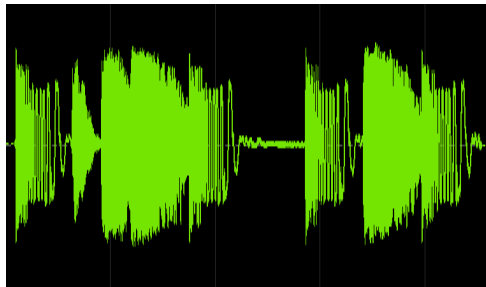
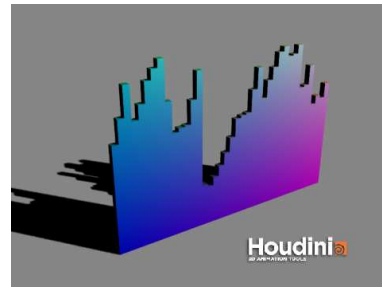


Workshop 04: Audio-driven Animation



We use audio data to control 3D elements



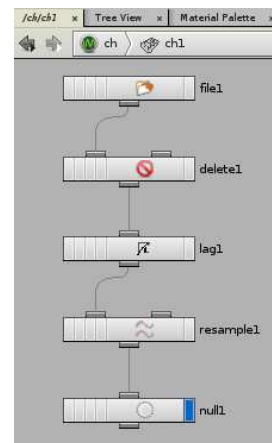
In this workshop we will look at several examples on how to create audio-driven animations.

Audio-driven animation

We will use the volume and frequency of an audio clip to control the 3D elements. It contains the following workflow:

Step 1) Under `/ch`, we construct a **CHOP Network**. A CHOP network is used to manipulate *channel*. The “channel” in Houdini means “value changes over time”. Audio data is also treated as *channel* in Houdini.

We can also use this CHOP network to perform simple audio processing, such as adjusting the volume and duration, shift the starting time, or perform a pitch analysis.



Step 2) Whenever you want to grab the audio data to control a parameter, you can use the expression

chopcf(a, b, c)

where

a is the name of OP in CHOP;

b is the channel number (each CHOP OP can contains more than one channel);

c is the frame number you want to grab.

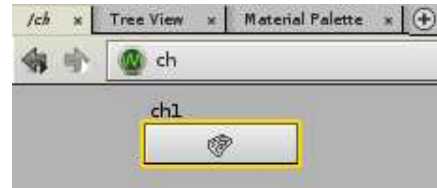
For example:



Volume-driven animation

Step 1) First, you need an audio-clip. Houdini supports AIFF, WAV and MP3 format. In the example folder I have put a file **sound1.wav**.

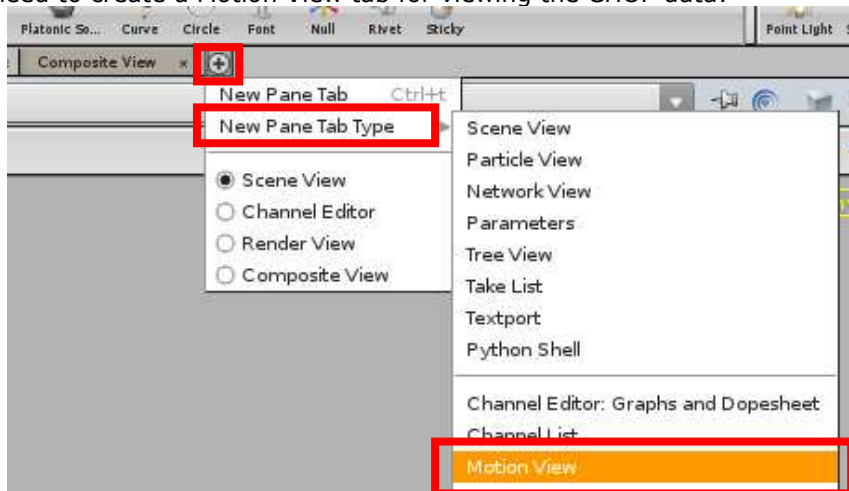
You should go to the **/ch** category, and create a (empty) CHOP Network. By default, the newly created CHOP network is called **/ch/ch1**.



