```
(
/*
This is the interface and score for [___south america___] miscellany.
In performance, text (words, phrases, numbers, dates, etc.) are read from a travel journal.
When reading, press the record button to bring the audio into the system.
All parameters in the GUI are free to manipulate.
Available at https://www.github.com/mwinter80/south_america_miscellany
*/


var length, recbuf, feedbackbuf, timebuf, envbuf, sinebus, grainbus, feedbackbus;

length = 60; recbuf = Buffer.alloc(s, s.sampleRate * length); feedbackbuf = Buffer.alloc(s, s.sampleRate * length);
timebuf = Buffer.alloc(s, 2000); envbuf = Buffer.loadCollection(s, Array.fill(1024, {1}));
sinebus = Bus.audio(s); grainbus = Bus.audio(s); feedbackbus = Bus.audio(s);

SynthDef(\rec,{
    var in, trig, step, time;
    in = SoundIn.ar(0); trig = KeyState.kr(65, 0, 1, 0);
    step = Stepper.kr(trig, 0, 0, 2000, 1); time = Integrator.kr(trig / s.sampleRate * s.options.blockSize).trunc(0.1);
    BufWr.kr(time, timebuf.bufnum, step);
    RecordBuf.ar(in, recbuf.bufnum, Latch.kr(time, trig), 1, 0, trig, 0, 0, trig);
    RecordBuf.ar(in, feedbackbuf.bufnum, Latch.kr(time, trig), 1, 0, trig, 0, 0, trig);
    SendTrig.kr(trig, 0, step);
}).add;

SynthDef(\play, { |num = 0 step = 0 min = 2 max = 5 grainprob = 0.1 fbprob = 0.1 delaytime = 0.02 fbc = 50, blur = 0|
    var trig, rand, pos, dur, env, blurenv, grain, sineout, grainout, fbtrig, feedbackout;
     trig = Impulse.kr(LocalIn.kr(1, 0.1).reciprocal);
    rand = TRand.kr(0, step, trig).trunc; pos = BufRd.kr(1, timebuf.bufnum, rand);
    dur = BufRd.kr(1, timebuf.bufnum, rand + 1) - pos;
    env = GrainBuf.ar(1, trig, dur, envbuf, 1, 0, 1, 0, envbuf);
    //blurenv = EnvGen.kr(Env.xyc([[0, 0, (blur - 1).abs * -500],
    //  [0.5, 1, (blur - 1).abs * 500], [1, 0, (blur - 1).abs * 500]]), env, timeScale: dur);
    grain = GrainBuf.ar(1, trig, dur, recbuf, 1, pos / 60, 1, 0, envbuf) * env.lag(blur * 0.1);
    sineout = SinOsc.ar(50, 1.0.rand) * (env - 1).abs.lag(blur * 0.5);
    grainout = grain * (TRand.kr(0, 1, trig) < Latch.kr(grainprob, trig));
    fbtrig = TRand.kr(0, 1, trig) < Latch.kr(fbprob, trig);
    feedbackout = CombC.ar(fbtrig * grain, 0.1, delaytime, fbc * env.lag(0, blur * 50))
        * (fbtrig * env).lag(0, blur * 50);
    Out.ar(grainbus, grainout); Out.ar(sinebus, sineout); Out.ar(feedbackbus, feedbackout);
    LocalOut.kr(TRand.kr(0, max - min, trig).trunc(0.1) + min + Select.kr(step > num + 1, [DC.kr(0.1), dur]));
    Poll.kr(trig, blur, \test);
}).add;


SynthDef(\out, {|recamp = 1 fbamp = 1 sinamp = 0.5|
```

```
    Out.ar([0, 1], (Clip.ar(In.ar(sinebus), -1, 1) * sinamp) + (In.ar(grainbus) * recamp) + (In.ar(feedbackbus) *
fbamp ));
}).add;

SynthDef(\bass, {
    Out.ar([0, 1], CombC.ar(SinOsc.ar(50) * KeyState.kr(56, 0, 1), 0.02, 0.02, 25))
}).add;
)

(
var bass, out, players, labels, sliders,  win, count = 0, record, saturate;

Synth(\rec);
bass = Synth(\bass);
out = Synth.tail(s, \out);
players = List.new();
sliders = List.new();

win = Window("[___South America___] miscellany", Rect(128, 64, 475, 240));
win.view.decorator=FlowLayout(win.view.bounds); win.view.decorator.gap=2@2;

labels = [" min grain gap  ", " max grain gap ", " grain prob ", " feedback prob ", " delay time ", " fbc ", " blur "];
labels.do({|l| sliders.add(EZSlider(win, 450@16, l, labelWidth: 110))});
sliders[0].controlSpec = ControlSpec(0, 2, \lin, 0, 2, ""); sliders[0].value = 2;
sliders[0].action = {|ez| if(ez.value > sliders[1].value, {sliders[1].valueAction =  ez.value});
    players.do({|i| i.set(\min, ez.value)})};
sliders[1].controlSpec = ControlSpec(0, 5, \lin, 0, 5, ""); sliders[1].value = 5;
sliders[1].action = {|ez| if(ez.value < sliders[0].value, {sliders[0].valueAction =  ez.value});
    players.do({|i| i.set(\max, ez.value)})};

sliders[2].controlSpec = ControlSpec(0, 1, \lin, 0, 0.1, ""); sliders[2].value = 0.1;
sliders[2].action = {|ez| players.do({|i| i.set(\grainprob, ez.value)})};
sliders[3].controlSpec = ControlSpec(0, 1, \lin, 0, 0.1, ""); sliders[3].value = 0.1;
sliders[3].action = {|ez| players.do({|i| i.set(\fbprob, ez.value)})};
sliders[4].controlSpec = ControlSpec(0.02, 0.03, \lin, 0, 0.02, ""); sliders[4].value = 0.02;
sliders[4].action = {|ez| (players.size() / 3).do({|i| players[i].set(\delaytime, ez.value)})};
sliders[5].controlSpec =  ControlSpec(0, 50, \lin, 0, 50, ""); sliders[5].value = 50;
sliders[5].action = {|ez| players.do({|i| i.set(\fbc, ez.value)})};
sliders[6].controlSpec =  ControlSpec(0, 1, \lin, 0, 0, ""); sliders[6].value = 0;
sliders[6].action = {|ez| players.do({|i| i.set(\blur, ez.value)})};


EZSlider(win, 450@16," grain amp  ", ControlSpec(0, 1, \lin, 0, 1, ""),
    {|ez| out.set(\recamp, ez.value)}, labelWidth: 110);
EZSlider(win, 450@16," sin amp   ", ControlSpec(0, 1, \lin, 0, 0.5, ""),
    {|ez| out.set(\sinamp, ez.value)}, labelWidth: 110);
```

```
EZSlider(win, 450@16," feedback amp  ", ControlSpec(0, 1, \lin, 0, 1, ""),
    {|ez| out.set(\fbamp, ez.value)}, labelWidth: 110);

record = Button(win, Rect(0, 0, 375, 50)).states_([["record", Color.black, Color.red]]);
saturate = Button(win, Rect(0, 0, 75, 50)).states_([["saturate", Color.black, Color.red]]);

OSCFunc({ arg msg, time;
    count = count + 1; players.add(Synth(\play, [\num, count,
        \min, sliders[0].value, \max, sliders[1].value, \grainprob, sliders[2].value, \fbprob, sliders[3].value,
        \delaytime, sliders[4].value, \fbc, sliders[5].value, \blur, sliders[6].value]));
    players.do({|i| i.set(\step, msg[3])});
},'/tr', s.addr);

win.front;
)
```