File - /Users/devinarne/IdeaProjects/TestHB/src/unheardVoices/unheardVoices_da.java

```java
package unheardVoices;

import de.sciss.net.OSCListener;
import de.sciss.net.OSCMessage;
import net.beadsproject.beads.data.Sample;
import net.beadsproject.beads.data.SampleManager;
import net.beadsproject.beads.events.KillTrigger;
import net.beadsproject.beads.ugens.*;
import net.happybrackets.core.HBAction;
import net.happybrackets.core.scheduling.Clock;
import net.happybrackets.device.HB;

import java.lang.invoke.MethodHandles;
import java.net.SocketAddress;

public class unheardVoices_da implements HBAction {

    int clockInterval = 1000;
    float counterMod1 = 20f;
    int  loopfactor = 1;
    float pitch1init = 1f;
    float pitch1final = 1f;
    float pbspeed1 = 0.8f;
    float tuneVariation = 0.1f;

    float randomgrain1 = 1f;
    float grainsize1 = 800f;
    float graininterval1 = 1000f;


    int groupInt = 5; //determines base group
    int groupBound = 2; //sets up amount of spread
    int groupSpread = 0; //rng.nextINT(groupBound)
    int myGroup = 5; //group played by individual pi
    int numberOfGroups =39;
    String groupName = "1"; //myGroup.toString
    float verbMIX =.3f;
    float sampleStretched = 1.0f;

    float pbSpread = 1f;

    //set up transpostion and tuning
    float harmonicGen = 1f;
    float myPitchStart = 1f;
    float myPitchEnd = 1f;
    int harmInt1 = 1;
    int harmInt2 =1;

    @Override
    public void action(HB hb) {
        hb.reset(); //Clears any running code on the device
        hb.setStatus("--- UnheardVoices ---");
        int myIndex = Math.abs(hb.myIndex());
        float indexFloat= (float)myIndex;


        Reverb reverb = new Reverb();
        Gain verbGain = new Gain(2, verbMIX);
        verbGain.addInput(reverb);
        reverb.setSize(0.6f);
        reverb.setDamping(.6f);
        HB.getAudioOutput().addInput(verbGain);


        SampleManager.group("1", "data/audio/unheard_Voices/1_wind");
        SampleManager.group("2", "data/audio/unheard_Voices/2_sp_copper");
        SampleManager.group("3", "data/audio/unheard_Voices/3_GardenDrones");
        SampleManager.group("4", "data/audio/unheard_Voices/4_waterPC");
        SampleManager.group("5", "data/audio/unheard_Voices/5_waterPC");
        SampleManager.group("6", "data/audio/unheard_Voices/6_GardenDrones");
        SampleManager.group("7", "data/audio/unheard_Voices/7_seedPod");
        SampleManager.group("8", "data/audio/unheard_Voices/8_pcSoft");
        SampleManager.group("9", "data/audio/unheard_Voices/9_pcSoftGran");
        SampleManager.group("10", "data/audio/unheard_Voices/10_pcPluck");
        SampleManager.group("11", "data/audio/unheard_Voices/11_pcPluck");
        SampleManager.group("12", "data/audio/unheard_Voices/12_pineConeBowed");
        SampleManager.group("13", "data/audio/unheard_Voices/13_pineconebass");
        SampleManager.group("14", "data/audio/unheard_Voices/14_pineconebass");
```

```
 79          SampleManager.group("15", "data/audio/unheard_Voices/15_pcSoftGran");
 80          SampleManager.group("16", "data/audio/unheard_Voices/16_pcSoft");
 81          SampleManager.group("17", "data/audio/unheard_Voices/17_bowedTree");
 82          SampleManager.group("18", "data/audio/unheard_Voices/18_StringBranches");
 83          SampleManager.group("19", "data/audio/unheard_Voices/19_bowedTree");
 84          SampleManager.group("20", "data/audio/unheard_Voices/20_cactus");
 85          SampleManager.group("21", "data/audio/unheard_Voices/21_cactusPitched");
 86          SampleManager.group("22", "data/audio/unheard_Voices/22_leaves");
 87          SampleManager.group("23", "data/audio/unheard_Voices/23_pineNeedles");
 88          SampleManager.group("24", "data/audio/unheard_Voices/24_StringBranches");
