Hearing and Creating Audio Quality in Csound

Michael Gogins
michael.gogins@gmail.com

22 June 2021

This article describes some *methods* for improving the audio quality of Csound pieces, and also some *methodologies* for evaluating audio quality and training one's hearing, using both an audio editor and an ABX comparator.

Hearing is like thinking: you think you are thinking, but if you go to school and study with a good teacher of thinking, you learn that you only *thought* you were thinking. Similarly, the naive ear hears things that are not there — things one hopes to hear, or fears to hear — and fails to hear other things that really are there — things for which one has not consciously and reliably heard a *standard of comparison*.

This article contains two main sections, one much more important than the other.

The first section is a laundry list of techniques that have been found to improve the subjective, or artistic, sound quality of "tape music" style compositions rendered using Csound [5]. For the most part, these techniques have to do with choosing the best opcodes for a particular task, avoiding certain signal processing artifacts such as clicking and aliasing, and understanding how to balance levels and frequencies for a transparent listening experience.

These techniques cover ground that normally comes under the heading of several fields, including software instrument design, musical composition and arrangement, and audio mastering.

To illustrate these techniques, I have applied some of them to two well-known sample compositions that are distributed with Csound. In the https://github.com/gogins/python-abx repository, you will find both the original (`trapped.csd`) and a high-definition (`trapped-high-resolution.csd`) version of ***Trapped in Convert*** by Richard Boulanger, and both the original (`xanadu.csd`) and a high-definition (`xanadu-high-resolution.csd`) version of ***Xanadu*** by Joseph T. Kung.

The second section describes a *scientific* approach to discovering the *artistic* effect of these and other techniques. One renders the same piece twice in almost exactly the same way, differing only by one opcode choice, or one level change, or one parameter of some other objective technique. One then compares these soundfiles in several ways:

Audio Editor Load both soundfiles, each one on its own track, into an audio editor such as Audacity. Enable the spectral view for both tracks, and set the spectral view to show only frequencies that you can actually hear on your actual sound system, e.g. display only frequencies from 40 Hz to 10000 Hz (I am older, for you it might be 16000 Hz), and at a high frequency resolution, e.g. 8192 FFT bins. If possible, scale

the frequency dimension logarithmically, if you can then the middle of the display will cover the frequencies where music happens. Then use the ***Solo*** button to take turns playing segments of the files that look different. Play them in a loop, and switch back and forth between the two soundfiles until you are sure you can hear a difference between the files. Make the loop smaller and smaller until you have zeroed in on only the audible difference.

ABX Comparator Run the ABX comparator, and load the same two soundfiles. The comparator allows you to play a selected segment of sound first from one known source A, then from another known source B, and finally from an unknown source X chosen completely at random from A or B. One must then guess whether A or B was the source of X.

In the ABX comparator, set up the same loop points that you set in the audio editor. Listen carefully to both the A soundfile and the B soundfile in a repeating loop. Then play the X (unknown) soundfile, then click on ***X is A*** if you think X is A, or on ***X is B*** if you think X is B. Do this a dozen or so times, then click on ***Show*** to see the probablility that your guesses were correct just by accident. If you really heard a difference with the audio editor, then this probability should be quite low, $p < 0.5$ or better yet, $p < 0.001$.

The ABX comparator is the most reliable possible way of finding out what one actually can and cannot hear. Scientifically speaking, it is a double-blind experiment. The experimenter (the ABX software) does not know which source was chosen for the X segment, and the subject (the listener) also does not know which source was chosen. Therefore, there is no opportunity for subjective bias to influence the results — at least, not as long as one does not start throwing out results one does not like. (It is surprising how tempting this can become!) The binomial theorem then gives the likelihood for N trials that one has identified X by chance and not skill. It does not take many trials to reduce the odds that one has identified X by sheer luck to the vanishing point.

Even better, because the ABX comparator is so reliable, it can be used to learn how to correctly discriminate the smallest perceptible differences. In other words, the ABX comparator *trains one to hear*. That is the real reason for using this tool. And Csound is uniquely well suited to it, for in live performance, or even in a recording studio, it is very hard to produce two versions of the same piece that differ only in one small parameter. But with Csound, all it takes is a text editor.

To illustrate the use of the methodology, I suggest some segments in the two renderings of ***Trapped in Convert*** and ***Xanadu*** to compare both in an audio editor and by using the ABX comparator. I am confident that after a few sessions with these tools, many listeners will experience a sense of revelation — just as I did.

