Dynamic management

Objects to modify the dynamic range and sample quality of audio signals.

Clip

class **`Clip`**(*input*, *min=-1.0*, *max=1.0*, *mul=1*, *add=0*) [source]

Clips a signal to a predefined limit.

Parent: `PyoObject`

Args: input: PyoObject

Input signal to process.

min: float or PyoObject, optional

Minimum possible value. Defaults to -1.

max: float or PyoObject, optional

Maximum possible value. Defaults to 1.

```
>>> s = Server().boot()
>>> s.start()
>>> a = SfPlayer(SNDS_PATH + "/transparent.aif", loop=True)
>>> lfoup = Sine(freq=.25, mul=.48, add=.5)
>>> lfodown = 0 - lfoup
>>> c = Clip(a, min=lfodown, max=lfoup, mul=.4).mix(2).out()
```

`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.

`setMin`(*x*) [source]

Replace the *min* attribute.

Args: x: float or PyoObject

New *min* attribute.

`setMax`(*x*) [source]

Replace the *max* attribute.

Args: x: float or PyoObject

New *max* attribute.

`input`

PyoObject. Input signal to process.

`min`

float or PyoObject. Minimum possible value.

`max`

float or PyoObject. Maximum possible value.

Degrade

class **`Degrade`**(*input*, *bitdepth=16*, *srscale=1.0*, *mul=1*, *add=0*) [source]

Signal quality reducer.

Degrade takes an audio signal and reduces the sampling rate and/or bit-depth as specified.

Parent: `PyoObject`

Args: input: PyoObject

Input signal to process.

bitdepth: float or PyoObject, optional

Signal quantization in bits. Must be in range 1 -> 32. Defaults to 16.

srscale: float or PyoObject, optional

Sampling rate multiplier. Must be in range 0.0009765625 -> 1. Defaults to 1.

```
>>> s = Server().boot()
>>> s.start()
>>> t = SquareTable()
>>> a = Osc(table=t, freq=[100,101], mul=.5)
>>> lfo = Sine(freq=.2, mul=6, add=8)
>>> lfo2 = Sine(freq=.25, mul=.45, add=.55)
>>> b = Degrade(a, bitdepth=lfo, srscale=lfo2, mul=.3).out()
```

`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.

`setBitdepth`(*x*) [source]

Replace the *bitdepth* attribute.

Args: x: float or PyoObject

New *bitdepth* attribute.

`setSrscale`(*x*) [source]

Replace the *srscale* attribute.

Args: x: float or PyoObject

New *srscale* attribute.

`input`

PyoObject. Input signal to process.

`bitdepth`

float or PyoObject. Signal quantization in bits.

`srscale`

float or PyoObject. Sampling rate multiplier.

Mirror

class `Mirror`(*input*, *min=0.0*, *max=1.0*, *mul=1*, *add=0*) [source]

Reflects the signal that exceeds the *min* and *max* thresholds.

This object is useful for table indexing or for clipping and modeling an audio signal.

Parent: `PyoObject`

Args: input: PyoObject

Input signal to process.

min: float or PyoObject, optional

Minimum possible value. Defaults to 0.

max: float or PyoObject, optional

Maximum possible value. Defaults to 1.

Note: If *min* is higher than *max*, then the output will be the average of the two.

```
>>> s = Server().boot()
>>> s.start()
>>> a = Sine(freq=[300,301])
>>> lfmin = Sine(freq=1.5, mul=.25, add=-0.75)
>>> lfmax = Sine(freq=2, mul=.25, add=0.75)
>>> b = Mirror(a, min=lfmin, max=lfmax)
>>> c = Tone(b, freq=2500, mul=.15).out()
```

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.

setMin(*x*) [source]

Replace the *min* attribute.

Args: x: float or PyoObject

New *min* attribute.

setMax(*x*) [source]

Replace the *max* attribute.

Args: x: float or PyoObject

New *max* attribute.

input

PyoObject. Input signal to process.

min

float or PyoObject. Minimum possible value.

max

float or PyoObject. Maximum possible value.

Compress

class **Compress**(*input*, *thresh=-20*, *ratio=2*, *risetime=0.01*, *falltime=0.1*, *lookahead=5.0*, *knee=0*, *outputAmp=False*, *mul=1*, *add=0*) [source]

Reduces the dynamic range of an audio signal.

Compress reduces the volume of loud sounds or amplifies quiet sounds by narrowing or compressing an audio signal's dynamic range.

