LHCSynth Application: Version 0.4 (alpha)

NB: this software is at a very early stage of development. Bugs are to be expected (any one or more of which may result in a crash). It is likely to change considerably as it is developed further. Feedback from users will directly influence its development.

Introduction - what, why, and how?

The purpose of this program is to read text data files containing columns of numbers, obtained from particle collision data from the Large Hadron Collider (LHC) at CERN, and turn them into sound. This is a process termed "sonification". In principle, the program can be used to sonify any suitably formatted data file - football results, the stock exchange, weather patterns, and so on. However the focus both of the program and of this documentation is on the data files from the LHC, and specifically on data from the ATLAS detector - one of four general-purpose detectors at the LHC. The LHC, like other similar particle colliders around the world, is trying to fill in the many gaps in the physicist's "Standard Model" describing how the universe is made (in terms of atomic and sub-atomic particles, such as electrons, protons, neutrons, quarks, gluons, and so on). The LHC can be understood as a huge camera – the scientists are trying to get a full picture of exactly what happens in a collision – except that in this case the camera does not directly produce images but seemingly endless streams of numbers.

Unfortunately, those gaps in the Standard Model cannot be filled simply by developing theories. Physicists need some "hard numbers", such as how massive a given particle is, and the current theories don't really offer much of a clue. These values have to be determined by experiment, by smashing particles together at extremely high energies, and inspecting the results. Physicists have proposed many ideas as to what these missing pieces might be. The key test of a theory is that it both explains currently observed behaviour, and (more importantly) accurately predicts behaviour not yet seen. Only by performing experiments of this kind can any of these proposed explanations of how the universe works be confirmed; and if some new observation is made that contradicts the theory, the theory has to be changed - or thrown out altogether.

The LHC is looking, among other things, for the Higgs Boson (the so-called "God Particle"), without which the mass of all the matter in the universe is currently inexplicable! There are a few other theories trying to fill this particular gap, but the Higgs particle is the "leading contender".

The problem for the scientists working on all this is that the ATLAS detector alone generates a *huge* amount of data - enough to fill a pile of CDs 2km high, every year! So the analysis of all that data is a formidably complex and demanding task. The normal tools of the physicist are mathematical, and occasionally graphical - processing the numbers in sophisticated ways to discover patterns that may be clues to "new physics". The idea behind sonification is that sound can also be used to analyse the data - the human ear being extremely good at detecting subtle patterns in a sound. The challenge is to find the right "recipe" - the best way of turning the data into sound, to reveal the patterns the physicists are interested in.

The sounds already published by the LHCsound project have proved very popular with composers. So, as well as the scientific goals, there may be exciting opportunities for music composition using the results of a sonification, whether or not it reveals any new physics.

How it works.

LHCSynth works as a sort of translator. It reads a data file containing columns of numbers, and writes a new text file that can be understood by *Csound* - a popular and powerful music synthesis program. The task of the user is to decide, for each column of data in the file, how to use the numbers to control one or more of the primary elements of music and sound - timing, loudness, pitch, tone colour (*timbre*), and space (stereo position). The program generates a "note list", where each line of data is processed as a single note.

A key element of this task is the use of one data column to control one or synthesis parameters for the sonification of some other column in the file. Thus, one data column may be rendered directly as sound, others will not be but are instead used to control the synthesis for that column.

The basic operation of the program is simple. You will start by loading a data file containing one or more columns of data (up to a maximum of eight) and, after choosing voicing settings, write a Csound "CSD" file (a combined "orchestra" and "score"), which can be directly rendered either to a soundfile or (sometimes) played as real-time audio by any current version of Csound. The full LHCsound distribution includes versions of Csound for both the Mac and the PC. The nature of the data generated by the LHC is that the numbers often cover an extreme range. LHCSynth provides a simple means to compress this range to one that can be applied effectively to sound synthesis.

There are facilities to store the settings you have used for a particular sonification, and import them to process another data file in the same way. This will also help you remember how you made a particular sound - easy when you have just done it, not so easy a few months later on!

This may seem to be a rather complicated way of doing things - you may well ask why the program does not just generate music directly, either as sound or as MIDI. The answer in this case is simple. Depending on the settings you use, you may be generating so many individual notes so quickly that neither the computer, nor any MIDI instrument, could keep up! You may be generating hundreds of overlapping of notes a second. This echoes in a modest way the same challenge facing the scientists at CERN.

That said, the incorporation of at least some direct audio generation facilities is planned for a future version; but it remains to be seen whether it will be able to cope with all you throw at it!

