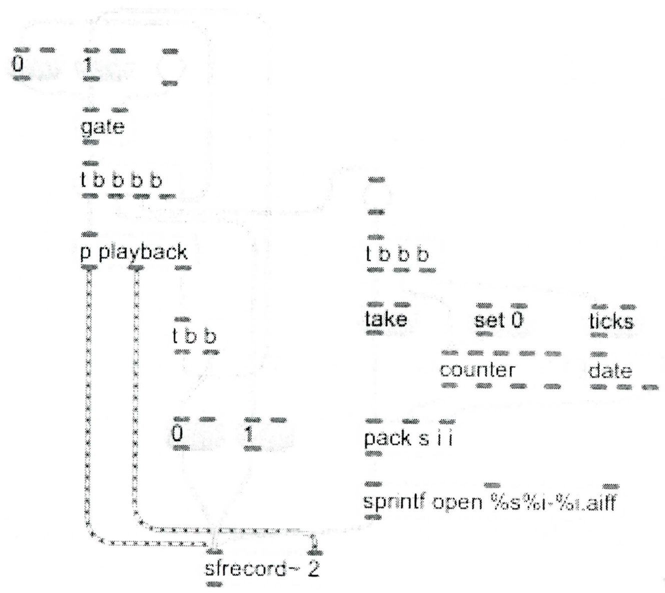
Expanding [sfplay~]: subpatches, [send~]/[receive~]

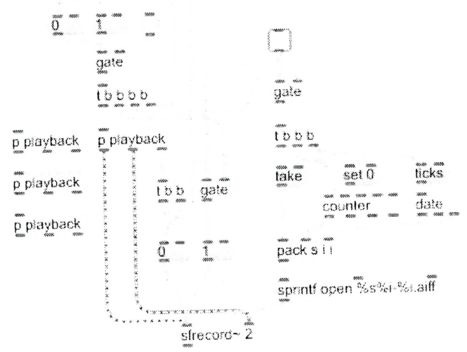
Electronic Music II

Spring 2014

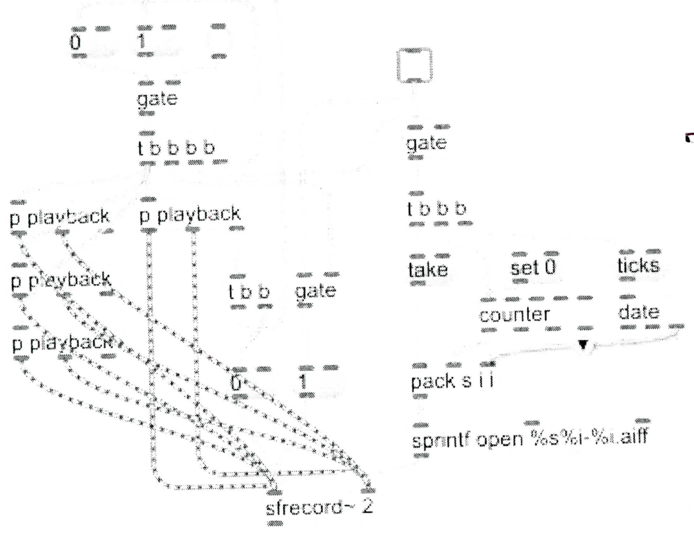
1. This presentation will demonstrate how to expand the functionality of our [sfplay~]/[sfrecord~] patch by using subpatches.
    a. The goals set for the patch in this presentation are:
        i. Play multiple soundfiles, triggered simultaneously.
        ii. Record **the aggregate** of audio output into one file.
        iii. Stop recording at the end of **all playback**.
    b. The solutions demonstrated here will be designed for a specific context. The techniques/concepts involved, however, can find applications in many other contexts.
2. Multiple soundfile playback, part 1: make subpatches
    a. We will begin with the patch created in a previous handout, shown below:



    b. Since we are making **one** recording of **multiple** file playbacks, the only component we need to reproduce is our [p playback] subpatch. We can do this by copying and pasting multiple instances of it:



c. Simply making connections, from the initiating [trigger] to each subpatch, and from each subpatch to [sfrecord~ 2], presents two main problems:



$\rightarrow$ so remote messaging could solve part of the problem?

i. There is no coordination the **end of the last playback instance** and the **end of the recording**.

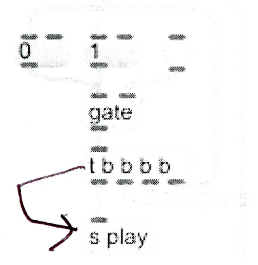
ii. The audio being sent to [sfrecord~ 2] will be summed; any values that exceed 1 or -1 will be recorded as distortion.

d. These two problems can be resolved with [send] and [receive], their audio versions, and a little math.

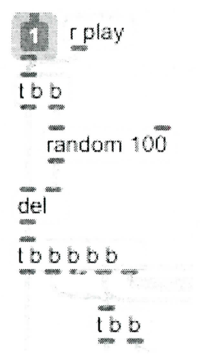
3. Multiple soundfile playback, part 2: [send]/[receive], etc.

a. First, delete the added subpatches. Since the contents of a subpatch have to be modified on a **per subpatch** basis, it is faster in the end to make adjustments to one subpatch and then copy it (thus copying all the adjustments).

b. First, we will add a [send] object (here the identifier being used is "play") where currently a bang is being sent into our subpatch:



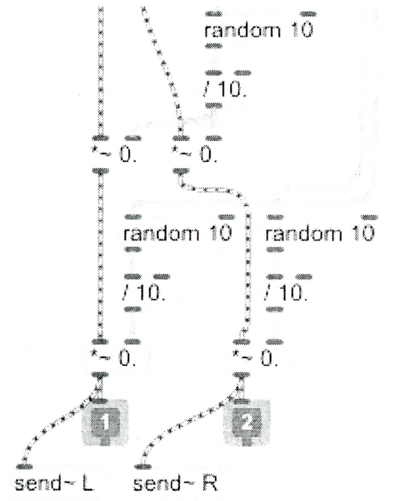
c. Now, in the [p playback] subpatch, add a [receive] object **(with the same identifier)**, where the inlet currently is connected:



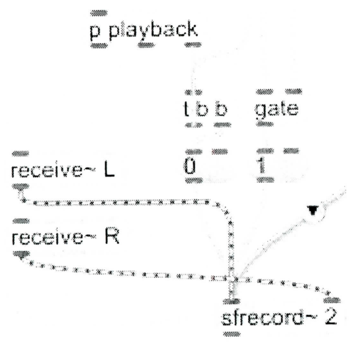
d. Now any input passed to [s play] (a bang) will be sent to [r play] (trigger the playback sequence).

e. [send] and [receive] have equivalent audio versions, [send~] and [receive~]. **NB**, [send] and [receive] can be abbreviated as seen above; [send~] and [receive~] **cannot**.

f. Within the subpatch, create two [send~] objects, naming them "L" and "R". Connect the output of the final stage of [*~] objects to these [send~] objects, as shown below:

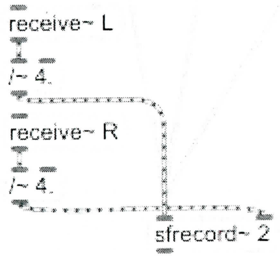


g. Create two [receive~] objects, again named "L" and "R", in the main patch. Connect them to the inlets of [sfrecord~ 2]:



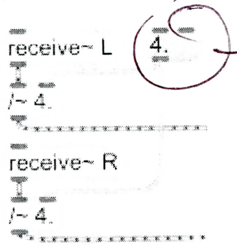
h. This creates a connection between the audio output of any possible copies of the subpatch and [sfrecord~ 2]. However, the possibility of distortion is still present.

i. This can be resolved by using a [/~] object. The creation argument of this object will have to match the number of subpatches in use (for this demonstration we will use 4):



j. Now, all audio sent to the [receive~] objects (remember, this audio has been summed together) will be divided by 4, the maximum possible value (if all four subpatches are sending 1's simultaneously).

k. As we change the number of subpatches being used, this value will have to be changed. For this demonstration, I will use a message box containing the needed value.



want to use the # you know you want + click it.

Part II

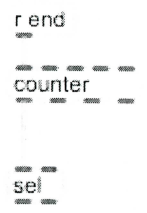
4. Stopping recording

   a. In order to record all of the soundfiles being played back, we need a way to count the number of playback-over bangs that have occurred. We can achieve this with a [counter] and a [sel] object.

   b. In the subpatch, create a [send] object attached to the rightmost outlet of [sfplay~ 2].

   c. In the main patch, create a [receive] object with the same label (in this demonstration I use "end"), and attach it to the leftmost inlet of [counter].

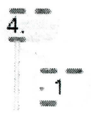
   d. Attach the leftmost outlet to a [sel] object as shown below. Since we will be varying what [sel] is 'looking for', here I leave it blank.



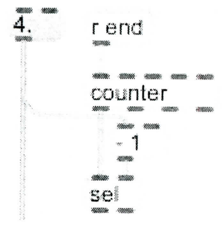
e. Now we can use the message box that is setting the value of [/~] to also set the value [sel] will respond to.

f. Recall that [counter] starts counting from 0. So, by the time the [counter] reaches 3, it will have received 4 bangs (0-1-2-3).

g. With that in mind, create a subtraction object with a creation argument of 1, and connect the message box to the left inlet:

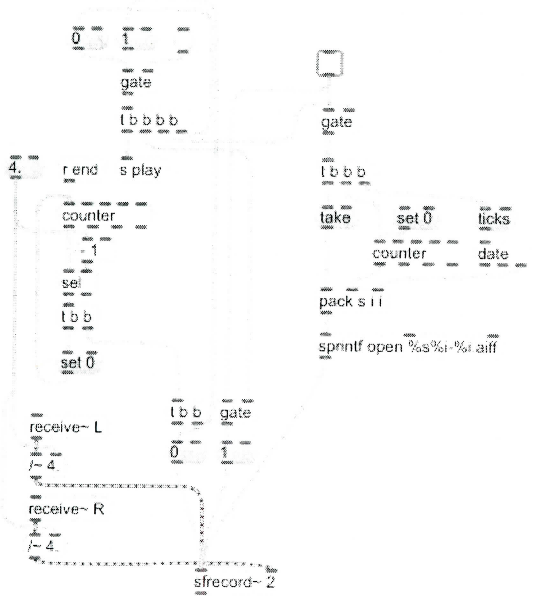


h. Connect the outlet of [- 1] to the right inlet of [sel]:



i. The left outlet of [sel] will now output a bang once the counter has received 4 bangs. To rephrase this, once all 4 subpatches have finished playing back their files, the [sel] object will output a bang. This makes it parallel to the bang coming from the single [sfplay~], in the original [sfrecord~] patch.

j. Since we are using a [counter], and [sel] is looking for one specific value, we will need to reset the [counter] using a "set 0" message. It must be reset when [sel] outputs its bang, so we will use a [trigger]:


0
1
gate
t b b b b
4.
r end
s play
counter
- 1
sel
t b b
set 0
gate
t b b b
take
set 0
ticks
counter
date
pack s i i
sprintf open %s%i-%i.aiff
receive~ L
/~ 4.
receive~ R
/~ 4.
t b b
gate
0
1
sfrecord~ 2