Compiling SpectMorph

To compile and install SpectMorph, you need libbse >= 0.7.2 – this library is part of BEAST; however there is no beast-0.7.2 release yet, so you will have to build BEAST from git (instructions can be found on http://beast.gtk.org/download). Distribution packages should be available for most other dependancies, with the exception that boost-numeric-bindings might be missing. It can be found here: http://mathema.tician.de/software/boost-numeric-bindings

What is SpectMorph?

SpectMorph is a collection of algorithms capable of creating new sounds from existing sounds. To do this, SpectMorph allows transforming sounds into a general model. Harmonic sounds such as the sound of one note of a piano or one note of a flute are good input data for the „encoder“ which builds a parametric model of the sound.

Right now (possibly this will change in future versions), the model consists of many successive frames (also called AudioBlocks in the source code), which contain two components:

- a weighted sum of sines of different frequencies – this models the harmonic part of the sound
- an envelope describing the rest of the frame (after subtraction of the sines) as noise

Once a model has been built, algorithms can use the models as a base for creating new sounds. Right now only the encoder and player are implemented, effectively limiting SpectMorph to recreating sounds that were already provided as input; so nothing additional is gained this way. However, using smenc and smplay can help debugging the encoder/decoder, this is one of the goals of SpectMorph development right now.

Building a model with smenc

Assuming that you have a wave file of a piano called piano.wav

```
$ smenc piano.wav piano.sm
```

will build a model from the sample. Please refer to the smenc manual page for details about the options that can be used. Commonly -O *level* and -m *midi-note* can be used, for instance like this:

```
$ smenc -O1 -m 24 piano.wav piano.sm
```

Resynthesis from a model

```
$ smplay piano.sm
```

will play the model. The options are documented in the smplay manual page.

Visualizing a model

```
$ smvisualize piano.sm piano.png db
```

will produce a png in which the original fft analysis data (black/white) will be drawn, as well as the partials used by the model (red). Some more information is available in the smvisualize manual page.

Building Instruments

Often musical instruments sound very different, depending on the note being played. For instance while a piano C4 sounds similar to the C#4 note on the same piano, the C1 or C8 notes sound very different. SpectMorph instruments allow including more than one model into the instrument file, and store which midi note corresponds to which model. The first step for building an instrument is initializing a new wavset, to store the instrument models:

```
$ smwavset init piano.smset
```

Then, a number of already encoded files can be added, and the midi note to which each file corresponds can be specified. Its useful to strip the .sm-files before adding them, using smstrip (this reduces the file size a lot).

```
$ smstrip c2.sm c3.sm c4.sm
$ smwavset add piano.smset 36 c2.sm
$ smwavset add piano.smset 48 c3.sm
$ smwavset add piano.smset 60 c4.sm
```

The final step is to link the instrument – this will copy all files into the smset, so that only this file is needed to use the instrument.

```
$ smwavset link piano.smset
```

Testing instruments:

There are two ways of testing instruments: the first is the BEAST plugin. All you need to do is create a SpectMorph Oscillator, and set its filename property to the full path of the smset. You should also connect its frequency input. The plugin works the same way like the BEAST wave osc plugin, and can be used to build instruments for songs or for live usage.

The other way to test instruments is using the JACK client. It takes an smset as argument, so starting

```
$ smjack piano.smset
```

should be all that is necessary – of course you have to connect the midi input and the audio output of smjack using qjackctl (or something similar).

The live decoder is not yet optimized for performance, so only a few simultaneous notes can be played using these methods.

Comments/Questions:

stefan@space.twc.de