Matrix
for George Lewis

written for Rage Thormbones

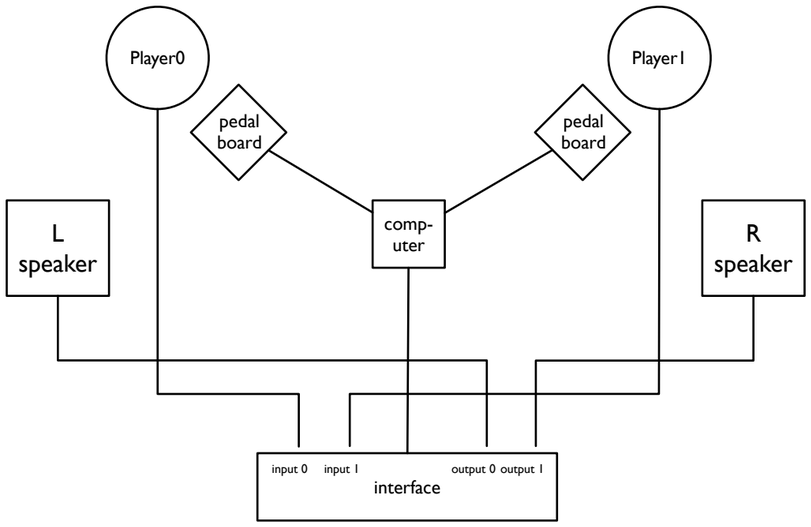
Sam Pluta

Matrix for George Lewis

Gear

- two trombones
- two Silent Brass Mutes, which both mute the direct signal and act as a microphone
- custom software environment written in SuperCollider
- two Line 6 FBV Express MKII USB pedal boxes or similar (two switch pedals and one expression pedal for each player
- audio interface with two inputs and two outputs (instrument in if possible)
- two speaker PA system

Setup



Instructions

General instructions and guidelines about how to play each section are provided. However, players should feel free explore the sonic environment of the software and play the piece as they like. The limitations of each section of the piece should provide ample guidance for the players. Suggested timings are simply that, suggested.

Pedal Markings

trigger - change occurs when the pedal is triggered
hold - change occurs when the pedal is triggered and goes back to its primary state when pedal is released

Section Pedal

Player0 - Pedal0 is the section pedal. Pressing the pedal moves on to the next section.

Processes

PopLip - onset triggers a short blip whose timbre is based on the pitch and spectral centroid of the incoming signal. Great for quiet popping lip sounds with variance in spectral content.
Splosions - onset triggers a low explosive tone whose whose timbre is based on the pitch and spectral centroid of the incoming signal. Works great with slap tongues and plosive attacks.
Noise|Buzz - uses two simultaneous processes: 1) a Phase Vocoder to timbrelly track and resynthesize noise and 2) a pitch tracker to resynthesize high harmonics of a fundamental tone. Works well with sustained sounds that change timbre from pure tones to full noise.
AmpFollowerGate - onset detection moves the module between percussive attacks, noise, and frequency modulation synthesis. Works best with a combination of attacks and sustains. It sometimes will not sustain a tone and it sometimes will, which may or may not be frustrating.
LPCError - uses Linear Predictive Coding analysis and resynthesis to create the tone. The volume pedal controls quality of the analysis. Lower values create more pops and clicks, with a significantly more lo-fi sound. Higher values are more true to source.
FMSynth - a frequency modulation circuit where Player0 is controlling the modulator and Player1 is controlling the carrier. In section 8, Player1 also controls the amplitude. The circuit uses pitch tracking to analyze the pitch of the players and applies this to the output. Changing settings maps the incoming frequencies to different ranges in the FM circuit.
Drone - a big low drone made of distorted sine waves. Circular breathing is suggested. Follows the pitch of the player. Will pitch shift any note above A3 down until it is below A3.

Post Processes

LoopBuf - in line with processes. Constantly records the output to an 8 second buffer. When the pedal is moved, LB interrupts the processed signal and plays back from a recorded 8 second buffer. Pedal position changes the speed of file playback. When the pedal does not move for 0.1 seconds, the live signal returns.

Grab - in line with processes and after LoopBuf. Grabs onto a 1 second buffer. Moving the pedal engages the effect and ceasing to move the pedal disengages the effect. Moving the pedal changes the size of the buffer that is looped. The lower the pedal value, the smaller the buffer played.

Use LoopBuf and Grab liberally. It is encouraged that the players focus exclusively on these processes at one or more points in the piece.

Spectral Freeze - Freezes the timbre and volume of the signal while the pedal is held.

Distort - Distorts the signal while the pedal is held.

Reverb - a big swampy reverb that the Splosions are sent through when the foot pedal is triggered. In sections 2-3.

Section 0

Player0
Pedal0 - starts the piece
Pedal1 - posts in the post window
Continuous - moves the fader

Player1
Pedal0 - posts in the post window
Pedal1 - posts in the post window
Continuous - moves the fader

Nothing is happening. Waiting to start the piece. The first click of Player0-Pedal0 will prepare the piece to start. The second click will start the piece.

Section 1 - 1 minute

Player0 - PopLip/Splosions
Pedal0 - advances to next section (trigger)
Pedal1 - switches to Splosions (hold)
Continuous - engages and controls LoopBuf

Player1 - PopLip/Splosions
Pedal0 - changes PopLip range between low and high (trigger)
Pedal1 - switches to Splosions (hold)
Continuous - engages and controls LoopBuf

Start sparse and increase in density. Short, popping lip sounds seem to work best. Intermittently add Splosions. The two sounds should have similiar volumes. Each player controls his/her own LoopBuf.

Section 2 - 30 seconds

Player0 - Splosions/VerbSplosions
Pedal0 - advances to next section (trigger)
Pedal1 - sends output to reverb
Continuous - engages and controls LoopBuf

Player1 - Splosions/VerbSplosions
Pedal0 - nil
Pedal1 - sends output to reverb
Continuous - engages and controls LoopBuf

This section only has Splosions as its sound. Focus on the contrast between dry and reverbed spaces.

Section 3 - 45 seconds

Player0 - Splosions/PopLip
Pedal0 - advances to next section (trigger)
Pedal1 - switches between the two processes (trigger)
Continuous - engages and controls LoopBuf

Player1- Splosions/PopLip
Pedal0 - changes PopLip range on random player (trigger)
Pedal1 - switches between the two processes (trigger)
Continuous - engages and controls LoopBuf

A nice mix of PopLip and Splosions.

Section 4 - 1 minute

Player0 - Noise|Buzz/AmpFollowerGate
Pedal0 - advances to next section (trigger)
Pedal1 - switches to AmpFollowerGate (trigger)
Continuous - engages and controls Grab

Player1 - Noise|Buzz/AmpFollowerGate
Pedal0 - nil
Pedal1 - switches to AmpFollowerGate (trigger)
Continuous - engages and controls LoopBuf

Noise|Buzz is looking for sustained and noisy sounds. AmpFollowerGate changes quality on each attack, thus is constantly changing its reaction.

Section 5 - 30 seconds

Player0 - AmpFollowerGate/LPCError
Pedal0 - advances to next section (trigger)
Pedal1 - switches between processes (trigger)
Continuous - during AmpFollowerGate, engages and controls Grab - during LPCError, changes analysis quality of LPCError

Player1 - AmpFollowerGate/LPCError
Pedal0 - nil
Pedal1 - switches between processes (trigger)
Continuous - during AmpFollowerGate, engages and controls LoopBuf - during LPCError, changes analysis quality of LPCError

AFGate functions as before. LPC trigger works well with sustained tones, especially with the pedal around the middle of its range. With the pedal at a lower setting, the effect gets more poppy and distorted, thus inviting more angular and noisy playing.