 89          SampleManager.group("25", "data/audio/unheard_Voices/25_branches");
 90          SampleManager.group("26", "data/audio/unheard_Voices/26_cactus");
 91          SampleManager.group("27", "data/audio/unheard_Voices/27_StringBranches");
 92          SampleManager.group("28", "data/audio/unheard_Voices/28_cactus");
 93          SampleManager.group("29", "data/audio/unheard_Voices/29_cactus");
 94          SampleManager.group("30", "data/audio/unheard_Voices/30_cactusPitched");
 95          SampleManager.group("31", "data/audio/unheard_Voices/31_agave");
 96          SampleManager.group("32", "data/audio/unheard_Voices/32_agaveAir");
 97          SampleManager.group("33", "data/audio/unheard_Voices/33_agave");
 98          SampleManager.group("34", "data/audio/unheard_Voices/34_seedPod");
 99          SampleManager.group("35", "data/audio/unheard_Voices/35_seedPod");
100          SampleManager.group("36", "data/audio/unheard_Voices/36_pineConeBowed");
101          SampleManager.group("37", "data/audio/unheard_Voices/37_agave");
102          SampleManager.group("38", "data/audio/unheard_Voices/38_agaveAir");
103          SampleManager.group("39", "data/audio/unheard_Voices/39_GardenDrones");
104
105
106          hb.addBroadcastListener(new OSCListener() {
107              @Override
108              public void messageReceived(OSCMessage oscMessage, SocketAddress
    socketAddress, long l) {
109
110                  if(oscMessage.getName().equals("/time")){
111                      clockInterval = (int)hb.getFloatArg(oscMessage, 0);
112                      counterMod1 = hb.getFloatArg(oscMessage, 1);
113                      loopfactor = (int)hb.getFloatArg(oscMessage, 2);
114                  }
115
116                  if(oscMessage.getName().equals("/pitch")){
117                      pitch1init = hb.getFloatArg(oscMessage, 0);
118                      pitch1final = hb.getFloatArg(oscMessage, 1);
119                      pbspeed1 = hb.getFloatArg(oscMessage, 2);
120                  }
121
122                  if(oscMessage.getName().equals("/gran")){
123                      grainsize1 = hb.getFloatArg(oscMessage, 0);
124                      graininterval1 = hb.getFloatArg(oscMessage, 1);
125                      randomgrain1 = hb.getFloatArg(oscMessage, 2);
126                  }
127
128                  if(oscMessage.getName().equals("/group")){
129                      groupInt = (int)hb.getFloatArg(oscMessage, 0);
130                      groupBound = (int)hb.getFloatArg(oscMessage, 1);
131
132
133                  }
134
135              }
136          });
137
138
139          Clock clock = HB.createClock(clockInterval);
140          clock.start();// End Clock Timer
141
142          clock.addClockTickListener((offset, this_clock) -> {// Write your code below
     this line
143
144              int counter = (int) this_clock.getNumberTicks();
145              this_clock.setInterval(clockInterval);
146
147              //set group and amount of variation of groups across devices
148              groupSpread = hb.rng.nextInt(groupBound);
149              myGroup= groupInt+groupSpread;
150              float loopStartSet= 0.2f*hb.rng.nextFloat();
151              float loopEndSet= 0.7f+(0.5f*hb.rng.nextFloat());
152              float gainVar = 0.25f*hb.rng.nextFloat();
153
154
```

```
155             if (myGroup<numberOfGroups){
156                 groupName = String.format("%d", myGroup);
157             }
158             else if (myGroup > numberOfGroups){
159                 groupName = String.format("%d", numberOfGroups);
160             }
161 
162             //Playback start time variations
163             pbSpread = counterMod1+Math.round(Math.sqrt(myIndex)+hb.rng.nextInt(9));
164 
