onset of the release phases, thus producing smoother transitions between the poly-AT values of consecutive notes (only in midipressuretopolylagdel).

Finally, an example of linear transitions with release delay:



APPENDIX

The midipressure_to_polyAT folder also contains a parse_midi folder. The folder contains a python script used to generate the pictures with the notes and poly-AT values.
The script needs the following packages: mido, numpy, matplotlib. First record the output of the JSFX script (e.g. midipressuretopolylagdel) in a midifile. Then run the python script as follows:

```
python parse_midi.py midifile
```