LOSSY DISTORTION AS A MUSICAL EFFECT

by

ARDEN BUTTERFIELD

A THESIS

Presented to the Department of Computer Science
and the Robert D. Clark Honors College
in partial fulfillment of the requirements for the degree of
Bachelor of Science

January 2023

An Abstract of the Thesis of

Arden Butterfield for the degree of Bachelor of Science
in the Department of Computer Science to be taken Winter 2023

Title: Lossy Distortion as a Musical Effect

Approved: *Dr. Jon Bellona*
Primary Thesis Advisor

Lossy audio compression is a digital process that uses models of human hearing to remove parts of the sound deemed less important, in order to compress audio to much smaller file sizes. The MP3 encoding process, one of the most famous lossy audio compression formats, can impart audio with a distinctive watery, muffled sound at higher levels of compression. This sound, which I call "lossy distortion," can be used as a musical effect to inspire nostalgia for early digital audio, or for a more abstract, ethereal sound. In analyzing creative uses of lossy distortion and existing plugins for lossy distortion, I identify some desirable features that are lacking from existing plugins. To fill these gaps, I built two lossy distortion plugins. One, called Empy, gives the user control over a wide variety of lossy distortion sounds. The other, Fish, emulates a particular sound of lossy distortion that other plugins struggle to achieve, by modifying a popular piece of MP3 encoding software. In their sound and user interface, these plugins explore new ground in the rapidly developing field of lossy distortion plugins.

Copyright Page

Acknowledgements

I would like to thank Dr. Jon Bellona for his invaluable assistance throughout this project as my primary advisor. I would also like to thank Dr. Brian McWhorter of the Clark Honors College and Jared Knofczynski of the computer science department for serving on my committee. This thesis would not have been possible without the patient advice of Dr. Trond Jacobsen, who encouraged me to pursue a topic that truly excited me, nor would it have been possible without the teachers and mentors I have been so fortunate to learn from. Thank you to Julie Wright, Branic Howard, and Dr. Julianne Newton, to name a few. I am also grateful to be surrounded by so many wonderful peers, who have offered helpful encouragement and ideas throughout this project. Thank you to Stephen Leveckis for design feedback, and Radio Platenberg for getting me to look at MP3 compression in a new light. Lastly, I would like to thank Zeke, who has been my muse from beginning to end, as well as my family, who have supported me in so many ways throughout my degree.

This project is supported by funding from the Undergraduate Research Opportunity Program, through the UROP Mini-grant.

Table of Contents

List of Accompanying Materials

1. **Audio Examples**
   This directory contains audio examples demonstrating the sound of Empy and Fish. The Empy examples show the effect of various Empy settings on the same drum track.[1] The Fish examples show the effect of various Fish settings on the same song.[2] Also included are a sample of "Hey Ya Low Quality,"[3] and Fish applied to "Hey Ya!" by Outkast.

2. **Video Examples**
   This directory contains video examples demonstrating the user interface of Empy and Fish, in Ableton Live.

3. **Source Code**
   This directory contains the source code of Fish and Empy, the two plugins I created for this thesis. The directory also contains instructions for building the source code.

4. **Install**
   This directory contains the compiled VST3 and Audio Units files for Empy and Fish, as well as instructions for installing them.

[1] Courtesy Tier, recorded at Brooklyn Recording, 2006 by Jon Bellona. Used with permission from Jon Bellona.
[2] Cover of "Kyoto" by Phoebe Bridgers. Audio Recording Tech III class MUS 482/582, University of Oregon. Alivia Nelson - vocals. Nik Barber - drums. Skyler Beard - bass. Evan Hamada - trumpet. Isaak Boorstein - guitar. Jack Keith - guitar. Lauren Griffith - French horn. Ben Asbury - pocket piano. Used with permission from Jon Bellona.
[3] Bizzymcbob, *Hey Ya! Low Quality*, March 7, 2021, YouTube video, March 7, 2021, https://www.youtube.com/watch?v=LMaG_uOa440.

List of Figures

List of Tables

Introduction / Background

You have probably heard lossy distortion before. From the thin sound of a poor Zoom connection to the muffled, underwater noise of an old audio file that's been sent around on the internet several times, lossy audio compression can distort sound in distinctive ways. These distortions come from perceptual audio compression algorithms that use models of human hearing to remove parts of the sound deemed less important, in order to compress audio to much smaller file sizes. These compression algorithms are referred to as "lossy" compression, meaning that they remove information from the sound.[4]

In this thesis, I investigate lossy distortion, the recognizable sound of audio compressed by perceptual lossy compression codecs such as MP3.[5] MP3 is a popular lossy compression format for music and other sounds. Released the 1990s, MP3 is still a common compression format today. Specifically, I argue that lossy distortion has a place as a creative musical effect, due to its distinctive sound, widespread use, and rich history.

To explore lossy distortion as a musical effect, I made two lossy distortion plugins. Plugins are digital audio effects or instruments that can run inside of digital audio workstations. Plugins are real-time effects: instead of processing an entire audio file at once, plugins process a continuous stream of audio, making them suitable for live use as well as in the studio. The user can change the sound of the plugin in real-time

---

[4] Other compression algorithms, such as FLAC, are "lossless." When a lossless format is decoded, you are left with the original sound; decoding a lossy format will never get parts of its signal back.

[5] A codec, short for encoder/decoder or compressor/decompressor, is an algorithm for encoding a bitstream such as an audio signal into a compressed form and later decoding the compressed form into a bitstream resembling the original. Scot Hacker, *MP3: The Definitive Guide* (O'Reilly & Associates, 2000), 8.

through knobs and other controls in a plugin's graphical user interface. These features mean that plugins are an easy and popular way for musicians to process audio. In recent years, several companies have released lossy distortion plugins, which are reviewed in a later section. Some are aimed at lossy distortion as a creative effect, while others are made for testing how a track will sound when released as an MP3. Some plugins act as a wrapper around existing lossy compression codecs, while others use custom distortion algorithms.

Despite this variation, there are two potentially useful directions where none of these plugins have gone. The first is to give the user a large amount of control over the steps of the lossy compression process. The user would be able to change the specifics of the model of human hearing in the plugin, as well as how it gets used to distort the sound, to achieve a wider variety of sounds than can be achieved with any existing plugin. The second direction is almost the opposite of the first: instead of laying the technical process of lossy distortion bare, a plugin in this direction would be as simple to use as possible. Several popular plugins, including Dada Life's Sausage Fattener, FL Studio's Soundgoodizer, and the Waves OneKnob plugins cater to non-technical, novice producers with a simple user interface, often featuring one or two knobs, which are labelled with cheerful, accessible names like "cut," "honk," "fatness," or "color."[6] The video "Hey Ya Low Quality," discussed in a later section, shows how lossy

[6] Computer Music, "Dada Life Sausage Fattener Review," musicradar, August 22, 2011, https://www.musicradar.com/reviews/tech/dada-life-sausage-fattener-490089; "FL Studio's Soundgoodizer Explained – How It Works," Producer Feed, August 2, 2019, https://www.producerfeed.com/2019/08/fl-studios-soundgoodizer-explained-how-it-works.html; Sam Inglis, "Waves OneKnob," Sound on Sound, June 2011, https://www.soundonsound.com/reviews/waves-oneknob.

compression can be used as a humorous "low-quality" effect.[7] It is plausible that the same type of musician who would seek out a simple plugin to make their song sound "fatter" might also want an equally simple plugin that achieves the low-quality, underwater sound of lossy compression.

These two directions served as the basis for the two plugins I made for this thesis. My plugin Empy gives the user a high amount of control over the process of lossy distortion. My plugin Fish is a one-knob plugin inspired by the lossy distortion of "Hey Ya Low Quality."

Although there are many types of lossy audio compression, this thesis will focus on lossy compression algorithms for files, specifically the MP3 algorithm. Phone and videoconference calls use lossy compression as well, and these voice over internet protocol (VoIP) codecs have their own history, but are simply outside of the scope of this thesis. As their name suggests, VoIP codecs are optimized for the human voice, and this limitation allows them to use further methods of compression than codecs such as MP3 that are built for a wider range of audio inputs.[8] Due to these differences, VoIP codecs are not explored; the scope of this thesis, which is focused on lossy distortion of music as effect.

[7] Bizzymcbob, *Hey Ya! Low Quality*.
[8] Speech coding algorithms also use linear predictive coding, which is able to compress the voice to a great extent by effectively simulating the vocal tract to synthesize speech. Erkelsen's *Autoregressive Modelling for Speech Coding* offers an introduction to the early techniques for speech coding. K Sairam and KJS Lorraine, "Design of Speech Codec for VoIP Applications," *International Journal of Scientific Engineering and Technology Research* 4, no. 27 (2015): 5350–55; Johan S Erkelens, *Autoregressive Modelling for Speech Coding: Estimation, Interpolation and Quatisation* (Delft University Press, 1996).

Lossy Distortion

Since its development in the 1990s, lossy compression has played a large role in consumer audio. MP3 compression can reduce the size of audio files more than tenfold, with only a minor impact in sound quality. In a 1999 article, journalist Matt Lake wrote that this high amount of compression enabled easier sharing and downloading of music, resulting in "a tremendous impact on almost all areas of music."[9] That same year, "MP3" famously surpassed "sex" as the most searched for word on the internet, according to searchterms.com,[10] highlighting the immense popularity of the format.

At lower bitrates, lossy compression imparts a recognizable sound on audio, which this document refers to as "lossy distortion." Two of the most noticeable characteristics of lossy compression are an underwater sound and a muffled sound.[11] The muffled sound, known as spectral clipping, is a consequence of lossy compression algorithms getting rid of high frequency information, and placing "all available bits to the low-frequency (LF) part, which is more relevant for human hearing."[12] If audio compression algorithms include more of the higher frequencies while still compressing the audio heavily, the result is the underwater sound. This sound, also called a "birdie," is the result of removing all sound from a range of frequencies, resulting in a valleys in the spectrum of the signal.[13] The peaks that are left behind between the valleys are

[9] Matt Lake, "The MP3 Revolution," *Sound & Vision*, December 1999, 126.
[10] "Searchterms.Com," April 23, 1999, https://web.archive.org/web/19990423152020/http://www.searchterms.com/.
[11] Chi-Min Liu, Han-Wen Hsu, and Wen-Chieh Lee, "Compression Artifacts in Perceptual Audio Coding," *IEEE Transactions on Audio, Speech, and Language Processing* 16, no. 4 (2008): 681.
[12] Liu, Hsu, and Lee, 682.
[13] Liu, Hsu, and Lee, 681.

reconstructed as sine tones; these tonal beeps end up sounding like bubbling in the upper register.

Another artifact of lossy distortion is pre-delay. Lossy distortion algorithms break the sound up into blocks, which are transformed to the frequency domain, and then later transformed back when decoded. If there is a sharp attack at the end of a block, some of the energy from that attack will leak into the earlier part of the block, smearing the attack and adding noise directly before the attack in time.[14] As a result, transients get softer and less distinct, especially on signals with quick attacks, such as castanets, as seen in figure 1.

[14] Jayaraman Jayaraman Thiagarajan and Andreas Spanias, *Analysis of the MPEG-1 Layer III (MP3) Algorithm Using MATLAB*, ed. Andreas Spanias, Synthesis Lectures on Algorithms and Software in Engineering ; #9 (San Rafael, Calif.: Morgan & Claypool Publishers, 2012), 40.

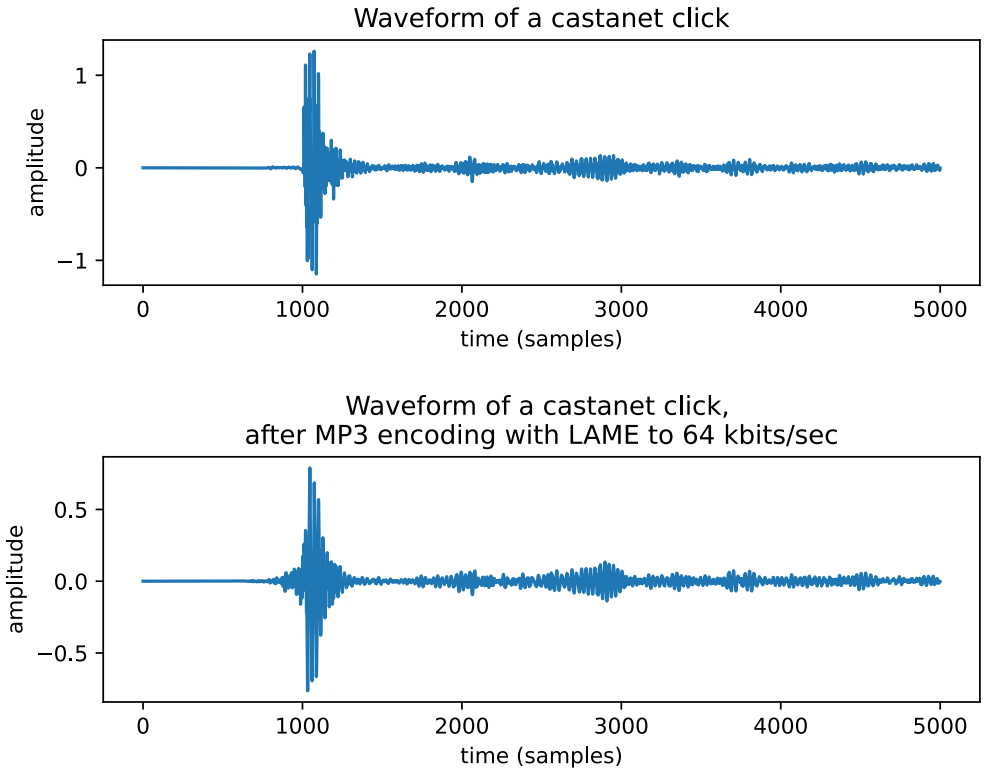

Figure 1: Waveform of a castanet click, with and without MP3 compression.

Note how the uncompressed signal is almost silent before the transient at 1000 samples, whereas the compressed signal has "predelay" immediately before, which gives the transient a less sharp sound.

Lossy distortion changes the emotional characteristics of music. In a listening test of 20 undergraduate students at Hong Kong University of Science and Technology, researchers found that MP3 compression made solo music by a number of tonal instruments sound less happy, heroic, and calm, and more mysterious, scary, and sad, especially at very high levels of compression (32 Kbps).[15]

[15] Ronald Mo et al., "The Effects of MP3 Compression on Perceived Emotional Characteristics in Musical Instruments," *Journal of the Audio Engineering Society* 64, no. 11 (2016): 858–67.

