Radiant Voices Documentation

Release 0.3.0

Matthew Scott

Apr 18, 2017

Contents

CHAPTER 1

Overview of Radiant Voices

Part of the Metrasynth project.

Radiant Voices provides tools to **create, read, modify, and write SunVox files**. This includes project files ending in `.sunvox`, and module/synth files ending in `.sunsynth`.

SunVox data structures and APIs

Radiant Voices has nearly 100% coverage of all data structures used by SunVox files, exposing a "Pythonic" API for creating and manipulating those structures.

Using the API, you can do things not possible with the standard SunVox interface or the SunVox DLL, such as:

- algorithmic composition
- parametric synth/module design
- structure and complexity analysis
- automatic graph layout of modules
- and more...

Our collective imagination is the limit!

Interaction with the SunVox DLL

By combining Radiant Voices with sunvox-dll-python, one can also create alternative editing and performance tools to use alongside, or instead of, the official SunVox app.

The two packages work together to provide convenient high-level APIs for loading project and module objects directly into playback slots managed by the SunVox DLL.

Some possibilities might include:

- alternative project editors
- generative sound design using genetic algorithms
- network-enabled performance tools

What can *you* come up with?

SunVox file format documentation

Radiant Voices intends to serve as a *de facto* source of documentation about the format, as there is currently no official documentation for the SunVox file format.

The interpretation of SunVox file formats is based on a mix of "clean room" style inspection of what SunVox writes to disk when a file is edited a specific way, as well as the most recent BSD-licensed source code for the SunVox audio engine.

Requirements

- Python 3.5
- OS supported by sunvox-dll-python, if working with SunVox DLL.
- GraphViz, if you want to make use of module auto-layout features.

Quick start

The "hello world" example will construct a SunVox project in memory containing a FM module connected to the Output module. It will then load it into the SunVox DLL and send a single note-on command to the FM module:

```
$ pip install radiant-voices
$ git clone https://github.com/metrasynth/radiant-voices
$ cd radiant-voices/examples
$ python helloworld.py
```

About SunVox

From the SunVox home page:

> SunVox is a small, fast and powerful modular synthesizer with pattern-based sequencer (tracker). It is a tool for those people who like to compose music wherever they are, whenever they wish. On any device. SunVox is available for Windows, OS X, Linux, Maemo, Meego, Raspberry Pi, Windows Mobile (WindowsCE), PalmOS, iOS and Android.

CHAPTER 2

Installation

Dependencies

Required

- Python 3.5 (or greater)

Recommended

- sunvox-dll-python (for audio playback)

Optional

- GraphViz (for module auto-layout)

Installing from PyPI

Use `pip` to install the latest version published to PyPI:

```
$ pip install radiant-voices
```

Installing from GitHub

Warning: When you install this version, you may run into code that does not yet work correctly, or code whose APIs don't match what is described in the documentation.

Although the project makes an effort to ensure code in the `master` branch is kept working and consistent with documentation, this may not always be the case.

Use `pip` to install the most recent version in the `master` branch:

```
$ pip install 'https://github.com/metrasynth/radiant-voices/#egg=radiant-voices'
```

CHAPTER 3

API Reference

rv

Radiant Voices

rv.**ENCODING = 'cp1251'**

Encoding used to convert 8-bit strings to/from Unicode strings.

SunVox uses the `cp1251` encoding which supports both US ASCII and Cyrillic scripts.

See also https://en.wikipedia.org/wiki/Windows-1251.

rv.modules

Convenient access to classes that represent all SunVox module types.

Although the list below refers to the full Python module names that contain each class, you can use a shorthand notation for easier access.

For example, to refer to the "Analog Generator" SunVox module, you only need to refer to `rv.modules.AnalogGenerator` (instead of `rv.modules.analoggenerator.AnalogGenerator`).

Synths

rv.modules.analoggenerator

class `rv.modules.analoggenerator.`**AnalogGenerator**(***kwargs*)

"Analog generator" SunVox Synth Module

Behaviors:

- receives_notes
- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	volume	<Range 0..256>	80
`02` (2)	waveform	<enum 'Waveform'>	<Waveform.triangle: 0>
`03` (3)	panning	<Range -128..128>	0
`04` (4)	attack	<Range 0..256>	0
`05` (5)	release	<Range 0..256>	0
`06` (6)	sustain	<class 'bool'>	True
`07` (7)	exponential_envelope	<class 'bool'>	True
`08` (8)	duty_cycle	<Range 0..1024>	512
`09` (9)	freq2	<Range 0..2000>	1000
`0a` (10)	filter	<enum 'Filter'>	<Filter.off: 0>
`0b` (11)	f_freq_hz	<Range 0..14000>	14000
`0c` (12)	f_resonance	<Range 0..1530>	0
`0d` (13)	f_exponential_freq	<class 'bool'>	True
`0e` (14)	f_attack	<Range 0..256>	0
`0f` (15)	f_release	<Range 0..256>	0
`10` (16)	f_envelope	<enum 'FilterEnvelope'>	<FilterEnvelope.off: 0>
`11` (17)	polyphony_ch	<Range 1..32>	16
`12` (18)	mode	<enum 'Mode'>	<Mode.hq: 0>
`13` (19)	noise	<Range 0..256>	0

class `AnalogGenerator.`**`Waveform`**

An enumeration.

