Soundfile Players

Play soundfiles from the disk.

SfMarkerXXX objects use markers features (store in the header) from an AIFF file to create more specific reading patterns.

SfPlayer

class **SfPlayer**(*path*, *speed=1*, *loop=False*, *offset=0*, *interp=2*, *mul=1*, *add=0*) [source]

Soundfile player.

Reads audio data from a file using one of several available interpolation types. User can alter its pitch with the *speed* attribute. The object takes care of sampling rate conversion to match the Server sampling rate setting.

Parent: PyoObject

Args: path: string

Full path name of the sound to read.

speed: float or PyoObject, optional

Transpose the pitch of input sound by this factor. Defaults to 1.
1 is the original pitch, lower values play sound slower, and higher values play sound faster.
Negative values results in playing sound backward.
Although the *speed* attribute accepts audio rate signal, its value is updated only once per buffer size.

loop: bool, optional

If set to True, sound will play in loop. Defaults to False.

offset: float, optional

Time in seconds of input sound to be skipped, assuming speed = 1. Defaults to 0.

interp: int, optional

Interpolation type. Defaults to 2.

1. no interpolation
2. linear
3. cosinus
4. cubic

Note: SfPlayer will send a trigger signal at the end of the playback if loop is off or any time it wraps around if loop is on. User can retrieve the trigger streams by calling obj['trig']:

```
>>> sf = SfPlayer(SNDS_PATH + "/transparent.aif").out()
>>> trig = TrigRand(sf['trig'])
```

Note that the object will send as many trigs as there is channels in the sound file. If you want to retrieve only one trig, only give the first stream to the next object:

```
>>> def printing():
...     print("one trig!")
>>> sf = SfPlayer("/stereo/sound/file.aif").out()
>>> trig = TrigFunc(sf['trig'][0], printing)
```

```
>>> s = Server().boot()
>>> s.start()
>>> snd = SNDS_PATH + "/transparent.aif"
>>> sf = SfPlayer(snd, speed=[.75,.8], loop=True, mul=.3).out()
```

setPath(*path*) [source]

Sets a new sound to read.

The number of channels of the new sound must match those of the sound loaded at initialization time.

Args: path: string

Full path of the new sound.

setSound(*path*) [source]

Sets a new sound to read.

The number of channels of the new sound must match those of the sound loaded at initialization time.

Args: path: string

Full path of the new sound.

setSpeed(*x*) [source]

Replace the *speed* attribute.

Args: x: float or PyoObject

new *speed* attribute.

setLoop(*x*) [source]

Replace the *loop* attribute.

Args: x: bool {True, False}

new *loop* attribute.

`setOffset`(*x*) [source]

Replace the *offset* attribute.

Args: x: float
new *offset* attribute.

`setInterp`(*x*) [source]

Replace the *interp* attribute.

Args: x: int {1, 2, 3, 4}
new *interp* attribute.

`path`

string. Full path of the sound.

`sound`

string. Alias to the *path* attribute.

`speed`

float or PyoObject. Transposition factor.

`loop`

bool. Looping mode.

`offset`

float. Time, in seconds, of the first sample to read.

`interp`

int {1, 2, 3, 4}. Interpolation method.

SfMarkerLooper

class `SfMarkerLooper`(*path*, *speed=1*, *mark=0*, *interp=2*, *mul=1*, *add=0*) [source]

AIFF with markers soundfile looper.

Reads audio data from a AIFF file using one of several available interpolation types. User can alter its pitch with the *speed* attribute. The object takes care of sampling rate conversion to match the Server sampling rate setting.

The reading pointer loops a specific marker (from the MARK chunk in the header of the AIFF file) until it received a new integer in the *mark* attribute.

Parent: `PyoObject`

Args: path: string

Full path name of the sound to read.

speed: float or PyoObject, optional

Transpose the pitch of input sound by this factor. Defaults to 1.
1 is the original pitch, lower values play sound slower, and higher values play sound faster.
Negative values results in playing sound backward.
Although the *speed* attribute accepts audio rate signal, its value is updated only once per buffer size.

mark: float or PyoObject, optional

Integer denoting the marker to loop, in the range 0 -> len(getMarkers()). Defaults to 0.

interp: int, optional

Choice of the interpolation method. Defaults to 2.

