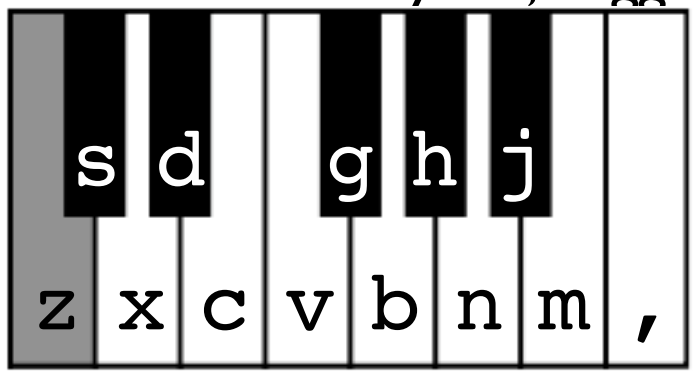
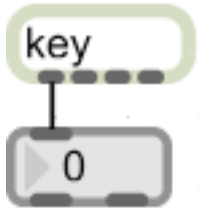
Max/MSP exercises 3a

Ex. 1

One of the tutorials you did for homework introduced the [key] and [keyup] objects. We'll make use of these objects to enable your computer keyboard to become a piano keyboard. We'll arrange it so that the keys z-, trigger the notes C-3 to C-4 as per this keyboard layout.



1. Copy the following:



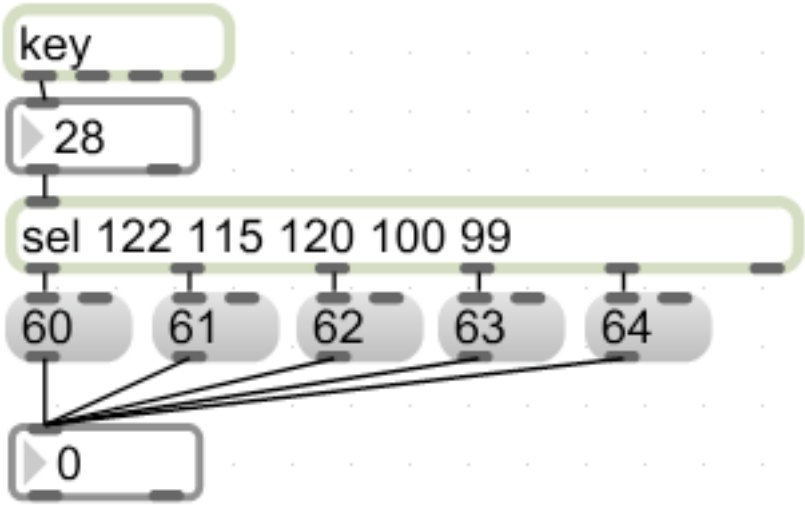
2. Lock the patch, then press the keys z-, on your keyboard as per the keyboard layout above. Note that you get ASCII numbers for each key that you press.

3. We will make a [select] object that sends a bang for each of these numbers. To make this quicker, just copy the following and paste it into a [select] object:
[122 115 120 100 99 118 103 98 104 110 106 109 44]

4. Connect the outlets of the [select] object to message boxes containing the MIDI note numbers for C-3 to C-4.

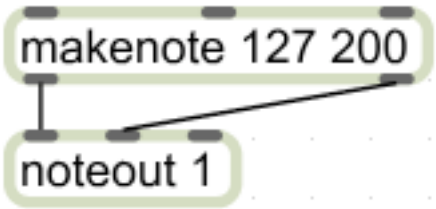
Ex.1 (cont)

You should have something like this:



(Though obviously yours should cover the full range of keys ‘z’-’,’ (60-72)).

5. You can now simply connect this to a means of generating notes from the MIDI synth:



Ex.2

So you have a means of triggering MIDI notes from your computer keyboard. But these are of a consistent length determined by the [makenote] object. What happens if you want to make them last as long as you hold the key down?

1. Duplicate the top part of your patch (i.e. the bit at the top of the last page). You can do this by selecting this and alt-click-dragging it.