Name	Value
triangle	0
saw	1
square	2
noise	3
drawn	4
sin	5
hsin	6
asin	7
drawn_with_spline_interpolation	8

class `AnalogGenerator.`**`Filter`**

An enumeration.

Name	Value
off	0
lp_12db	1
hp_12db	2
bp_12db	3
br_12db	4
lp_24db	5
hp_24db	6
bp_24db	7
br_24db	8

class `AnalogGenerator.`**`FilterEnvelope`**

An enumeration.

Name	Value
off	0
sustain_off	1
sustain_on	2

class `AnalogGenerator.`**`Mode`**

An enumeration.

Name	Value
hq	0
hq_mono	1
lq	2
lq_mono	3

`rv.modules.drumsynth`

class `rv.modules.drumsynth.`**`DrumSynth`**(***kw*)

"DrumSynth" SunVox Synth Module

Behaviors:

- receives_notes
- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	volume	<Range 0..512>	256
`02` (2)	panning	<Range -128..128>	0
`03` (3)	polyphony_ch	<Range 1..8>	4
`04` (4)	bass_volume	<Range 0..512>	200
`05` (5)	bass_power	<Range 0..256>	256
`06` (6)	bass_tone	<Range 0..256>	64
`07` (7)	bass_length	<Range 0..256>	64
`08` (8)	hihat_volume	<Range 0..512>	256
`09` (9)	hihat_length	<Range 0..256>	64
`0a` (10)	snare_volume	<Range 0..512>	256
`0b` (11)	snare_tone	<Range 0..256>	128
`0c` (12)	snare_length	<Range 0..256>	64

`rv.modules.fm`

class `rv.modules.fm.`**`Fm`**(***kw*)

"FM" SunVox Synth Module

Behaviors:

- receives_notes
- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	c_volume	<Range 0..256>	128
`02` (2)	m_volume	<Range 0..256>	48
`03` (3)	panning	<Range -128..128>	0
`04` (4)	c_freq_ratio	<Range 0..16>	1
`05` (5)	m_freq_ratio	<Range 0..16>	1
`06` (6)	m_feedback	<Range 0..256>	0
`07` (7)	c_attack	<Range 0..512>	32
`08` (8)	c_decay	<Range 0..512>	32
`09` (9)	c_sustain	<Range 0..256>	128
`0a` (10)	c_release	<Range 0..512>	64
`0b` (11)	m_attack	<Range 0..512>	32
`0c` (12)	m_decay	<Range 0..512>	32
`0d` (13)	m_sustain	<Range 0..256>	128
`0e` (14)	m_release	<Range 0..512>	64
`0f` (15)	m_scaling_per_key	<Range 0..4>	0
`10` (16)	polyphony_ch	<Range 1..16>	4
`11` (17)	mode	<enum 'Mode'>	<Mode.hq: 0>

class `Fm.`**`Mode`**

An enumeration.

Name	Value
hq	0
hq_mono	1
lq	2
lq_mono	3

`rv.modules.generator`

class `rv.modules.generator.`**`Generator`**(***kwargs*)

"Generator" SunVox Synth Module

Behaviors:

- •receives_notes
- •receives_modulator
- •sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	volume	<Range 0..256>	128
`02` (2)	waveform	<enum 'Waveform'>	<Waveform.triangle: 0>
`03` (3)	panning	<Range -128..128>	0
`04` (4)	attack	<Range 0..512>	0
`05` (5)	release	<Range 0..512>	0
`06` (6)	polyphony_ch	<Range 1..16>	8
`07` (7)	mode	<enum 'Mode'>	<Mode.stereo: 0>
`08` (8)	sustain	<class 'bool'>	True
`09` (9)	freq_modulation_input	<Range 0..256>	0
`0a` (10)	duty_cycle	<Range 0..1022>	511

class `Generator.`**`Waveform`**

An enumeration.

Name	Value
triangle	0
saw	1
square	2
noise	3
dirty	4
sin	5
hsin	6
asin	7
psin	8

class `Generator.`**`Mode`**

An enumeration.

Name	Value
stereo	0
mono	1

`rv.modules.input`

class `rv.modules.input.`**`Input`**`(`***kw*`)`

"Input" SunVox Synth Module

Behaviors:

- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	volume	<Range 0..1024>	256
`02` (2)	channels	<enum 'Channels'>	<Channels.mono: 0>

class `Input.`**`Channels`**

An enumeration.

Name	Value
mono	0
stereo	1

`rv.modules.kicker`

class `rv.modules.kicker.`**`Kicker`**`(`***kw*`)`

"Kicker" SunVox Synth Module

Behaviors:

- receives_notes
- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	volume	<Range 0..256>	256
`02` (2)	waveform	<enum 'Waveform'>	<Waveform.triangle: 0>
`03` (3)	panning	<Range -128..128>	0
`04` (4)	attack	<Range 0..512>	0
`05` (5)	release	<Range 0..512>	32
`06` (6)	vol_addition	<Range 0..1024>	0
`07` (7)	env_acceleration	<Range 0..1024>	256
`08` (8)	polyphony_ch	<Range 1..4>	1
`09` (9)	anticlick	<class 'bool'>	False

class `Kicker.`**`Waveform`**

An enumeration.

