SpectMorph User Manual

Contents

1 Introduction

1.1 About

SpectMorph is a free software project which allows to analyze samples of musical instruments, and to combine them (morphing). It can be used to construct hybrid sounds, for instance a sound between a trumpet and a flute; or smooth transitions, for instance a sound that starts as a trumpet and then gradually changes to a flute.

In its current version, SpectMorph ships with many ready-to-use instruments which can be combined using morphing. There is also an Instrument Editor which allows you to build your own instruments and create new sounds by morphing.

SpectMorph can be used as a VST/LV2 plugin which can be loaded by many sequencers. It runs on pretty much every operating system:

- Linux (VST/LV2/JACK)
- macOS (64-bit VST)
- Windows (64-bit VST)

SpectMorph is implemented in C++ and licensed under the GNU LGPL version 3.

2 User Interface

2.1 Main Window

At the left side, there is a list of operators that generate the sound. New operators can be added using the “Add Operator” menu.

Each operator has a name, which must be unique. To change the name of an operator, double-click on the title. Operators can be folded with the triangle button at the left upper corner, to save screen space. The x at the upper right side deletes the operator.

Operators can be moved upwards and downwards by clicking on the title and dragging the operator. The color of the title indicates the status of the operator:

- green: this operator is selected as output
- white: the operator is part of the output sound
- grey: the operator is not used (not part of the output sound)

The output operator at the right side contains global settings which control how the sound output is performed. There is always one output operator.

See also: List of all supported Operators

3 Operators

3.1 Source

The **Source** operator generates the sound of one standard instrument, like *Trumpet* or *Pan Flute*. Earlier versions of SpectMorph required using source operators fairly often: to morph a *Trumpet* and a *Pan Flute*, one source was necessary for the *Trumpet*, one for the *Pan Flute* and both sources were connected to one **Linear Morph**.

However, in recent versions of SpectMorph, **Linear Morph** and **Grid Morph** can be used to morph standard instruments directly. So **Source** can be avoided in many cases.

3.2 WavSource

The **WavSource** operator generates the sound for a user defined instrument. It is called *WavSource* because users often want to morph an audio file, for instance `example.wav` with something else. In order to use samples in the **WavSource** operator, some editing is required to provide informations necessary for morphing.

User defined instruments are stored into one of 128 possible slots and can be created/edited using the Instrument Editor.

3.3 Linear Morph

The **Linear Morph** operator produces morphing between two sources.

Source A
: Set the first source used for morphing. This can be any standard instrument, or another operator. A **WavSource** can be used for morphing with a user defined instrument, or **Linear Morph** can be used for morphing with the output of another linear morph.

Source B
: Set the second source used for morphing.

Control Input
: Specify how to control the morphing, can be **Gui Slider** for doing it manually, external **Control Signal #1/#2** or another source (typically LFO).

dB Linear Morphing
: Select method of the morphing of the amplitude of the parials works. If selected, the amplitudes are converted to dB, morphed and converted back, otherwise the amplitudes are morphed with a linear method. This doesn't make much of a difference for many samples, but for some, enabling or disabling this produces a better result.

3.4 Grid Morph

The **Grid Morph** operator allows morphing instruments that are arranged on a plane. For instance a **2 x 2** grid can morph four instruments which are placed on the corners of a square.

Width
Set the width of the grid

Height
Set the height of the grid

X/Y Control
Specify how to control the morphing, can be **Gui Slider** for doing it manually, external **Control Signal #1/#2** or another source (typically LFO).

The instruments in the grid can be edited by clicking on one of the grid nodes. This will select the node, and the following options can be set for the node:

Source
The source for this grid node. It can be any standard instrument or another operator.

Volume
The relative volume of the grid node.

3.5 LFO

The **LFO** operator (low frequency oscillator) provides a control signal which slowly changes. It can for instance be used to control the morphing of other operators.

The **LFO** parameters that control the wave type, frequency and so on are

- Wave Type
- Frequency
- Depth
- Center
- Start Phase

Normally, the **LFO** operator is computed for each voice separately, which means that each voice has its own phase. If the **Sync Phase for all voices** is selected, all voices share the same phase (i.e. all voices go up and down at the same time).

3.6 Output

The **Output** operator controls how the sound generated from the sources at the left side is post processed and played. The **Source** property selects which operator is played. The user interface displays the selected operator title in green.

The **Velocity Sns** property specifies how much the note velocity of the midi input affects the volume SpectMorph uses for playback.

