

Jasuto Pro

manual [version 1.0]

The manual will be completed the first week of January 2010, sorry Apple really surprised me with an unbelievably quick release.

See below for a quick start guide and a node reference chart for now.

Quick start guide

1. Creating an oscillator

Tap the “Create” menu, this will show you all of the module categories. Tap the “Synth” menu, this will show you all of the oscillators available. Tap the “Sin” button at the bottom, this will add a “Sin” oscillator where the current scene cursor is (the spinning cross).

2. Creating a speaker

Now that we have a sound source we need to add a node that can process this and send it to the devices output so we can hear it. Tap in an empty spot in the scene (somewhere black) and drag upwards to make a little room underneath the “Sin” node. Double tap under the “Sin” node to place the cursor. This is where the next node will be created.

If the “Create” menu is closed, tap it to reopen it, tap the “IO” menu and finally tap the “Speaker” node. This will add the “Speaker” node where the cursor is, just under the “Sin” node. You can tap and drag a node to move it, do this to give the nodes a little room.

3. Making a connection

Now that we have both a generator and somewhere to output it, we need to create a directed connection between the two. Signals flow from the top down, so that implies that the bottom node the “Speaker” is the output.

To wire these nodes double tap an empty space somewhere above and to the left of the “Sin” node, make sure to hold your finger down on the 2nd tap and drag your finger just below and to the right of the “Speaker” node, you should see a light gray box encompassing both nodes, now release your finger. You’ve just selected both nodes, you can tell if you were successful because both of the nodes will have red names.

With both nodes selected, tap the “Edit” menu, you’ll see several operations you can do on nodes, we are interested in the “Wire” button. Go ahead and tap that, you should now see a downward moving wire between the two nodes. If you don’t then you either had the nodes too far/close together or they both weren’t selected.

Once you’ve wired the nodes you should here the “Sin” playing back, just a pure simple tone. The distance of the “Sin” node to the “Speaker” effects output level. Try to adjust the level by moving either node apart.

4. Changing the pitch

Now that you have a tone playing back we can adjust it's frequency. Double tap the "Sin" node. Each node has an editor where various settings can be controlled. You will see an element that says "fm", this stands for frequency modulation and is how you control the pitch of the oscillator.

Go ahead and tap/hold the "fm" element, this is an example of a numeric slider. While holding this slider element down you can drag your finger left/right, up/down to adjust it's value. If you are following this correctly you should hear the pitch of the oscillator change while you drag your finger around.

5. Playing your synth

Setting the pitch of the oscillator numerically will only get you so far. What we need to do is add a midi style keyboard to the scene to allow us to play the synth in a more traditional manner.

To do this drag the entire scene down a little to create some space above the "Sin" node, double tap just above (in the newly created space) to place the cursor.

Tap the "Create" menu, then the "IO" menu, then the "Keys" button. This will add a midi'ish keyboard controller to the scene named "Keys". Wire the "Keys" node to the "Sin" node, make sure there is a wire

between the “Keys” and “Sin” node. Once you have verified there is a connection between them move the “Keys” node so that it just overlaps the “Sin” node.

Remember the distance of the nodes effect their values, and we want a true pitch, so we need to insure that the “Sin” node is receiving 100% of the keyboards output pitch by overlapping them just slightly.

Now that we’ve made a connection the output should have stopped, that’s because the “Keys” node is now controlling the oscillators pitch and we aren’t playing any notes.

To play a note, double tap the “Keys” node. You should see a little keyboard, albeit a very simple one. Go ahead and push one of the gray keys, you should here the oscillator outputting again. Once you remove your finger the oscillator will stop, you can tap the “Hold” toggle to tell the “Keys” node to hold the last note you pressed.

To control the current octave tap the “Up/Down” button, you’ll see the octave label in the bottom-left of the screen reflect this change.

Node reference

Signal ports are formatted as <"port name">, they clip the input (after the gain is applied) to -1..1.

Parameter ports are formatted as ["port name"] and they clip the input (after the gain is applied) to 0..1.

Math modules

Sig2Cv: "Signal to Control Voltage"

$$\langle \text{output} \rangle = \langle \text{input} \rangle * 0.5 + 0.5$$

Use this node to convert any bipolar signal into it's unipolar equivalent. Useful for converting any of the oscillators (including samplers) into LFOs.

