Huw Catchpole-Davies

Generative Triptych of Percussive Music

For: Synthesised Percussion; Electronic Oscillators; and Corrupted Sentence

Generative Triptych of Percussive Music (a.k.a. MUSIC SYSTEMS) Preface

A video programme note/demonstration for this work can be found on the accompanying DVD. The video discusses all the detail that can be found here and should be viewed in parallel with this text note.

In this note I will explain the code for the generative triptych. I will then go on to explain some of the interesting explorations within the individual works contained within the triptych. Towards the end of the video is an explanation of how you can compose your own scores for this instrument within the Interactive Portfolio. Help on 'How to Operate' can be found at any time within the portfolio by clicking the help interface in the bottom right corner of the interface.

The individual works within this triptych are explorations of an instrument of my design. The programming language *ChucK*, designed for Princeton Laptop Orchestra was used to code these works. Herein I include the design of the instrument.

This instrument was designed to generate individual bars of music. I first designed a class, which I named generateBar. This class included a play method. Upon calling this method the bar will be generated and played. The play method accepts various different parameters, which define the qualities of the bar to be generated. Although the created bar's microcosm is at the computers discretion the macrocosm is determined within the specified parameters. Within these parameters are: the chances for a specific sound file to fire, the number of beats in a bar, the bar's tempo and number of repetitions of that bar. The method of generation is similar in each of the individual works.

The parameters controlling bar beats, tempo and repetitions are self-explanatory. Beats are accepted as an integer greater than 0; tempo is accepted as an integer in BPM; repetitions is accepted as an integer greater than 0. Those determining chance require a small amount of extra explanation.

Chance that a specific sound will be played is determined through the following algorithm: A number is set for the total number of beats in the bar, in our example we will use the number 8. Then a number is set for how many chances a specific sound will play, lets take the number 4. As there are 8 available beat slots to fill the program will now determine randomly 4 times which of these available beats the sound file will play on. If we take the generations to have produced a 1, a 5, a 8 and another 1. Shown here is that each individual roll was unrelated to the last and therefore two 1's were rolled. The bar produced will play the sound on beat 1, beat 5, and beat 8. As there was a second 1 rolled and beat one had already been filled, this second roll is disregarded.

Careful choice of the ratios between these two parameters results in many unique outcomes. Two examples employed in this portfolio are discussed below. In a bar of length X, giving the sound file only a single chance to play will produce a certainty that that sound will play exactly once during that bar. Conversely, in a bar of length X, raising the chances above the value of X can produce greater and greater chances that the bar will be filled completely. The opening bars of the generative sonata for synthesized drum-kit use both of these techniques simultaneously. I describe these techniques as the standard techniques for this instrument.

The structure of these works is created by many consecutive calls to the play method. A video programme note to this work is provided on the accompanying DVD and should be viewed in parallel with this text programme note.

Two more advance techniques became apparent to me upon composing the second and third works in the triptych. The first is regarding the composite-bar. A composite-bar is a single bar made up of many different calls to the play method. These contributing calls are each made up of highly likely parameter combinations. The effect of this is the ability to create defined gestures in a work of highly chance-like material. This technique is used in the second work and more prominently in the third.

The second advanced technique was in the types of sound files chosen. In the third work, the use of vocal sound files allows a greater freedom of expression through use of the tempo parameter. This is due to the rich qualities found from within even relatively small duration vocal sound clips. Longer or shorter tempos combined with the instrument's percussive repetition allowed these micro-qualities to come to the fore in the third work. In the first two works I had kept my choices of tempo change to mathematical ratios of the tempos immediately preceding or succeeding. I had also generally maintained patterns within sections.