Compared with other types of distortion, the choice of input audio affects the tone of lossy distortion in nontrivial ways. One listening test in the year 2000 found that Pearl Jam's song "Daughter," despite being "unstressful pop music that should be easy to encode well," sounded more noticeably compressed than other audio sources. The intro of "Daughter" is relatively sparse, with a guitar panned left and some percussion panned right. As an article on the listening test explains, a tonal signal such as a guitar and a non-tonal signal such as percussion "require very different behaviors from a codec, and having them occur simultaneously leads to artifacts that are more clearly audible here than on most music."[16]

Both MP3 file compression and analog audio processing can introduce subtle distortions to the sound, yet musicians and audio experts generally have much more positive opinions on analog sound. Many musicians use analog gear out of a belief that it adds imperfections to the sound that gives a positive emotional reaction in listeners, whereas lossy distortion is often looked down upon by audio professionals.[17] Since the rise of digital recording and MP3 compression, members of the music industry have criticized the lifelessness of MP3 compression. These arguments are often muddled: for instance, one well-known film criticizing MP3 compression conflates MP3 compression with the unrelated (though also commonly used) technique of dynamic range compression, using harsh-sounding examples of the latter as evidence of the shortcomings of the former.[18] Scholars have suggested that these criticisms of MP3 compression may be less about aesthetic preference, and more about asserting "audio

[16] David Ranada, "Download Showdown II," *Sound & Vision*, September 2000, 118.
[17] Trevor Pinch and David Reinecke, "Technostalgia: How Old Gear Lives on in New Music," in *Sound Souvenirs* (Amsterdam University Press, 2009), 152.
[18] Jacob Rosenberg, "The Distortion of Sound," 2014.

capital." In other words, MP3 distortion represents a low-brow sound. Having the high-quality equipment and well-trained ears needed to perceive this distortion is a way of asserting capital and status.[19] Many critiques of MP3s focus on the ways that the algorithms remove parts of the sound. This contrasts with discussions of analog sound transmission methods like vinyl, where people are more likely to talk about sound or warmth being added. This is rooted in the algorithm of the MP3, in which parts of the sound that humans are psychoacoustically not able to perceive are removed, to allow for a smaller file size.[20] In 1999, Journalist Corey Greenberg wryly commented that "professional presentation givers" warned that MP3 would "throw an entire continent's worth of folks out of work," from record labels to A/V magazines, because the "'kids today' don't care too much about issues like sound quality, being mostly boombox proles rather than sound-conscious audiophiles."[21] The criticism of MP3 was also due to its association with music piracy. Journalist Stephen Witt observes "the people searching 'mp3' [in 1999, when it was the most popular search term] were looking for free, pirated music, a concept for which the term 'mp3' was now synecdoche."[22] In this way, the MP3 format was a disruptor of audio capital in more ways than one: MP3 was a low-brow, yet highly popular, audio format that did not meet the standards of audiophiles, and was also a vehicle for disruption in the music industry, in the form of wide-spread piracy.

[19] Pat O'Grady, "Rethinking Criticism about Lossy Compression: Sound Fidelity, Large-Scale Production and Audio Capital in Pop Music," *Convergence (London, England)* 27, no. 4 (2021): 1075–91, https://doi.org/10.1177/1354856520976454.
[20] Jonathan Sterne, "The MP3 as Cultural Artifact," *New Media & Society* 8, no. 5 (2006): 825–42.
[21] Corey Greenberg, "Free to Be, MP3," *Sound & Vision*, June 1999, 40.
[22] Stephen Witt, *How Music Got Free: The End of an Industry, the Turn of the Century, and the Patient Zero of Piracy* (Viking, 2015), 130.

With increasing sound quality in how people listen to music, lossy compression may become even more consigned to nostalgia in future years. As far back as 2010, Musicologist Arved Ashby theorized that because "bandwidth is increasing over time while dimensions of musical utterances are not, one might wonder if MP3 is in fact a temporary audio encoding standard—a kind of stopgap measure that will become obsolete."[23] Ashby's prediction may be coming true. At the time of writing, popular streaming services such as Apple Music and Spotify are just starting to introduce lossless audio streaming.[24] With lossy distortion less commonly found as an encoding artifact in the music people listen to, it becomes more of a figment of the past, similar to tape distortion, opening it up to a more nostalgic role as a musical effect. The complex social role of the MP3 means that the use of lossy distortion as a musical effect can make a complex and politically charged statement.

Distortion in Music

For decades, musicians have used audio degradation to add meaning and emotion to their music. For instance, in William Basinski's piece "The Disintegration Loops," decaying tape loops are used as a meditation on 9/11. Through this music, one author argues, the listener experiences the destruction up close, toeing the line between protagonist and witness, while still being out of danger.[25] Many genres, such as punk

[23] Arved Ashby, *Absolute Music, Mechanical Reproduction* (Berkeley and Los Angeles, California: University of California Press, 2010), 164.
[24] Ben Patterson, "Spotify HiFi Release Date: When Is Spotify's Lossless Tier Coming?," *TechHive*, October 14, 2022, https://www.techhive.com/article/790882/spotify-hifi-release-date-when-is-spotifys-lossless-tier-coming.html; "About Lossless Audio in Apple Music," Apple, October 25, 2021, https://support.apple.com/en-us/HT212183.
[25] Ellis Jones, "The Slow Sublime and 9/11: Insecurity and Fear in William Basinski's the Disintegration Loops," *Music & Politics* VIII, no. 1 (2014).

and industrial music, use distortion as an emotional representation of the fallibility or corruption of social systems.[26] By using sounds associated with mechanical failure, musicians are able to draw parallels with other events and ideas for emotional effect.

Musical distortion often exaggerates, or otherwise draws from, actual sources of distortion in the amplification, recording, or distribution of music. Throughout the latter half of the 20th century, the technologies that defined the sound of popular music were not the cutting-edge technologies, but were often more outdated ones that working-class musicians could actually afford. Barlindhaug observes that the "growth in popular electronic music is to a great extent a result of experimenting with cheap and outdated technology," such as Roland drum machines in the late 1980s.[27] Since these drum machines were rarely used in popular music when they were released, the use of them after the fact is less likely to be because of nostalgia, but instead because of the errors that made these tools desirable compared to more recent digital drum machines.[28] Likewise, amplifier distortion can increase the lyricality and emotional power of guitar music,[29] despite its mechanical roots.

The use of (simulated) technological failure for creative effect is particularly strong in the art and music movement of vaporwave. Vaporwave artists repurpose symbols of the digital age from the 1980s and 1990s, "pushing us to nostalgically recall

[26] Paul Hegarty, *Noise/Music: A History* (New York: Continuum, 2007).
[27] Gaute Barlindhaug, "Analog Sound in the Age of Digital Tools: The Story of the Failure of Digital Technology," in *A Document (Re)Turn*, ed. R. Skare & L. Windfield L. & A. Varheim (Peter Lang Ltd, 2007), 90.
[28] Barlindhaug, 80.
[29] Jan-Peter Herbst, "Shredding, Tapping and Sweeping: Effects of Guitar Distortion on Playability and Expressiveness in Rock and Metal Solos," *Metal Music Studies*, 2017.

our former faith in the digital utopia."[30] In this way, the vaporwave movement is less concerned with taking a clear stance for or against capitalist utopian thought. Instead, it explores "hyperreality," or a representation of something that is not just unreal, but in some way more than real. In vaporwave art, "elements from the Windows 95 interface and analog-like visual glitches engulfs the viewer in familiar, yet distant; sentimental, yet melancholic; celebratory, yet dystopian affective atmosphere of hyperreal past, highlighting how much affective content these symbols still do have."[31] These glitches do not directly accurately represent real technological glitches from the past, but exaggerate them in such a way that they are more affecting than if they were accurate.

Sound distortion is also used to inspire nostalgia. Cultural theorist Svetlana Boym described forms of nostalgia for a home left behind, either because of immigration or political change. Boym describes reflective nostalgia, a form of nostalgia in which there is no way to return to the times that people feel nostalgia for, and so the nostalgia can take on a detached, ironic effect, aware of the separation between past and present, but attempting to tell a story of their connection. Reflective nostalgia can make communities aware of their collective memories, that are only discovered through mourning and reflection.[32] These theories can be applied to changing technologies: specifically, the concept of reflective nostalgia has connections to the way sound distortion is used for nostalgic effect. This use of technological nostalgia occurs most notably in the internet art and music genre vaporwave. Vaporwave exploits glitches reminiscent of VHS tapes and early computer interfaces, to

[30] Gytis Dovydaitis, "Celebration of the Hyperreal Nostalgia: Categorization and Analysis of Visual Vaporwave Artefacts," *Menos Istorija Ir Kritika (Spausdinta)* 17, no. 1 (2021): 117.
[31] Dovydaitis, 127.
[32] Svetlana Boym, "The Future of Nostalgia" (New York: Basic Books, 2001).

create an ironic glorification of, and nostalgia for, consumerism of the 1980s and '90s.[33] By using distortions that call to mind previously cutting-edge consumer technologies, vaporwave is able to use the distortion itself to highlight this dual critique and nostalgia of consumerism.[34] We see this mix of ironic detachment and nostalgia for a simpler but imperfect past in the next section, in commenters' reactions to the lossy distortion of the video "Hey Ya Low Quality," as a disparate group of people experience collective nostalgia for lossy distortion.

In summary, musical distortion can be used for emotional effect, including for nostalgic connection to the past. As seen in the example of Roland drum machines, this nostalgia is can be for the imperfections of affordable and accessible tools. MP3 compression is also a format associated with low-cost, low-brow music production, which imparts particular imperfections to the sound, placing it in a position to potentially have a similar nostalgic role as these drum machines. Crucially, an accurate modelling of how that distortion would have sounded is not what's important for this nostalgic remembering, as this remembering is more grounded in hyperreality.

Lossy Distortion as a Musical Effect

As lossy distortion ages, it has potential to become widely used as a musical effect. In 2015, Jonathan Sterne wrote that "it is too early to tell whether some genres of music will be especially attuned to the sound of the MP3… But many other sonic artifacts have become central aspects of the sounds normally associated with particular

[33] Dovydaitis, "Celebration of the Hyperreal Nostalgia: Categorization and Analysis of Visual Vaporwave Artefacts."
[34] Ross Cole, "Vaporwave Aesthetics: Internet Nostalgia and the Utopian Impulse," *ASAP Journal* 5, no. 2 (2020): 297–326.

technologies or genres of music," such as overloaded analog sounds that have become standard in many genres.[35] As Brian Eno predicted in 1994, "whatever you now find weird, ugly, uncomfortable and nasty about a new medium will surely become its signature. CD distortion, the jitteriness of digital video, the crap sound of 8-bit— all these will be cherished and emulated as soon as they can be avoided. It's the sound of failure: so much of modern art is the sound of things going out of control, of a medium pushing to its limits and breaking apart."[36] What Eno predicted about these digital mediums can be applied to musical applications of lossy distortion as well. Although there are relatively few pieces of music that use lossy distortion, the examples of its use in music are varied and interesting.

Ryan Patrick Maguire's project "The Ghost in the MP3" explores the sounds in a song MP3 compression erases. In his track "moDernisT," Maguire remixes Suzanne Vega's song "Tom's Diner," which was used extensively for testing during the development of the MP3 codec. Instead of simply applying MP3 compression to the song, Maguire isolates the sounds that are removed during MP3 compression.[37] The result is a haunting track of skittering whispers and ringing tones. Knowing the context of its creation adds meaning to the song: this beautiful track is made of sounds deemed unnecessary: what is the damage to a song when these sounds are left out?

Lossy distortion, along with heavy video compression, is a staple of online memes. One particularly relevant example is the YouTube video "Hey Ya! Low

[35] Jonathan Sterne, *MP3: The Meaning of a Format* (Durham: Duke University Press, 2012), 234.
[36] Brian Eno, *A Year with Swollen Appendices* (Faber and Faber Ltd., 1996), 283.
[37] Ryan Maguire, "The Ghost in the MP3," 2014, https://www.theghostinthemp3.com/theghostinthemp3.html.

quality."[38] As the title suggests, this audio of this video is the Outkast song "Hey Ya!" with a high amount of lossy distortion. The audio plays over an animation of a fish rotating over a white background, with a high amount of video compression.

The reaction to the video is almost more fascinating than the video itself. The video is quite popular, with over 3 million views on YouTube in November 2022. The video also has thousands of comments, many of which discuss, with great specificity, the way the song made the commenter feel. Using some of the most upvoted comments as a case study, this comment section can offer some insight into the way lossy distortion in internet memes makes people feel. Some commenters mention that the video gives them nostalgia for lossy compression itself. One commenter, aerorain, writes that "I didn't have wifi as a kid in the late 2000s so I would go record music from the tv and the replay would sound like this. The quality sounds so strangely nostalgic to me now lol." Another, Brandon, writes that "This sounds like when we used to sneak earphones up the inside of our sleeves so we could rest our head on our hand and listen to music from our 40 song MP3 player without the teacher knowing." Yet another, Bobrian Fo, compares the sound to being on hold with customer service. Other commenters compared the song to other forms of low-quality audio. As Birdacorn describes "THIS SOUNDS LIKE WHEN THEY PLAY A SONG OVER THE SPEAKERS ON THE LAST DAY OF SCHOOL." *KaoruSimpz!* writes that this song "feels like when you're in the car with the windows up at a stop light and that one car that always blasts their music is right next to you." Part of this may be because of the aggressive lowpass filter: as shown in figure 2, the track has very little energy about

[38] Bizzymcbob, *Hey Ya! Low Quality*.

2000 Hz. Low-frequency sounds pass through walls much easier than high-frequency sounds, and low-pass filtering audio is a common way to emulate distant, muffled sounds. Like the vaporwave art discussed in an earlier section, this distortion in this track brought up nostalgia for listeners.