Section 6 - 1 minute

Player0 - Noise|Buzz/AmpFollowerGate
Pedal0 - advances to next section (trigger)
Pedal1 - switches between the two processes (trigger)
Continuous - engages and controls Grab
Player1 - Noise|Buzz/AmpFollowerGate
Pedal0 - nil
Pedal1 - switches between the two processes (trigger)
Continuous - engages and controls LoopBuf

Like section 4.

Section 7 - 1 minute or more

Player0 - LPCNoise/Spectral Freeze
Pedal0 - advances to next section (trigger)
Pedal1 - engages Spectral Freeze on both players (hold)
Continuous - changes analysis quality of LPCError
Player1 - LPCNoise /Distortion
Pedal0 - nil
Pedal1 - engages Distortion on both players (hold)
Continuous - changes analysis quality of LPCError

Focus on LPCError with Freeze and Distortion interruptions. Sustained tones may be best in this section.

Section 8 - 1.5 minutes

Player0 - FMSynth Carrier
Pedal0 - advances to next section (trigger)
Pedal1 - nil
Continuous - nil
Player1 - FMSynth Modulator
Player1 controls on/off of entire sound with amplitude
Pedal0 - sets the frequency range of the oscillators (trigger)
Pedal1 - nil
Continuous - nil

The output signal is a single frequency modulation circuit that is controlled by both players.

Player1 is in control of the amplitude of the circuit. This a simple on/off control. This section should focus on turning the sound on and off. Between each burst of sound, Player1 should press Pedal0 to change the frequency settings of the oscillators, giving each burst a unique timbre.

Section 9 - 2 minutes

Player0 - FMSynth Carrier
Pedal0 - advances to next section (trigger)
Pedal1 - engages Spectral Freeze on both players (hold)
Continuous - engages and controls Grab on both signals
Player1 - FMSynth Modulator
amplitude no longer controls on/off
Pedal0 - set both oscillator circuits to the same frequency ranges. The phases of the left and right channels are randomly in and out of phase.
Pedal1 - set the oscillator ranges independent (left and right channels will have different values and thus sound different)
Continuous - engages and controls LoopBuf

Like Section 8, but with no amplitude control. Grab and LoopBuf are added, as is Spectral Freeze and Distortion. Go nuts. Change values often. Loop and Grab all over the place.

Section 10 - 2 minutes

Player0 - PopLip
Pedal0 - ends the piece (trigger)
Pedal1 - changes the range of PopLip
Continuous - engages and controls LoopBuf
Player1 - Drone
Pedal0 - nil
Pedal1 - nil
Continuous - nil

