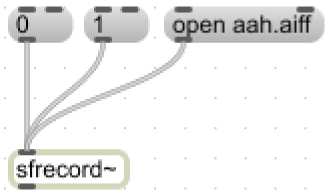
[sfrecord~] object

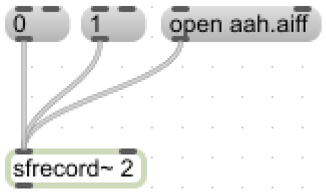
Electronic Music II

Spring 2014

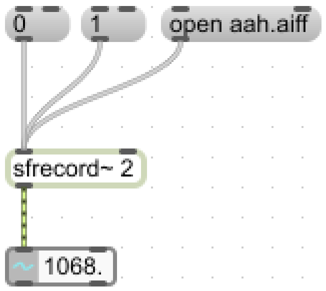
1. [sfrecord~] is used to write audio files to the hard drive. It behaves like [sfplay~] in several ways.
   a. Basic operation of [sfrecord~] requires the following information:
      i. The file to create on the hard drive.
      ii. A message "1" to start.
      iii. A message "0" to stop recording and write the file.
   b. The information must be given in this order. Unlike [sfplay~], the "open" message does not persist; once a recording is stopped, an "open" message must be sent again **before the start message**, even if the file being recorded is the same (resending the same "open" message will overwrite the file that exists).
   c. This basic setup is shown below:



   d. As with [sfplay~], the number of channels can be specified by a creation argument. In this case, creating an object [sfrecord~ 2] will create a 2-channel (that is, stereo) audio file. This is shown below:



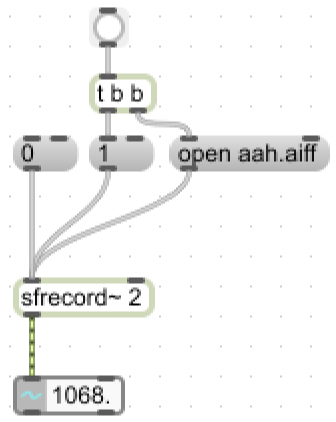
   e. Notice that, despite there now being multiple inlets, control messages are **still sent to the left inlet**.
   f. The outlet of [sfrecord~] is an audio signal equivalent to the duration of the current recording. It can be monitored using the [number~] object, shown below.



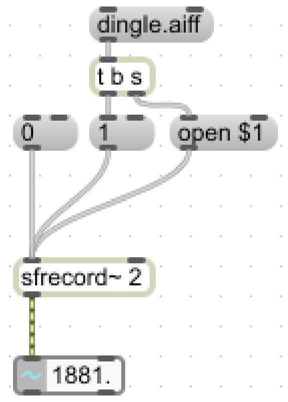
   g. A few notes about [number~]:
      i. Notice that when [sfrecord~] receives an "open" message, [number~] resets to 0.
      ii. The number indicated will increase as long as [sfrecord~] is recording.

iii. As soon as a “0” message is received by [sfrecord~], recording will stop. However, the final duration of the recording won't be reflected in [number~] for about a second.

2. The patch shown above can be improved in several ways.
   a. Message order: a [trigger] object can be added to ensure that the “open” message is always sent before the “1” message. This is shown below.



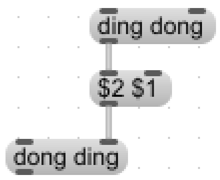
   b. This will start the recording as soon as the bang button is clicked, overwriting the previous “aah.aiff” file.
   c. Currently, to change the filename the “open aah.aiff” message box contents would have to be changed by hand. This can be streamlined somewhat by using a **$ argument**, as shown below.