Go into the **/ch/ch1**, press TAB and create a **FILE OP**. Locate your audio file, and turn on the blue-flag:



You also need to create a Motion View tab for viewing the CHOP data:



Note that this FILE OP is called **/ch/ch1/file1** if you haven't renamed it. You will need this name in the later steps. Moreover, middle-click on the FILE OP, you may find that there are two channels inside this OP (it is because my audio file is stereo). They are called **"chan0"** and **"chan1"**. The "chan0" is the channel number 0, and "chan1" is the channel number 1. You also need the channel number later.

```

Full Name:    /ch/ch1/file1
Operator type: file
-----
Channels: 2
  Start/End: 0 to 81414i, 1 to 45.31f, 0 to 1.85s
  Length: 81415i, 44.31f, 1.85s
  Sample Rate: 44100 Hz
  Min/Max: -1 to 0.99
Memory Usage: 637 Kb
-----
chan0 H,H    -0.956 to    0.958 ( 0.00458)
chan1 H,H     -1 to    0.994 ( 0.00269)
-----
Time Dependent: No
    
```

Step 2) Now, go back to **/obj**. Create a GEOMETRY and go into it. In this simple example let's use a SPHERE. Under whatever parameter you like (say, the y-center of the sphere), type the following expression:

```
chopcf( "/ch/ch1/file1", 0, $F )
```



which means we want to use the audio data (i.e. the volume of the channel in this case) at the current frame number to control the y-center of the sphere.

If you want to hear the audio at the same time, you should do the following:

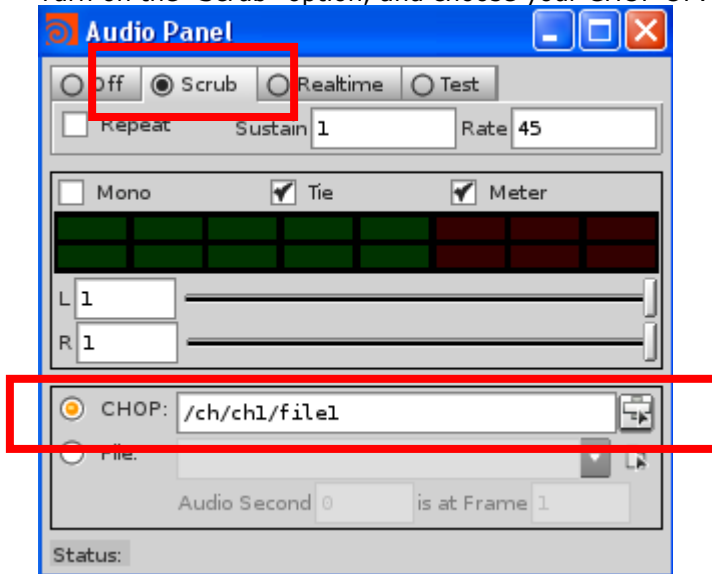
Turn on the real-time playback option:



Then, open the Audio Panel Window:



Turn on the "Scrub" option, and choose your CHOP OP:



Let's try a more complicated example: create a row of boxes, each box access the audio data at a different frame:

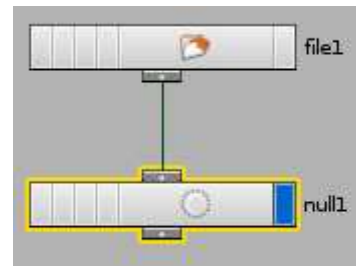
A BOX with center $y = 0.5$

A GRID with rows=1, columns=10.

Use `chopcf("$...$", 0, $F-$PT) * 5` to make different copies of the boxes grab **different frame** of the audio clip.

How to make it better?