Player1 creates a drone by playing a low sustained note below A3. This note should glissando slowly and change timbre. Player0 should wait about a minute before playing in this section. Player0 plays PopLip on top of the drone. The piece ends when Player0 presses Pedal0.

```
RageTrombones_Mod : Module_Mod {
	var mixerGroup, fxGroup0, fxGroup1, infoTexts, sectionTexts, currentSection, sectionSeq, sectionNames,
inBus0, inBus1, currentSynths, nextSynths, transferBus, verbBus, numSections, numItemsPerGroup;
	var rateBusses, lpcValBusses, loopBuf0, loopBuf1;
	var inVolBusses, volBusses;
	var freezeDist, grabBuf0, grabBuf1, grabBusses;
	var firmata, arduinoAddress, arduinoButton, clearButton, currentArduinoVals, arduinoDelays, arduinoTimes;
	var whichRange0, whichRange1, rangeSwitch, section346Switch0, section346Seq0, section346Switch1,
section346Seq1, section5Switch0, section5Seq0, section5Switch1, section5Seq1;

	*initClass {
		StartUp.add {
			SynthDef("loopMachine_tbn", {	arg inBus, outBus, recBuf, rateBus, gate = 1;
				var in, phasor, vol, env, smallGate, sound, smallEnv, inTrig, rate, phase, phaseStart,
impulse;

				inTrig = InTrig.kr(rateBus);

				impulse = Decay.kr(inTrig.abs, 0.1)>0;
				smallGate = 1-EnvGen.kr(Env.asr(0,1,0), impulse);

				phasor = Phasor.ar(0, BufRateScale.kr(recBuf)*smallGate, 0, BufFrames.kr(recBuf));

				in = In.ar(inBus);

				env = EnvGen.kr(Env.asr(0.02,1,0.02), gate, doneAction: 2);
				smallEnv = EnvGen.kr(Env.asr(0.02,1,0.02), smallGate);

				BufWr.ar(in*smallEnv, recBuf, phasor, loop:1);

				rate = In.kr(rateBus);
				phaseStart = Latch.kr(phasor, impulse);

				phase = (Phasor.ar(0.5-smallEnv, BufRateScale.kr(recBuf)*rate, 0, BufFrames.kr(recBuf),
phaseStart)).wrap(0, BufFrames.kr(recBuf));

				sound = BufRd.ar(1, recBuf, phase, loop:1)*(1-smallEnv);

				sound = (in*env*smallEnv)+sound;

				Out.ar(outBus, sound);
			}).writeDefFile;

			SynthDef("ampGate_tbn",{arg inBus, outBus, inVolBus, volBus, panPos, muteGate=0, gate=1;
				var sound, amp, synthGate, freq, hasFreq, osc, sim, filtersOsc, filtersNoise, amps, index,
modulator, chain, centroid, trig, gliss, thresh, noiseThresh, noiseGate;
				var env, muteEnv;

				env = EnvGen.kr(Env.asr(0,1,0), gate, doneAction:2);
				muteEnv = EnvGen.kr(Env.asr(0,1,0), muteGate, doneAction:0);

				sound = In.ar(inBus)*EnvGen.kr(Env.asr, 1)*In.kr(inVolBus)*muteEnv;

				thresh = 0.1;
				noiseThresh = thresh/32;

				amp = Amplitude.ar(sound, 0.007, 0.035);

				chain = FFT(LocalBuf(1024), sound);
				centroid = min(SpecCentroid.kr(chain), 4000);

				synthGate = LagUD.ar((amp>thresh), 0.007, 0.035);

				noiseGate = LagUD.ar((amp>noiseThresh), 0.007, 0.035);

				#freq, hasFreq = Pitch.kr(sound);

				amps = Amplitude.ar(BPF.ar(sound, Array.series(60, 100, 100), 50/Array.series(60, 100,
100)));

				filtersNoise = Mix(BPF.ar((BrownNoise.ar(1)+Dust.ar(amp*1000))*noiseGate,
Array.series(60, 100, 100), 100/Array.series(60, 100, 100), amps*32));

				index = centroid*2;

				modulator = SelectX.ar(Lag.kr(hasFreq,0.1), [filtersNoise*index*4, SinOsc.ar(freq*2, 0,
index)]);

				osc = SinOsc.ar(freq+modulator, 0.5, min(0.1, synthGate))+(filtersNoise/4);
				filtersOsc = Mix(BPF.ar(osc, Array.series(60, 100, 100), 100/Array.series(60, 100, 100),
amps*4));


				trig = Trig1.ar((amp>thresh),0.015);
				env = EnvGen.ar(Env.new([1,0,1], [0.015,0]), trig);
				gliss = Pulse.ar(env.linlin(0,1,TRand.ar(50, 450, trig), centroid*3), 0.5, Lag.ar(trig,
0.01)*0.2);

				gliss = LPF.ar(gliss, TRand.ar(200, 15000, trig));

				Out.ar(outBus, Pan2.ar(SelectX.ar(TChoose.kr(trig, [0,0.5,1]), [gliss, osc])*env,
panPos)*In.kr(volBus));
			}).writeDefFile;

			SynthDef("buzz_tbn", {arg inBus, outBus, inVolBus, volBus, panPos, muteGate=0, gate=1;
				var sound, onsets, chain, chain2, noise, centroid, out, smallEnv, sine, freq, hasFreq, amp,
chooseArray, array;
				var env, muteEnv;

				env = EnvGen.kr(Env.asr(0,1,0), gate, doneAction:2);
				muteEnv = EnvGen.kr(Env.asr(0,1,0), muteGate, doneAction:0);

				sound = In.ar(inBus)*EnvGen.kr(Env.asr, 1)*In.kr(inVolBus)*muteEnv;
```

```
		#freq, hasFreq = Pitch.kr(sound);

		amp = Amplitude.ar(sound, 0.007, 0.035);

		array = [15, 21, 27, 33, 39];
		chooseArray = [
			TChoose.kr(hasFreq, array),
			TChoose.kr(hasFreq, array+1),
			TChoose.kr(hasFreq, array+2),
			TChoose.kr(hasFreq, array+3),
			TChoose.kr(hasFreq, array+4),
			TChoose.kr(hasFreq, array+5)];

		sine = SinOsc.ar(freq*chooseArray/2, 0, amp*8).softclip/32;

		Out.ar(outBus, Pan2.ar(Mix(sine)*env, panPos));
	}).writeDefFile;

	SynthDef("harshNoise_tbn", {arg inBus, outBus, inVolBus, volBus, panPos, muteGate=0, gate=1;
		var sound, amp, synthGate, freq, hasFreq, osc, sim, filtersOsc, filtersNoise, amps, index,
modulator, chain, centroid, trig, gliss, thresh, noiseThresh, noiseGate, noise, bpfArray, arraySize=30;
		var env, muteEnv;

		env = EnvGen.kr(Env.asr(0,1,0), gate, doneAction:2);
		muteEnv = EnvGen.kr(Env.asr(0,1,0), muteGate, doneAction:0);

		thresh = 0.1;
		noiseThresh = thresh/8;

		sound = In.ar(inBus)*EnvGen.kr(Env.asr, 1)*In.kr(inVolBus)*muteEnv;

		amp = Amplitude.ar(sound, 0.007, 0.035);

		chain = FFT(LocalBuf(1024), sound);
		centroid = max(arraySize, min(SpecCentroid.kr(chain), 4000));

		noiseGate = LagUD.ar((amp>noiseThresh), 0.007, 0.035);

		#freq, hasFreq = Pitch.kr(sound);

		freq = Lag.kr(max(80, min(freq, 18000/arraySize)));

		bpfArray = Array.series(arraySize, 1, 1)*freq;

		bpfArray[arraySize-1];

		amps = Amplitude.ar(BPF.ar(sound, bpfArray, 50/bpfArray), 0.05, 0.05);

		freq = freq*(Peak.kr(freq, Impulse.kr(10)).linlin(80, 400, 0.5, 3));

		noise = BrownNoise.ar(1!arraySize)*noiseGate;

		filtersNoise = BPF.ar((noise), bpfArray, 50/bpfArray, amps*32);
		filtersNoise = Latch.ar(filtersNoise, Impulse.ar(Array.fill(arraySize, {rrand(800,
1200)}))).softclip;

		filtersNoise = RHPF.ar(Mix(filtersNoise), freq, 0.2);
		filtersNoise = LPF.ar(filtersNoise, freq*SinOsc.ar(LFNoise2.kr(0.2).range(0.05,
0.1)).range(4,12));

		filtersNoise = Limiter.ar(filtersNoise*2, 0.5)*In.kr(volBus)*env;

		Out.ar(outBus, Pan2.ar(filtersNoise, panPos));
	}).writeDefFile;

	SynthDef("splosions_tbn", {arg inBus, outBus, panPos, verbBus, verbSwitch=0, inVolBus, volBus,
muteGate=0, gate=1;
		var sound, trig, onsets, chain, chain2, noise, centroid, out, smallEnv, sine, sine0, sine1,
sineB, ramVal, decayTime, gliss, verb;
		var env, muteEnv;

		env = EnvGen.kr(Env.asr(0,1,0), gate, doneAction:2);
		muteEnv = EnvGen.kr(Env.asr(0,1,0), muteGate, doneAction:0);

		sound = In.ar(inBus)*EnvGen.kr(Env.asr, 1)*In.kr(inVolBus)*muteEnv;

		chain = FFT(LocalBuf(512), sound);

		chain2 = PV_Copy(chain, LocalBuf(512));

		centroid = max(400, min(SpecCentroid.kr(chain2), 10000));

		onsets = Onsets.kr(chain, 0.4, \power);
		trig = Trig1.kr(onsets, 0.005);

		ramVal = max(20, (19400 - Peak.kr(20000-centroid, trig))/2);

		sine = SinOsc.ar(ramVal, 0, 20).clip2(0.5).softclip;
		sineB = SinOsc.ar(ramVal+TRand.kr(5,10, trig), 0, 20).clip2(0.5).softclip;

