hyperboles are the worst thing ever
(aka Epimenides would have been gutted)
for flute and computer

Michael Edwards
michael@sumtone.com

October 4, 2014

Programme Note

Humans exaggerate on a regular basis. Typical hyperboles might be "this bag weighs a ton", "I'm so hungry I could eat a horse", or any of the invariably (!) hilarious "Yo' mama so fat…" jokes. Perhaps even worse than footballers ("I hit the post. I was gutted") are artists. Have you ever sat silently suffering ("dying", anyone?) at a contemporary poetry reading, as the reciter over-emotes their way through a litany of subtexts we can summarise by "me, me, me! I'm so deep and clever!"? Such occasions often merely reinforce the popular perception of artistic outputs as being expressive of the creator's emotions. But that is less interesting than artworks' invitation to be social, communal, and at the same time to introspect and inspect our personal, perhaps emotional reactions to intrinsically neutral objects:

When faced with a Nitsch picture of animal guts resting on a man's genital area, do we feel horror and offence or see the skin as a mirror, a symmetrical inversion of nothing more than what's behind the man's (and by implication our) skin at all times, sobering as that may be ("hello mortality!")?

Whilst on the subject of male genitalia, do Mapplethorpe's motor oil crotch smearings (accompanied by tight testicular ties) excite homophobic panic or amazement at the textures and play of light and shadow?

The choice (to grow) is ours.

So, at the risk of being called a Cretan, here's a piece without any emotional content whatsoever.

Background

This is a flexible, musician-tweaked algorithmic piece originally written for flute and computer in 2013/14 but since developed and extended for other solo and group instrumental combinations with or without the computer. There is no permanently fixed score as such; rather, a score is generated by the underlying software via configuration of the parameter space presented by a separate software interface. There is no requirement to do this on the part of the musician(s) however as a pre-configured version can be delivered by the composer.

The generative ideas then are the crux of the piece, rather than the details of any one score generated by them. Or to put it another way, the fixed score is not the essence of the music but a tool towards reaching one possible realisation of the ideas as expressed in software—I'm just swapping one code for another, really.

The objective of this approach is the algorithmic[1] specification of global structure, parameters, tendencies, data, and data processing techniques which are then used to lead to different versions of the piece. These, though all different, share identifiable common characteristics.

[1] The algorithms are deterministic, i.e. there is no randomness here.