165             //set up transpostion and tuning
166 
167             harmInt1 = (hb.rng.nextInt(4)+2);
168             harmInt2 = harmInt1-1;
169 
170 
171             if (myIndex%2 ==0) {
172                 harmonicGen = (float)(harmInt1/harmInt2);
173             }
174             if (myIndex%2 !=0) {
175                 harmonicGen = 1f;
176             }
177 
178             myPitchStart = (pitch1init+tuneVariation)*harmonicGen;
179             myPitchEnd = (pitch1final+tuneVariation)*harmonicGen;
180 
181 
182             if (counter % (pbSpread) == 0) {
183                 Sample sample1 = SampleManager.randomFromGroup(groupName);
184                 float SampleLength = (float) sample1.getLength();
185                 GranularSamplePlayer gs1 = new GranularSamplePlayer(sample1);
186 
187 
188                 //Pitch Envelope---------------------------------------
189                 Envelope pitchEnv = new Envelope(myPitchStart);
190                 pitchEnv.addSegment(myPitchEnd, SampleLength);
191                 gs1.setPitch(pitchEnv);
192 
193                 Envelope loopStart = new Envelope(0f);
194                 loopStart.addSegment(loopStartSet, 10);
195                 gs1.setLoopStart(loopStart);
196                 Envelope loopEnd = new Envelope(0f);
197                 loopEnd.addSegment(loopEndSet, 15);
198                 gs1.setLoopEnd(loopEnd);
199 
200                 //Set Loop Mode-----------------------------------
201 
202                 if (loopfactor <= 2) {
203                     gs1.setLoopType(SamplePlayer.LoopType.NO_LOOP_FORWARDS);
204                 }
205                 else if (loopfactor == 3) {
206                     gs1.setLoopType(SamplePlayer.LoopType.LOOP_FORWARDS);
207                 }
208                 else if (loopfactor == 4) {
209                     gs1.setLoopType(SamplePlayer.LoopType.LOOP_ALTERNATING);
210                 }
211                 else if (loopfactor == 5) {
212                     gs1.setLoopType(SamplePlayer.LoopType.LOOP_BACKWARDS);
213                 }
214                 else {
215                     gs1.setLoopType(SamplePlayer.LoopType.NO_LOOP_FORWARDS);
216                 }
217 
218                 //Set Tuning & Pitch Variation
219 
220                 if (myGroup %3 !=0) {
221                     tuneVariation= (float)Math.sqrt(indexFloat)*hb.rng.nextFloat();
222                 }
223 
224                 else if (myGroup %3 == 0) {
225                     tuneVariation = (float)(.01*Math.sqrt(myIndex)); //detune of .01
    -.05 (chorus like effect)
226                 }
227 
228                 else {
229                     tuneVariation = 0.1f*hb.rng.nextInt(4);
230                 }
231 
```

```
232                 //Granular Parameters-------------------------------------
233                 Envelope grainsize = new Envelope(120);
234                 grainsize.addSegment(grainsize1, SampleLength);
235                 gs1.setGrainSize(grainsize);
236
237                 Envelope grainInterval = new Envelope(100);
238                 grainInterval.addSegment(graininterval1, SampleLength);
239                 gs1.setGrainInterval(grainInterval);
240
241                 gs1.getRateUGen().setValue(pbspeed1);
242                 gs1.getRandomnessUGen().setValue(randomgrain1);
243
244                 //Gain and Gain Envelope----------------------------------
245
246                 sampleStretched =SampleLength*pbspeed1;
247                 Envelope gainEnvelope = new Envelope();
248                 Gain gain = new Gain(2, gainEnvelope);
249
250
251                 if (myGroup %3 != 0 && loopfactor == 1) {
252                     gainEnvelope.addSegment(0.9f+gainVar, 10);
253                     gainEnvelope.addSegment(0.8f+gainVar, sampleStretched*.75f);