I believe this work is successful in creating variety over multiple playsthroughs while maintaining the essence making each composed work its own. Therefore multiple playthroughs become a feature of the work. It's important to note that it was not the intent, from my point of view with this work or any others in this portfolio, to create works of infinite interest. The term 'multiple playthroughs' here refers to an equivalent interest that might be achieved from multiple recordings of an acoustic work – greater than zero and yet non-infinite. In other words, though the number of variations a single score of these sonatas *could* produce is very large I do not see a listener's interest extending to this number of iterations. This is a psychological feature bound by humans' innate skill at pattern recognition. More

research would be required to find a point at which these works no longer excite due to this biological feature.

The video programme note/demonstration goes on to demonstrate how to use the ‘Compose your own’ scoring system within the Interactive Portfolio application and should be viewed by the reader now.

The scores for the play(); methods for each version of the instrument are found overleaf.

```
//Sonata for Sythesised Percussion
//Huw Catchpole-Davies 2012
//
//ChucK loopGen class before this.
//
//To adjust the piece:
//Each call of the play function receives parameters in this way:
//(barbeats, iterations, tempo,
// hihatChance, openhihatChance, kickChance, snareChance
// tomtomChance, cowbellChance, maracasChance)


//MUSIC GENERATION


//All the function calls
loopGen generateBar;

//Chaining Calls to generateBar.play()

//INTRO
generateBar.play(3, 1, 60, 32, 32, 0, 0, 0, 0, 0);
generateBar.play(3, 2, 60, 2, 2, 0, 0, 0, 0, 0);
generateBar.play(1, 1, 60, 32, 32, 0, 0, 0, 0, 0);
generateBar.play(2, 1, 60, 0, 0, 0, 0, 0, 0, 0);

// EXPOSITION
// FIRST SUBJECT
for (0 => int i; i < 2; i++) // for loop will allow repeat
{
    generateBar.play(3, 1, 60, 0, 6, 6, 6, 0, 0, 0);     // b1
    generateBar.play(12, 2, 180, 32, 0, 1, 0, 0, 0, 0);  // b2
    generateBar.play(3, 1, 60, 0, 6, 3, 3, 0, 0, 0);     // b3
    generateBar.play(12, 2, 180, 32, 0, 1, 0, 0, 0, 0);  // b4
    generateBar.play(3, 1, 60, 0, 6, 3, 3, 0, 0, 0);     // b5
    generateBar.play(12, 2, 180, 32, 0, 1, 0, 0, 0, 0);  // b6
    generateBar.play(3, 1, 60, 0, 6, 3, 3, 0, 0, 0);     // b7
    generateBar.play(8, 1, 180, 0, 6, 4, 6, 0, 0, 0);    // b8

    generateBar.play(12, 1, 180, 16, 16, 2, 2, 0, 0, 0); // b9
    generateBar.play(12, 2, 180, 16, 16, 2, 2, 0, 0, 0); // b10
    generateBar.play(3, 1, 60, 5, 6, 3, 3, 0, 0, 0);     // b11
    generateBar.play(12, 2, 180, 16, 16, 2, 2, 0, 0, 0); // b12
    generateBar.play(12, 2, 180, 16, 16, 2, 2, 0, 0, 0); // b13
    generateBar.play(3, 1, 60, 5, 6, 2, 0, 0, 0, 0);     // b14
    generateBar.play(12, 2, 180, 16, 16, 2, 2, 0, 0, 0); // b15
    generateBar.play(8, 1, 180, 16, 16, 4, 6, 0, 0, 0);  // b16

// MAIN BRIDGE
    generateBar.play(8, 2, 180, 16, 16, 4, 6, 0, 0, 0);  // b17
    generateBar.play(2, 1, 150, 0, 0, 12, 12, 0, 0, 0);  // b18
```

```
// SECOND SUBJECT
    generateBar.play(3, 1, 150, 0, 6, 0, 6, 6, 2, 3);    // b20
    generateBar.play(6, 4, 150, 4, 0, 1, 0, 0, 4, 6);    // b21
    generateBar.play(3, 1, 100, 0, 6, 3, 0, 3, 2, 3);    // b22
    generateBar.play(6, 4, 150, 4, 0, 1, 0, 0, 4, 6);    // b23
    generateBar.play(3, 1, 100, 0, 6, 3, 0, 3, 2, 3);    // b24
    generateBar.play(6, 4, 150, 4, 0, 1, 0, 0, 4, 6);    // b25
    generateBar.play(3, 1, 100, 0, 6, 3, 0, 3, 2, 3);    // b26
    generateBar.play(8, 1, 150, 0, 6, 4, 6, 0, 0, 8);    // b27

    generateBar.play(3, 1, 150, 0, 6, 0, 1, 6, 0, 16);   // b28
    generateBar.play(6, 4, 150, 4, 0, 1, 0, 0, 7, 16);   // b29
    generateBar.play(3, 1, 100, 0, 6, 3, 1, 3, 4, 16);   // b30