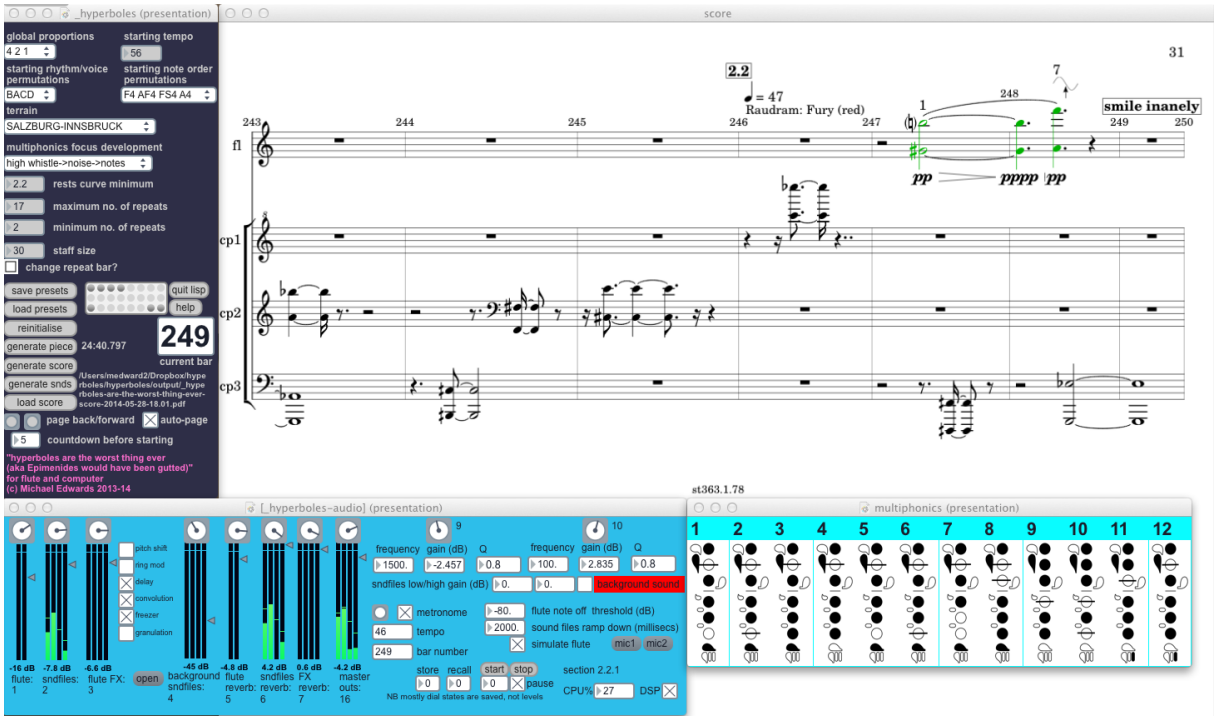

Figure 1: The algorithm interface

Structure

The formal structure of the piece is generated by a recursive process expanding outwards from three simple proportions, either 4:3:2, 3:2:1, or 4:2:1 (others could be possible but these produce suitable durations of c. 52, 15, or 25 minutes respectively, accepting the other default parameter values). The depth of recursion in this process is theoretically unlimited but we use two generations for this piece so that durations are manageable. Using the 4:3:2 proportions we can see how the section lengths unfold accordingly:

```
((((1) 324)
  (((1 1) 81)
   (((1 1 1) 36) ((1 1 1 1) 9) ((1 1 1 2) 9) ((1 1 1 3) 9) ((1 1 1 4) 9))
   (((1 1 2) 27) ((1 1 2 1) 9) ((1 1 2 2) 9) ((1 1 2 3) 9))
   (((1 1 3) 18) ((1 1 3 1) 9) ((1 1 3 2) 9)))
  (((1 2) 81)
   (((1 2 1) 36) ((1 2 1 1) 9) ((1 2 1 2) 9) ((1 2 1 3) 9) ((1 2 1 4) 9))
   (((1 2 2) 27) ((1 2 2 1) 9) ((1 2 2 2) 9) ((1 2 2 3) 9))
   (((1 2 3) 18) ((1 2 3 1) 9) ((1 2 3 2) 9)))
  (((1 3) 81)
   (((1 3 1) 36) ((1 3 1 1) 9) ((1 3 1 2) 9) ((1 3 1 3) 9) ((1 3 1 4) 9))
   (((1 3 2) 27) ((1 3 2 1) 9) ((1 3 2 2) 9) ((1 3 2 3) 9))
   (((1 3 3) 18) ((1 3 3 1) 9) ((1 3 3 2) 9)))
  (((1 4) 81)
   (((1 4 1) 36) ((1 4 1 1) 9) ((1 4 1 2) 9) ((1 4 1 3) 9) ((1 4 1 4) 9))
   (((1 4 2) 27) ((1 4 2 1) 9) ((1 4 2 2) 9) ((1 4 2 3) 9))
   (((1 4 3) 18) ((1 4 3 1) 9) ((1 4 3 2) 9))))
 (((2) 243)
  (((2 1) 81)
   (((2 1 1) 36) ((2 1 1 1) 9) ((2 1 1 2) 9) ((2 1 1 3) 9) ((2 1 1 4) 9))
   (((2 1 2) 27) ((2 1 2 1) 9) ((2 1 2 2) 9) ((2 1 2 3) 9))
   (((2 1 3) 18) ((2 1 3 1) 9) ((2 1 3 2) 9)))
  (((2 2) 81)
   (((2 2 1) 36) ((2 2 1 1) 9) ((2 2 1 2) 9) ((2 2 1 3) 9) ((2 2 1 4) 9))
   (((2 2 2) 27) ((2 2 2 1) 9) ((2 2 2 2) 9) ((2 2 2 3) 9))
```

```
  (((2 2 3) 18) ((2 2 3 1) 9) ((2 2 3 2) 9)))
 (((2 3) 81)
  (((2 3 1) 36) ((2 3 1 1) 9) ((2 3 1 2) 9) ((2 3 1 3) 9) ((2 3 1 4) 9))
  (((2 3 2) 27) ((2 3 2 1) 9) ((2 3 2 2) 9) ((2 3 2 3) 9))
  (((2 3 3) 18) ((2 3 3 1) 9) ((2 3 3 2) 9))))
((((3) 162)
 (((3 1) 81)
  (((3 1 1) 36) ((3 1 1 1) 9) ((3 1 1 2) 9) ((3 1 1 3) 9) ((3 1 1 4) 9))
  (((3 1 2) 27) ((3 1 2 1) 9) ((3 1 2 2) 9) ((3 1 2 3) 9))
  (((3 1 3) 18) ((3 1 3 1) 9) ((3 1 3 2) 9)))
 (((3 2) 81)
  (((3 2 1) 36) ((3 2 1 1) 9) ((3 2 1 2) 9) ((3 2 1 3) 9) ((3 2 1 4) 9))
  (((3 2 2) 27) ((3 2 2 1) 9) ((3 2 2 2) 9) ((3 2 2 3) 9))
  (((3 2 3) 18) ((3 2 3 1) 9) ((3 2 3 2) 9)))))
```

The sum of the proportions defines the basic length of the rhythmic sequences (or phrases: see below) used in the piece. So 4:3:2 produces 9-bar sequences. These sequences are then organised in groups of four, three, and two 9-bar sequences, as we see above, with this combination forming a single higher-level grouping of a subsection which is then repeated, e.g. the subsections labelled (1 1) (section 1, subsection 1), (1 2) ... (3 2) in the above. Each subsection is of exactly the same 81-bar duration (9x4 + 9x3 + 9x2 bars). The number of subsections in each section is then determined by the same basic proportions: 4:3:2. A similar process is undertaken with different proportions, e.g. 3:2:1 would involve 6-bar sequences organised into 36-bar subsections, hence the selected proportions determine the overall duration quite directly.

Rhythm Sequences and Terrain Data

The rhythm sequences used are pre-defined as three single-sequence groups of four-part counterpoint (originally one for the solo instrument and three for the computer). These are mapped onto the instruments over the duration of the piece via a simple permutation routine, using only a single group of four-part counterpoint per section. However, the choice of whether a computer voice will play or not depends on a curve provided by the elevation data of a road journey as input from Google Maps. Currently three journeys are available: Salzburg to Mittersill and Salzburg to Innsbruck (Austria), and Banff to Lake Louise (Alberta, Canada):

This terrain data is stretched over the duration of the piece and used for a variety of further processes in the piece:

1. To add rests to the four parts: Though the same rhythm sequences are used and re-used in several permutations throughout, the notes in these are processed by a method which turns notes to rests according to an activity-level algorithm driven by the current elevation on the journey.
2. Certain bars in the rhythm sequences are 'stuck on' (i.e. repeated) at the end of every sub-subsection (see structure extract below). The number of repeats at a given point in the piece is determined by the terrain data so that the higher the elevation the more repeats there are (we could think of this as pausing to admire the view).

```
(((1) 324)
 (((1 1) 81)
  (((1 1 1) 36) ((1 1 1 1) 9) ((1 1 1 2) 9)  ((1 1 1 3) 9) ((1 1 1 4) 9))
                                                  repeats in here ^
```

3. From section two, the first rhythm sequence allows improvisational intervention (see below). Whether this extends into the second or third written sequence depends on the terrain data curve. A higher elevation implies more improvisation.
4. Many of the notes in the instrumental parts will include a symbol indicating a manner of playing. For example, the flute version includes different multiphonic spectral foci (see below). The focus—or another method of playing in a different version of the piece—changes more quickly when at a higher elevation in the terrain data.