		decayTime = TRand.kr(2,4, trig);

		smallEnv = EnvGen.kr(Env.perc(0.05, decayTime), trig);

		sine = [sine, sineB];

		sine = MoogFF.ar((1-Trig1.ar(Dust.kr(TRand.kr(50, 100, trig)),
LFNoise0.kr(LFNoise0.kr(1).range(3,7)).range(0.01, 0.0001)))*sine, Decay.kr(trig, decayTime).linexp(0,1,ramVal,
LFNoise2.kr(1).range(1500, 5000)));

		sine0 = Latch.ar(sine[0]*0.5, Impulse.ar(TRand.kr(200, 3000, trig))).clip2(0.8);
		sine1 = Latch.ar(sine[1]*0.5, Impulse.ar(TRand.kr(200, 3000, trig))).clip2(0.8);

		gliss = LFTri.ar(EnvGen.kr(Env.linen(0,0,0.07), trig)*TRand.kr(80, 120,
trig)+TRand.kr(20, 33, trig), 0, 1)*EnvGen.kr(Env.perc(0.01, 0.1), trig);
```

```
		sine = (sine0+sine1)+gliss;

		out = (sine*smallEnv).clip2(1);

		out = Limiter.ar(out, 1, 0.001)*In.kr(volBus)*env;

		Out.ar(outBus, Pan2.ar(out, panPos));
		Out.ar(verbBus, out*env*Lag.kr(verbSwitch));
	}).writeDefFile;

	SynthDef("popLip_tbn", {arg inBus, outBus, panPos, panChoose, inVolBus, volBus, rangeSwitch =
0, muteGate=0, gate=1;
		var sound, onsets, chain, chain2, noise, centroid, out, smallEnv, sine;
		var env, muteEnv;

		env = EnvGen.kr(Env.asr(0,1,0), gate, doneAction:2);
		muteEnv = EnvGen.kr(Env.asr(0,1,0), muteGate, doneAction:0);

		sound = In.ar(inBus)*In.kr(inVolBus)*muteEnv;

		chain = FFT(LocalBuf(512), sound);

		chain2 = PV_Copy(chain, LocalBuf(512));

		centroid = max(400, min(SpecCentroid.kr(chain2), 10000));

		onsets = Onsets.kr(chain, 0.2, \power);

		noise = WhiteNoise.ar(1);

		noise = BPF.ar(noise, centroid*2, TRand.kr(0.01, 0.05, onsets));

		centroid = Select.kr(rangeSwitch>0.5, [centroid/2, centroid*2]) + (noise*2);

		sine = SelectX.ar(TRand.kr(0,1,onsets), [SinOsc.ar(centroid), LFTri.ar(centroid)]);

		smallEnv = Decay.kr(onsets, LFNoise2.kr(4).range(0.01, 0.05));

		out = (noise*sine*smallEnv*4+0.2).clip2(0.5)-0.2/2;

		out = Pan2.ar(out, Select.kr(panChoose, [panPos, LFNoise2.kr(3).range(-0.9,0.9)]));

		Out.ar(outBus, out*env*In.kr(volBus));
}).writeDefFile;

SynthDef("drone_tbn", {arg inBus, outBus, topFreq=120, inVolBus, volBus, muteGate=0, gate=1;
		var sound, lpSound, freq, hasFreq, amp, sins, sins2, distSins, delayTime;
		var env, muteEnv;

		env = EnvGen.kr(Env.asr(0,1,0), gate, doneAction:2);
		muteEnv = EnvGen.kr(Env.asr(0,1,0), muteGate, doneAction:0);

		sound = In.ar(inBus)*In.kr(inVolBus)*muteEnv;

		amp = Amplitude.kr(sound, 0.1, 0.1);

		lpSound = LPF.ar(sound, 200);
		freq = 100;
		#freq, hasFreq = Pitch.kr(lpSound);

		freq = Select.kr(freq<topFreq, [freq/2, freq]);
		freq = Select.kr(freq<topFreq, [freq/2, freq]);
		freq = Select.kr(freq<topFreq, [freq/2, freq]);
		freq = Select.kr(freq<topFreq, [freq/2, freq]);

		sins = (Pan2.ar(SinOsc.ar(freq, 0, 2), SinOsc.ar(0.19))+Pan2.ar(SinOsc.ar(freq+1, 0, 2),
SinOsc.ar(0.22, pi)))*amp;

		delayTime = (SinOsc.kr(0.4, 0.05, 0.01)) + 0.15;

		distSins = (DelayC.ar(sins, 0.25, delayTime)+sins*8).clip2(0.5);

		sins2 = (Pan2.ar(SinOsc.ar(freq/2, 0, 2), SinOsc.ar(0.2))+Pan2.ar(SinOsc.ar(freq+1/2, 0,
2), SinOsc.ar(0.21, pi)))*amp;

		Out.ar(outBus, (sins+distSins+sins2)*In.kr(volBus)*env);
	}).writeDefFile;


	SynthDef("LPCError_tbn", {arg inBus, outBus, lpcValBus, panPos, inVolBus, volBus, muteGate=0,
gate=1;
		var sound, sound1, freq, hasFreq, out;
		var env, muteEnv;

		env = EnvGen.kr(Env.asr(0,1,0), gate, doneAction:2);
		muteEnv = EnvGen.kr(Env.asr(0,1,0), muteGate, doneAction:0);

		sound = In.ar(inBus)*In.kr(inVolBus)*muteEnv;

		#freq, hasFreq = Pitch.kr(sound);

		out = LPCError.ar(sound, In.kr(lpcValBus)).clip2(0.4);

		out = LPF.ar(Limiter.ar(out*2, 0.5), 10000);

		Out.ar(outBus, Pan2.ar(out, panPos)*env*In.kr(volBus));
	}).writeDefFile;

	SynthDef("fmSynth_tbn", {arg inBus0, inBus1, inVolBus, volBus, outBus, whichRange0=2,
whichRange1=0, whichRange2=2, whichRange3=0, envOrNo = 0, /*thresh = 0.05, */muteGate=0, gate=1, dupOrNo=0;
		var sound, sound1, sine0, sine1, freq, hasFreq, modulator, modulator2, modulatorFreq,
hasModFreq, amp, amp1, chooseArray, array, impulse, smallGate, env, muteEnv, thresh;

		env = EnvGen.kr(Env.asr(0,1,0), gate, doneAction:2);
		muteEnv = EnvGen.kr(Env.asr(0,1,0), muteGate, doneAction:0);

		sound = In.ar(inBus0)*muteEnv;
```

```
		sound1 = In.ar(inBus1)*muteEnv;

		#freq, hasFreq = Pitch.kr(sound);
		amp = Amplitude.kr(sound1, 0.1, 0.1); //player1 controls volume

		thresh = In.kr(inVolBus);

		impulse = Decay.kr(amp>thresh, 0.1)>0;
		smallGate = Select.kr(envOrNo, [EnvGen.kr(Env.asr(0,1,0), impulse), 1]);

		#modulatorFreq, hasModFreq = Pitch.kr(sound1);

		modulator = SinOsc.ar(modulatorFreq.linlin(50, 900, 0.5, Select.kr(whichRange0, [20, 200, 2000, 10000])));
		modulator2 = SinOsc.ar(modulatorFreq.linlin(50, 900, 0.5, Select.kr(whichRange2, [20, 200, 2000, 10000])));

		sine0 = LPF.ar(SinOsc.ar(freq.linlin(50, 900, 200, Select.kr(whichRange1, [20, 200, 2000,10000]))+(modulator*3000)).softclip, 16000);

		sine1 = SelectX.ar(Lag.kr(dupOrNo,0.01), [sine0, LPF.ar(SinOsc.ar(freq.linlin(50, 900, 200, Select.kr(whichRange3, [20, 200, 2000,10000]))+(modulator2*3000)).softclip, 16000)]);

		Out.ar(outBus, [sine0,sine1]*smallGate*env*In.kr(volBus));
	}).writeDefFile;

	SynthDef("verb_tbn", {arg verbBus, outBus, volBus, whichSound, whiteNoiseVol, whichFilterFreq, startVal, endVal, dur, lfoFreq = 1, lpfRQStart=1, lpfRQEnd=1, rqDur=1, muteGate=0, gate=1;
		var sound, env, muteEnv, lpfFreq, hpfFreq, lpfRQ;

		sound = SelectX.ar(Lag.kr(whichSound, 3), [In.ar(verbBus), WhiteNoise.ar(whiteNoiseVol)])*In.kr(volBus);

		lpfFreq = Select.kr(whichFilterFreq, [
			XLine.kr(startVal, endVal, dur),
			SinOsc.kr(lfoFreq).range(100, 10000)]);
		hpfFreq = Select.kr(whichFilterFreq, [20, lpfFreq/2]);

		lpfRQ = Line.kr(lpfRQStart, lpfRQEnd, rqDur);

		sound = RLPF.ar(GVerb.ar(sound, 200, 4, 1, 1, 15, 0, 0.7, 1), lpfFreq, lpfRQ);
		sound = HPF.ar(sound, hpfFreq);

		Out.ar(outBus, sound);
	}).writeDefFile;

	SynthDef("loopMachine_tbn", {	arg fxBus, recBuf, rateBus, gate = 1;
		var in, phasor, vol, env, smallGate, sound, smallEnv, inTrig, rate, phase, phaseStart, impulse;

		inTrig = InTrig.kr(rateBus);

		impulse = Decay.kr(inTrig.abs, 0.1)>0;
		smallGate = 1-EnvGen.kr(Env.asr(0,1,0), impulse);

		phasor = Phasor.ar(0, BufRateScale.kr(recBuf)*smallGate, 0, BufFrames.kr(recBuf));

		in = In.ar(fxBus);

		env = EnvGen.kr(Env.asr(0.02,1,0.02), gate, doneAction: 2);
		smallEnv = EnvGen.kr(Env.asr(0.02,1,0.02), smallGate);

		BufWr.ar(in*smallEnv, recBuf, phasor, loop:1);

		rate = In.kr(rateBus);
		phaseStart = Latch.kr(phasor, impulse);

		phase = (Phasor.ar(0.5-smallEnv, BufRateScale.kr(recBuf)*rate, 0, BufFrames.kr(recBuf), phaseStart)).wrap(0, BufFrames.kr(recBuf));

		sound = BufRd.ar(1, recBuf, phase, loop:1)*(1-smallEnv);

		sound = (in*env*smallEnv)+sound;

		ReplaceOut.ar(fxBus, sound);
	}).writeDefFile;

	SynthDef("freezeDist_tbn", { arg fxBus, freeze=0, dist = 0;
		var in, chain0, chain1, out, out0, out1;

		in = In.ar(fxBus, 2);

		chain0 = FFT(LocalBuf(2048), in[0]);