    generateBar.play(6, 4, 150, 4, 0, 1, 1, 0, 7, 16);   // b31
    generateBar.play(3, 1, 100, 0, 6, 3, 1, 3, 4, 16);   // b32
    generateBar.play(6, 4, 150, 4, 0, 1, 1, 0, 7, 16);   // b33
    generateBar.play(3, 1, 100, 0, 6, 3, 1, 3, 4, 16);   // b34
    generateBar.play(8, 1, 150, 0, 6, 4, 6, 0, 0, 16);   // b35

} // End of repeat comprising the whole exposition



// SMALL BRIDGE PASSAGE TO DEVELOPMENT
generateBar.play(3, 1, 80, 0, 6, 6, 6, 0, 0, 0);         // b36
generateBar.play(1, 1, 80, 0, 8, 8, 8, 0, 0, 0);         // b37

//DEVELOPMENT
generateBar.play(11, 3, 240, 32, 2, 0, 0, 0, 0, 6);      // b38
generateBar.play(5, 1, 80, 0, 3, 4, 4, 4, 0, 3);         // b39
generateBar.play(11, 3, 240, 32, 2, 1, 0, 0, 0, 6);      // b40
generateBar.play(5, 2, 80, 0, 3, 0, 4, 4, 4, 3);         // b41
generateBar.play(11, 3, 240, 32, 4, 2, 0, 0, 0, 8);      // b42
generateBar.play(5, 1, 80, 0, 3, 0, 4, 4, 4, 4);         // b43
generateBar.play(11, 3, 240, 32, 6, 3, 3, 0, 0, 8);      // b44
generateBar.play(8, 2, 240, 0, 3, 4, 4, 4, 4, 4);        // b45

generateBar.play(12, 2, 240, 32, 6, 3, 3, 3, 0, 8);      // b46
generateBar.play(12, 2, 240, 32, 6, 3, 3, 3, 0, 8);      // b47
generateBar.play(4, 1, 80, 0, 3, 0, 4, 4, 4, 4);         // b48
generateBar.play(12, 2, 240, 32, 6, 3, 3, 0, 3, 8);      // b49
generateBar.play(12, 2, 240, 32, 6, 3, 3, 0, 3, 8);      // b50
generateBar.play(4, 1, 80, 0, 3, 0, 4, 4, 4, 4);         // b51
generateBar.play(8, 2, 240, 0, 3, 4, 4, 4, 4, 4);        // b52
generateBar.play(8, 2, 220, 0, 3, 4, 4, 4, 4, 4);        // b53

//RECAP
//material from second subject
generateBar.play(3, 1, 180, 0, 6, 0, 1, 6, 0, 16);       // b54
generateBar.play(6, 4, 180, 4, 0, 1, 0, 0, 7, 16);       // b55
generateBar.play(3, 1, 120, 0, 6, 3, 1, 3, 4, 16);       // b56
generateBar.play(6, 4, 180, 4, 0, 1, 1, 0, 7, 16);       // b57
```

```
generateBar.play(3, 1, 120, 0, 6, 3, 1, 3, 4, 16);       // b58
generateBar.play(6, 4, 180, 4, 0, 1, 1, 0, 7, 16);       // b59
generateBar.play(3, 1, 120, 0, 6, 3, 1, 3, 4, 16);       // b60
generateBar.play(8, 2, 180, 0, 6, 4, 6, 0, 0, 16);       // b61

//BRIDGING PASSAGE
generateBar.play(4, 1, 190, 0, 0, 6, 6, 0, 0, 0);        // b62
generateBar.play(4, 1, 210, 0, 0, 6, 6, 0, 0, 0);        // b63
generateBar.play(4, 1, 230, 8, 8, 2, 3, 0, 0, 0);        // b64
generateBar.play(4, 1, 250, 8, 8, 2, 3, 0, 0, 0);        // b65


//material from first subject
generateBar.play(12, 2, 270, 16, 16, 2, 2, 0, 0, 0);     // b66
generateBar.play(6, 2, 270, 16, 16, 2, 2, 0, 0, 0);      // b67
generateBar.play(3, 1, 90, 5, 6, 3, 3, 0, 0, 0);         // b68
generateBar.play(12, 2, 270, 16, 16, 2, 2, 0, 0, 0);     // b69
generateBar.play(6, 2, 270, 16, 16, 2, 2, 0, 0, 0);      // b70
generateBar.play(3, 1, 90, 5, 6, 2, 0, 0, 0, 0);         // b71
generateBar.play(12, 2, 270, 16, 16, 2, 2, 0, 0, 0);     // b72
generateBar.play(8, 2, 270, 16, 16, 4, 6, 0, 0, 0);      // b73

//CODA
generateBar.play(3, 1, 90, 7, 7, 0, 0, 0, 0, 7);         // b74
generateBar.play(1, 1, 90, 0, 0, 0, 0, 0, 0, 0);         // b75
generateBar.play(3, 1, 90, 7, 7, 7, 7, 0, 0, 0);         // b76
```

```
//Sonata for 10 Electronic Oscillators
//Huw Catchpole-Davies (Quixatocs) 2012
//
//ChucK plugPulseGen class before this.

//MUSIC GENERATION

//All the function calls
plugPulseGen generateBarPitchGroup1;
plugPulseGen generateBarPitchGroup2;
plugPulseGen generateBarPitchGroup3;

//Chaining Calls to generateBar.play()

//INTRO
generateBarPitchGroup1.play(8, 2, 240, 0, 0, 0, 0, 0, 0, 4, 4, 0, 0);    // b1
generateBarPitchGroup1.play(8, 2, 240, 0, 0, 0, 0, 0, 0, 0, 0, 4, 4);    // b2
generateBarPitchGroup1.play(8, 2, 240, 0, 0, 0, 0, 0, 0, 4, 4, 0, 0);    // b3
generateBarPitchGroup1.play(8, 2, 240, 0, 0, 0, 0, 0, 0, 0, 0, 4, 4);    // b4