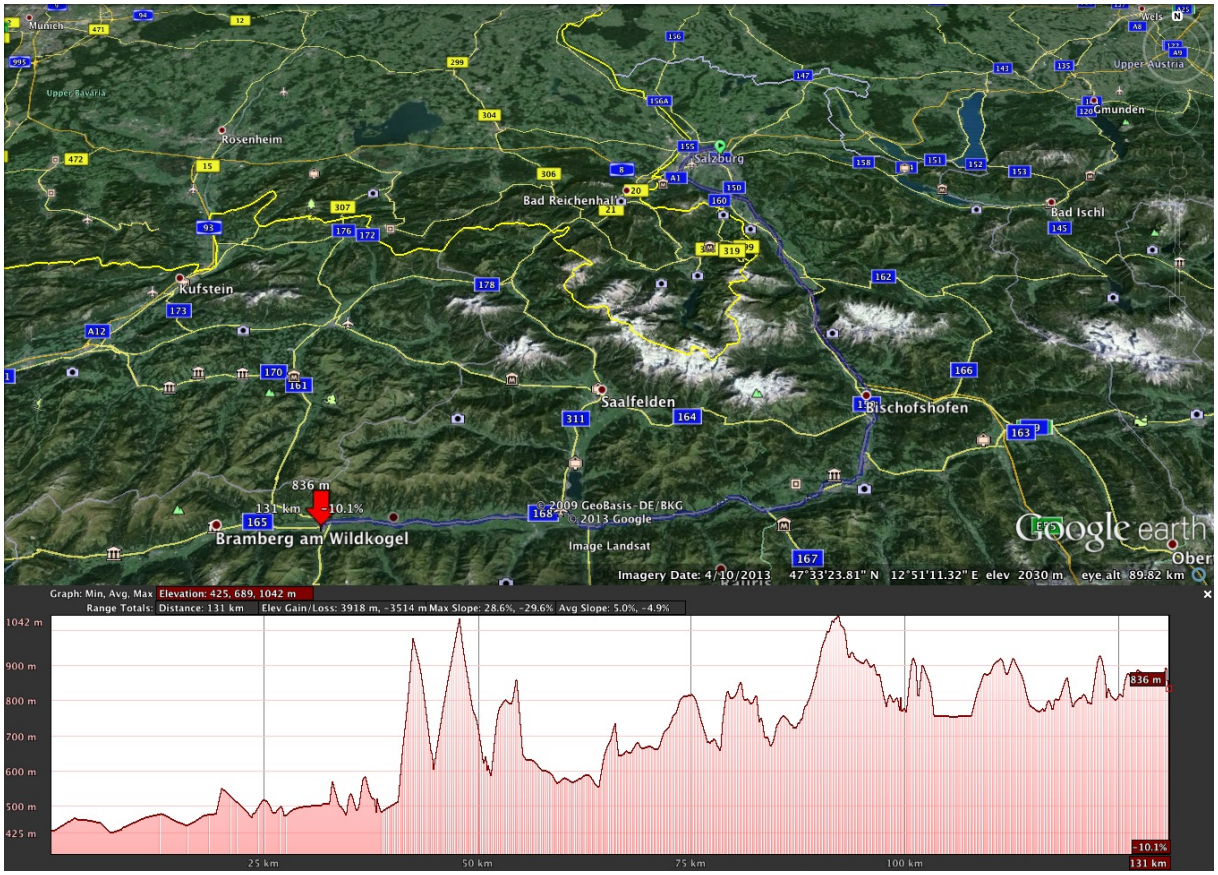

Figure 2: Salzburg to Mittersill

5. Though dynamics are quiet throughout, higher elevations will also result in slightly higher overall dynamics.

Score Interpretation: General Points

1. Red notes: unstable sonorities. These are so quiet that for the notated duration we move between unstable, extremely quiet sounds and silence, as the performer tries to make audible the most tiny sounds possible. These are used at the beginning of each section.

2. Green notes: These indicate parts of the piece where the performer may deviate from the score with improvisational interventions. The performer may stick to or deviate from the indicated rhythms/pitches/dynamics as they see fit, but materials inserted here should not interrupt the character of the piece too much and definitely should not rise above the prevailing dynamic. E.g. in the flute version, as several of the sound files contain amplified mouth noises, these may be a good addition to the textures in these improvised interventions.

3. Pauses may be inserted in the performance at any point; pressing the space bar halts the MIDI sequencing (see below).

4. Each section and subsection (i.e. 2 and 2.2 but not sub-subsections such as 2.2.2) is accompanied by an Indian Rasa and associated colour. The Rasa, or mental state, should be internalised by the performer(s) and used to set the mood of the music that follows. This should be a subtle effect however. At no point should histrionics or extreme movements or dynamics be used to portray the Rasa, even in the improvised interventions. Where possible, the colours should be used in the selection of light projected onto the stage and/or performer. A light mixer with RGB (red, green, blue) and white faders is useful here if no light automation is to be used. The colour black can be interpreted as no coloured light; grey then is a little white; yellowish is a little red and green in equal levels, with yellow being more of both. The iPad Mira software (see below) will show when

