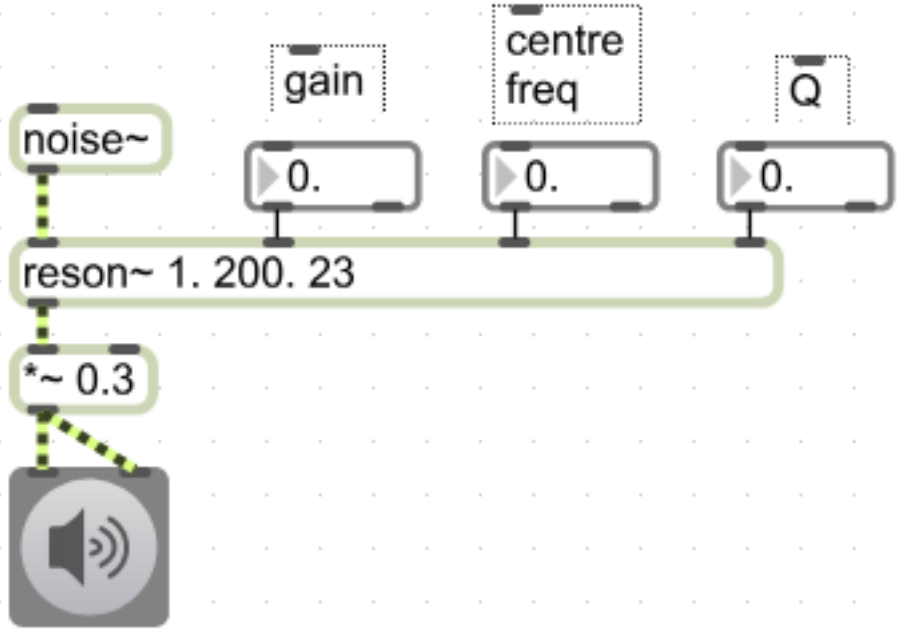
Max/MSP exercises 5b

Ex.1

1. Copy the following:



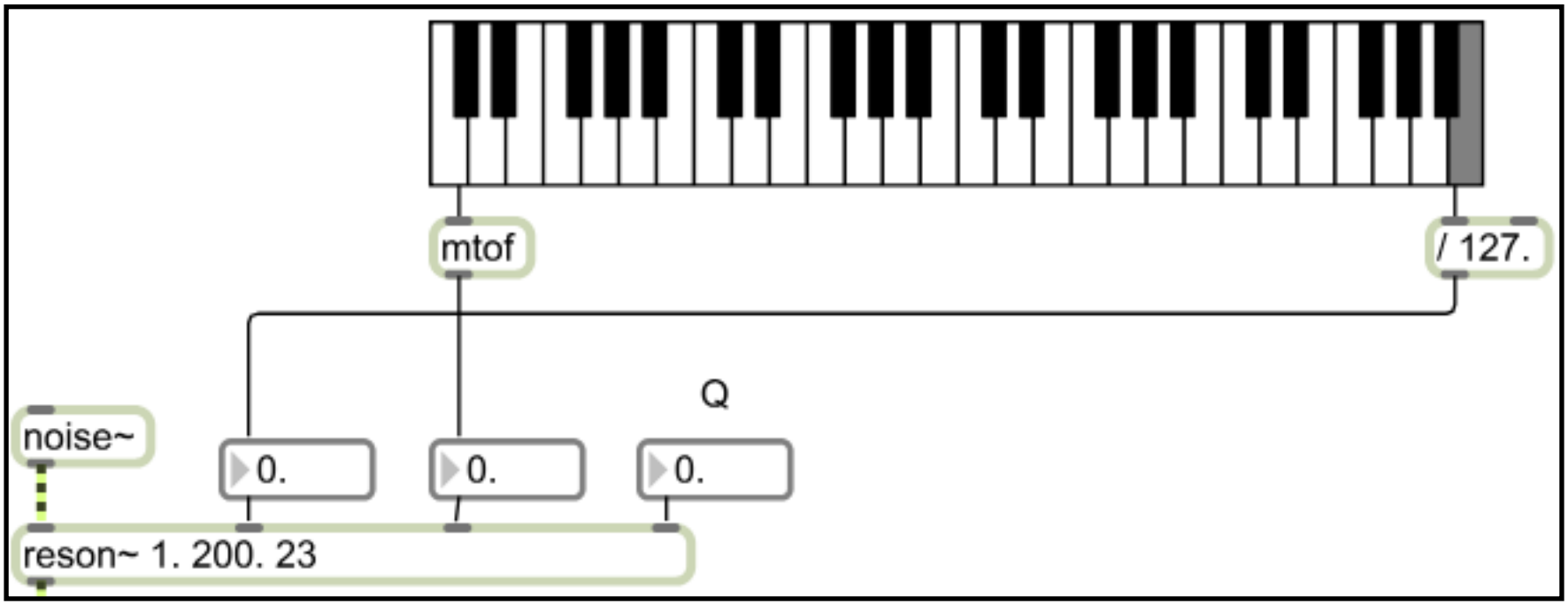
Try it out by manipulating the three [float] inputs.