// FIRST SUBJECT
for (0 => int i; i < 2; i++) // for loop will allow repeat
{
    generateBarPitchGroup1.play(8, 2, 240, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4); // b5
    generateBarPitchGroup1.play(8, 2, 240, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4); // b6
    generateBarPitchGroup1.play(8, 2, 240, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4); // b7
    generateBarPitchGroup1.play(8, 2, 240, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4); // b8

    generateBarPitchGroup1.play(8, 2, 240, 4, 4, 0, 0, 0, 0, 0, 0, 0, 0); // b9
    generateBarPitchGroup1.play(8, 2, 240, 4, 4, 4, 4, 0, 0, 0, 0, 0, 0); // b10
    generateBarPitchGroup1.play(8, 2, 240, 4, 4, 0, 0, 0, 0, 0, 0, 0, 0); // b11
    generateBarPitchGroup1.play(8, 2, 240, 4, 4, 4, 4, 0, 0, 0, 0, 0, 0); // b12

// MAIN BRIDGE
    generateBarPitchGroup1.play(8, 2, 240, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8); // b13
    generateBarPitchGroup1.play(8, 2, 240, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8); // b14

// SECOND SUBJECT SECOND SUBJECT SECOND SUBJECT SECOND SUBJECT
    generateBarPitchGroup2.play(4, 2, 120, 0, 0, 4, 4, 0, 1, 0, 0, 0, 0);   // b15
    generateBarPitchGroup2.play(4, 2, 120, 0, 0, 0, 4, 4, 4, 0, 1, 0, 0);   // b16
    generateBarPitchGroup2.play(4, 2, 120, 0, 0, 0, 0, 4, 4, 0, 1, 0, 0);   // b17
    generateBarPitchGroup2.play(1, 1, 30, 32, 32, 0, 0, 0, 0, 0, 0, 0, 0);  // b18
    generateBarPitchGroup2.play(4, 2, 120, 0, 0, 4, 4, 0, 1, 0, 0, 0, 0);   // b19
    generateBarPitchGroup2.play(4, 2, 120, 0, 0, 0, 4, 4, 4, 0, 1, 0, 0);   // b20
    generateBarPitchGroup2.play(4, 2, 120, 0, 0, 0, 0, 4, 4, 0, 1, 0, 0);   // b21
    generateBarPitchGroup2.play(1, 1, 30, 32, 32, 0, 0, 0, 0, 0, 0, 0, 0);  // b22-a
    generateBarPitchGroup2.play(1, 1, 30, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);    // b22-b
    generateBarPitchGroup2.play(1, 1, 30, 32, 32, 0, 0, 0, 0, 0, 0, 0, 0);  // b22-c
    generateBarPitchGroup2.play(1, 1, 30, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);    // b22-d
    generateBarPitchGroup2.play(1, 1, 60, 32, 32, 0, 0, 0, 0, 0, 0, 0, 0);  // b22-e
    generateBarPitchGroup2.play(1, 1, 60, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);    // b22-f
} // End of repeat comprising the whole exposition
```

```
// SMALL BRIDGE PASSAGE TO DEVELOPMENT
generateBarPitchGroup3.play(1, 1, 120, 32, 32, 0, 0, 0, 0, 0, 0, 0, 0);      // b23-a
generateBarPitchGroup3.play(1, 1, 120, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);        // b23-b
generateBarPitchGroup3.play(1, 1, 120, 32, 32, 0, 0, 0, 0, 0, 0, 0, 0);      // b23-c
generateBarPitchGroup3.play(1, 1, 120, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);        // b23-d
generateBarPitchGroup3.play(1, 1, 240, 32, 32, 0, 0, 0, 0, 0, 0, 0, 0);      // b23-e
generateBarPitchGroup3.play(1, 1, 240, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);        // b23-f
generateBarPitchGroup3.play(1, 1, 240, 32, 32, 0, 0, 0, 0, 0, 0, 0, 0);      // b23-g
generateBarPitchGroup3.play(1, 1, 240, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);        // b23-h
generateBarPitchGroup3.play(4, 1, 240, 32, 32, 0, 0, 0, 0, 0, 0, 0, 0);      // b23-i

//DEVELOPMENT
generateBarPitchGroup1.play(8, 2, 240, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4);        // b24
generateBarPitchGroup2.play(4, 2, 120, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2);        // b25
generateBarPitchGroup3.play(2, 4, 240, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);        // b26
generateBarPitchGroup1.play(8, 2, 240, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4);        // b27
generateBarPitchGroup2.play(4, 2, 120, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2);        // b28
