

Figure 1. Using a Transposition scalar to Scale the Sample Count

One other change we need to make is to replace `index~` with `play~`. Since `index~` does not interpolate sample values, we will degrade the quality of the sound if we use it. The `play~` object uses 4-point interpolation to read "fractional samples" from the `buffer~`, so its output will sound better when we read the `buffer~` at faster or slower speeds. Since the `play~` object takes millisecond values instead of sample values as input, we need to add a `sampstoms~` object to convert the samples to milliseconds and read the proper size chunk of sound from our `buffer~`.

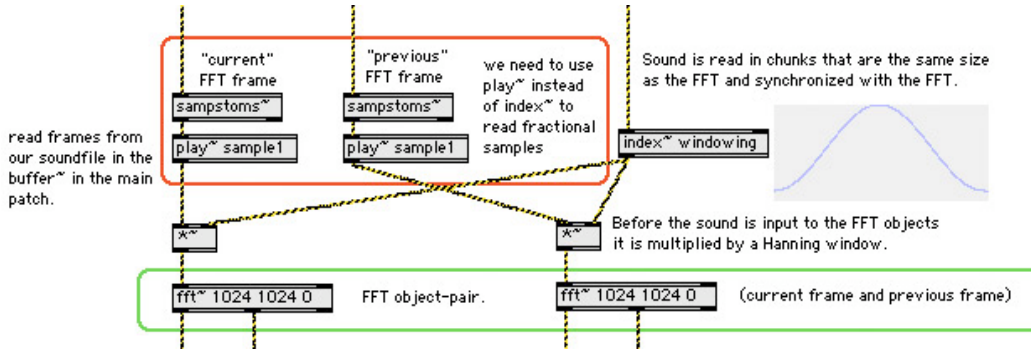


Figure 2. Using `play~` instead of `index~`.

Putting it all together, we can use this transposition for BOTH the polar and cartesian patches, since these changes do not affect the actual phase vocoder part of the patch.

One other important change we are making to the cartesian patch is to use a unique number for the `send~` and `receive~` name. We do this by beginning the name with a `#0` which will be replaced with a different number in each instance. This is explained in the Max/MSP documentation (Max4.6Topics.pdf, "Arguments: \$ and #, Changeable Arguments to Objects"), and lets us have multiple phase vocoder patches open at once without the `send/receive` names interfering with one another.

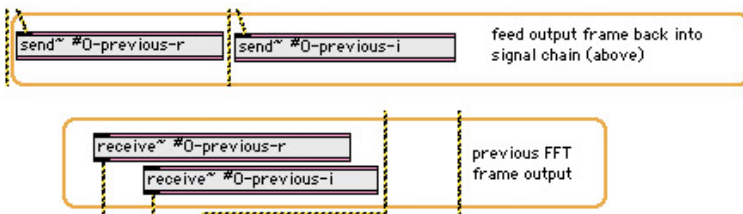


Figure 3. Using the Unique Patch ID Variable `#0`

Note that for both patches we make use of the "args" possibility for `pfft~`. Following the 5th argument to the `pfft~`, note the word "args" and the FFT size. This is convenient for allowing you to change the FFT sizes for the `fft~` and `ifft~` objects in the `pfft~` subpatch. (If you do change the FFT size, make sure to change the size of the windowing function in the message box below the loadbang, and double-click the loadbang to recalculate the window function at the new size.) Also, you might want to refer to the "Time vs. Frequency Resolution" technical detail in MSP Tutorial 26, since different sounds might work better with different FFT sizes.

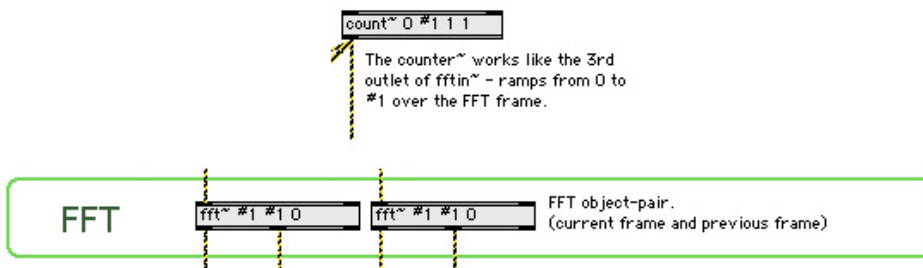


Figure 4. The `#1` Variable in the `fft~` and `count~` Objects.