The output sound is composed of a *sine* and a *noise part*. **Enable Noise/Sine Synthesis** can be used to disable or enable them seperately.

3.6.1 Unison

The unison effect makes the sound more fat by adding up multiple detuned versions of the sound. The parameters **Voices/Detune** can be used to configure the effect.

3.6.2 ADSR

Normally SpectMorph tries to reproduce the volume envelope of an instrument as it was in the original sample. By using the ADSR envelope, you can change the shape of the volume envelope. The Parameters for the shape of the envelope are

- Attack
- Decay
- Sustain
- Release

Since your desired attack may be sharper than the original instrument attack, the ADSR envelope skips a part of the original instrument. This is controlled using the **Skip** parameter. So if the ADSR is enabled, we play somewhere from the middle of the input, in order to be able to produce a sharp attack.

3.6.3 Portamento

In portamento/mono mode, SpectMorph only has one single voice. Each time notes overlap, the pitch will glide from the old note to the new note. The speed can be controlled by the **Glide** parameter.

3.6.4 Vibrato

This effect adds vibrato, controlled by the **Depth** and **Frequency** parameters. If **Attack** is set, the vibrato builds up slowly during the selected time.

4 Instrument Editor

4.1 Overview

The instrument editor can be started from a WavSource operator by pressing the **Edit** button.

4.2 User Instrument Storage

In order to store user defined instruments, SpectMorph has a global list of 128 instruments. Each slot can contain one user defined instrument.

The instrument number can be set from the WavSource operator. It is important that you use an empty position for each new instrument. Suppose you assigned number 7 for an intrument based on a vocal sample. Whenever you use SpectMorph, slot number 7 will contain this instrument. As long as you are experimenting with a new instrument, you can change it in any way you like. However, as soon as you start using the instrument in a song, you should never change it again.

4.3 Samples

An instrument is created based on one or more samples. For simple use cases, one sample is enough. However, if you play a sample with much higher frequency or lower frequency than the original recording, it will no longer sound natural.

If this happens, you need to add more samples. For some instruments, two samples per octave could be enough, for others, you need 3 or 4 samples per octave to get a good quality. Many standard instruments that ship with SpectMorph even use 12 samples per octave. During playback, SpectMorph will pick the closest sample to the midi note it needs to play.

SpectMorph will analyze each sample you load, and build a model of the sound. This model is used to reproduce your sound, and to perform morphing. The model consists of the frequencies/amplitudes of the partials of the sound (modelled as sine waves) and noise for each frame. You can expect good quality from the morphing only if this model is good.

Not all samples work well as input. In all cases, samples should contain one single note. To test if a sample works well, play “Original Sample” and then “SpectMorph Instrument” in the instrument editor. If these sound approximately the same, then this sample should work. Note that you need to set the correct midi note (pitch) first, before you compare “Original Sample” and “SpectMorph Instrument”, as the quality of the model also depends on the midi note.

Looking at a spectrogram of your input sound can also be helpful. If the partial structure looks very regular, then you probably have a sample that works well with SpectMorph.

Finally, all the standard instruments obviously work well. So if your sample sounds somewhat similar to one of the existing instruments, it should work well. Acoustic musical instruments like strings, brass or human voice usually have the regular partial structure SpectMorph needs.

4.4 Loop Modes

For each sample, you need to define a loop. The loop markers are blue and have click/drag handles at the top of the window. Since your sample ends after some time, a loop is required to handle notes that are longer than this time.

4.4.1 Single Frame

A single frame loop will freeze the sound once it enters the loop. It does this by playing the same analysis frame again and again. This is good for sounds that don't change over time, like for instance organ sounds.

For sounds that change over time, like sounds with vibrato/tremolo, this loop mode is not useful.

4.4.2 Forward

A forward loop will play from left to right loop marker and then jump back to left. Since the forward loop jumps back, this will only sound good if the sample sounds very similar at the left and right loop point.

Note that to define a loop, you don‘t have to find the exactly right time points. The SpectMorph analysis/synthesis method only needs loop points that are approximately correct. This is different from samplers, that replay the sample data directly, where the loop points must be at the exactly perfect position to avoid clicks.

4.4.3 Ping Pong

A ping pong loop will play from left to right loop marker, then reverse play back to left, play to right, back to left…

The advantage of a ping pong loop is that there are no jumps, which means that no matter how the sample sounds at the left/right marker, it will always sound good.