generateBarPitchGroup3.play(2, 4, 240, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);        // b29
generateBarPitchGroup3.play(2, 4, 240, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);        // b30
generateBarPitchGroup3.play(2, 4, 240, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);        // b31

//RECAP
generateBarPitchGroup3.play(8, 2, 240, 4, 4, 0, 0, 0, 0, 0, 0, 0, 0);        // b32
generateBarPitchGroup3.play(8, 2, 240, 4, 4, 4, 4, 0, 0, 0, 0, 0, 0);        // b33
generateBarPitchGroup3.play(8, 2, 240, 4, 4, 0, 0, 0, 0, 0, 0, 0, 0);        // b34
generateBarPitchGroup3.play(8, 2, 240, 4, 4, 4, 4, 0, 0, 0, 0, 0, 0);        // b35

generateBarPitchGroup3.play(8, 2, 240, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4);        // b36
generateBarPitchGroup3.play(8, 2, 240, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4);        // b37
generateBarPitchGroup3.play(8, 2, 240, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4);        // b38
generateBarPitchGroup3.play(8, 2, 240, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4);        // b39

//CODA
generateBarPitchGroup1.play(1, 1, 240, 32, 32, 0, 0, 0, 0, 0, 0, 0, 0);      // b40-a
generateBarPitchGroup1.play(1, 1, 240, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);        // b40-b
generateBarPitchGroup1.play(1, 1, 240, 32, 32, 0, 0, 0, 0, 0, 0, 0, 0);      // b40-c
generateBarPitchGroup1.play(1, 1, 240, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);        // b40-d
generateBarPitchGroup2.play(1, 1, 120, 32, 32, 0, 0, 0, 0, 0, 0, 0, 0);      // b41-a
generateBarPitchGroup2.play(1, 1, 120, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);        // b41-b
generateBarPitchGroup2.play(1, 1, 120, 32, 32, 0, 0, 0, 0, 0, 0, 0, 0);      // b41-c
generateBarPitchGroup2.play(1, 1, 120, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);        // b41-d
generateBarPitchGroup3.play(1, 1, 60, 32, 32, 0, 0, 0, 0, 0, 0, 0, 0);       // b42-a
generateBarPitchGroup3.play(1, 1, 60, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);         // b42-b
generateBarPitchGroup3.play(1, 1, 60, 32, 32, 0, 0, 0, 0, 0, 0, 0, 0);       // b42-c
```

```
//Sonata for Corrupted Sentence
//Huw Catchpole-Davies (Quixatocs) 2012
//
//ChucK SentenceGen class before this.

//MUSIC

//All the function calls
SentenceGen generateBar;

//Chaining calls to a generateBar.play()

//INTRO
generateBar.play(1, 1, 90, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);      // b1-a
generateBar.play(1, 1, 90, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);      // b1-b
generateBar.play(1, 1, 90, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);      // b1-c
generateBar.play(1, 1, 90, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0);      // b1-d
generateBar.play(1, 1, 90, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0);      // b1-e
generateBar.play(1, 1, 90, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0);      // b1-f
generateBar.play(1, 1, 90, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0);      // b1-g
generateBar.play(1, 1, 90, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0);      // b1-h
generateBar.play(1, 1, 90, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0);      // b1-i
generateBar.play(1, 1, 90, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0);      // b1-j
generateBar.play(1, 1, 70, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0);      // b1-k
generateBar.play(1, 1, 90, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0);      // b1-l
generateBar.play(1, 1, 90, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2);      // b1-m
generateBar.play(1, 1, 40, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2);      // b1-n

// FIRST SUBJECT
for (0 => int i; i < 2; i++) // for loop will allow repeat
{