Parent: `PyoObject`

Args: input: PyoObject

Input signal to process.

thresh: float or PyoObject, optional

Level, expressed in dB, above which the signal is reduced. Reference level is 0dB. Defaults to -20.

ratio: float or PyoObject, optional

Determines the input/output ratio for signals above the threshold. Defaults to 2.

risetime: float or PyoObject, optional

Used in amplitude follower, time to reach upward value in seconds. Defaults to 0.01.

falltime: float or PyoObject, optional

Used in amplitude follower, time to reach downward value in seconds. Defaults to 0.1.

lookahead: float, optional

Delay length, in ms, for the “look-ahead” buffer. Range is 0 -> 25 ms. Defaults to 5.0.

knee: float optional

Shape of the transfert function around the threshold, specified in the range 0 -> 1.
A value of 0 means a hard knee and a value of 1.0 means a softer knee. Defaults to 0.

outputAmp: boolean, optional

If True, the object’s output signal will be the compression level alone, not the compressed signal.
It can be useful if 2 or more channels need to linked on the same compression slope.
Defaults to False.
Available at initialization only.

```
>>> s = Server().boot()
>>> s.start()
>>> a = SfPlayer(SNDS_PATH + '/transparent.aif', loop=True)
>>> b = Compress(a, thresh=-24, ratio=6, risetime=.01, falltime=.2, knee=0.5).mix(2).out()
```

`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.

`setThresh`(*x*) [source]

Replace the *thresh* attribute.

Args: x: float or PyoObject

New *thresh* attribute.

`setRatio`(*x*) [source]

Replace the *ratio* attribute.

Args: x: float or PyoObject
New *ratio* attribute.

`setRiseTime`(*x*) [source]

Replace the *risetime* attribute.

Args: x: float or PyoObject
New *risetime* attribute.

`setFallTime`(*x*) [source]

Replace the *falltime* attribute.

Args: x: float or PyoObject
New *falltime* attribute.

`setLookAhead`(*x*) [source]

Replace the *lookahead* attribute.

Args: x: float
New *lookahead* attribute.

`setKnee`(*x*) [source]

Replace the *knee* attribute.

Args: x: float
New *knee* attribute.

`input`

PyoObject. Input signal to process.

`thresh`

float or PyoObject. Level above which the signal is reduced.

`ratio`

float or PyoObject. in/out ratio for signals above the threshold.

`risetime`

float or PyoObject. Time to reach upward value in seconds.

`falltime`

float or PyoObject. Time to reach downward value in seconds.

`lookahead`

float. Delay length, in ms, of the “look-ahead” buffer.

`knee`

float. Shape of the transfert function around the threshold.

Gate

class **`Gate`**(*input*, *thresh=-70*, *risetime=0.01*, *falltime=0.05*, *lookahead=5.0*, *outputAmp=False*, *mul=1*, *add=0*) [source]

Allows a signal to pass only when its amplitude is above a set threshold.

A noise gate is used when the level of the signal is below the level of the noise floor. The threshold is set above the level of the noise and so when there is no signal the gate is closed. A noise gate does not remove noise from the signal. When the gate is open both the signal and the noise will pass through.

Parent: `PyoObject`

Args: input: PyoObject

Input signal to process.

thresh: float or PyoObject, optional

Level, expressed in dB, below which the gate is closed. Reference level is 0dB. Defaults to -70.

risetime: float or PyoObject, optional

Time to open the gate in seconds. Defaults to 0.01.

falltime: float or PyoObject, optional

Time to close the gate in seconds. Defaults to 0.05.

lookahead: float, optional

Delay length, in ms, for the “look-ahead” buffer. Range is 0 -> 25 ms. Defaults to 5.0.

outputAmp: boolean, optional

If True, the object’s output signal will be the gating level alone, not the gated signal.
It can be useful if 2 or more channels need to linked on the same gating slope.
Defaults to False.
Available at initialization only.

```
>>> s = Server().boot()
>>> s.start()
>>> sf = SfPlayer(SNDS_PATH + '/transparent.aif', speed=[1,.5], loop=True)
>>> gt = Gate(sf, thresh=-24, risetime=0.005, falltime=0.01, lookahead=5, mul=.4).out()
```

`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.

`setThresh`(*x*) [source]

Replace the *thresh* attribute.

Args: x: float or PyoObject

New *thresh* attribute.

`setRiseTime`(*x*) [source]

Replace the *risetime* attribute.

Args: x: float or PyoObject

New *risetime* attribute.

`setFallTime`(*x*) [source]

Replace the *falltime* attribute.

Args: x: float or PyoObject

New *falltime* attribute.

`setLookAhead`(*x*) [source]

Replace the *lookahead* attribute.

Args: x: float

New *lookahead* attribute.

`input`

PyoObject. Input signal to process.

`thresh`

float or PyoObject. Level below which the gate is closed.

`risetime`

float or PyoObject. Time to open the gate in seconds.

`falltime`

float or PyoObject. Time to close the gate in seconds.

lookahead

float. Delay length, in ms, of the “look-ahead” buffer.