		chain0 = PV_Freeze(chain0, freeze);
		out0 = IFFT(chain0);

		chain1 = FFT(LocalBuf(2048), in[1]);
		chain1 = PV_Freeze(chain1, freeze);
		out1 = IFFT(chain1);
		out = [out0,out1];
		out = SelectX.ar(Lag.kr(dist),[out, (out*8).fold2(0.5)]);

		ReplaceOut.ar(fxBus, out);
	}).writeDefFile;

	SynthDef("grab_tbn", {arg fxBus, grabBuf, grabBus, gate = 1;
		var in, phasor, vol, env, smallGate, sound, smallEnv, inTrig, grabSize, phase, phasePos, impulseRate, impulse, center;

		inTrig = InTrig.kr(grabBus);

		impulse = Decay.kr(inTrig.abs, 0.1)>0;
		smallGate = (1-EnvGen.kr(Env.asr(0,1,0), impulse));

		phasor = Phasor.ar(0, BufRateScale.kr(grabBuf)*smallGate, 0, BufFrames.kr(grabBuf));
```

```
                        in = In.ar(fxBus);

                        env = EnvGen.kr(Env.asr(0.02,1,0.02), gate, doneAction: 2);
                        smallEnv = EnvGen.kr(Env.asr(0.02,1,0.02), smallGate);

                        BufWr.ar(in*smallEnv, grabBuf, phasor, loop:1);

                        grabSize = In.kr(grabBus).linlin(0,1,0.01,0.5);
                        phasePos = Latch.kr(phasor, impulse);

                        center = (phasePos/BufSampleRate.kr(grabBuf)-grabSize).wrap(0,1);
                        impulseRate = Select.kr(smallEnv, [(1/grabSize), 0]);
                        sound = Mix(TGrains2.ar(2, Impulse.kr(impulseRate), grabBuf, 1, center, grabSize, 0, 1,
0, 0));


                        sound = (in*env*smallEnv)+sound;

                        ReplaceOut.ar(fxBus, sound);

                }).writeDefFile;

                SynthDef("toOut_tbn", {arg fxBus, outBus;
                        Out.ar(outBus, In.ar(fxBus,2));
                }).writeDefFile;
        }
}

nextSectionChangeSynths {
        //clear the currentSynths
        if(currentSynths.size>0, {currentSynths.flatten.do{arg item; item.set(\gate, 0)}});
        currentSynths = List.newClear(0);
        //make the next the current
        currentSynths.addAll(nextSynths);
        //add the nextSynths
        nextSynths = List.newClear(0);
        switch(currentSection,
                0, {
                        //end of piece - beginning of piece
                        //synths for player0
                        nextSynths.add([
                                Synth("popLip_tbn", [\inBus, inBus0, \outBus, transferBus, \panPos, -1,
\panChoose, 0, \inVolBus, inVolBusses[0][0], \volBus, volBusses[0][0]], fxGroup0),
                                Synth("splosions_tbn", [\inBus, inBus0, \outBus, transferBus, \panPos, -1,
\verbBus, verbBus, \verbSwitch, 0, \inVolBus, inVolBusses[0][1], \volBus, volBusses[0][1]], fxGroup0)]);

                        //synths for player1
                        nextSynths.add([
                                Synth("popLip_tbn", [\inBus, inBus1, \outBus, transferBus, \panPos, 1,
\panChoose, 0, \inVolBus, inVolBusses[0][0], \volBus, volBusses[0][0]], fxGroup0),
                                Synth("splosions_tbn", [\inBus, inBus1, \outBus, transferBus, \panPos, 1,
\verbBus, verbBus, \verbSwitch, 0, \inVolBus, inVolBusses[0][1], \volBus, volBusses[0][1]], fxGroup0)]);

                        nextSynths.add(Synth("verb_tbn", [\verbBus, verbBus, \outBus, outBus, \volBus,
volBusses[0][2], \whichSound, 0, \whiteNoiseVol, 0, \whichFilterFreq, 0, \startVal, 2000, \endVal, 2000, \dur, 1],
fxGroup1));
                },
                1, {
                        nextSynths.add([
                                Synth("harshNoise_tbn", [\inBus, inBus0, \outBus, transferBus, \inVolBus,
inVolBusses[1][0], \volBus, volBusses[1][0], \panPos, -1], fxGroup0),
                                Synth("buzz_tbn", [\inBus, inBus0, \outBus, transferBus, \inVolBus,
inVolBusses[1][1], \volBus, volBusses[1][1], \panPos, -1], fxGroup0),
                                Synth("ampGate_tbn", [\inBus, inBus0, \outBus, transferBus, \panPos, -1,
\inVolBus, inVolBusses[1][2], \volBus, volBusses[1][2]], fxGroup0),
                                Synth("LPCError_tbn", [\inBus, inBus0, \outBus, transferBus, \lpcValBus,
lpcValBusses[0], \panPos, -1, \inVolBus, volBusses[1][3], \volBus, volBusses[1][3]], fxGroup0)
                        ]);

                        nextSynths.add([
                                //inBus, outBus, inVolBus, volBus
                                Synth("harshNoise_tbn", [\inBus, inBus1, \outBus, transferBus, \inVolBus,
inVolBusses[1][0], \volBus, volBusses[1][0], \panPos, 1], fxGroup0),
                                Synth("buzz_tbn", [\inBus, inBus1, \outBus, transferBus, \inVolBus,
inVolBusses[1][1], \volBus, volBusses[1][1], \panPos, 1], fxGroup0),
                                Synth("ampGate_tbn", [\inBus, inBus1, \outBus, transferBus, \panPos, 1,
\inVolBus, inVolBusses[1][2], \volBus, volBusses[1][2]], fxGroup0),
                                Synth("LPCError_tbn", [\inBus, inBus1, \outBus, transferBus, \lpcValBus,
lpcValBusses[1], \panPos, 1, \inVolBus, volBusses[1][3], \volBus, volBusses[1][3]], fxGroup0)
                        ]);

                        currentSynths[0][0].set(\muteGate, 1);
                        currentSynths[0][1].set(\muteGate, 0);
                        currentSynths[1][0].set(\muteGate, 1);
                        currentSynths[1][1].set(\muteGate, 0);
                },
                4, {

                        nextSynths.add(Synth("fmSynth_tbn", [\inBus0, inBus0, \inBus1, inBus1, \inVolBus,
inVolBusses[2][0], \volBus, volBusses[2][0], \outBus, transferBus], fxGroup0));
                        currentSynths[0][0].set(\muteGate, 1);
                        currentSynths[0][1].set(\muteGate, 1);
                        currentSynths[1][0].set(\muteGate, 1);
                        currentSynths[1][1].set(\muteGate, 1);
                },
                8, {
                        nextSynths.add(Synth("drone_tbn", [\inBus, inBus1, \outBus, outBus, \inVolBus,
inVolBusses[3][0], \volBus, volBusses[3][0]], fxGroup0));
                        nextSynths.add(Synth("popLip_tbn", [\inBus, inBus0, \outBus, transferBus, \panPos, -1,
\panChoose, 1, \inVolBus, inVolBusses[3][1], \volBus, volBusses[3][1]], fxGroup0));
                        currentSynths[0].set(\muteGate, 1);
                },
                10, {
                        currentSynths.do{arg item; item.set(\muteGate, 1)};
                },
                nil
```

```
		)
}

buttonPressed {arg vals;
		//receives button number and value
		//section switches
		switch(vals[0],
			0/*player0*/, {
				if(vals[1]==1,{
					//go to the next section
					currentSection = sectionSeq.next;
					sectionTexts.do{arg item; item.string = currentSection};
					2.do{arg i; infoTexts[i].string_(sectionNames[currentSection][i])};
					switch(currentSection,
						0, {
							this.nextSectionChangeSynths;
						},
						1, {
							this.nextSectionChangeSynths;
						},
						2, {
							//swap the synths
							currentSynths[0][0].set(\muteGate, 0);
							currentSynths[0][1].set(\muteGate, 1);
