```
/*
varilude (orgelpark)
michael winter (berlin, 2022)
to be played before, between, or after pieces where there is an organ that can be controlled by a computer
originally written for the hyperorgan (the utopia organ + the sauer organ) at orgelpark,
but the idea can be applied to other organs
*/
(
var appEnvironment;

//push new environment
appEnvironment = Environment.make;
appEnvironment.push;

s.waitForBoot({

    var hyperOrganAddr, noManuals, noOctaves, pClasses, notes,
    utopiaRegisters, sauerRegisters, sauerDivisions,
    utopiaRMStates, sauerDRStates, sauerDMStates, consoleMOStates,
    stateSelect, synth;

    hyperOrganAddr = NetAddr( "192.168.0.1", 1803 ); // define address and port
    //hyperOrganAddr = NetAddr( "127.0.0.1", NetAddr.langPort ); // for local testing

    noManuals = 4;
    noOctaves = 5;
    pClasses = [0, 4, 7, 11];
    notes = pClasses.collect({arg pClass; noOctaves.collect({arg octave; 36 + pClass + (12 * octave)})});
    utopiaRegisters = (36..68);
    sauerRegisters = [
        [12, 13, 15, 16, 17],
        [14, 18, 19, 20],
        [1, 3, 4, 7, 8, 11],
        [2, 5, 6, 9, 10, 19],
        [22, 23, 24, 25, 26, 27, 28],
        [30, 31, 32, 33, 34, 35]
    ]; // removing 21 "motor ab" and 29 "tremelo" (for now)
    sauerDivisions = (1..6);
    utopiaRMStates = utopiaRegisters.collect({noManuals.collect({-1})});
    sauerDRStates = sauerRegisters.collect({arg division; division.collect({-1})});
    sauerDMStates = sauerDivisions.collect({noManuals.collect({-1})});
    consoleMOStates = noManuals.collect({noOctaves.collect({-1})});

    stateSelect = {arg prob;  [-1, 1].wchoose([prob, 1].normalizeSum)};

    OSCFunc({arg msg;
        var register_prob = msg[3];

        //set sauer registers
        sauerDivisions.do({arg division, d;
            // set registers per division
            sauerRegisters[d].do({arg register, r;
                var selState = stateSelect.value(register_prob);
                if(selState != sauerDRStates[d][r], {
                    sauerDRStates[d][r] = selState;
                    hyperOrganAddr.sendMsg('/D' ++ division ++ '/S', register * selState);
                });
            });
            // set division per manual
            noManuals.do({arg manual;
                var selState = stateSelect.value(3);
                if(selState != sauerDMStates[d][manual], {
                    sauerDMStates[d][manual] = selState;
                    hyperOrganAddr.sendMsg('/M' ++ manual ++ '/L1/D', division * selState);
                });
            });
        });

        //set utopia registers
        utopiaRegisters.do({arg register, r;
            noManuals.do({arg manual;
                var selState = stateSelect.value(register_prob);
                if(selState != utopiaRMStates[r][manual], {
                    utopiaRMStates[r][manual] = selState;
                    hyperOrganAddr.sendMsg('/M' ++ manual ++ '/L1/S', register * selState);
                });
            });
        });

        //set note ons/offs
        noManuals.do({arg manual;
            noOctaves.do({arg octave;
                var selState = stateSelect.value(5);
                if(selState != consoleMOStates[manual][octave], {
                    consoleMOStates[manual][octave] = selState;
                    hyperOrganAddr.sendMsg('/M' ++ manual ++ '/V', selState.clip(0, 0.8), notes[manual][octave]);
                });
            });
        });
    }, '/tr');

```

```
92
93      s.freeAll;
94
95      SynthDef(\organ, {
96          var hierarchical_dust, register_prob, sines, brown_noise, white_noise;
97
98          hierarchical_dust = (
99              (TIRand.kr(0, TIRand.kr(1, 2, Impulse.kr(5)), Impulse.kr(20)) < 1) *
100             TIRand.kr(0, 1, Impulse.kr(10)) *
101             TIRand.kr(0, 1, Impulse.kr(5)) *
102             (TIRand.kr(0, TIRand.kr(1, 2, Impulse.kr(0.25)), Impulse.kr(0.5)) < 1)
103         );
104
105         register_prob = 10;
106         // these sine tones are a very crud mockup of what might happen at orgelpark
107         sines = 6.collect({arg k; 4.collect({arg j; 4.collect({arg i;
108             SinOsc.ar((36 + [0, 4, 7, 11][j]).midicps * pow(2, i) * pow(2, k)) * (TRand.kr(0, register_prob,
                    hierarchical_dust) < 1).lag3 * 0.1
109         })})});
110         //change multiplier / last value for brown noise level
111         brown_noise = BrownNoise.ar() * (TIRand.kr(0, 2, hierarchical_dust) < 1) * 0.1;
112         //change multiplier / last value for white noise level
113         white_noise = WhiteNoise.ar() * (TIRand.kr(0, 2, hierarchical_dust) < 1) * 0.025;
114         Out.ar([0,1],
115             //Mix.ar(sines.flat)
116             Mix.ar([brown_noise, white_noise]);
117             //Mix.ar(sines.flat ++ [brown_noise, white_noise])
118         );
119         SendTrig.kr(hierarchical_dust, 0, register_prob);
120     }).play(s);
121 });
122 appEnvironment.pop;
123 )
124
125
126 /*
127 (
128 // just for testing the messages
129 var hyperOrganAddr = NetAddr( "127.0.0.1", NetAddr.langPort ); // for local testing
130 OSCdef.freeAll;
131 OSCdef(\test, {arg msg, time, addr, recvPort;
132 msg.postln;
133 }, '/M0/L1/D', hyperOrganAddr);
134 )
135 */
```