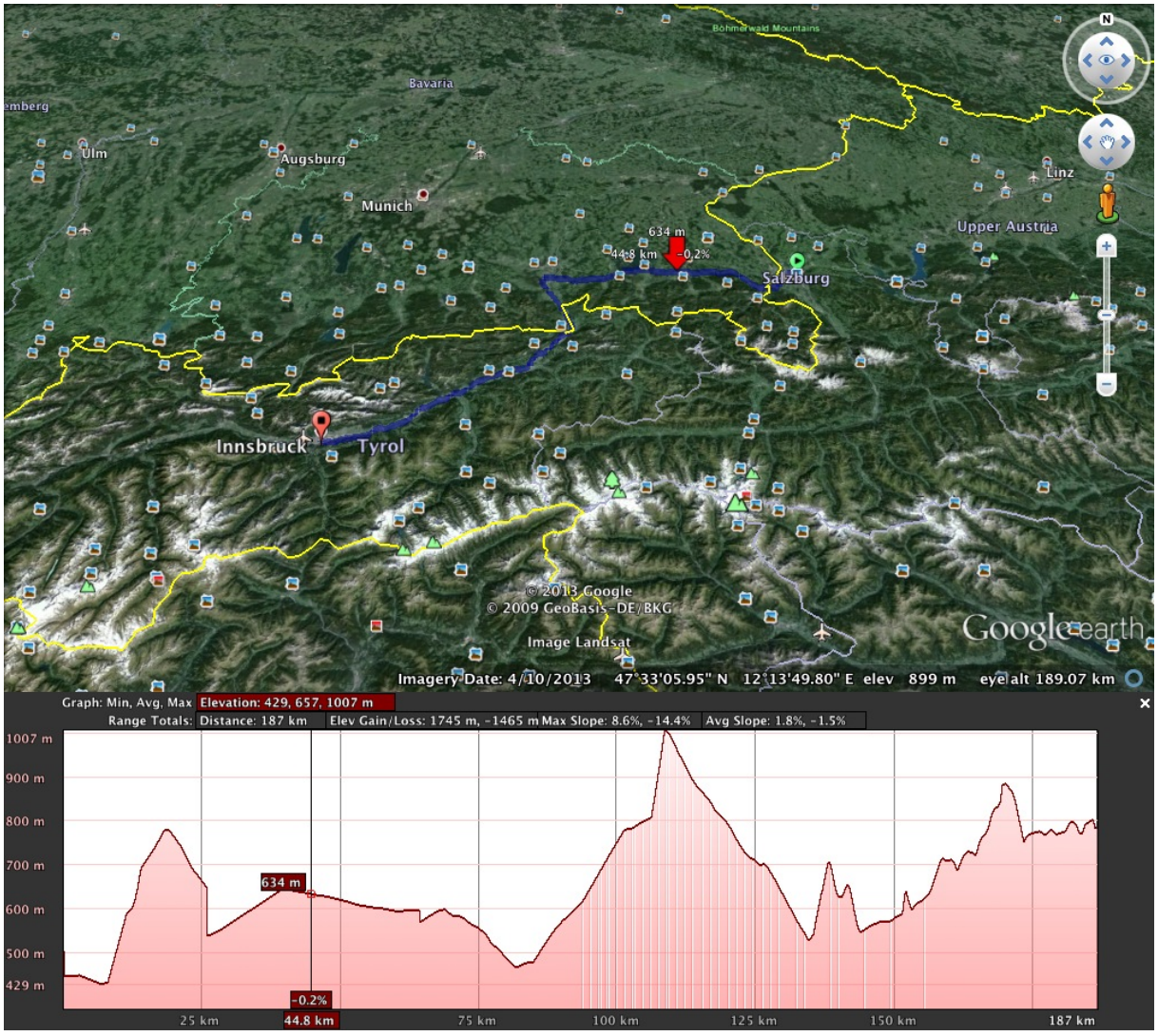

Figure 3: Salzburg to Innsbruck

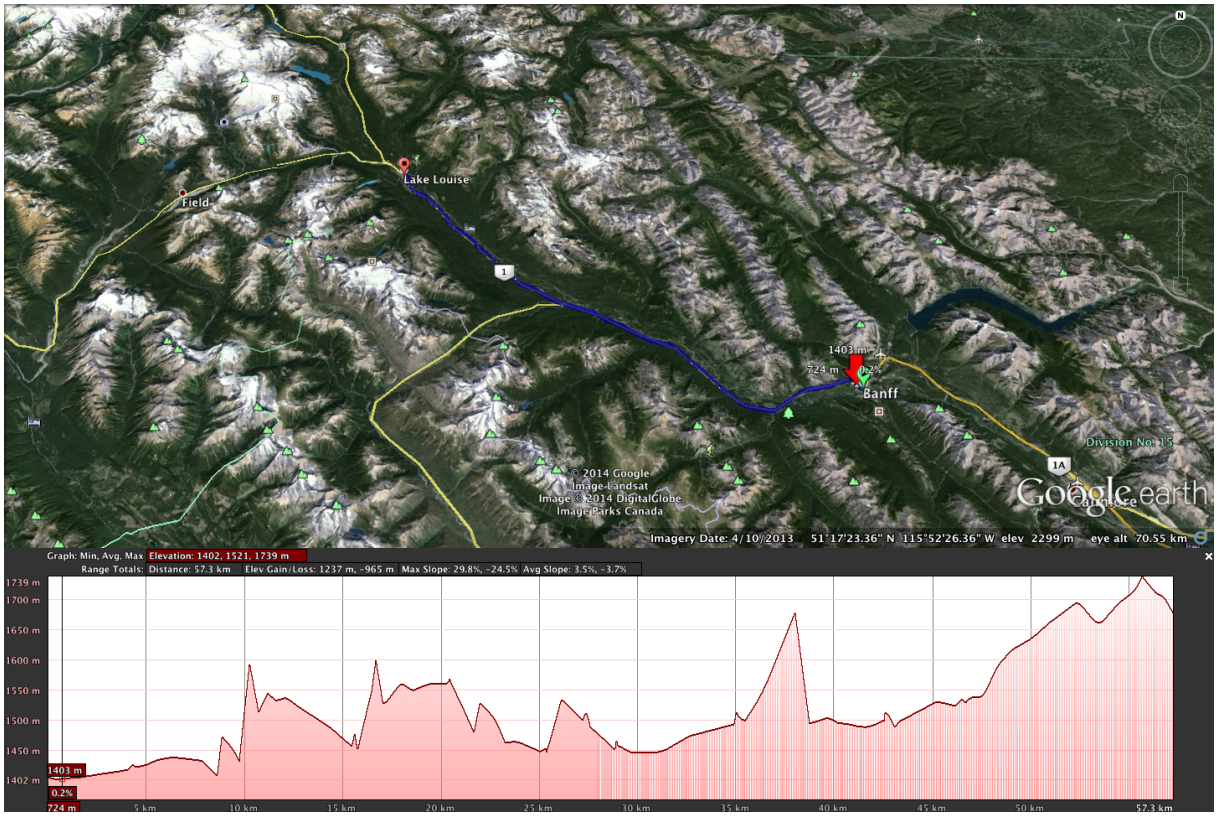

Figure 4: Banff to Lake Louise

these colours are to change (or timings/cues can be used instead). Once the colour changes, a slow fade should be set in motion: please do not change colours abruptly.

5. Versions for solo instrument and computer:

    1. Following the example of (or perhaps unashamedly stolen from, depending on your point of view) Samuel Beckett's "Not I", a narrowly-focussed spotlight should be aimed at the performer's mouth (no more than this) throughout the performance. However, the performer may move in and out of this light *ad lib*, i.e. they are not expected to keep their mouth in the light beam throughout the complete performance. The colour of this light may change along with the Rasa (see above) or remain white, according to the desires of the performer/set designer and the resources of the performance space. In the premiere (Splendor, Amsterdam, 14th June 2014; Anne La Berge, flute) we used a white halogen light bulb taped to the microphone stand, with a little tape blocking out light leakage to the audience and focussing the beam onto the performer's face when she stepped into it.
    2. At three points in the piece a mouth action instruction is given in a box ("bare teeth", "stick out tongue", "smile inanely"). In each case this should be performed with the mouth in the spotlight in a neutral, clear, but unexaggerated manner. These instructions have no intended meaning whatsoever, rather, they may or may not provoke a variety of responses in the listener, depending upon their sympathies towards the performance and their individual experiences and characteristics.
    3. At the end of the piece, after a significant pause, the player is invited to ask the audience "Is that enough, or do you want more?". There is a sound file of Anne La Berge asking this question; this can be triggered by the performer by pressing the 'i' key on the computer, if they prefer. What to do if the audience requests more is left to the discretion of the performer.

