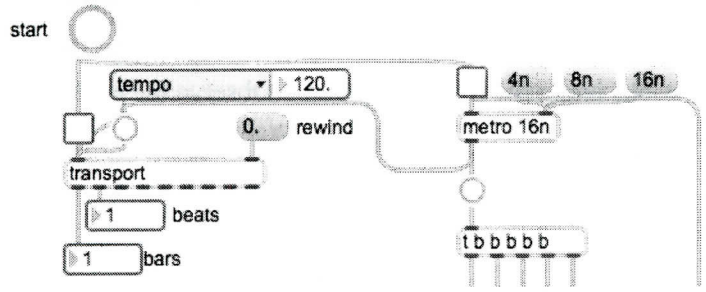
Composition: Electronic Media II
Spring 2015
Max 6: Writing MIDI with the EMS MIDI note writer patch

1. The purpose of this handout is to demonstrate the EMS MIDI note writer patch. It can be found in the EMS Max Patches folder on the studio desktops.

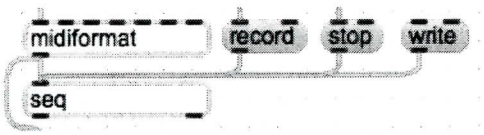
2. This patch can be broken down into three sections: the transport, pitch selection and recording, and auditioning.

3. The Transport:
   a. The transport object allows one to synchronize events in a Max patch much like the transport does in Pro Tools. The transport object is linked with metro objects that have an argument such as 4n, 8n, or 16n. These arguments allow metro objects to send out bangs based on a particular note value for a particular tempo. Observe the example below:



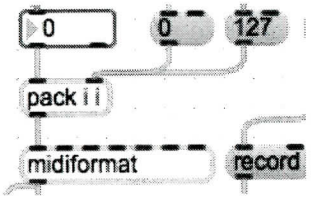
   b. The transport is set at a tempo of 120 bpm.
   c. The metro object has an argument of 16n. What this means is, when the transport is activated by the toggle, the [metro 16n] object will send out a bang every 16th note at 120 bpm. For an 8th note, click the "8n" message. For a quarter note, click on the "4n" message.
   d. This is a helpful way to send out bangs at a particular rate that will translate to a musical context.
   e. Notice that below the transport there are two integer objects. One is labeled as bars and the other as beats. This is just a way for the user of the patch to keep track of elapsed time in relation to a 4/4 time signature.
   f. *N.B.* Tempo information (and time signature information) will NOT be saved to your MIDI file. The transport is only being used as a way to generate note values at a steady and musically rhythmic rate.
   g. The rewind button will reset the transport so that the bars and beats will start over at bar 1 beat 1.

4. Pitch selection and recording:
   a. Observe the example below:



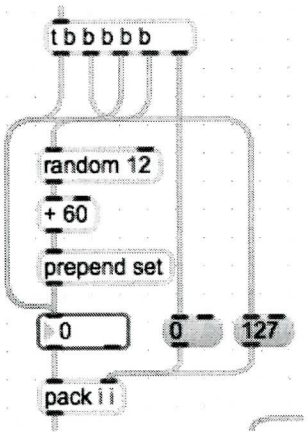
      i. The [seq] object allows one to record and save incoming MIDI data. It listens for messages to tell it when to record, stop, or save the data as a MIDI file. This patch will automatically select the "record" and "stop" messages when appropriate. The user will not have to click on "record" or "stop" messages.
      ii. The "write" message must be clicked on by the user in order to save your MIDI file. Clicking on this message will open a save-as dialog window, where you can name your MIDI file and save it to an appropriate location.
      iii. Now, we want to send the [seq] object some MIDI note data. The [midiformat] object is used to interpret incoming data as MIDI information. The first left inlet of [midiformat] interprets that data as MIDI note and on/off information.

d. Observe the example below:



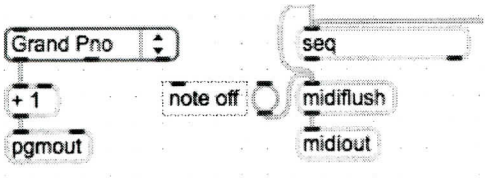
i. The [midiformat] object listens to and interprets incoming data. Notice we are only using the first left inlet. The information we send to this inlet will be interpreted as MIDI note number and note on or note off. MIDI note data must be in the form of a specific MIDI note followed by an "on" or "off" value.
ii. The [pack i i] object will pack together multiple items into a list. The number of items it will pack can be determined by the argument that it is given. The argument of two "i"s means it is expecting two integers, and it gives us two inlets. The list we need is a note number followed by an on or off.
iii. For the MIDI note number, we will use an integer box. Note on will be the "127" message, and note off will be the "0" message.

e. The order of messages that the [pack i i] object packs into a list is very important. Observe the example below:



i. At the top of the example we have a trigger object with five bangs. The bangs will be sent out in order from right to left.
ii. Bang 1 = note off
Bang 2 = integer box (pitch)
Bang 3 = select new random integer (pitch)
Bang 4 = note on
Bang 5 = integer box (pitch)
iii. The above ordering allows us to have steady note values with no rests. First, it cancels the current note value by sending a note off message. Then, it selects a new note value and a note on message.
iv. The above ordering is also significant in that with certain objects, data entering their left-most inlet triggers output. In other words, sending the [pack i i] object an integer into it's left inlet will send out a list, whether or not I have sent it the second item or not. That is why we are sending [pack i i] the note on or note off message first. The pack object will remember the message and add it to the list when we send it the pitch number and trigger output.
v. The pitch selection is done by a very basic random process. The [random 12] object will select an integer value of 0-11 at random. The [+ 60] object adds 60 to whatever value the random object selects. The [prepend set] object allows us to set the integer box with our note number without triggering output (messing up our trigger 5-bang process).
vi. For a list of MIDI note numbers, see the last page of this handout.

5. Auditioning:
   a. Observe the example below:



   b. This section of the patch will allow one to listen to the MIDI notes as they are selected.
   c. There is a drop-down menu on the left side. Clicking on the the arrows allows the user to select the general MIDI instrument of their choice.
   d. The "note off" button sends a bang to the [midiflush] object. This allows the user to stop playback of all MIDI notes. Sometimes we encounter what is called a "stuck note" when working with MIDI. This is essentially a safety kill switch.
   e. The [midiout] object allows us to hear the MIDI.