							currentSynths[1][0].set(\muteGate, 0);
							currentSynths[1][1].set(\muteGate, 1);
						},
3, {
		if(section346Switch0 == 1, {
			section346Switch0 = section346Seq0.next;
		});
		if(section346Switch1 == 1, {
			section346Switch1 = section346Seq1.next;
		});
},
4, {
		if(section346Switch0 == 1, {
			section346Switch0 = section346Seq0.next;
		});
		if(section346Switch1 == 1, {
			section346Switch1 = section346Seq1.next;
		});
		this.nextSectionChangeSynths;
},
5, {
		currentSynths[0][0].set(\muteGate, 0);
		currentSynths[0][1].set(\muteGate, 0);
		currentSynths[0][2].set(\muteGate, 1);

		currentSynths[1][0].set(\muteGate, 0);
		currentSynths[1][1].set(\muteGate, 0);
		currentSynths[1][2].set(\muteGate, 1);
},
6, {
		if(section346Switch0 == 1, {
			section346Switch0 = section346Seq0.next;
		});
		if(section346Switch1 == 1, {
			section346Switch1 = section346Seq1.next;
		});

		currentSynths[0][0].set(\muteGate, 1);
		currentSynths[0][1].set(\muteGate, 1);
		currentSynths[0][2].set(\muteGate, 0);
		currentSynths[0][3].set(\muteGate, 0);

		currentSynths[1][0].set(\muteGate, 1);
		currentSynths[1][1].set(\muteGate, 1);
		currentSynths[1][2].set(\muteGate, 0);
		currentSynths[1][3].set(\muteGate, 0);
},
7, {
		currentSynths[0][0].set(\muteGate, 0);
		currentSynths[0][1].set(\muteGate, 0);
		currentSynths[0][2].set(\muteGate, 0);
		currentSynths[0][3].set(\muteGate, 1);

		currentSynths[1][0].set(\muteGate, 0);
		currentSynths[1][1].set(\muteGate, 0);
		currentSynths[1][2].set(\muteGate, 0);
		currentSynths[1][3].set(\muteGate, 1);
	},
	8, {
		freezeDist.set(\freeze, 0, \dist, 0);
		this.nextSectionChangeSynths;
	},
	9, {
		currentSynths[0].set(\envOrNo, 1);
	},
	10, {
		freezeDist.set(\freeze, 0, \dist, 0);
		this.nextSectionChangeSynths;
	},
	nil);

	})
},

1/*player0*/, {
	switch(currentSection,
		0, {
			"player0 - button 2".postln;
		},
		1, {
			if(vals[1]==0,{
				currentSynths[0][1].set(\muteGate, 0);
```

```
                currentSynths[0][0].set(\muteGate, 1);
        },{
                currentSynths[0][0].set(\muteGate, 0);
                currentSynths[0][1].set(\muteGate, 1);
        })
},
2, {
        currentSynths[0][1].set(\verbSwitch, vals[1]);
},
3, {
        if(vals[1]==1,{
                section346Switch0 = section346Seq0.next;
                if(section346Switch0==0,{
                        currentSynths[0][0].set(\muteGate, 0);
                        currentSynths[0][1].set(\muteGate, 1);
                },{
                        currentSynths[0][1].set(\muteGate, 0);
                        currentSynths[0][0].set(\muteGate, 1);
                })
        })
},
4, {//new state
        if(vals[1]==1,{
                section346Switch0 = section346Seq0.next;
                if(section346Switch0==0,{
                        currentSynths[0][0].set(\muteGate, 1);
                        currentSynths[0][1].set(\muteGate, 1);
                        currentSynths[0][2].set(\muteGate, 0);
                },{
                        currentSynths[0][0].set(\muteGate, 0);
                        currentSynths[0][1].set(\muteGate, 0);
                        currentSynths[0][2].set(\muteGate, 1);
                })
        })
},
5, {
        if(vals[1]==1,{
                section5Switch0 = section5Seq0.next;
                if(section5Switch0==0,{
                        currentSynths[0][2].set(\muteGate, 1);
                        currentSynths[0][3].set(\muteGate, 0);
                },{
                        currentSynths[0][2].set(\muteGate, 0);
                        currentSynths[0][3].set(\muteGate, 1);
                })
        });
},
6, {
        if(vals[1]==1,{
                section346Switch0 = section346Seq0.next;
                if(section346Switch0==0,{
                        currentSynths[0][0].set(\muteGate, 1);
                        currentSynths[0][1].set(\muteGate, 1);
                        currentSynths[0][2].set(\muteGate, 0);
                },{
                        currentSynths[0][0].set(\muteGate, 0);
                        currentSynths[0][1].set(\muteGate, 0);
                        currentSynths[0][2].set(\muteGate, 1);
                })
        })
},
7, {
        freezeDist.set(\freeze, vals[1]);
},
8, {
        //freezeDist.set(\freeze, vals[1]);
},
9, {
        freezeDist.set(\freeze, vals[1]);
},
10, {
        currentSynths[1].set(\rangeSwitch, rangeSwitch.next)
}, nil)
},
2/*player1*/, {
        "player1 - button 1".postln;
        switch(currentSection,
                0, {

                },
                1, {
                        if(vals[1]==1,{
                                var range = rangeSwitch.next;
                                2.do{arg i; currentSynths[i][0].set(\rangeSwitch, range)};
                        });/*,{
                        currentSynths[0][0].set(\rangeSwitch, 0);
                        currentSynths[1][0].set(\rangeSwitch, 0);
                        })*/
                },
                2, {

                },
                3, {
                        if(vals[1]==1,{
                                var range = rangeSwitch.next;
                                2.do{arg i; currentSynths[i][0].set(\rangeSwitch, range)};
                        });
                },
                4, {

                },
                5, {

                },
                6, {
```

```
                                        },
                                        7, {

                                        },
                                        8, {
                                                if(vals[1]==1,{
                                                        var temp0, temp1;
                                                        temp0 = whichRange0.next;
                                                        temp1 = whichRange1.next;
                                                        currentSynths[0].set(\whichRange0, temp0, \whichRange1,
temp1, \whichRange2, temp0, \whichRange3, temp1);
                                                });
                                        },
                                        9, {
                                                if(vals[1]==1,{
                                                        var temp0, temp1;
                                                        temp0 = whichRange0.next;
                                                        temp1 = whichRange1.next;
                                                        currentSynths[0].set(\dupOrNo, [0,1].choose, \whichRange0,
temp0, \whichRange1, temp1, \whichRange2, temp0, \whichRange3, temp1);
                                                });

                                        }, nil)
                                },
                                3/*player1*/, {
                                        switch(currentSection,
                                                0, {
        "player1 - button 2".postln;
},
1, {
