Using `line`

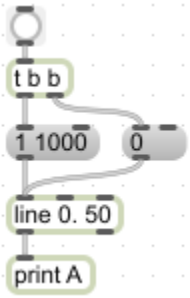
Electronic Music II

Spring 2013

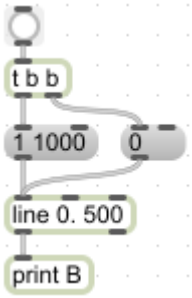
1. The `line` object is used to create sequences of numbers between a starting and an ending value, within a specified amount of time.
   a. The format for a ramp message is [(*ending value*) (*time in ms to reach ending value*)], as shown below.

   b. This will cause `line` to output values between it's current value and 10, such that it reaches 10 in 2000 ms.
   c. It is recommended to always give `line` an argument of "0.", as shown below. This will ensure that it outputs float values.

   d. `line` outputs a value at a fixed rate, determined by the object's grain value. This can be set as the third argument, or sent to the rightmost inlet.
   e. The grain value represents a duration in ms, after which the object will output the next 'step' between its starting and ending values. The default value is 20 ms.
   f. For example, consider these sample outputs of a line object going from 0 to 1 in 1000 ms, using a grain rate of 50 and 500.

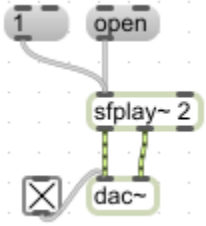


Object	Message
A	0.000000
A	0.050000
A	0.100000
A	0.150000
A	0.200000
A	0.250000
A	0.300000
A	0.350000
A	0.400000
A	0.450000
A	0.500000
A	0.550000
A	0.600000
A	0.650000
A	0.700000
A	0.750000
A	0.800000
A	0.850000
A	0.900000
A	0.950000
A	1.000000

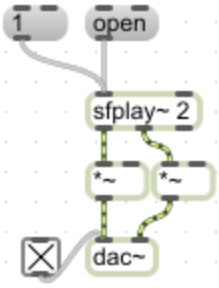


B	0.000000
B	0.500000
B	1.000000

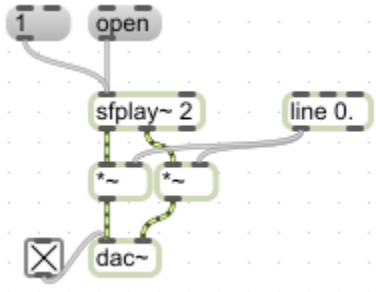
2. We can use `line` to control the amplitude of `sfplay~`'s output.
   a. Start with a basic `sfplay~` setup.



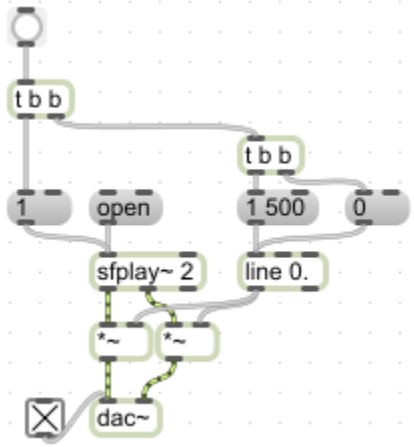
   b. Add a *~ object for each output channel of `sfplay~`.



c. Now create a `line` object and connect its outlet to the right inlet of each *~.



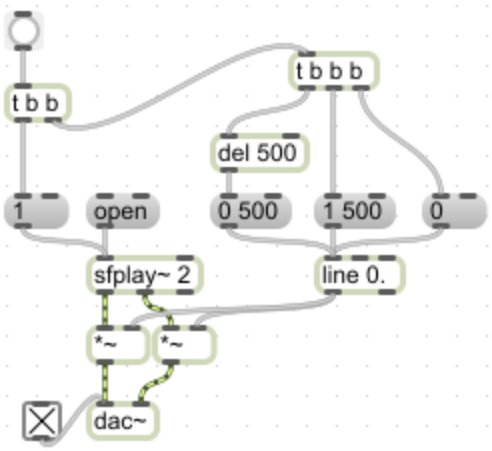
d. To create a simple fade-in, create the following:
   i. a message "0", so that our playback always fades in from 0.
   ii. a message "1 [fade in time]". For this example I will use "1 500".
   iii. two `trigger` objects, each with two bangs, connected as shown below.



e. This will trigger our `line` object to start generating values and begin playback from `sfplay~` at the same time. Notice also, that retriggering this setup will always play the file back from the beginning, with the fade in we have created.