1/x: “Inverse of X”

$$\langle \text{output} \rangle = 1.0 / \langle \text{input} \rangle$$

Use this node anytime you need to divide a signal, used mostly in scripts.

1/sqrt: “Inverse Square Root”

$$\langle \text{output} \rangle = 1.0 / \text{sqrt}(\langle \text{input} \rangle)$$

Use this node to find the inverse square root of the input signal, used mostly in scripts.

Const: “Constant”

$$\langle \text{output} \rangle = \langle \text{value} \rangle$$

Use this node to set a ports value to a constant, useful when used with recorded motion data.

Madd: “Multiply Add”

$$\langle \text{output} \rangle = \langle \text{input} \rangle * \langle \text{mul} \rangle + \langle \text{add} \rangle$$

Use this node for all sorts of effects, e.g. mapping a signal into a certain range, ring modulation, etc...

Msub: “Multiply Subtract”

$$\langle \text{output} \rangle = \langle \text{input} \rangle * \langle \text{mul} \rangle - \langle \text{sub} \rangle$$

Use this node for all sorts of effects, e.g. mapping a signal into a certain range, ring modulation, etc...

Delta: “Discrete Time Derivative”

$$\langle \text{output} \rangle = \langle \text{input} \rangle - \langle \text{input-1} \rangle$$

Use this node to detect edges in signals, mostly used in scripts.

xFade: “Crossfade”

$$\langle \text{output} \rangle = (\langle \text{inputX} \rangle * [\text{amount}]) + (\langle \text{inputY} \rangle * (1 - [\text{amount}]))$$

Crossfades between the two inputs signals (X and Y). An amount of 0 will output 100% of inputX and 0% of inputY, an amount of 1 will output 0% of inputX and 100% of inputY.

LtEqGt: “Less than 0, Equal to 0, Greater than 0”

$\langle \text{GT} \rangle = \langle \text{input} \rangle > 0.0$

$\langle \text{equal} \rangle = \langle \text{input} \rangle == 0.0$

$\langle \text{LT} \rangle = \langle \text{input} \rangle < 0.0$

Use this node to do simple comparisons. If the condition is satisfied the output will be 1.0 else it will be 0.0, used mostly in scripts.

Synthesizer modules

Sin: “Sinusoidal Oscillator”

$\langle \text{output} \rangle = \sin(\langle \text{fm} \rangle * t + \langle \text{pm} \rangle)$

The purest tone generator. Use the “fm” port to control the fundamental frequency, this can be a constant value or another oscillator to achieve *frequency modulation*. The “pm” port can be used to phase shift the signal by a constant or another signal (*phase modulation*). The “sync” port controls another internal oscillator’s frequency. This is disabled when the port’s value is 0. When this is

activated the sync oscillator becomes the master and resets the slave (the audible) oscillator's phase each cycle. Modulate the "fm" port when sync is enabled to hear the effects.

Saw: "Sawtooth Oscillator"

$$\langle \text{output} \rangle = \text{saw}(\langle \text{fm} \rangle * t) + (\sin(\langle \text{fm} \rangle / 2 * t) * [\text{sub}])$$

This tone generator contains a rich set of harmonics. Use the "fm" port to control the fundamental frequency, this can be a constant value or another oscillator to achieve *frequency modulation*. The "sub" port can be used to add subsonic bass to fatten the sound up. The "sync" port controls another internal oscillator's frequency. This is disabled when the port's value is 0. When this is activated the sync oscillator becomes the master and resets the slave (the audible) oscillator's phase each cycle. Modulate the "fm" port when sync is enabled to hear the effects.

Tri: "Triangle Oscillator"

$$\langle \text{output} \rangle = \text{tri}(\langle \text{fm} \rangle * t) + (\sin(\langle \text{fm} \rangle / 2 * t) * [\text{sub}])$$

This tone generator contains all odd harmonics which gives it a hollow sound. Use the "fm" port to control the fundamental frequency, this can be a constant value or another oscillator to achieve *frequency modulation*. The "sub" port can be used to add subsonic bass to fatten the sound up. The "sync" port controls another internal oscillator's frequency. This is disabled when the port's value is 0.