Name	Value
triangle	0
square	1
sin	2

`rv.modules.sampler`

class `rv.modules.sampler.`**`Sampler`**`(`***kwargs*`)`

"Sampler" SunVox Synth Module

Note: Radiant Voices only supports sampler modules in files that were saved using newer versions of SunVox.

Files created using older versions of SunVox, such as some of the files in the `simple_examples` included with SunVox, must first be loaded into the latest version of SunVox and then saved.

Behaviors:

- receives_notes
- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	volume	<Range 0..512>	256
`02` (2)	panning	<Range -128..128>	0
`03` (3)	sample_interpolation	<enum 'SampleInterpolation'>	<SampleInterpolation.spline: 2>
`04` (4)	envelope_interpolation	<enum 'EnvelopeInterpolation'>	<EnvelopeInterpolation.linear: 1>
`05` (5)	polyphony_ch	<Range 1..32>	8
`06` (6)	rec_threshold	<Range 0..10000>	4
`07` (7)	vibrato_type	<enum 'VibratoType'>	<VibratoType.sin: 0>
`08` (8)	vibrato_attack	<Range 0..255>	0
`09` (9)	vibrato_depth	<Range 0..255>	0
`0a` (10)	vibrato_rate	<Range 0..63>	0
`0b` (11)	volume_fadeout	<Range 0..8192>	0

class `Sampler.`**`SampleInterpolation`**

An enumeration.

Name	Value
off	0
linear	1
spline	2

class `Sampler.`**`EnvelopeInterpolation`**

An enumeration.

Name	Value
off	0
linear	1

`rv.modules.spectravoice`

class `rv.modules.spectravoice.`**`SpectraVoice`**`(`***kwargs*`)`

"SpectraVoice" SunVox Synth Module

Behaviors:

- receives_notes
- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	volume	<Range 0..256>	128
`02` (2)	panning	<Range -128..128>	0
`03` (3)	attack	<Range 0..512>	10
`04` (4)	release	<Range 0..512>	512
`05` (5)	polyphony_ch	<Range 1..32>	8
`06` (6)	mode	<enum 'Mode'>	<Mode.hq_spline: 4>
`07` (7)	sustain	<class 'bool'>	True
`08` (8)	spectrum_resolution	<Range 0..5>	1
`09` (9)	harmonic	<Range 0..15>	0
`0a` (10)	h_freq_hz	<Range 0..22050>	1098
`0b` (11)	h_volume	<Range 0..255>	255
`0c` (12)	h_width	<Range 0..255>	3
`0d` (13)	h_type	<enum 'HarmonicType'>	<HarmonicType.hsin: 0>

class `SpectraVoice.`**`Mode`**

An enumeration.

Name	Value
hq	0
hq_mono	1
lq	2
lq_mono	3
hq_spline	4

class `SpectraVoice.`**`HarmonicType`**

An enumeration.

Name	Value
hsin	0
rect	1
org1	2
org2	3
org3	4
org4	5
sin	6
random	7
triangle1	8
triangle2	9
overtones1	10
overtones2	11
overtones3	12
overtones4	13

`rv.modules.vorbisplayer`

class `rv.modules.vorbisplayer.`**`VorbisPlayer`**`(`***kwargs*`)`

"Vorbis player" SunVox Synth Module

Behaviors:

- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	volume	<Range 0..512>	256
`02` (2)	original_speed	<class 'bool'>	True
`03` (3)	finetune	<Range -128..128>	0
`04` (4)	transpose	<Range -128..128>	0
`05` (5)	interpolation	<class 'bool'>	True
`06` (6)	polyphony_ch	<Range 1..4>	1
`07` (7)	repeat	<class 'bool'>	False

Effects

`rv.modules.amplifier`

class `rv.modules.amplifier.`**`Amplifier`**`(`***kw*`)`

"Amplifier" SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
01 (1)	volume	<Range 0..1024>	256
02 (2)	panning	<Range -128..128>	0
03 (3)	dc_offset	<Range -128..128>	0
04 (4)	inverse	<class 'bool'>	False
05 (5)	stereo_width	<Range 0..256>	128
06 (6)	absolute	<class 'bool'>	False
07 (7)	fine_volume	<Range 0..32768>	32768

rv.modules.compressor

class `rv.modules.compressor.`**`Compressor`**`(**kw)`

“Compressor” SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
01 (1)	volume	<Range 0..512>	256
02 (2)	threshold	<Range 0..512>	256
03 (3)	slope_pct	<Range 0..200>	100
04 (4)	attack_ms	<Range 0..500>	1
05 (5)	release_ms	<Range 1..1000>	300
06 (6)	mode	<enum 'Mode'>	<Mode.peak: 0>
07 (7)	sidechain_input	<Range 0..32>	0

class `Compressor.`**`Mode`**

An enumeration.

Name	Value
peak	0
rms	1

rv.modules.dcblocker

class `rv.modules.dcblocker.`**`DcBlocker`**`(**kw)`

“DC Blocker” SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
01 (1)	channels	<enum 'Channels'>	<Channels.stereo: 0>

class `DcBlocker.`**`Channels`**

An enumeration.