        if(vals[1]==0,{
                currentSynths[1][0].set(\muteGate, 1);
                currentSynths[1][1].set(\muteGate, 0);
        },{
                currentSynths[1][0].set(\muteGate, 0);
                currentSynths[1][1].set(\muteGate, 1);
        })
},
2, {
        currentSynths[1][1].set(\verbSwitch, vals[1]);
},
3, {
        if(vals[1]==1,{
                section346Switch1 = section346Seq1.next;
                if(section346Switch1==0,{
                        currentSynths[1][0].set(\muteGate, 0);
                        currentSynths[1][1].set(\muteGate, 1);
                },{
                        currentSynths[1][1].set(\muteGate, 0);
                        currentSynths[1][0].set(\muteGate, 1);
                })
        })
},
4, {//new state
        if(vals[1]==1,{
                section346Switch1 = section346Seq1.next;
                if(section346Switch1==0,{
                        currentSynths[1][0].set(\muteGate, 1);
                        currentSynths[1][1].set(\muteGate, 1);
                        currentSynths[1][2].set(\muteGate, 0);
                },{
                        currentSynths[1][0].set(\muteGate, 0);
                        currentSynths[1][1].set(\muteGate, 0);
                        currentSynths[1][2].set(\muteGate, 1);
                })
        })
},
5, {
        if(vals[1]==1,{
                section5Switch1 = section5Seq1.next;
                if(section5Switch1==0,{
                        currentSynths[1][2].set(\muteGate, 1);
                        currentSynths[1][3].set(\muteGate, 0);
                },{
                        currentSynths[1][2].set(\muteGate, 0);
                        currentSynths[1][3].set(\muteGate, 1);
                })
        });
},
6, {
                if(vals[1]==1,{
                        section346Switch1 = section346Seq1.next;
                        if(section346Switch1==0,{
                                currentSynths[1][0].set(\muteGate, 1);
                                currentSynths[1][1].set(\muteGate, 1);
                                currentSynths[1][2].set(\muteGate, 0);
                        },{
                                currentSynths[1][0].set(\muteGate, 0);
                                currentSynths[1][1].set(\muteGate, 0);
                                currentSynths[1][2].set(\muteGate, 1);
                        })
                })
        },
        7, {
                freezeDist.set(\dist, vals[1]);
        },
        8, {
                //currentSynths[0].set(\whichRange0, whichRange0.next, \whichRange1,
whichRange1.next);
        },
        9, {
                if(vals[1]==1,{
                        currentSynths[0].set(\dupOrNo, 1, \whichRange0,
whichRange0.next, \whichRange1, whichRange1.next, \whichRange2, whichRange0.next, \whichRange3, whichRange1.next);
                });
        },
```

```
                                        10, {

                                        }, nil)
                                }, nil
                        );
}

sliderMoved {arg vals;
                switch(currentSection,
                        0, {},
                        1, {
                                rateBusses[vals[0]].set(vals[1].linlin(0,1,-4,4));
                        },
                        2, {
                                rateBusses[vals[0]].set(vals[1].linlin(0,1,-4,4));
                        },
                        3, {
                                rateBusses[vals[0]].set(vals[1].linlin(0,1,-4,4));
                        },
                        4, {
                                if(vals[0]==0,{
                                        grabBusses.do{arg item; item.set(vals[1])}
                                },{
                                        rateBusses.do{arg item; item.set(vals[1].linlin(0,1,-4,4))};
                                });
                        },
                        5, {
                        if(vals[0]==0,{
                                if(section5Switch0==0, {
                                        grabBusses.do{arg item; item.set(vals[1])}
                                },{
                                        lpcValBusses[vals[0]].set(vals[1].linlin(0,1,2,45))
                                })
                        },{
                                if(section5Switch1==0, {
                                        rateBusses.do{arg item; item.set(vals[1].linlin(0,1,-4,4))};
                                },{
                                        lpcValBusses[vals[0]].set(vals[1].linlin(0,1,2,45))
                                })
                        })
                },
                6, {
                        if(vals[0]==0,{
                                grabBusses.do{arg item; item.set(vals[1])}
                        },{
                                rateBusses.do{arg item; item.set(vals[1].linlin(0,1,-4,4))};
                        });
                },
                7, {
                        lpcValBusses[vals[0]].set(vals[1].linlin(0,1,2,45))
                },
                8, {
                        /*if(vals[0]==0,{
                                grabBusses.do{arg item; item.set(vals[1])}
                                },{
                                rateBusses.do{arg item; item.set(vals[1].linlin(0,1,-4,4))};
                                });*/
                        },
                        9, {
                                if(vals[0]==0,{
                                        grabBusses.do{arg item; item.set(vals[1])}
                                },{
                                        rateBusses.do{arg item; item.set(vals[1].linlin(0,1,-4,4))};
                                });
                        },
                        10, {
                                if(vals[0]==0,{
                                        rateBusses.do{arg item; item.set(vals[1].linlin(0,1,-4,4))};
                                },{
                                        rateBusses.do{arg item; item.set(vals[1].linlin(0,1,-4,4))};
                                });
                        }

                )
}

init {

        this.makeWindow("RageTrombones",Rect(500, 500, 600, 400));
        numItemsPerGroup = [3,4,1,2];
        this.initControlsAndSynths(6+(numItemsPerGroup.sum));

        this.makeMixerToSynthBus(8);

        inBus0 = mixerToSynthBus.index;
        inBus1 = mixerToSynthBus.index+1;

        transferBus = Bus.audio(group.server, 2);
        verbBus = Bus.audio(group.server);

        numSections = 10;

        lpcValBusses = Array.fill(2, {Bus.control(group.server)});
        lpcValBusses.do{arg item; item.set(32)};

        whichRange0 = Pxrand([0,1,2,3], inf).asStream;
        whichRange1 = Pxrand([0,1,2,3], inf).asStream;

        rangeSwitch = Pseq([1,0], inf).asStream;

        section346Seq0 = Pseq([0,1], inf).asStream;
        section346Seq1 = Pseq([0,1], inf).asStream;