When this is activated the sync oscillator becomes the master and resets the slave (the audible) oscillator's phase each cycle. Modulate the "fm" port when sync is enabled to hear the effects.

Square: "Square/Pulse Oscillator"

```
<output> = pulse(<fm> * t, [width])
```

This tone generator contains all odd harmonics which gives it a hollow sound. Use the "fm" port to control the fundamental frequency, this can be a constant value or another oscillator to achieve *frequency modulation*. The "width" port can be used to control the width/duty cycle of the pulse. The "sync" port controls another internal oscillator's frequency. This is disabled when the port's value is 0. When this is activated the sync oscillator becomes the master and resets the slave (the audible) oscillator's phase each cycle. Modulate the "fm" port when sync is enabled to hear the effects.

Noise: "White Noise Oscillator"

```
<output> = lowpass(random(), [color])
```

This tone generator outputs a random bipolar stream of numbers, use the "color" port to darken the sound.

Lfo: “Low Frequency Oscillator”

$$\langle \text{output} \rangle = \sin(\langle \text{fm} \rangle * t + \langle \text{pm} \rangle) * 0.5 + 0.5$$

This generator is the same as the “Sin” node except it has a lower frequency range and the output is scaled and shifted to be in the control signal range 0..1.

Filter modules

LP: “24dB Low Pass Filter”

$$\langle \text{output} \rangle = \text{lowpass}(\langle \text{input} \rangle, [\text{cutoff}], [\text{resonance}])$$

This filter will attenuate all high frequency content above the “cutoff” limit. The “resonance” amplifies frequencies near the “cutoff”.

HP: “24dB High Pass Filter”

$$\langle \text{output} \rangle = \text{highpass}(\langle \text{input} \rangle, [\text{cutoff}], [\text{resonance}])$$

This filter will attenuate all low frequency content below the “cutoff” limit. The “resonance” amplifies frequencies near the “cutoff”.

BP: “24dB Band Pass Filter”

<output> = bandpass(<input>, [cutoff], [resonance])

This filter will attenuate all frequencies that exist outside of the band centered around the “cutoff” frequency. The “resonance” amplifies frequencies near the “cutoff”.

Moog: “24dB Low Pass Filter”

<output> = moog(<input>, [cutoff], [resonance])

This filter will attenuate all high frequency content above the “cutoff” limit. The “resonance” amplifies frequencies near the “cutoff”. This is similar to the LP filter but it has a more warm/analog sound.

Signal modules

Comp: “Compressor”

`<output> = compressor(<input>, [threshold], [ratio])`

This module is used to control the dynamic range of a signal. You can use it to prevent clipping, add new life to a sample, etc. The “threshold” port controls the level that the compressor is activated. The “ratio” port controls how aggressive the compressor will try to attenuate the signal. The “input” port isn’t clipped as it can handle signals outside of the -1..1 range.

Gate: “Gate”

`<output> = gate(<input>, [threshold], [release])`

This module is used to attenuate signals less than the “threshold” level. You can use it to prevent background noise when using the microphone or another line device. The “release” port is used to smooth the gating effect, reduce this if you need a faster response.

Limitter: “Limitter”

<output> = limiter(<input>, [threshold], [release])

This module is used to control the ceiling of a signal. You can use it to prevent clipping without losing the overall loudness of a signal. Very useful as the final stage of a scene. The “input” port isn’t clipped as it can handle signals outside of the -1..1 range.

Follow: “Signal Follower/Detector”

<output> = follow(<input>, [attack], [release])

This module is used to track the overall amplitude/signature of a signal and output it as a control signal. This can be used for things like converting the microphone into a breath controller, smoothing the accelerometer values and so on. The attack/release work together to control the response time, the smaller these values are the faster the follower will react, at the expense of an increased noise level.

AR: “Attack Release Envelope Generator”

`<output> = ar(<input>, [attack], [release])`

This module is used to generate a simple envelope with only an attack and release stage. It must be within the range of another node that sends trigger-on pulses such as the: “Trigger”, “Seq”, “Keys”. When this node is triggered it’s internal phase is reset and the the attack stage is started, once the attack stage has peaked the release stage is started and continues until the level reaches a zero.