Flute-Specific Notes

Multiphonics

The pitch structure of this piece is formed by transitions between a user-selected permutation of the fundamental pitches F, F#, G#, and A, as multiphonics, followed by low C and high G. For each of the four fundamental pitches there are two possible multiphonics (or in the case of G#, six) which will be cycled through as each pitch is selected. This results in a total of 12 multiphonics, as shown below.

Either the multiphonic number or the fingering is shown above the notes in the score; the fingering graphics can be displayed throughout the performance by clicking on the 'flute fingerings' button in the audio control window to display these. Due to limitations of the current MaxMSP environment (6.1) the zoom level for this window does not display all 12 fingerings; however they will be correctly shown once the 'start' button (or 's' key) is pressed.

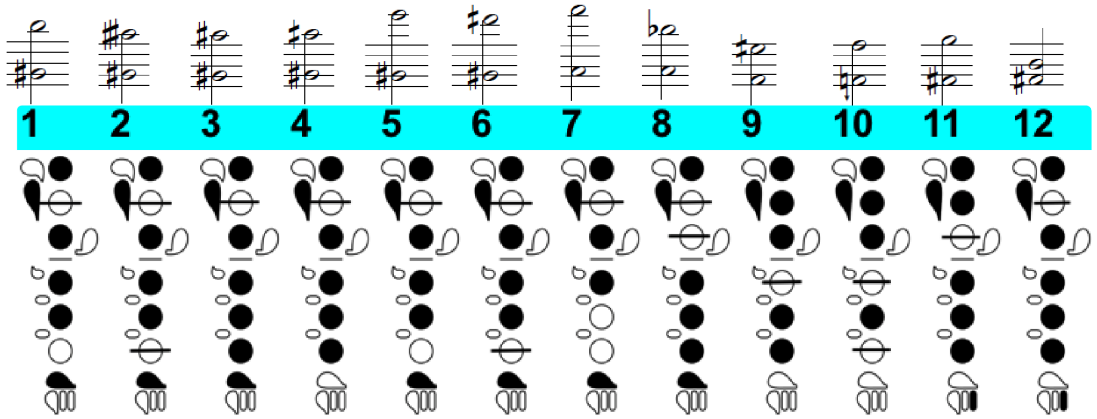

Figure 5: Flute multiphonics

Score Interpretation

1. Purple notes: These notes are to be sung gently along with the indicated multiphonics/single notes. If a sung note is difficult to execute, for example in conjunction with the G6 harmonics, establish the note(s) first then add the voice tentatively and explore the resulting sonority.
2. Diamond note heads indicate the fundamental note from which the high G should be played from. This will be one of low C (C4), E flat, G, or the 2nd C (C5).
3. In the body of the multiphonics there are three spectral areas which the performer should try to bring out according to the indicated context:
    1. the lower part of the spectrum, i.e. the notated pitches (indicated by normal noteheads);
    2. a higher noise band (indicated with an erratic scribbled line over the notes);
    3. a very high whistle tone (indicated by a sine wave with an upward pointing arrow).

