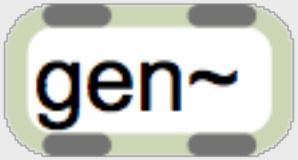


Graham Wakefield
Cycling '74

Max

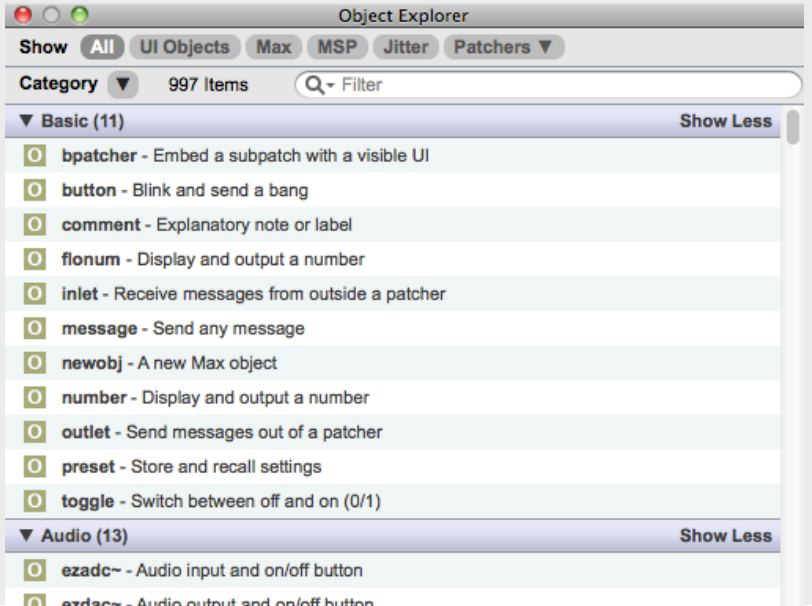
1989 Max
A way of combining C modules via messages

1997 MSP
Signal processing chains, synchronous

2003 Jitter
Matrix processing, 2D and 3D graphics

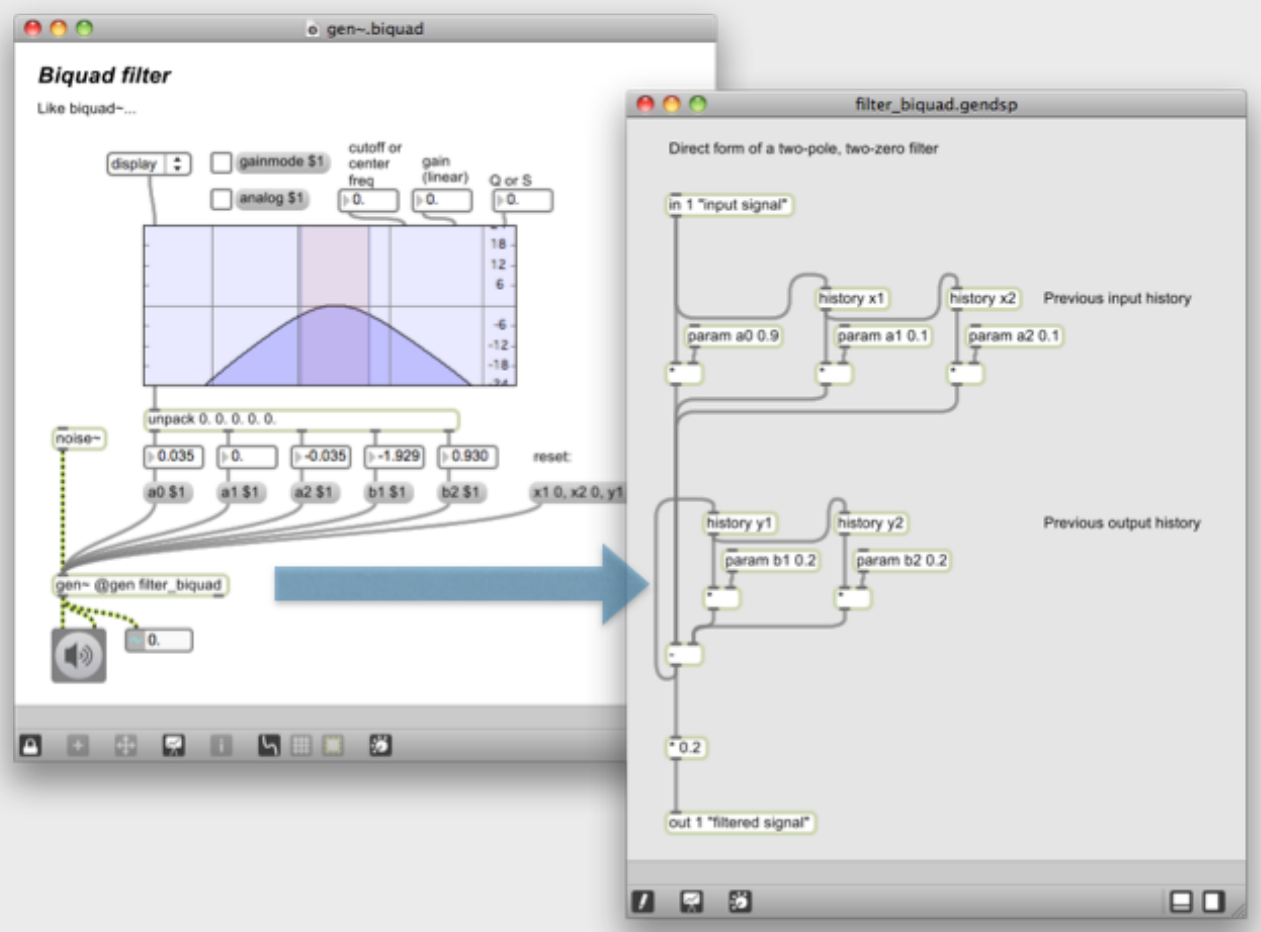
2011 Gen
Code generation and runtime compilation of audio and matrix processing objects

