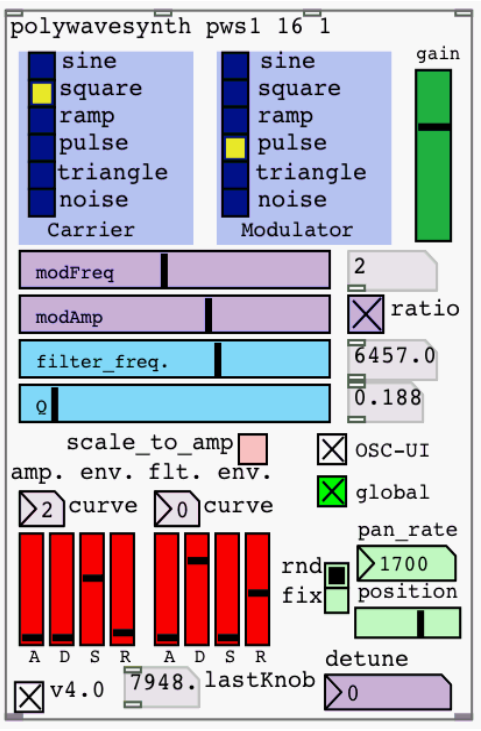
[polywavesynth]

a polyphonic synthesizer for Pure Data

Version 5.0



[polywavesynth] is a ready-to-use, CPU-cycle-stingy, OSC-addressable polyphonic "wave" synthesizer for Pd. The tag "wave" is used for want of a better (short) descriptor for the somewhat unusual mix of synthesis technologies employed, which is table-driven subtractive synthesis with a dash of FM.

[polywavesynth] is a front end/UI which directs control messages to multiple instances of [polywavevoice~], which is the synthesizer engine for each voice. It is theoretically possible to use a [polywavevoice~] on its own, but it would just be a mono synth with a somewhat cumbersome method of parameter addressing. Its real power comes from being invoked and managed by Frank Barknecht's [polypoly] object (see below), as is done by [polywavesynth].

You can hear some music I've made with this synthesizer here.

Download and Setup

NOTE: if you are already a user of an earlier version of [polywavesynth], be aware that there are slight differences between version 3 (and later) presets and those of version 2. See here for more information.

Download the polywavesynth archive here.

If you are using Pd-extended of a recent vintage, you should be ready to go. Specifically, **Pd-0.40.3-extended or higher** includes Martin Peach's OSC library and the [import] object, both of which are required. Download it here -- look for the correct version for your platform.

For use with vanilla Pd, you need to do a few extra things:

- extract "mrpeach osc" from Pd-extended and put it somewhere in your Pd path (such as "extras").

- do the same for "cyclone/svf~" (state variable filter).
- make an empty "import.pd" object and save it to the "polygrainsynth" folder. (This will avoid unimportant but annoying error messages.)

[polywavesynth]'s UI appearance and patch layout work best with the cross-platform-compatible font used in the latest versions of Pd-extended -- there may be a little layout awkwardness with vanilla Pd's font on the various platforms (though it will still be useable).

[polywavesynth] Operation

It should be pretty easy to get a sound out of [polywavesynth]. See [polywavesynth_example] (which includes presets), or even just the help patch, to quickly make some noise. Most controls should be familiar to those with subtractive synthesis and/or FM synthesis experience.

Anything in [polywavesynth] displayed in a "number2" box (the ones with ">" on their left edge) is an adjustable parameter, which can be saved and recovered using [sssad]. Normal number boxes are display-only. Most controls should be fairly obvious in function, with a few notes:

- the toggle box in the lower left hand corner **enables** the instance; when it is off, incoming note-ons are ignored. In a multi-synthesizer setup, this allows quick switch-in, switch-out of modules.
- the '**filter_freq.**' slider controls the low-pass cutoff frequency for the enveloped filter; the '**Q**' slider controls filter resonance, with the added feature that when 'Q' is set to zero, enveloped low-pass filtering is switched completely off.
- '**modFreq**' is interpreted differently, depending on whether "**ratio**" is on (it selects from a list of fixed ratios to the carrier frequency) or off (it is interpreted as an actual frequency).
- the '**detune**' value is in cents. It is particularly useful when ganging up multiple [polywavesynth]s for a very fat sound. It can also be used for transposition.
- random per-note **panning** can be switched on with the "rnd" setting of the "**rnd/fix**" radio button. Each new note is given a random position, and the new voice moves toward that position at a rate in milliseconds given by the "**pan_rate**" number box. The "fix" setting corresponds to a "fixed" position, set by the position slider.
- the '**lastKnob**' display is ganged up to all the envelope sliders' outputs, and shows whichever one changed last (a space saving compromise).
- the '**OSC-UI**' toggle box enables live UI response to incoming OSC data. This provides valuable graphic feedback when using OSC to modulate [polywavesynth] parameters, but it can also consume a great deal of available CPU bandwidth on slower machines.
- the '**global**' toggle box switches how most UI and OSC controls are sent to synthesizer voice parameters:
    - global-on: parameters affect all voices of a given [polywavesynth] module in real-time.
    - global-off: parameter changes only affect each newly-attacked voice.