Similarly, I will suggest adding a **NULL OP** under your FILE OP, and modify all the expression:
`chopcf ("/ch/ch1/file1",`
 to
`chopcf ("/ch/ch1/null1",`

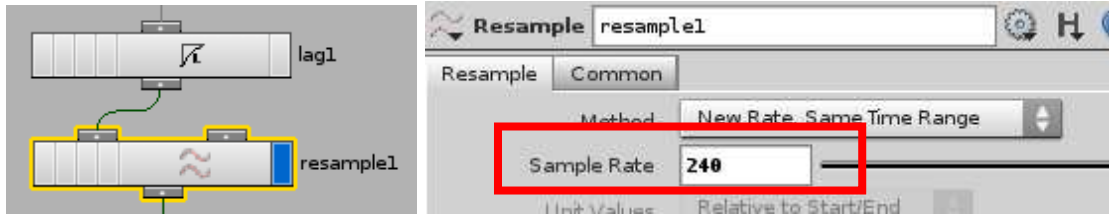


In doing so, you can insert any number of CHOP OP between the FILE and NULL, without need to modify your expressions on the SOP side.

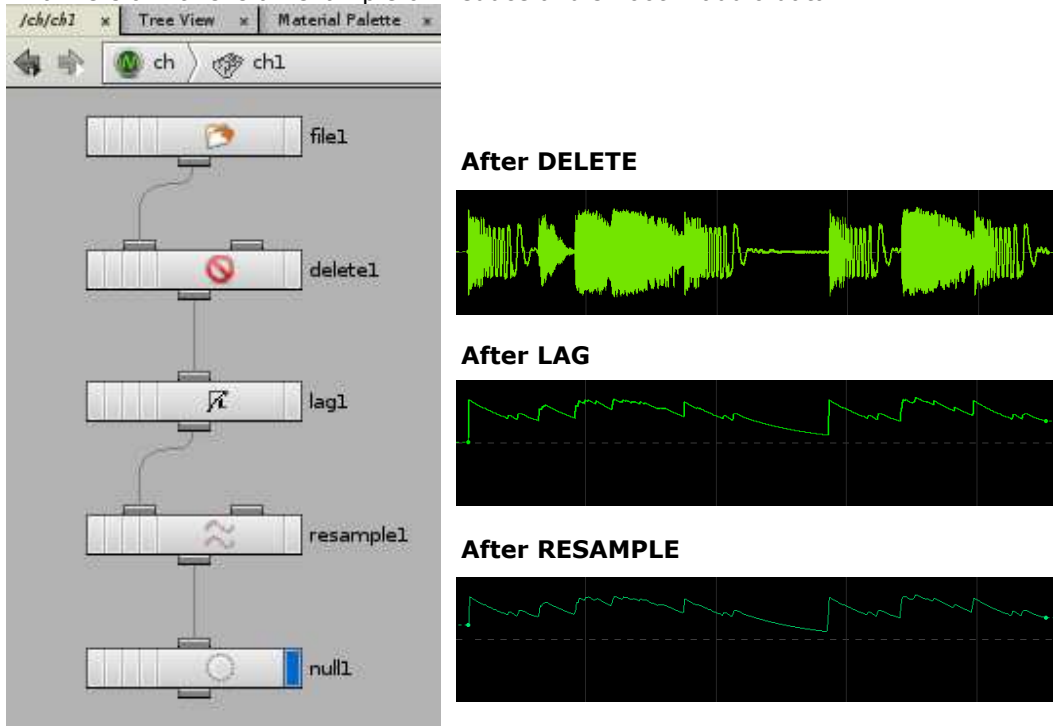
For example, you may want to insert a **DELETE OP** to delete the un-used channel (channel number 1 in our case):

Then, you can insert a **LAG OP**, which can be used to "smooth" out the audio data:

Lastly, you may also want to insert a **RESAMPLE OP**, which re-sample the audio data, i.e. to reduce the number of data in the channel. This can reduce the calculation time. Note that to control 3D animation we don't need so many audio data. A sampling rate of 240 (i.e. 240 data per second) is more than enough for 3D animation:



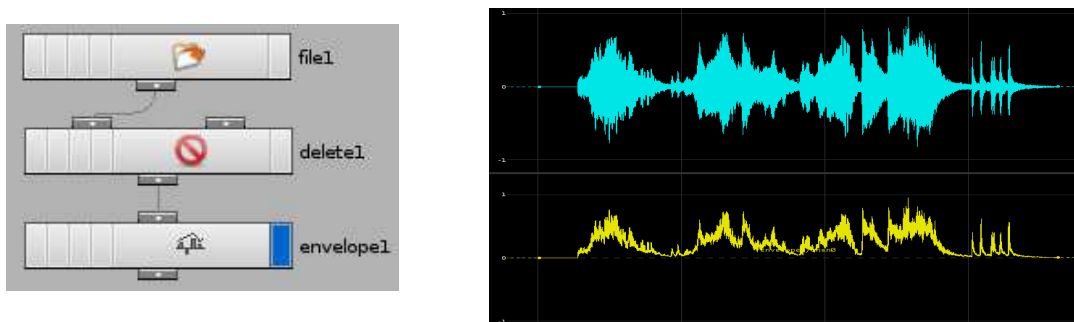
Final version: this is an example of "reduce and smooth" audio data:

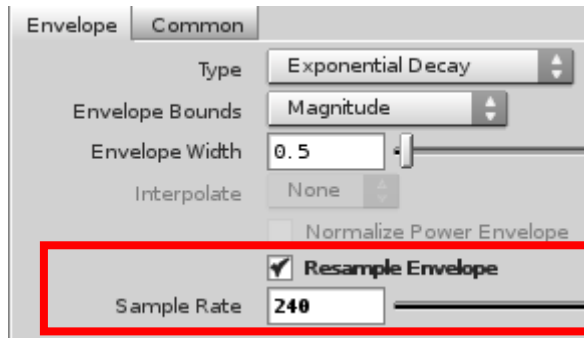


CHOP provides some other OPs for processing audio data. If you are familiar with audio and sound, you can try to explore more possibilities.

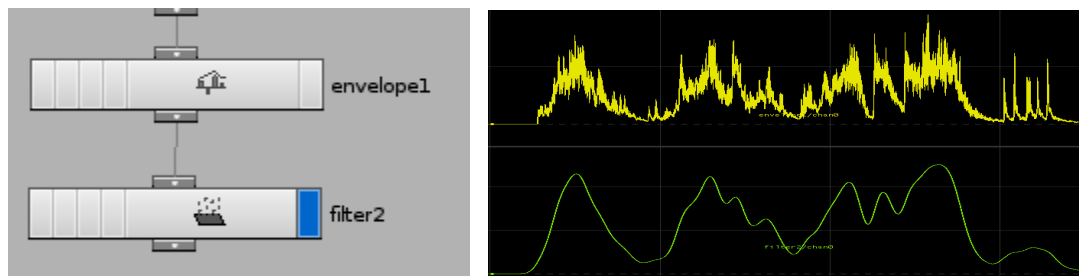
Another "reduce and smooth" method

In the previous example, the LAG serves 2 functions: remove the negative part of the audio, and gives a little bit "sustain" on the audio data. If you don't want the "sustain" effect, but just want to obtain the *envelop* of the audio data, you can use an **ENVELOP CHOP** instead. Note that the ENVELOP CHOP can also re-sample (i.e. reduce) the audio data at the same time:





Most likely you may also want to smooth-out the envelop data; you can use a **FILTER**
CHOP:

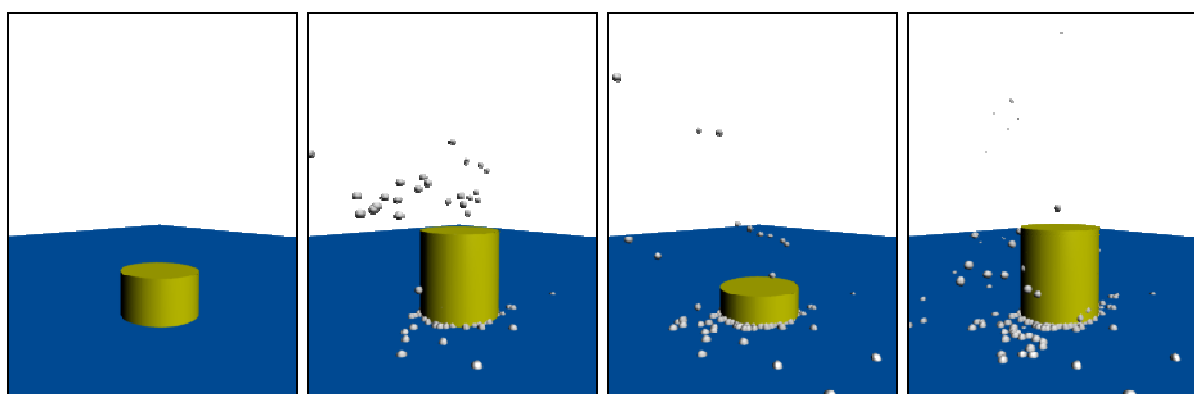


Note that this method sometimes will “over-smooth” an audio data, so the result may be quite different from the first method. Depends on your own requirement, you can choose the more appropriate method.

Exercise 1

Give the file **drums.wav**, try to apply the above techniques and PARTICLE to create the following audio-driven effect (the animation will be shown during the class).

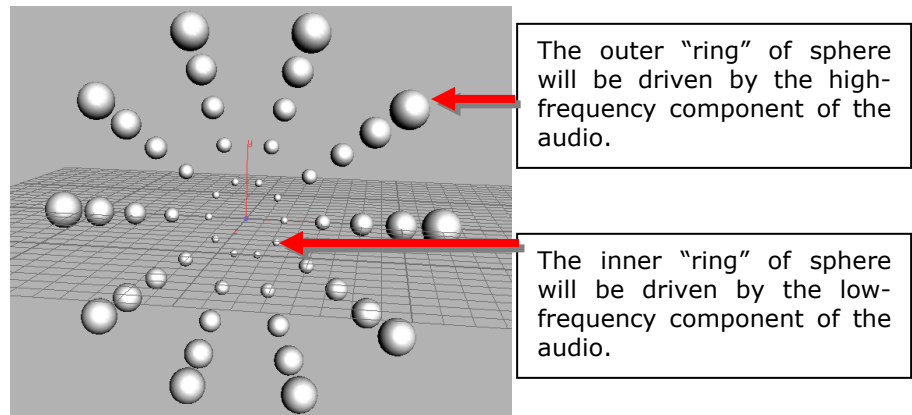
Hints: I used chopcf() to control the y-scale of the TUBE OP, and “Birth”, “Force (y-axis)” and “Turbulence” of the PARTICLE OP.



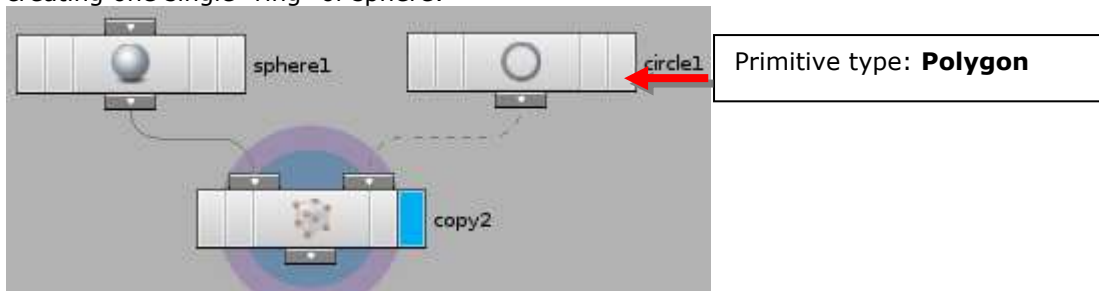
Beyond this exercise: try to copy METABALL onto the particles. It can give you a sticky liquid effect. You can also try the materials “Basic Liquid”. Examples will be given during the class.

