Max and Sequences

Max is not a good sequencer. Cuebase, Logic, Performer and the like all give you high resolution recording and playback of both MIDI and audio with user friendly interfaces. However, it is often necessary in Max to do sequencer like things such as record streams of MIDI data and play them back in time. While they are in Max the temptation to edit is almost irresistible. A question that often appears on the Max list is "Why doesn't someone write a real sequencer object?". The answer is that it would be a lot of work, and duplicate what is already easily available in other ways. But the outlook isn't all that depressing. Here's how to deal with sequences in Max.

First it helps to understand what a sequence is: a list of MIDI messages and timing information. There is a standard file format called SMF that allows sharing of these files by various applications. Most sequencers implement the concept of tracks, analogous to the tracks on a tape recorder. This enables the user to build complex polyphonic pieces a layer at a time. Tracks are not necessarily associated with MIDI channels, but they may be. There are actually 3 types of standard MIDI file.

Type 0 contains a single track
Type 1 contains multiple tracks that play at the same time
Type 2 contains multiple tracks that play sequentially, drum machine style.

Capture and Play with seq

The easiest way to just grab some MIDI data is with Coll. The tutorial Max and Rhythm demonstrates how to do this. The next step up is the seq object.

Seq is very simple and reliable. It accepts midi data directly from a midiin object, or Max output created by makenote or midiformat. These commands control it:

- Record begins recording data
- Stop ends recording
- Bang plays the recording back
- Start 1024 begins playback at normal speed (512 is half speed, etc)
- Delay n will adjust the time of the first event to be n (otherwise, it's the time between clicking record and something happening.)
- Write saves the sequence
- Read loads a sequence.

Seq can read and write type 0 files or its own text format. You can edit the text files, or use another program to edit MIDI files, although very few support type 0 MIDI files any more.

Editing text format files from seq is a bit puzzling. You will see a column of numbers like:

1380 144 69 58;

1680 144 69 0;
1730 144 67 60;
2055 144 64 67;
2095 144 67 0;
2340 144 64 0;

This may remind you of the contents of a coll, and with good reason. The two objects share the formatting code. What you have here is a time in milliseconds, followed by the data to be put out. You can fix wrong notes, delete notes and adjust times. Remember that 144 represents the note on + channel number message, so to move a note to channel 2 you'd put 145.

Follow

Follow is a slightly elaborated seq. In addition to the seq commands it has hook, which allows you to change the speed of playback as you play. The argument to hook is a ratio, so hook 0.5 will slow things by half.

The follow n message puts the object into score following mode. N refers to which message to watch for initially. (The first has an index of 0.) Follow waits for a message that matches, sends the index, and waits for the next one. Follow looks ahead a couple of messages and jumps if one is skipped. All of this lets the patcher know where the performer is in their part.

Follow is not limited to MIDI style messages. You can put note numbers or any other data into it.

Outside help

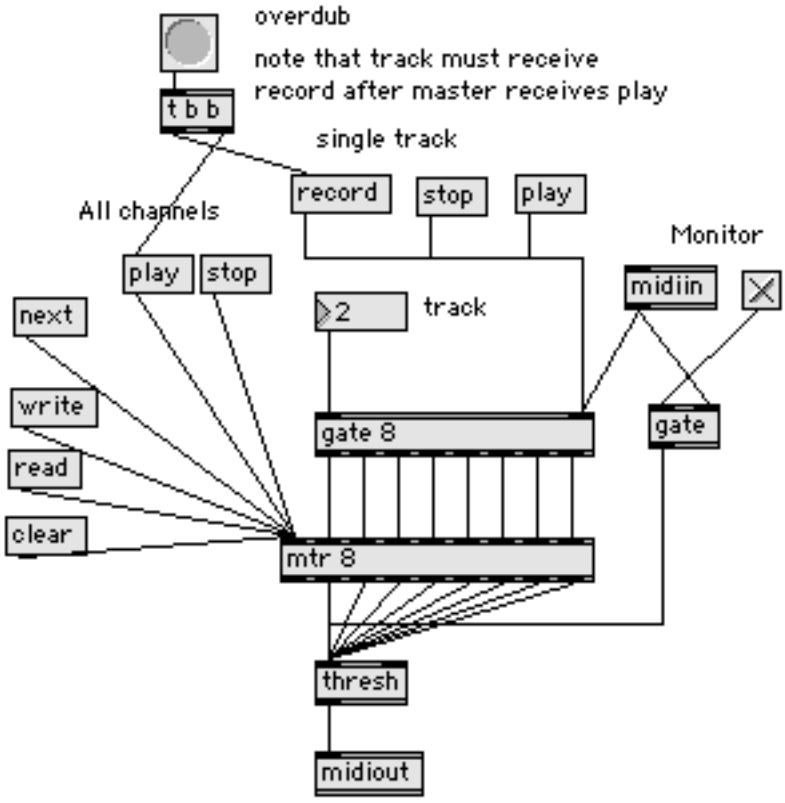
Most veteran Maxers prefer to use Eric Singer's seqPlayer object to play sequences recorded elsewhere. It's still limited to type 0 MIDI files, but has a lot of nice features. Primarily, it is beat driven, and you provide the beats (after defining a tempo, of course). This way it can speed up and slow down to match a performance. It also does loops and other neat things. Since it's not in the Max documentation, I've appended the instructions to this tutorial. SeqPlayer does not record.