Name	Value
stereo	0
mono	1

`rv.modules.delay`

class `rv.modules.delay.`**`Delay`**`(`***kw*`)`

"Delay" SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	dry	<Range 0..512>	256
`02` (2)	wet	<Range 0..512>	256
`03` (3)	delay_l	<Range 0..256>	128
`04` (4)	delay_r	<Range 0..256>	160
`05` (5)	volume_l	<Range 0..256>	256
`06` (6)	volume_r	<Range 0..256>	256
`07` (7)	channels	<enum 'Channels'>	<Channels.stereo: 0>
`08` (8)	inverse	<class 'bool'>	False
`09` (9)	delay_units	<enum 'DelayUnits'>	<DelayUnits.sec_16384: 0>

class `Delay.`**`Channels`**

An enumeration.

Name	Value
stereo	0
mono	1

class `Delay.`**`DelayUnits`**

An enumeration.

Name	Value
sec_16384	0
ms	1
hz	2
tick	3
line	4
line_2	5
line_3	6

`rv.modules.distortion`

class `rv.modules.distortion.`**`Distortion`**`(`***kw*`)`

"Distortion" SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	volume	<Range 0..256>	128
`02` (2)	type	<enum 'Type'>	<Type.lim: 0>
`03` (3)	power	<Range 0..256>	0
`04` (4)	bit_depth	<Range 1..16>	16
`05` (5)	freq_hz	<Range 0..44100>	44100
`06` (6)	noise	<Range 0..256>	0

class `Distortion.`**`Type`**

An enumeration.

Name	Value
lim	0
sat	1

`rv.modules.echo`

class `rv.modules.echo.`**`Echo`**`(**kw)`

"Echo" SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	dry	<Range 0..256>	256
`02` (2)	wet	<Range 0..256>	128
`03` (3)	feedback	<Range 0..256>	128
`04` (4)	delay	<Range 0..256>	256
`05` (5)	channels	<enum 'Channels'>	<Channels.stereo: 1>
`06` (6)	delay_units	<enum 'DelayUnits'>	<DelayUnits.sec_256: 0>

class `Echo.`**`Channels`**

An enumeration.

Name	Value
mono	0
stereo	1

class `Echo.`**`DelayUnits`**

An enumeration.

Name	Value
sec_256	0
ms	1
hz	2
tick	3
line	4
line_2	5
line_3	6

`rv.modules.eq`

class `rv.modules.eq.`**`Eq`**`(`***kw*`)`

"EQ" SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	low	<Range 0..512>	256
`02` (2)	middle	<Range 0..512>	256
`03` (3)	high	<Range 0..512>	256
`04` (4)	channels	<enum 'Channels'>	<Channels.stereo: 0>

class `Eq.`**`Channels`**

An enumeration.

Name	Value
stereo	0
mono	1

`rv.modules.filter`

class `rv.modules.filter.`**`Filter`**`(`***kw*`)`

"Filter" SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	volume	<Range 0..256>	256
`02` (2)	freq	<Range 0..14000>	14000
`03` (3)	resonance	<Range 0..1530>	0
`04` (4)	type	<enum 'Type'>	<Type.lp: 0>
`05` (5)	response	<Range 0..256>	8
`06` (6)	mode	<enum 'Mode'>	<Mode.hq: 0>
`07` (7)	impulse	<Range 0..14000>	0
`08` (8)	mix	<Range 0..256>	256
`09` (9)	lfo_freq	<Range 0..1024>	8
`0a` (10)	lfo_amp	<Range 0..256>	0
`0b` (11)	set_lfo_phase	<Range 0..256>	0
`0c` (12)	exponential_freq	<class 'bool'>	False
`0d` (13)	roll_off	<enum 'RollOff'>	<RollOff.db_12: 0>
`0e` (14)	lfo_freq_unit	<enum 'LfoFreqUnit'>	<LfoFreqUnit.hz_0_02: 0>
`0f` (15)	lfo_waveform	<enum 'LfoWaveform'>	<LfoWaveform.sin: 0>

class `Filter.`**`Type`**

An enumeration.

Name	Value
lp	0
hp	1
bp	2
notch	3

class `Filter.`**`Mode`**
An enumeration.

Name	Value
hq	0
hq_mono	1
lq	2
lq_mono	3

class `Filter.`**`RollOff`**
An enumeration.

Name	Value
db_12	0
db_24	1
db_36	2
db_48	3

class `Filter.`**`LfoFreqUnit`**
An enumeration.

Name	Value
hz_0_02	0
ms	1
hz	2
tick	3
line	4
line_2	5
line_3	6

class `Filter.`**`LfoWaveform`**
An enumeration.