   d. Notice the differences between the patch shown in item c and the one shown in item a:
      i. The bang button has been replaced by a message box, containing our desired filename.
      ii. The [trigger] arguments have changed from [t b b] to [t b s]. This reflects the new data type that [trigger] is being asked to pass through the right outlet (**s**ymbol, instead of **b**ang).
      iii. The “open” message now no longer contains an explicit filename. Instead it reads “open $1”, since it will be receiving an input from the [trigger] object, which we want to place after “open”. Once the “open $1” message box receives this input in its **left inlet**, it will output a message consisting of “open *(received input)*”. In this case, the output message will be “open dingle.aiff”.
   e. $ arguments are extremely useful in patching, and merit some consideration.
      i. A $ argument is a special character that can be included in a message box, followed by numbers 1-9, to

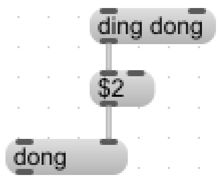
indicate a value in the message box contents that will be replaced by whatever value is passed to the box's left inlet.

ii. They are essentially placeholders, but can also serve a routing function. For example, the following patch:



iii. A message has been created from the input of "ding dong", with the elements of the original reordered (first 2, then 1), resulting in the output message "dong ding".

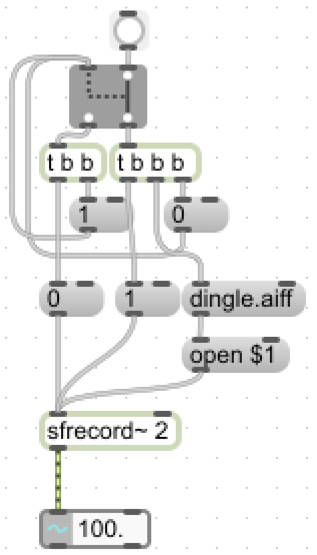
iv. The number following the $ argument is an instruction for the message box, to find element *n* in whatever list is passed to the message box's left inlet. This can be used to extract values from lists, as below:



v. The various functions and behaviors of $ arguments have many applications in patching, all context-dependent. Awareness of them is well-advised.

f. The ability to control the length of the recording is frequently ideal. This can be accomplished in one of two ways: using bangs, or using a "record" message.

g. To control the duration using bangs, presumably connected to some automated process within the patch, a switch like this, using the [ggate] object, is extremely useful:



i. The [ggate] object (automatically substituted for the gray box GUI seen above) has two inlets: the left inlet accepts either 0 or 1, and determines which outlet will pass output. The right inlet accepts any input, and this is passed to either the left or right outlet, based on the value sent to the left inlet most recently.

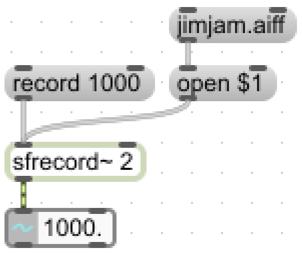
ii. In this patch [trigger] is used to change the switch position before any messages are passed to [sfrecord~].

iii. The procedure for using this particular patch would be to first send a filename to the right inlet of the message box. This will replace the current contents of the message box. Then, when desired, trigger the recording to start; then, also when desired, trigger it to stop.

h. Alternatively, we can specify the desired duration of the recording when we start recording, by sending [sfrecord~] a "record *(duration in ms)*" message.

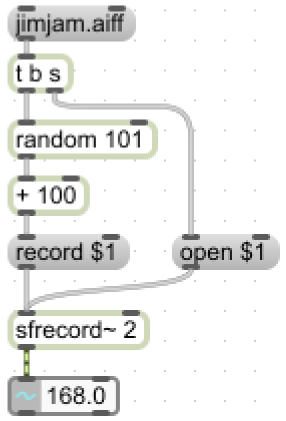
i. This method obviates the need for "1" and "0" messages. Passing [sfrecord~] a message of "record 1000" will trigger it to start recording for 1000 ms; the recording is stopped automatically.

ii. A patch made this way is shown below:



iii. This will record a file "jimjam.aiff" for a duration of exactly 1000 ms (from the time the "record" message is sent).

iv. $ arguments can be implemented here as well. The following patch, for example, records "jimjam.aiff" for a random duration between 100 and 200 ms.


jimjam.aiff
t b s
random 101
+ 100
record $1
open $1
sfrecord~ 2
168.0