  A few [polywavesynth] parameters cannot be switched:

  ### always global

    - pitch bend
    - all notes off

never global

- (all envelope controls)
- frequency
- amplitude

The control mode for all other parameters is determined by the setting of the 'global' toggle box. The 'global-on' setting is especially useful (and fun!) for responding to quickly-changing OSC data or for live playing of the UI. The 'global-off' setting allows a more subtle and clean "phase-in" of parameter changes.

The states of the 'global' or 'OSC-UI' toggles are **not** saved as [sssad] parameters.

N.B.: [polywavesynth] implements its wave tables in a singleton. Therefore, you can have multiple [polywavesynth]s open at once, and only one set of tables will be allocated for all of them, with no conflict. However, if you close the first-opened patch that contains a [polywavesynth], the single instance of the tables will disappear along with it. This is an unavoidable side-effect of using singletons with a [closebang]-less Pd. If this happens, just close all patches containing [polywavesynth], and re-open -- the singleton will re-establish.

Specifications

Inlets:

1. frequency of note in Hertz; initiates attack or release
2. amplitude of note, between 0 and 1 (>0 = attack, 0 = release)
3. external "mute" - note-ons are ignored when this is non-zero
4. OSC inlet (see OSC implementation, below)

Arguments:

1. synth name, used for OSC addressing and preset saving/loading (allows multiple polywavesynths to be used simultaneously yet controlled independently)
2. number of voices
3. voice stealing (0=off; 1=on)

Outlets:

- left: audio left channel
- right: audio right channel

OSC implementation tree:

- `/(synth name)`
  - `/controlmode --(value = 0: UI and OSC controls are per-voice; 1: controls are global)`
  - `/enable -------(value = 0: incoming note-ons are ignored; 1: all note-ons honored)`
  - `/gain ---------(value >= 0.0, though be careful with gain > 1.0)`
  - `/global`
    - `/pitchbend ----(value in cents)`
    - `/allnotesoff --(any value)`

- /note
  - /freq ---(value in Hertz, triggers or releases note, so should be sent last)
  - /amp ----(amplitude multiplier, 0.0-1.0; 0=release >0=attack)
- /osc
  - /carrier
    - /wave ----(0=sine 1=square 2=ramp 3=pulse 4=triangle 5=noise)
    - /detune --(value in cents)
  - /mod
    - /wave -------(0=sine 1=square 2=ramp 3=pulse 4=triangle 5=noise)
    - /amount -----(0.0 - 1.0, modulation index)
    - /freq -------(0.0 - 1.0)
    - /ratorfreq --(freq. mode switch 0=frequency 1=ratio)
  - /env
    - /atk ---(value in msecs.)
    - /dec ---(value in msecs.)
    - /sus ---(0-100, percent of full amplitude)
    - /rel ---(value in msecs.)
    - /exp ---(envelope curve "exponent" -- see multicurveadsr-help.pd)
- /filt
  - /freq --(value in Hertz)
  - /q -----(0=filter switched off -- reasonable values=between 0.1 and 1.0)
  - /env
    - /atk --------(value in msecs.)
    - /dec --------(value in msecs.)
    - /sus --------(0-100, percent of full amplitude)
    - /rel --------(value in msecs.)
    - /exp --------(envelope curve "exponent" -- see multicurveadsr-help.pd)
    - /fampscale --(0=filter opens fully; 1=scale filter opening to amplitude)
- /pan
  - /position --(0=full left; 1=full right)
  - /speed -----(speed of transition to new position, in msecs., >=2)
  - /mode ------(0=random; 1=fixed)
  - Made panning equal-power

[polywavevoice~] Architecture

At the heart of each [polywavevoice~] is a pair of table-driven waveform oscillators, in a carrier/modulator configuration. Both carrier and modulator can be switched between five simple waveforms (sine, square, pulse, sawtooth, triangle) or a random-filled "noise" table. Modulation may be switched between **rational** (modulator can be a harmonic of the carrier, like in most commercial FM synthesizers), and **free** (useful for inharmonic sounds, or at very low frequencies, vibrato.) Note that the "freq" value is interpreted differently for the two modes -- either as selecting from a list of fixed ratios, or as a literal frequency.