700+ Max objects



maxobjects.com 4781+ objects

What is Gen?



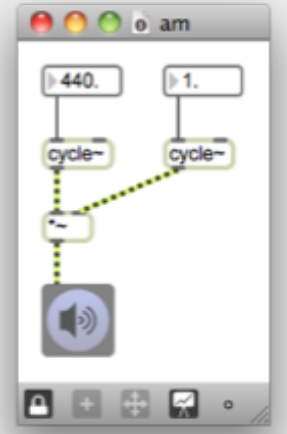
Efficient low-level DSP in Max

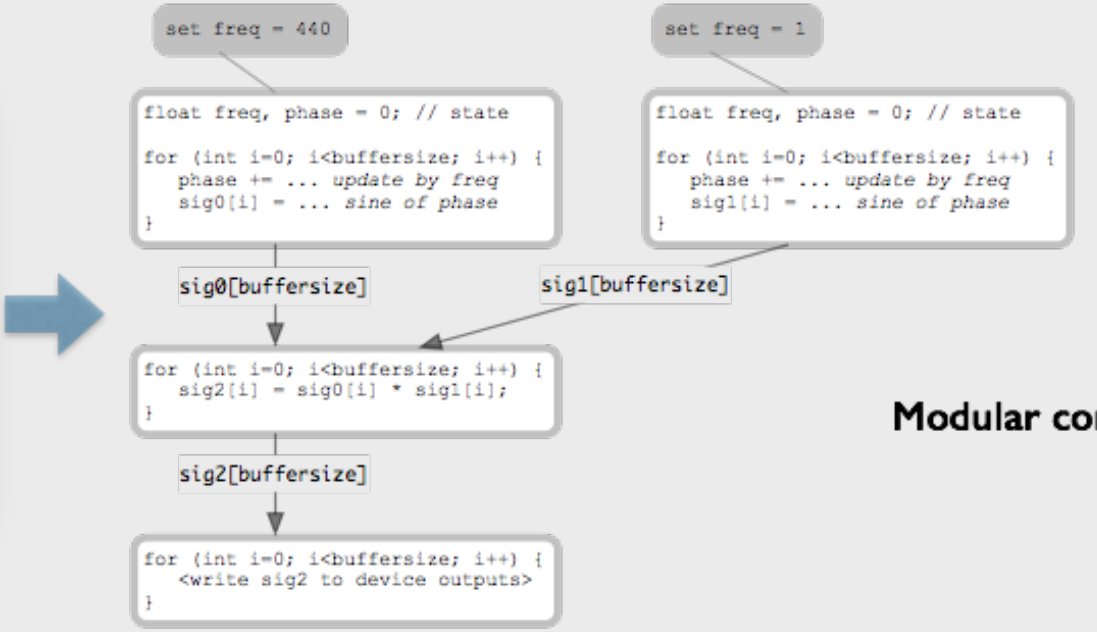
"*Generate code and take it with you*"

Flexibility + efficiency via run-time code generation

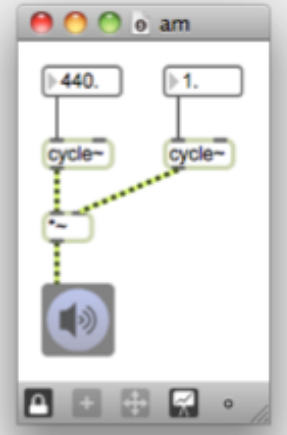
Why not convert to machine-code on the fly?

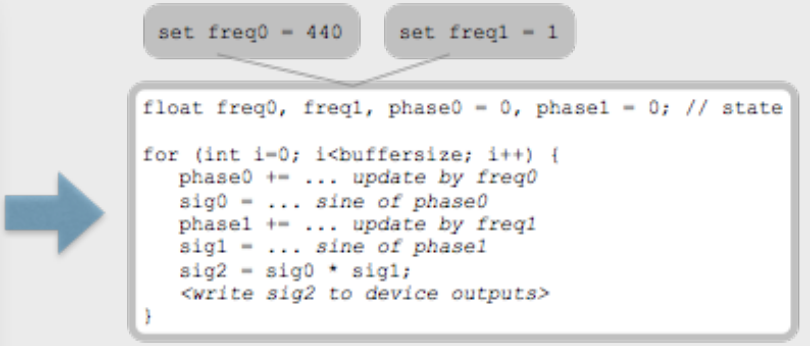
(reflective meta-programming & dynamic compilation)