    generateBar.play(4, 2, 100, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 7); // b2
    generateBar.play(4, 2, 100, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 5); // b3
    generateBar.play(4, 2, 100, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 7, 0); // b4
    generateBar.play(4, 2, 100, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 5, 0); // b5
    generateBar.play(4, 2, 100, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 5, 0, 7); // b6
    generateBar.play(4, 2, 100, 0, 0, 5, 0, 0, 0, 0, 0, 0, 1, 0, 7, 0, 0); // b7
    generateBar.play(4, 2, 100, 0, 0, 7, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1); // b8
    generateBar.play(4, 2, 100, 0, 0, 0, 7, 0, 0, 0, 5, 0, 0, 0, 1, 0, 0); // b9

    generateBar.play(4, 2, 100, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 5, 7); // b10
    generateBar.play(4, 2, 100, 2, 0, 3, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0); // b11
    generateBar.play(4, 2, 100, 3, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0); // b12
    generateBar.play(4, 2, 100, 2, 0, 4, 0, 5, 5, 0, 0, 0, 0, 0, 0, 0, 0); // b13
    generateBar.play(4, 2, 100, 1, 0, 1, 0, 5, 7, 0, 0, 0, 0, 0, 0, 0, 1); // b14
    generateBar.play(4, 2, 100, 1, 0, 1, 7, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0); // b15
    generateBar.play(4, 2, 100, 1, 0, 1, 5, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0); // b16
    generateBar.play(4, 2, 100, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1); // b17

// MAIN BRIDGE
    generateBar.play(1, 1, 60, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);  // b18-a
    generateBar.play(1, 1, 60, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0);  // b18-b
```

```
    generateBar.play(1, 1, 60, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6);  // b18-c
    generateBar.play(1, 1, 30, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);  // b18-d

// SECOND SUBJECT SECOND SUBJECT SECOND SUBJECT SECOND SUBJECT
    generateBar.play(4, 1, 90, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0);  // b19
    generateBar.play(4, 1, 90, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0);  // b20
    generateBar.play(4, 1, 90, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0);  // b21
    generateBar.play(4, 1, 90, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1);  // b22
    generateBar.play(4, 1, 90, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0);  // b23
    generateBar.play(4, 1, 90, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0);  // b24
    generateBar.play(4, 1, 90, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0);  // b25
    generateBar.play(4, 1, 90, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1);  // b26

    generateBar.play(4, 1, 80, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);  // b27
    generateBar.play(4, 1, 80, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);  // b28
    generateBar.play(4, 1, 80, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);  // b29
    generateBar.play(4, 1, 80, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);  // b30
    generateBar.play(4, 1, 80, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);  // b31
    generateBar.play(4, 1, 80, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);  // b32
    generateBar.play(4, 1, 80, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);  // b33
    generateBar.play(4, 1, 80, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);  // b34

} // End of repeat comprising the whole exposition

  // SMALL BRIDGE PASSAGE TO DEVELOPMENT
generateBar.play(1, 1, 60, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);      // b35-a
generateBar.play(1, 1, 60, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);      // b35-b
generateBar.play(1, 1, 60, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6);      // b35-c
