

## Creative Computing II Digital Music Files

8th December 2009

This lab sheet covers the creation and playback of digital music files, both in *Octave* and *Processing*. The `audio` package in *Octave* provides `wavread` and `wavwrite`; modern versions of *Processing* provide the Minim audio library.

1. This part covers the use of the `wavread` and `wavwrite` operators.
  - (a) Construct a vector representing a two-second sinusoidal signal with a frequency of 440Hz, sampled at 8kHz.  

```
signal = sin(2*pi*440*[0:1/8000:2]);
```
  - (b) The `wavwrite` operator writes vector data to an audio file in WAV format. Use `wavwrite(data, filename)` to write the vector from part 1a to a file, and play it back using the system media player. Check that it sounds the same as playing the vector data using the `Octave` `sound` command.
  - (c) The `wavwrite` operator has optional arguments for the signal sampling frequency and for the bits per sample (read ‘`help wavwrite`’ for details). Try varying these from their default values, and note any effects you hear or otherwise observe.  
*Varying the sampling frequency in the call to `wavwrite` has the effect of changing the pitch and duration of the sound in the audio file; this is because there is an absolute equivalence between the samples of, for example, a two-second sinusoid of frequency 440Hz sampled at 8kHz, and a four-second sinusoid of frequency 220Hz sampled at 4kHz. (You may want to verify this by constructing the second signal and testing it for equality against the signal constructed in part 1a above.)*
  - (d) The `wavread` operator reads signal data from an audio file in WAV format. Find a WAV file on the system you are using, use `wavread` to read in the data, and visualize the waveform.
  - (e) As well as the signal data, the `wavread` operator returns the sampling frequency and the bits per sample as secondary and tertiary return values; in general, in *Octave*, these extra return values are thrown away unless they are used. Use the syntax

```
[y, fs, bits] = wavread(filename);
```

to establish the sampling frequency and bits per sample of the WAV file you have found.

*As discussed in part 1c above, the samples themselves cannot tell the whole story: they will be the same for various different signals. The*

*sampling frequency  $fs$  is needed to relate the samples in terms of vector index to the audio signal's value at any given time.*

2. This part acts as an introduction to the Minim library that comes with Processing. The quickstart guide to Minim at <http://code.compartmental.net/tools/minim/quickstart/> provides a useful set of examples.
  - (a) First, check that you can actually use Minim code, by constructing a sketch like the 'Setup and Shutdown' example from the quickstart guide. Instead of "song.mp3" you can use a full path to the WAV file you found in the previous part. If any parts of that example cause an error, try to understand that error.
  - (b) Actually playing an audio file in Processing is marginally more complex; the `AudioPlayer` object that is constructed from loading the audio file must have its `play()` method called. Adapt your sketch so that it actually plays a sound.
  - (c) Minim also provides support for accessing the data that is currently playing; if `player` is an `AudioPlayer` object, then it has buffers of size `player.bufferSize()`; the  $i^{\text{th}}$  samples in those buffers can be acquired with `player.left.get(i)` (and `right` if the audio file is stereo). Use this to visualize the sound that is currently playing.
3. This part is about visualising audio using Sonic Visualizer, a powerful and highly-customisable visualization and annotation of digital audio.
  - The Sonic Visualiser application is installed on the Windows lab machines: a double click should start up the User Interface
  - Use the `Import Audio File...` entry in the File menu to import an audio file. Experiment with playing it back, and with the various UI controls to see what they do.
  - In order to visualize things other than the audio waveform, some plugins need to be installed; this can be done as an unprivileged user, but only by setting a Windows Environment Variable, `VAMP_PATH`, to point to a directory that you can write to, for example `G:\vamp\`. Set this environment variable, and also download the Queen Mary plugin set from the VAMP plugins home page, saving the plugins to that directory.
  - Restart Sonic Visualizer, reimport an audio file, and then experiment with the transforms in the `Transform` menu.

Other resources:

- Minim documentation: <http://code.compartmental.net/tools/minim/>

- The WAV file format. In *"Multimedia Programming Interface and Data Specifications 1.0*, Chapter 3; available at [http://www.tactilemedia.com/info/MCI\\_Control\\_Info.html](http://www.tactilemedia.com/info/MCI_Control_Info.html)
- Sonic Visualizer home page: <http://www.sonicvisualiser.org>
- VAMP plugins: <http://www.vamp-plugins.org/>