ADSR: “Attack Decay Sustain Release Envelope Generator”

`<output> = adsr(<input>, [attack], [release])`

This module is used to generate a more complex envelope. It must be within the range of another node that sends trigger-on/off pulses such as the: “Trigger”, “Seq”, “Keys”. When a trigger-on pulse is received the attack stage is started and continues until the end of the decay stage. At this point if a note is being held the sustain level will be the output value. Only when the note is released and the trigger-off pulse is received will the release stage begin. The key concept here is that you can have sustained notes by using the ADSR where the AR envelope completes it’s cycle autonomously.

Effects modules

S&H: “Sample and Hold”

`<output> = lowpass(s&h(<input>, [rate]), [smooth])`

This module is used to sample a signal at the frequency specified by “rate”. The “smooth” port controls how smooth the transition is from adjacent samples. If nothing is connected to the “input” port then it samples a white noise stream, otherwise it samples the input. You can use this to generate slowly varying control signals.

Delay: “Interpolating Delay line”

`<output> = delay(<input>, [time], [feedback])`

This module is the basis for many different effects and is definitely the workhorse when it comes to effects. The “time” port controls the length of the delay line, this can be modulated without any artifacts. The “feedback” port controls how much of the signal is mixed back in with the current “input”. You can set the “gain” to a negative value for negative feedback effects. To set the time constant to a beat value, double tap it then use the “Beats” mode in the number editor.

Phaser: “6 Stage Analog Phaser”

<output> = phaser(<input>, [rate], [feedback])

This module is used to add tiny phase shifts to a signal which gives it a swooshing quality. The “rate” port controls the frequency of its internal LFO and “feedback” controls how “metallic” the phaser sounds. Phasers sound best when used with signals that are rich in frequency content, e.g. white noise.

RVerb: “Reverberation”

<output> = reverb(<input>, [size], [dampening])

This module is used to add space/dimension to a signal. The “size” port controls the size of the room and the “damp” port controls how bright the room should sound. Rooms made of more dense material would have a smaller dampening factor. The reverb effect processes in stereo and you can access the stereo channels via the “left” and “right” ports.

Analog: “High Order Distortion”

`<output> = analog(<input>, [gain], [feedback])`

You can use this module to add warmth to a signal, it's somewhat subtle but sometimes that's all you need. The “gain” port is used to adjust the level of the pre-effected signal and “feedback” controls how much of the signal is mixed back in.

Fold: “Foldback Distortion”

`<output> = fold(<input>, [threshold], [feedback])`

This module wraps the signal around the positive and negative “threshold” values, giving a very interesting distortion effect especially when used with simpler signals.

Lofi: “Decimator/Quantizer”

`<output> = lofi(<input>, [rate], [bits])`

This module is similar to the S&H in that it samples the input signal at the “rate” frequency, it also quantizes the level of the signal. If “bits” is set to 1 then it’s uses the entire 32bits of the sample, 0.5 -> 16bits, 0 -> 0bits. Use this effect to get that old-school lofi sound made famous by many of the old gaming consoles.

Shaper[1,2,3]: “High Order Shapers”

`<output> = shaper(<input>, [gain], [feedback])`

You can use these modules to add warmth to a signal, they are more aggressive than the other distortion units. Use the “gain” and “feedback” ports to control the intensity of the effect. Try adding a shaper after a filter node to get a feel for how they can transform a sound.

IO (input/output) modules

Keys: “MIDI’ish Keyboard”

<mod> = keys().modulation()

<pitch> = keys().pitch()

<gate> = keys().gate()

This module is used to send pitch, note-on/off, modulation signals into the scene. The “mod” port is controlled by the sliding your finger vertically on a note, this can be used to open filters, etc... The “pitch” port is generally wired to an oscillator to control it’s pitch in a traditional manner, you can slider your finger horizontally for pitch bends. The “gate” port is controlled by hitting a note, holding it, and finally releasing it. When a note is pushed the “gate” port will be set to 1 to a single sample. When you release the note a -1 will be sent for a single sample. These trigger pulses are also sent out into the scene in 2D space. You can control the range of this pulse by using the node settings page, a value of 0 will disable the the 2D ranged pulses.