WARNING: be careful when manipulating the Q if you are wearing headphones as low values can yield loud output.

2. Referring to Exercises5a, Ex.2 how might you make this patch respond to a monophonic keyboard input controlling pitch and level?

Ex. 1 (cont)

Solution:



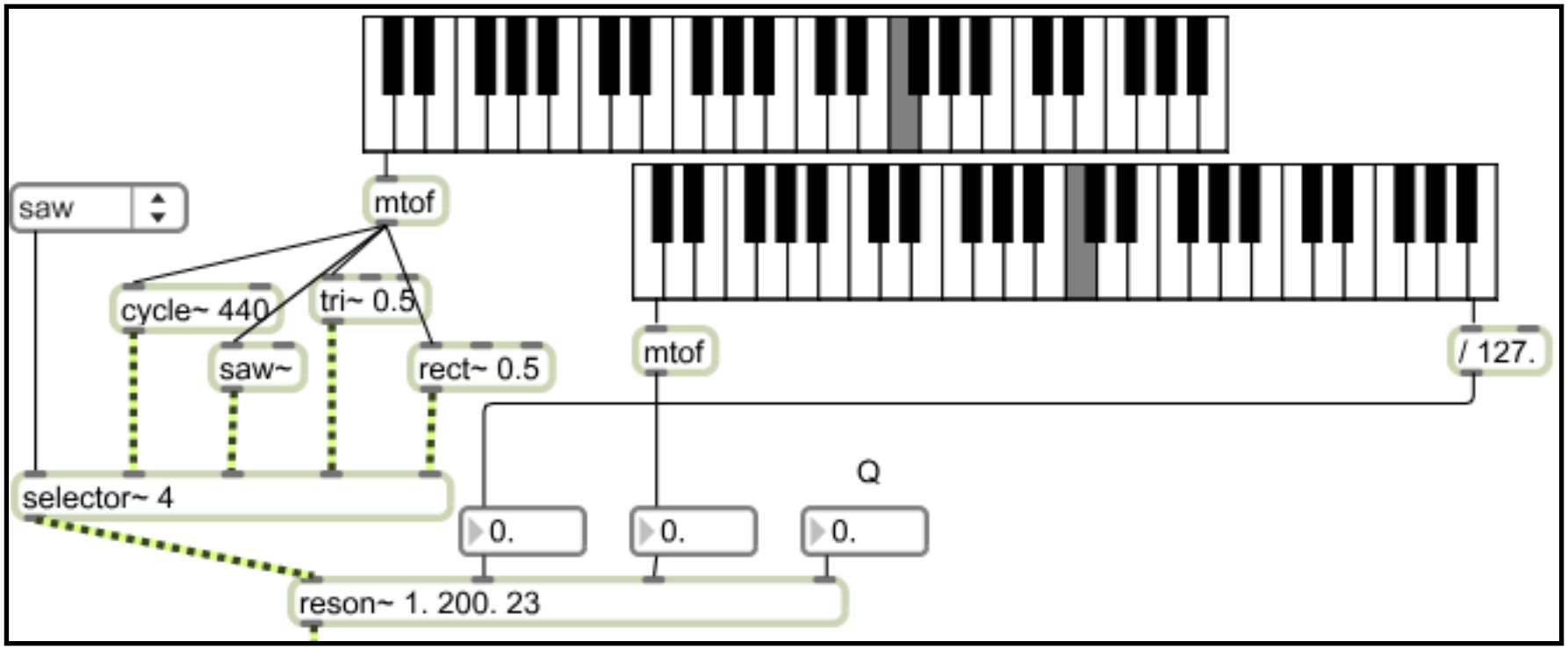
3. You could also map the velocity output to control Q. Try using the [scale] object to do this (you may not want to map the whole Q range).