As an example how to use ping pong loops: if you have recorded some instrument with vibrato, you select half a period of the vibrato:

- set the left loop marker the point where the vibrato is down
- set the right loop marker to the next point where the vibrato is up

4.5 Midi Note (Pitch)

For each sample, you need to assign the correct pitch. SpectMorph needs to know which pitch your sample has to transpose it correctly up and down during synthesis. If you click on “Edit” next to “Midi Note”, a dialog will show up. This dialog has been designed to find the correct midi note in case you don't know it already.

Using “Space” you can toggle playback of the selected sample. If you left click on one of the notes, you can hear the pitch of a reference instrument. By trying to click on some of the notes

while your selected sample is played you can find the correct pitch. Once you are certain what the pitch should be, double-click and close the dialog.

4.6 Volume Normalization

Usually, we want that all samples that a user instruments contains are played at the same volume. Which means that we want that all notes of a user instrument are equally loud. Volume normalization should be used to ensure this. Since we usually want to morph with existing standard instruments, the volume normalization should also make the user defined instrument as loud as the existing instruments.

4.6.1 From Loop

The typical volume normalization - which almost always works - computes the volume in the loop region, and normalizes the sounds based on this volume (we use energy to determine the volume, not peaks). So usually you check this and then the samples will be normalized properly.

4.6.2 Global

There are some rare cases where the automatic volume normalization doesn't work well. In this case, it may be desirable to keep the relative volume of the samples exactly as they were. To still be able to adjust the replay volume to the same level as the standard instruments, global normalization can adjust all samples using some dB global dB level.

This should only be used if "From Loop" did not produce a good result.

4.7 Tuning

From the midi note, SpectMorph computes the frequency a recorded note has. So if you select a midi note "69 : A3", this frequency would be 440 Hz. The auto tuning methods described here are ways to deal with samples that have a slightly different frequency than the theoretic value. As an example, if we recorded a vocal at 450 Hz, the job of auto tuning is to correct it downwards.

Note that auto tuning only works for a narrow range around the selected note, so in our example, you can't take a recording of 600 Hz and expect auto tune to do the job.

4.7.1 Partials

SpectMorph uses the first few partials (1..3) to estimate the frequency from the analysis data. So all algorithms below allow you to specify how many partials are used to compute the fundamental frequency. For many samples, this doesn't make a big difference, and 3 is a good value; however, there are a few cases where a lower value is a better choice, you can find this by trial and error.

4.7.2 Display Tuning

If you work with auto tuning, it is helpful to display the tuning over time as estimated by SpectMorph. If you enable this option, the tuning will be drawn as white line over the sample. It is possible to specify a number of cents for the tuning range. For instance the default value of 100 Cent means that if the tuning line is at the top of the sample view, the sample is 1 semitone above the midi note, in the center that it is exactly the midi note, and at the bottom it is 1 semitone below the midi note.

4.7.3 Simple

Simple auto tuning is sufficient for cases where the whole sample is too high or too low. The auto tuning algorithm will adjust the frequency so that the sample will be correct on average. This doesn't work well if some parts of the sample are too low and others are too high.

4.7.4 All Frames

This algorithm will correct each frame individually, and make the tuning effectively flat. This should work well if some parts of the sample are too low and some parts of the sample are too high. However, the approach may make the sound liveless, because it removes natural inexactness from the sample.

4.7.5 Smooth

Original developed for vocal recordings, this is somewhat similar to the all frames tuning. However, instead of forcing all frames to the exact pitch required, it tries to preserve some of the inexactness. As an example, a vibrato sound would not be completely flat, but the degree of vibrato would be reduced, and as a result, the sample would be more close to the desired midi note on average.

The "Amount" parameter specifies how much of the original tuning curve should be preserved, the "Time" parameter specifies a duration for estimating the average tuning.

4.8 Custom Analysis Parameters

It is possible to pass custom analysis parameters to the analysis process. This is an "expert only" feature, which is to say, if you don't know what this does, usually there is no problem. Analysis parameters take the

```
key=value
```

form, and although there are some possibilities here, there is only one worth mentioning.

The `min-frame-size` parameter can be used to set the shortest possible analysis frame. In general, analysis frames need to be longer if the sound has a low midi note, and shorter if the sound has a high midi note. Normally SpectMorph uses 40ms frames as shortest possible value. In some cases setting

```
min-frame-size=10
```

produces better results for higher midi notes, possibly at the expense of larger data files.