Installation : OS X Leopard and Snow Leopard, Windows

Copy the *LHCSynth* folder to your machine.

Windows: copy *LHCSynth.exe* to any convenient location. We suggest you create a desktop "Shortcut" icon for it, for convenient access. Alternatively, if you prefer to work from the command line, copy it to your working directory, or to to any directory in your PATH.

OS X: *LHCSynth.app* can be put into any convenient folder. If you have administrator rights you can copy it to the */Applications* folder. You can then copy it from its new location to the Dock for convenient access from the Desktop.

Create a project folder to contain the CSD and other files written by *LHCSynth*. Copy the supplied samples folder into this new project folder. Currently the CSD files written by *LHCSynth* list require this folder structure. This is a temporary limitation of the program - in a future version all the sound will be directly synthesized using Csound, and no sample soundfiles will be needed.

Install Csound.

The full CD-based distribution package provided by LHCsound includes Csound installers for each platform. If you are downloading *LHCSynth* from the internet you will need to download and install Csound separately. You can get the latest version of Csound by accessing http://www.csounds.com and following the "download" link, which will take you to the required SourceForge page for your platform. Follow the install instructions according to your computer platform (OS X or Windows).

Csound consists of two basic packages. The most important of these is the set of dynamic libraries on which everything else depends. There is also a command line program called *csound*, which can be run directly from Terminal (OS X) or a DOS session (Windows), should you choose to work in that environment. However, you will most likely want to install one or other of the popular graphic "Front-

Ends" which control *Csound* behind the scenes. These include "WinXound" and "QuteCsound". Like *Csound* itself, these GUI tools are under continuing development by their authors, and it is always worth checking the various websites (including SourceForge) for the latest versions. For example, at the time of writing, the author of QuteCsound is promising a fully self-contained version including *Csound*, which will not require Admin permissions to install. The latest version is available here:

http://qutecsound.sourceforge.net

It is important that *Csound* is installed before any of the GUI front ends, as they generally depend on it, and look for it when they are installed.

Once a CSD file is "associated" with the program (which may be set up by the installer, or otherwise the first time the GUI program is launched), double-clicking any CSD file will automatically open it in e.g. WinXound (or other front-end of your choice), ready to "Play" or "Run" (real-time audio) or "Render" (to soundfile).

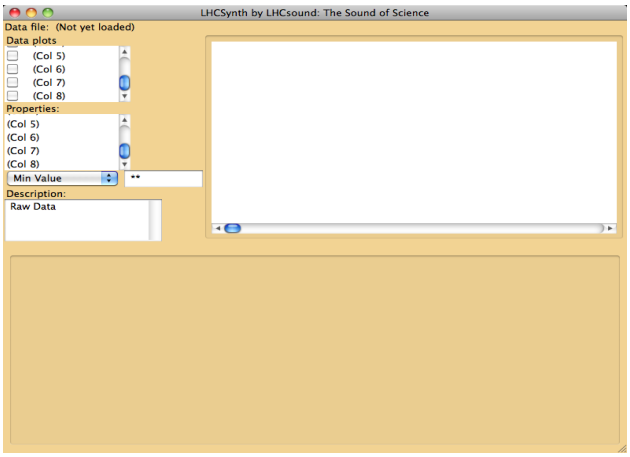
Note that we only need to use *Csound* to "perform" or compile the CSD file output by LHCSynth; any reasonably recent version will do. It is beyond the scope of this documentation to provide a tutorial for composing with *Csound*.

In this document the examples demonstrate the use of *WinXound*. This GUI has a relatively simple interface (remember that all we are interested in most of the time is rendering or playing a CSD file). *QuteCsound* offers more facilities, but has a correspondingly more complex interface. Both will display the CSD file in their text window, enabling you to see exactly how the CSD file works, and maybe even edit it a little as you learn more about *Csound*.

You will find it convenient to have two folder windows open (*Finder* on on OS X , or *Windows Explorer* on the PC). One will contain the data files to be sonified (they are text files, which can be opened and read using any standard text editor, e.g. *Notepad* on the PC, *TextEdit* on OS X). The other will contain the project folder; when a new CSD file is written, it will appear in the file list and you can simply double-click on it to launch your *Csound* GUI.

Using LHCSynth.

Launch *LHCSynth* by double-clicking on its icon. It will display its startup screen.



Available Menu commands.

File menu.