Modular compromise





Via code generation:

no wasted memory buffers
better cache locality
global optimizations
single-sample feedback
possibly reduced latency
simpler / more expressive primitives

Summary

Consider patch as specification for compiler,
rather than interpreted network of black-box objects

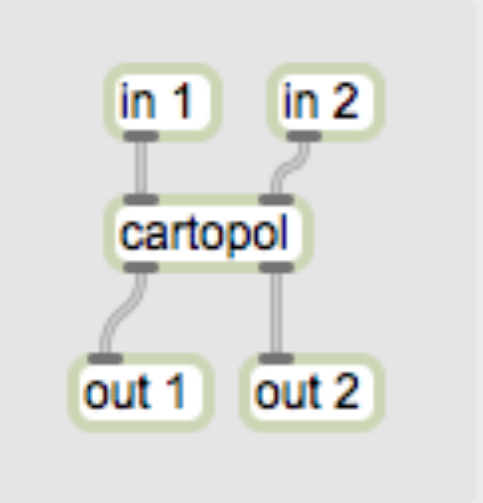
Embed compiler in Max, invoke it at each edit
Embed results in Max, or export as C++

Requires/allows a new patching interface with slightly different semantics

Differences from Max/MSP patching

Different (smaller) set of objects

- inspired by Max/MSP objects
- most low-level objects exist
- many shared between gen~, jit.gen
- no tilde (~) postfix



No messages (synchronous like MSP)

- no right-to-left order output
- no left-inlet triggering, no hot/cold inlets
- multiple input connections are summed
- @attributes are static
- no UI objects :-((maybe in the future?)

Type agnostic

- no need to distinguish ints and floats
- gen recompiles to adapt to input type
- signal- or control-rate according to what is connected to gen~

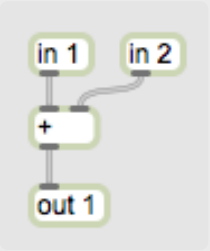
Differences from Max patching

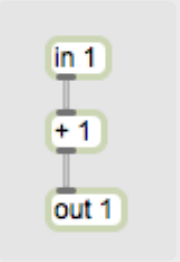
Connect with outside world via **in, out** and also **param, buffer**.

- use @comment for inlet/outlet assist
- **param** objects are "control-rate"
- **buffer** references can be changed dynamically

Objects are highly argument-dependent

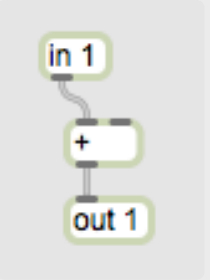
- **e.g.** binary operators with an argument have only one inlet
- **e.g.** delay operator argument sets the number of delay taps
- ... many more examples





Unconnected inputs get default values

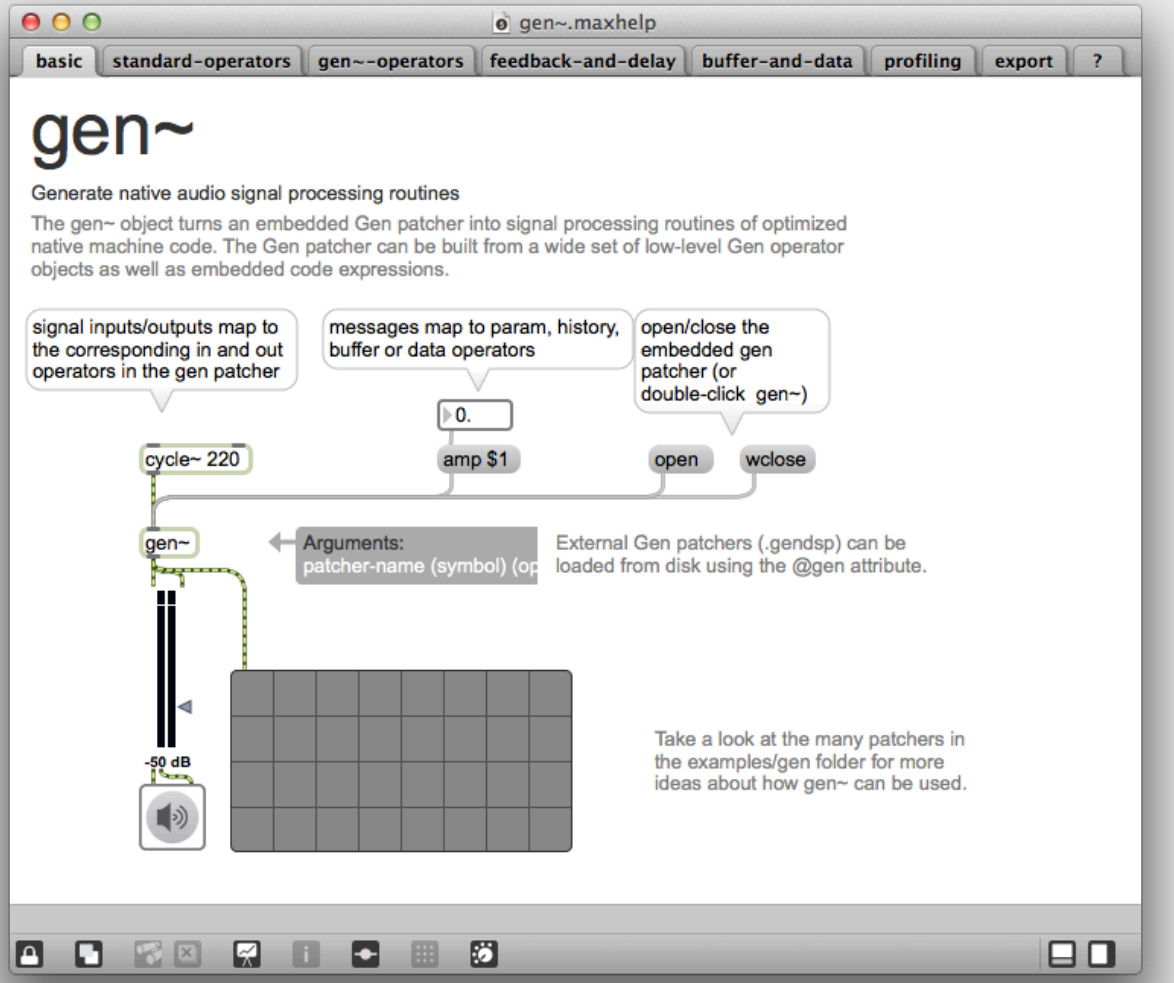
- unconnected inlets get default values
- identity or other sane default