So we have created a rudimentary monophonic subtractive synthesiser. You will likely want to have it respond to other waveform inputs, however, so...

4. Replace (or supplement) the [noise~] input with other waveform types (see Exercises5a, p10 to remind yourself how you might do this).

Ex. 1 (cont)

Solution:

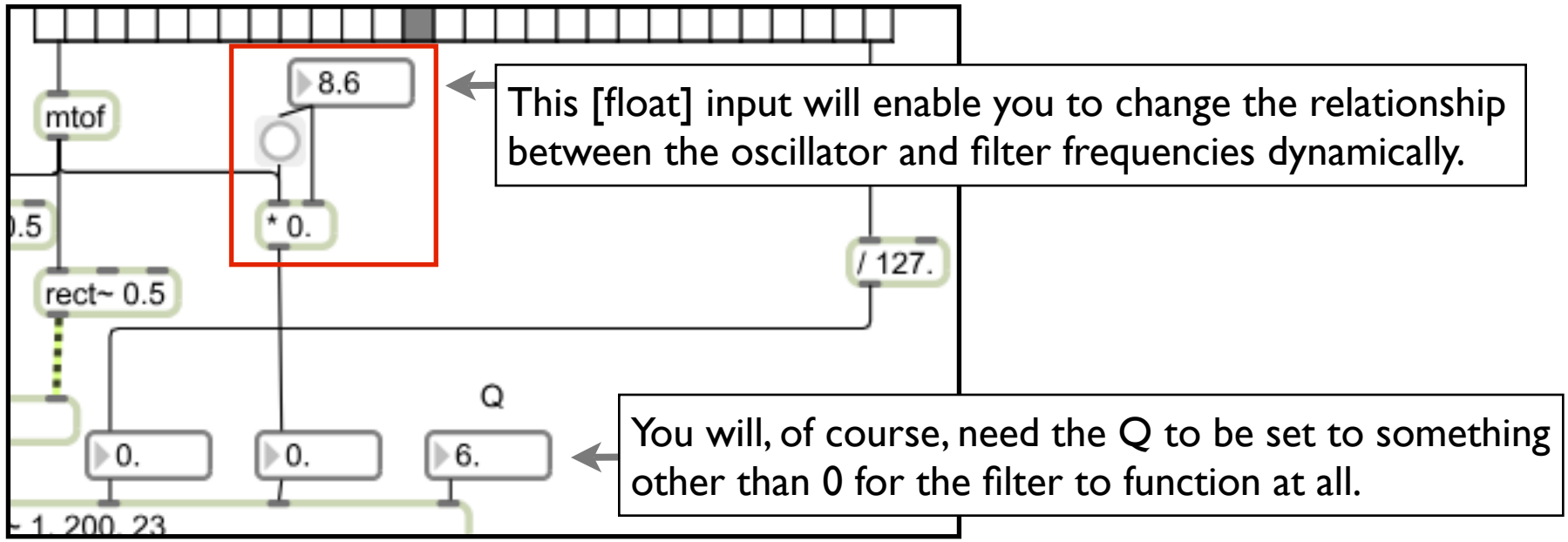


This is not a particularly elegant solution, but it does show that if you choose different notes on the two keyboards independently, you will get different timbres. While this is interesting, you will likely want to ensure a consistent relationship between the pitch of the oscillator and the filter. So...:

5. Consider how you might ensure a consistent relationship between oscillator and filter frequencies. (hint: use one keyboard and a [* 0.] object)

Ex. 1 (cont)

Solution:

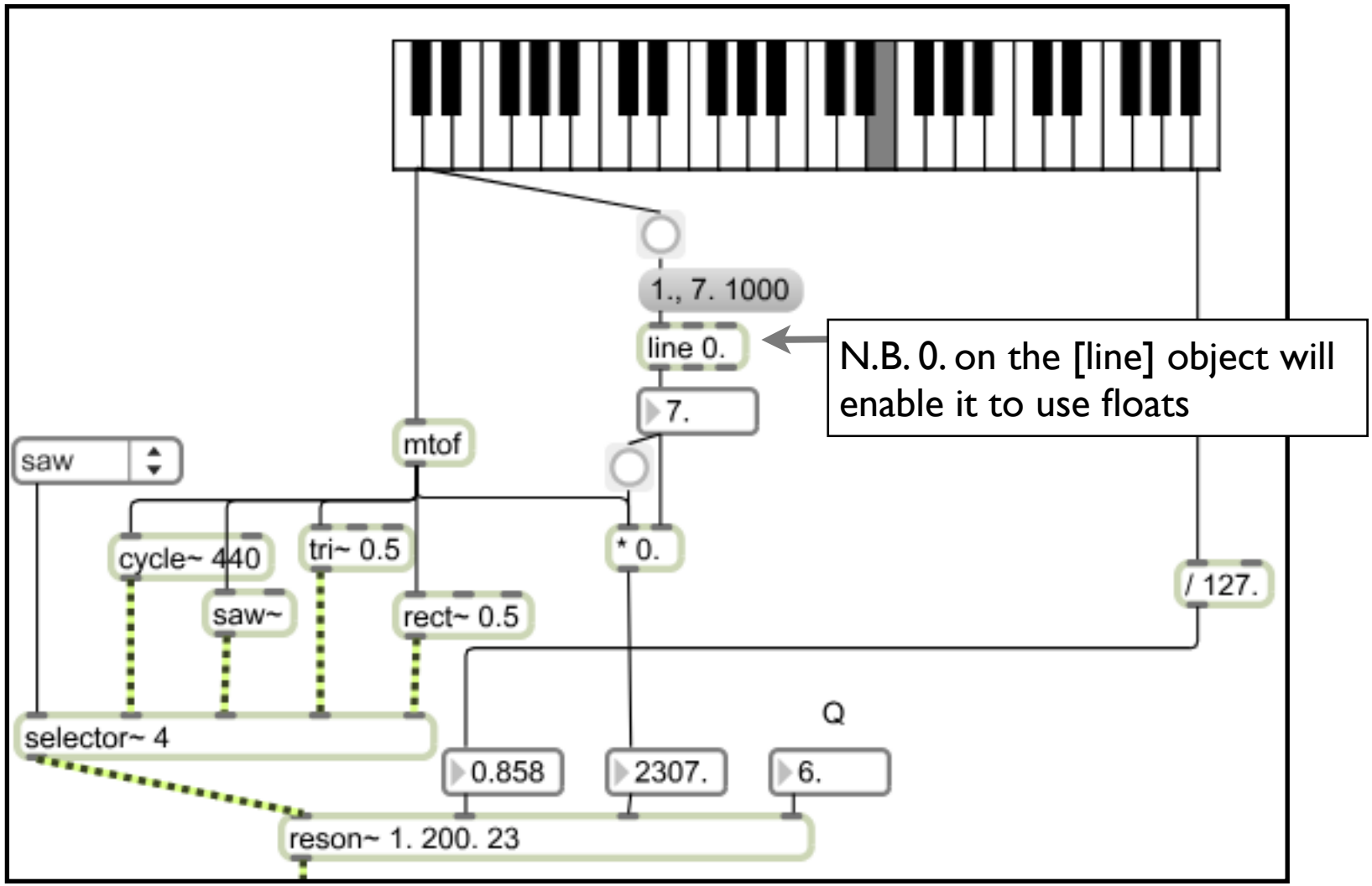


6. Attaching a [line] object to this new part of the patch, how might you make the tone get brighter over the course of a note?

7. If you manage this successfully, try this as **a challenge**: use [function] and [line~] objects to enable you to give a more complicated filter envelope to the sound (you'll need to a) change the limits of the [function] object; and b) replace the [*] object with a [*~] object, sending it directly to [reson~]'s centre frequency input (bypassing the [float] object)).