- **Open**: opens a data file with either the .txt or .lhcs file extension. On loading, controls are populated with either generic names, or names and description from the lhcs file. Upon loading, the main parameter control panel is displayed, with some simple default settings.
- **Save As lhcs File**: Save the loaded data in lhcs format. You will generally do this after editing the default column names and file description. However, if you have modified the data numerically, you may want to save it to a new lhcs file.
- **Generate ScoreFile**: having chosen the sonification parameters, this command creates the *Csound* CSD file ready to pass to *Csound*. It automatically writes an auxiliary "settings" text file (by appending ".settings.txt" to the CSD output filename), into the same directory. You can select any convenient directory using the File Selector panel, but for the file to be processed by *Csound* that directory must currently include the supplied "samples" directory of soundfiles.

  The display dynamically calculates the estimated duration of each column as determined by the number of events to be rendered, and the tempo. It is possible for each column to derive timing information from different sources, so that the duration will be different for each. However in practice for the great majority of the time a common timing source will be used (see the description of the Timing Panel, below) so that each column will report the same estimated duration.

  You can select which of the columns to sonify (render as sound). In most cases you will not directly render each column. Typically one column is used for timing, and a second column may be used to modulate the synthesis applied to a third column - which will be the one column that is then rendered. Select *only* the column(s) you intend to render directly as sound.

  The tempo control has a very wide range as it is in practice the most straightforward way to manage the pace of the output, which can vary considerably according to the nature of the numbers in the data file. It has the effect of writing a "tempo directive" into the output CSD file - by opening the file in any text editor (or using the editor in any of the *Csound* GUI tools) it is possible to edit the tempo value in the file and re-render. It will usually be a good idea to render a small number of events to begin with, and play or render the resulting CSD file, before committing to rendering the whole of a possibly large data file.

  There is an option to add a simple preset global reverb to the render - this is reflected in the generated *Csound* code. While this is primarily a musical option, it may also be seen as suggesting the relatively cavernous space within which particles are colliding.

  Note that settings are remembered between renders (for the current data file), and they are also written to the Settings file.
- **Import Settings**: load a compatible settings file as generated by **Generate ScoreFile**. Compatible means (a) the number of columns must be the same, and (b) the type of column must be the same. This concerns data that is either ascending in value (as in the case of "jet" data), or Boolean (values are only 0 or 1). Use this to render a series of data files of the same type, using identical parameter settings. You are offered the option to import the column names as well (see below). One important limitation of the settings file facility is that while all parameters are recorded in the settings file, at present it does not record any data modifications (see **Modify Data** below). The modification facility is one that is very likely to change over time, such that it would be especially premature to fix its format in the settings file. The modified data can be saved to a new **lhcs** file. An alternative approach is to save the original data in **lhcs** format and add information on modifications to the Description text (See **The Main Window** below).

For the File Open/Save functions, the program stores the directories used, so that a new session will default to those same directories.

Edit menu.

- **Modify Data**: displays a panel offering various simple ways to 'warp" the numbers in a selected column of data. We often need to do this for the simple and obvious reason that in many cases the

numbers cover such a huge range that they cannot be directly applied to any musical parameter. In particular, if we simply scale the numbers e.g by multiplication by some fraction, we will like lose all the information in the small numbers, which may be as important for the physics as the large ones. Therefore we generally need a "non-linear" formula that enlarges the small values while reducing the large ones. Each column panel in this window shows the minimum and maximum values for each warp option. Any change will be reflected in the data display once the dialog box is closed. As discussed above, note that changes made here are currently *not* written to the settings file.

- **Undo Modify**: returns to the previous column values. This is multi-level - it remembers each change in a list, so you can revert back to the original values if necessary. Note however that there is currently no "redo" option. Once a change is undone, it is lost.
- **Edit Descriptions**: A raw data file (consisting entirely of columns of numbers) provides no information either on the meaning of each column, or on the file as a whole. Graphic user interfaces tend to need names for things, so default names such as "Col 1", "Col 2" are used. This menu command opens a dialog in which you can change the name of any column to something that reflects the nature of the data, e.g. "Energy", "Angle", "Distance" and so on. The data files made available by LHCsound will all include some level of accompanying description or explanation, which can be used to decide on these names. You can also write a short free-form textual description of the data, which will be displayed in the main "Description" panel. Most importantly, if you save the data using the **Save As lhcs File** command, these names along with the description text will be saved with the file. The names are also preserved when a settings file is created – they can then optionally be imported when you select **Import Settings**.

Data file formats.