generateBar.play(1, 1, 30, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);      // b35-d
generateBar.play(1, 1, 60, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6);      // b35-e
generateBar.play(1, 1, 60, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);      // b35-f
generateBar.play(1, 1, 60, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0);      // b35-g
generateBar.play(1, 1, 30, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);      // b35-h

//DEVELOPMENT
generateBar.play(4, 1, 80, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);      // b36
generateBar.play(4, 2, 160, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);     // b37
generateBar.play(4, 1, 80, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);      // b38
generateBar.play(4, 2, 160, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);     // b39
generateBar.play(4, 1, 80, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);      // b40
generateBar.play(4, 2, 160, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);     // b41
generateBar.play(4, 1, 80, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);      // b42
generateBar.play(1, 1, 40, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0);      // b43-a
generateBar.play(1, 1, 20, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);      // b43-b

generateBar.play(4, 2, 160, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);     // b44-a
generateBar.play(1, 1, 20, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0);      // b44-b
generateBar.play(1, 1, 80, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);      // b44-c
generateBar.play(4, 2, 160, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);     // b45-a
generateBar.play(1, 1, 20, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0);      // b45-b
generateBar.play(1, 1, 80, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);      // b45-c
generateBar.play(4, 1, 80, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);      // b46-a
```

```
generateBar.play(4, 2, 160, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);      // b46-b
generateBar.play(1, 1, 20, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0);      // b46-c
generateBar.play(1, 1, 80, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);      // b46-d
generateBar.play(4, 1, 80, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);      // b47-a
generateBar.play(4, 2, 160, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);      // b47-b
generateBar.play(1, 1, 20, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0);      // b47-c
generateBar.play(1, 1, 80, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);      // b47-d
generateBar.play(1, 1, 20, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0);      // b47-e
generateBar.play(1, 1, 80, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);      // b47-f
generateBar.play(1, 1, 20, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0);      // b47-g
generateBar.play(1, 1, 80, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);      // b47-h
generateBar.play(4, 2, 160, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);      // b48-a
generateBar.play(1, 1, 20, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0);      // b48-b
generateBar.play(1, 1, 80, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);      // b48-c
generateBar.play(4, 2, 80, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);      // b49
generateBar.play(4, 2, 160, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);      // b50
generateBar.play(4, 2, 160, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);      // b51

//RECAP
generateBar.play(1, 1, 20, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);      // b52-a
