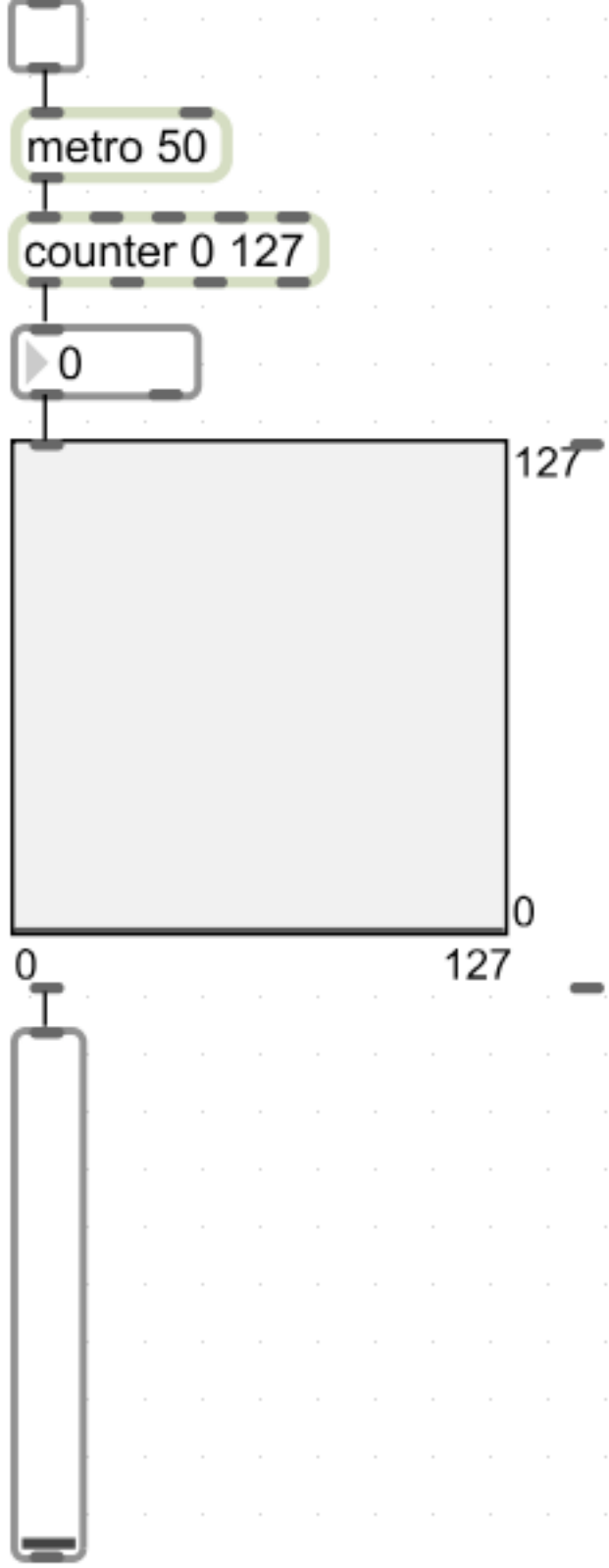
Max/MSP exercises 3c

Ex. 1

We've investigated the [table] and [itable] objects as means of controlling entered note data through time. We can also use them to *record* data through time. For this series of exercises we'll use an [itable] object to store automation data for a [slider] object.

1. Copy the routine on the right. This is basically the same patch as in the previous exercises but this time we are using the [itable] to control a [slider].

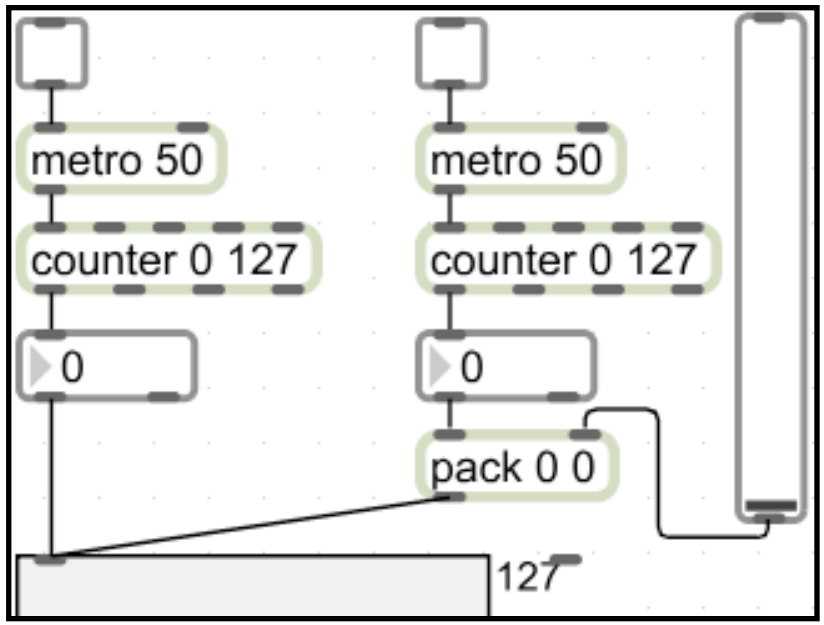
2. Draw some squiggles into the [itable] and start the [metro]. [slider] will now move according to your squiggles.