LHCSynth supports 'raw" data files (plain columns of numeric data) and a custom formatted file format using the ".lhcs" file extension. Note that this format is at an early stage of development and is likely to change, especially as new types of data are accommodated. The purpose of the **lhcs** format is to be able to populate various controls with expressive names for each data column, to include other information about the data, and generally to attach basic documentational information to the data.

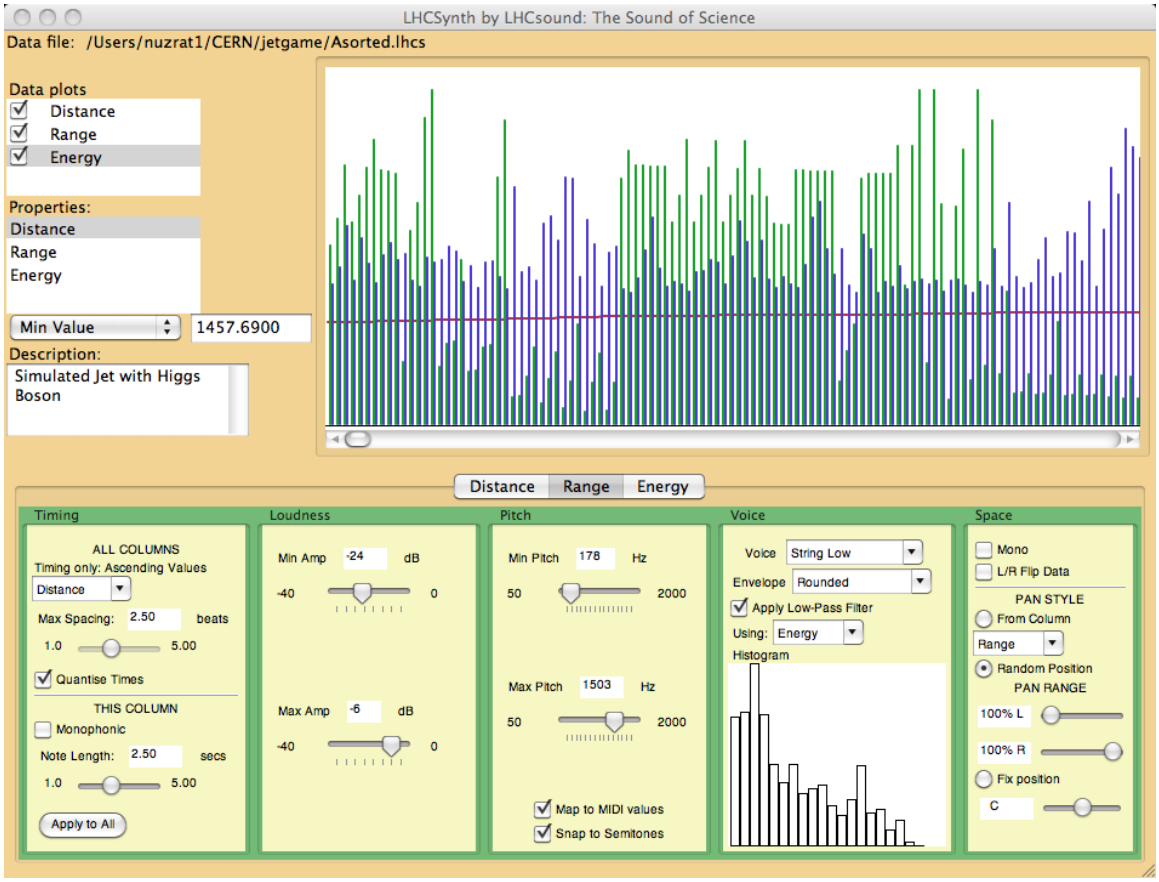
The data itself needs to be in a plain "space-separated" format. The raw data files made available on the LHCsound website are all in this form.

The Main Window.

Upon loading, the data is plotted in the large upper panel, as simple vertical lines, according to the choices made in the "Data Plots" list. Each column is plotted in a different colour. Currently the plot is very naive, and serves simply to give a general overview of the character of each data column. Each value is simply ordered from left to right, and the numeric range of each column is "normalised" to fit the entire display. It is therefore not appropriate or meaningful to compare the heights of lines from different columns (drawn in different colours). Similarly the display takes no account of any timing. However, the relative patterns formed by different columns over successive values may carry useful physics information. For example, it will give some idea of the degree of "correlation" between different columns that may be apparent (e.g. if values ascend and descend together). Where such a correlation exists, we hope that it will also be apparent in the sonification.

The lower combo box reports some basic statistics about each column, including minimum and maximum values. Currently this is for informative purposes only. It is possible that the numbers obtained may later prove useful to guide the selection of synthesis parameters.