The output of this wave generator is run through an envelope-controlled, variable-frequency and Q-factor low-pass filter. The envelope opens the filter to the frequency indicated by the "filter_freq." slider, scaled by the amplitude of the note. This behavior can be defeated by de-selecting "scale_to_amp" near the filter envelope controls; in this mode, the filter will open to full range for all notes.

The output of the filter is in turn passed through an amplitude envelope. Both the filter envelope and amplitude envelope have variable curves, and ADSR controls.

Finally, the output of the amplitude envelope is split and panned in one of two ways: either 1) fixed, the l/r balance set by a number between 0 (full left) and 1 (full right), or 2) a random value between l and r is chosen just before note attack, with a transition time set by the pan rate variable.

[polywavevoice~] is fairly thrifty with CPU cycles. It uses lookup tables for envelope curves, and each voice switches itself off when its amplitude envelope completes, so only voices actively playing consume CPU. Also, the enveloped filter in each [polywavevoice~] switches itself off if its 'Q' value is set to zero. So, if you really need more voices, don't have CPU cycles to burn, and don't mind giving up per-note enveloped filtering, set 'Q' to zero.

Polyphony

[polywavevoice~], an individual voice instance for this synthesizer, is compatible with [polypoly], which is a powerful object combining the replication features of [nqpoly/nqpoly4] with the allocation and management capabilities of [poly]. [polypoly] can clone a compatible object 'n' times, where 'n' is limited only by available CPU cycles. Using [poly], it automatically assigns a voice instance on attack, tagging it by frequency so as to be able to mark it as available on subsequent release. This makes it very easy to instantly create a polyphonic synthesizer as is done in [polywavesynth]:

[polypoly $2 $3 polywavevoice~ $0]

The first argument is for 'n', or the number of voices desired, and is controlled by [polywavesynth]'s second argument. High numbers give rich polyphony, but can also use more CPU. Stick a CPU meter (like [loadometer] in [polywavesynth_example]) in your patch, and watch it to see what kind of bandwidth you can get away with under various musical "loads"; adjust 'n' accordingly in the creation argument for [polywavesynth]. Trying to suck more CPU cycles than exist will lead to "interesting music", so be conservative with 'n'. Anecdotally, on my MacBook Pro, sustaining 64 notes simultaneously uses about 75% CPU.

The next argument tells the internal [poly] whether to use voice stealing, and is controlled by [polywavesynth]'s third argument.

[polypoly] allows up to four more arbitrary arguments, which are passed through to the object being cloned. [polywavesynth] currently uses only one of them to pass the $0 variable, which serves as a unique prefix to communicate GLOBAL parameters to all [polywavevoice~] voices of this [polywavesynth] instance.

When any message is sent to '$0-GLOBAL-allOff' (OSC: /(synthname)/global/allnotesoff), all notes currently attacked are released and deallocated from the voice-managing [poly] object). This is useful for quashing stuck notes, which can be caused by incomplete or unbalanced note messages sent to [polywavesynth].

State Saving

[polywavesynth] supports [sssad] state-saving. See the [polywavesynth_example] object for how to do it.

Migrating v2 presets to v3

[sssad]-saved presets made with version 2 [polywavesynth] will be missing a couple of lines needed for version 3 (and later) presets. Simply add the following two lines to each of your version 2 presets, and they will be compatible with version 3 and up. Be sure to replace "wav1" with the name of your synth (you'll know what this is because all the lines in your [polywavesynth] patch files start with it -- it's also the first argument to [polywavesynth]).

wav1/fampscale 1;
wav1/enable 1;

v4 OSC changes

A couple of OSC nodes used to be spelled in camel case (/allNotesOff and /ratOrFreq). **All** OSC nodes are now completely lowercase: /allnotesoff, /ratorfreq -- sorry for any inconvenience, but this is better for the long run! If you didn't use these OSC nodes across these versions, don't worry about this at all.