One spectral area is focussed upon in each of the three sections, the order of which is user-selectable. The other two spectral areas will be used to a lesser degree in each section, with more changes of focus as the elevation increases (see Terrain Data above). Please note that at no point should we hear an ascending/descending whistle tone harmonic series, for example, on low C when trying to achieve the high whistle tone or any other tone for that matter. ## Essential Hardware for versions with computer

1. Macintosh running OSX 10.7.5 or above
2. High-quality external sound card with 4-channel output (stereo also possible)
3. High-quality condensor mic(s)

It is envisaged that solo performers will read the score from the computer screen and thus be able to follow the bar numbers and other interface elements as the performance proceeds. The software will turn pages automatically if desired (see below).

Optional Hardware

1. Additional microphones, clip-on or otherwise
2. iPad (retina display) running Mira software (from cycling74.com; see below) *or* faderfox FX3 or similar MIDI controller

The piece may be performable without the help of another person but for generally it would be useful to have someone balance levels during the performance with either a MIDI controller or an iPad.

iPad + Mira The iPad Mira interface is particularly useful for controlling the levels during performance as it allows someone in an audience position to do this wirelessly. The hyperboles.app software sends two frames to Mira: one for the lighting control, the other for audio level balance (so two iPads might be useful if the lighting engineer is remote from the person mixing).

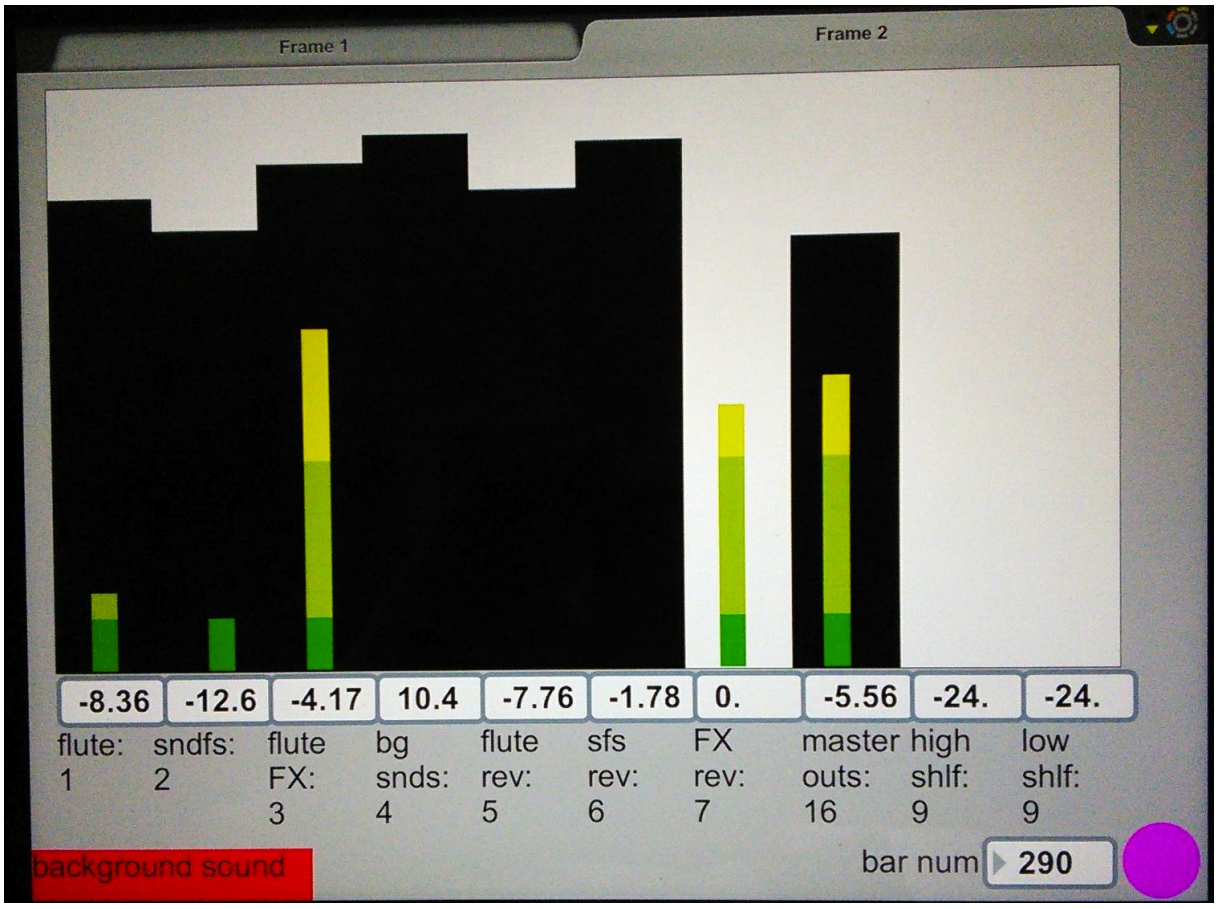

Figure 6: iPad Mira control

As with the MIDI faders, once the initial preset is loaded, the interface dial might be out of line with the MIDI controller and/or Mira fader. In order to avoid sudden parameter changes, the software dials will not reflect the MIDI controller or Mira fader levels until the latter reaches the stored value.

Particular attention will need to be paid to the sound files and the instrumental FX levels during performance as either of these might— depending upon the configuration and performance context— become too loud.

Essential Software for versions with computer

This piece is dependent on a number of different software packages. It currently runs only Macintosh computers; it is optimised for a screen resolution of 1280x800 pixels. If you just want to play a pre-configured version of the piece all you will need is hyperboles.app which this documentation forms a part of.