Multitrack recording.

The next step up is MTR. This is just a data recorder, not a midi recorder per se, but you can easily record MIDI data with it. You have to do a bit of building to get just what you want, but it does allow overdubbing. It has an inlet for each track as well as an inlet for master control. Commands to the master affect all tracks, but you can send play 2 to the master and just hear track 2

- Record begins recording data
- Stop ends recording
- Bang plays the recording back
- Play begins playback at normal speed
- Mute and unmute control individual channels
- Delay n will adjust the time of the first event to be n (otherwise, it's the time between clicking record and something happening.)
- Write saves the sequence
    - Read loads a sequence.

The command next steps through the recorded data. It also outputs the time until the next data point, which you could use to control a metro, as described below for detonate. Here's a simple 8 track patch:



As you can see, you need to direct the commands to the proper inlets. The thresh is necessary to get the data back into lists for the midiout object. If you were just recording ints this would not be necessary.

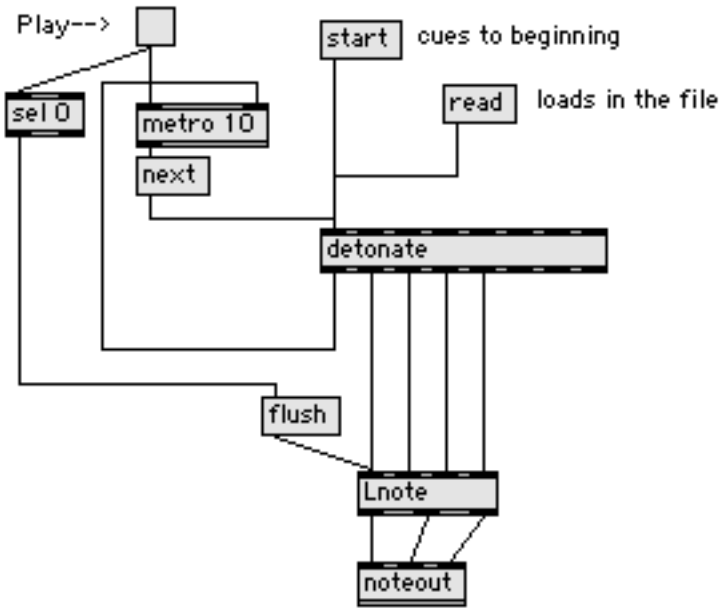
Mtr files can also be edited as text, but the formatting is a bit different. Like

Track 1;
1005 144;
0 62;
0 64;
140 144;
0 62;
0 0;

Here, the times are between events, and each int gets its own line. A midi note on message will get three lines, with 0 delta time for the second and third numbers. If you change a time, you must make a complimentary adjustment to the following time.

Detonate

Detonate does some basic recording, but most importantly, it plays type 2 MIDI files. It's really a sequencer kit (you'd need one per track to do a multi track recorder). Here's how to use it as a simple MIDI file player:



All detonate does is provide data for the current event and time to the next event when it gets the next message. This time can be fed to a metro to give durations.