2. Change the [key] object to a [keyup] object. This will mean that numbers will be sent when you *release* the key.

MIDI INTERLUDE

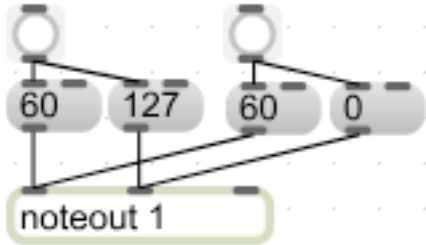
Thus far we have been using the [makenote] object to:
a) send the note number with a velocity of 127 to [noteout] to start the note;
b) pause for 200ms;
c) send the note number again with a velocity of 0 to [noteout] to *stop* the note.
This essentially gives us a **note-on** for the note, then a **note-off**.
We no longer need the [makenote] object to determine the length of the note, but we still need to send note-on and note-off messages.

3. Make the following and test it:

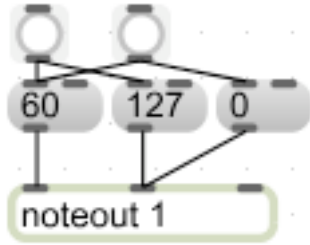
Note that sending a list to the [noteout] object is the same as sending the two numbers to the first and second inlets simultaneously.

So this:

is the same as this:



and this:



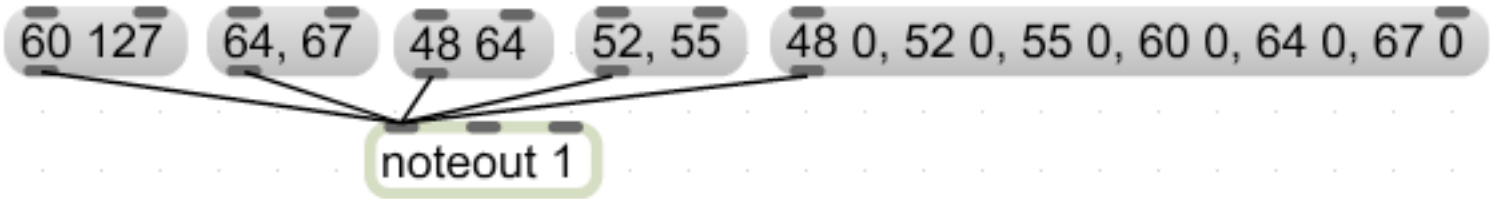
...but somewhat neater! This holds true of many objects with multiple inlets (e.g. [+], [makenote])

MIDI INTERLUDE (cont)

Notice that:
'60 127' tells [noteout] to play middle-C with a velocity of 127;
60 0 tells [noteout] to release middle-C (by giving it a velocity of 0).

All notes therefore need to be told when to start (with a velocity of 1-127) and when to stop (with a velocity of 0).

4. Copy the following:



Hit each of the [message box]es in turn (left to right) and listen carefully to the results each time.

MIDI INTERLUDE (cont)