Below this, the Description box contains either information read from a loaded **lhcs** file, or a default text indicating "raw data".



The main Voicing Panel.

This is displayed when a first data file is loaded, and remains visible thereafter, being updated as required when any further data file is loaded. It is where you decide all your sound parameters. Each column has an identical set of sub-panels for Timing, Loudness, Pitch, Voicing and Space, selected by mean of the tabs immediately above the panel. The tabs display either the default name for each column ("Col 1", "Col 2" etc) or any custom name provided either by you or loaded from an **lhcs** file. The general presumption is that each column can be rendered independently; but special consideration must inevitably be given to how events are timed. This is affected by the nature of the input data, and in particular whether data in the chosen column contains ascending values (e.g. a "jet" data file). Any such file which includes as a data column the increasing distance from the point of collision (the "interaction point"), will contain ascending values. Usually the only meaningful way to sonify such a column is to use it as a timing source, so that the file as a whole represents a "scan" outwards from the interaction point.

NB: it is assumed at present that only one such "ascending column" will be present in the data file. Results are currently undefined if this assumption is false! All current LHCsound-supplied example data files have either no ascending columns or just one such column, always the first.

The various list controls that select a column as a modulation source only show appropriate columns. For example, a column with ascending data is not regarded as suitable for a modulation source, so will be excluded from the list. Similarly, in many cases a Boolean data column is not appropriate. The term used to describe a universally applicable column is "dynamic" - i.e. it has a range of data values varying in sign and magnitude. In the somewhat challenging eight-column jet files provided by LHCsound, only three columns fit the "dynamic" category. The rest comprise the first "ascending" column and four Boolean columns.

Timing panel.

The Timing panel is divided into two sections.

The upper section contains controls applying to all data columns, regardless of which column is currently selected. A list box selects the column to use for timing. Currently, if a column has ascending values it can only be used for timing, as indicated by the text message above the selection box. Note that the program does allow you to render a timing column as sound (e.g. as a progressively ascending pitch). This may offer some interesting musical results, and in principle provides useful aural cues, but in terms of the science is in most cases not a useful thing to do, as it adds sonic complexity (albeit predictable) to the sound without imparting very much information.

Note that where a column contains ascending values, its timing combo box will *not* be modified if a different column is selected as a timing source for other columns. The recommendation whenever an ascending column is present is to use it for all timing!

Max Spacing: this control is enabled only if the data in the column does *not* contain strictly ascending values. In this case timing is derived from the difference between successive column values. The Spacing control sets the maximum time in *score beats* to which those differences are mapped. When the data is loaded the program calculates minimum and maximum values for each column (these can be inspected in the "Properties" section of the main window); the latter is in turn mapped to the maximum duration value set by the slider. Currently a small minimum duration is fixed internally (0.1 beats). This is an aspect that likely needs further refinement. The value set is in beats, rather than seconds, i.e. it is further modified by the global tempo set for score generation. Further controls may be added to vary this in a later version of the program.

Quantise Times: you can elect to quantise start times to a (currently fixed) 96 steps per beat; at slower tempi that is enough to enable the rendered sound to be combined effectively with (for example) a rhythm track.

The lower section of the Timing panel contains controls for parameters which can vary from one column to another.

Monophonic: in normal operation, a fixed note duration is employed, with the usual consequence that successive notes overlap (or may even be coincident). This can therefore be said to be a polyphonic mode of rendering. If Monophonic is checked, the duration of each note is derived directly from the time difference between successive values. An internal minimum duration ensures that no event is lost.

Note Length: this slider control sets the duration of each note (data line), in seconds, for polyphonic rendering. If Monophonic is checked, the control sets the duration of the final note. In each case the duration set is defined in seconds and is independent of any tempo setting selected in the render dialog.

Apply To All: this button copies the chosen per-column settings to the Timing panel for the other columns.

Loudness Panel.

There are only two controls here: the data in this column is automatically mapped to the selected loudness range expressed in Decibels (dB, where "0dB" signifies maximum loudness or "digital peak"). Note that each slider is constrained by the other – the "Min" and "Max" values cannot be inverted. This also enables a single "fixed" amplitude to be employed. In practice, a wide dynamic range is best avoided, as there is little advantage to generating sounds that are almost inaudible!