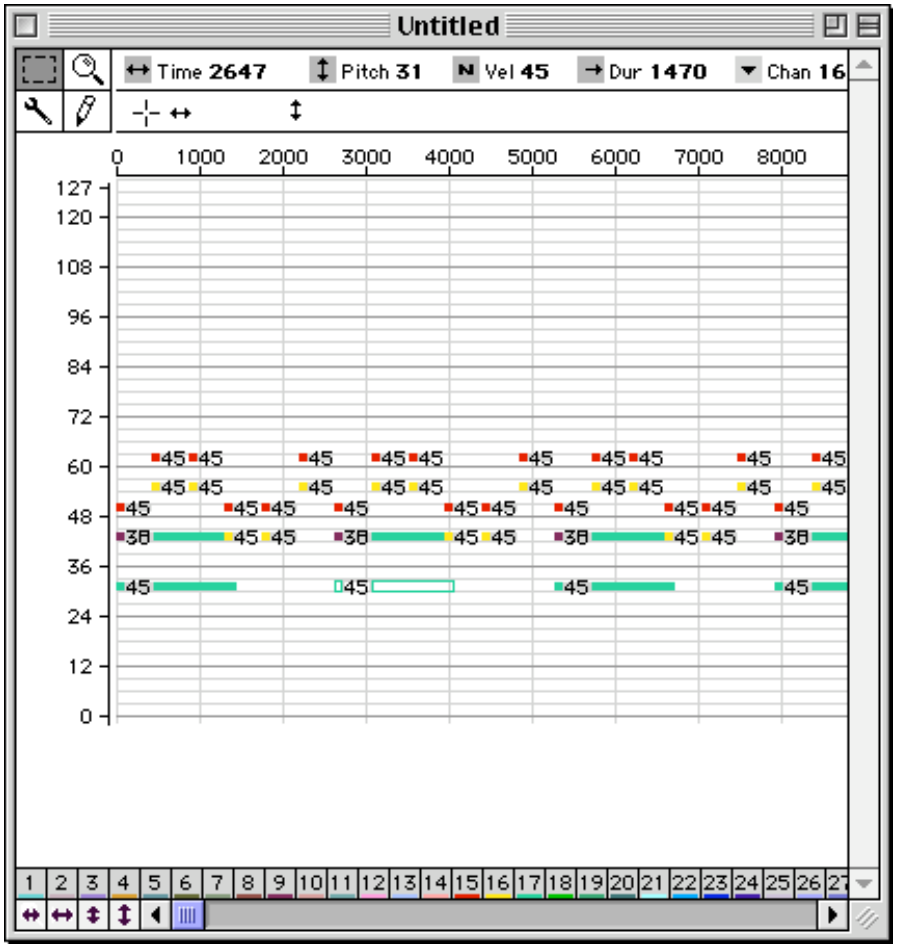
If you want to change playback speed, you have to modify the timings on the way to the metro.

You have to use Lnote to get the channels under control -- makenote would not send the note offs properly. The flush message cuts all notes when playback is stopped.

You can do score following too. Detonate is more forgiving of mistakes, depending on how some parameters are set.

One of the most interesting features of detonate is that it will open type 1 files, but saves type 0 files, so they could be played by seqPlayer.

Detonate has some very nice editing features: double click on the object to get started.

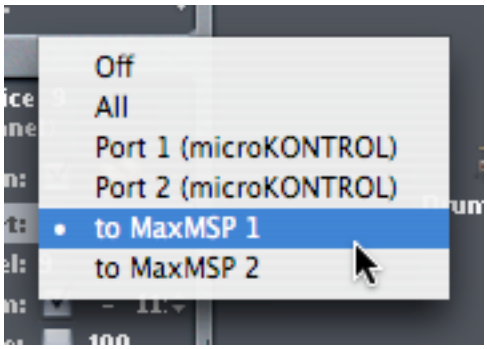


Communicating With Other Sequencers

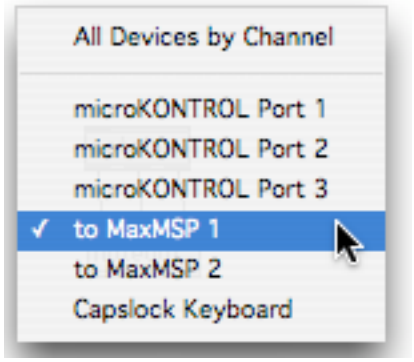
These are all useful as far as they go, but pretty lame compared to logic. For best results, you should integrate Max with a real sequencer. There are two ways to do this:

- Build your patch as a plug-in with pluggo
- Use the backdoor MIDI paths that Max installs