1 Audio Quality in Csound

Currently, studio recording is done to stereo, surround sound (5.1 or 7.1), or Ambisonics on computers, or on hard disk recorders using 24-bit or floating-point samples at a rate of 48,000, 88,200, 96,000 or even 192,000 sample frames per second. This is "high-definition audio." Analog tape recorders also are still sometimes used, and can also be high-definition, if run at high speed and/or with noise reduction. CD-quality audio is of rather lower definition: stereo sound with 16 bit integer samples at 44,100 samples per second.

At the present time, the only consumer electronics formats that can reproduce high-definition audio are, roughly in increasing order of quality:

1. AAC (the audio part of MP4s) at high bit rates and depths.
2. SACD (i.e., DSD).
3. DVD-Audio or, alternatively, DVD-Video containing high-resolution audio.
4. High-definition PCM soundfiles, or lossless encodings of them such as FLAC, ALE in an Ogg Vorbis container, or ALE in an MP4 container.

Right now, the closest thing to a "universal" format for high-definition audio is Apple Lossless Encoding (ALE) in an MP4 container. This can hold both high-resolution video and lossless high-resolution audio. It will play on many devices and computers, but does not appear to be supported on YouTube.

I hope that will change, because YouTube is becoming the most common way for most people to listen to music. For YouTube, first produce a high-resolution PCM soundfile, then convert it to an MP4 video with high-resolution AAC. This, streamed from YouTube, can provide better than CD quality.

High-definition audio, on good speakers or headphones, sounds distinctly airy, present, spacious, and undistorted. CD-quality audio, by contrast, can sound flat, shrill, harsh, and flat or boxed in. Usually, this is the result of cumulative mistakes made in this less forgiving medium – CDs actually are precise enough to reproduce most of what we hear. Therefore, CDs made by experts can sound very good indeed, except for their more limited dynamic range and slightly less detailed quiet sounds. Normally, it takes educated ears to hear these differences.

Vinyl records of high quality are not directly comparable to digital recordings. They have their own virtues and flaws. They are more detailed, airy, and spacious than CDs, but can have harmonic distortion, rumbling, hiss, and crackling. In general, well-made records, especially if pressed on thick vinyl from direct metal masters, are roughly equal to high-definition audio in aesthetic quality, even if they are not really as precise.

All of these remarks set aside questions of "warmth" or "musical quality" in sound. Vinyl records, audio tape, and analog electronics introduce a little harmonic distortion, which creates a soft, burnished glow on the sound that some people prefer to hear. Such "warmth" is not what this article is about. If the composer or producer wants this sound, it can easily be created using Csound alone, without any analog gear, simply by convolving the signal with an appropriate impulse response.

Csound is eminently capable of high-definition audio. It can render to *any* number of channels, at *any* sampling rate, using floating-point samples. Csound also contains high-quality software implementations of all the effects applied by mastering engineers. Therefore, Csound can sound as good or better than the best studio gear. However, this does *not* happen by default. It takes a little work, and that is what this essay is about.

If you have a professional or semi-professional audio interface on your computer, you can play high-definition soundfiles made with Csound, although you will not hear their full dynamic range unless you have professional monitoring gear.

The constant goal in critical listening is to *hear as accurately as possible the signal as it actually exists on the recording.* Similarly, the constant goal in audio production is not to make a piece sound good on a typical listener's sound system — it is to *make the piece sound as good as possible on the most accurate possible listening system.* If you lose sight of these realities for any reason, then whether you know it or not, you will become lost in a wilderness of illusions. Experienced mastering engineers know that making a piece sound good on the most accurate possible sound system will make the piece sound better on most listeners' systems and worse on a few, whereas trying to make the piece sound

good on one sort of inferior sound system will indeed make the piece sound much better on that one type of system, but only at the cost of making it sound much worse on almost all other systems.

I strongly recommend that you listen to all soundfiles from this article through real studio monitor speakers, with the flattest possible frequency response, in an acoustically deadened room, at a volume that is about as loud as you can listen to indefinitely (around 80 dBSP). If you don't have such a listening environment, then use real studio monitor headphones plugged into a high-definition interface, but be careful not to listen at too high a level.