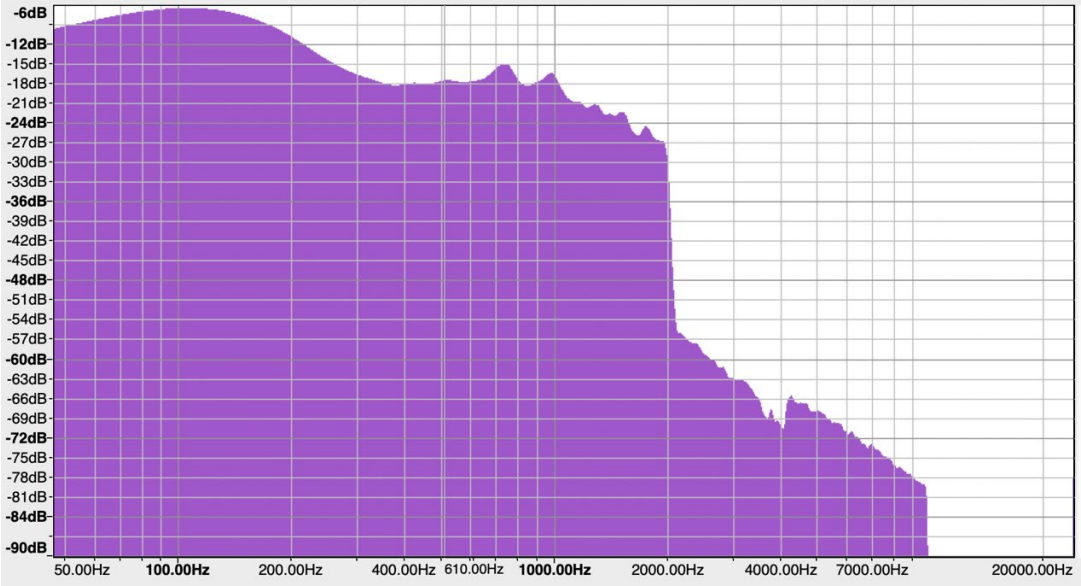

Figure 2: Spectrum graph of "Hey Ya! Low quality"

This graph, generated using Audacity, shows the amplitude in dB of each frequency in Hz of "Hey Ya! Low quality." Note the steep cutoff at 2000 Hz.

Some of the comments, on the video, however, described their reaction to the song in less literal ways. SnippSnappDoggo writes that "Something about this is comforting to a degree I can't fathom. Listening to this is like.. a long hug in a dimly lit room. Sitting on a couch, on the edge of sleep. You are safe, you are loved." Crispay described it more simply: "Why is this making me feel emotionally nostalgic." These reactions are a far cry from the results of the listening test on the impact of MP3 compression on the emotional experience of music discussed earlier. Far from turning calm music into scary music as suggested by Mo et al., these comments show that a

form of lossy distortion can contribute positive emotions to music for some listeners. This suggests that the emotional effects of lossy distortion are likely dependent on the cultural context of lossy compression as well as the specific tone of distortion used.

In recent years lossy distortion has been used as an audio effect in popular music as well. Maguire's work was a source of inspiration for the audio plugin Lossy, by the company Goodhertz.[39] This plugin is one of the few existing plugins for lossy distortion; more are discussed in a later section. According to Goodhertz, Lossy was used in songs by Phoebe Bridgers, Charli XCX, and Mahalia, all well-known artists.[40] Whereas Maguire engages with lossy distortion directly in "The Ghost in the MP3," the songs that use Goodhertz Lossy place lossy distortion in a backseat role, creating flowing, slightly alien textures behind the main elements.

A related form of distortion used for musical effect is the sound of packet loss, commonly heard in phone calls breaking up. Any form of real-time streaming audio, from phone calls to videoconferences, requires packets of audio to be sent over a potentially unreliable network. Systems for streaming audio generally implement error concealment techniques for responding to lost packets. One error concealment technique is to replace lost packets with silence. Although it is not very effective at concealing lost packets, this was historically a widespread technique, because of its simplicity.41 Another simple technique is packet repetition, used in the Global System for Mobile Communications standard for cellular networks. When errors are concealed

[39] Ryan Maguire, "An Aesthetic of the Irreducible" (PhD Thesis, University of Virginia, 2019), 62.
[40] "Lossy," Goodhertz, n.d., https://manuals.goodhertz.com/3.6/lossy/. The specific songs that use their plugin are "Whatever Simon Says" by Mahalia, "Move Me" by Charli XCX, and "Savior Complex" by Phoebe Bridgers.
[41] Colin Perkins, Orion Hodson, and Vicky Hardman, "A Survey of Packet Loss Recovery Techniques for Streaming Audio," *IEEE Network* 12, no. 5 (1998): 45.

using repetition, a small segment of audio from before the loss is repeated until new packets are received.[42]

Both of these forms of error concealment are used as a musical effect, to emulate a phone call breaking up. For a couple popular examples, this is used in Lady Gaga's "Telephone" and Charli XCX's "Lucky." In "Lucky," Charli sings "Call you, you got no reception/ You're breaking up." The poor reception is illustrated in the production on the phrase "breaking up:" there are small gaps of silence in the vocal track, as if those packets of data were lost because of the poor reception and replaced with silence. In "Telephone," Gaga sings "I have got no service/ In the club, you see, see/ Wha-wha-what did you say?/ Oh, you're breaking up on me." On the third "what," the vocals "stick" to a small segment of the syllable, playing a segment on loop, as if the phone used a particularly glaring version of the repetition error concealment strategy.

How Lossy Compression Works

There are many different lossy compression formats, devoted to different purposes in telecommunications and file compression. However, this project focuses on the MP3 compression algorithm specifically. This is primarily because of its popularity: as discussed above, MP3s have been a touchstone of the audio industry for over two

[42] Perkins, Hodson, and Hardman, 45. The Global System for Mobile Communications algorithm modifies the repetition strategy slightly: as Perkins Hodson and Hardman write, it "advocates the repetition of the first 20 ms with the same amplitude followed by fading the repeated signal to zero amplitude over the next 320 ms."

decades. For instance, the steps in the Ogg Vorbis codec's encoding algorithm are roughly the same to the steps described here.[43]

The MP3 encoder works on short frames of sound, each a fraction of a second long. These frames overlap, to avoid discontinuities at the frame boundaries. Each frame is transformed from the time domain to the frequency domain, where it can be compressed.

Next, it uses a psychoacoustic model to remove components of the sound deemed less important to the listening experience. Given the sound at a certain moment, the psychoacoustic model calculates thresholds, where if an audio signal at a certain frequency is quieter than the threshold, the average listener will not be able to hear it.[44] Different frequencies will have different thresholds.

This threshold is based on the two thresholds: the absolute threshold and the masking threshold. The absolute threshold "characterizes the amount of energy needed in a pure tone such that it can be detected by a listener in a noiseless environment."[45] Since music and speech consists of more than just a pure sine tone,[46] the absolute threshold alone cannot determine the full set of inaudible frequencies at a point in a track. The MP3 algorithm must also take into account masking, or the "process where one sound is rendered inaudible because of the presence of another sound."[47] Sounds are able to mask other sounds that are close to them in frequency; this masking effect diminishes when tones are further apart. A sound can also mask another sound even if

[43] "Vorbis," Wiki, Hydrogenaudio Knowledgebase, n.d., https://wiki.hydrogenaud.io/index.php?title=Vorbis.
[44] Thiagarajan and Spanias, *Analysis of the MPEG-1 Layer III (MP3) Algorithm Using MATLAB*, 25.
[45] Thiagarajan and Spanias, 5.
[46] The works of Sachiko M are a possible exception.
[47] Thiagarajan and Spanias, *Analysis of the MPEG-1 Layer III (MP3) Algorithm Using MATLAB*, 6.

they do not happen at the same time: a loud sound can mask a quieter sound at a similar frequency up to 200 milliseconds after the end of the loud sound, or even 50 milliseconds before it starts.[48]

The MP3 algorithm uses these masking thresholds to determine the amount of quantization permissible at each frequency in a frame. At higher amounts of quantization, the amplitude at a frequency requires fewer bits to encode, but this quantization introduces noise, by changing the amplitude slightly. The amount of noise that is acceptable is dependent on the threshold: if the signal at a frequency is below its threshold, quantization can erase it completely.[49] MP3 encoders quantize each frame of audio with nested loops, attempting to find amounts of quantization for each frequency that will minimize audible distortion, while keeping the number of bits used for that frame within the budgeted bitrate.[50]

Although this basic process is applied across encoders, different MP3 encoders implement the encoding algorithm in slightly different ways, resulting in a variety of sounds. The MP3 standard is not completely normative: while the MP3 standard clearly defines the format of an MP3 file and how to decode it, the exact design of the encoder is left up to the implementer.[51] The graphs in figure 3 demonstrate MP3 compression using two encoders, plus a commercial plugin (Goodhertz Lossy[52]) that emulates lossy compression, and the two plugins that I made for this project. The first encoder

[48] Thiagarajan and Spanias, 8.
[49] Karlheinz Brandenburg, "MP3 and AAC Explained," in *Audio Engineering Society Conference: 17th International Conference: High-Quality Audio Coding* (Audio Engineering Society, 1999), 5.
[50] Thiagarajan and Spanias, *Analysis of the MPEG-1 Layer III (MP3) Algorithm Using MATLAB*, 61.
[51] Brandenburg, "MP3 and AAC Explained," 4.
[52] "Lossy." This plugin is discussed at greater length in a later section.

demonstrated, Blade,[53] was released from 1998 to 2001. (The version used in this demonstration was released in 2001.[54]) In a guide to the MP3 format from 2000, Scot Hacker described Blade as a "legendary" encoder that was "once hailed as being one of the best quality encoders available," although, in the rapidly changing landscape, it was already being outperformed by the time that book was published.[55] The other, LAME, had its latest release in 2017, which is what I used for this demonstration.[56] As shown in figure 3, all of these pieces of software remove much of the quieter parts of the sound, including much of the high frequency.

When sounds are loud enough (for instance, during a drum hit), more frequencies pass above the threshold, and are included in the sound. At other times, when things are quieter, more of the frequency spectrum is silent. LAME goes even farther, with a hard cutoff around 8 kHz, opting for more of the muffled sound of spectral distortion than the bubbly sound of spectral valleys. Of the three encoders, the audio encoded by Blade has the messiest sound: the transients are far less crisp, and it adds a lot of tonal bubbly noises that were not present in the original. This can be seen a bit in the graphs: Blade's output has much more high frequency information in the quieter parts of the track, between the drum hits, while the Lossy output only uses that high frequency range during the transients, resulting in a cleaner sound. Empy also prioritizes lower frequencies, and (at least in the settings for this spectrogram) distorts the low end more than the other programs do.

[53] The Blade MP3 encoder is also sometimes called BladeEnc.
[54] Tord Jansson, "Blade MP3 Encoder," C, March 7, 2001.
[55] Hacker, *MP3: The Definitive Guide*, 178.
[56] "History," lame.cvs.sourceforge.net, n.d., https://archive.ph/M0Yvq.

The spectrogram of Fish's output has a strange mirage-like pattern between 8kHz and 16kHz; this is the result of aliasing from the downsampling process, as is discussed in a later section.

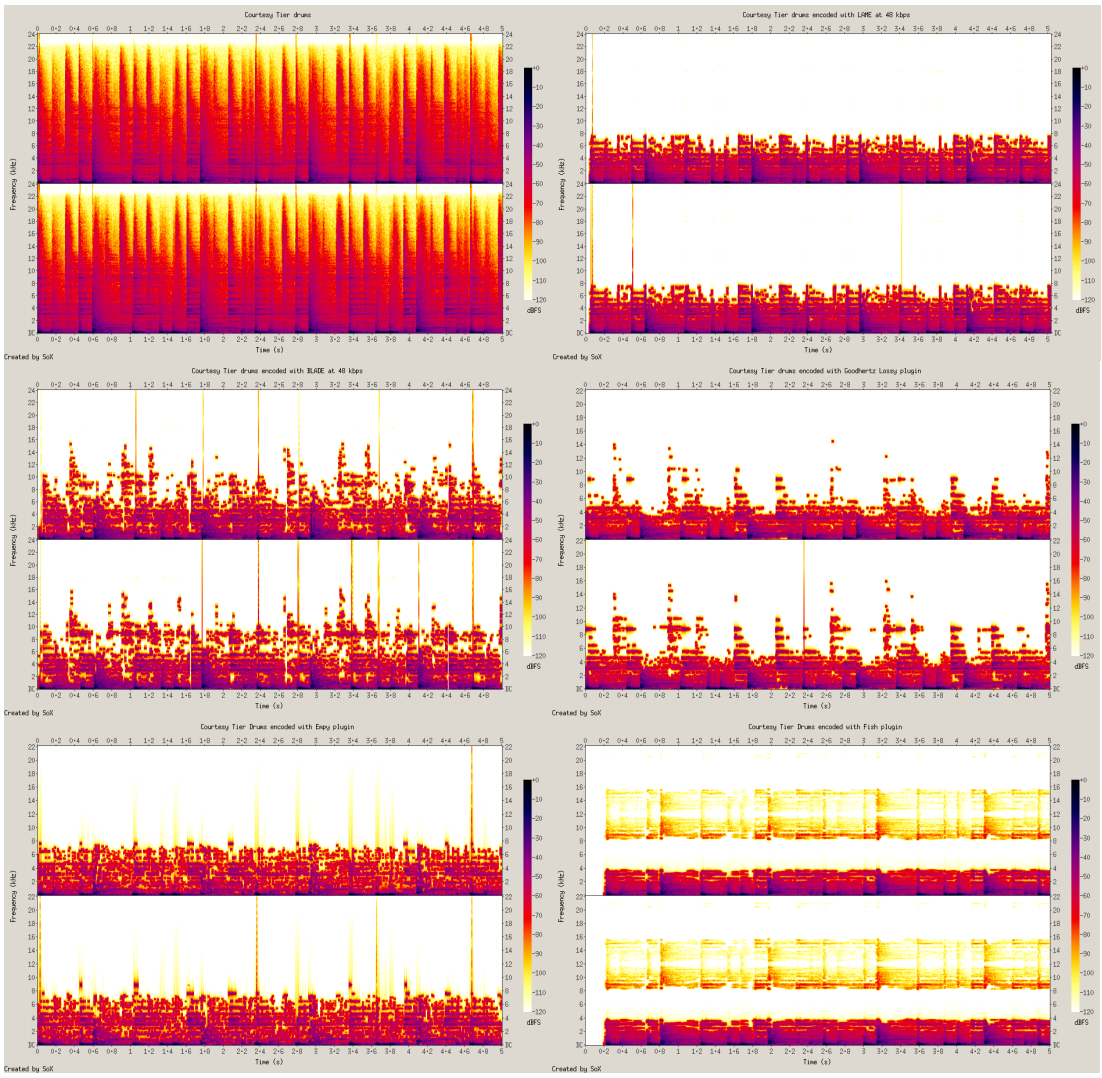

Figure 3: Comparison of MP3 encoding algorithms on a drum track

These three spectrograms show the effect of MP3 compression on a five second drum track.[57] The top left graph shows the frequency content of the uncompressed track, the top right graph shows the result of MP3 compression to 48 kbps using BLADE, the middle left graph shows the result of MP3 compression, again to 48 kbps, using LAME, the bottom right graph shows the result of compression with Goodhertz's plugin Lossy, the bottom left shows the result of compression with my plugin, Empy, and the bottom right shows the result of compression using my plugin Fish. The graphs were made using the SoX command line tool.[58]

Figure 4 shows the same audio as figure 3, but plotted as a spectrum instead of a spectrogram: instead of showing the strength of each frequency at each moment in time,

the graph below shows the strength of each frequency across the entire track. From this view, we can see that the encoding algorithms erase the signal more aggressively at higher frequencies. This graph also gives some clues as to the way these different algorithms handle tonal frequencies. The 3000 Hz to 10000 Hz range of the raw drum track spectrum is marked by roughly evenly spaced peaks, coming from the ride hits. This is in contrast to the other parts of the drum track— the kick, snare, and hi-hat— that are much more noise-based. In Figure 4, we can see that these different algorithms handle these this tonal component amid a wash of noise. Here, it is particularly clear that Lossy, and to a lesser extent Empy, favor the prominent tonal components: in between these peaks, Lossy gets rid of much more of the sound, as evidenced by the deep troughs in Lossy's spectrum curve. This results in an unmistakable MP3 compression-like sound in Lossy's output, while not obscuring the timbre of the ride as much as Blade does.