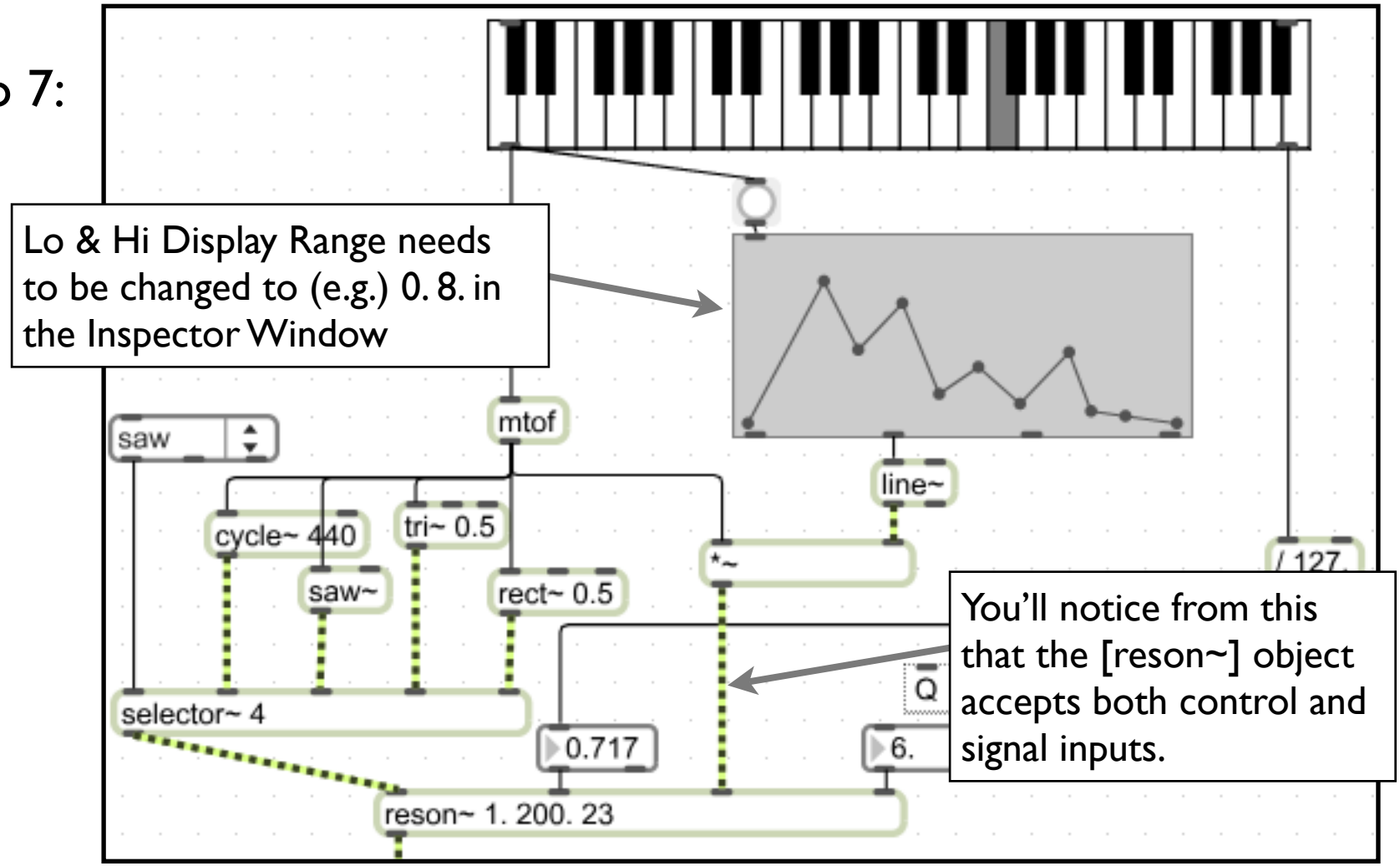
Ex. 1 (cont)

Solution to 6:



Ex. 1 (cont)

Solution to 7:

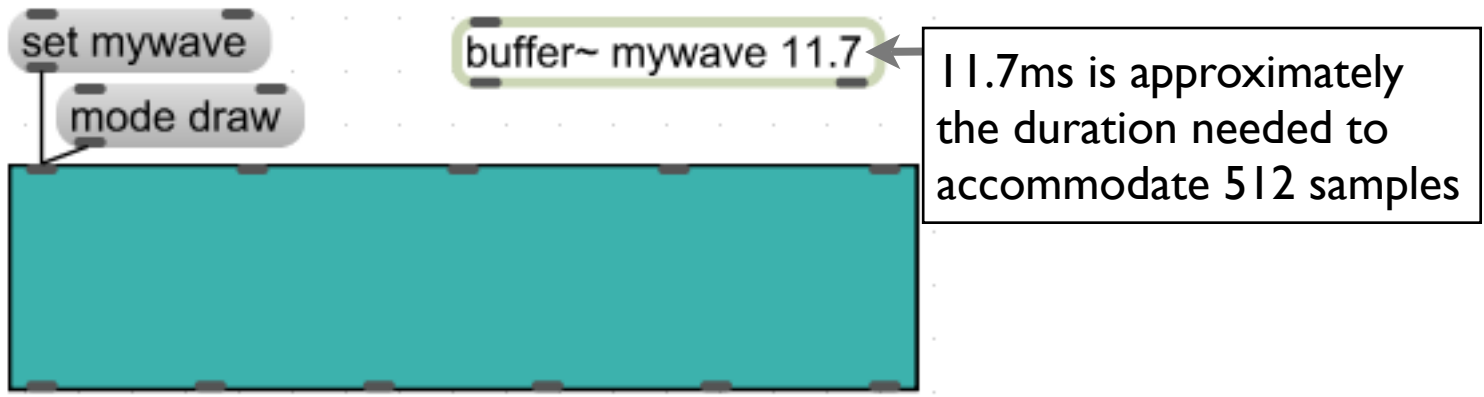


8. Use a [function] object to control the Q of the sound over time, too.

Ex.2

All of the oscillators we've looked at work by reading a wavetable of 512 samples that contain the shape of one cycle of the waveform. It is possible to determine the shape of the wavetable yourself in order to yield some more complex timbres which you can then work with either additively or subtractively. For this we will revisit the [buffer~] object.

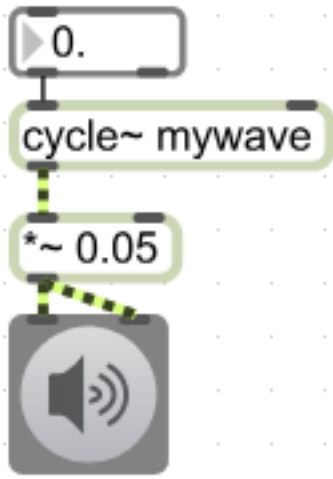
1. Copy the following:



2. Click the 'set mywave' and 'mode draw' [message] boxes. The cursor in the [waveform] object will change to a cross-hairs. You can then draw a shape into the display.

3. Lock the patch and double-click on the [buffer~] object to see another view of your shape.

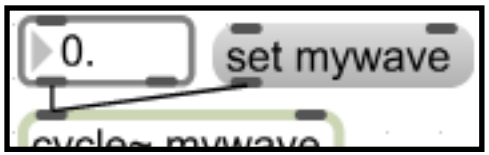
Ex.2 (cont)



4. You can use a [cycle~] object to read through the waveshape by copying the routine on the right (i.e. add this to your patch). As usual, the [float] box here controls the frequency.