## Freeze Effect

Although we can set the playback speed to zero, and thus "freeze" the sound at a certain point in time, the effect is rather static and mechanical. We can enliven our freeze effect by adding two things to our patch -- some random variance in the playback location, and some additional small random variance in the phase. Together they produce a much better freeze effect than can be achieved without them.

Here's what we need to do:

First, we need to only activate our freeze parameters when the playback speed is set to zero. Since our first inlet to the **pfift~** subpatch is the user-defined playback speed, we can simply check this value in order to activate our additional freeze parameters.

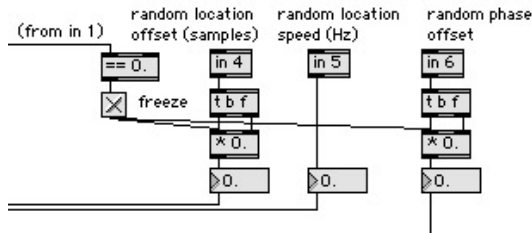


Figure 5. Checking the Input Speed to Turn on Addition Freeze Parameters

Next, we can use the **rand~** object to randomly oscillate the **buffer~** read location around the given playback location. We can control both the oscillation speed (with a frequency to **rand~**) and the oscillation depth -- which is our random playback location variance (with a signal multiply). Using just this technique automatically enlivens the frozen sound.

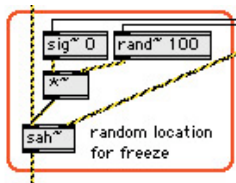


Figure 6. Offsetting the Frame Location with the **rand~** Object

Finally, we can add a very small amount of random phase deviation to the bins of our spectrum. Generally this is something we try to avoid in a phase vocoder, because it adds strange audio artifacts to our sound; however, in the case of a freeze, a very small amount of phase deviation from bin to bin actually breaks the mechanical sound of the freeze!

In our polar coordinate phase vocoder adding the phase is straightforward -- we simply add low volume white noise to our phase component.

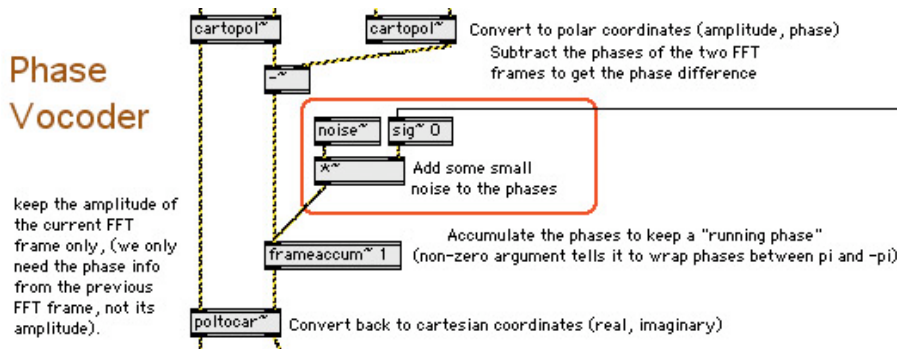
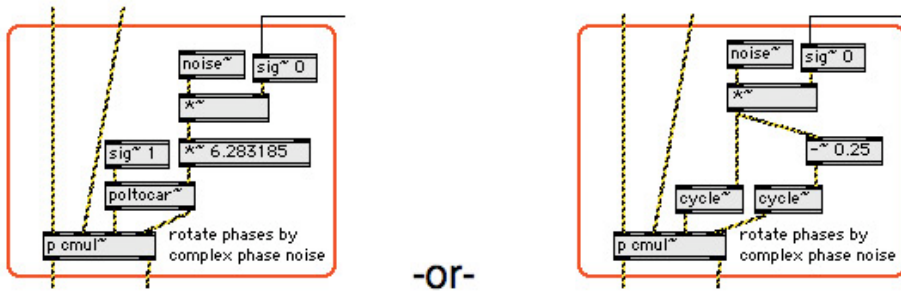