1. no interpolation
2. linear
3. cosinus
4. cubic

```
>>> s = Server().boot()
>>> s.start()
>>> a = SfMarkerLooper(SNDS_PATH + '/transparent.aif', speed=[.999,1], mul=.3).out()
>>> rnd = RandInt(len(a.getMarkers()), 2)
>>> a.mark = rnd
```

`setSpeed`(*x*) [source]

Replace the *speed* attribute.

Args: x: float or PyoObject

new *speed* attribute.

`setMark`(*x*) [source]

Replace the *mark* attribute.

Args: x: float or PyoObject

new *mark* attribute.

`setInterp`(*x*) [source]

Replace the *interp* attribute.

Args: x: int {1, 2, 3, 4}

new *interp* attribute.

getMarkers() [source]

Returns a list of marker time values in samples.

speed

float or PyoObject. Transposition factor.

mark

float or PyoObject. Marker to loop.

interp

int {1, 2, 3, 4}. Interpolation method.

SfMarkerShuffler

class **SfMarkerShuffler**(*path*, *speed=1*, *interp=2*, *mul=1*, *add=0*) [source]

AIFF with markers soundfile shuffler.

Reads audio data from a AIFF file using one of several available interpolation types. User can alter its pitch with the *speed* attribute. The object takes care of sampling rate conversion to match the Server sampling rate setting.

The reading pointer randomly choose a marker (from the MARK chunk in the header of the AIFF file) as its starting point and reads the samples until it reaches the following marker. Then, it choose another marker and reads from the new position and so on...

Parent: PyoObject

Args: path: string

Full path name of the sound to read. Can't e changed after initialization.

speed: float or PyoObject, optional

Transpose the pitch of input sound by this factor. Defaults to 1.
1 is the original pitch, lower values play sound slower, and higher values play sound faster.
Negative values results in playing sound backward.
Although the *speed* attribute accepts audio rate signal, its value is updated only once per buffer size.

interp: int, optional

Choice of the interpolation method. Defaults to 2.

1. no interpolation
2. linear
3. cosinus
4. cubic

```
>>> s = Server().boot()
>>> s.start()
```

```
>>> sound = SNDS_PATH + "/transparent.aif"
>>> sf = SfMarkerShuffler(sound, speed=[1,1], mul=.3).out()
>>> sf.setRandomType("expon_min", 0.6)
```

setSpeed(*x*) [source]

Replace the *speed* attribute.

Args: x: float or PyoObject

new *speed* attribute.

setInterp(*x*) [source]

Replace the *interp* attribute.

Args: x: int {1, 2, 3, 4}

new *interp* attribute.

setRandomType(*dist=0*, *x=0.5*) [source]

Set the random distribution type used to choose the markers.

Args: dist: int or string

The distribution type. Available distributions are:

0. uniform (default)
1. linear minimum
2. linear maximum
3. triangular
4. exponential minimum
5. exponential maximum
6. double (bi)exponential
7. cauchy
8. weibull
9. gaussian

x: float

Distribution specific parameter, if applicable, as a float between 0 and 1. Defaults to 0.5.

Note: Depending on the distribution type, *x* parameter is applied as follow (names as string, or associated number can be used as *dist* parameter):

0. **uniform**
   - x: not used
1. **linear_min**
   - x: not used
2. **linear_max**

- x: not used

3. **triangle**
    - x: not used
4. **expon_min**
    - x: slope {0 = no slope -> 1 = sharp slope}
5. **expon_max**
    - x: slope {0 = no slope -> 1 = sharp slope}
6. **biexpon**
    - x: bandwidth {0 = huge bandwidth -> 1 = narrow bandwidth}
7. **cauchy**
    - x: bandwidth {0 = huge bandwidth -> 1 = narroe bandwidth}
8. **weibull**
    - x: shape {0 = expon min => linear min => 1 = gaussian}
9. **gaussian**
    - x: bandwidth {0 = huge bandwidth -> 1 = narrow bandwidth}

`getMarkers`() [source]

Returns a list of marker time values in samples.

`speed`

float or PyoObject. Transposition factor.

`interp`

int {1, 2, 3, 4}. Interpolation method.