Use predefined constants and param names in arguments

Gen~ operators

Overview in gen~.maxhelp and reference pages.
< 100 operators in total, mostly inspired by Max/MSP objects
Objects are mostly low-level; for oscillators, filters etc. see gen~ examples folder.



Single-sample feedback

The flow of data inside the gen~ object is like MSP in that it's synchronous,
but instead of operating on a block of samples, we're working with one sample at a time
– which lets us do things with single-sample feedback that we could never do before.

[history]

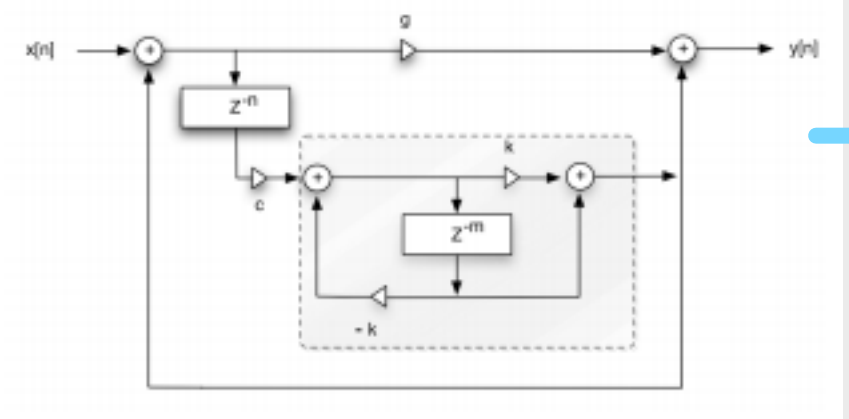
- The Z^{-1} of gen patching
- Provides one sample of delay
- Allows feedback patching
- Essential to filter design, signal analysis etc.
- Can also be named and accessed externally like [param] objects

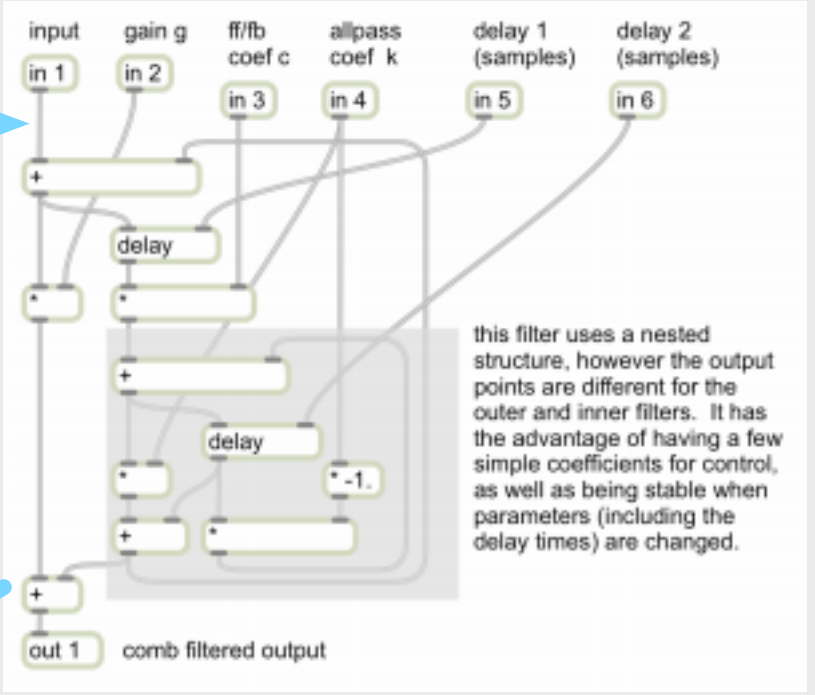
[delay]