Where the timing column has ascending values, multiple notes may be coincident, which will raise the output level accordingly. There is no sense here of a fixed maximum polyphony ("number of voices") as used in a standard MIDI synthesiser. The density of the sound arises directly from the nature of the data, in

conjunction with the rendering settings chosen. Any notes that overlap simply add in terms of signal amplitude, and can result in digital overflows (clipping). This is clearly determined not only by the specified note duration but also be the tempo setting. A simple rule of thumb therefore is to specify a lower maximum loudness value (e.g. -12dB, which should cover most situations) for fast tempo settings and/or longer note durations. Note that *Csound* generates a comprehensive report of amplitude values for each generated note as the performance runs, and a further final report of the maximum amplitude generated, together with a count of the number of over-range values if any.

Pitch panel.

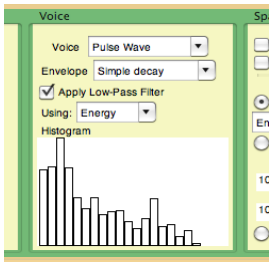
In an exactly analogous way to Loudness, this enables to user to map the data in this column to a selected frequency range. The range is currently limited to a maximum of 50Hz to 2000Hz. Where multiple columns are to be rendered together, it may often be useful to map each column to a distinct frequency range.

Map to MIDI values: when unchecked, column values are mapped directly to a raw freqneyc value within the selcted range. When checked, values are mapped quasi-logarithmically to the notes of the musical chromatic scale, represented as possibly fractional MIDI values.

Snap to Semitones: whether mapped linearly to frequency or to MIDI values, values can be "snapped" to the nearest semitone frequency value (equally-tempered semitones). This approach has some obvious limitations in terms of "preserving the physics", but does lead to results which might be regarded as more musically acceptable. It may also be well worth experimenting with "log" modifications to data to be rendered by pitch.

Voice panel.

This is in many ways the heart of the program, and at the same time the section most at the "proof of concept" stage. All the elements currently provided (voice list, envelope, filter) are preliminary facilities awaiting a much more comprehensive synthesiser-like implementation. Currently a set of fourteen "voices" is provided - ten are paired samples of acoustic instruments, each in "low" and high" versions (A220 and A440) - the latter being slightly lighter in timbre and better suited to large upwards transpositions. Additionally four standard waveforms are supported - sine, square, sawtooth and band-limited pulse. The first three are currently confined to a mere nine partials (using lookup tables), to ensure at the maximum frequency (2KHz) they will not alias. The pulse wave option uses the *Csound* opcode "buzz" for efficient generation of N harmonics. When using the pulse wave, it will usually be important to apply the filter, (see below) if the sounds are not to be tiring to listen to.



A separate set of percussive (un-pitched) samples is employed for use on Boolean columns. Samples can be selected for either or both 0 and 1 Boolean values. In most cases you will only want to sonify instances of the value 1. Future versions of the program will offer more flexible options for working with Boolean data, which must be regarded as a particularly challenging form of data to sonify. Boolean columns are at present only found in the 8-column Jet data files.

The native envelope shape of the selected sound can be modified by a choice of synthesis envelope shapes. A special feature of this list is the choice to use the displayed histogram as an envelope. The histogram (otherwise known as a "bar chart") shows the overall distribution of values in the column from low to high

values. This display may be a useful guide for deciding on an effective range for other parameters, such as pitch range and panning.

Associated with the voice selection is the option to apply a low pass filter to the data in this column, based on the data in this or another selected column. A low-pass filter progressively reduces the strength of higher frequency components (like a simple amplifier tone control), above some threshold level ("cut-off frequency"). Low data values will lead to dark or muffled tones, high values to bright tones. This enables, for example, the original (now famous/infamous) rendering of the "Higgs Jet Simple" on the LHCsound website to be replicated closely.

TIP: when using the filter in this way, it is likely that the column supplying the filter control values will not also be directly rendered - this is controlled by the final *Generate* dialog. For the 3-column Jet examples (including the Higgs Jet examples), timing for column 2 is taken from column 1 ("Distance"), and filter control from column 3 ("Energy"); the only column to be selected for rendering is column 2.

Space panel.

Whether the data as a whole in rendered in mono or stereo depends on (a) whether any column is specifically set to render in stereo, and (b) on whether reverb has been enabled in the Render dialog.

Use this panel to decide any stereo panning for the column.

Mono: check the Mono box to force the column to be rendered in mono (other controls remain as set). Where the overall CSD file is in stereo, a mono column will be presented as would be expected, equally over both channels. This check box enables you to switch between different panning options for comparison purposes, without having to change other controls.