Balance

class **Balance**(*input*, *input2*, *freq=10*, *mul=1*, *add=0*) [source]

Adjust rms power of an audio signal according to the rms power of another.

The rms power of a signal is adjusted to match that of a comparator signal.

Parent: PyoObject

Args: input: PyoObject

Input signal to process.

input2: PyoObject

Comparator signal.

freq: float or PyoObject, optional

Cutoff frequency of the lowpass filter in hertz. Default to 10.

```
>>> s = Server().boot()
>>> s.start()
>>> sf = SfPlayer(SNDS_PATH + '/accord.aif', speed=[.99,1], loop=True, mul=.3)
>>> comp = SfPlayer(SNDS_PATH + '/transparent.aif', speed=[.99,1], loop=True, mul=.3)
>>> out = Balance(sf, comp, freq=10).out()
```

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

Replace the *input* attribute.

Input signal to process.

Args: x: PyoObject

New signal to process.

fadetime: float, optional

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

setInput2(*x*, *fadetime=0.05*) [source]

Replace the *input2* attribute.

Comparator signal.

Args: x: PyoObject

New signal to process.

fadetime: float, optional

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

`setFreq`(*x*) [source]

Replace the *freq* attribute.

Cutoff frequency of the lowpass filter, in Hertz.

Args: x: float or PyoObject

New *freq* attribute.

`input`

PyoObject. Input signal to process.

`input2`

PyoObject. Comparator signal.

`freq`

float or PyoObject. Cutoff frequency of the lowpass filter.

Min

class `Min`(*input*, *comp=0.5*, *mul=1*, *add=0*) [source]

Outputs the minimum of two values.

Parent: `PyoObject`

Args: input: PyoObject

Input signal to process.

comp: float or PyoObject, optional

Comparison value. If *input* is lower than this value, *input* is send to the output, otherwise, *comp* is outputted.

```
>>> s = Server().boot()
>>> s.start()
>>> # Triangle wave
>>> a = Phasor([249,250])
>>> b = Min(a, comp=a*-1+1, mul=4, add=-1)
>>> c = Tone(b, freq=1500, mul=.5).out()
```

`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.

`setComp`(*x*) [source]

Replace the *comp* attribute.

Args: x: float or PyoObject

New *comp* attribute.

`input`

PyoObject. Input signal to process.

`comp`

float or PyoObject. Comparison value.

Max

class `Max`(*input*, *comp=0.5*, *mul=1*, *add=0*) [source]

Outputs the maximum of two values.

Parent: `PyoObject`

Args: input: PyoObject

Input signal to process.

comp: float or PyoObject, optional

Comparison value. If *input* is higher than this value, *input* is send to the output, otherwise, *comp* is outputted.

```
>>> s = Server().boot()
>>> s.start()
>>> # Assimetrical clipping
>>> a = Phasor(500, mul=2, add=-1)
>>> b = Max(a, comp=-0.3)
>>> c = Tone(b, freq=1500, mul=.5).out()
```

`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.

`setComp`(*x*) [source]

Replace the *comp* attribute.

Args: x: float or PyoObject

New *comp* attribute.

`input`

PyoObject. Input signal to process.

`comp`

float or PyoObject. Comparison value.

Wrap

class **`Wrap`**(*input*, *min=0.0*, *max=1.0*, *mul=1*, *add=0*) [source]

Wraps-around the signal that exceeds the *min* and *max* thresholds.

This object is useful for table indexing, phase shifting or for clipping and modeling an audio signal.

Parent: `PyoObject`

Args: input: PyoObject

Input signal to process.

min: float or PyoObject, optional

Minimum possible value. Defaults to 0.

max: float or PyoObject, optional

Maximum possible value. Defaults to 1.

Note: If *min* is higher than *max*, then the output will be the average of the two.

```
>>> s = Server().boot()
>>> s.start()
>>> # Time-varying overlaping envelopes
>>> env = HannTable()
>>> lff = Sine(.5, mul=3, add=4)
>>> ph1 = Phasor(lff)
>>> ph2 = Wrap(ph1+0.5, min=0, max=1)
>>> amp1 = Pointer(env, ph1, mul=.25)
>>> amp2 = Pointer(env, ph2, mul=.25)
>>> a = SineLoop(250, feedback=.1, mul=amp1).out()
>>> b = SineLoop(300, feedback=.1, mul=amp2).out(1)
```

`input`

PyoObject. Input signal to process.

`max`

float or PyoObject. Maximum possible value.

`min`

float or PyoObject. Minimum possible value.

`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.

`setMax`(*x*) [source]

Replace the *max* attribute.

Args: x: float or PyoObject

New *max* attribute.

`setMin`(*x*) [source]

Replace the *min* attribute.

Args: x: float or PyoObject

New *min* attribute.