Pitch-driven Animation

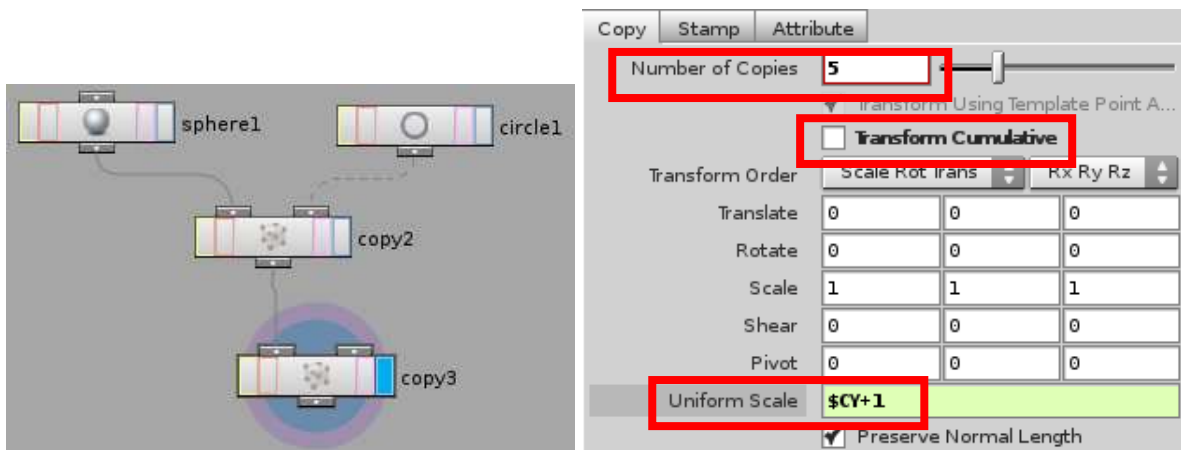
Besides amplitude-driven, you can also extract the pitch (i.e. frequency) of the audio, and drive the 3D scene using the pitch information. Let's look at a simple example. Say, I want to construct a scene like this:



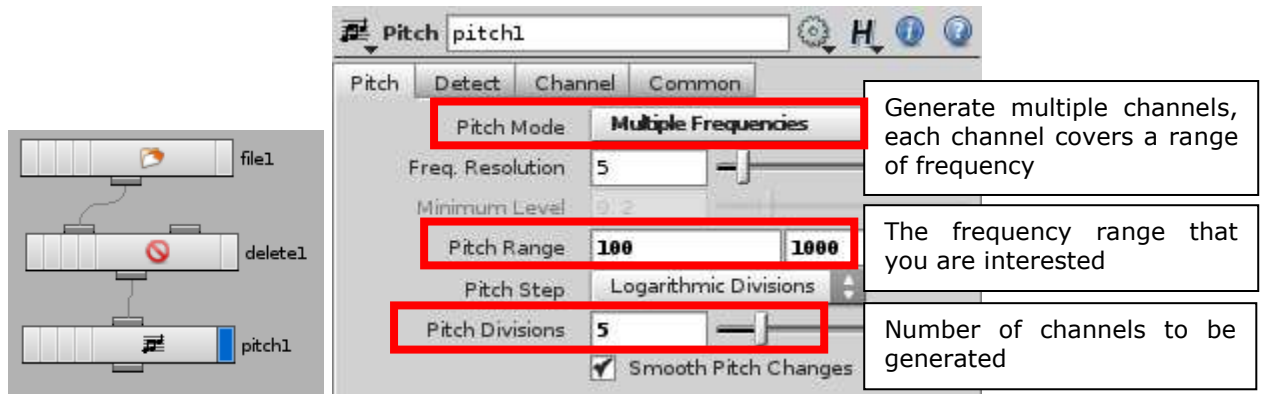
Firstly, in the 3D scene I create a GEOMETRY. GO inside the geometry, I start from creating one single "ring" of sphere:



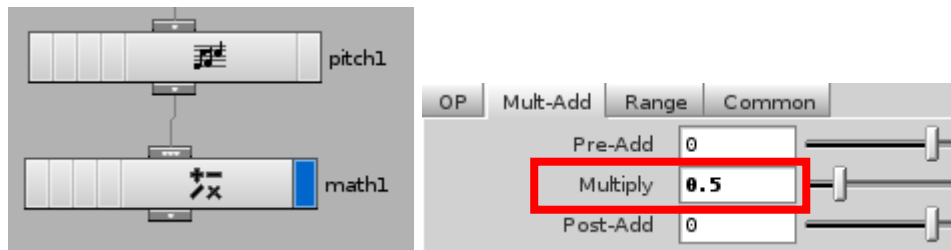
Afterward, I copy this component 5 times, using a COPY (without the 2nd input):