The essential software infrastructure is based on communication between MaxMSP and the composer's algorithmic composition package *slippery chicken*, written in the Common Lisp programming language. If you wish to experiment with the underlying algorithms and generate your own version of the piece you will need hyperboles.app and *slippery chicken* both. You can download the *slippery chicken* app (free) at: http://www.michael-edwards.org/sc/source.html (Please read the _readme.txt file which comes with the app in order to download further software necessary for the installation.) Once the app is installed and running please update to the latest software release by typing (update-app-src) at the Lisp prompt (see http://michael-edwards.org/sc/robodoc/utilities_lsp.html#utilities2fupdate2dapp2dsrc for more details).

You will also need the free and open source music notation software Lilypond: http://download.linuxaudio.org/lilypond/binaries/darwin-x86/lilypond-2.17.96-1.darwin-x86.tar.bz2 or any later version of your choosing.

Running the Software

If you only want to perform the pre-configured version of the piece just start hyperboles.app after making sure it is configured to run in 32-bit mode: if you click on the file in the Finder, "apple i" will show the inspector and you can click on "Open in 32-bit mode" there.

Once hyperboles.app has started, click the open button at the bottom left next to DSP, and make sure the sampling-rate is to 48KHz. Select the instrument version from the big bold drop-down menu at the top. When you are ready to begin, press the 's' key to start the countdown to the beginning of the piece. (You might also want to check the Max window to make sure no errors have occurred. There should be "sflist~: can't find file..." errors, though you might see these when configuring a new piece.)

If, on the other hand, you want to "roll your own" (i.e. make your own version of the piece), start the *slippery chicken* app. Once you get to the SC> prompt, type the following Lisp call:

```
(osc-call)
```

Now start hyperboles.app, and set the sampling-rate to 48KHz as detailed above. Press the 'algorithm tweaks' button and select a preset, etc.: see the instructions below. Once you're happy press the **generate piece** button. The patch window will turn red whilst Lisp is doing the necessary calculations. Once the patch is back to its original colour, press the **generate score** button. Again, the patch will turn red while Lilypond renders the score PDF. Once it is ready, you will see the score in the window to the right of the main patch. You can turn pages by using the buttons on the patch or the left and right arrow keys. Before you can perform the piece you will also have to render the sound files by pressing the 'generate snds' button. Each time you start hyperboles.app it will load the last generated version of the piece.

Algorithm Interface Details

The **global proportions** menu sets the proportions of the global structure (see the Structure section above).

The **starting tempo** sets not only the tempo with which the piece begins but the tempo from which all other tempi are proportionally derived.

The **starting rhythm/voice permutations** menu sets the rhythm sequence permutation which begins the piece. Changing this will result in different rhythmic material being allocated to the solo instrument and three computer parts throughout the piece.

The **starting note order permutations** refers to the ordering of the basic pitches which form the background to the piece. For instance, in the flute version, the pitches in the drop-down menu are the lowest notes of the multiphonics we use. A4 refers to middle A, FS4 is the F sharp below, GS4 is G sharp, and F4 is F. So the first note will form the fundamental starting multiphonic of the piece. In all versions, a transition takes place over the duration of the piece through the ordering of these notes and then on through to some single pitches, low C and high G in the flute version. For each of the four given pitches there are several ways of playing them which will be cycled through as each pitch is selected in sequence (e.g. in the flute version, two to six possible multiphonics for each basic pitch).

The **terrain** menu sets the elevation curve data that will be used in several processes and as described in the Terrain Data section above.

The **focus transition** menu determines the order of the manner of playing the notes/multiphonics for the three sections; in the flute version this means the dominant multiphonic spectral areas. See the Terrain Data and Score Interpretation sections above for more details.

The **rests curve minimum** value will set the minimum value of the scaled terrain curve when it is used to turn notes into rests (as described in the Terrain Data section above). The lower this value is, the more rests will be inserted (in particular at the beginning).

The **maximum no. of repeats** and **minimum no. of repeats** values set the min./max. number of repeated bars 'stuck on' at the end of every sub-subsection. See the Terrain Data section above.

The **change repeat bar** tick box determines whether the bar to be repeated will change on a sub-subsection boundary or not. The effect of this will only be noticeable once the number of repeats is high enough (i.e. longer than a complete rhythm sequence, e.g. towards the end of the piece).

The **presets** section allows you to save interface configurations. Once you have set the parameters, you can hold the shift key and click on any of the little round buttons in order to save state in that place. To recall a state, you simply click on the associated button. You will be prompted to save the presets file if you try and quit without saving. You can also save your presets in a file of your choosing by pressing the **save presets as** button. Similarly, you can load a previously saved presets file by clicking the **load presets** button. The **save presets** button saves to the default file for the current instrument; this will be reloaded automatically upon starting the patch. To re-initialise the piece to the values determined by the composer and specified in the hyperboles.lsp file, click the **reinitialise** button.

The **staff size** number controls how large the generated notation will be. The higher the number, the larger the notes.

Once you have configured the piece, you must click on the **generate piece** button to create the notes and other data, as well as the MIDI file. You can then choose to generate the score—this will be displayed in the score window once rendered by Lilypond.

Click on **generate snds** in order to generate the sound files for the piece. These may or may not have already been generated, depending on whether you're creating a new configuration or not, but only those which have not been generated will be created here. Note that this is a computationally intensive operation so may take a long time, perhaps 20 minutes. Note that the sound files are all generated from samples kindly provided by the following musicians:

1. Flute: Anne La Berge, Amsterdam, August 18th 2013
2. Violin: Mieko Kanno, Edinburgh, June 30th 2014
3. Soprano saxophone: Gianpaolo Antongirolami, Macerata, Italy, June 22nd 2014

The samples are also used in the simulation of the solo part in the different versions of the piece. They are not to be repurposed without the express permission of the performers and the composer.

When the score is rendered as a MIDI file, the solo part will be on channels 1 (with microtones on channel 2) and the computer parts spread over channels 3-8.

The **load score** button allows you to display a previously generated score PDF file in the score window.