5. Running this patch you will hear what your waveshape sounds like.

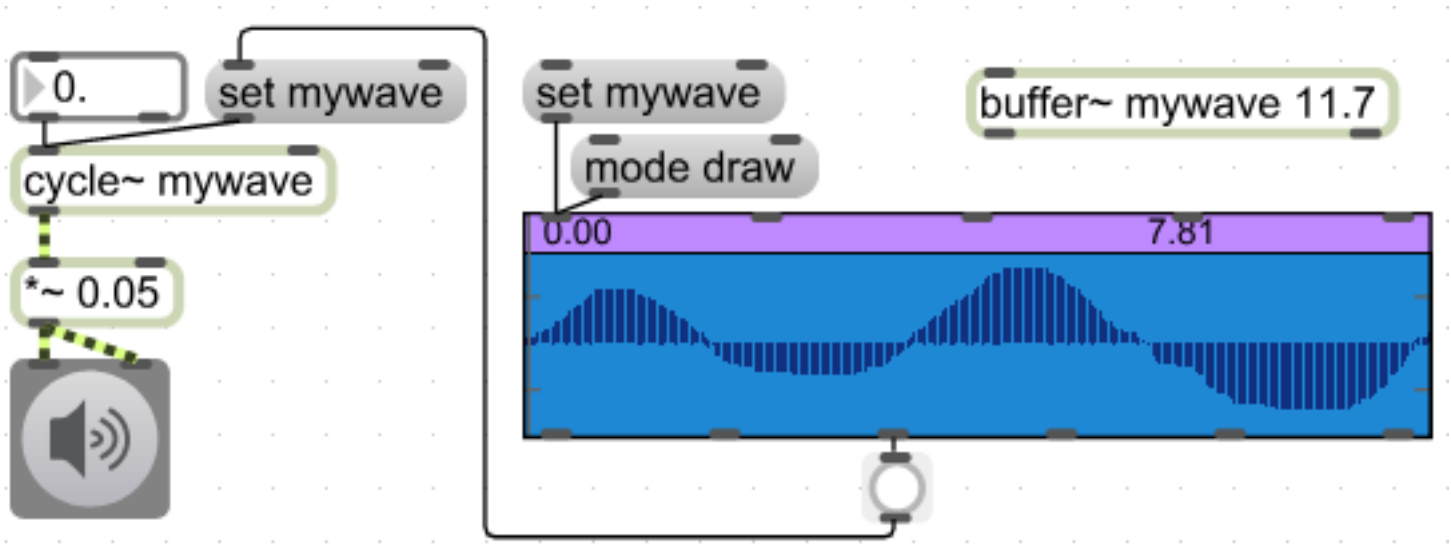
6. Try changing the waveshape in the [waveform~] object. Nothing will happen. This is because [cycle~] needs to update its contents if you want it to refer to the new shape. So use a [message] box to send it a 'set mywave' message.



7. How might you get [cycle~] to update automatically every time you change the waveshape?

Ex.2 (cont)

Solution:

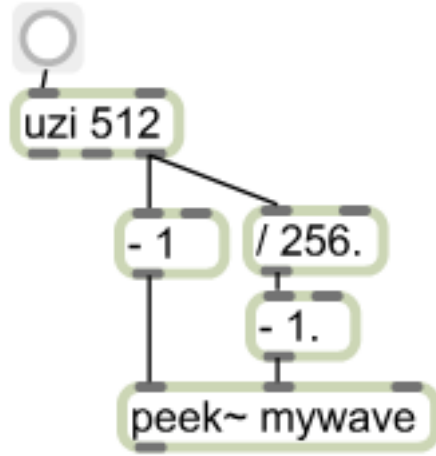


This allows you to change the sound very dynamically by altering the waveshape while the oscillator is playing.

Besides drawing the waveshape manually, we can also get Max to generate one for us.

Ex.2 (cont)

8. Add the following to your patch:



9. Click the [button] object and watch what happens to the [waveform~] display. What is happening here? To find out:

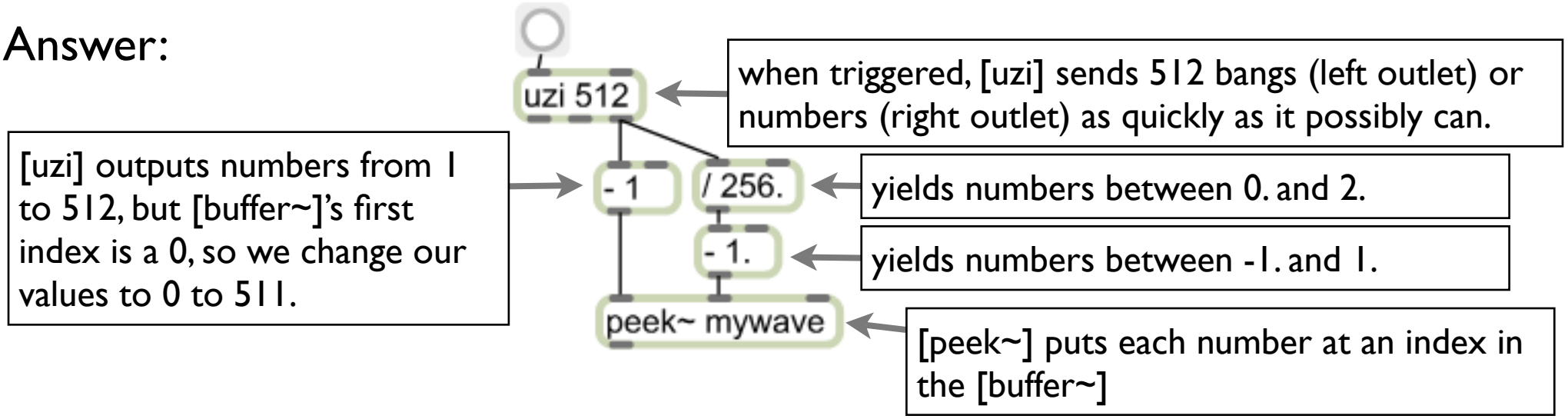
a) hover your cursor over the relevant outlets and see what they are outputting.

