partition and gate

for 1 or more performers playing sustaining instruments

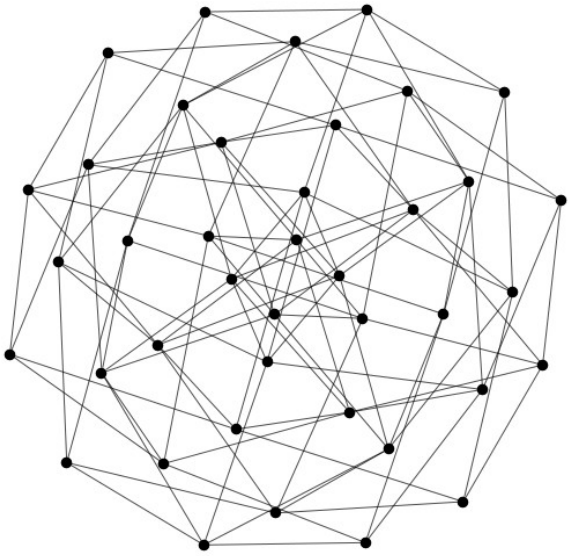
two microphones are placed equidistant from a single speaker such that performers can move freely in the space among the microphones and the speaker.

each tone (throughout, one pitch per player--not necessarily unique) should be sustained for a long time (a comfortable breath or bow stroke); entering and exiting as quietly as possible. sometimes overlapping; sometimes not. as if spontaneously iterating through subsets of the performers; attempting to favor solos, duos, trios, etc. that have not yet played together. always in constant interaction/consideration of the system, the setting, and each other. sometimes slowly moving about the space; sometimes still.

at any given time, the microphones are mapped to disjoint subsets (including the empty set) of four sources: a high frequency sine tone, a low frequency sine tone, and two recordings. the subset mapped from the the microphone with the louder* signal is output to the speaker while the other is gated. every 15 to 30 seconds, the mapping changes such that one microphone maps to the same subset while the other either adds or removes a source; always favoring mappings that have not yet occurred.

the recordings used in the first performances were 'black woman' recorded by alan lomax and 'greenwich' recorded at the san pedro, ca 2011 sacred harp convention. the choice of recordings, be it songs or field recordings, should be in reverent consideration of a particular space or place; its roots and/or environment.

at least 10 minutes; clear, not loud; perhaps in a dim space, only softly lit.



*loudness is determined by an amplitude follower as implemented in the attached supercollider code.

-michael winter (los angeles, 2014)

```
//electronics for partition and gate by michael winter. first realization. Supercollider 3.6.3. gpl.

(
var samp1, samp2;

samp1 = Buffer.read(s, "/home/mwinter/Portfolio/partition_and_gate/black_woman_mono_norm.wav");
samp2 = Buffer.read(s, "/home/mwinter/Portfolio/partition_and_gate/greenwich_183_mono_norm.wav");

SynthDef(\ampSelect, { arg a = #[0,0,0,0], b = #[0,0,0,0];
	var in, src, loudest, outbus=0, att = 0.01, rel = 0.1, lag = 0.0;
	in = {|i| SoundIn.ar(i)} ! 2;
	src = [DC.ar(0),
		SinOsc.ar(40, 0, 1 /*adjust accordingly*/), SinOsc.ar(5000, 0, 1 /*adjust accordingly*/),
		PlayBuf.ar(1, samp1.bufnum, loop: 1.0), PlayBuf.ar(1, samp2.bufnum, loop: 1.0)];
	rel = LocalIn.ar(1);
	loudest = Select.ar((Amplitude.ar(in[0], att, rel) < Amplitude.ar(in[1], att, rel)),
		[[Mix.fill(4, {|i|  SelectX.ar(a[i], src)}), DC.ar(0)], [Mix.fill(4, {|i|  SelectX.ar(b[i], src)}), DC.ar(1)]]);
	LocalOut.ar(TRand.ar(0.1, 0.6, Changed.ar(loudest[1]))**2);
	Out.ar(0, loudest[0])
}).add;
)

(
var edges, vertices, count, walk;

edges = [[4,5,6,15,16,17],[4,7,8,15,22,23],[5,7,9,16,22,25],[6,8,9,17,23,25],[0,1,10,11,26,27],[0,2,10,12,29,30],[0,3,11,12,32,33],[1,2,10,13,18,35],
[1,3,11,13,19,36],[2,3,12,13,20,24],[4,5,7,14,37],[4,6,8,14,38],[5,6,9,14,39],[7,8,9,14,21],[10,11,12,13],[0,1,18,19,29,32],[0,2,18,20,26,33],
[0,3,19,20,27,30],[7,15,16,21,34],[8,15,17,21,31],[9,16,17,21,28],[13,18,19,20],[1,2,24,26,29,36],[1,3,24,27,32,35],[9,22,23,28,39],
[2,3,30,33,35,36],[4,16,22,28,38],[4,17,23,28,37],[20,24,26,27],[5,15,22,31,39],[5,17,25,31,37],[19,29,30,36],[6,15,23,34,39],[6,16,25,34,38],
[18,32,33,35],[7,23,25,34,37],[8,22,25,31,38],[10,27,30,35],[11,26,33,36],[12,24,29,32]];

vertices = [[[],[1]],[[],[2]],[[],[3]],[[],[4]],[[],[1,2]],[[],[1,3]],[[],[1,4]],[[],[2,3]],[[],[2,4]],[[],[3,4]],[[],[1,2,3]],[[],[1,2,4]],[[],
[1,3,4]],[[],[2,3,4]],[[],[1,2,3,4]],[[1],[2]],[[1],[3]],[[1],[4]],[[1],[2,3]],[[1],[2,4]],[[1],[3,4]],[[1],[2,3,4]],[[2],[3]],[[2],[4]],[[2],[3,4]],
[[3],[4]],[[1,2],[3]],[[1,2],[4]],[[1,2],[3,4]],[[1,3],[2]],[[1,3],[4]],[[1,3],[2,4]],[[1,4],[2]],[[1,4],[3]],[[1,4],[2,3]],[[2,3],[4]],[[2,4],[3]],
[[1,2,3],[4]],[[1,2,4],[3]],[[1,3,4],[2]]];

count = {1} ! 40;

walk = {arg cur; var adjEdges, probs, choice;
	adjEdges = edges[cur]; probs = ({|i| count[adjEdges[i]]**3} ! adjEdges.size).normalizeSum;
	choice = adjEdges.wchoose(probs); count = count + 1; count[choice] = 0; probs.postln; count.postln; choice};

x = Synth(\ampSelect);

r = Routine({
    var delta, curVertex = 1, curOut, lastOut = vertices[1];
    loop {
        delta = rrand(15, 30) * 0.5; curVertex = walk.value(curVertex); curOut = vertices[curVertex];
		curOut[0] = curOut[0].extend(4, 0); curOut[1] = curOut[1].extend(4, 0);
		if((curOut[0] == lastOut[1]) || (curOut[1] == lastOut[0]), {curOut = curOut.reverse});
		curVertex.postln; curOut.postln;
		lastOut = curOut; x.set(\a, curOut[0], \b, curOut[1]); delta.yield;
    }
});
)

r.play;
r.stop;
```