Ex.2

Thus far we've only been sending one number to [itable] at a time. These numbers run through the indices on the table (the X or horizontal axis), calling the values on the Y or vertical axis. This is pretty obvious from watching the vertical movement of the fader. Thus we *read* through the [itable]. Sending *two* numbers at a time – as a list – tells [itable] that we want to *write* to it.

1. Switch off [metro]. Add to the patch as follows:

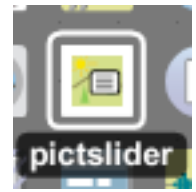


2. Switch on the [metro] for this new part of the patch and move the slider. Your movements will be recorded in the [itable]. You can then switch *off* this 'record' [metro] and switch on the 'playback' metro to replay this recorded automation.

3. You'll probably find that an [itable] size of 127 isn't big enough to store a decent amount of data, so give it a bigger Table Range using the inspector window.

Ex.3

We can use exactly the same model to automate more than one [slider], or an x/y slider (the [pictslider] object). But we'd want to use the same timing control from each to ensure synchronisation. :



1. Using the same model as Ex.2, automate a [pictslider] object
Hint: you will need two [pack] objects and two [itable] objects, but these should both be provided with a count from the same [counter] object.

2. Provide a means for the user to restart the automation playback.
Hint: reset the [counter] for this. Check its inlets for a way to do this.

3. Provide a means to only record data for the duration of the [itable]
Hint: this is a little tricky! But consider that we only want [counter] to count to its maximum number before stopping. If you check the outlets of [counter], there is one that should tell us when the maximum has been reached. You could then use this to stop [metro].

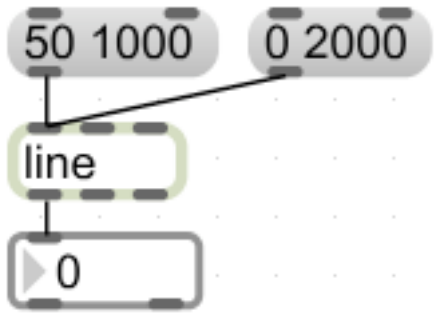
4. Provide a means to speed up or slow down playback of the automation.

Ex.4

You will likely have noticed that with the last exercise (Ex.3.4) as you slow down the [metro] object the movements of your automated [picslider] get rather jerky. Try this now if you haven't already noticed this.

It would be good, perhaps, to smooth out this movement at slower speeds. For this we look at the [line] object. This is an important object to get to grips with as you will use it (or its equivalent, [line~], a lot in MSP).

1. Copy the following:

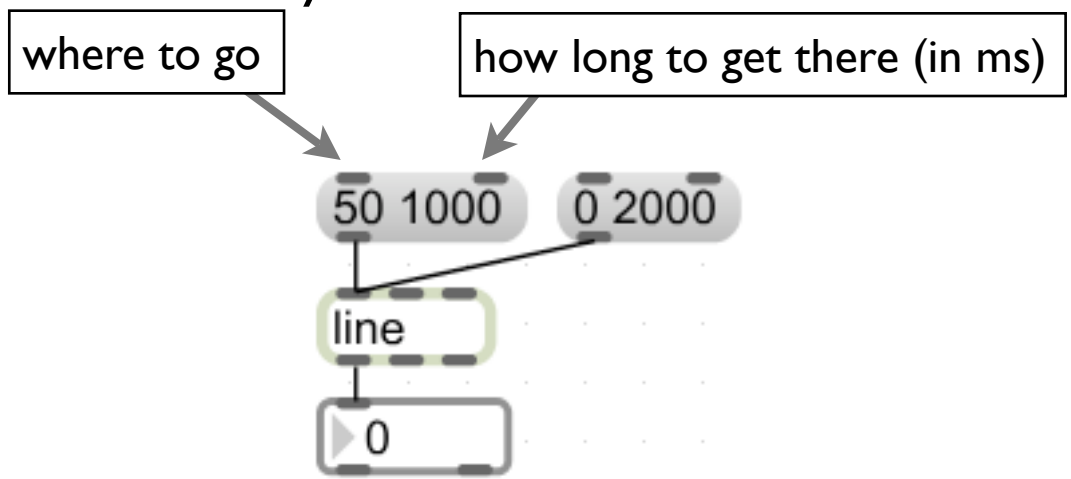


2. Click on the '50 1000' [message box] and note what happens. Then do the same with the '0 2000' box.

3. Click the '50 1000' [message box]. Now click the same box again. What happens? If nothing, why?

Ex.4 (cont)