- A variable delay down to 1 sample (0 samples if @feedback is disabled)
- Allows feedback patching
- Essential to high-frequency physical models, diffusers, low-latency FX, etc.
- Delay data retained between edits!
- Supports multi-tap outputs, many interpolation modes

Single-sample feedback

Jae hyun Ahn, Richard Dudas. ***Musical Applications of Nested Comb Filters for Inharmonic Resonator Effects.*** *ICMC 2013.*






```
Delay delay_1(44100);
Delay delay_2(44100);
tap_3 = delay_1.read(in6);
mul_4 = in4 * -1.;
mul_5 = tap_3 * mul_4;
tap_6 = delay_2.read(in5);
mul_7 = tap_6 * in3;
add_8 = mul_7 + mul_5;
mul_9 = add_8 * in4;
add_10 = mul_9 + tap_3;
add_11 = in1 + add_10;
mul_12 = add_11 * in2;
add_13 = mul_12 + add_10;
out1 = add_13;
delay_1.write(add_8);
delay_2.write(add_11);
```

Also:

The TR-808 Cymbal: a Physically-Informed, Circuit-Bendable, Digital Model. Kurt James Werner, Jonathan S. Abel, Julius O. Smith.

A Physically-Informed, Circuit-Bendable, Digital Model of the Roland TR-808 Bass Drum Circuit. Kurt James Werner, Jonathan S. Abel, Julius O. Smith.

Buffer and data

[buffer] and [data] are for multi-channel data-storage, with read & write operations. Contents are retained between edits.

[buffer]

- References an MSP [buffer~] object (32-bit)
- Reference can be changed by Max message to gen~

[data]

- A 64-bit multi-channel storage, local to genpatcher
- Can copy data from MSP buffer~ by Max message to gen~

[sample], [wave], [peek], [lookup], [nearest]

- Basically the same object but different @attribute defaults:
- @index by samples, phase, lookup/signal, or wave (start/end)
- @boundmode ignore, wrap, fold/mirror, clip/clamp
- @channelmode ignore, wrap, fold/mirror, clip/clamp
- @interp none/step, linear, cosine, cubic, spline

= 1280 code-generated permutations!

[poke], [splat]

- Writing samples into buffer/data
- Splat adds support for interpolated overdubbing

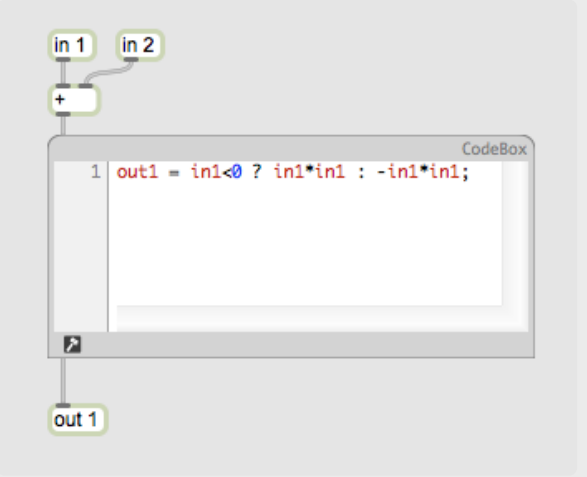
[dim], [channels]

- Reports size of buffer/data

GenExpr: expr and codebox

Code side-bar shows textual-equivalent of any visual patcher.
This simplified C-like language is called *GenExpr*.
It can also be used within the patcher:

[expr]
- short expressions can be neater than multiple objects



[codebox]
- complex, multi-line code
- inline error reporting

Beyond visual patching:
- if/then/else conditionals
- while and for loops
- user-defined functions
- include external .genexpr files of functions

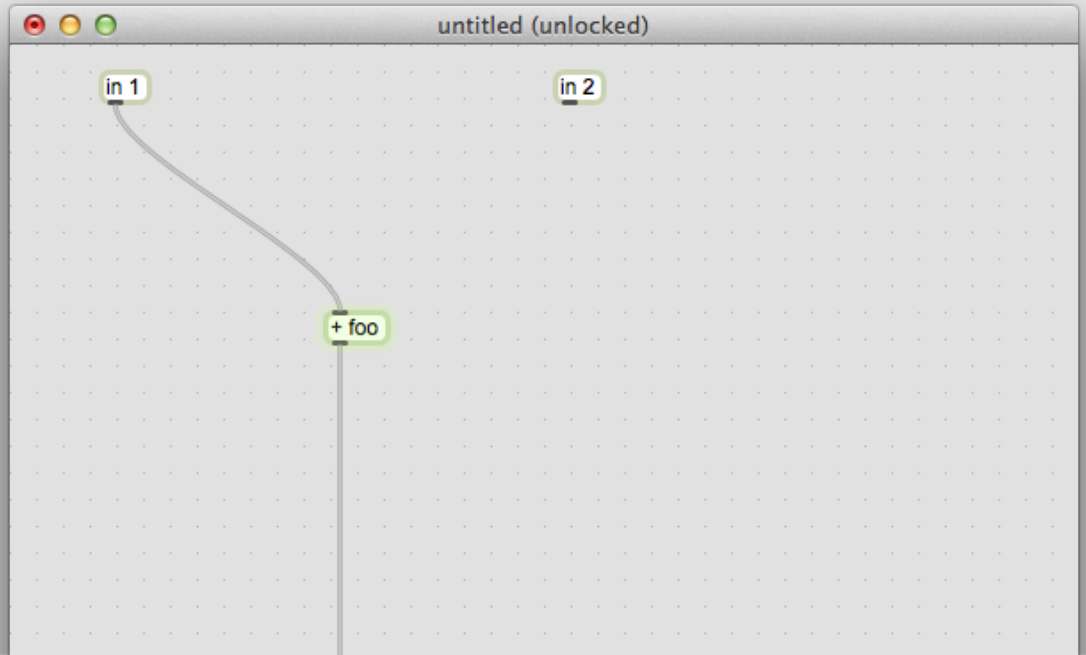
Easy to port existing DSP code (e.g. musicdsp.org) to GenExpr!

