Max/MSP Seaboard Tutorial Notes

Welcome!

This folder contains seven Max/MSP patches that explain how Seaboard MIDI data works, how it can be parsed in Max/MSP and how to use it to control a synthesiser or external application. This includes demonstrations of the Seaboard's unqiue polyphonic Pitch Bend and Aftertouch, and how to output Seaboard data to external applications with using the Open Sound Control (OSC) protocol.

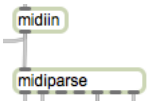
Before opening a patch, please power up and connect your Seaboard using the power switch on the Back Panel (the Seaboard takes around 15 seconds to launch; the SoundDial will settle once launch is completed). Each patch opens in Presentation Mode, and is fully interactive – just start playing the Seaboard. To start hacking, enter Edit Mode to reveal the patching structure.

Each patch is annotated with comments, so if you are already familiar with Max then they may be sufficient to get you hacking. The comments document how and why each object was used, and will help you create your own Seaboard patches. If you're new to Max, the below articles should be helpful. Happy hacking!

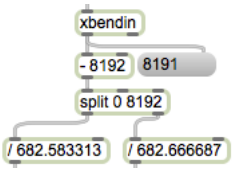
Table of Contents

Tutorial 1: Seaboard MIDI Data

This patch shows how to select the Seaboard as a MIDI input device and how to input MIDI data from it into a Max patch using the 'midiparse' object. This object will allow to input MIDI pitch and velocity data, polyphonic Aftertouch and the MIDI Channel being used.



Instead of using the 'midiparse' object for Pitch Bend, the 'xbendin' object is used to be able to easily input 14 bit Pitch Bend from the Seaboard.

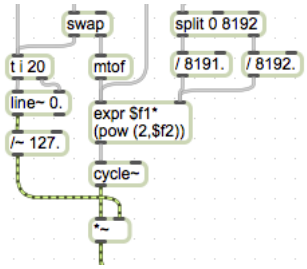


Tutorial 2: Parsing Seaboard MIDI Data

This patch shows how the data from the Seaboard can be split into individual channels for each note played. By playing multiple notes it can be seen that Aftertouch and Pitch Bend are polyphonic and apply individually for each note. To achieve this you can see that each note played is on a different MIDI Channel, using up to 10 channels.

Tutorial 3: Mono Sine Synth

This patch shows how a simple monophonic sine synthesiser being can be controlled by the Seaboard. To avoid any clicks when a note is released, a 'line~' object is added to ramp down the velocity to 0 over 20ms. In a more advanced synth patch, this could be replaced with an amplitude envelope that would give more control over the amplitude of a note.



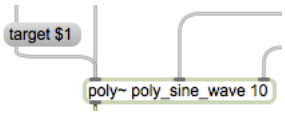
Continuous Aftertouch is applied to the amplitude of each note, varying the level of a note played after the initial velocity value. Pitch Bend is then applied to vary the frequency of the Oscillator by ± 1 Octave. This is achieved by compressing the 14-bit Pitch Bend from -8192 to +8191 to -1 to +1. The formula:

base frequency x 2^(Pitch Bend)

is used to double the base frequency for a +1 octave Pitch Bend and half the frequency for a -1 octave Pitch Bend.

Tutorial 4: Poly Sine Synth

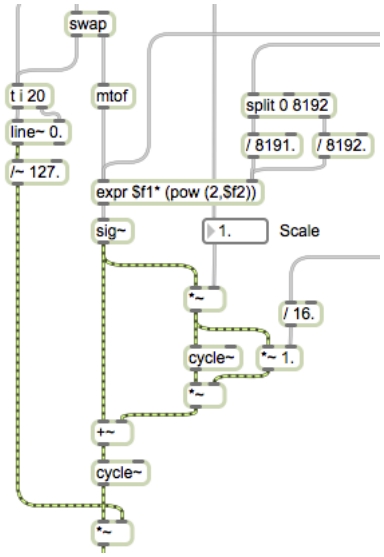
The simple sine synth created in the previous patch is now encapsulated into a 'poly~' object to allow multiple instance of the patch to be used simultaneously. As the Seaboard uses up to 10 MIDI Channels, 10 voices of the original sine synth are created.



To create polyphonic Aftertouch and Pitch Bend, each voice needs to input note data, Aftertouch and Pitch Bend for a specific MIDI Channel. For instance, Pitch Bend, Aftertouch and note data on MIDI Channel 1 will only go to the first voice of the patch. To achieve this, the message 'target $1' is sent to the first inlet of the 'poly~' object, where $1 will be replaced by the current MIDI Channel.

Tutorial 5: Mono FM Synth

This patch shows how a simple monophonic Frequency Modulation (FM) synthesiser can be controlled by the Seaboard.



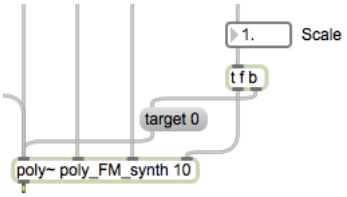
As before, Pitch Bend is set to modulate the pitch of the oscillator by ± 1 octave. Aftertouch however is no longer assigned to the amplitude of the note played, but instead controls the modulation depth of the modulating frequency. This changes the timbre of the sound created, and works well at creating a pseudo frequency cut-off, where adding more pressure opens up the filter further.

There is also control over the harmonic ratio, or the scale between the carrier and modulation frequency. Having the value of the scale as an integer value will create a consonant sound, while having the value in-between integers will create more of a dissonant sound.

Tutorial 6: Poly FM Synth

The FM patch created in the previous tutorial is now encapsulated into a 'poly~' object similar to Tutorial 4, creating a 10 voice Polyphonic FM Synthesiser, with full polyphonic Aftertouch and Pitch Bend from the Seaboard.

The difference with this patch and Tutorial 4 is that there is an extra parameter added that controls the Scale between the Carrier and Modulating frequency. As this is a parameter that is local to all voices, every time this value changes, a message of 'target 0' needs to be sent to the left inlet of the poly~ object to update all voices. If this is not there then, it will only update the current instance being used.



This is something to consider when designing your own polyphonic synths, as this is needed if you want to have control of parameters over all instances, such as envelopes or oscillator types.

Tutorial 7: Open Sound Control (OSC) Output

This patch shows how the 'udpsend' object can be used to output any data from Max to external applications with the use of Open Sound Control

(OSC). When using OSC, the IP address of the receiving application or device needs to be set, as well as the port it is going to be send to. This can be manually updated in this patch using the text boxes provided.