Specific technical advice in decreasing order of importance (all this assumes you don't care how long it takes to render a piece, only if it sounds good):

1. Some of the sounds made by Csound have no counterpart in other kinds of music. They may contain excessive high frequencies, aliasing distortion, or other kinds of noise. On the other hand, the sounds can be of extreme clarity and precision — *hyper-real*. You need to be constantly aware of what your sounds *actually sound like*.

2. Always render to floating-point soundfiles at 96,000 sample frames per second. You can translate them to 24 bits or to CD quality later if you want, but having the extra precision and dynamic range is vital.

3. If you use sampled sounds, use the best samples you can possibly find. Pay if you must!

4. Also if you use sampled sounds, beware of their own ambience clashing with any reverberation or other ambience you set up using Csound. Samples may also have unwanted noise — it may be possible to de-noise them (Csound has facilities for doing this too).

5. If you hear any unwanted clicks, or see any unwanted vertical lines in a spectral view of the soundfile, use a "de-clicking" envelope to wrap all your instrument final output signals.

6. Watch out for aliasing, which can make sounds buzzy or harsh, in frequency modulation and wavetable oscillators. Aliasing happens when the signal contains frequencies above half the sampling rate (the Nyquist frequency), so that under digital sampling they reflect or fold back under the Nyquist frequency. For so-called "analog" sounds with simple waveforms such as square or sawtooth waves, use non-aliasing opcodes such as `vco` or `vco2`. You do not need to worry about aliasing with plain sine or cosine waves.

7. Use a-rate variables for envelopes and, in general, wherever opcodes permit. This enables decent results with `ksmps = 100` or so.

8. For final renderings, always render with `ksmps = 1`. If you use a-rate envelopes this may not be not necessary. With `ksmps > 1`, render with the `--sample-accurate` option.

9. In general, if an opcode has both an interpolating form and a non-interpolating form, use the interpolating form, e.g. use `tablei` instead of `table`.

10. Use only the most precise interpolating oscillators, e.g. use `poscil3` instead of `oscil`.

11. For all wavetable oscillators, the larger the wavetable, the less noisy the signal. For each doubling of wavetable size, the noise goes down by about 6 dB. For 65536, the noise floor is about -96 dB.

12. Be vigilant for artifacts and noise introduced by various digital signal processing algorithms, especially echoes in reverberation. Don't over-use effects — this is a very common error that you can fix by listening to good examples of studio and live recording.

13. Try rendering with dither (`-Z` option).

14. Experiment with some modest compression, e.g. by using the `compress` or `dam` opcodes.

15. Use the 64-bit sample version of Csound (this is now standard).

16. For final (also known as "reference") monitoring, if you must listen in a smallish room with near-field monitors as most of us do, start listening at a C-weighted dBSPL of between 75 and 85, closer to 75, and your digital signal should be about -20 dBSPL; this level should go all the way down the signal chain. ***Sound on Sound*** magazine has a useful article on this [4].

The above are *technical* considerations. *Artistic* considerations are more subjective, but the following rules of thumb are generally followed in music production:

1. For art music that is not acousmatic, the use of signal processing and effects should be minimized. In general, the listener should not be aware that such effects have been used. If they are audible to the listener, they should normally be perceived as being an integral part of the listening space or of a particular voice.

2. For acousmatic music that uses processed sound, listen very critically; avoid using the same processing on all sounds, and avoid the almost inevitable builidup of convolution smear.

3. If more than one voice is sounding at the same time, the composer usually intends either to fuse the sounds, or to separate the sounds. To fuse the sounds, their spectra should overlap, their spatial locations should overlap, and their pitches should either be a unison, or in an octave relationship. Their envelopes may or may be the same shape, but the attack portions should not be too different. To separate sounds, any one or more of the above considerations should be negated: their spectra should not overlap; and/or their spatial locations should be different; and/or their pitch-classes should be different; and/or their envelopes should not be the same shape.

4. Usually, solo voices and bass lines should be acoustically separated from the rest of the music.

5. Computer music and electroacoustic music tends to be shrill in comparison with historical traditions for art music. Many such pieces can be improved by rolling off the treble equalization or, better yet, changing the design of the instruments themselves.

6. Computer music and electroacoustic music tends to be bass-shy in comparison with other genres of music. Many such pieces can be improved with a little "big bottom."

7. Computer music and electroacoustic music tends to be loud in comparison with other genres of music, excepting the louder forms of rock and dance music. Some pieces would benefit from a quieter average level combined with a larger dynamic range.