Learning Gen

Gregory Taylor's Tutorial Videos:
http://cycling74.com/wiki/index.php?title=gen~_For_Beginners

Gen Forums (helpful community, plenty of sharing):
http://cycling74.com/forums/forum/gen/

In-Max help:
- Look at the examples folder first!
- Alt/option-click objects for assistance bubble
- Ref sidebar as you select objects
- Double-click on Max Window errors to highlight gen operator



gen~.chaos.maxpat
gen~.chopper_repeat.maxpat
gen~.chopper.maxpat
gen~.comb.maxpat
gen~.computed_sine.maxpat
gen~.count.maxpat
gen~.crossover.maxpat
gen~.deltaclip.maxpat
gen~.drunk.maxpat
gen~.edge.maxpat
gen~.fbam.maxpat
gen~.ffm.maxpat
gen~.filters.maxpat
gen~.flange_chorus.maxpat
gen~.flute.maxpat
gen~.fm_bells.maxpat
gen~.freeverb.maxpat
gen~.gigaverb.maxpat
gen~.interpolation.maxpat
gen~.karplus_strong_strange.maxpat
gen~.karplus_strong.maxpat
gen~.livelooper.maxpat
gen~.minmax.maxpat
gen~.modfm.maxpat
gen~.moogladder.maxpat
gen~.overdrive.maxpat
gen~.performance.maxpat
gen~.pfft_centroid.maxpat
gen~.pfft_example.maxpat
gen~.pfft.vectral.maxpat
gen~.phasor.maxpat
gen~.pitchshift.maxpat
gen~.pulsar.maxpat
gen~.random.maxpat
gen~.shaker.maxpat
gen~.sincinterpolation.forloop.maxpat
gen~.sincinterpolation.maxpat
gen~.slicer.maxpat
gen~.slide.maxpat
gen~.spectraldelay_feedback.maxpat
gen~.spectraldelay.maxpat
gen~.thresh.maxpat
gen~.trapezoid.maxpat
gen~.vosim.maxpat

Gen Tips

- Debugging: add an [out] object, hooked up to number~, scope~, spectroscope~, capture~, etc.

- No messages means no [trigger] etc.; use 0/1 signals.

- Use code sidebar to help understand processes

- Use [param] for "control-rate" processing

- Use abstractions for repeated units (no feedback yet)

- Efficiency gains of gen~ increase as patcher gets bigger.

Algorithm	**Instances**		μ	σ	max	σ
Single-addition	100	MSP	5.7%	0.39	11%	0.49
		Gen	5.0%	0.53	10%	0.45
Multi-addition	20	MSP	11.4%	0.26	22%	0.67
		Gen	2.8%	0.05	7%	0.49
Dual FM	20	MSP	22.1%	0.62	37%	1.10
		Gen	12.7%	0.26	23%	0.50
Sinc interpolator	10	MSP	27.4%	0.25	46%	1.97
		Gen	8.5%	0.18	18%	0.70