For instance, to send data from Logic to Max:
Launch Max before Logic. This will give two new options for output in logic. They are called "to MaxMSP" 1 and 2.
You select one of these as the output port in logic



In Max all notein etc objects can read these ports in the pop-up you get when you double click:

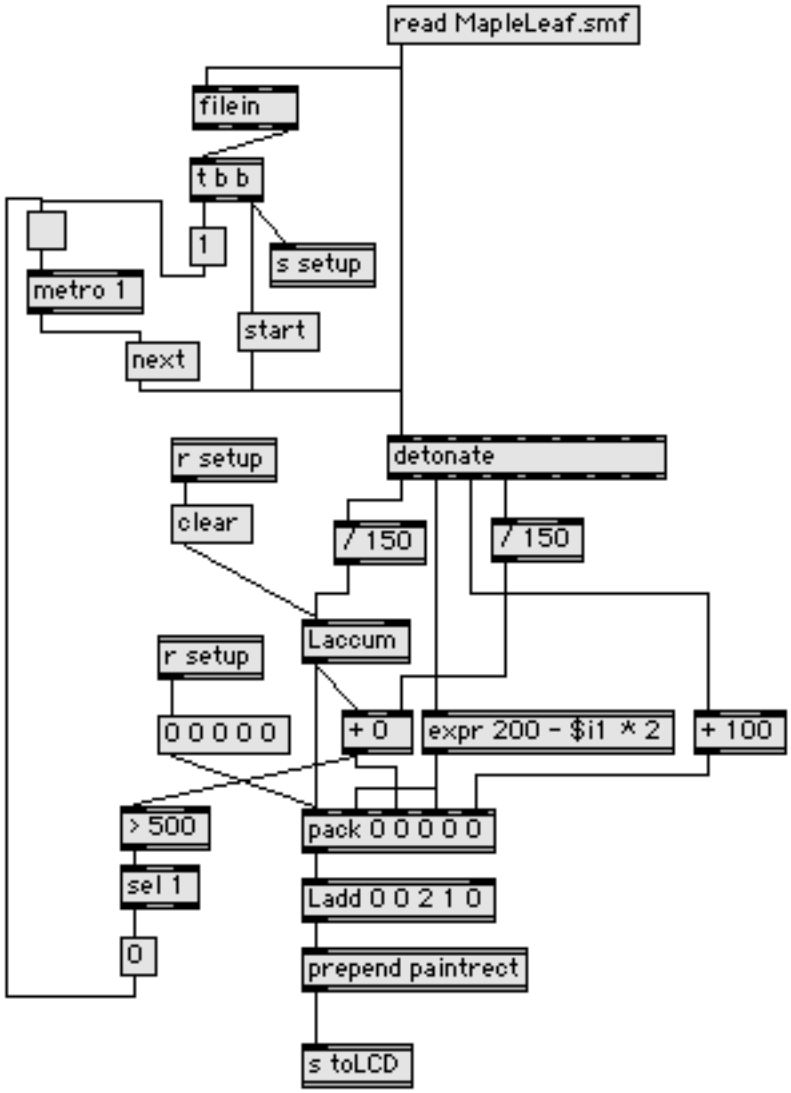


Likewise, you can send stuff the other way by selecting from maxMSP as the output port. It will just turn up in Logic with no extra settings.

Displaying Sequences

Here's a patch that shows a sequence on the LCD instead of playing it. Basically, detonate is set to play the file as fast as possible. The filein object is just there to give a bang when the file is loaded. (right to left precedence means the filein won't load until detonate has finished.) This data is captured and displayed as described in the Max and graphics tutorial. The time and duration outputs of detonate are used to set the left and right coordinates of rectangles.

It wouldn't be hard to then play the detonate in the normal way and move a cursor (as a sprite) across the display.



Appendix

SeqPlayer

eric@ericsinger.com
http://ericsinger.com/cyclopsmax.html

SeqPlayer plays Type 0 Standard MIDI Files. SeqPlayer accepts bangs and outputs the MIDI events in the sequence in a beat-by-beat manner out the left-most outlet. Each bang causes MIDI events for a certain amount of sequence time to be output. The amount of time depends on the bang resolution.

The bang resolution is set with a 'bangsperqn' or 'qnsperbang' message ("bangs per quarter note," and "quarter notes per bang," respectively). The default is one bang per quarter note, or quarter note per bang. The values of these messages are the inverse of each other and set the same parameter.