You can change pages in the score by clicking on the two buttons at the bottom of the interface or by using the left and right arrow keys. Alternatively, MIDI notes 0 and 1 on channel 13 (the first two green buttons—top left—on a faderfox FX3 MIDI controller) will perform page back/forward. As the performance is driven by a MIDI sequencer (see below), we can have the patch automatically turn pages for us by clicking the **auto-page** toggle. Of course, each slight variation in algorithmic parameters will result in a different score. As it's not possible to know in advance how Lilypond will lay out the bars, in order for automatic page turns to happen, click the **enter page turn bars** and edit the bar number list to reflect the first bar number of each score page.

The **init lisp** button loads the Lisp source code for the piece and initialises the environment for the generation of all the data files, etc. This must be pressed before any of the presets are loaded and the piece is rendered.

The **quit lisp** button disconnects the MaxMSP<–>Lisp connection and returns you to the Lisp prompt in the *slippery chicken* application. If you wish to resume this connection, simply type (osc-call) in the Lisp interpreter once more.

In order to start the piece, click the 'start' button on the audio patch or press the 's' key; to stop it, click the 'stop' button (this will also reset levels and other parameters to preset 1). You can pause the performance at any point by pressing the space bar. **countdown before starting** sets the number of seconds the software will wait before actually starting. You can see the countdown happen in the audio window, just below the start button.

The audio patch

The blue audio patch allows control over various aspects of the real-time audio processing in the piece. Sequencing of real-time effects and sound files is done via a MIDI file which begins once the 'start' button is pressed and the countdown is over. MaxMSP does not allow much control over MIDI file playback so for rehearsal purposes I would suggest playing the MIDI file (found in the output folder) via software such as Rondo (http://www.fracturedsoftware.com/rondo/). This allows scrolling through the piece to start at any arbitrary bar (they are numbered in Rondo's interface) and if the output of Rondo is set to IAC Bus 1 (via the Sequence->Destination menu), then the MaxMSP patch will pick up the MIDI notes and trigger the sound files and effects presets in all the right places. Whether using Rondo or not, the solo instrument simulation can be turned on and off via the 'simulate solo' toggle; clearly it should be off (cross not visible in the toggle box) for live performances (its state is saved in the presets).

The dials along the top of the audio window reflect MIDI faders/dials on channel 13: controller numbers 0-3 (labelled 1-4 for convenience) control levels for the solo instrument, sndfiles, solo DSP (effects aka FX) and background sound files respectively; numbers 4-6 control the reverb levels for the solo instrument, sound files, and solo DSP: all reverb levels are post-fader e.g. if the solo signal is attenuated by 6db on controller 0 and the solo reverb by 3db on controller 4, then the solo level sent to the reverberator is attenuated overall by 9db.

Controller 15 controls the master gain. Controller numbers 8 and 9 control the gain on high and low shelf filters applied to the scored computer parts, i.e. the sound files under controller 1. In addition, gain levels for high and low sound files can be entered in the two number boxes below the filters; the division point is middle C in the computer parts of the score.

Two further number box controls allow the setting of:

1. the solo plus FX threshold below which the signal must fall before the FX will be changed (solo FX are changed automatically on-the-fly yet changing whilst the soloist plays a note, or the FX are still ringing, might cause clicks or other disturbances, hence we wait until the signal drops below the threshold before changing);
2. the ramp down time for the sound files triggered via the score. The ramp starts at the point in the score where the note stops, which means that in the computer performance the sound files hang on over the rests. Also worth mentioning is that the computer parts' pitches in the score will generally be reflected in the sound files: they are related to the soloist's nearest pitches via proportions from the harmonic series. However, shorter notes are generally played back as short percussive effects; these may deviate considerably from the scored pitch.

The two 'mic' buttons will open controls for two microphone inputs for the soloist (on inputs 1 and 2). The controls for these should be familiar and/or self-explanatory. The two signals can be panned and processed separately but are mixed to mono before being sent to the effects.

The tempo and bar numbers are indicated along with a metronome, which can be turned off via the toggle.

Solo FX The toggles here are read-only, i.e., they show the currently activated real-time FX being applied to the solo instrument signal and whose combined gain is set by controller 2. The 'open' button below the FX toggles will reveal the FX patch; changes can be made here and the individual effects levels can be viewed and changed but it is not envisaged that changes should be made just prior to or during a performance.

Presets The 'store' and 'recall' numbers allow various preset settings to be stored in a file and recalled in a later session. If you store any presets you will be prompted to save the file once the patch is closed. Note that preset 1 is loaded from the file hyperboles-flute.json or similar file reflecting the instrumental configuration for the version you are playing (e.g. hyperboles-violin.json). This is found in the data directory, and is loaded by default at startup. This file should therefore be used for local performance settings. Feel free to overwrite the file with new settings (though you might want to back up the original). Also note that interface dial values will override fader levels (which are controlled by the dials) and should therefore be set properly before storing a preset.

N.B. Once a preset is loaded, the interface dial might be out of line with the MIDI controller/Mira. In order to avoid sudden parameter changes, the software dials will not reflect the MIDI faders/dials levels until the latter reaches the stored value.

Background sound files The three computer parts are combined onto the single 'sndfiles' fader along with the glissandi sound files which start the piece and subsequent sections. (The latter are notated in the score by the words [gliss. sound] above the top computer part; they begin on every section and subsection, i.e. 1, 1.2, 1.3, 1.4, 2, 2.1 etc.)

In addition, on the second sub-subsection in every subsection (i.e. 1.1.2, 1.2.2, 1.3.2 etc.) a background sound is triggered; this is indicated in the score by the words [background sound]. This is a generally quiet and/or distant sound which may be mixed in at more or less low levels in order to provide some textural variation. This is desirable but not essential. When a background sound is playing, it will be indicated in green on the audio interface; when not playing, it will be red.