Credits

I built on a great deal of pre-existing Pd work to make this synthesizer.

I owe big thanks to Tom Erbe and a couple of his students, who designed some wonderful synthesizer emulations back in 2005. One of these in particular, Patrick Sanan's Minimoog, inspired me to experiment with making my own Pd synth. I started with the waveform data from Patrick's synth. Since his comments lead me to believe he got the band-limited table-generating code from Tom, I named my modification of this lifted code [erbe_tables] in Tom's honor.

Chris McKormick's "s-abstractions" taught me how to do GOP and more generally, how to organize my thinking about patches. I learned how to use [sssad] state-saving from his examples, too.

Frank Barknecht designed [polypoly] and [sssad] and [singleton], among a multitude of other useful Pd objects. He makes magic happen in Pd.

(the creator(s) of svf~): this filter has a long lineage, so I'm not sure who to thank, but it sounds beautiful and functions superbly.

Areas for Improvement

I can only stand designing instruments for so long, and then I need to play them, so I sometimes take expedient shortcuts to get something functioning. As a result, this synthesizer could use improvement in some areas.

- I'm only using a small part of [svf~]. More of its functionality should be exposed.
- OSC and state-saving in [polywavesynth] are a bit "impedance-mismatched" right now, with some overlapping functionality. I suspect that there is better a way of integrating them, perhaps something more like Frank Barknecht's RRADical.
- An lfo or two internal to each [polywavevoice~] would be great...perhaps used during sustain for vibrato, tremolo or other per-note modulations.

Carpe diem, hackor! I am very slow about making changes, so don't wait for me! Make your own improvements/customizations if you have some ideas -- or just use [polywavesynth] as a template for something completely different.

Please **let me know** (via the "contact" link at the top of the page) if you find any bugs or bad behavior. Also, I'd be glad to hear if you got any good use from this synthesizer.

Revision History

- 2012-01-22 - v 5.0
  - Fixed glitch in quartic curve for multicurveadsr
  - Fixed bug in gain changer inside multicurveadsr
  - Generally cleaned up envelopes - no unwanted "snaps"
  - Added OSC parameter: /pan/mode
- 2009-12-09 - v 4.1
  - Changed all AD and R sliders on envelopes to log response, with bottom of 8
  - Made triangle wave band-limitied
- 2008-11-22 - v 4.0
  - Added "global" toggle, which enables live output of global-capable parameters to all voices.
  - Added "OSC-UI" toggle, which enables the UI to follow OSC input.
  - Fixed bug in "noise" wave table (only an eighth of it was initialized before!)
  - OSC nodes are all-lowercase now (e.g. /allNotesOff is now /allnotesoff).
- 2008-10-02 - v 3.1
  - Set gain back to being an attack-set parameter, not a global one.
- 2008-08-31 - v 3.0
  - added "mute" inlet and "enable" togglebox and associated state variables/OSC mappings.
  - added "scale to wave amp" control. When unchecked, the filter opens fully on attack, instead of scaled by the note amplitude (scaling was always on in earlier versions).
  - made "gain" an instance-GLOBAL parameter; moving the "gain" slider instantly attenuates all voices associated with the [polywavesynth] instance. OSC mapping of gain changed accordingly.
  - tweaked gui to more closely match [polygrainsynth].
- 2007-10-27 - v 2.3
  - Gain parameter moved to [polyWaveVoice~], to prevent changes between notes.
  - Gain parameter made accessible through OSC.
- 2007-09-29 - v 2.2
  - Fixed bug causing voice parameters to change on release during polyphony.
- 2007-09-17 - v 2.1
  - fixed [polypoly] path issue; polywavesynth folder no longer needs to be in Pd path.
- 2007-09-16 - v 2.0

- eliminated all global receives (except for SSSAD)
- added OSC inlet and OSC addressing of all parameters
- parameters now only changed on attack (used to be on release, too)
- added more values for modulation ratios
- included dependent abstractions in local "lib" directory
- cleaned up panning considerably
- exposed voice-stealing switch as a [polywavesynth] argument
- made all objects lowercase...dedicated to frankbarknecht :-)

- 2007-09-05 - First release
  - (named polyWaveSynth and polyWaveVoice~)

License

Everything in the polywavesynth download is released under the same license as Pd, with myself and others included as authors:

Phil Stone
http://pkstonemusic.com