INSTITUTE OF SONOLOGY

Royal Conservatoire The Hague

More Patterns

Programming and Music

http://www.bjarni-gunnarsson.net

Pbind

Pbind = Pattern Bind - connects parameter names with patterns of values.

Concepts

- Events: Dictionaries of instructions for making sound
- Each event specifies: pitch, timing, amplitude, and other parameters
- Patterns generate sequences of values
- .play starts the pattern on a clock (TempoClock by default)

Control

.stop - stop the pattern

.pause / .resume - pause and resume

.reset - start from beginning

Pitch

1. \degree - Scale degrees

Uses scale steps: 0, 1, 2, 3...

Works with \scale (Scale.major, Scale.minor, Scale.chromatic, etc.)

2. \note - Chromatic notes within an octave

0-11 (12 semitones)

Combine with \octave to specify register

3. \midinote - MIDI note numbers

60 = middle C, 69 = A440

Each number = one semitone

4. \freq - Direct frequency in Hz

Raw frequency values (440 = A4)

Duration

\dur - Controls WHEN the next note plays

=> \dur, 1 // 1 beat

\legato - Controls HOW LONG the note lasts

=> 0.8 (default) = 80% of duration

Rest() - Creates silence

=> \midinote, Pseq([60, 64, Rest(), 72], inf)

TempoClock - Global tempo control

=> t = TempoClock(120/60); // 120 BPM

Amplitude

\amp - Direct amplitude (0.0 to 1.0)

=> \amp, Pseq([0.6, 0.2, 0.3], inf) // accents

\db - In decibels (more natural)

=> \db, Pseq([-20, -10, -6], inf)

Advanced

Probability Distributions

Shaping Musical Character

Probability distributions create different sonic behaviours.

* Exponential (Pexprand) favors lower values
* (Pgauss) clusters around center, focused pitch regions.
* Cauchy (Pcauchy) creates mostly centered values with outliers.
* Weighted random (Pwrand) lets you bias choices for introspection.

Patterns: *Pexprand, Pgauss, Pcauchy, Ppoisson, Phprand/Plprand, Pwrand*

Tendency Masks

Controlled Randomness

Tendency masks use two envelopes (upper/lower bounds) that evolve over time, creating a corridor for random values.

Created by Koenig, this balances chance with direction.

Penv and Pseg serve as boundaries, combined with Pwhite for controlled chaos.

Patterns: *Penv(levels, times), Pseg(values, times), Pwhite(lower_env, upper_env)*

State Machines & Markov Chains

Memory and Context

State machines (Pfsm) make decisions based on current state, creating pathways with branching choices. Markov chains add probability: define how frequently certain moves occur.

First-order chains remember only current state; higher-order remember longer histories.

Patterns: *Pfsm([states...]), MarkovSetN([[state, next, probs]...], order)*

Exercises

```
(
NF(\iop, {|freq=78, mul=1.0, add=0.0|
    var noise = LFNoise1.ar(0.001).range(freq, freq + (freq * 0.1));
    var osc = SinOsc.ar([noise, noise * 1.04, noise * 1.02, noise * 1.08],0,0.2);
    var out = DFM1.ar(osc,freq*4,SinOsc.kr(0.01).range(0.92,1.05),1,0,0.005,0.7);
    HPF.ar(out, 40)
}).play;
)

(
NF(\dsc, {|freq = 1080|
    HPF.ar(
        BBandStop.ar(Saw.ar(LFNoise1.ar([19,12]).range(freq,freq*2), 0.2).excess(
        SinOsc.ar( [freq + 6, freq + 4, freq + 2, freq + 8])),
        LFNoise1.ar([12,14,10]).range(100,900),
        SinOsc.ar(20).range(9,11)
    ), 80)
    ;
}).play;
)
```

```
var <>pindex, <>cindex;

initialize {
    if(pindex.isNil, { pindex = 1000 });
    if(cindex.isNil, { cindex = 2000 });
}

clearProcessSlots {
    pindex = 1000;
    (this.pindex - 1000).do{|i| this[this.pindex+i] = nil; }
}

clearOrInit {|clear=true|
    if(clear == true, { this.clearProcessSlots() }, { this.initialize() });
}

transform {|process, index|
    if(index.isNil && pindex.isNil, {
        this.initialize();
    });

    pindex = pindex + 1;
    this[pindex] = \filter -> process;
}

control {|process, index|
    var i = index;

    if(i.isNil, {
        this.initialize();
        cindex = cindex + 1;
        i = cindex;
    });

    this[i] = \pset -> process;
}
```

```
(
NF(\depfm, {|freqMin=5, freqMax=20, mul=20, add=80, rate=0.5, modFreq=2100, index=0.3, amp=0.2|
    var trig, seq, freq;
    trig = Dust.kr(rate);
    seq = Diwhite(freqMin, freqMax, inf).midicps;
    freq = Demand.kr(trig, 0, seq);
    HPF.ar(PMOsc.ar(LFCub.kr([freq, freq/2, freq/3, freq/4], 0, mul, add),
    LFNoise1.ar(0.3).range(modFreq,modFreq*2), index) * amp, 50)
}).play;
)
```

Exercises

Exercise 1: Exponential Distribution

Create a Pbind using Pexprand for durations (0.05 to 0.4) and Pgauss for pitch (mean 60, deviation 8). Use \instrument \sine.

Exercise 2: Weighted Random

Create a Pbind using Pwrand with three pitches [60, 61, 66] and weights [0.6, 0.3, 0.1]. Duration 0.15.

Exercise 3: Simple Tendency Mask

Create a Pbind where Pwhite selects pitches between two envelopes:
- Lower: Penv([50, 70, 50], [3, 3])
- Upper: Penv([60, 90, 60], [3, 3])

Duration 0.1.

Exercise 4: Envelope as Pitch

Create a Pbind where pitch is controlled by Penv([40, 80, 50], [2, 2]). Use Pn() to make it loop infinitely. Duration 0.08.

Exercises

Exercise 5: Simple State Machine

Create a Pbind using Pfsm with three states:

- State 0 (value 60) can go to state 0 or 1
- State 1 (value 73) can go to state 2
- State 2 (value 54) can only go to state 0

Duration 0.2.

Exercise 6: Pattern Nesting

Create a Pbind where pitch alternates between two groups using Pseq:

- First group: Pseq([60, 61], 3) - play 3 times
- Second group: Pseq([66, 67], 2) - play 2 times

Duration 0.15.

Exercise 7: Time-Limited Pattern

Use Pfindur to limit a Pbind to exactly 3 seconds.

Use Pwhite(54, 78) for pitch and 0.08 for duration.