The bang resolution is used to calculate how much of the sequence to output on each bang. For example, 'bangsperqn 2' sets the bang resolution to 2 bangs per quarter note, meaning that SeqPlayer will output an eighth note's worth of time for each bang received. 'qnsperbang 2' sets the bang resolution to 2 quarter notes per bang, so SeqPlayer will output a half note's worth of time for each bang received.

Bang rate and event timing

In external rate mode (the default state, or set with the 'external' message), SeqPlayer calculates a bang rate on each bang and uses this to extrapolate timing for events. For example, say you are beating at a rate of 1000 ms, beatsperqn is set to 1, and the current beat in the sequence contains a pair of eighth notes. The first eighth-note will play immediately on the bang and the second eighth-note will play 500 ms later. The rate is calculated as the time from the previous to the current bang; in other words, it is the interval between the bangs.

In fixed rate mode (set with the 'fixed' message), SeqPlayer doesn't calculate the bang rate, but uses a fixed value instead. For example, given a 'fixed 700' message, SeqPlayer will subsequently operate in 'fixed' mode and use a rate of 700 to extrapolate timings each time a bang is received.

About other messages

'countoff' sets the number of countoff beats before starting or continuing playback. For example, upon receiving a 'countoff 4' message, SeqPlayer will "eat" the next four bangs. This must be set each time you want a countoff.

In external rate mode, when setting a countoff of 0, an initial rate can be specified as a second argument. For example, 'countoff 0 600' causes SeqPlayer to use a

rate of 600 for the initial beat. This is to get around the problem of having no previous beat from which to calculate the rate when the countoff is 0. This value is ignored in fixed rate mode.

The 'goto' message causes the sequence to jump to the specified bar and beat on the next bang. For example, 'goto 9 3' jumps to bar 9, beat 3, when the next bang is received.

'top' is equivalent to 'goto 1 1'.

'loop' causes the sequence to loop between a starting bar/beat and an ending bar/beat. For example, 'loop 5 1 10 1' causes the sequence to play between bar 5, beat 1 and bar 10, beat 1.

Looping works by jumping to the start bar/beat when the end bar/beat is reached. The jump occurs on a bang. Note that if the end bar/beat does not fall on a bang, then the jump will occur on the first bang after the end bar/beat is reached. 'loop' with no arguments turns off looping.

'map' remaps events from one channel to another, or turns events off for a channel. For example, 'map 1 9' causes all events occurring on channel 1 to be remapped to channel 9. 'map 1 0' turns off events on channel 1. 'map' with no arguments resets mappings to normal.

'flush' outputs note-offs for any notes which are "hanging" (ie. note-ons which haven't yet been followed by corresponding note-offs).

'autoflush' is used to set the auto-flush state. When auto-flushing is on (the default), SeqPlayer will automatically do a flush after a 'goto', 'top', 'map' or loop jump. This eliminates potential stuck notes.

'limitrate' limits the minimum and maximum rate for external rate calculations. For example, 'limitrate 100 1000' insures that the calculated rate will lie between 100 and 1000 milliseconds. This limits very short and very long durations, respectively.

About the other outlets

The second, third, fourth and fifth outlets output the bar, beat, bang count and bang rate, respectively. The bang count is the number of bangs so far in the current measure. The bang rate is the calculated time from the last to the current bang. These values are updated each time a bang is received, before any MIDI is output. Note that if the bang resolution is set to output more than one beat of metric time per bang, the beat counter will skip intervening beat values. For example, with 'qnsperbang 2' in 4/4 time, the beat counter will count 1, 3, 1, 3, etc. Also, if the bang resolution is set to output a whole measure or more per bang, the bang counter will remain at 1. SeqPlayer reads time signature information from the file so the display is updated properly.

The sixth outlet outputs a bang when the end of the sequence file is reached or an SMF end-of-sequence marker is encountered.

The seventh outlet outputs meta events as a list of ints, with the first int representing the meta event command. See the Standard MIDI File specs for more info.

Changes as of version 1.5
- now PPC-native (fat)
- added new outlet for bang rate (outlet 5)
- moved end of sequence outlet to outlet 6 to accomodate new outlet

Changes as of version 1.6
- added new outlet for meta events (outlet 7)

Changes as of version 2.1
- this and future versions are OSX only

file:///Users/peterelsea/Documents/writings/Max%20writingsG5/Max%20%26/Max%20and%20Programming