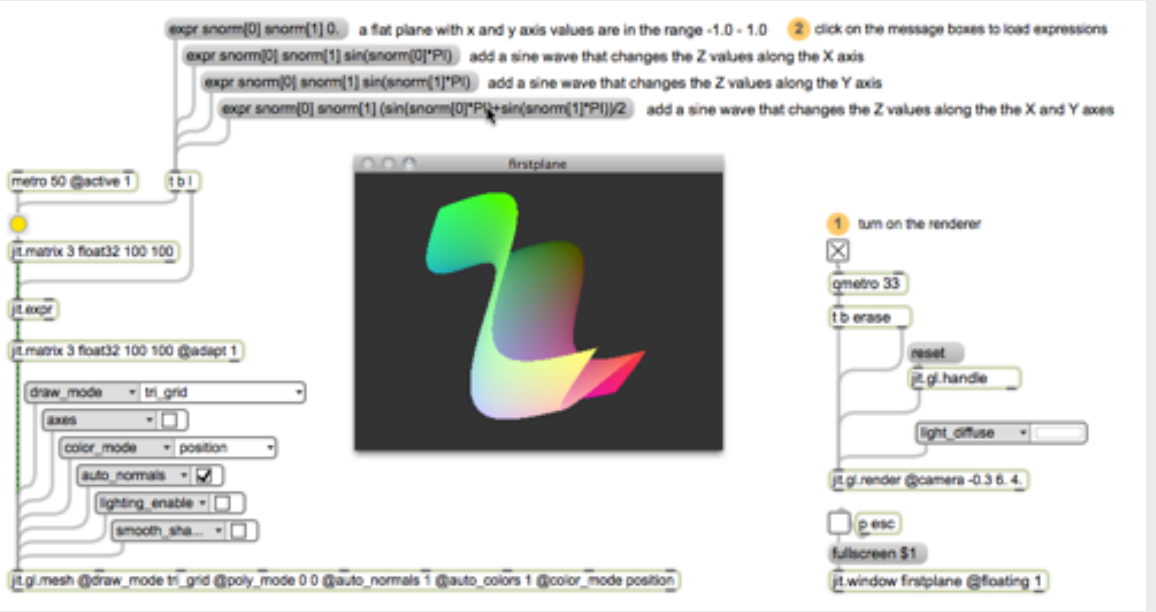
Gen and Jitter

Jitter domains:

jit.gen generalized matrix processing (C++)
jit.pix image processing (C++)
jit.gl.pix graphic hardware accelerated image processing (GLSL)

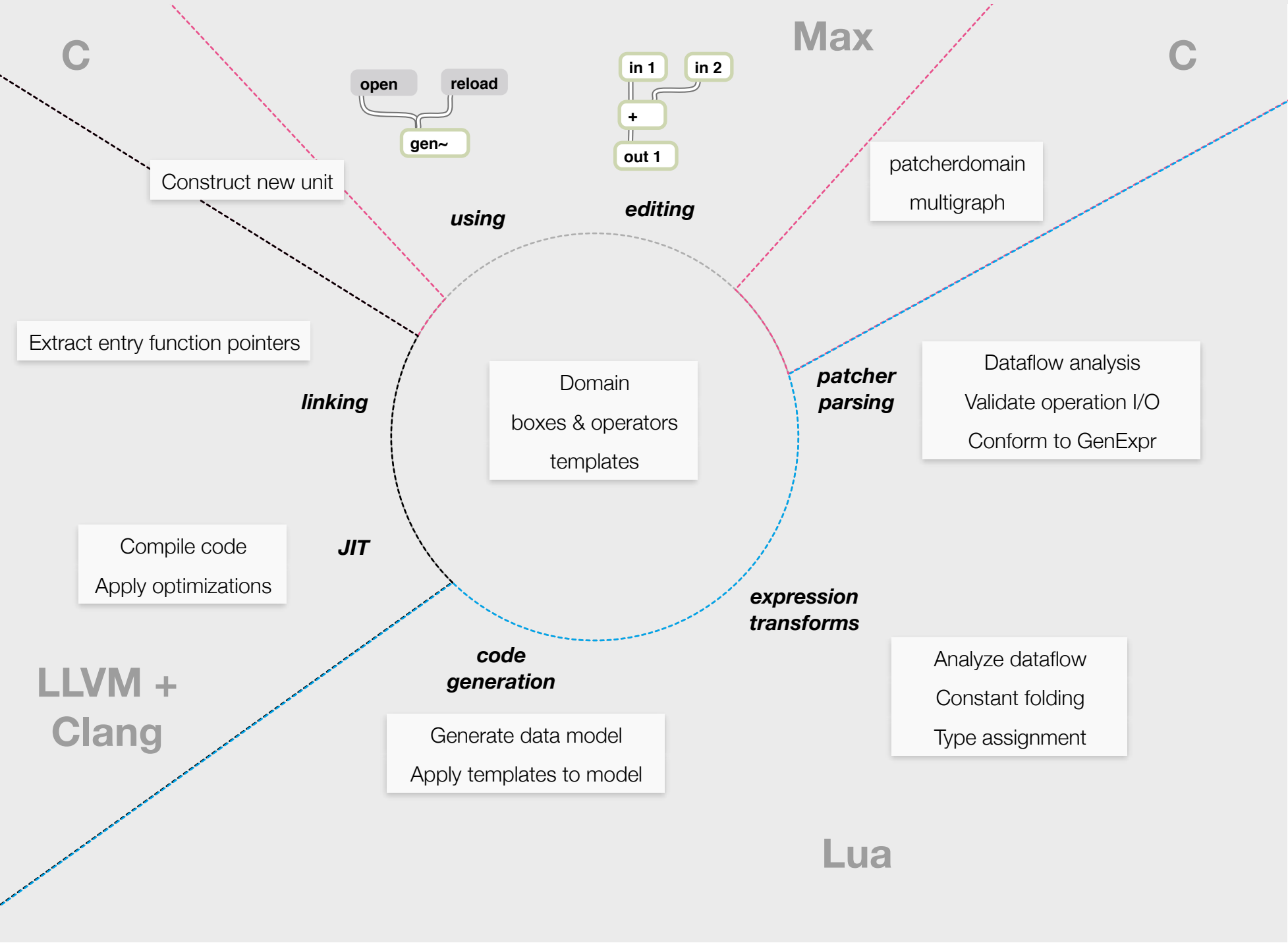
Many operators shared with gen~

Some operators specific to Jitter
(e.g. vec, swiz, ...)