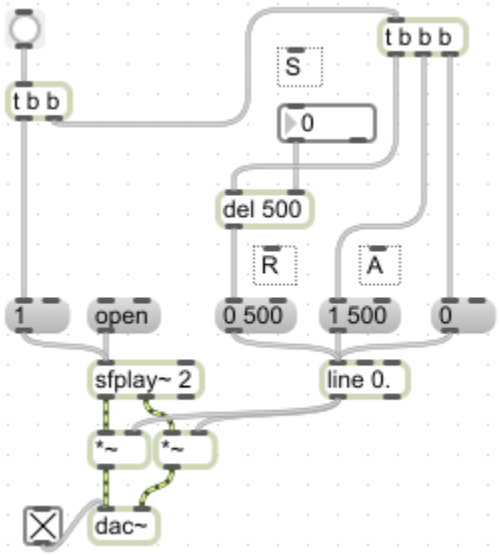
f. To use envelope terminology, the above patch creates an "Attack" for the sound. The opposite concept, the change in amplitude back to 0, is termed the "Release."

g. To add Release to our envelope, we need another message of values, and a `delay` object, as shown below.



h. This will cause the amplitude to move down to 0 in 500 ms, after waiting 500 ms from the time the envelope was triggered.

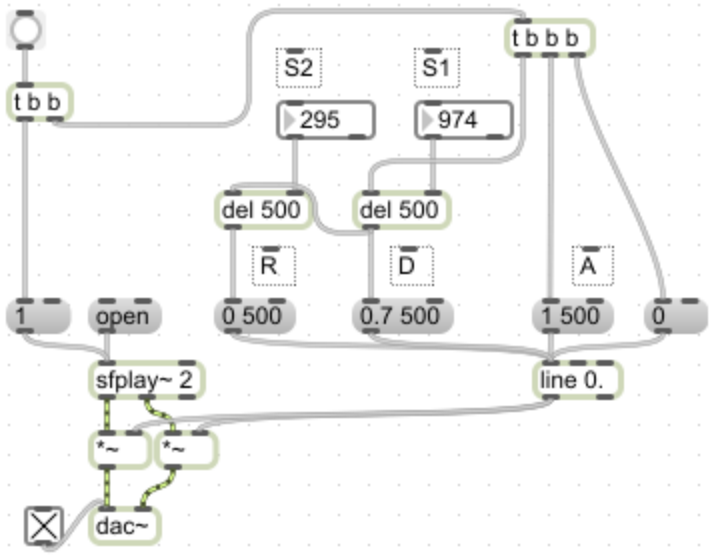
i. The result of this is a simple up-down, triangular shaped envelope. To create a "Sustain" stage, we need to change the amount of delay before the Release is triggered. We can do this by adding an integer box, connected to the right inlet of `delay`.



j. Notice that I've added comments above each stage, labeling what each value represents.

k. This will give us the ability to delay the Release, or alternatively to trigger the Release **before** the Attack ramp has finished.

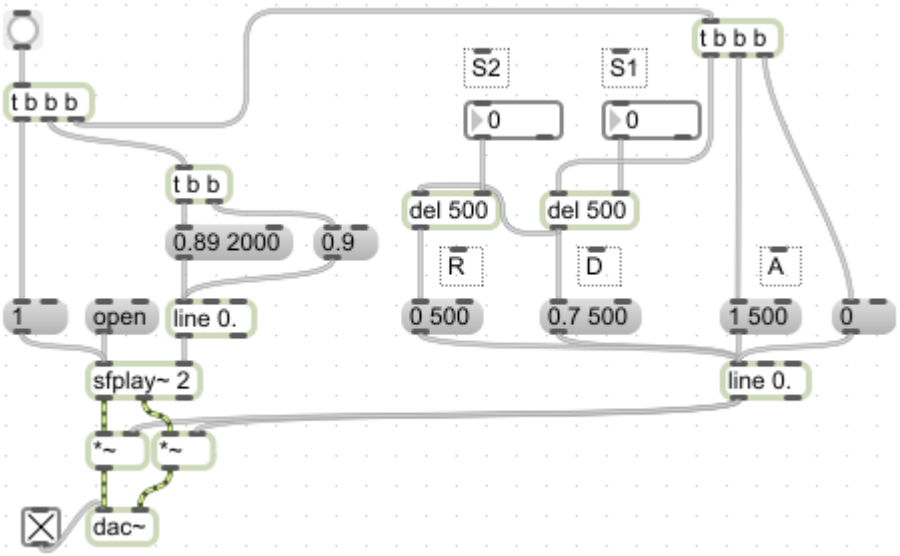
l. To add an internal stage, or "Decay," uses the same components we needed to add Sustain and Release.



m. Notice that the trigger for the Release now comes from the bang that initiates the Decay, instead of from the `trigger`.

n. We can also use `line` to manipulate the playback speed variable of `sfplay~`.

o. Because this will need to operate independently of the amplitude envelope, use a new `line` setup similar to the original Attack setup used above.



p. Also keep in mind that, if `sfplay~` is set to a playback speed of 0, it will output nothing. Any non-zero value will resume playback from the point when it received a 0 playback speed.

q. The fastest `line` can output values is once per millisecond. This will serve most applications well. If some distortion is created, usually when using very short ramp durations, consider using `line~`. It is functionally identical to `line`, except that it outputs a value every sample (compared to every 44 samples with `line`, approximately).