SC

A fun and easy way to do distortion is with the methods .tanh and .distort, just up the amplitude and it will distort without digital clipping. (also .softclip)

Patterns are a new vocabulary to learn. Until you know a critical mass of them, it can be hard to trust them. That's the purpose of this guide!

```
a = Array.fill(1000, {arg i; i+1});
Post <<  a;
```

The user writes poetry (so to speak) in the SuperCollider language which is then paraphrased in OSC prose by the sclang interpreter, to be sent to the server.

```
{SinOsc.ar(2, 0, 1).clip(0, 0.2).poll}.plot(2);
```

```
[3,4] ++ (7 ! 7) vs [3,4] ++ 7 ! 7 (difference)
```

```
([3,4] ++ (7 ! 7)).stutter(2).scramble
```

```
([3,4] ++ (7 ! 7)).stutter(2).scramble.putSeries(0, 2, 16, 5.rand)
```

//thisProcess.nowExecutingPath returns a string representing the path to the SC code file in which code is currently being evaluated. The SC file must have been saved at least once. In combination with PathName.new, the parentPath method, and string concatenation using the "++" operator, it is possible to create dynamic file names that change appropriately when a project folder is moved to a new location or new computer. It is always a good idea to create a project folder that contains your code file, and a subfolder of audio files, so that your audio resources and code will always travel together and remain as a single package.

//these lines should read two files into buffers, and you shouldn't need to make any changes to the code on your computer, so long as the "livestream_code_week04" folder remains intact.

```
~flower = Buffer.read(s,
PathName.new(thisProcess.nowExecutingPath).parentPath ++ "sounds/
flower_pot.aiff");

~bowl = Buffer.read(s,
PathName.new(thisProcess.nowExecutingPath).parentPath ++ "sounds/
```

prayer_bowl_roll.aiff");

~flower.play;

~bowl.play

LOOk for Pswitch - https://swiki.hfbk-hamburg.de/MusicTechnology/186
Pfsm

Rest for silent event

Pdef
Pdefn
Pbindef
- from proxy classes live coding

```
(
e = (
    a: Pbind(\dur, 0.1, \degree, Pseq([0, 5, 4, 3, 2])),
    b: Pbind(\dur, 0.06, \degree, Pseq([7, 8, 7, 8])),
    c: Pbind(\dur, 0.3, \degree, Pseq([0, 1, 2], 2))
);

x = Pdict(e, Pseq([
            \a, \b,
            Prand([\a, \c])
        ], 4)
    );
x.play;
)

//
```

```
// an already existing Pdef can be incrementally changed

Pdef(\x, Pbind(\instrument, \Pdefhelp, \dur, 0.3));
Pdef(\x).play;

Pbindef(\x, \degree, 7.rand);
Pbindef(\x, \degree, Pseq([0, 7, 3, 7, 4], inf), \dur, Pn(Pseries(0.2, -0.02, 10)));
Pbindef(\x, \stretch, 2);
```

```
//
________________________________________________________________________________
______

Pdefn(\x, Pn(1, 3));
Pdefn(\y, Prand([5, 9, 1], 2));
Pdefn(\z, Pdefn(\y) * 2);

(
Pdef(\play,
    Pbind(
        \instrument, \gpdef,
        \harmonic, Pnsym(Pseq([\x, \x, Prand([\x, \y]), [\z, \y], \y], inf)).trace,
        \dur, 0.2, \note, 10
    )
).play;
)

// change root pattern:
Pdefn(\x, Pn(2, 3));
Pdefn(\x, Pseq([1, 3, 1, 2, 1, 4, 5]));
Pdefn(\x, Pseq([1, 3, 1, 2, [1, 3], 4, 5]));

Pdefn(\x).quant_(4);
or
Pdefn(\x, Pn(2, 3)).quant_(4);
or
Pdefn(\x, Pn(2, 3)).quant = 4;


Pbind(\type, \note..... by default for note-type events
Pbind(\type, \rest.... for rest-type events


EnvGen.kr(Env.adsr(atk, dec, sus, rel), gate, doneAction: 2);

Pn(Pgeom(0.5, 07, 7), inf) =

0.5
0.35
0.245
0.1715
0.12005
0.084035
0.0588245
x_inf
```

```
(100,300..900) -> [ 100, 300, 500, 700, 900 ]
(100,200..900) -> [ 100, 200, 300, 400, 500, 600, 700, 800, 900 ]
(100,200..900).clipExtend(2) -> [ 100, 200 ]
(100,200..900).clipExtend(10) -> [ 100, 200, 300, 400, 500, 600, 700, 800, 900, 900 ]
```

```
Pbindef(\key).play(~clock, quant: Quant(4, 0, 0));
Pbindef(\key, \amp, Pseq([0.1, 0.5], inf)).quant_(Quant(4, 0, 0));
// or quant: [4, 0, 0] or quant: 4 or quant = 4
```