        section346Switch0 = section346Seq0.next;
        section346Switch1 = section346Seq1.next;
```

```
section5Seq0 = Pseq([0,1], inf).asStream;
section5Seq1 = Pseq([0,1], inf).asStream;
section5Switch0 = section5Seq0.next;
section5Switch1 = section5Seq1.next;
//volume busses

inVolBusses = List.newClear(0);
volBusses = List.newClear(0);

numItemsPerGroup.do{arg item;
	inVolBusses.add(Array.fill(item, {Bus.control(group.server)}));
	volBusses.add(Array.fill(item, {Bus.control(group.server)}));
};

//---------------------------------

sectionSeq = Pseq((0..numSections), inf).asStream;
//currentSection = sectionSeq.next;
this.nextSectionChangeSynths;

fxGroup0 = Group.tail(group);
fxGroup1 = Group.tail(group);

loopBuf0 = Buffer.alloc(group.server, group.server.sampleRate*8);
loopBuf1 = Buffer.alloc(group.server, group.server.sampleRate*8);
rateBusses = Array.fill(2, {Bus.control(group.server)});
rateBusses.do{arg item; item.set(1)};

grabBuf0 = Buffer.alloc(group.server, group.server.sampleRate);
grabBuf1 = Buffer.alloc(group.server, group.server.sampleRate);
grabBusses = Array.fill(2, {Bus.control(group.server)});

//create the chain of fx

freezeDist = Synth.tail(fxGroup1, "freezeDist_tbn", [\fxBus, transferBus.index]);

Synth.tail(fxGroup1, "loopMachine_tbn", [\fxBus, transferBus.index, \recBuf, loopBuf0, \rateBus,
rateBusses[0]]);
Synth.tail(fxGroup1, "loopMachine_tbn", [\fxBus, transferBus.index+1, \recBuf, loopBuf1, \rateBus,
rateBusses[1]]);
Synth.tail(fxGroup1, "grab_tbn", [\fxBus, transferBus.index, \grabBuf, grabBuf0, \grabBus, grabBusses[0]]);
Synth.tail(fxGroup1, "grab_tbn", [\fxBus, transferBus.index+1, \grabBuf, grabBuf1, \grabBus,
grabBusses[1]]);
Synth.tail(fxGroup1, "toOut_tbn", [\fxBus, transferBus, \outBus, outBus]);

4.do{arg i;
	controls.add(CCButton()
		.states_([["", Color.green, Color.green],["", Color.blue, Color.blue]])
		.action_{|butt|
			this.buttonPressed([i, butt.value])
		}
		.controlSpec_(ControlSpec(0,1,'lin',1))
		.minWidth_(100).minHeight_(100).maxWidth_(100).maxWidth_(100)
	);
	this.addAssignButton(i, \onOff);
};

currentSynths = List.newClear(0);
nextSynths = List.newClear(0);

this.nextSectionChangeSynths;

sectionNames = [
	["READY","READY"],
	["PopLip/Splosions", "PopLip/Splosions"],
	["Splosions/VerbSplosions", "Splosions/VerbSplosions"],
	["Splosions/PopLip", "Splosions/PopLip"],
	["Noise|Buzz/AmpFollowerGate","Noise|Buzz/AmpFollowerGate"],
	["AmpFollowerGate/LPCError", "AmpFollowerGate/LPCError"],
	["AmpFollowerGate/Noise|Buzz","AmpFollowerGate/Noise|Buzz"],
	["LPCError/SpectralFreeze","LPCError/Distortion"],
	["FMSynth (Carrier)","FMSynth (Modulator and Vol)"],
	["FMSynth (Carrier)","FMSynth (Modulator)"],
	["PopLip","Drone"]
];

sectionTexts = List.newClear(0);
2.do{arg i;
	sectionTexts.add(StaticText()
		.string_("0").font_(Font("Monaco", 360)));
};

infoTexts = List.newClear(0);
2.do{arg i;
	infoTexts.add(StaticText()
		.string_(sectionNames[0][i]).font_(Font("Monaco", 20)));
};

2.do{arg i;
	controls.add(CCSlider()
		.action_{arg sl; this.sliderMoved([i,sl.value])}
		.controlSpec_(ControlSpec(0,1,'amp'))
	);
	this.addAssignButton(i+4, \continuous);
};

numItemsPerGroup.do{|items, section|
	items.do{|i|
		controls.add(QtEZSlider(nil, ControlSpec(0, 8, 'amp'), {arg sl;
inVolBusses[section][i].set(sl.value)}, 1, true, \horz));
		controls.add(QtEZSlider(nil, ControlSpec(0, 8, 'amp'), {arg sl;
volBusses[section][i].set(sl.value)}, 1, true, \horz));
	}
};

win.layout_(
```

```
                VLayout(
                        HLayout(
                                VLayout(
                                        HLayout(
                                                VLayout(HLayout(VLayout(controls[0],
assignButtons[0].layout), VLayout(controls[1], assignButtons[1].layout)), infoTexts[0]),
                                                VLayout(controls[4], assignButtons[4].layout)),
                                        HLayout(
                                                VLayout(HLayout(VLayout(controls[2],
assignButtons[2].layout), VLayout(controls[3], assignButtons[3].layout)), infoTexts[1]),
                                                VLayout(controls[5], assignButtons[5].layout)),
                                ), sectionTexts[0]),
                                StaticText().string_("Sections 1-3"),
                                HLayout(StaticText().string_("Pop Lip"), controls[6].layout, controls[7].layout),
                                HLayout(StaticText().string_("Splosions"), controls[8].layout, controls[9].layout),
                                HLayout(StaticText().string_("Verb"), controls[10].layout, controls[11].layout),
                                StaticText().string_("Sections 4-8"),
                                HLayout(StaticText().string_("Noise"), controls[12].layout, controls[13].layout),
                                HLayout(StaticText().string_("Buzz"), controls[14].layout, controls[15].layout),
                                HLayout(StaticText().string_("AFGate"), controls[16].layout, controls[17].layout),
                                HLayout(StaticText().string_("LPCError"), controls[18].layout, controls[19].layout),
                                StaticText().string_("Section 9"),
                                HLayout(StaticText().string_("FMSynth"), controls[20].layout, controls[21].layout),
                                StaticText().string_("Section 10"),
                                HLayout(StaticText().string_("Drone"), controls[22].layout, controls[23].layout),
                                HLayout(StaticText().string_("PopLip"), controls[24].layout, controls[25].layout)
                ));
        }

        /*              saveExtra {arg saveArray;
        saveArray.add(arduinoAddress.string);
        }

        loadExtra {arg loadArray;
        arduinoAddress.string=loadArray[4].asString;
        }*/


        load {arg loadArray;

                loadArray[1].do{arg controlLevel, i;
                        //it will not load the value if the value is already correct (because Button seems messed up) or if
dontLoadControls contains the number of the controller
                        if((i>3)&&(controls[i].value!=controlLevel)&&(dontLoadControls.includes(i).not),{
                                controls[i].valueAction_(controlLevel);
                        });
                };

                loadArray[2].do{arg msg, i;
                        waitForSetNum = i;
                        if(msg!=nil,{
                                MidiOscControl.getFunctionNSetController(this, controls[i], msg, group.server, setups);
                                assignButtons[i].instantButton.value_(1);
                        })
                };

                if(win!=nil,{
                        win.bounds_(loadArray[3]);
                        win.visible_(false);
                });

                this.loadExtra(loadArray);
        }
}
```