The spectrum reveals that Blade and Lossy, not just LAME, devote most of their available bits to lower frequencies. Blade and Lossy's spectrum has a sharp drop-off at 16 kHz, a bandwidth restriction that is common in MP3 compression.[59] This graph shows that Fish introduces further distortion above 20000 Hz, and also that Empy applies more distortion in the lower frequencies: of the encoders compared here, Empy is the only one that noticeably changes the frequency response below about 2000 Hz.

[57] Drum track provided to the author by Jon Bellona, engineer.
[58] Chris Bagwell, "Sound EXchange (SoX)," C, February 22, 2015, https://sox.sourceforge.net/.
[59] Liu, Hsu, and Lee, "Compression Artifacts in Perceptual Audio Coding," 682.

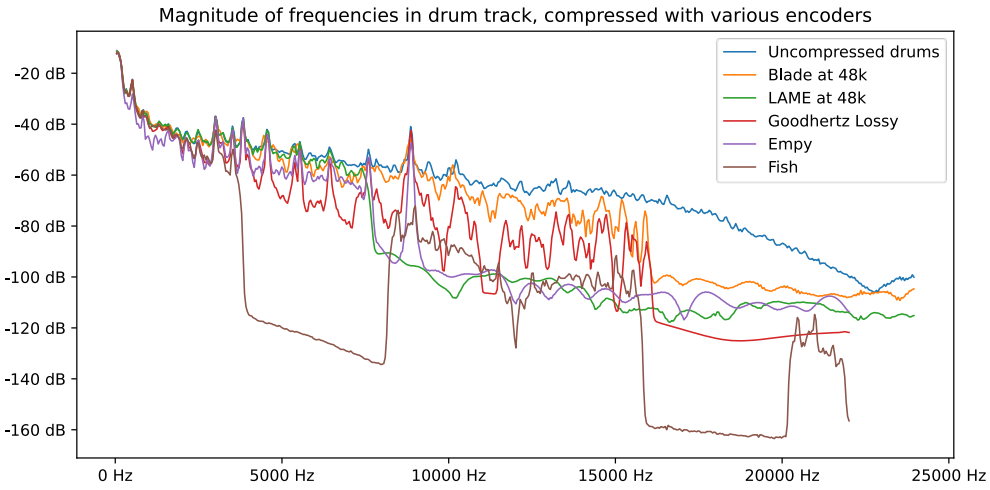

Figure 4: Magnitude of frequencies in drum track, compressed with various encoders

This graph shows the overall energy of each frequency for each of the six examples in the previous figure. Spectra calculated with Audacity, plotted using Matplotlib.

Existing Plugins for Lossy Distortion

Multiple companies have developed plugins that emulate lossy distortion. Some, such as LameVST[60] and Fraunhofer Pro-Codec,[61] accurately implement the code of lossy compression algorithms. These are aimed less as a creative effect, and more as mastering tool, to ensure that songs will sound acceptable when encoded as MP3s. Similarly, Audio Ease's Speakerphone uses "actual telecommunication industry standard audio data compression" to give the sound of a telephone call, as part of an impressive (and expensive) suite of effects.[62] Several other plugins, however, are focused towards the use of lossy distortion as a creative tool. By comparing these plugins, I got a sense of what features I wanted to include in my plugins. They also

[60] Iunusov, "LameVST," January 14, 2017, https://github.com/Iunusov/LameVST.
[61] "Fraunhofer Pro-Codec," Sonnox, accessed May 28, 2022, https://www.sonnox.com/plugin/fraunhofer-pro-codec.
[62] "Speakerphone User Guide," n.d., https://www.audioease.com/speakerphone/files/speakerphone-manual.pdf.

provided a baseline of the surprisingly varied ways how lossy distortion plugins could sound.

Goodhertz Lossy

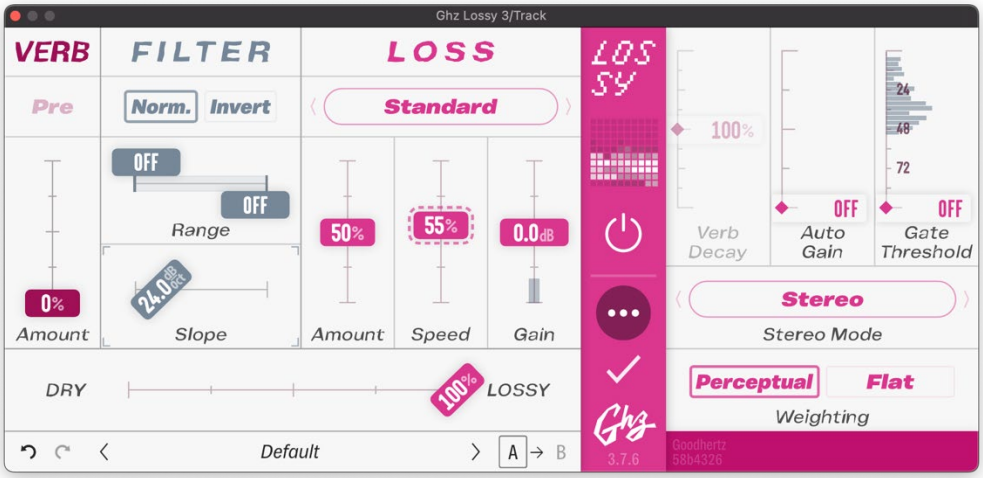

Figure 5: Goodhertz Lossy GUI

Goodhertz Lossy is a plugin devoted to emulating lossy distortion for creative use. This plugin was initially released in 2014, although further versions have been released since then. With this plugin, developers aimed for "exaggeration rather than exactitude," allowing the controls to move far past the level of distortion one would find in a real MP3 file.[63] The marketing for Lossy makes clear that it is not meant as an emulator or a testing program for existing codecs, but rather an artistic tool.[64] This is also clear when working with the plugin itself: the GUI shows loss amount as a range from 0% to 100%, rather than as a scale from high to low bitrates, as it is communicated in codecs such as MP3.

[63] Rob Stenson, "Lossy's Uncharted Waters," Goodhertz, *Tonal* (blog), April 7, 2015, https://goodhertz.com/tonal/uncharted-waters/.
[64] "Lossy."

The user of Lossy can control the speed of the lossy algorithm in addition to the loss amount. At higher speed values, Lossy's tone is garbled and bubbly; slower speed settings seem to smear the artifacts, removing the distracting bubbles of higher speed settings while keeping the same amount of loss. This speed control sounds achieves excellent sounds, and provides a level of control missing from many other plugins for lossy distortion.

Lossy features several other parameters to control the loss model. The user can change the loss model to process sound in stereo, joint stereo or mono. Additionally, the "weighting" of the loss model can be changed between flat and perceptual weighting; when this control is set to flat, the sound has a bit of a fuller low-end. The user can even set the loss model to work in reverse, leaving only frequencies that would be eliminated normally (referencing the *Ghost in the MP3* project of Ryan Maguire[65]). Lossy also has several options that emulate a packet loss from an unreliable telecommunication connection, using the recovery methods discussed in an earlier section. In one packet loss mode, lost packets are replaced with silence; in the other, they are replaced by looping the previous packet.

When switching to these other models, the same knobs that were used to adjust the characteristics of the loss— the speed, amount, and stereo mode— can still be used, even though they now control completely different algorithmic functions. For instance, when the loss model is set to standard, the speed knob controls the rate of the bubbly noises in the loss model; when the loss model is set to packet loss, the same speed knob

[65] Maguire, "An Aesthetic of the Irreducible," 62.

controls the length of each gap. Reusing the same knobs this way makes the plugin intuitive to use, achieving a lot of control without a daunting display of knobs.

About half of Lossy's GUI real-estate is taken up by pre- and post- effects. The plugin features reverb (applied either before or after the loss model), low-pass and high-pass pre-filtering, a gate and auto-gain controls, as well as the usual gain and mix controls. These effects are well-selected: sending a heavily reverberated signal through the loss model results in an ethereal, watery sound, and aggressive filtering goes a long way towards convincingly emulating low-quality MP3 or cell phone audio.

Overall, the plugin gives a wide range of usable sounds, grounded in lossy codecs and telecommunication glitches, such as packet loss.

Lese Codec

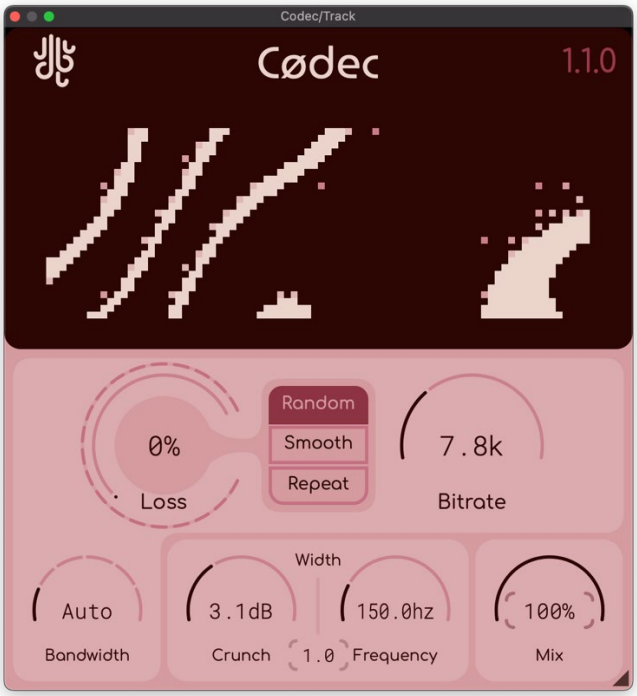

Figure 6: Lese Codec GUI.

Although Lese Codec and Goodhertz Lossy were developed with similar intentions, Lese Codec was made in a slightly different way. This plugin, released in 2022, was designed with a goal to "simulate a bad internet connection" with a higher standard of accuracy than other plugins, which, in the words of the Lese's blog, "always seemed a little bit 'off.'"[66] In an attempt to get a more accurate sound, this plugin serves as a container around the popular freely-licensed Opus audio codec.

Lese Codec has a few other controls outside of the bitrate and bandwidth of the codec. Like Lossy, this plugin features a packet loss model, and a mix percentage knob.

[66] "New Release: Codec, an Audio Plugin for Modern Degradation," *Lese* (blog), August 6, 2022, https://lese.io/blog/new-plugin-codec/.

The other section of knobs, which is unique to this plugin, is “crunch,” which lets the user apply more distortion to a specific frequency range.

The sound of this plugin’s distortion is different from that of the other plugins described here. The codec distortion of Goodhertz Lossy has a waterier, more tonal sound, while this plugin sounds grittier, and more reminiscent of a bitcrusher. This effect sounds good on vocals and drums, giving them a dirty and noisy sound. More complex instrumentals or full songs sound nearly unintelligible at lower bitrates, reduced to crunchy noise.

The bitrate knob on this plugin does not change the amount of loss as smoothly as it does on other plugins. This may be because, unlike other plugins, Codec is a wrapper on a codec that was not designed for use in a DAW, and so has no use for instantaneous, smooth gradations of quality.

Aberrant DSP Digitalis

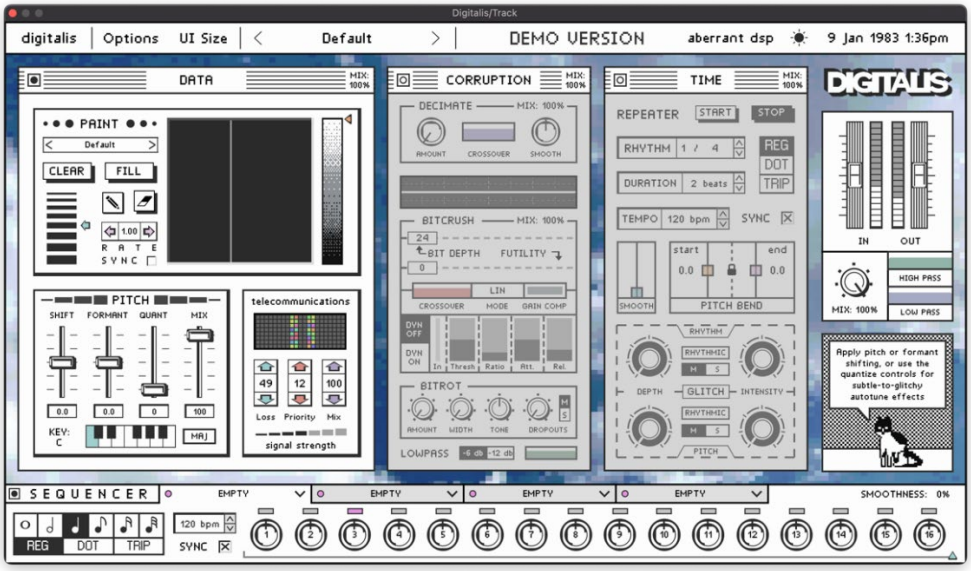

Figure 7: Aberrant DSP Digitalis GUI.