8. Both computer music and electroacoustic music use a great deal of signal processing, which often causes pieces to acquire a particular artifact technically known as "convolution smear." It can sound like smearing, ringing, or a sheen overlaying the sound. This sound may or may not be artistically desirable, but the composer needs to know when it is there so that he or she can decide whether or not to use it.

9. Computer music, as opposed to purely electronic music, uses digital signal processing which, in turn, frequently causes aliasing distortion. It can manifest itself as false tones, false harmonics, or graininess or grittiness in sounds. Again, the effect may or may not be desirable, but the composer needs to know when it is there.

10. Computer music, electroacoustic music, and studio recordings in general tend to combine sounds into an artificial sound stage. Our sensation of the location of sounds is complex, rather accurate, and depends on several cues including the relative *loudnesses* of a sound with respect to direction, the relative *phases* of the sound with respect to different directions, the type of echoes or reverberation associated with the sound, and even the frequency equalization of the sound (high frequencies are attenuated by distance), not to mention Doppler effects as sound sources move through space. Most recorded music features a collapsed, artificial, static sound stage. In computer music and electroacoustic music, especially when using sampled sound, it is common to use only relative loudness as a spatial cue, and to attempt to place sounds with quite different reverberant qualities onto the same sound stage. Again, this may or may not be desirable, but the composer needs to hear what the sound stage actually is, and to be able to identify its causes....

2 Comparing Renderings

There used to be a good free ABX comparator that could be downloaded from `http://www.kikeg.arrakis.es/winabx`. This no longer appears to be available. I have therefore created a new version of this application, which can be downloaded from `https://github.com/gogins/python-abx`. In that repository, you will find both the original and high-definition versions of ***Trapped in Convert*** and ***Xanadu***:

```
trapped.csd
trapped-high-resolution.csd
xanadu.csd
xanadu-high-resolution.csd
```

To see what changes I have made to improve sound quality in these pieces, you can run the ABX comparator to highlight the differences between versions, as shown in Figure 1. Open a terminal, navigate to your `python-abx` directory, and render each of the two pieces in both the normal version and the high-definition version, using the following commands:

```
csound trapped.csd
csound trapped-high-resolution.csd
csound xanadu.csd
csound xanadu-high-resolution.csd
```

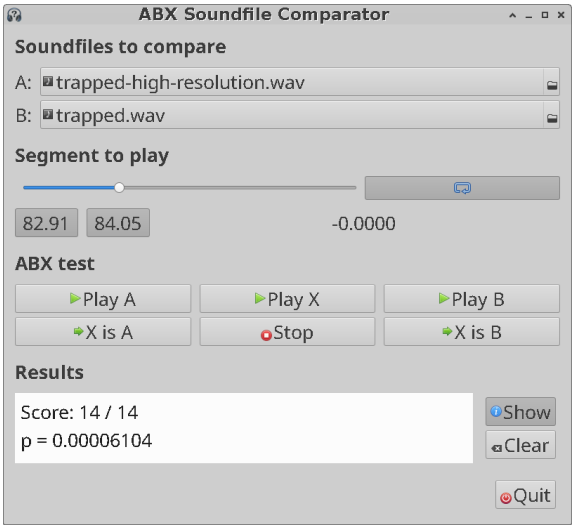

Figure 1: **Comparing Versions of *Xanadu***

Use a soundfile editor such as Audacity [1] to determine if the soundfile amplitudes are the same in both renderings – which means within 0.25 dB of each other. If, but *only* if, the amplitudes of the renderings are different, then use the editor to remove any DC bias, and normalize the level in each of these soundfiles to -3 dBFS, to ensure that each source has the same subjective loudness, as shown in Figure 2.

Equal amplitudes are *essential* whenever you do an ABX comparison, because:

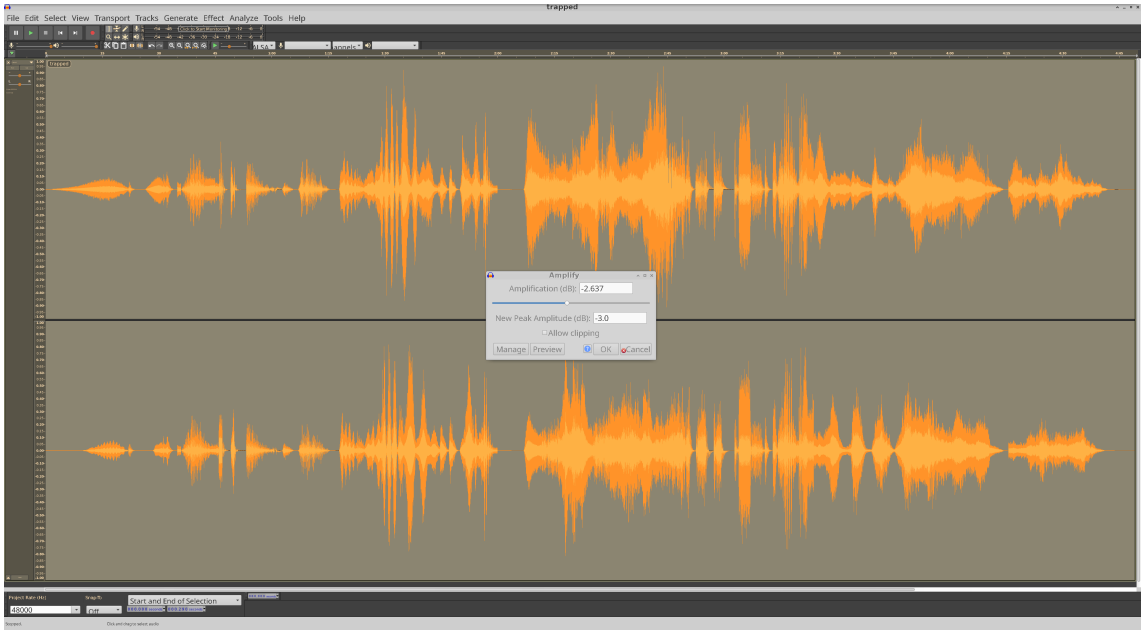

Figure 2: **Normalizing *Trapped in Convert***

- People are *quite* sensitive to loudness.
- Given two sounds A and B, if A is louder, people will tend to prefer it, even if at the same loudness they might prefer B.

- Different synthesis and signal processing techniques, e.g. compression, can modify signal amplitudes.

2.1 Comparing with Audacity

When all of your pieces are rendered and, if necessary, normalized, import them as two separate tracks in Audacity.

Expand the selection to fit the window, select the Spectrogram view, and change the settings for the Spectrogram view to use logarithmic scale, with a frequency range from the lowest your sound system accurately plays to the highest you can actually hear (in reality, 20 KHz is too high, a young person should use 16 KHz and an older person might need to use 10 KHz). You should set the number of frequency bins to show frequency as accurately as possible without, however, losing too much time accuracy; 8192 bins is a good compromise. At first glance, the two spectrograms may look identical. On closer inspection, however, you will begin to see differences, e.g. at about 83 seconds, `xanadu.wav` (the lower track) is visibly different from `xanadu-high-precision.wav` (the upper track). This is a subtle difference indeed but if you loop over a selection from 83 to 84 seconds, and alternately ***Solo*** first the upper then the lower track, you should clearly hear it (Figure 3).