b) use the [print] object to do the same

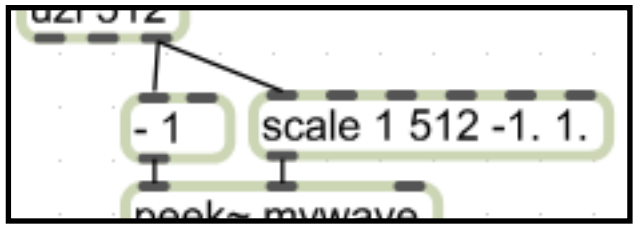
c) use the help files for [uzi] and [peek~]

Ex.2 (cont)

Answer:



We could of course achieve the same with this:



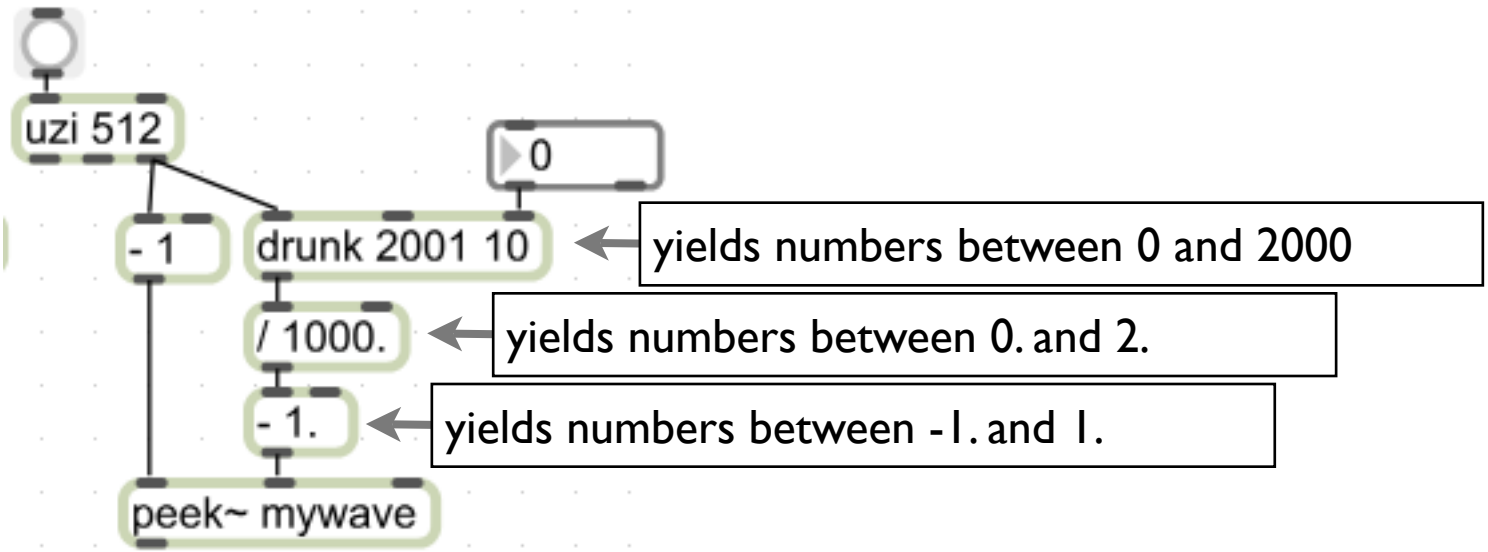
but we're going to need to think a bit about maths (nothing too scary) when we come to the next series of exercises on Amplitude and Frequency Modulation.

In the meantime...

Ex.2 (cont)

We can generate other more random waveshapes by using the [drunk] object.

10. Copy the following:



This routine generates a pseudo-random waveshape based on a drunk-random walk. You can determine the eccentricity of the shape (and how loud it will sound) by increasing the drunk step size.

11. Try triggering this process using a [metro] object to get a rhythmically changing timbre.