Name	Value
sin	0
saw	1
saw2	2
square	3
random	4

`rv.modules.filterpro`

class `rv.modules.filterpro.`**`FilterPro`**(***kw*)
"Filter Pro" SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	volume	<Range 0..32768>	32768
`02` (2)	type	<enum 'Type'>	<Type.lp: 0>
`03` (3)	freq_hz	<Range 0..22000>	22000
`04` (4)	freq_finetune	<Range -1000..1000>	0
`05` (5)	freq_scale	<Range 0..200>	100
`06` (6)	exponential_freq	<class 'bool'>	False
`07` (7)	q	<Range 0..32768>	16384
`08` (8)	gain	<Range -16384..16384>	0
`09` (9)	roll_off	<enum 'RollOff'>	<RollOff.db_12: 0>
`0a` (10)	response	<Range 0..1000>	250
`0b` (11)	mode	<enum 'Mode'>	<Mode.stereo: 0>
`0c` (12)	mix	<Range 0..32768>	32768
`0d` (13)	lfo_freq	<Range 0..1024>	8
`0e` (14)	lfo_amp	<Range 0..32768>	0
`0f` (15)	lfo_waveform	<enum 'LfoWaveform'>	<LfoWaveform.sin: 0>
`10` (16)	set_lfo_phase	<Range 0..256>	0
`11` (17)	lfo_freq_unit	<enum 'LfoFreqUnit'>	<LfoFreqUnit.hz_0_02: 0>

class `FilterPro.`**`Type`**

An enumeration.

Name	Value
lp	0
hp	1
bp_const_skirt_gain	2
bp_const_peak_gain	3
notch	4
all_pass	5
peaking	6
low_shelf	7
high_shelf	8

class `FilterPro.`**`RollOff`**

An enumeration.

Name	Value
db_12	0
db_24	1
db_36	2
db_48	3

class `FilterPro.`**`Mode`**

An enumeration.

Name	Value
stereo	0
mono	1

class `FilterPro.`**`LfoWaveform`**

An enumeration.

Name	Value
sin	0
saw	1
saw2	2
square	3
random	4

class `FilterPro.`**`LfoFreqUnit`**

An enumeration.

Name	Value
hz_0_02	0
ms	1
hz	2
tick	3
line	4
line_2	5
line_3	6

`rv.modules.flanger`

class `rv.modules.flanger.`**`Flanger`**`(`***kw*`)`

"Flanger" SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	dry	<Range 0..256>	256
`02` (2)	wet	<Range 0..256>	128
`03` (3)	feedback	<Range 0..256>	128
`04` (4)	delay	<Range 0..1000>	200
`05` (5)	response	<Range 0..256>	2
`06` (6)	lfo_freq	<Range 0..512>	8
`07` (7)	lfo_amp	<Range 0..256>	32
`08` (8)	lfo_waveform	<enum 'LfoWaveform'>	<LfoWaveform.hsin: 0>
`09` (9)	set_lfo_phase	<Range 0..256>	0
`0a` (10)	lfo_freq_unit	<enum 'LfoFreqUnit'>	<LfoFreqUnit.hz_0_05: 0>

class `Flanger.`**`LfoWaveform`**

An enumeration.

Name	Value
hsin	0
sin	1

class `Flanger.`**`LfoFreqUnit`**

An enumeration.

Name	Value
hz_0_05	0
ms	1
hz	2
tick	3
line	4
line_2	5
line_3	6

rv.modules.lfo

class `rv.modules.lfo.`**`Lfo`**`(**kw)`

"LFO" SunVox Effect Module

Behaviors:

- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	volume	<Range 0..512>	256
`02` (2)	type	<enum 'Type'>	<Type.amplitude: 0>
`03` (3)	amplitude	<Range 0..256>	256
`04` (4)	freq	<Range 0..2048>	256
`05` (5)	waveform	<enum 'Waveform'>	<Waveform.sin: 0>
`06` (6)	set_phase	<Range 0..256>	0
`07` (7)	channels	<enum 'Channels'>	<Channels.stereo: 0>
`08` (8)	frequency_unit	<enum 'FrequencyUnit'>	<FrequencyUnit.hz_64: 0>
`09` (9)	duty_cycle	<Range 0..256>	128

class `Lfo.`**`Type`**

An enumeration.

Name	Value
amplitude	0
panning	1

class `Lfo.`**`Waveform`**

An enumeration.

Name	Value
sin	0
square	1
sin2	2
saw	3
saw2	4
random	5

class `Lfo.`**`Channels`**

An enumeration.

Name	Value
stereo	0
mono	1

class `Lfo.`**`FrequencyUnit`**

An enumeration.

Name	Value
hz_64	0
ms	1
hz	2
tick	3
line	4
line_2	5
line_3	6

`rv.modules.loop`

class `rv.modules.loop.`**`Loop`**`(**kw)`

"Loop" SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	volume	<Range 0..256>	256
`02` (2)	delay	<Range 0..256>	256
`03` (3)	channels	<enum 'Channels'>	<Channels.stereo: 1>
`04` (4)	repeats	<Range 0..64>	0
`05` (5)	mode	<enum 'Mode'>	<Mode.normal: 0>

class `Loop.`**`Channels`**

An enumeration.

Name	Value
mono	0
stereo	1

`rv.modules.modulator`

class `rv.modules.modulator.`**`Modulator`**`(**kw)`

"Modulator" SunVox Effect Module

Behaviors:

- receives_audio
- receives_modulator
- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	volume	<Range 0..512>	256
`02` (2)	modulation_type	<enum 'ModulationType'>	<ModulationType.amplitude: 0>
`03` (3)	channels	<enum 'Channels'>	<Channels.stereo: 0>

class `Modulator.`**`ModulationType`**

An enumeration.

Name	Value
amplitude	0
phase	1
phase_abs	2

class `Modulator.`**`Channels`**

An enumeration.