L/R Flip Data: often the numbers in a column are very unevenly distributed (as the histogram will show very clearly) - a few negative values but many positive ones, or a few small values and many large ones. The flip option provides a simple way to bias the panning to one side or the other. Without it, large values would always end up panned heavily to the right; and in many cases you will want to ensure a clear spatial separation between columns.

Pan Style: the radio buttons select between the tree options:

- **From Column**: use this or another data column for pan control This works especially well for fairly evenly spread bipolar data (roughly equal numbers of negative and positive values). Using data purely as a control for panning is a fairly subtle technique, and in most cases you may want to *reinforce* it with other synthesis parameters such as timbre (**Voicing** panel). If the data in this column is Boolean, the sounds will flip between the limits set in the Pan Range sliders.
- **Random Position**: this simply distributes individual notes more or less evenly over the specified stereo range.
- **Fix Position**: as its name suggests, this sets a single fixed pan position.

Pan Range: for the first two options above, sets the Left/Right range over which the panning is distributed.

APPENDIX A - the files used by LHCSynth.

The nature both of the sonification task generally, and of the LHCsound tools themselves, is such that a number of text files are involved. These files support the sonification process itself by to some extent streamlining operations, but, more importantly, they also help to document your work, e.g. by storing all parameters used in a human-readable form. Note in particular that the *LHCSynth*-specific files are at an state of early development just as is the program itself. Where documenting your procedures is critical, e.g. for research or compositional reasons, use your preferred method of external record-keeping to complement

the information stored in these files.

1. The raw data file.

A plain numeric text data file (*LHCSynth* looks for the file extension **.txt**) contains plain columns of numbers, with no other information. The data in the files provided by LHCsound has been extracted from the very complex files from CERN describing a whole collision event. Data is presented in rows, each line containing one or more numbers separated by spaces, e.g.:

```
1457.69 0.61361   65.64
1457.69 0.70215  108.63
1457.69 0.88536  374.63
1459.51 0.69244   73.87
1459.51 0.78438  262.07
1459.51 0.87767  119.41
1459.51 0.97193   68.31
1462.24 0.87064  157.55
1465.89 0.86427  223.27
1470.46 0.85859  134.39
1475.94 0.21335  163.94
1475.94 0.66166   72.92
1475.94 0.75735   61.77
...
```

When such a file is loaded into LHCSynth, each column is identified in the GUI as "Col 1", "Col 2" etc.

2. The formatted LHCS file.

The LHCS file format adds a simple html-style header to the raw data. The primary purpose of this format is to associate the data itself with informative text such as names for each column of data, and a short free-form text description, e.g.:

```
<lhcsound>
<header>
<version>
1
</version>
<names>
Distance
Angle
Energy
</names>
<description>
Simulated Jet with Higgs Boson
</description>
</header>
<data>
1457.69 0.61361   65.64
1457.69 0.70215  108.63
1457.69 0.88536  374.63
1459.51 0.69244   73.87
1459.51 0.78438  262.07
...
```

This file format is at a very early stage of development; it is to be expected that it will be extended in time

to contain more "meta-information" of this kind. While it can be helpful to open the file in a text editor and read the information, it is strongly recommended you do not try to edit the file directly, unless you are confident you "know what you are doing".

This file format uses the extension **.lhcs**. It is possible to configure your operating system (OS X or Windows) to launch *LHCSynth* with a selected file when you double-click on the file name. The first time you do this, your operating system will ask you what program you want to use to open the file - just select *LHCSynth*, wherever you have put it. This creates a "File Association" for all files using that extension. The computer will remember this association until such time as you decide to change it.

3. Csound CSD file.

This is the primary output file of *LHCSynth*. *Csound* uses the "orchestra+score" model of sound synthesis. The orchestra code defines (in very precise detail) one or more instruments which generate sound. This may involve loading externally provided soundfiles to use as samples. The score contains a precisely formatted note list, which specifies the instrument to be used for a given note, its start time and duration, and a list of parameters that each instrument expects. Historically, separate files were used for the orchestra and score (*Csound* still supports this usage), but more recently a "unified" file format has been designed which merges both components into a single file, with the extension .csd. *LHCSynth* generates a CSD file with a mostly fixed (preset) orchestra – the one change that will be apparent on inspection is whether the orchestra as a whole is in mono or stereo, and whether it uses reverb. This is a simple example of a *Csound* CSD file:

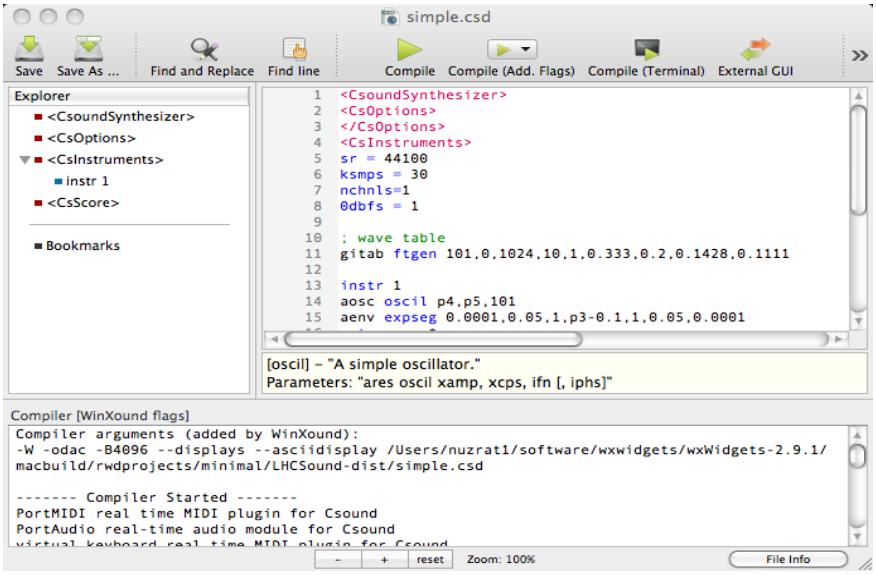
```
<CsoundSynthesizer>
<CsOptions>
</CsOptions>
<CsInstruments>
sr    = 44100
ksmps = 30
nchnls= 1
0dbfs = 1

; global wave table with 5 partials
gitab ftgen 101,0,1024,10,1,0.333,0.2,0.1428,0.1111

instr 1
aosc oscil p4,p5,101
aenv expseg 0.0001,0.05,1,p3-0.1,1,0.05,0.0001
outs aosc * aenv
endin
</CsInstruments>
<CsScore>
t0 120 ; tempo
;     t   dur  amp      freq
i1    0    6    0.2    440
i1   0.1   4    0.1    555
i1   0.3   6    0.3    333
e
</CsScore>
</CsoundSynthesizer>
```

One key facility offered by *Csound* which is relied upon in *LHCSynth* is the ability to list note events in any time order; they are sorted by *Csound* itself when the file is read. This enables *LHCSynth* to write the note list for each column separately, one after the other, safe in the knowledge that *Csound* will sort all the notes into the required time order automatically. The CSD files generated by *LHCSynth* are all ready to play - you will not need to inspect or edit the file in any way.

As the above example demonstrates, the score section supports a "tempo" instruction. This is written by *LHCSynth* according to the user's setting (see **Generate ScoreFile** above). The program applies a fixed range for tempo; this is however an easy value to modify by hand in the CSD file without having any other understanding of how the *Csound* code works.

With your chosen *Csound* front-end GUI installed, you should be able to play the CSD file by double-clicking on its name. This will open the GUI and (depending on the exact nature of the GUI) usually a single click on a button or menu item will render the file either in real time, or to a named soundfile. You will need to use the latter approach much of the time, as note densities (except perhaps for very slow tempo settings) will often be too high for the audio to be synthesised in real time without breakup.

The example below demonstrates the WinXound interface:



Note that for the CSD file to play, the directory containing the file *must* contain within it the "samples" directory containing the supplied set of specially named soundfiles. This aspect is very likely to change in future version of the program. The intention is to have all sounds directly synthesised by *Csound*, rather than rely on external soundfiles.

4. Settings File.

This fixed-form text file is used to store parameter settings (and current column names), and is written automatically each time a CSD file is generated. The large extension "-settings.txt" is appended to the csd filename. The file can be regarded as a "patch" representing a particular sonification recipe. It can be loaded into *LHCSynth* via the Menu command **Import Settings** (see above). The file is designed to be easily human-readable. Again, you should not casually edit the file in any way that changes the format. The order of each line is fixed; and each line consists of a parameter name followed by a value (which in some cases is a string enclosed in double-quotes). It includes some information not otherwise available, such as the full path to the source data file, and a timestamp recording when the file was created. As the file covers all program parameters, it will necessarily change if/when new facilities are added to *LHCSynth*. It is in this sense a temporary format while the program remains at an early stage of development.

A key use for this file is to enable multiple source data files to be sonified using the same parameter settings, so that results can be meaningfully compared. As noted above, currently it does not record any arithmetic modifications applied to the data within a session.