Routing

Set of objects to manage audio voice routing and spread of a sound signal into a stereo or multi-channel sound field.

Mixer

class **Mixer**(*outs=2*, *chnls=1*, *time=0.025*, *mul=1*, *add=0*) [source]

Audio mixer.

Mixer mixes multiple inputs to an arbitrary number of outputs with independant amplitude values per mixing channel and a user defined portamento applied on amplitude changes.

Parent: PyoObject

Args: outs: int, optional

Number of outputs of the mixer. Available at initialization time only. Defaults to 2.

chnls: int, optional

Number of channels per output. Available at initialization time only. Defaults to 1.

time: float, optional

Duration, in seconds, of a portamento applied on a new amplitude value for a mixing channel. Defaults to 0.025.

Note: User can retrieve each of the output channels by calling the Mixer object with the desired channel between square brackets (see example).

```
>>> s = Server().boot()
>>> s.start()
>>> a = SfPlayer(SNDS_PATH+"/transparent.aif", loop=True, mul=.2)
>>> b = FM(carrier=200, ratio=[.5013,.4998], index=6, mul=.2)
>>> mm = Mixer(outs=3, chnls=2, time=.025)
>>> fx1 = Disto(mm[0], drive=.9, slope=.9, mul=.1).out()
>>> fx2 = Freeverb(mm[1], size=.8, damp=.8, mul=.5).out()
>>> fx3 = Harmonizer(mm[2], transpo=1, feedback=.75, mul=.5).out()
>>> mm.addInput(0, a)
>>> mm.addInput(1, b)
>>> mm.setAmp(0,0,.5)
>>> mm.setAmp(0,1,.5)
>>> mm.setAmp(1,2,.5)
>>> mm.setAmp(1,1,.5)
```

addInput(*voice*, *input*) [source]

Adds an audio object in the mixer's inputs.

This method returns the key (voice argument or generated key if voice=None).

Args: voice: int or string

Key in the mixer dictionary for this input. If None, a unique key between 0 and 32767 will be automatically generated.

input: PyoObject

Audio object to add to the mixer.

`delInput`(*voice*) [source]

Removes an audio object from the mixer's inputs.

Args: voice: int or string

Key in the mixer dictionary assigned to the input to remove.

`getChannels`() [source]

Returns the Mixer's channels dictionary.

`getKeys`() [source]

Returns the list of current keys in the Mixer's channels dictionary.

`setAmp`(*vin*, *vout*, *amp*) [source]

Sets the amplitude of a mixing channel.

Args: vin: int or string

Key in the mixer dictionary of the desired input.

vout: int

Ouput channel where to send the signal.

amp: float

Amplitude value for this mixing channel.

`setTime`(*x*) [source]

Sets the portamento duration in seconds.

Args: x: float

New portamento duration.

`time`

float. Portamento.

Pan

class `Pan`(*input*, *outs=2*, *pan=0.5*, *spread=0.5*, *mul=1*, *add=0*) [source]

Cosinus panner with control on the spread factor.

Parent: `PyoObject`

Args: input: PyoObject

Input signal to process.

outs: int, optional

Number of channels on the panning circle. Available at initialization time only. Defaults to 2.

pan: float or PyoObject

Position of the sound on the panning circle, between 0 and 1. Defaults to 0.5.

spread: float or PyoObject

Amount of sound leaking to the surrounding channels, between 0 and 1. Defaults to 0.5.

```
>>> s = Server(nchnls=2).boot()
>>> s.start()
>>> a = Noise(mul=.2)
>>> lfo = Sine(freq=1, mul=.5, add=.5)
>>> p = Pan(a, outs=2, pan=lfo).out()
```

input

PyoObject. Input signal to process.

pan

float or PyoObject. Position of the sound on the panning circle.

setInput(*x*, *fadetime=0.05*) [source]

Replace the *input* attribute.

Args: x: PyoObject

New signal to process.

fadetime: float, optional

Crossfade time between old and new input. Default to 0.05.

setPan(*x*) [source]

Replace the *pan* attribute.

Args: x: float or PyoObject

new *pan* attribute.

setSpread(*x*) [source]

Replace the *spread* attribute.

Args: x: float or PyoObject

new *spread* attribute.

spread

float or PyoObject. Amount of sound leaking to the surrounding channels.

SPan

class **SPan**(*input*, *outs=2*, *pan=0.5*, *mul=1*, *add=0*) [source]

Simple equal power panner.

Parent: PyoObject

Args: input: PyoObject

Input signal to process.

outs: int, optional

Number of channels on the panning circle. Available at initialization time only. Defaults to 2.

pan: float or PyoObject

Position of the sound on the panning circle, between 0 and 1. Defaults to 0.5.

```
>>> s = Server(nchnls=2).boot()
>>> s.start()
>>> a = Noise(mul=.2)
>>> lfo = Sine(freq=1, mul=.5, add=.5)
>>> p = SPan(a, outs=2, pan=lfo).out()
```

input

PyoObject. Input signal to process.

pan

float or PyoObject. Position of the sound on the panning circle.

setInput(*x*, *fadetime=0.05*) [source]

Replace the *input* attribute.

Args: x: PyoObject

New signal to process.

fadetime: float, optional

Crossfade time between old and new input. Default to 0.05.

setPan(*x*) [source]

Replace the *pan* attribute.