generateBar.play(1, 1, 80, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);      // b52-b
generateBar.play(1, 1, 90, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);      // b52-c
generateBar.play(1, 1, 90, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);      // b52-d
generateBar.play(1, 1, 90, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);      // b52-e
generateBar.play(1, 1, 90, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0);      // b52-f
generateBar.play(1, 1, 90, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0);      // b52-g
generateBar.play(1, 1, 90, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0);      // b52-h
generateBar.play(1, 1, 90, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0);      // b52-i
generateBar.play(1, 1, 90, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0);      // b52-j
generateBar.play(1, 1, 90, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0);      // b52-k
generateBar.play(1, 1, 70, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0);      // b52-l
generateBar.play(1, 1, 90, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0);      // b52-m
generateBar.play(1, 1, 90, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0);      // b52-n
generateBar.play(1, 1, 40, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2);      // b52-o


//Material from first subject
generateBar.play(4, 2, 90, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 7);      // b53
generateBar.play(4, 4, 180, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 5);      // b54
generateBar.play(4, 2, 90, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 7, 0);      // b55
generateBar.play(4, 4, 180, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 5, 0);      // b56
generateBar.play(4, 1, 90, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);      // b57
generateBar.play(4, 2, 180, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);      // b58
generateBar.play(4, 1, 90, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2);      // b59
generateBar.play(4, 2, 180, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2);      // b60

//Material from second subject
generateBar.play(4, 2, 90, 2, 0, 0, 5, 0, 5, 0, 4, 0, 0, 4, 0, 0, 0);      // b61
generateBar.play(4, 4, 180, 1, 0, 1, 0, 5, 0, 3, 0, 2, 0, 4, 0, 0, 1);      // b62
generateBar.play(4, 2, 90, 1, 1, 1, 4, 5, 0, 4, 0, 0, 4, 3, 0, 3, 0);      // b63
generateBar.play(4, 4, 180, 0, 0, 1, 5, 0, 5, 0, 2, 2, 0, 3, 0, 0, 0);      // b64
generateBar.play(4, 1, 90, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2);      // b65
```

```
generateBar.play(4, 2, 180, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2);     // b66
generateBar.play(4, 1, 90, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2);      // b67
generateBar.play(4, 2, 180, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2);     // b68

//CODA
generateBar.play(1, 1, 60, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);      // b69-a
generateBar.play(1, 1, 60, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);      // b69-b
generateBar.play(1, 1, 60, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6);      // b69-c
generateBar.play(1, 1, 30, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);      // b69-d
generateBar.play(1, 1, 60, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6);      // b69-e
generateBar.play(1, 1, 60, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);      // b69-f
generateBar.play(1, 1, 60, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0);      // b69-g
generateBar.play(1, 1, 60, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0);      // b69-h
generateBar.play(1, 1, 40, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0);      // b69-i
generateBar.play(1, 1, 180, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0);     // b69-j
```