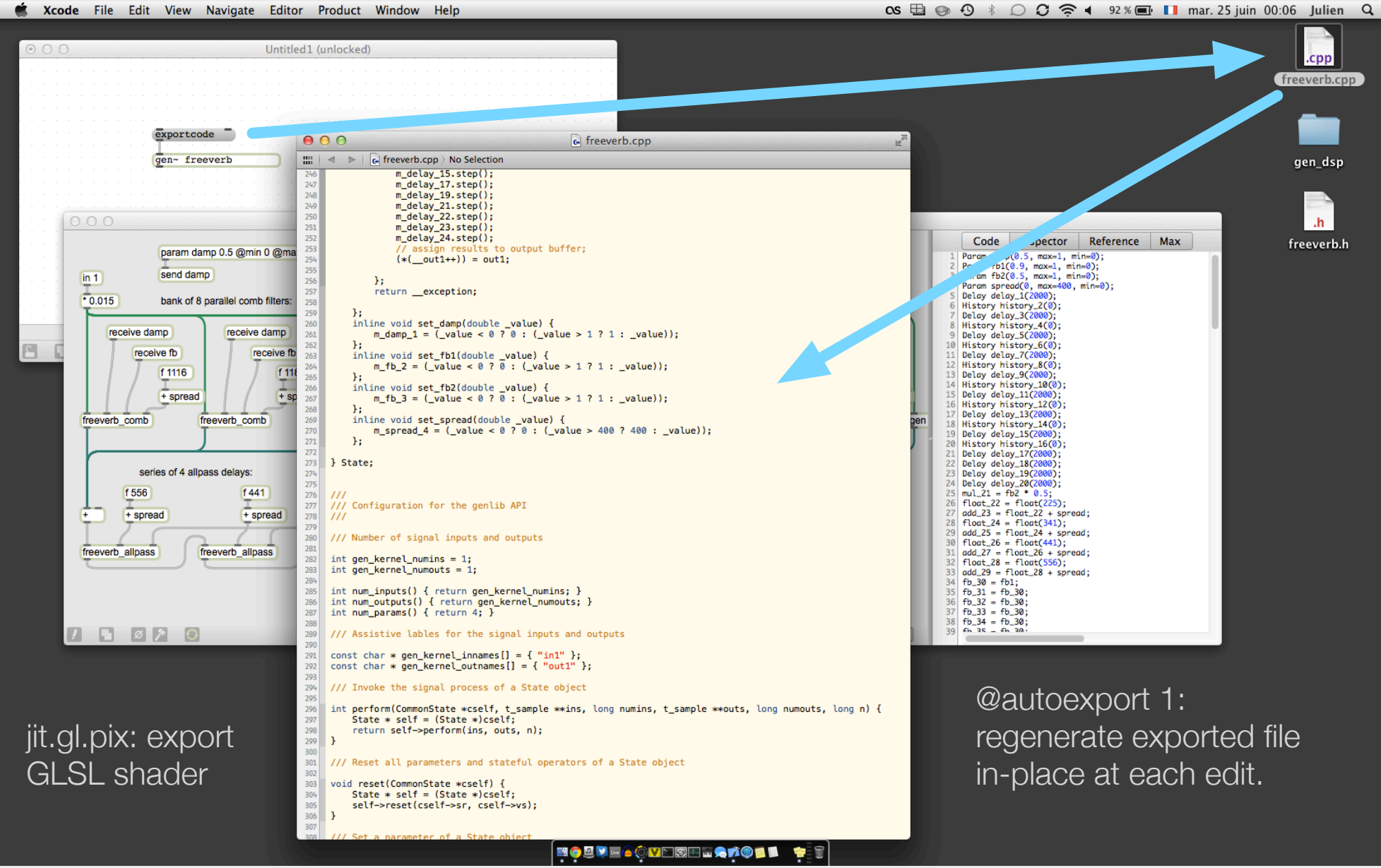
- vector processing similar to GLSL fragment shaders
- up to 32D vectors
- up to 32D matrices
- has coordinate and vector ops
- has matrix sampling capabilities
- automatic parallelization of calculations
- settable kernel precision (fixed, float, double)
- all inputs and outputs are coerced to the same format

What happens when you make an edit



Code Export

http://cycling74.com/products/gen/codeexport/
Also see Julien Bayle's blog!



jit.gl.pix: export GLSL shader

@autoexport 1: regenerate exported file in-place at each edit.

Now being used to develop devices at Ableton, sound design for cars at Audi, ...

Observations

Enhances the fluidity of user experience

- barrier between high- and low- levels of abstraction is reduced
- get immediate feedback on a change for real-time auditioning of edits
- no more C coding (and no more OSX/Window issues)

Increases the space of exploration

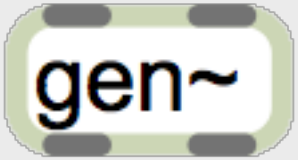
- better efficiency gains from use of runtime information
- easy to handle the combinatorial explosion of structural permutations
- uses less memory since only those permutations used are instantiated
- sub-block size processing
- control flow in GenExpr

Gen patchers are self-contained units sharing a common interface

- encourages sharing, posting and discussion since Gen patchers are interchangeable
- used in teaching/research at Columbia, CCRMA, etc.

Related:

- faustgen~
- LuaAV, Extempore
- Reaktor Core



Graham Wakefield
Cycling '74