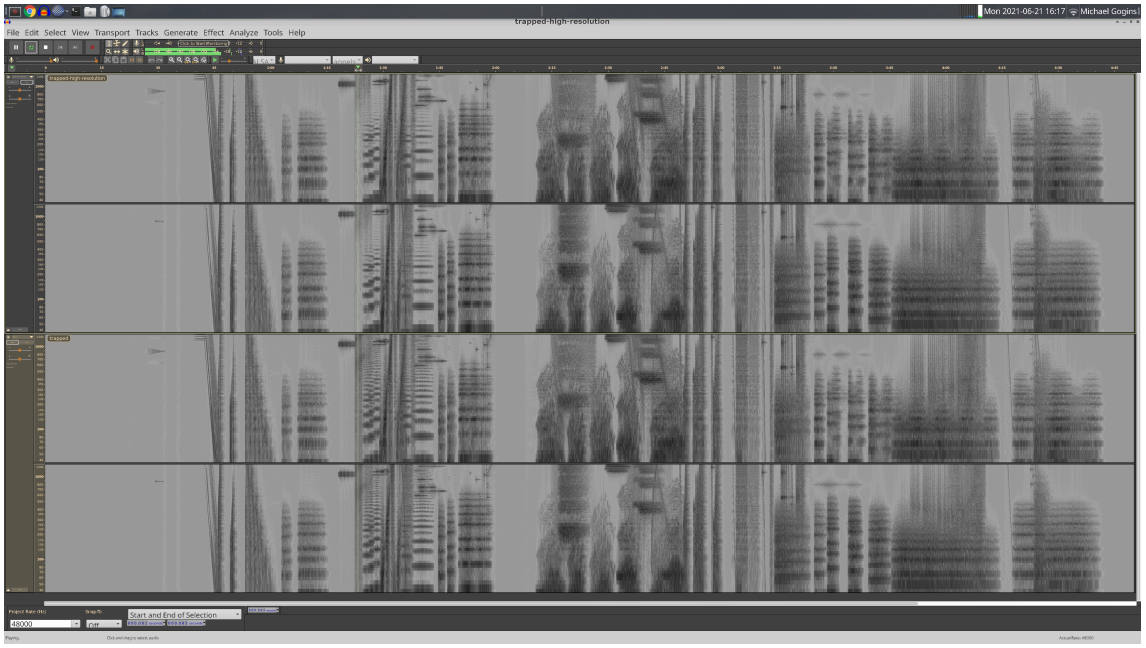

Figure 3: **Comparing *Trapped in Convert* in Audacity**

Zooming into this selection (Figure 4), the audible difference is manifest in the slighty different bands of energy in the midrange frequencies.

2.2 Comparing with ABX

Now, run the ABX comparator and load the same two versions of ***Trapped in Convert***. Select the same segment from 83 seconds to 84 seconds, and enable looping. The reason for using such a short segment is that human short-term memory for sounds is much more accurate than long-term memory, and short-term memory only extends to about 5 seconds. Make sure that your listening volume is loud, but not uncomfortable. If while listening your ears hurt or pop, *immediately* reduce the volume until they do not.

Figure 4: **Comparing a Selection of *Trapped in Convert* in Audacity**

Listen to A and B several times to see if you think you can hear any differences between them. Then, listen to X and decide whether X is A or B. You are free to listen to A, B, and X any number of times and in any order. I find that the best approach is to listen to A and B repeatedly until some feature that is different begins to emerge from the listening process. I can then listen to X and see if it does or does not have this discriminating feature.

When you have made up your mind, click on the ***X is A*** button or the ***X is B*** button to indicate your choice, then click on the *Next trial* button. Keep repeating these trials. If after 10 or 20 trials the probability that you are guessing goes below 5% and stays there, then you probably actually *can* hear a difference between A and B. But if after a large number of trials you can never get the probability you are guessing to stay below 10%, then no matter what you may think, you probably *cannot* hear any difference between A and B.

This is painstaking work, but it is the *only* way to make sure you really *are* hearing what you *think* you are hearing.

As time goes on, you should find that your hearing of this piece becomes quite a bit more perceptive. More importantly, you should be able to form a reliable judgment of which rendering is better according to your own musical taste. This may or may not be the high-definition version! *The vital thing is to improve the accuracy of your hearing with respect to your own musical judgment.*

You also should find that you can more quickly decide whether or not you really can hear a difference between the sources — which means that you really are improving your musical hearing.

I will not, in this article, explain what I think the other differences are between the regular rendering and the high-definition rendering of ***Trapped in Convert***. You should do your own experiments. But when you have listened to a number of segments, you may wish to try listening to each version all the way through, in order to see if you can still hear the differences that you had learned to identify.

Once you know where you are truly hearing differences between the soundfiles, load them both into Audacity and view then side by side in the time/frequency view.

I will finish with one more comparison (Figure 5). This screen shot from Audacity shows a selection near the beginning of ***Xanadu***. Note, in the original version, that the glissading partials go both up and down, which sounds bad and probably is not what the composer intended, whereas in the high-precision rendering, the glissading partials go only down. However, this does not tell us which of the several changes made in the high-resolution version stopped the false glissades.