Digitalis, released in 2022 by Aberrant DSP, is a plugin that offers an extensive suite of digital distortions and glitches, including lossy distortion. The controls offered are loss, priority, and signal strength. Loss controls the amount of lossy distortion. Priority determines how much high frequency sounds are let through. Signal strength controls the coarseness of the gate: at lower signal strengths, wilder sets of bands get quantized as one.

The sound of this effect is similar to that of Lossy, but with a bit more of a metallic sound. The coarseness afforded by low signal strength sounds interesting, and gives a unique sound compared with the other plugins discussed here.

Both Digitalis and Lossy take a unique approach to metering, representing the frequency spectrum using a grid of twinkling squares rather than a more conventional spectrum graph. Each square is dark when the sound at the corresponding frequency is silenced by the gate, and light when sound is able to pass through.

The Digitalis user interface is endearing as well. More than the other plugins discussed here, it leans into the pixelated retro-computing aesthetic. A clock display at the top of the screen informs us of the current time and date… but in 1982. Like in plugins from the company Valhalla,[67] Digitalis displays tooltips of the focused knob off to the side. However, in Digitalis, the text is placed in a speech bubble coming from a pixelated cat.

One minor issue with the Digitalis user interface is that all three of the controls for lossy distortion affect the amount of distortion. This means that any adjustments to the signal strength or priority can easily result in the gate having virtually no effect, or having such an effect that no sound is able to pass through at all. This means that any changes to the quality of the sound cannot be done with just one knob. Adjustments require the user to switch back and forth between the loss knob and one of the other two knobs, balancing the movements of both.

[67] Sean Costello, "Valhalla Supermassive: The Controls," *Valhalla* (blog), May 6, 2020, https://valhalladsp.com/2020/05/06/valhallasupermassive-the-controls/.

Toneboosters BitJuggler

Figure 8: Toneboosters BitJuggler GUI.

BitJuggler, released by Toneboosters in 2010 and last updated in 2020, is a plugin for digital distortion, offering quantization and resampling reduction, as expected from a bitcrusher. One of its quantization modes, called "transform," models the effects of lossy codecs. According to BitJuggler's manual, in the transform mode, the "input signal is converted to the frequency domain, and each frequency band is quantized in accordance with the sensitivities of the human auditory system."[68] It also offers two controls not found in other lossy distortion plugins: "noise fill" and "attack." According to the manual, when noise fill is turned up, audio below the threshold is replaced not with silence, but with random noise. At higher attack settings, BitJuggler allocates more bits to the transients.

[68] "BitJuggler / BitJuggler IOS Installation and User Manual" (Tonebusters, 2020 2010), https://www.toneboosters.com/manuals/TB_BitJuggler.pdf.

Both noise fill and attack give modify the signal in interesting ways. Noise fill sounds particularly interesting on drums and vocals. At high levels of lossy distortion, drums can sound strangely tonal, and turning up noise fill gives drums a fluffier sound. When using the noise fill on vocals, it adds the a whispery quality to the vocals, reminiscent of noise-vocoded speech.[69] When the attack knob is turned up, the attacks are cleaner, while the sustained parts of the signal are still distorted. Like increasing the attack of a compressor, increasing this attack setting makes the signal more intelligible. This is particularly useful for vocals: without attack to bring out the starts of words, a high level of lossy distortion makes it difficult to tell what the singer is saying.

While lossy distortion is just a small part of this plugin, it sounds quite good, and the noise floor and attack controls are both good ideas that aren't present in other lossy distortion plugins.

[69] This similarity is likely not coincidental, since noise fill, if I understand it correctly, works as a vocoder, replacing the audio in frequency bands that are quieter than the threshold with noise bandpassed noise at the same amplitude and frequency.

Empy and Fish: Two Plugins for Lossy Distortion

To explore lossy distortion as a musical tool, I designed and built two audio plugins. It seemed impossible to capture the complicated and varied scope of lossy distortion in one coherent plugin. I built both Empy and Fish using JUCE, a library for building audio plugins in C++.[70] With Empy, I followed the basic concepts of removing masked frequencies, but I did not attempt to accurately follow the MP3 algorithm. As a result, the sound of the plugin, although it is lossy distortion, is not identical to the sound of an MP3. Empy offers the user a great deal of control, with 12 knobs and sliders to control the quality of the distortion. The other plugin, Fish, is different from Empy in almost every regard. Rather than having many knobs that expose the inner workings of the plugin, Fish has one, unlabeled knob. Because of this, it is very limited in the variety of sounds it is able to offer. Instead of giving control to the user to discover new sounds, as I did with Empy, I carefully crafted Fish to imitate the sound of the "Hey Ya Low Quality" YouTube video as closely as I could. Fish is also visually striking, with an animated image of a spinning fish taking up the entire window.

[70] Raw Material Software Limited, "JUCE," 2020, https://juce.com/.

Empy User Manual

Empy is a plugin for a wide range of lossy distortion sounds, drawing inspiration from lossy audio compression codecs like MP3. Empy can sound bubbly and metallic. It can resemble a bitcrusher. It can freeze sound at spots, like a choppy phone connection. An any setting, Empy gives sound distinctive lossy distortion.

The algorithms within Empy are much simpler than the MP3 codec. The MP3 codec loops through different encoding levels, trying to find one that fits within the bit budget and the allowable distortion. Empy is, at its core, a spectral gate. An ordinary gate works in the time domain: it uses the amplitude of each sample, passed in one at a time, to determine if the audio is loud enough to pass through the gate or not. An ordinary gate has one threshold: audio louder than the threshold passes through unchanged, while audio below the threshold is attenuated, often all the way to zero. Empy, on the other hand, works in the frequency domain.[71] Each frequency has its own gate, and different gates can have different thresholds. This means that rather than acting only on the amplitude of a sound, Empy can change the timbre of the sound, removing quieter frequencies to give the sound a watery, MP3-like quality, or biasing its gate towards high or low frequencies to filter the sound.

The gates in Empy differ from normal gates in one other important way. In Empy, there are two thresholds that a sound must pass to be allowed through the gate.

[71] Lossy distortion plugins are not the only gates to work in the frequency domain. One plugin, appropriately named Spectral Gate, works similar to Empy's static threshold with a flat curve, and even has the option to tilt the threshold towards higher or lower frequencies. Another plugin, Convoluter, works similar to Empy's dynamic threshold, masking frequencies that are close to louder frequencies. Both of these plugins, though seemingly simple, have a pleasant sound, reminiscent of lossy distortion. Andrew Reeman, "Spectral Gate," 2022, https://www.andrewreeman.com/spectralsuite/; Leighton Hargreaves, "Convoluter," Anarchy Effects (Anarchy Sound Software, 2013).

The first is the static threshold, as you would see in a traditional gate: if the signal at 400 Hz is louder than -20 dB, it passes. The second is the dynamic threshold, which changes over time to follow the amplitude of the frequencies: if the signal at 400 Hz is sufficiently loud, compared to the recent amplitudes in the neighborhood of 400 Hz, it passes the gate. A signal only passes through Empy if it is above both of these gates.

These two gates represent the absolute threshold and masking threshold of the MP3 psychoacoustic model. The static threshold models the removal by the MP3 algorithm of sounds below the absolute threshold of hearing, and the dynamic threshold does the same for the masking threshold. Unlike the MP3 codec, Empy is not an attempt to maximize sound quality while minimizing file size. Empy is a creative tool. Therefore, Empy offers further controls to shape the sound. The static threshold can be raised and lowered, and the user can also transition between a flat curve, where each frequency has the same static threshold, and a perceptual curve, which follows the absolute threshold of hearing. The user can raise and lower the dynamic threshold as well, and can change the distance between frequencies that sounds are able to mask for.

Empy also offers some controls inspired by traditional time-domain gates. The user can control the rate at which the dynamic threshold moves, which provides rough control of the attack and release of the gate. Additionally, Empy offers a ratio knob, to control how much the signal below the threshold is attenuated.

Further shaping of the sound can be achieved with the bias, quantization, frequency resolution, and stick controls. Some of these controls are my attempts to reverse engineer the controls offered by other lossy distortion plugins. For instance, Goodhertz's Lossy has a "packet repeat" mode that I wanted to implement here. Other

controls, like frequency resolution and quantization, were inspired by components of the MP3 encoding process itself. My goal was to give the user of Empy as much control as possible in designing their sound, letting the user fine-tune the parameters of packet repeat much more than they can with Lossy.

Processing the signal in the frequency domain causes Empy to have some unavoidable latency. To convert from the time domain to the frequency domain, Empy uses the Modified Discrete Cosine Transform, or MDCT. This algorithm, which similar to the Fast Fourier Transform (FFT), converts a frame of audio samples into their frequencies.[72] The MDCT uses overlapping windowed frames of audio, as shown in Figure 9, to avoid harsh discontinuities at the transition points between windows. The MDCT cannot convert a frame to the frequency domain until every sample in that frame has arrived, meaning that Empy's latency is determined by the MDCT window size. Larger window sizes are useful for encoding because of their greater frequency resolution. The Ogg Vorbis codec, a popular lossy compression algorithm that uses the MDCT, supports window sizes ranging from 64 all the way up to 8192 samples.[73] At the common sampling rate of 44100 Hz, an 8192 sample window incurs 185.8 milliseconds of latency, rendering it virtually unusable for a real-time audio plugin. AAC Low Delay, a lossy compression algorithm designed for low latency, real-time

[72] The MP3 algorithm uses the MDCT as part of a more complicated filter bank, due to complicated disputes between the MPEG standards organization and the developers of MP3. As journalist Stephen Witt wryly writes, "MPEG approached Fraunhofer with a compromise. The committee would make multiple endorsements. Fraunhofer would be included [in the MPEG standard] but only if they agreed to play by certain rules, dictated by MUSICAM. In particular, they would have to adopt a gangrenous piece of proprietary technology called a 'polyphase quadrature filter bank.' Four uglier words did not exist." Witt, *How Music Got Free: The End of an Industry, the Turn of the Century, and the Patient Zero of Piracy*, 19.

[73] *Vorbis I Specification* (Xiph.Org Foundation, 2020), 10.

applications, uses the MDCT transform with 50% overlap as well, with a 512 sample window, resulting in under 20 milliseconds of latency.[74] Drawing inspiration from AAC Low Delay, I set Empy's default window size to 512 samples, while allowing the user to change the window size using the frequency resolution drop-down menu.

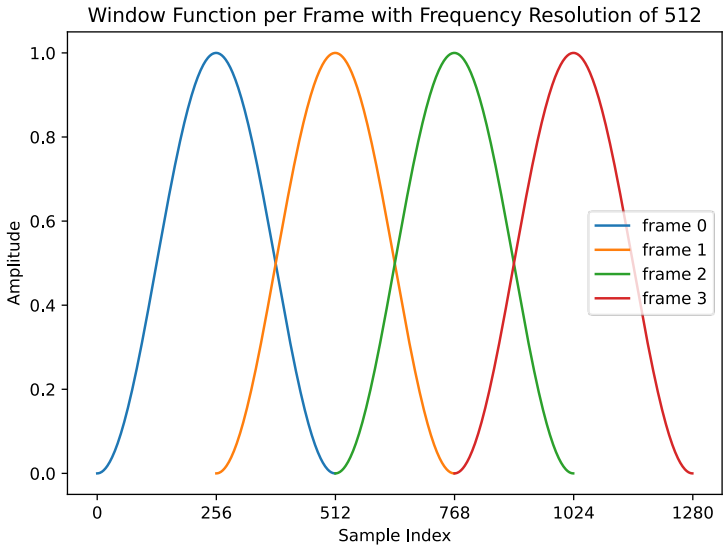

Figure 9: Window Function per Frame with Frequency Resolution of 512

[74] Ralf Geiger et al., "Structural Analysis of Low Latency Audio Coding Schemes," *Journal of The Audio Engineering Society*, 2005.

The Empy GUI

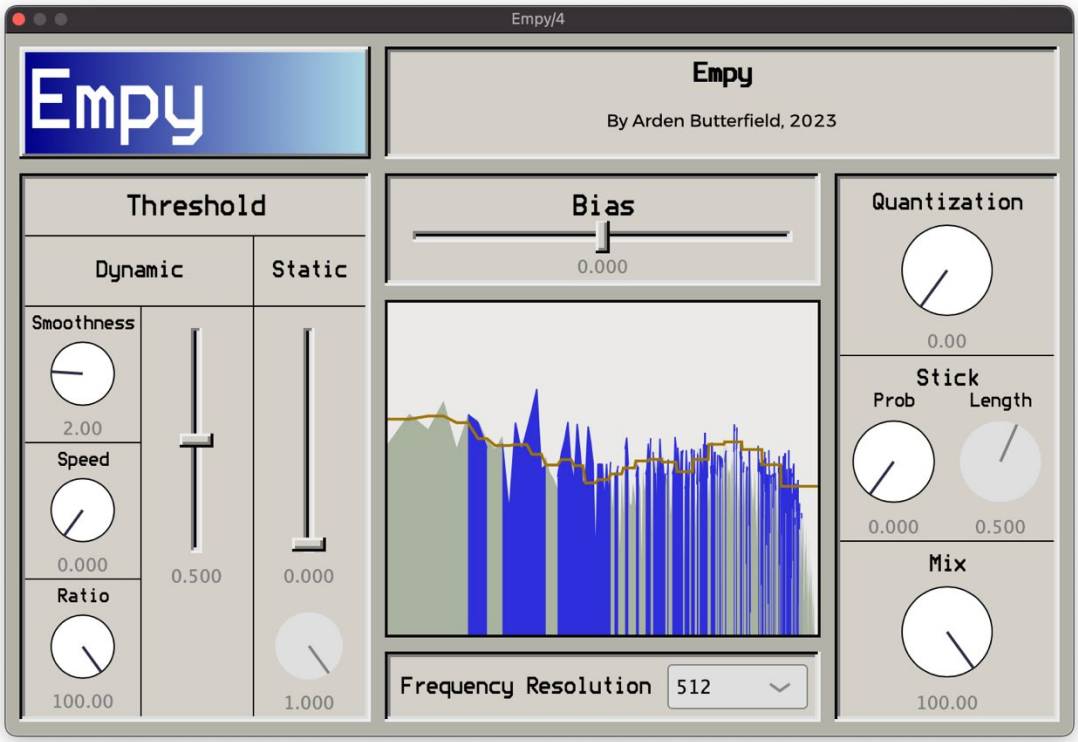

Figure 10: Empy graphical user interface (GUI).