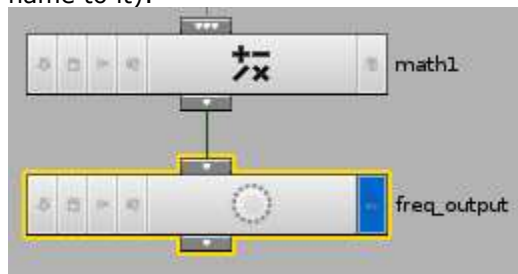
Then on the CHOP side, I can load an audio file using FILE (e.g. try **piano.wav**), and attach a **PITCH** to analyze its pitch:



After that, 5 channels are generated, where each channel covers a different range of the frequency. You may find that the value (i.e. the vertical axis) of the channel may be very large, not suitable for rotation. If this is the case, you can attach a **MATH** to re-scale the value:



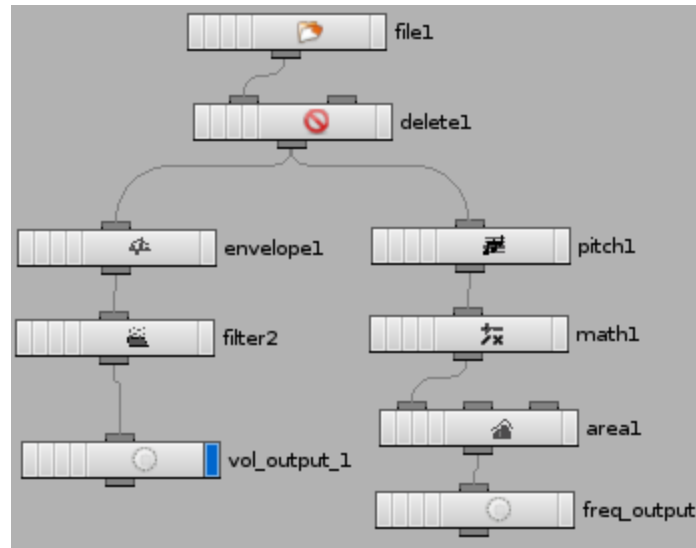
As a common practice, I usually attach a **NULL** for the final result (and give a meaningful name to it):



Lastly, I can use the audio data (the result of *freq_output* CHOP above) to control the rotation of individual "rings" of sphere. I will leave it as a class exercise.

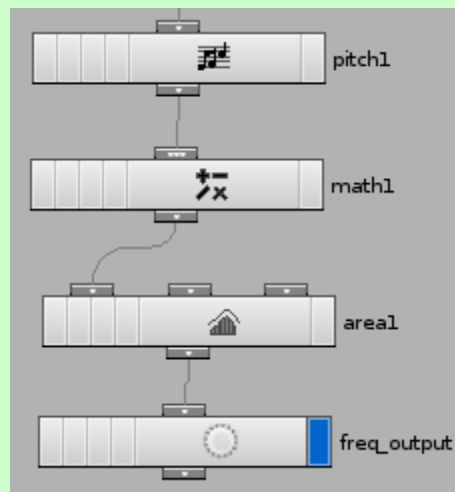
Exercise 2

1. Try to finish the above example.
2. Moreover, make the scale of the sphere to be controlled by the amplitude of the input audio. You don't need to create another CHOP network. You can simply branch out the audio FILE input into 2 different branches. It doesn't matter where the blue-flag is on: it only affects which channel will be shown on the screen, but it doesn't affect the expression calculation.



Beyond this exercise:

In case you do not want to use the pitch value directly control the rotation amount, but want to *accumulate the rotation value*, you can use an **AREA CHOP**. An AREA CHOP will use its input as "changes", and output an accumulated value:



Try to compare the results with AREA and without AREA, and you can see the difference.

**** Week 04 END ****