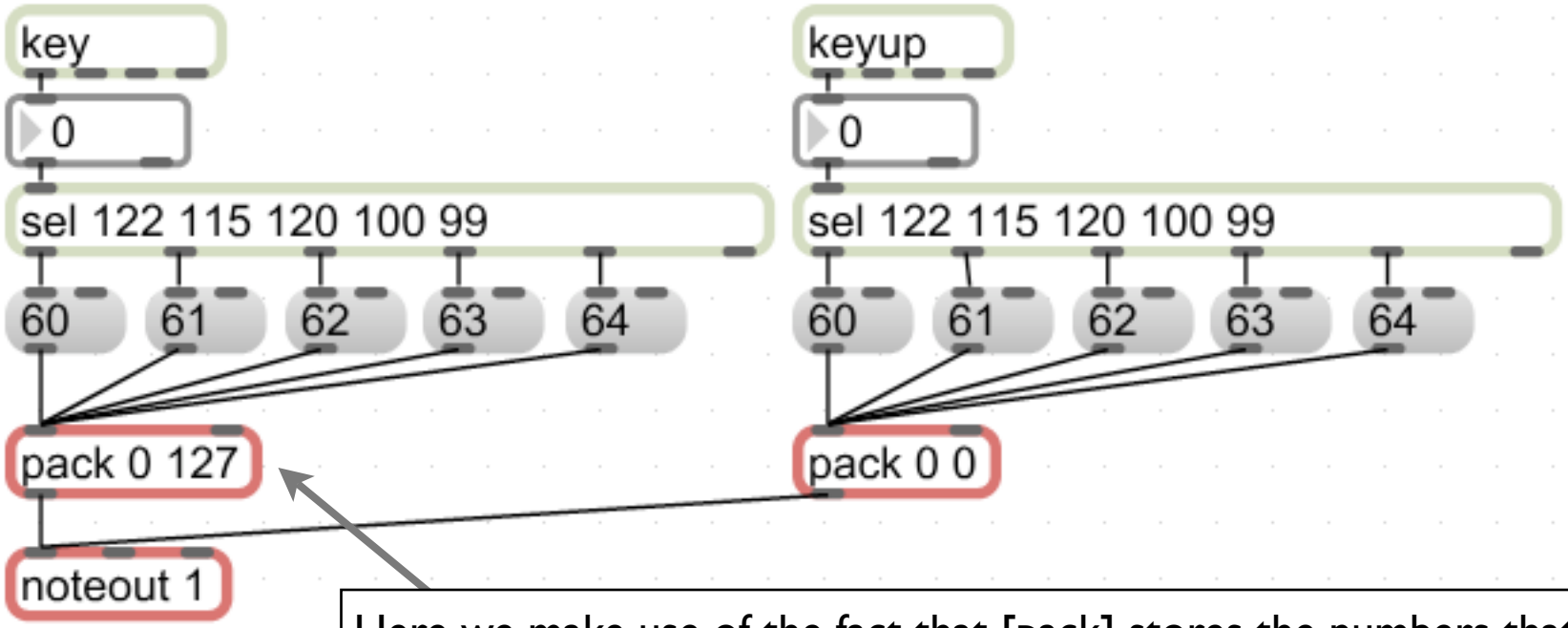
You should have noticed that:

60 127	plays a C-3 at velocity 127
64, 67	plays notes E-3 and G-3 at velocity 127 ([noteout] stores the previous velocity value)
48 64	plays a C-2 at velocity 64 (you may not be able to hear this at the moment)
52 55	plays a E-2 and G-2 at velocity 64 (again storing the previous velocity)
48 0, 52 0, 55 0 etc	turns all notes off (notice that we can send multiple lists from the same [message box] provided we separate them with commas)

Ex.2 (cont)

This all tells us that we need to send 127 velocities with note numbers from [key] and 0 velocities with note numbers from [keyup].

5. We can do just that by adding the objects in red to our patch as follows.

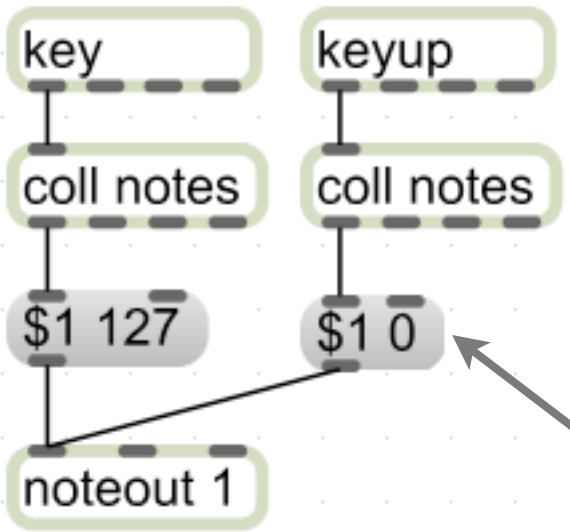


Here we make use of the fact that [pack] stores the numbers that you give it as arguments *unless* they're updated. We don't update the right-hand inlet so it remains 127. So messages will be 60 127 or 61 127 or 62 127 etc.

6. Complete your keyboard by adding the remaining notes.

Ex.3

1. Open the patch 'keyControl2.maxpat' from the MUST1002 website (see schedule page). This patch does exactly the same thing, but *much* more elegantly, using [coll] (which we met last week).



Double-click on [coll notes] to open it. The contents should be fairly self-explanatory. Note the syntax, though:
index, stored-materials;
The comma and semi-colon are *very important*!!!
(alt-click on [coll] for more information on this invaluable object)

This serves as a slightly neater alternative to using [pack]. It does *exactly* the same thing. The $1 is replaced by whatever enters the object from [coll]. Attach this to a [print] object to make sure you understand what's happening.