Name	Value
stereo	0
mono	1

rv.modules.pitchshifter

class `rv.modules.pitchshifter.`**`PitchShifter`**(***kw*)

"Pitch shifter" SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	volume	<Range 0..512>	256
`02` (2)	pitch	<Range -600..600>	0
`03` (3)	pitch_scale	<Range 0..200>	100
`04` (4)	feedback	<Range 0..256>	0
`05` (5)	grain_size	<Range 0..256>	64
`06` (6)	mode	<enum 'Mode'>	<Mode.hq: 0>

class `PitchShifter.`**`Mode`**

An enumeration.

Name	Value
hq	0
hq_mono	1
lq	2
lq_mono	3

rv.modules.reverb

class `rv.modules.reverb.`**`Reverb`**(***kw*)

"Reverb" SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	dry	<Range 0..256>	256
`02` (2)	wet	<Range 0..256>	64
`03` (3)	feedback	<Range 0..256>	256
`04` (4)	damp	<Range 0..256>	128
`05` (5)	stereo_width	<Range 0..256>	256
`06` (6)	freeze	<class 'bool'>	False
`07` (7)	mode	<enum 'Mode'>	<Mode.hq: 0>
`08` (8)	all_pass_filter	<class 'bool'>	True
`09` (9)	room_size	<Range 0..128>	16

class `Reverb.`**`Mode`**

An enumeration.

Name	Value
hq	0
hq_mono	1
lq	2
lq_mono	3

rv.modules.vibrato

class `rv.modules.vibrato.`**`Vibrato`**`(**kw)`

“Vibrato” SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	volume	<Range 0..256>	256
`02` (2)	amplitude	<Range 0..256>	16
`03` (3)	freq	<Range 1..2048>	256
`04` (4)	channels	<enum ‘Channels’>	<Channels.stereo: 0>
`05` (5)	set_phase	<Range 0..256>	0
`06` (6)	frequency_unit	<enum ‘FrequencyUnit’>	<FrequencyUnit.hz_64: 0>
`07` (7)	exponential_amplitude	<class ‘bool’>	False

class `Vibrato.`**`Channels`**

An enumeration.

Name	Value
stereo	0
mono	1

class `Vibrato.`**`FrequencyUnit`**

An enumeration.

Name	Value
hz_64	0
ms	1
hz	2
tick	3
line	4
line_2	5
line_3	6

rv.modules.vocalfilter

class `rv.modules.vocalfilter.`**`VocalFilter`**`(**kw)`

“Vocal filter” SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	volume	<Range 0..512>	256
`02` (2)	formant_width_hz	<Range 0..256>	128
`03` (3)	intensity	<Range 0..256>	128
`04` (4)	formants	<Range 1..5>	5
`05` (5)	vowel	<Range 0..256>	0
`06` (6)	voice_type	<enum 'VoiceType'>	<VoiceType.soprano: 0>
`07` (7)	channels	<enum 'Channels'>	<Channels.stereo: 0>

class `VocalFilter.`**`VoiceType`**

An enumeration.

Name	Value
soprano	0
alto	1
tenor	2
bass	3

class `VocalFilter.`**`Channels`**

An enumeration.

Name	Value
stereo	0
mono	1

`rv.modules.waveshaper`

class `rv.modules.waveshaper.`**`WaveShaper`**`(`***kwargs*`)`

"WaveShaper" SunVox Effect Module

Behaviors:

- receives_audio
- sends_audio

Controllers:

Number	Name	Type	Default
`01` (1)	input_volume	<Range 0..512>	256
`02` (2)	mix	<Range 0..256>	256
`03` (3)	output_volume	<Range 0..512>	256
`04` (4)	symmetric	<class 'bool'>	True
`05` (5)	mode	<enum 'Mode'>	<Mode.hq: 0>
`06` (6)	dc_blocker	<class 'bool'>	True

class `WaveShaper.`**`Mode`**

An enumeration.

Name	Value
hq	0
hq_mono	1
lq	2
lq_mono	3

Misc

`rv.modules.feedback`

class `rv.modules.feedback.`**`Feedback`**`(`***kw*`)`

“Feedback” SunVox Misc Module

Behaviors:

- receives_audio
- receives_feedback
- sends_audio
- sends_feedback

Controllers:

Number	Name	Type	Default
`01` (1)	volume	<Range 0..10000>	1000
`02` (2)	channels	<enum ‘Channels’>	<Channels.stereo: 0>

class `Feedback.`**`Channels`**

An enumeration.