Args: x: float or PyoObject

new *pan* attribute.

Selector

class **Selector**(*inputs*, *voice=0.0*, *mul=1*, *add=0*) [source]

Audio selector.

Selector takes multiple PyoObjects in input and interpolates between them to generate a single output.

Parent: PyoObject

Args: inputs: list of PyoObject

Audio objects to interpolate from.

voice: float or PyoObject, optional

Voice position pointer, between 0 and len(inputs)-1. Defaults to 0.

```
>>> s = Server().boot()
>>> s.start()
>>> a = SfPlayer(SNDS_PATH + "/transparent.aif", speed=[.999,1], loop=True, mul=.3)
>>> b = Noise(mul=.1)
>>> c = SfPlayer(SNDS_PATH + "/accord.aif", speed=[.999,1], loop=True, mul=.5)
>>> lf = Sine(freq=.1, add=1)
>>> d = Selector(inputs=[a,b,c], voice=lf).out()
```

inputs

List of PyoObjects. Audio objects to interpolate from.

setInputs(*x*) [source]

Replace the *inputs* attribute.

Args: x: list of PyoObject

new *inputs* attribute.

setMode(*x*) [source]

Change the algorithm used to interpolate between inputs.

if inputs are phase correlated you should use a linear fade.

Args: x: int {0, 1}

If 0 (the default) the equal power law is used to interpolate bewtween sources. If 1, linear fade is used instead.

setVoice(*x*) [source]

Replace the *voice* attribute.

Args: x: float or PyoObject
new *voice* attribute.

voice

float or PyoObject. Voice position pointer.

Switch

class **Switch**(*input*, *outs=2*, *voice=0.0*, *mul=1*, *add=0*) [source]

Audio switcher.

Switch takes an audio input and interpolates between multiple outputs.

User can retrieve the different streams by calling the output number between brackets. obj[0] retrieve the first stream, obj[outs-1] the last one.

Parent: PyoObject

Args: input: PyoObject
Input signal to process.

outs: int, optional
Number of outputs. Available at initialization time only. Defaults to 2.

voice: float or PyoObject
Voice position pointer, between 0 and (outs-1) / len(input). Defaults to 0.

```
>>> s = Server(nchnls=2).boot()
>>> s.start()
>>> a = SfPlayer(SNDS_PATH + "/transparent.aif", speed=[.999,1], loop=True, mul=.3)
>>> lf = Sine(freq=.25, mul=1, add=1)
>>> b = Switch(a, outs=6, voice=lf)
>>> c = WGVerb(b[0:2], feedback=.8).out()
>>> d = Disto(b[2:4], drive=.9, mul=.1).out()
>>> e = Delay(b[4:6], delay=.2, feedback=.6).out()
```

input

PyoObject. Input signal to process.

setInput(*x*, *fadetime=0.05*) [source]

Replace the *input* attribute.

Args: x: PyoObject
New signal to process.

fadetime: float, optional

Crossfade time between old and new input. Default to 0.05.

setVoice(*x*) [source]

Replace the *voice* attribute.

Args: x: float or PyoObject
new *voice* attribute.

voice

float or PyoObject. Voice position pointer.

VoiceManager

class **VoiceManager**(*input*, *triggers=None*, *mul=1*, *add=0*) [source]

Polyphony voice manager.

A trigger in input ask the object for a voice number and the object returns the first free one. The voice number is then disable until a trig comes at the same position in the list of triggers given at the argument *triggers*.

Usually, the trigger enabling the voice number will come from the process started with the object output. So, it's common to leave the *triggers* argument to None and set the list of triggers afterward with the *setTriggers* method. The maximum number of voices generated by the object is the length of the trigger list.

If there is no free voice, the object outputs -1.0 continuously.

Parent: PyoObject

Args: input: PyoObject
Trigger stream asking for new voice numbers.

triggers: PyoObject or list of PyoObject, optional
List of mono PyoObject sending triggers. Can be a multi-streams PyoObject but not a mix of both.
Ordering in the list corresponds to voice numbers. Defaults to None.

```
>>> s = Server().boot()
>>> s.start()
>>> env = CosTable([(0,0),(100,1),(500,.5),(8192,0)])
>>> delta = RandDur(min=.05, max=.1)
>>> vm = VoiceManager(Change(delta))
>>> sel = Select(vm, value=[0,1,2,3])
>>> pit = TrigChoice(sel, choice=[midiToHz(x) for x in [60,63,67,70,72]])
>>> amp = TrigEnv(sel, table=env, dur=.5, mul=.25)
>>> synth1 = SineLoop(pit, feedback=.07, mul=amp).out()
>>> vm.setTriggers(amp["trig"])
```

input

PyoObject. Trigger stream asking for new voice numbers.

setInput(*x*, *fadetime=0.05*) [source]

Replace the *input* attribute.

Args: x: PyoObject

New signal to process.

fadetime: float, optional

Crossfade time between old and new input. Defaults to 0.05.

setTriggers(*x*) [source]

Replace the *triggers* attribute.

Args: x: PyoObject or list of PyoObject

New *triggers* attribute.

triggers

list of PyoObject. Trigger streams enabling voices.