254                     gainEnvelope.addSegment(0f, 10, new KillTrigger(gain));
255                     verbMIX =.6f;
256                 }
257
258
259                 else if (myGroup %3 == 0 && loopfactor == 1) {
260                     gainEnvelope.addSegment(0.7f+gainVar, SampleLength/2);
261                     gainEnvelope.addSegment(0.8f+gainVar, SampleLength/4);
262                     gainEnvelope.addSegment(0.5f+gainVar, SampleLength/4);
263                     gainEnvelope.addSegment(0f, 10, new KillTrigger(gain));
264                     verbMIX =.6f;
265                 }
266
267                 else if (myGroup %3 != 0 && loopfactor == 2) {
268                     gainEnvelope.addSegment(0.8f+gainVar, 10);
269                     gainEnvelope.addSegment(0.7f+gainVar, sampleStretched);
270                     gainEnvelope.addSegment(0f, 10, new KillTrigger(gain));
271                     verbMIX =.6f;
272                 }
273
274                 else if (myGroup %3 == 0 && loopfactor == 2) {
275                     gainEnvelope.addSegment(0.6f+gainVar, SampleLength/4);
276                     gainEnvelope.addSegment(0.8f+gainVar, SampleLength/4);
277                     gainEnvelope.addSegment(0.4f+gainVar, SampleLength/2);
278                     gainEnvelope.addSegment(0f, 10, new KillTrigger(gain));
279                     verbMIX =.8f;
280                 }
281
282                 else if (myGroup %3 != 0 && loopfactor >=3) {
283                     gainEnvelope.addSegment(0.6f+gainVar, SampleLength/4);
284                     gainEnvelope.addSegment(0.4f+gainVar, SampleLength/2);
285                     gainEnvelope.addSegment(0.6f+gainVar, SampleLength/2);
286                     gainEnvelope.addSegment(0.3f+gainVar, SampleLength/2);
287                     gainEnvelope.addSegment(0f, 10, new KillTrigger(gain));
288                     verbMIX = .9f;
289                 }
290
291                 else if (myGroup %3 == 0 && loopfactor >= 3) {
292                     gainEnvelope.addSegment(0.4f+gainVar, SampleLength/2);
293                     gainEnvelope.addSegment(0.75f+gainVar, SampleLength/2);
294                     gainEnvelope.addSegment(0.3f+gainVar, SampleLength/2);
295                     gainEnvelope.addSegment(0.6f+gainVar, SampleLength/2);
296                     gainEnvelope.addSegment(0f, 10, new KillTrigger(gain));
297                     verbMIX =.9f;
298                 }
299
300
301                 //Main and FX output
302                 gain.addInput(gs1);
303
304
305                 HB.getAudioOutput().addInput(gain);
306                 reverb.addInput(gain);
307
308                 hb.setStatus(" |clockinterval " + clockInterval +"GroupInt" +
    groupInt+ "LoopFactor" + loopfactor);
```

```
309
310                 //hb.setStatus(" loopfactor: " + loopfactor + " looptype: "+ gs1.
    getLoopType()+ "loopstart:  "+loopStartSet+ "loopend: "+ loopEndSet);
311                 //hb.setStatus("groupSpread"+ groupSpread + "myGroup" + myGroup + "
    samplePosition"+ sampleposition);
312
313             }
314
315
316             // Write your code above this line
317         });
318
319
320
321
322
323
324
325
326         // write your code above this line
327     }
328
329
330     //<editor-fold defaultstate="collapsed" desc="Debug Start">
331
332     /**
333      * This function is used when running sketch in IntelliJ IDE for debugging or
    testing
334      *
335      * @param args standard args required
336      */
337     public static void main(String[] args) {
338
339         try {
340             HB.runDebug(MethodHandles.lookup().lookupClass());
341         } catch (Exception e) {
342             e.printStackTrace();
343         }
344     }
345     //</editor-fold>
346 }
347
```