Figure 7. Using **noise~** to Add some Phase Randomness

However in the cartesian coordinate version of the patch things become slightly more complicated. We have to create a complex signal whose phase component has the noise in it, and perform a complex multiplication to rotate the phases. (Phase rotation via complex multiplication and division is explained in part 1 of the phase vocoder article.) One easy way to create the complex phase-only noise would be to use the **poltocar~** object with a constant amplitude value of 1 and the low-volume white noise as our phase. Another, more efficient, way is to use two **cycle~** objects whose phase is 90 degrees apart to represent the sine and cosine components of the complex signal, and control their phase input directly with the white noise. In both cases we would use our complex multiply subpatch, used elsewhere in the phase vocoder, to

add the phase deviation to our complex signal.



-or-

Figure 8. Making Cartesian Noise

Our cartesian phase vocoder now looks like this:

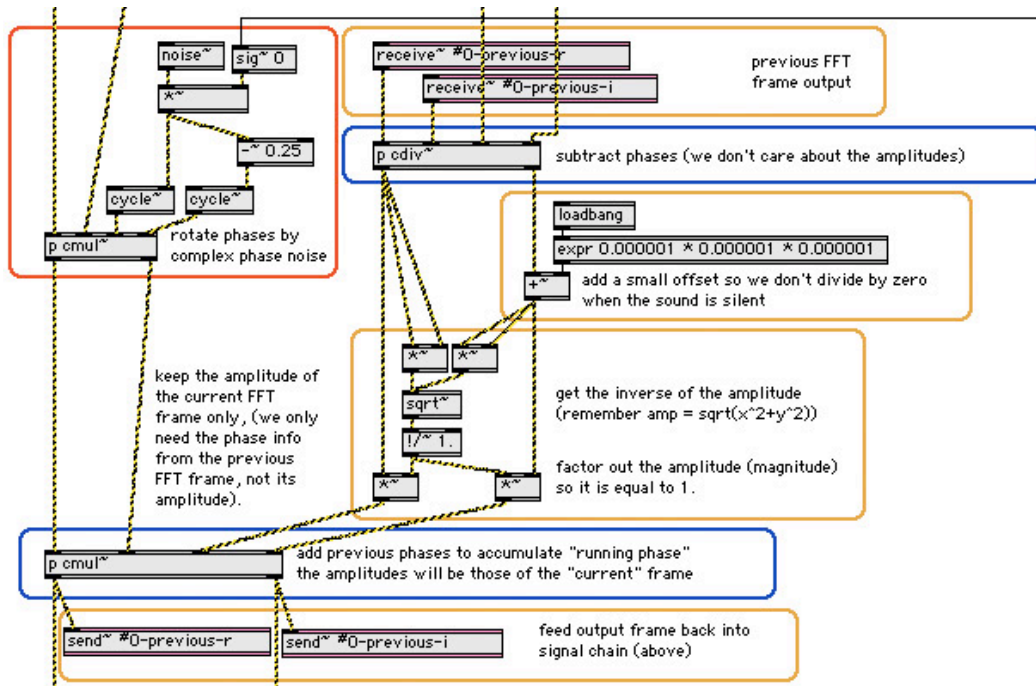


Figure 9. The Cartesian Version of the Phase Vocoder

It is quite a bit more complicated than the equivalent polar version shown in Figure 7, but you will notice that, as with the simple cartesian version shown in part 1 of the phase vocoder article, it is markedly more efficient. For complete views of these patches, we suggest opening up the patches themselves in Max/MSP and trying them out!

## Conclusion

With these additions to the phase vocoder we can control a sound's playback speed and transposition independently of each other, as well as "freeze" the sound with a bit of added liveliness. We have also improved the patch so we can provide arguments to the `pfft~` subpatcher in order to change the FFT size of the `fft~` and `ifft~` objects that we must use to read the sound from the `buffer~`. The patches provided with this article require Max/MSP 4.6.3, or will optionally run in other Max/MSP 4.6.x versions with the updated `fftin~` and `fftout~` objects found on the Cycling '74 website's [Incremental Updates page](#).

< [Email to Customers \(June 28, 2007\)](#) (0 comments) | [Interview with Leafcutter John, Musician](#) (0 comments) >

[The Phase Vocoder - Part II](#) | 0 comments ( topical, editorial, 0 hidden)

Display:  Sort:

-->