The “Hold” button tells the keyboard that it should continue to output the last frequency touched. Make sure this is on if you are using an AR or ADSR envelope.

The “Glide” slider controls the portamento of sequential notes. A value of 1 disable portamento, smaller values increase the amount of time it takes a note to arrive at it’s specified pitch.

The “Up/Down” buttons change the current octave that the keyboard will playback at, you can see the current octave in the lower left of the screen.

Seq: “64 Step Sequencer”

<mod> = seq().modulation()

<pitch> = seq().pitch()

<gate> = seq().gate()

This module is used to record sequences of notes and/or modulation. Each step is a 16th note and the overall tempo is controlled by the “tempo” slider found in the “xPort” menu.

If you wire a “Keys” module to the “record” port of the sequencer it will record your notes and modulation into the sequencer. The “xpose” allows you to procedurally transpose the notes without effecting the stored note data. The “length” port allows you to control the end loop point of the sequence, this can be used for all sorts of swing type effects.

The sequencer outputs trigger-on/off pulses just like the “Keys” module. To disable the 2D ranged pulses, set the “range” slider to 0 which is found in the node settings page.

The “Start/End” sliders control the endpoints of the loop.

The “<<<“ and “>>>” allow you to cycle through all the pages of the sequence. The number in the middle of them shows you the current page, if you tap this number it will activate the auto-page mode. This will automatically show you the current page that is being played back.

The “Up/Down” buttons control the current octave that is being displayed.

To enter notes you can just tap a single note or you can slide your finger to draw a sequence of notes. This also works for the modulation data below.

The modulation data can be used for things like velocity, filter cutoff, etc... If you tap a modulation slider it will turn red to indicate that the note shouldn't send out a note-on pulse. You can use this to achieve glissando.

In the edit menu you will find useful operations that pertain to the current bar or the entire pattern. The notes sub-menu has all the controls the are “Keys” module, and allow you to control how notes are processed.

Tempo: “Global Tempo”

```
<output> = tempo([BPM])
```

You can use this module to override the scenes current tempo. This can be used along with a sequencers modulation data to slightly vary the tempo on certain beats, like the old-school trackers. This is a cheap way to add a “groove” to the scene...

Speaker: “Device Output/DAC”

```
system.output(<left> + <mono>, <right> + <mono>)
```

This is the only output module, you need to have one of these in your scene in order to hear anything. You can have multiple speaker modules in your scene, they are all summed/clipped and output to the DAC. The final clipping stage is a hard limiter so make sure you add your own limiter/compressor to keep the levels sane. Anything sent to the “mono” port will be equally sent to the left and right stereo channels.

Mic: “Microphone Input/ADC”

```
<left> = system.inputLeft();
```

```
<mono> = (system.inputLeft() + system.inputRight()) / 2 ;
```

```
<right> = system.inputRight()
```

This module gives you access to the microphone/line input. This allows you to make guitar effects, etc... If your input device only supports mono then the signal will be routed to the “left” port.

Send/Recv: “Send/Receive”

```
recv[“name”]<input> = send[“name”]<output>
```

These modules allow you to “wirelessly” send signals in within the scene. You can use this if you need to send a signal to a separate group, feedback effects, etc... When you create a send node it will automatically create a unique name for the send, so you will see the nodes name as “S: 0” the “0” is the name of the send. You can manually change the name by double tapping the node and tapping the “Name” button. Any recv node you create will inherit the last sends name, you can also set the name manually by double tapping as well. The send node also acts as a passthrough, so you can wire it directly in a patch and it will copy it’s input to it’s output.

Accel: “Accelerometer”

<x> = system.accelerometerX()

<y> = system.accelerometerY()

<z> = system.accelerometerZ()

This module gives you access to the devices accelerometer. The values are updated @ 30hz which and are rather noisy. To get a nice smooth output wire in a “follow” node, this will slow the response time but it will give you a really smooth control signal to work with.

Trigger: “Trigger-on/off”

This module is used with nodes that output a gate signal. When the trigger receives a positive pulse it sends out a ranged note-on, when it receives a negative pulse it sends out a note-off pulse. All the nodes that are within it’s range will receive the note-on/off message. This module is really in here for legacy reasons and Jasuto Pro allows the “Seq”, “Keys” to generate the ranged pulse directly.