I took figure skating lessons for a number of years as a child. I remember while skating we would listen to music played over loudspeakers mounted on the ceiling of the rink and that the sound would reverberate and echo off the walls and ice, creating a rhythmically complex and cacophonous sonic texture. This piece is designed with the resonance of an ice rink in mind. Two expendable loudspeakers should be skated around the rink by interested onlookers for the duration of the installation, causing the pan and echo to shift.

This patch applies a resonant low pass filter to a ramp wave tuned to a semi-random frequency which can be adjusted in the graphical user interface. The ramp wave is resonated at a similarly semi-randomly determined frequency and is emitted in short pulses of variable length. This signal is then run through a series of delay and dynamics filters, creating a modifiable distribution with characteristics described in the comments below. The result is a texture of changing irregular blips.

```plaintext
items.funcInit({arg items, freq, rfceil, rq, dist, delay, decay, amp, scope = ScopeV;
    items.name_("s. i. r. r. laptop patch");
    items.setItems(CR,
        freq .freq(160) .name_("frequency "),
        rfceil .freq(20000) .name_("res. freq.") ,
        rq .sp(0.1, 0.0001, 1, 0, 'exponential') .name_("bandwith "),
        dist .sp(10, 1, 100) .name_("distortion"),
        CR, delay .sp(0.85, 0.05, 5) .name_("delay time"),
        decay .sp(3.5, 0.2, 10) .name_("decay time"),
        CR, amp .amp(0.2) .name_("amplitude "),
        CR, scope
    });
```
items.sound_({var sfunc;

    sfunc =
        RLPF.ar( //a resonant low pass filter
            Saw.ar(LFNoise1.ar(20, freq.kr * 0.85, freq.kr), dist.kr).distort, //input
            LFNoise0.ar(1, rfceil.kr * 0.75, rfceil.kr * 0.9), rq.kr, //res freq,
            bandwith
                LFPulse.ar(1 + LFNoise0.ar(1, 0.9), 0.1) //short pulses //control
        );

    Scope.ar(
        scope.myView,
        Pan2.ar(
            CombN.ar(sfunc, 5, delay.kr, decay.kr) //0.05-5s comb delay
            +
            (DelayN.ar( //0-6s delay
                CombN.ar(sfunc, 0.05, 0.05, 1.2) * 0.1,
                6, LFNoise0.ar(4, 3, 3)
            )).0, amp.kr //amp
            )
        )
    );
});
})