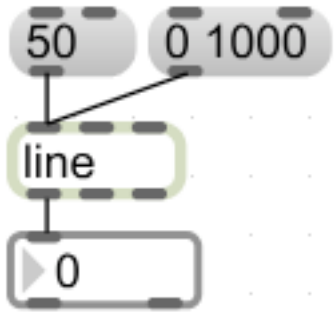
Hopefully you noticed that when you clicked the '50 1000' box once, the bottom [number] box counted from 0 to 50 over 1000 ms.
When you clicked it again, nothing happened.
Clicking the '0 2000' box initiated a count from 50 to 0 over 2000ms. What [line] is doing is counting from *wherever it's currently at* to the first number over the duration of the second number:



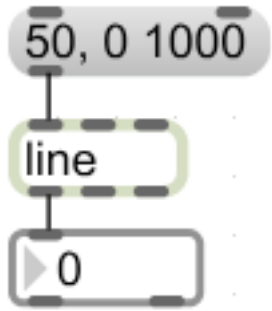
Thus when you hit the '50 1000' the second time, [line] counted from 50 to 50 over 1000ms (i.e. it didn't change).

Ex.4 (cont)

4. Copy the following:



5. Hit the '50' [message box]. You'll have noticed that the [number] box jumps to 50 immediately. What we've effectively done is to send a message of '50 0' (i.e. go to 50 in 0ms). Hitting the '0 1000' box will count you back to 0. We can combine these messages as follows:



Remember that the comma separates the messages, so we jump to 50, then count back to 0 over 1000ms.

6. Try the same with these messages:

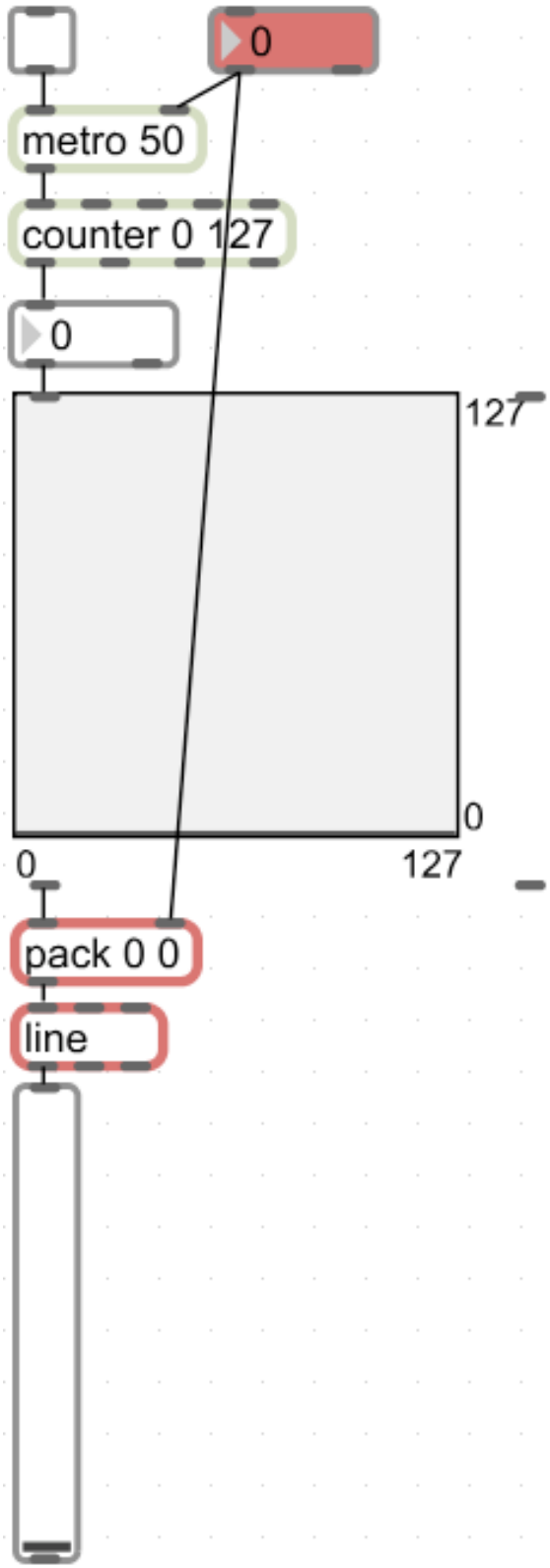
Ex.5

You won't actually need to use the information in Ex.4.4-6 to smooth your automation patch. But it will be useful later on. 4.1-3, however, is relevant here:

1. Copy the routine on the right (the red bits are new).

Here, [pack] is being used to provide the list for [line]. The 'where to go' is provided with each tick of the metro. Note that we use the same [number] box to control the speed of [metro] and the 'time to get there' part of [line]'s input. This makes sense, because we want the interpolation to occur *between* clicks of the [metro].

2. Transfer the relevant bits of this routine to the patch you made in Ex.3 (you will need separate [line] objects for each axis of the [pictslider].



Ex.6

1. Download the patch automation.maxpat from the MUST1002 Schedule page. This does the same thing, rather more efficiently, using [coll].

Read the comments carefully on the patch as they give you more information about the objects and how the patch is structured.