Songlib: installation of instrument sample packs

written by: Song Li Buser

Revision Date: April 17, 2015

Installing the songlib free sample repository

Songlib is a sample-based music development library. Thus the quality of the output greatly depends on the quality of the input samples. The **songlib** free sample repository (*songlib-fsr*) is provided with no restrictions. Because RRA files are rather large-ish, the samples are encoded using flac. These flac samples will need to be converted to RRA before **songlib** can use them. The basic strategy for unpacking samples is to download a sample pack to a convenient directory and then (this is for illustration only; there is a more convenient method further down):

```
cd convenientDirectory
tar xvzf instrument-basename.tgz
./install
```

Of course, *convenientDirectory* and *instrument-basename* are to be replaced with the appropriate names. Also, you will need to install *flac* utility:

```
sudo apt-get install flac
```

since the samples are stored as *flac* files (*flac* is a compressed, but lossless, audio format). The installation processes converts the *flac* files to *rra* format (hence the need for the *flac* utility). The converted rra note files are then stored in the `/usr/local/share/samples/` hierarchy.

The current set of samples in the repository can be found here.

There is a Bash shell script, named *getpack*, that automates these tasks. For example, to install the *banjo-a.tgz* sample pack, run the command:

```
getpack banjo-a
```

This will fetch, unpack, and install the instrument sample pack. Do this in a temporary directory as a bunch of files are left behind in the process.

Making your own sample packs

The converse of *getpack* is *mkpack*. To make a sample pack, you start with a set of WAV or RRA files that are properly named. Here are three ways to properly name RRA files that contain notes from octave 4, using a filename prefix of *soft_*:

note-octave	*octave-note*	*note number*
soft_c4.rra	soft_4c.rra	soft_48.rra
soft_c#4.rra	soft_4c\#.rra	soft_49.rra
soft_db4.rra	soft_4db.rra	soft_49.rra
soft_d4.rra	soft_4d.rra	soft_50.rra
soft_d#4.rra	soft_4d\#.rra	soft_51.rra
soft_eb4.rra	soft_4eb.rra	soft_51.rra
soft_e4.rra	soft_4e.rra	soft_52.rra
soft_f4.rra	soft_4f.rra	soft_53.rra
soft_f#4.rra	soft_4f\#.rra	soft_54.rra
soft_gb4.rra	soft_4gb.rra	soft_54.rra
soft_g4.rra	soft_4g.rra	soft_55.rra
soft_g#4.rra	soft_4g\#.rra	soft_56.rra
soft_ab4.rra	soft_4ab.rra	soft_56.rra
soft_a4.rra	soft_4a.rra	soft_57.rra
soft_a#4.rra	soft_4a\#.rra	soft_58.rra
soft_bb4.rra	soft_4bb.rra	soft_58.rra
soft_b4.rra	soft_4b.rra	soft_59.rra

The last column uses absolute note numbers. The lowest absolute note number is zero which corresponds to a C in octave 0. The C note in octave one has a note number of 12, since there are 12 notes in an octave and **songlib** uses zero-based counting.

Let's suppose I have a set of tinwhistle notes, stored as properly named WAV files, where the WAV files start with the prefix *soft_*. For example, one of the files might be named:

```
soft_f#5.wav
```

Suppose further, I wish to create a sample pack named *tinwhistle-soft*. I would run this command:

```
mkpack tinwhistle soft soft_*.wav
```

The first two arguments to *mkpack* will be used to construct the sample pack name.

The *mkpack* program will then perform the following steps:

1. convert the WAV files to RRA
2. fade in and fade out the RRA amplitudes so that the amplitude values both start and end with a zero
3. normalize the amplitudes so that all notes have the same volume
4. convert the sample notes to FLAC format, which is a lossless, but compressed

If RRA files are passed to *mkpack* instead of WAVs, the first step is ommitted.

Next, the FLAC files and an install script are tarred up into a tarball. Finally, the tarball is then shipped to the **songlib** server. If the notes are already in RRA format, then the command would be something like:

```
mkpack tinwhistle soft soft_*.rra
```

The fading and normalizing steps will still be performed. Note: the second argument to *mkpack* (in the example, *soft*) does not have to match the prefix of the WAV or RRA files.

For every sample pack, you should have a file named `<prefix>.README`, where `<prefix>` is the prefix of your notes (like *soft*), that gives the provenance of the notes in the sample pack. You may also have a file named `<prefix>.include` that can be used to simplify the use of your sample pack. Such an include file might load the sample pack into a predefined variable, as in:

```
instrument = readScale("/usr/local/share/samples/tinwhistle","soft_");
```

or for a set of drum kit samples:

```
setCrash(readScale("/usr/local/share/samples/beatbox/","dpe-crash_"));
setHHOpen(readScale("/usr/local/share/samples/beatbox/","dpe-hhopen_"));
setHHClosed(readScale("/usr/local/share/samples/beatbox/","dpe-hhclosed_"));
setHHPedal(readScale("/usr/local/share/samples/beatbox/","dpe-hhpedal_"));
setSnare(readScale("/usr/local/share/samples/beatbox/","dpe-snare_"));
setTomHi(readScale("/usr/local/share/samples/beatbox/","dpe-tomhi_"));
setTom(readScale("/usr/local/share/samples/beatbox/","dpe-tom_"));
setTomLo(readScale("/usr/local/share/samples/beatbox/","dpe-tomlo_"));
setKick(readScale("/usr/local/share/samples/beatbox/","dpe-kick_"));
setRim(readScale("/usr/local/share/samples/beatbox/","dpe-rim_"));
setStick(readScale("/usr/local/share/samples/beatbox/","dpe-stick_"));
```

If you don't have an account on the **songlib** server, this last step will fail. In this case, you can mail the sample pack to the email address at the bottom of this document.