Name	Value
stereo	0
mono	1

`rv.modules.glide`

class `rv.modules.glide.`**`Glide`**`(`***kw*`)`

“Glide” SunVox Misc Module

Behaviors:

- receives_notes
- sends_notes

Controllers:

Number	Name	Type	Default
`01` (1)	response	<Range 0..1000>	500
`02` (2)	sample_rate_hz	<Range 1..32768>	150
`03` (3)	offset_on_1st_note	<class ‘bool’>	False
`04` (4)	polyphony	<class ‘bool’>	True
`05` (5)	pitch	<Range -600..600>	0
`06` (6)	pitch_scale	<Range 0..200>	100
`07` (7)	reset	<class ‘bool’>	False

`rv.modules.gpio`

class `rv.modules.gpio.`**`Gpio`**`(`***kw*`)`

“GPIO” SunVox Misc Module

Behaviors:

- receives_audio

- sends_audio

Controllers:

Number	Name	Type	Default
01 (1)	out	<class 'bool'>	False
02 (2)	out_pin	<Range 0..64>	0
03 (3)	out_threshold	<Range 0..100>	50
04 (4)	**in_**	<class 'bool'>	False
05 (5)	in_pin	<Range 0..64>	0
06 (6)	in_note	<Range 0..128>	0
07 (7)	in_amplitude	<Range 0..100>	100

`rv.modules.metamodule`

class `rv.modules.metamodule.`**`MetaModule`**(***kwargs*)

"MetaModule" SunVox Misc Module

In addition to standard controllers, you can assign zero or more user-defined controllers which map to module/controller pairs in the project embedded within the MetaModule.

Behaviors:

- receives_audio
- receives_notes
- sends_audio

Controllers:

Number	Name	Type	Default
01 (1)	volume	<Range 0..1024>	256
02 (2)	input_module	<Range 1..256>	1
03 (3)	play_patterns	<class 'bool'>	False
04 (4)	bpm	<Range 1..800>	125
05 (5)	tpl	<Range 1..31>	6
06 (6)	user_defined_1	<Range 0..32768>	0
07 (7)	user_defined_2	<Range 0..32768>	0
08 (8)	user_defined_3	<Range 0..32768>	0
09 (9)	user_defined_4	<Range 0..32768>	0
0a (10)	user_defined_5	<Range 0..32768>	0
0b (11)	user_defined_6	<Range 0..32768>	0
0c (12)	user_defined_7	<Range 0..32768>	0
0d (13)	user_defined_8	<Range 0..32768>	0
0e (14)	user_defined_9	<Range 0..32768>	0
0f (15)	user_defined_10	<Range 0..32768>	0
10 (16)	user_defined_11	<Range 0..32768>	0
11 (17)	user_defined_12	<Range 0..32768>	0
12 (18)	user_defined_13	<Range 0..32768>	0
13 (19)	user_defined_14	<Range 0..32768>	0
14 (20)	user_defined_15	<Range 0..32768>	0
15 (21)	user_defined_16	<Range 0..32768>	0
16 (22)	user_defined_17	<Range 0..32768>	0
17 (23)	user_defined_18	<Range 0..32768>	0

Continued on next page

Table 3.1 – continued from previous page

Number	Name	Type	Default
`18` (24)	user_defined_19	<Range 0..32768>	0
`19` (25)	user_defined_20	<Range 0..32768>	0
`1a` (26)	user_defined_21	<Range 0..32768>	0
`1b` (27)	user_defined_22	<Range 0..32768>	0
`1c` (28)	user_defined_23	<Range 0..32768>	0
`1d` (29)	user_defined_24	<Range 0..32768>	0
`1e` (30)	user_defined_25	<Range 0..32768>	0
`1f` (31)	user_defined_26	<Range 0..32768>	0
`20` (32)	user_defined_27	<Range 0..32768>	0

`rv.modules.multictl`

class `rv.modules.multictl.`**`MultiCtl`**(***kwargs*)

“MultiCtl” SunVox Misc Module

Behaviors:

- sends_controls

Controllers:

Number	Name	Type	Default
`01` (1)	value	<Range 0..32768>	0
`02` (2)	gain	<Range 0..1024>	256
`03` (3)	quantization	<Range 0..32768>	32768
`04` (4)	out_offset	<Range -16384..16384>	0

`rv.modules.multisynth`

class `rv.modules.multisynth.`**`MultiSynth`**(***kwargs*)

“MultiSynth” SunVox Misc Module

Behaviors:

- receives_notes
- sends_notes

Controllers:

Number	Name	Type	Default
`01` (1)	transpose	<CompactRange -128..128>	0
`02` (2)	random_pitch	<Range 0..4096>	0
`03` (3)	velocity	<Range 0..256>	256
`04` (4)	finetune	<Range -256..256>	0
`05` (5)	random_phase	<Range 0..32768>	0
`06` (6)	random_velocity	<Range 0..32768>	0
`07` (7)	phase	<Range 0..32768>	0

`rv.modules.output`

class `rv.modules.output.`**`Output`**(***kw*)

“Output” SunVox Output Module

This is a special module that you should never create on your own. It is automatically created as module `00` of a `Project`.

Behaviors:

- receives_audio

This module has no controllers.

rv.modules.sound2ctl

class `rv.modules.sound2ctl.`**`Sound2Ctl`**`(`***kw*`)`

"Sound2Ctl" SunVox Misc Module

Behaviors:

- receives_audio
- sends_controls

Controllers:

Number	Name	Type	Default
`01` (1)	sample_rate_hz	<Range 1..32768>	50
`02` (2)	channels	<enum 'Channels'>	<Channels.mono: 0>
`03` (3)	absolute	<class 'bool'>	True
`04` (4)	gain	<Range 0..1024>	256
`05` (5)	smooth	<Range 0..256>	128
`06` (6)	mode	<enum 'Mode'>	<Mode.hq: 1>
`07` (7)	out_min	<Range 0..32768>	0
`08` (8)	out_max	<Range 0..32768>	32768
`09` (9)	out_controller	<Range 0..32>	0

class `Sound2Ctl.`**`Channels`**

An enumeration.