Example: `Pbindef(\nupgssamp2, \freq, 1).quant(8)`

Event Types: https://doc.sccode.org/Overviews/Event_types.html

Rest() = \

See Ptuple
Pfunc

Pseg - good for fading in and fading out patterns
or \amp, AnyPattern * Env.new([1,0],[10],[-3]).asPseg, etc

.sort

```
~clock = TempoClock.new(150/60).permanent_(true);
~postinfo = {(~clock.beats%8+1).postln;1;};
~clock.schedAbs(~clock.nextBar, {~postinfo.()});
~postinfo = {};
```

```
Pbindef(\pat1,
\instrument, \granulator,
\dur, Pseq([1/4,1/8], inf)
).quant(4);

Pbindef(\pat1, \bufnum, Prand([~buffer1.bufnum, ~buffer2.bufnum], inf)).play;

Pbindef(\pat1).pause;
Pbindef(\pat1).resume;
Pbindef(\pat1).stop;
```

Shortcut notation : Just like you can concatenate arrays with ++, you can also concatenate patterns the same way. Writing pattern1 ++ pattern2 is the same as writing Pseq([pattern1, pattern2], 1) .

Direct array indexing
Patterns can be chosen in arbitrary order by index. This gives you more control than Pwrand. Both Pindex and Pswitch can be used for this.

```
// scale degree segments, every fifth choice is odd-numbered only
(descending)
(
var    n = 10,
    scaleSegments = Array.fill(n, { |i|
        if(i.odd) {
            Pseries(11, -1, rrand(5, 10))
        } {
            Pseries(rrand(-4, 4), 1, i+2)
        }
    });

TempoClock.default.tempo = 1;
p = Pbind(
    \degree, Pswitch(scaleSegments, Pseq([Pwhite(0, n-1, 4), Pwhite(0, n-1,
1).select(_.odd)], inf)),
    \dur, 0.125
).play;
)

p.stop;

(
Pbindef(\patz).clear;
Pbindef(
        \patz,
        \instrument, \test,
    \a, Pseries(1, 1, inf).trace,
        \dur, Pif(
        Pkey(\a) > 5,
                Pseq([0.1,0.2,0.3,0.2, 0.1], inf),
        0.6
    ).trace
).play;
)
```

```
{SinOsc.ar(5).unipolar(1) * 20}.plot(2);
{SinOsc.ar(5).unipolar(20)}.plot(2);
{SinOsc.ar(5).range(0, 20)}.plot(2);

{Lag2.ar(Fold.ar(LinExp.ar(LFNoise2.ar(50), -1.0, 1.0, 20, 45), 25, 40),
2)}.plot(2);
```

one track one EventStreamPlayer, all patterns paralleled
or
one track one Pbindef, iterate via another Pbindefs below

```
Pbindef(
\example,
\instrument, Pseq([
[\synth1, \synth3],
[\synth2, \synth1, \synth5],
[etc…]
], inf)
).play(~clock, quant:8);

Pbindef(
\example,
\dur, 1
).clock(~clock);
```

reverse (?)

