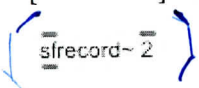
Expanding [sfplay~]: [sfrecord~]

Electronic Music II

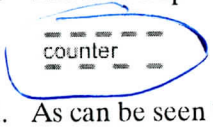
Spring 2014

1. This presentation will show how to record the output of [sfplay~], in a way that is efficient to automate.
   a. The procedures necessary to accomplish are:
      i. Create [sfrecord~]
      ii. Create a filename
      iii. Link recording to [sfplay~]
      iv. Enable/disable recording
   b. Note that again, at the beginning of the project I'm laying out general goals. The realization of the project then is the realization, in specific terms and in a specific context, of the general goals.
2. Create [sfrecord~]
   a. Since we will be recording stereo audio, recall that we need to provide a creation argument of "2" to [sfrecord~]. This object is shown below:

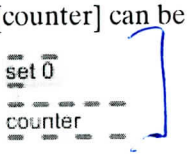


   b. Since the filename will be created by another section of the patch, for now we only will create the "1" and "0" messages to start and stop recording:

3. Create a filename, part 1: [counter] and [date]
   a. A prime concern in creating new files through Max is that the filename is **unique**.
   b. If we open and write a new audio file using a non-unique filename, Max will overwrite the original file.
   c. The first step we can take to avoid this is to use the [counter] object:



   d. As can be seen by the number of inlets and outlets, [counter] has many different functions and options. We will focus on one specific function: When [counter] is created as shown above, **with no creation arguments**, it will count upwards by 1, starting at 0, whenever it receives a **bang** in its **leftmost inlet**.
   e. [counter] can be reset by sending it a "set 0" message, as shown below:



   f. Once [counter] receives this message, the **next value** it outputs will be 0. Receiving this message does not trigger [counter] to output a value.

g. Another step we can take to ensure a unique filename is to use the [date] object.

h. [date] has several different functions, related to measuring elapsed time or accessing system time. For this demonstration we will use its "ticks" function.

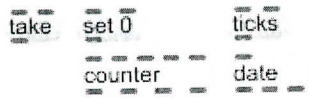
i. The "ticks" function is accessed by sending [date] a message of the word "ticks", as shown below:

j. This function measures how many 60ths of a second have elapsed since the particular computer where the patch is running was booted up. This number is obviously difficult to predict; one benefit of using this number over, say, a [random] object with an extremely high creation argument is that the value of [date] is always increasing, eliminating the chance of reproducing a past value.

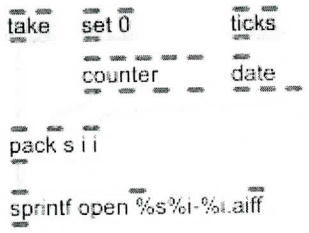
k. A third measure we can take to ensure a unique filename is simply by using a message box, containing some word we provide for that particular file or group of files. For this demonstration, I will use "take".

4. Create a filename, part 2: [trigger], [pack], [sprintf]

a. To summarize, these three components will be combined and formatted into an "open" message, to be sent to [sfrecord~ 2]:



b. The combination and formatting can be as demonstrated earlier for [sfplay~], using a [pack] object and a [sprintf] object, as shown below:



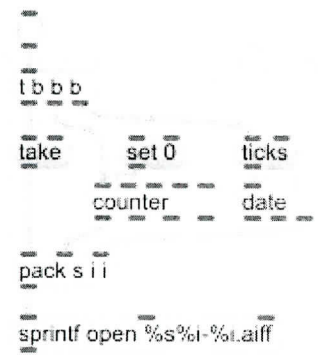
c. Note the following:

i. As with [sfplay~], the order of items in our [pack] list corresponds to the order they're formatted into the [sprintf] message.

ii. The output of [date] when it receives the "ticks" message comes out of the **rightmost** outlet.

iii. The output of [counter] as used in this patch comes out of the **leftmost** outlet.

d. To coordinate these, we will use a [trigger]:

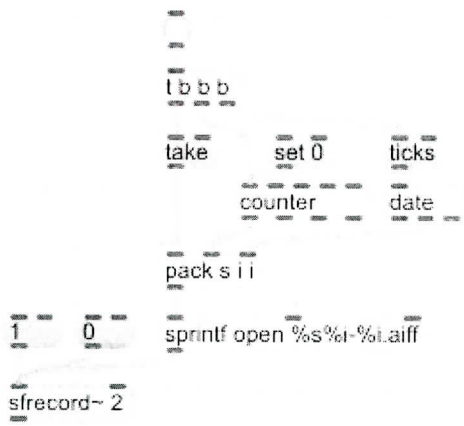


e. Here, notice:

i. [date] is not being sent a bang. Instead the message "ticks" is being banged, causing it to be sent to [date], which causes [date] to output the value we need.

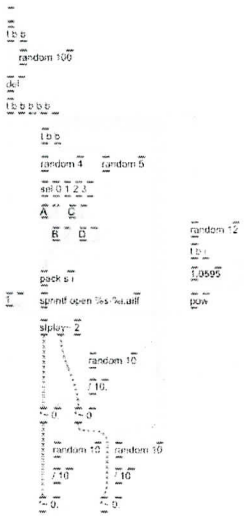
ii. [counter] on the other hand **is** receiving a bang directly.

f. We can now connect [sprintf] to our [sfrecord~ 2] object:



5. Link recording to [sfplay~]

a. For this section, I will simply use the [sfplay~] patch created in an earlier demonstration:

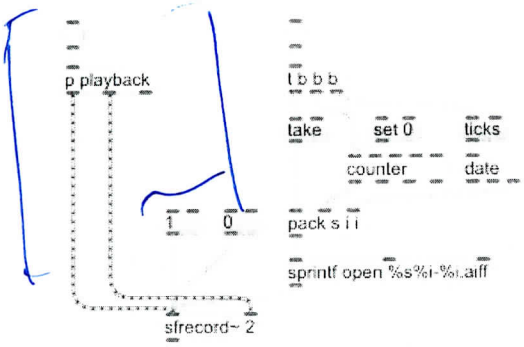


b. For convenience, I will create a subpatch to contain the [sfplay~] section. This will be explored further in a

later presentation; for now it is simply an efficiency measure.

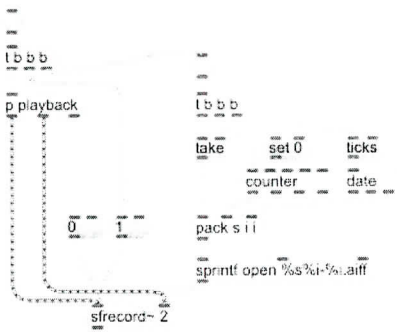
c. The inlet of the subpatch is connected to the initiating trigger object in the original. The three outlets from left to right are: audio output L, audio output R, bang from [sfplay~ 2].

d. First, connect the audio outlets to the inlets of [sfrecord~ 2]:



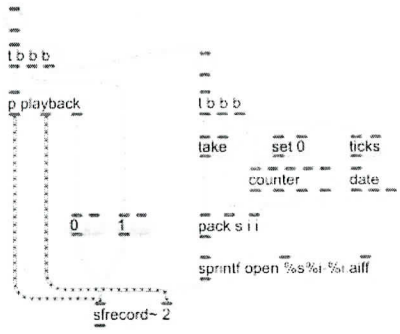
e. This stage has two goals: to start recording before playback, and to end recording after it.

f. To meet the first goal, we can simply use a [trigger] object sending three bangs, as shown:



g. This ensures that, in this order, a unique filename will be created, recording will begin, and then playback will begin.

h. To stop recording, we will utilize the bang being sent by [sfplay~ 2] once playback is finished:



i. This patch satisfies our two goals as they stand. However, consider the order of events that have to be satisfied to successfully record a sound file:

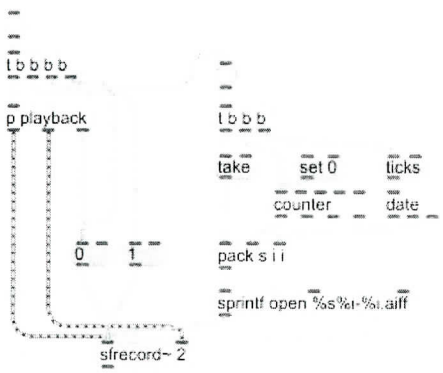
   i. Receive "open" message

   ii. Receive "1" message

   iii. Receive "0" message

j. Only the third step can occur out of this sequence without creating an error. The third step of the sequence is also **required to restart the sequence**: a "0" has to be received before a new "open" message will be understood by [sfrecord~]

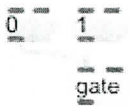
k. A simple solution is to add a bang to our [trigger] object, sending a bang to "0" **before** creating the "open" message:



l. This will stop any current recording **first**, then execute the patch as usual.

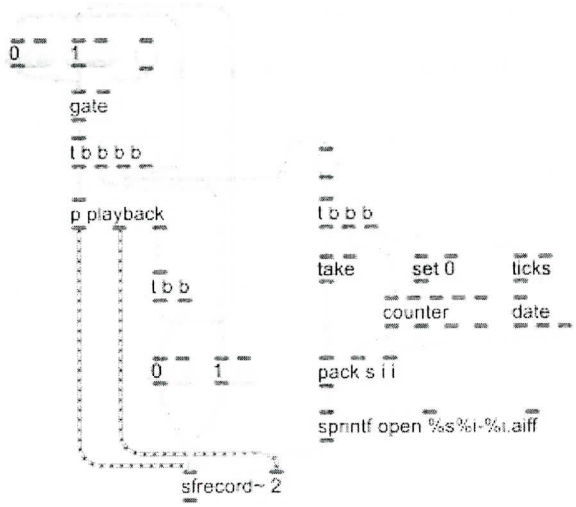
m. Consider that, with this new step, **if** a second bang is sent into the patch **before** playback has ended, the output file will have an abrupt end (a click, more than likely).

n. An alternative option, then, is to use the [gate] object. This object will either open or close its outlet, based on a value sent to its left inlet:



o. A "0" message will close the outlet; a "1" opens it. The outlet will output anything sent to the right inlet, while it is open.

p. We will use this object to block any incoming bangs while playback/recording is underway. When playback/recording has ended, it will allow incoming bangs to pass to the initiating [trigger] object:



q. Notice:

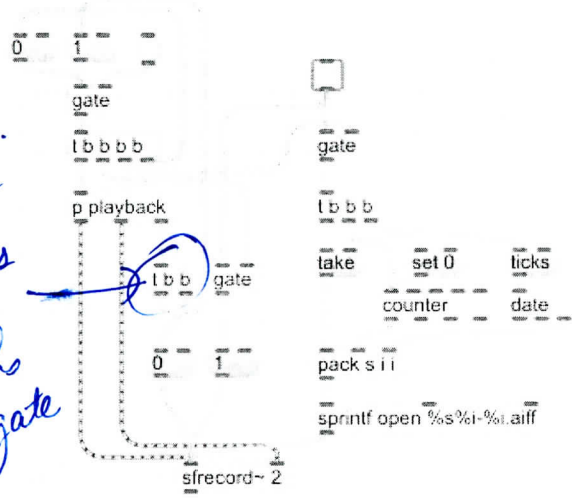
i. The first bang coming from the initiating [trigger] object closes the gate, then the following three bangs execute that patch as usual.

ii. While the [gate] is closed, no bangs will be passed to this [trigger] object, avoiding any sequence miscues/errors.

iii. A [trigger] has been added, such that the bang coming from [sfplay~ 2] now serves two functions: first it stops the recording, and second it opens the [gate].

6. Enable/disable recording

a. If our desire is to audition a particular sound first, without creating undesired audio files, we can insert two [gate] objects and a toggle as shown:



b. Notice that we only need one toggle, but two [gate]s. This is because the "1" being sent to [sfrecord~ 2] is connected to its own bang from the initiating [trigger]; if we only had a [gate] controlling the [pack]/[sprintf] section, we would still get an error when a recording was told to start without first receiving a filename.