Name	Value
mono	0
stereo	1

class `Sound2Ctl.`**`Mode`**

An enumeration.

Name	Value
lq	0
hq	1

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

Bug reports

When reporting a bug please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Documentation improvements

Radiant Voices could always use more documentation, whether as part of the official Radiant Voices docs, in docstrings, or even on the web in blog posts, articles, and such.

Feature requests and feedback

The best way to send feedback is to file an issue at https://github.com/metrasynth/radiant-voices/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that code contributions are welcome :)

Development

To set up *radiant-voices* for local development:

1. Fork radiant-voices (look for the "Fork" button).
2. Clone your fork locally:

```
git clone git@github.com:your_name_here/radiant-voices.git
```

3. Create a branch for local development:

```
git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

4. When you're done making changes, run all the checks, doc builder and spell checker with tox one command:

```
tox
```

5. Commit your changes and push your branch to GitHub:

```
git add .
git commit -m "Your detailed description of your changes."
git push origin name-of-your-bugfix-or-feature
```

6. Submit a pull request through the GitHub website.

Pull Request Guidelines

If you need some code review or feedback while you're developing the code just make the pull request.

For merging, you should:

1. Include passing tests (run `tox`)[1].
2. Update documentation when there's new API, functionality etc.
3. Add a note to `CHANGELOG.rst` about the changes.
4. Add yourself to `AUTHORS.rst`.

Tips

To run a subset of tests:

```
tox -e envname -- py.test -k test_myfeature
```

To run all the test environments in *parallel* (you need to `pip install detox`):

```
detox
```

[1] If you don't have all the necessary python versions available locally you can rely on Travis - it will run the tests for each change you add in the pull request.

It will be slower though ...

CHAPTER 5

Changelog

0.3.0 (2017-04-18)

Additions

- Add `propagate` argument to `MultiCtl.reflect()`. Defaults to `True` which causes the new `MultiCtl.value` to immediately propagate to all mapped controllers, including the one that was just reflected.

  Set to `False` if you only want to set `MultiCtl.value` without propagating to mapped controllers.

- Pass a value for `initial` when calling `MultiCtl.macro()` to set and propagate an initial value. Default behavior is to not set a value.

Changes

- The `repr` of a `CompactRange` instance now shows that class name, instead of `Range`.

Fixes

- Fix algorithm for propagating `MultiCtl.value` changes to mapped controllers.
- Fix algorithm for reflecting mapped controllers back to `MultiCtl.value`.

0.2.0 (2017-04-02)

Additions

- Add `Controller.pattern_value()` instance method, to map a controller's value to a pattern value in the range of 0x0000-0x8000.

- Add `ALL_NOTES` constant to see if a `NOTECMD` is a note or a command. (Example: `if some_note in ALL_NOTES: ...`)
- Add `tabular_repr()` instance methods to `Note` and `Pattern`, returning a tabular representation suitable for inclusion in text documents.
- Add `behaviors` attribute to all module classes, describing the types of information each module can send and receive.
- Add package-specific exception base classes to `rv.errors`.
- Add support for reading, writing, and modifying controller MIDI mappings.
- Add a `MultiCtl.macro()` static method, for quickly creating a `MultiCtl` that controls several similar controllers on connected modules.
- Add a `MultiCtl.reflect()` instance method, for setting a `MultiCtl`'s value based on the destination controller mapped at a given index.
- Add `# TODO: ...` notes to indicate unimplemented features.
- Allow property-style access to user-defined controllers on `MetaModule``s using a ``u_` prefix. For example, if there's a user-defined controller named "Attack", it will be accessible via the `.u_attack` property.
- Add `ArrayChunk.set_via_fn()` method, for setting various curves using the output of a function.
- Add `DRUMNOTE`, `BDNOTE`, `HHNOTE`, and `SDNOTE` enumerations to `DrumSynth` class, providing note aliases for easier programming of drum sequences.
- Add `Pattern.set_via_fn()` and `.set_via_gen()` instance methods, for altering a pattern based on the output of a function or generator.

Changes

- Rename `Output` module's module group to `"Output"`.
- When using `Project.layout()`, default to using `dot` layout engine.
- Use a direct port of SunVox's algorithm for mapping `MultiCtl` values to destination controllers.
- Use 1.9.2.0 as SunVox version number when writing projects to files.
- Allow using separate x/y offsets and factors during `Project.layout()`

Fixes

- Use same sharp note notation as used by SunVox (lowercase indicates sharp).
- Honor `prog` keyword arg when passed into `Project.layout()` method.
- Do not require pattern `x` or `y` to be divisible by 4.
- Assign correct controller number to user-defined controllers on ``MetaModule``s.
- Correct the max value allowed in a `MultiSynth` velocity/velocity curve.
- Move `pygraphviz` from `requirements/base.txt` to `.../tools.txt` to be more Windows-friendly.

0.1.1 (2016-11-09)

- Fix upload to PyPI.

0.1.0 (2016-11-09)

- Initial release.

CHAPTER 6

Authors

- Matthew Scott

CHAPTER 7

Radiant Voices license

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index