‘[1, 3, 6]
‘ is Ref.new(thing)
~ create a Ref of an object.
examples:

```
a = [3, 7, 5, -12].midiratio.asRef;
b = `[0.11, 0.11, 0.11, 0.2];
```

instead [0.11, 0.11, 0.11, 0.2] is going to duplicate the ugen which using it 4 times for four channel expansion

Splay.ar > multichannel expanded signal summed to stereo
{Splay.ar(Impulse.ar([1, 3, 5, 7, 9]).scramble)}.play;

TZeroXBufRd.ar - buffer player for pulsar

Daniel Mayer | kitchen studies

NTube.ar - tube physical modelling filter
JPverb.ar - reverb
SawDPW.ar - saw with lowest aliasing
SawDPW.ar - pulse with lowest aliasing

```
{NTube.ar(Splay.ar(Impulse.ar([1, 3, 5, 7, 9]).scramble),
`[0.97,1.0,1.0,1.0,1.0,0.97], `[0.5,MouseY.kr(-1.0,1.0),0.2,-0.4],
`([0.01,0.02,0.01,0.005,0.05]*MouseX.kr(0.001,1.0,'exponential')))*0.1}.play;

{NTube.ar(Splay.ar(Impulse.ar([1, 3, 5, 7, 9]).scramble),
`[0.97,1.0,1.0,1.0,1.0,0.97], `[0.5,MouseY.kr(-1.0,1.0),0.2,-0.4],
`([0.01,0.02,0.01,0.005,0.05]*SinOsc.kr(MouseX.kr(1, 3)).linlin(-1.0, 1.0, 0.001,
0.1)))*0.5}.play;

(
{
	var snd, input, delayline, feedback, delaymix;
	snd = (NTube.ar(Splay.ar(Impulse.ar([1, 3, 5, 7, 9]).scramble),
`[0.97,1.0,1.0,1.0,1.0,0.97], `[0.5,MouseY.kr(-1.0,1.0),0.2,-0.4],
`([0.01,0.02,0.01,0.005,0.05]*LFNoise0.kr(MouseX.kr(1, 30)).linlin(-1.0, 1.0,
0.001, 0.1)))*2).softclip;
	input = snd;
	feedback = LocalIn.ar(1) + input;
	delayline = DelayC.ar(PitchShift.ar(feedback, MouseX.kr(0.0001,0.5), 0.1,
0.001, 0.001), 1.0, MouseX.kr(0.0001,0.5));
	feedback = LocalOut.ar(delayline * 0.99);
	delaymix = Limiter.ar(delayline);
	Out.ar(0, Pan2.ar(delaymix, 0.0, 1.0));
}.play
)

(
SynthDef(\t, { | distr = 1 |
        var array, mode, snd;
        array = [[20],[30]];
        mode = Select.kr(distr, array);
        mode.postln;
        snd = mode[0];
}).add;
)

Synth(\t, [\distr, 0]);


(
SynthDef(\testx, { arg fade = 0;
  var arrayMy, selectParam;
  arrayMy = Array.interpolation(5, [0.1, 1.0], [1.0, 0.1]);
  selectParam = Select.kr(fade, arrayMy);
  // selectParam[0].postln;
  Out.ar(0, [
    SinOsc.ar(440.0, 0, selectParam[0]),
    SinOsc.ar(440.0, 0, selectParam[1])
  ]);
}).add;
)
```

PbindFx

ProxySpace or Pbindef?

Pdict is good for scheduling patterns and composition (?) for ProxySpace patterns

```
(
s.options.numPrivateAudioBusChannels = 1024;
s.options.memSize = 8192 * 16;
s.options.numInputBusChannels = 0;
s.options.numOutputBusChannels = 16;
s.boot;
c = TempoClock(1).permanent_(true);
p = ProxySpace(s, 'proxyName', c);
p.push;
)
```

```
{VarLag.kr(LFPulse.kr(10), 0.04, 0, \sin).range(-0.5, 0.5)}.plot(2);

{2}.dup = 2.dup(2) = 2 ! 2
```