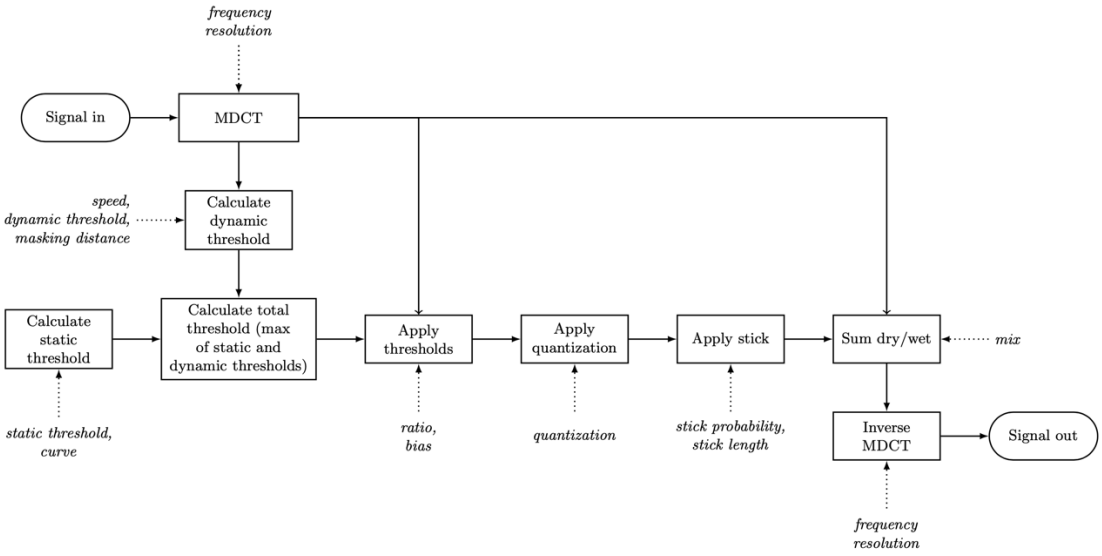

Figure 11: Empy signal flow.

The three columns of the Empy graphical user interface (GUI) roughly correspond to the signal flow through the plugin. The first column contains controls for building the dynamic and static thresholds. These thresholds then get tilted towards the low or high frequencies by the bias control in the middle column. After the thresholds are applied to the sound, the sound can be further processed by the controls on the right column, in the order they appear from top to bottom.

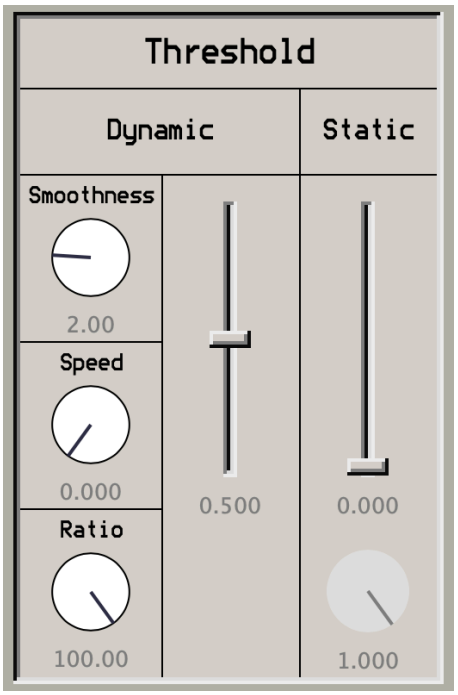

Figure 12: Left column of the Empy user interface

Threshold Panel

The left panel of the plugin controls the thresholds and their effects on the signal. To save processing power, the dynamic thresholds are calculated in bands of approximately 1 octave, hence the jagged shape of the spectrogram.

Smoothness

This knob controls how far the energy in a band will spread to calculate the dynamic threshold. Higher values will result in smoother masking, while lower values tend to bias the low frequencies slightly.

Speed

The dynamic threshold for a band is based on the root-mean-square of the amplitude, and speed controls the length of that RMS window. A value of 0 only takes the most recent frame into account, while higher values place weight on a longer window. At lower speed values, Empy has a bubblier sound, which gets smoother at higher speed values. High speed values can also give an interesting effect with attack-heavy sounds, where the attack is able to pass through without being distorted by the threshold.

Ratio

Ratio controls the strength of the gate at attenuating sounds below its threshold. With a ratio of a, a sound x decibels below the threshold will be attenuated until it is $a*x$ decibels below the threshold. When the ratio is set to a high number, the gate essentially silences frequencies below the threshold; when the ratio is set to 1, the gate has no effect.

Dynamic Threshold

This slider controls the level of the dynamic threshold, shifting it up or down by the same amount for all frequencies. At zero, no dynamic threshold is applied, so the speed and masking distance knobs have no effect.

Static Threshold

This slider controls the level of the static threshold, shifting it up or down by the same amount for all frequencies. At zero, no static threshold is applied, so the curve knob has no effect.

Curve

This curve knob interpolates between the two static threshold curves. At 0, the static threshold is flat, and at 1 the static threshold follows the absolute threshold of hearing for humans.

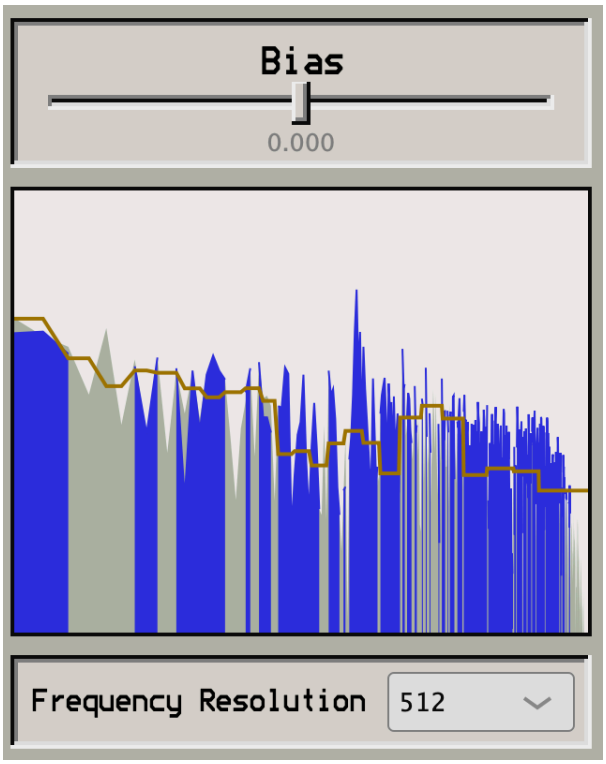

Figure 13: Middle panel of the Empy user interface.

Bias

Bias tilts the thresholds calculated in the previous section towards the low or high frequencies. At lower bias settings, more low-frequencies and less high-

frequencies will get through the threshold, and vice versa for high settings. At 0, there is no bias towards low or high frequencies. When the static and dynamic threshold knobs are both at zero, bias has no effect.

Frequency Resolution

Frequency resolution determines the number of frequency lines we use in the transform. Using more lines require a wider input window, which allows Empy to break the spectrum down into more frequencies. Higher frequency resolution results in a smoother sound, but introduces more latency. At very low frequency resolution, Empy functions more like a conventional bitcrusher, and has a harsh, crackling sound.

Figure 14: Right panel of the Empy user interface.

Quantization

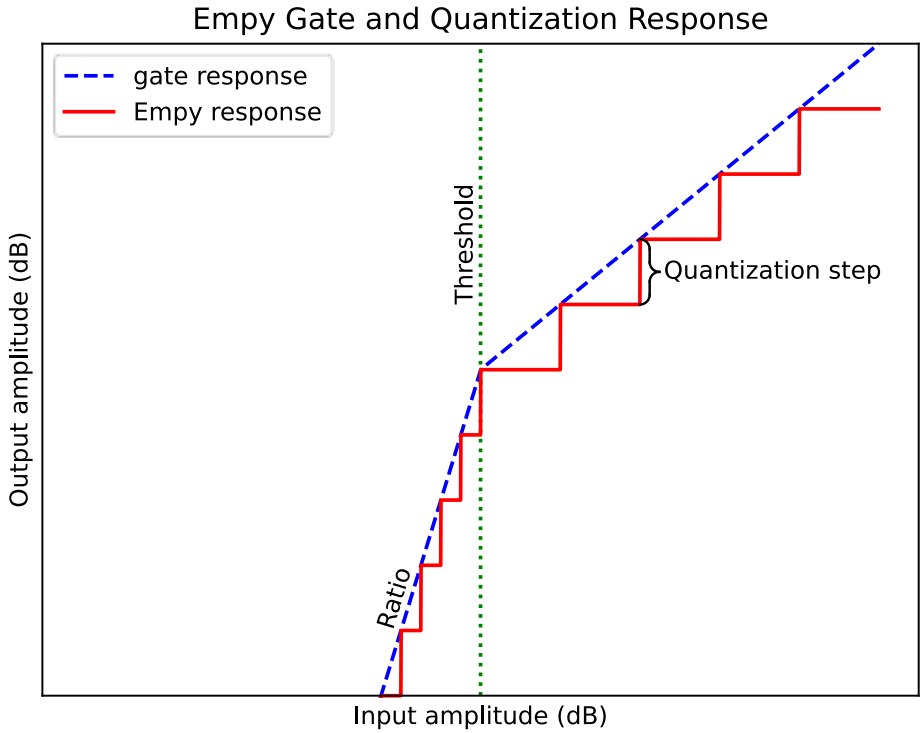

Figure 15: Empy Gate and Quantization Response

This graph shows how Empy changes the amplitude of the signal at a specific frequency, based on the ratio and quantization parameters, as well as the frequency's threshold, which is calculated from the static and dynamic thresholds. The blue dashed line is the output of the gate. Above the threshold it has a slope of 1, below the threshold it has a slope equal to the ratio. If the quantization step is non-zero, this output is then quantized, resulting in the red stair-stepped output. The rise of each stair is equal to the quantization step, set by the quantization knob.

Quantization works a bit like bit reduction in a bitcrusher, reducing the number of possible values for each amplitude and rounding the amplitudes down to the nearest option. The quantization knob sets the number of decibels between each quantization option, so higher values will change the sound more aggressively. A setting of 0 results in no quantization.

Quantization in Empy works differently from quantization in the MP3 codec. In MP3 encoding, the step between quantization options is determined using a loop, to minimize bit usage while keeping distortion under the threshold. In initial versions of Empy, I tried setting the quantization step to be dependent on the threshold. However, I preferred the sound of a set quantization step for all frequencies.

Stick

With stick, the sound can get "stuck" on a frame. This section draws inspiration from packet loss recovery algorithms in telecommunications, where a program might try to cover up a gap by stretching out the current sound. This section of the plugin has a gray "indicator light" that flashes on when the sound is currently stuck.

Probability

The proportion of time spent stuck. At 0, the sound will never get stuck, at 0.5 it will be stuck about half the time. At 1 it will always be stuck. Note that at probabilities of 1, the sound will still change, as often as is set by the length knob.

Length

The average length that the sound will stay stuck for. The exact length of each stick is random.

Mix

The balance between input and processed signal returned by the plugin.

Spectrum Graph

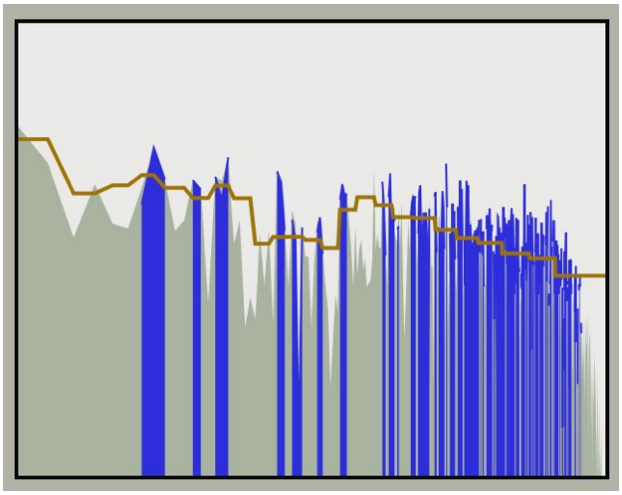

Figure 16: The Empy spectrum graph.

The spectrum graph shows what Empy is currently doing to the audio passing through it. The gray spectrum shows the amplitude at each frequency of the dry signal, and the blue spectrum shows the amplitude at each frequency of the wet signal. Lower

frequencies are on the left, higher frequencies are on the right. A taller bar corresponds to more amplitude at the corresponding frequency.

The graph also shows a brown line, which is the current total threshold, equal to the maximum of the dynamic and static thresholds. If a peak of the gray spectrum is below the brown line, that means that the corresponding frequency is below the threshold, and its amplitude will be reduced by the amount set with the ratio control.

When the mouse moves over certain controls, an orange line will appear on the spectrum graph as well. This orange line gives some information on what that control is doing to the sound.

- For smoothness, the orange line shows how far the energy in each band gets spread to the neighboring bands.
- For the speed knob and the dynamic threshold slider, the orange line shows the dynamic threshold.
- For the static threshold slider and curve knob, the orange line shows the static threshold.
- For the bias slider, the orange line shows the bias curve that gets multiplied by the threshold to bias it towards the low or high frequencies.

Double-clicking on the spectrum graph toggles it on and off.

Tooltips

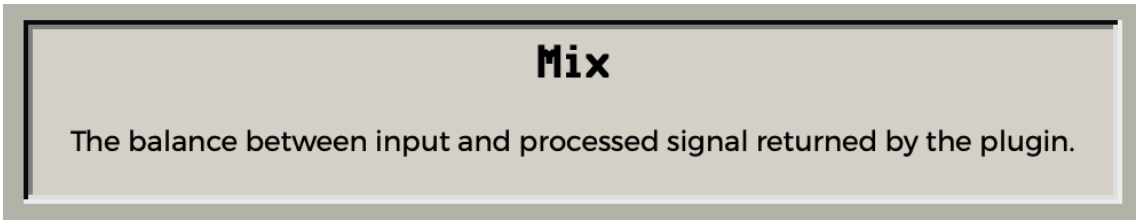

Figure 17: Example of a tooltip in the Empy user interface.

As you move your mouse over a control, the window in the top right of the GUI will display a description of what that control does.

