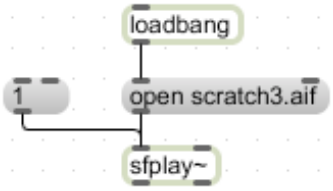
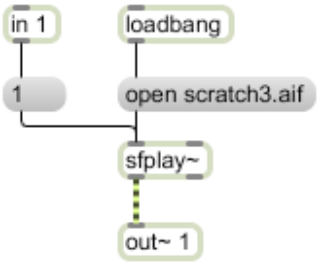
Composition: Electronic Media II
Spring 2012
poly~ and gizmo~

1. poly~ is used to create multiple instances of the same sound. It is useful in order to avoid clipping caused by multiple triggers of the same sound being cut off before the sample is finished playing. poly~ functions in the following ways:
   a. It is saved as a separate max patch, but recalled by Max when triggered.
   b. The number of instances is decided by the programmer.
   c. It functions much like a subpatcher in Max, but is saved as a separate file.
   d. Can help to clean up your patch by consolidating items into a single sub file.

2. To begin, start with a new Max patch.
   a. Open a new file and begin by creating an patch to play a single sound file using sfplay~.
   b. Do not include the dac~ or volume devices. Only include the items needed to trigger a sound file.
   c. Open a sound source in the patch that you already have decided to use. Use loadbang to automatically trigger the sound. Make sure your sound source is in the same folder as the rest of your patch.
   d. Your patch should look like the following



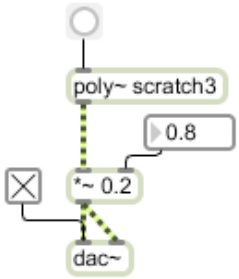
3. Now, you must include a way to trigger these events from an outside source by adding inlets and outlets.
   a. Add a single inlet for the message box "1" that will trigger this sound. This must be called "in 1" with the argument 1 in order for information to come from an external patch.
   b. Add a single outlet for the signal coming out of sfplay~. This must be called "out~ 1" with the argument 1 in order for signal to pass from the sfplay~ out to an external patch.
   c. This looks like the following:



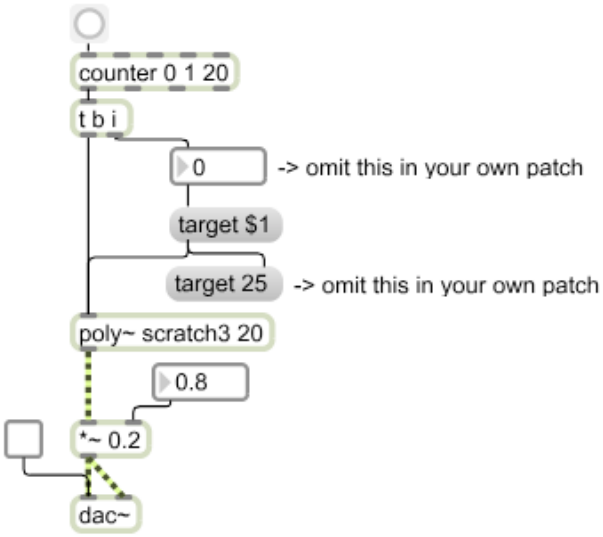
3. Save Item 3 in the folder where your main patch is being stored.
   a. Name the patch the name of the sound file you are triggering. This makes it easier to locate if you are in need of making adjustments.
   b. Make sure not to include spaces in the name of your file. This will cause errors later when creating the poly~ object.

4. Once your file is saved, you can close this patch and return to your main patch and begin building the poly~ object.
   a. Open your main patch and create an object called "poly~ whateveryourpatchnameis".
   b. Notice there is one inlet and one outlet. These are the inlets and outlets created in Item 3.
   c. Build the rest of your patch that you would normally use for a subpatcher that triggers a sound file that includes a dac~, *~, a button, etc.

d. Your patch should look like the following:

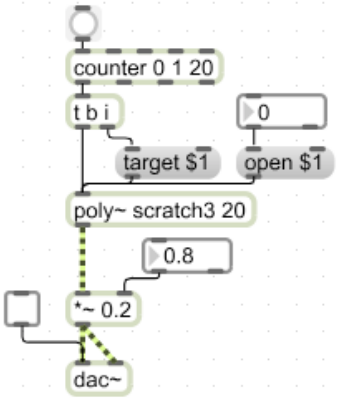


5. An argument is now placed at the end of your poly~ object to determine how many instances of the sound file can be played without clipping.
   a. Make a space at the end of the poly~ object and type in a number you would like to use. Typically 20 instance is plenty and will not cause problems.
   b. You must tell poly~ object which instance of the patch you would like to open. This is achieved by using a counter and targeting your instances.
   c. Create a counter that counts ascending from 1 to whatever number of instances you have chosen for your patch.
   d. Use a "t b i" to receive the integer coming out of the outlet and to create a bang once it is triggered by the counter.
   e. Create a message box that is called "trigger $1". The function of $1 is to be replaced by whatever interger is coming into the message box.
   f. Use a number box and message box to see what is being output from the patch if you would like.
   g. Connect the "target $1" message to the inlet and the most left outlet of "t b i" to the inlet of your poly~ object.
   h. This looks like the following:



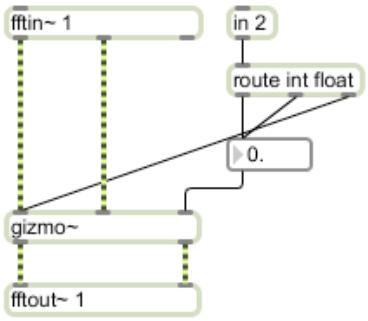
6. If you want to open a specific instance of your patch, create an open message.
   a. Just like the "target $1" message sent to poly~ to tell it what you which instance to open, you can open a specific instance by adding a message called "open $1" with a number box connected to the message box.
   b. Once you choose the number you would like to open, click on your "open $1" message box and that instance will open.
   c. Notice that you cannot edit the patch that opens. This is because it is just the single instance and in order to edit anything in your poly~ object, you must go to the source patch to edit and save any changes you want to make.

d. Your patch should now look like the following:



7. Now, every time you hit the button above the counter, you will be able to be able to simultaneously trigger the same sound source without the risk clipping.

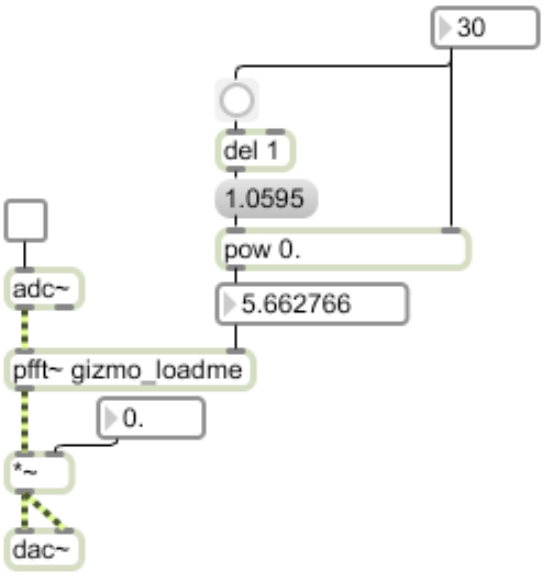
8. gizmo~ uses fast Fourier Transformation is an algorithm which is used to analyze the frequencies being input to Max.
   a. The short hand for fast Fourier Transformation is fft, which is found in a few different objects.
   b. In order to use gizmo~, we must create an external Max patch similar to the poly~ object.
   c. In a new patcher window, create the following objects:
      A. "fftin~ 1"
      B. "fftout~ 1"
      C. "in 2"
      D. "route int float"
      E. "gizmo~"
      F. float box
   d. Connect these in the following way:



   e. Save this patch as "gizmo" in the same folder as your main patch.

9. In a new window, create a patch that receives and outputs live sound using adc~ and dac~.
   a. Include an object called "pfft~ gizmo_loadme". This is similar to the poly~ object because it will function as an external patch that your main patch will call up.
   b. Also include the twelfth root of two semitone algorithm used in other patches.

c. Connect your patch in the following way:



10. Now if you turn on the adc~ and turn up the volume, you can hear your voice coming through Max. If you begin to change the semitone on the right of the patch, your voice is now transposed to that semitone in real time.

11. A note about gizmo~: this object does not come automatically in the online download version of Max. You can find the download in the library of the following URL:
    < http://www.maxobjects.com>
    a. Two files will be download.
       A. gizmo~.mxo will go in the following folder:
       Applications>Max5 (or 6)>Cycling '74>msp-externals
       B. gizmo~.maxhelp will go in the following folder:
       Applications>Max5 (or 6)>Cycling '74>msp-help

12. Other useful information about gizmo~ can be found at the following URL:
    < http://www.cycling74.com/docs/max5/refpages/msp-ref/gizmo~.html>