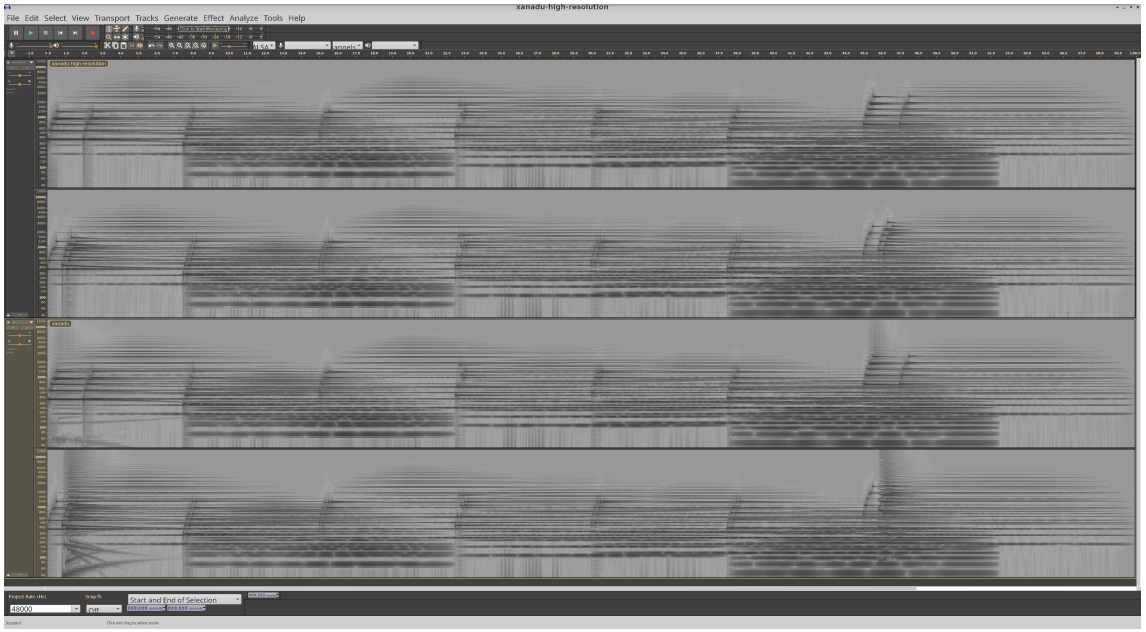

Figure 5: **Odd Glissades in *Xanadu***

If you *really* want to understand what is doing on, make copies of `trapped.csd` and `xanadu.csd`, use a diff tool to add one modification at a time to your copies from `trapped-high-resolution.csd` and `xanadu-high-resolution.csd` files, and do the ABX comparison all over for each modification. In preparing `xanadu-high-resolution.csd`, I made the following changes:

1. Use `--sample-accurate`: Small difference at 38 seconds; both renderings have funny glissades at the beginning.
2. Set `ksmps = 1`: Additional difference at beginning; funny glissades now gone from the high-resolution rendering.
3. Wavetables enlarged to 524288: Looks and sounds like the previous step.
4. Use `poscil3` for `oscil`: Looks and sounds like the previous step.
5. Use `deltap3` for `deltapi`: Looks and sounds like the previous step.

I infer that the false glissades are largely caused by linearly interpolating delay times and/or the phases in frequency modulation across blocks of samples. Indeed, further experiments showed that using either `ksmps = 1` *or* arate envelopes removes the false glissades.

Another source of deeper insight might be to visit Dominique Bassal's web site on mastering [2], download some of his pre-mastered/post-mastered example soundfiles, and do ABX comparisons on them. Bassal is an acknowledged expert in mastering computer music, and his on-line article *The Practice of Mastering in Electroacoustics* [3] provides a much more experienced and in-depth review of some of the issues (i.e., those not specific to Csound) that I have tried to cover here.

3 Conclusion

Well, I hope this article has been useful to you!

I believe that learning these methods, and above all the ABX methodology, has made an enormous difference to my own ability to hear my own music more objectively.

I also have a renewed appreciation of what I am now better equipped to realize are astonishing feats of perception and signal processing on the part of the best computer musicians, recording engineers, and mastering engineers.

And I believe my own ability to work at that level has improved, at least a little bit, as a result of the ABX comparator.

References

[1] *Audacity*. https://www.audacityteam.org/ (cit. on p. 8).

[2] Dominique Bassal. *Mastering*. http://cec.concordia.ca/pep/mastering_e.html (cit. on p. 11).

[3] Dominique Bassal. *The Practice of Mastering in Electroacoustics*. http://cec.concordia.ca/ftp/The_Practice_of_Mastering.pdf. December 2002 (cit. on p. 12).

[4] Hugh Robjohns. *Establishing Project Studio Reference Monitoring Levels*. https://www.soundonsound.com/techniques/establishing-project-studio-reference-monitoring-levels. May 2014 (cit. on p. 6).

[5] Barry Vercoe, John ffitch, et al. *Csound*. http://csound.com. Accessed 14 April 2021 (cit. on p. 2).