Fish User Manual

Fish is an easy-to-use plugin with just one knob, that emulates the low-quality lossy distortion of "Hey Ya Low Quality."[75] Fish's one knob takes sound on a journey from a light lossy distortion to a distinctively nauseating low-quality sound, and finally to the limit of lossy distortion, where the sound is reduced to a sparse stippling of tones.

In designing Fish, I set out to produce a distinctive tone not found in other lossy distortion plugins. The lossy distortion of "Hey Ya Low Quality" had a full, churning tone to it that other plugins could not create. This sound intrigued me, and may have intrigued many of the commenters on the video as well.

How Fish Works

After experimenting with many settings on multiple MP3 encoders, as well as various chains of filters, saturators, distortion effects, lossy distortion plugins and bitcrushers, I was finally able to design a plugin that captured the sound of "Hey Ya Low Quality" that had intrigued me. The solution I found was to speed the input audio

[75] Bizzymcbob, *Hey Ya! Low Quality*.

up, apply MP3 compression, and then slow the audio back to its initial speed. The result was not a perfect match. However, this resampling and encoding process had the desired effect, removing the high end of the sound and giving it a slow, grinding MP3 compression.

Fish uses the MP3 encoder LAME to distort the audio. I used LAME for several reasons. LAME is an open-source MP3 encoder and decoder, so I could freely use the LAME library in my plugin, and even modify the LAME code. LAME is also a popular MP3 encoder, and has is used inside other pieces of open-source software, including the command-line audio processor FFmpeg[76] and the digital audio editor Audacity.[77] LAME was able to give the tone I was looking for, perhaps because of its widespread use establishing it as a de facto standard for how MP3 compression should sound. Since LAME is open source, and the patent for MP3 compression has expired,[78] there was no reason for me to not use this actual encoder to get the authentic sound I wanted from this plugin. The architecture of Fish also draws some inspiration from the plugin LameVST.[79] LameVST is a plugin that compresses audio using the LAME library. Although LameVST was not programmed using the JUCE library, it was programmed in C++, and referencing the code for LameVST was helpful in learning how to use the LAME library interface.

Designing Fish as a plugin presented other challenges besides merely identifying how to create the desired tone. Musicians generally expect a plugin to work on real-time

[76] "FFmpeg MP3 Encoding Guide," FFmpeg, 2015, https://trac.ffmpeg.org/wiki/Encode/MP3.
[77] "FAQ:Installing the LAME MP3 Encoder," Audacity, n.d., https://manual.audacityteam.org/man/faq_installing_the_lame_mp3_encoder.html.
[78] Andrew Orlowski, "MP3 'died' and Nobody Noticed: Key Patents Expire on Golden Oldie Tech," *The Register*, May 16, 2017, https://www.theregister.com/2017/05/16/mp3_dies_nobody_noticed/.
[79] Iunusov, "LameVST."

audio with relatively low latency, and for the tone to change smoothly, without pops or gaps, as they turn the plugin's knobs. LameVST can introduce much more latency than other plugins: in some of my tests, it delayed audio by over 450 milliseconds. To make matters worse, the amount by which it delays audio changes, apparently at random and without warning, each time the user changes a parameter. LameVST also introduces glaring gaps of silence, about 350 milliseconds long, whenever the user changes a parameter. These conditions are not necessarily a problem for LameVST. According to its author, LameVST is meant to be a plugin for "if you need to preview you *[sic]* track in the mp3-mode during playback." 450 milliseconds of latency or a brief pause when changing settings would likely be acceptable to someone previewing how a track would sound as an MP3 at a specific bitrate. Fish, on the other hand, is intended for use as a musical effect, so being able to use it live, or automate the parameters without gaps, would be expected features found in the vast majority of distortion plugins.

Several settings in the LAME library helped to limit Fish's latency. One source of delay that can be eliminated is the bit reservoir, as explained in an FAQ page for LAME.

> "The MP3 data for frame N is not stored in frame N, but can be spread over several frames. In a typical case, the data for frame N will have 20% of it stored in frame N-1 and 80% stored in frame N. If the encoder builds up a large bit reservoir, the data for frame N can actually be stored 4088 bits back in the bitstream."[80]

As audio data gets spread across more frames, the latency of the full encoding and decoding process grows. Luckily, the LAME API gives us the option to disable the bit reservoir, which means that each frame can be decoded as soon as it is encoded, without

[80] Mark Taylor, "LAME Technical FAQ," June 2000, https://lame.sourceforge.io/tech-FAQ.txt.

the need to wait for subsequent frames. The LAME API also lets us decrease delay in the encoding process further by setting the `ENCDELAY` parameter in `encoder.h`.

To achieve smooth automation, I modified the code of LAME. Unlike LameVST, Fish does not have to accurately model the sound LAME gives at a certain bitrate. The core of the LAME encoding process occurs in the file `quantize.h`, with the function call `outer_loop(gfc, cod_info, l3_xmin, xrpow, ch, targ_bits[ch]);`. This function takes a granule[81] of audio, the allowable distortion at each frequency (as calculated by the psychoacoustic model), the channel being encoded, and the target number of bits that should be used to encode that channel of that granule.[82] As seen in figure 18, the average value of `targ_bits` increases with the bitrate. Generally speaking, more bits are allocated to the first channel (the mid channel in mid/side encoding) than the second. If, instead of passing `targ_bits[ch]` as the final parameter in `outer_loop`, we pass some other number, we can actually change the quality.[83] This hack opens up three exciting possibilities. First, we can change the amount of lossy distortion on the fly, by changing the target bit value. Second, we can gradually adjust the target bit value, resulting in a smooth gradation of quality, instead of the dozen or so bitrates supported by the LAME API. Finally, we can push the target

[81] The MP3 encoder processes audio in 1152-sample frames, which are each divided into two granules. Thiagarajan and Spanias, *Analysis of the MPEG-1 Layer III (MP3) Algorithm Using MATLAB*, 9.
[82] It may seem odd that the two channels of a stereo signal would be allocated different numbers of bits. In joint stereo encoding, which Fish uses, if the left and right channels of a frame are similar, then the MP3 algorithm encodes the two channels as mid/side rather than left/right. Often, the mid channel will receive a higher bit budget.
[83] As I found, it only works to pass in values that are smaller than `targ_bits[ch]`. Passing in a larger value leads to a slew of errors, which is unsurprising: if we pass in too large of a number, we are essentially telling the function to write more bits than are expected from other parts of the code, which could lead to buffer overflows and overwrites.

bit value even lower than even the lowest bitrates supported by LAME, resulting in an exaggerated, bubbly lossy distortion.

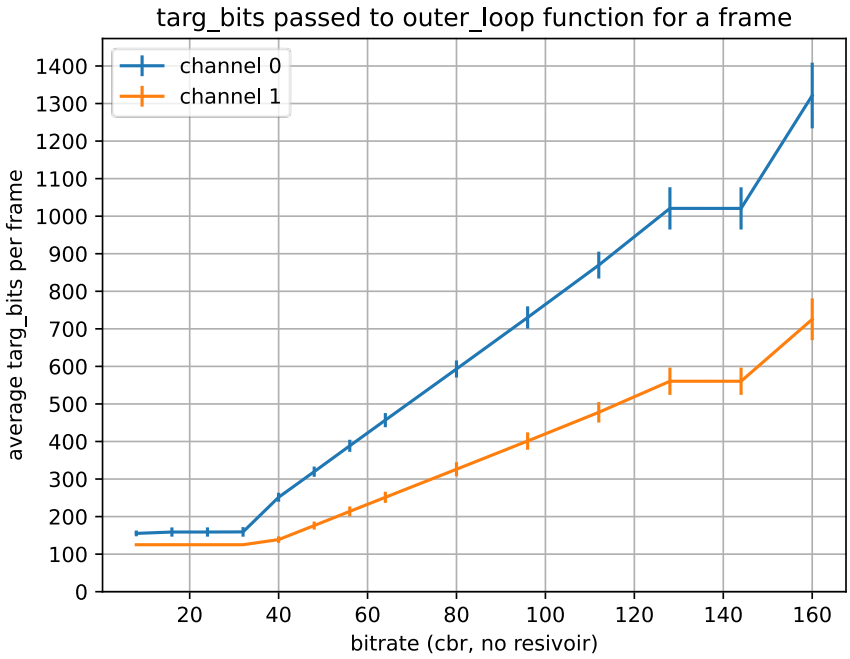

Figure 18: Average value and standard deviation of `targ_bits` passed to the `outer_loop` function of LAME in encoding of a track at various bitrates.

This hack comes at a cost. Although it may be possible, I could not find a way to change the cutoff frequency of LAME's low-pass filter on the fly. Since the muffled low-passed sound is an important part of the sound of lossy distortion (particularly in the sound of "Hey Ya Low Quality"), I added a simple low-pass biquad filter on the signal before passing it into the encoder. This filter's cutoff frequency lowers as the fish parameter raises.

To get the slow, churning distortion that I wanted from Fish required downsampling the audio before compressing it with LAME, and then upsampling it afterwards. When the signal is downsampled, each MP3 frame covers a greater amount

of time, intensifying the pre-delay artifact and slowing down the bubbling sound of the birdie artifact. As shown in the signal flow diagram, Fish first applies a low-pass filter on the incoming signal. This gives the sound the desired muffled tone, while also removing much of the high-frequency signal that would fold over into as noise.[84] Then, Fish downsamples the signal, applies MP3 encoding and decoding, and then finally upsamples the signal to the initial sample rate. To upsample the audio, I used linear interpolation, followed by a final low-pass filter. Although linearly interpolating between samples is one of the simplest approaches, it is, as researcher Olli Niemitalo writes, "not adequate for high-quality resampling." Linear interpolation adds more high frequency noise than higher-order (and more computationally expensive) polynomial interpolations.[85] With just linear interpolation, this high frequency noise was noticeable and annoying. By following it up with low-pass filtering, this noise was reduced noticeably.

Fish also lowers the amplitude of the incoming signal by about 6 dB before processing, and raises it by 6 dB after processing. Without this step, loud signals could clip during the MP3 compression step, leading to unwanted distortion.

[84] Because the roll-off of the low-pass filter used is not very steep, Fish still folds over some high frequencies, although they are attenuated, as shown in figure 3.
[85] Olli Niemitalo, "Polynomial Interpolators for High-Quality Resampling of Oversampled Audio," October 2001, 3, http://yehar.com/blog/wp-content/uploads/2009/08/deip.pdf.

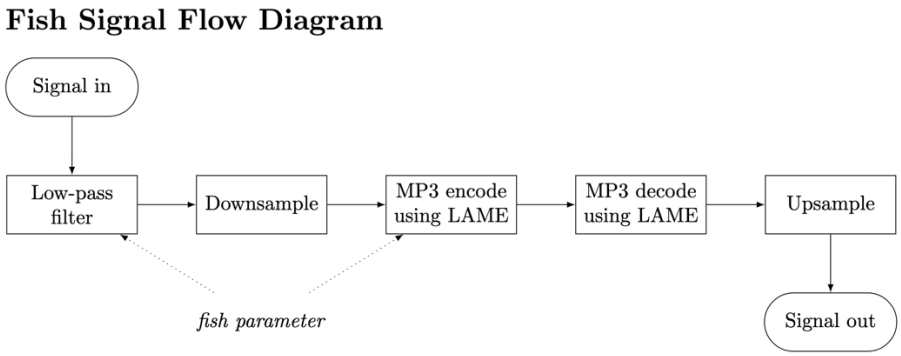

Figure 19: Fish signal flow.

The Fish GUI

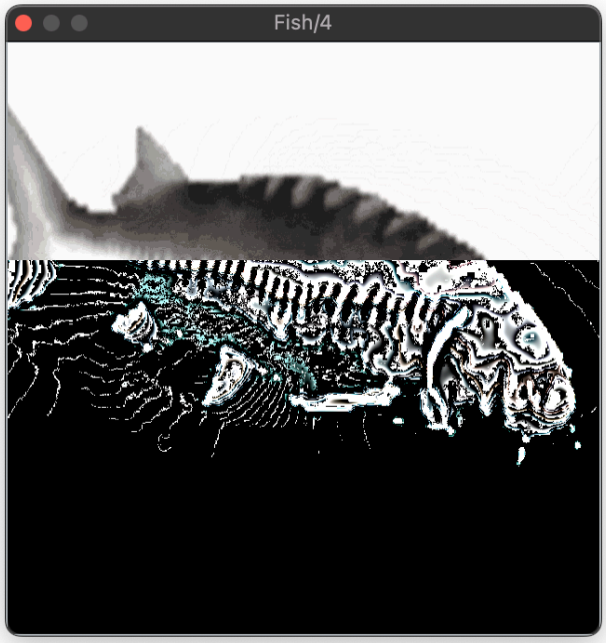

Figure 20: the Fish GUI.

Fish has one parameter, called fish. It is controlled by dragging up and down on the GUI. At higher amounts of fish, Fish lowers the target bit value per granule, and lowers the cutoff frequency of the lowpass filter. The animation spins faster the more fish there is.

Comparison of Plugin Performance

Both latency and CPU performance are important considerations for plugins. Musicians will often run many plugins at once, so plugins that use less computing power are more practical. Low latency is also a desirable feature in audio plugins, enabling them to be used in live settings. Using the plugin analysis tool PluginDoctor, we can see Empy and fish have comparable performance to other lossy distortion plugins, actually using less computing power than Digitalis or Codec. Empy and Fish both incur relatively high latency. In the case of Fish, downsampling audio by a factor of four quadruples the encoding latency. Even so, Fish's latency is only slightly longer than Digitalis' latency. Empy's latency is determined by the frequency resolution set by the user: higher frequency resolutions require wider windows, and thus more latency. While Empy can have much lower latency than any of the other plugins, Empy's tone at those low frequency resolutions does not sound like lossy distortion. These low frequency resolutions also increase the amount of computing power Empy uses. In Empy, a number of internal parameters get calculated once per frame, so at small window sizes, these calculations must be performed much more often, resulting in higher CPU usage.

Plugin	**Latency (milliseconds)**[86]	**Performance (microseconds/sample)**[87]

[86] Calculated based on the impulse response of the plugin in the plugin analysis tool PluginDoctor, at a sample rate of 44.1 kHz.

[87] Performance is a measure of how many milliseconds of processing time it takes to process a block of audio. For instance, Goodhertz Lossy has a performance of 0.2 microseconds/sample, so it would take approximately 200 microseconds, or 0.2 milliseconds, for Lossy to process a buffer of 1000 samples. For all plugins tested, the amount of processing time was approximately proportional to the buffer size, with each plugin having a slightly different slope. Performance data was collected using PluginDoctor. All trials were conducted on the same computer, a 2019 MacBook Pro.

Goodhertz Lossy	17.3	0.2
Toneboosters Bitjuggler	23.2	0.2
Aberrant DSP Digitalis	92.8	0.5
Lese Codec	56.6	0.9
Empy	0.2–185.8	0.3–0.5
Fish	103.2	0.2

Table 1: Latency and performance of lossy distortion plugins.

Conclusion

Lossy distortion is a difficult topic to pin down. Throughout this thesis, we have seen lossy distortion as a technical solution for limiting file size, as the low-brow way to listen to music, and as a tool for piracy. It can be an underwater voice, a boom-box in another room, or a comforting hug. It can be a gate, a filter, a bitcrusher, or a vocoder. In removing part of a sound, lossy distortion adds a distinctive tone, and the rich history that goes with it.

There is no way of knowing what role lossy distortion will play in music production in the future, and there are likely many untapped creative applications of lossy distortion. This is an exciting time to be studying the creative uses of lossy distortion, as the field is rapidly changing. Several of the plugins reviewed here were not yet released when I started this thesis. During the COVID-19 pandemic, many people became intimately familiar with the audio quality of videoconference meetings, which are rife with lossy distortion, and other digital glitches.[88] If Eno's predictions are any guide, the digital failure of videoconference meetings may return as a creative musical effect.

Future Improvements to Empy and Fish

As often happens with software, I am releasing initial versions of Empy and Fish while there are still further features I want to implement in both of them. One feature I want to add in Empy is something similar to the "signal strength" parameter in

[88] In fact, the developers of one microphone and speaker emulator plugin Speakerphone cashed in on the trend, marketing a set of presets that emulate distortions from conference call audio. AudioEase, "Speakerphone Conference Calls," YouTube, July 17, 2020, https://www.youtube.com/watch?v=clZMIpLey3k.

Digitalis. When this is turned up, larger groups of frequency lines are treated as a single unit, and either all pass the gate or all do not pass. Adding this feature would not involve much restructuring to the back end of Empy. Empy could also benefit from presets: it could have factory presets, and the option for users to make presets of their own. This could make the program less intimidating, and help users discover new sounds. Additionally, Empy could benefit from an option to turn off the spectrum. Some users might find its flashing lights distracting, or might prefer a meter-less experience that lets them trust their ears. User feedback could also help to guide further development of Empy.

In further versions of Fish, I want to explore giving the user slightly more control over the sound, while still keeping the user interface very simple. In keeping Fish easy to use, I want the sound to change noticeably any time the user moves any knob. This is not always the case in more complicated plugins: sometimes a knob changes the sound in a very subtle way, or the change does nothing because of the current settings of other parameters (for instance, changing the modulation rate on an LFO when the modulation depth is at zero). That said, there are some extra controls that would likely satisfy this criteria, and would also let Fish make a wider range of sounds. For instance, the user could change the frequency of Fish's low-pass filter independently from the encoding bitrate. Fish could also have some soft clipping distortion or waveshaping, which would add to the low-quality sound. I also want to explore lowpass filters with steeper roll-offs for Fish, to minimize the noise from downsampling.

Further Directions for Future Lossy Distortion Plugins

One direction I plan to explore is digital circuit bending with LAME. Circuit-bending is a form of glitch art in which consumer sound circuits are "bent," usually by connecting disparate points on the circuit. In modifying the parameters passed to the `outer_loop` function in LAME, I dabbled in digital circuit bending, sending the wrong value through a point in the control flow. A core element of circuit bending is experimentation. Circuit-bending artist Reed Ghazala argues that whereas the initial circuits are built to produce sounds that the manufacturers think people want to hear, "circuit-bending probes the circuitry to hear its intrinsic music, allowing it a personal prose beyond programmed recitation."[89] While circuit-bending is more commonly implemented with hardware, the theory and methods of circuit-bending can apply to software as well. One example of this is the artist and programmer Gankra's project "Let's Corrupt the Web," in which she modified the rendering code of the Firefox web browser to glitch websites in various ways. Gankra writes:

> "I've gotten my best results with a technique I like to call Right Value Wrong Place. The basic idea is simple: rather than just inserting random changes or disabling some important check, take some of the real and correct values being fed into the code you're corrupting, and use them in a completely wrong place."[90]

This approach has clear similarities with circuit-bending, in that the signal from one part of the algorithm is being sent to another part. Initial tests with circuitbending LAME have proven promising, although there is still a long way to go to create a useful plugin.

[89] Reed Ghazala, *Circuit-Bending: Build Your Own Alien Instruments*, vol. 15 (John Wiley and Sons, 2005).
[90] Gankra, "Let's Corrupt The Web," Faultlore, n.d., https://faultlore.com/glitch/.

A plugin that circuitbends LAME could also include other non-standard uses of LAME, such as re-encodings. As Sterne writes, "if you send MP3-coded audio into an MP3 coder, the artifacts of the encoding process are hypertrophied."[91] Although multiple encodings would only increase the latency of the plugin, the sound may be worth it, especially in a plugin that is not intended for live use, so latency would be less of an issue.

[91] Sterne, *MP3: The Meaning of a Format*, 235.

Bibliography

Apple. “About Lossless Audio in Apple Music,” October 25, 2021. https://support.apple.com/en-us/HT212183.

Ashby, Arved. *Absolute Music, Mechanical Reproduction*. Berkeley and Los Angeles, California: University of California Press, 2010.

Audacity. “FAQ:Installing the LAME MP3 Encoder,” n.d. https://manual.audacityteam.org/man/faq_installing_the_lame_mp3_encoder.html.

AudioEase. “Speakerphone Conference Calls.” YouTube, July 17, 2020. https://www.youtube.com/watch?v=clZMIpLey3k.

Bagwell, Chris. “Sound EXchange (SoX).” C, February 22, 2015. https://sox.sourceforge.net/.

Barlindhaug, Gaute. “Analog Sound in the Age of Digital Tools: The Story of the Failure of Digital Technology.” In *A Document (Re)Turn*, edited by R. Skare & L. Windfield L. & A. Varheim. Peter Lang Ltd, 2007.

“BitJuggler / BitJuggler IOS Installation and User Manual.” Tonebusters, 2020 2010. https://www.toneboosters.com/manuals/TB_BitJuggler.pdf.

Bizzymcbob. *Hey Ya! Low Quality*. March 7, 2021. YouTube video. https://www.youtube.com/watch?v=LMaG_uOa440.

Boym, Svetlana. “The Future of Nostalgia.” New York: Basic Books, 2001.

Brandenburg, Karlheinz. “MP3 and AAC Explained.” In *Audio Engineering Society Conference: 17th International Conference: High-Quality Audio Coding*. Audio Engineering Society, 1999.

Cole, Ross. “Vaporwave Aesthetics: Internet Nostalgia and the Utopian Impulse.” *ASAP Journal* 5, no. 2 (2020): 297–326.

Computer Music. “Dada Life Sausage Fattener Review.” musicradar, August 22, 2011. https://www.musicradar.com/reviews/tech/dada-life-sausage-fattener-490089.

Costello, Sean. “Valhalla Supermassive: The Controls.” *Valhalla* (blog), May 6, 2020. https://valhalladsp.com/2020/05/06/valhallasupermassive-the-controls/.

Dovydaitis, Gytis. “Celebration of the Hyperreal Nostalgia: Categorization and Analysis of Visual Vaporwave Artefacts.” *Menos Istorija Ir Kritika (Spausdinta)* 17, no. 1 (2021): 113–34.

Eno, Brian. *A Year with Swollen Appendices*. Faber and Faber Ltd., 1996.

Erkelens, Johan S. *Autoregressive Modelling for Speech Coding: Estimation, Interpolation and Quatisation*. Delft University Press, 1996.

FFmpeg. "FFmpeg MP3 Encoding Guide," 2015. https://trac.ffmpeg.org/wiki/Encode/MP3.

Gankra. "Let's Corrupt The Web." Faultlore, n.d. https://faultlore.com/glitch/.

Geiger, Ralf, Manfred Lutzky, Markus Schmidt, and Markus Schnell. "Structural Analysis of Low Latency Audio Coding Schemes." *Journal of The Audio Engineering Society*, 2005.

Ghazala, Reed. *Circuit-Bending: Build Your Own Alien Instruments*. Vol. 15. John Wiley and Sons, 2005.

Greenberg, Corey. "Free to Be, MP3." *Sound & Vision*, June 1999.

Goodhertz. "Lossy," n.d. https://manuals.goodhertz.com/3.6/lossy/.

Hacker, Scot. *MP3: The Definitive Guide*. O'Reilly & Associates, 2000.

Hargreaves, Leighton. "Convoluter." Anarchy Effects. Anarchy Sound Software, 2013.

Hegarty, Paul. *Noise/Music: A History*. New York: Continuum, 2007.

Herbst, Jan-Peter. "Shredding, Tapping and Sweeping: Effects of Guitar Distortion on Playability and Expressiveness in Rock and Metal Solos." *Metal Music Studies*, 2017.

Hydrogenaudio Knowledgebase. "Vorbis." Wiki, n.d. https://wiki.hydrogenaud.io/index.php?title=Vorbis.

Inglis, Sam. "Waves OneKnob." Sound on Sound, June 2011. https://www.soundonsound.com/reviews/waves-oneknob.

Iunusov. "LameVST," January 14, 2017. https://github.com/Iunusov/LameVST.

Jansson, Tord. "Blade MP3 Encoder." C, March 7, 2001.

Jones, Ellis. "The Slow Sublime and 9/11: Insecurity and Fear in William Basinski's the Disintegration Loops." *Music & Politics* VIII, no. 1 (2014).

Lake, Matt. "The MP3 Revolution." *Sound & Vision*, December 1999.

lame.cvs.sourceforge.net. "History," n.d. https://archive.ph/M0Yvq.

Lese. “New Release: Codec, an Audio Plugin for Modern Degradation,” August 6, 2022. https://lese.io/blog/new-plugin-codec/.

Liu, Chi-Min, Han-Wen Hsu, and Wen-Chieh Lee. “Compression Artifacts in Perceptual Audio Coding.” *IEEE Transactions on Audio, Speech, and Language Processing* 16, no. 4 (2008): 681–95.

Maguire, Ryan. “An Aesthetic of the Irreducible.” PhD Thesis, University of Virginia, 2019.

———. “The Ghost in the MP3,” 2014. https://www.theghostinthemp3.com/theghostinthemp3.html.

Mo, Ronald, Ga Lam Choi, Chung Lee, and Andrew Horner. “The Effects of MP3 Compression on Perceived Emotional Characteristics in Musical Instruments.” *Journal of the Audio Engineering Society* 64, no. 11 (2016): 858–67.

Niemitalo, Olli. “Polynomial Interpolators for High-Quality Resampling of Oversampled Audio,” October 2001. http://yehar.com/blog/wp-content/uploads/2009/08/deip.pdf.

O’Grady, Pat. “Rethinking Criticism about Lossy Compression: Sound Fidelity, Large-Scale Production and Audio Capital in Pop Music.” *Convergence (London, England)* 27, no. 4 (2021): 1075–91. https://doi.org/10.1177/1354856520976454.

Orlowski, Andrew. “MP3 ‘died’ and Nobody Noticed: Key Patents Expire on Golden Oldie Tech.” *The Register*, May 16, 2017. https://www.theregister.com/2017/05/16/mp3_dies_nobody_noticed/.

Patterson, Ben. “Spotify HiFi Release Date: When Is Spotify’s Lossless Tier Coming?” *TechHive*, October 14, 2022. https://www.techhive.com/article/790882/spotify-hifi-release-date-when-is-spotifys-lossless-tier-coming.html.

Perkins, Colin, Orion Hodson, and Vicky Hardman. “A Survey of Packet Loss Recovery Techniques for Streaming Audio.” *IEEE Network* 12, no. 5 (1998): 40–48.

Pinch, Trevor, and David Reinecke. “Technostalgia: How Old Gear Lives on in New Music.” In *Sound Souvenirs*, 152. Amsterdam University Press, 2009.

Producer Feed. “FL Studio’s Soundgoodizer Explained – How It Works,” August 2, 2019. https://www.producerfeed.com/2019/08/fl-studios-soundgoodizer-explained-how-it-works.html.

Ranada, David. “Download Showdown II.” *Sound & Vision*, September 2000.

Raw Material Software Limited. “JUCE,” 2020. https://juce.com/.

Reeman, Andrew. “Spectral Gate,” 2022. https://www.andrewreeman.com/spectralsuite/.

Rosenberg, Jacob. “The Distortion of Sound,” 2014.

Sairam, K, and KJS Lorraine. “Design of Speech Codec for VoIP Applications.” *International Journal of Scientific Engineering and Technology Research* 4, no. 27 (2015): 5350–55.

“Searchterms.Com,” April 23, 1999. https://web.archive.org/web/19990423152020/http://www.searchterms.com/.

Sonnox. “Fraunhofer Pro-Codec.” Accessed May 28, 2022. https://www.sonnox.com/plugin/fraunhofer-pro-codec.

“Speakerphone User Guide,” n.d. https://www.audioease.com/speakerphone/files/speakerphone-manual.pdf.

Stenson, Rob. “Lossy’s Uncharted Waters.” Goodhertz. *Tonal* (blog), April 7, 2015. https://goodhertz.com/tonal/uncharted-waters/.

Sterne, Jonathan. *MP3: The Meaning of a Format*. Durham: Duke University Press, 2012.

———. “The MP3 as Cultural Artifact.” *New Media & Society* 8, no. 5 (2006): 825–42.

Taylor, Mark. “LAME Technical FAQ,” June 2000. https://lame.sourceforge.io/tech-FAQ.txt.

Thiagarajan, Jayaraman Jayaraman, and Andreas Spanias. *Analysis of the MPEG-1 Layer III (MP3) Algorithm Using MATLAB*. Edited by Andreas Spanias. Synthesis Lectures on Algorithms and Software in Engineering ; #9. San Rafael, Calif.: Morgan & Claypool Publishers, 2012.

Vorbis I Specification. Xiph.Org Foundation, 2020.

Witt, Stephen. *How Music Got Free: The End of an Industry, the Turn of the Century, and the Patient Zero of Piracy*. Viking, 2015.