Implementation of a Distortion Filter in C++

Konstantin Kharitonov, Undergrad Student, University of Regina, SID: 200354502

***Abstract*—Distortion effects are one of the most sought after effects in the music industry perfecting them is what musicians and engineers have spent their entire careers doing. What followed was years and years of trying to prefect the art of creating the best distortion sound, in an analog and in a digital aspect.**

I. Introduction

THE process of digital filtering requires manipulating an existing recorded signal by converting it to a digital signal, such that each sample recorded can then be processed through filters in order to recreate the intended effect. This process, while can be simple to conceive in a simplistic sense, it is still a daunting task to perfect an effect. This is because with most audio effects and filters are analog, which a digital system can only simulate rather than recreate fully. That hasn't stopped the rise of digital effects from being implemented however, because of it being a much more cost effective method of producing effects.

II. Distortion

Distortion is one of the oldest effects that has been developed, and since the age of computing, musicians and programmers have been trying to recreate it on a computer since the days of midi sound cards. Distortion happens when an audio signal becomes too intense for a recording and/or amplification device, the signal will be clipped and any sample that has an amplitude higher or lower than the threshold of the device, then the sample will instead have an amplitude that is equal to that threshold.

In this paper, two such implementations of this phenomenon will be discussed. The first method is a simple clipping method. This algorithm takes each single sample from a particular signal, and checks to see if the amplitude has exceeded the predetermined threshold. If it has, then the amplitude of the particular sample is then changed to equal to the established threshold.

Definition II.1. *Simple Distortion*

$$f(x) = \begin{cases} 0.07 & \text{if } x > 0.07 \\ -0.07 & \text{if } x < -0.07 \end{cases}$$

III. Implementation of the Simple Distortion Algorithm

Implementing the algorithm is simple. Once the the amplitudes of the particular audio samples are stored onto an array, each particular sample is checked through a for loop. If a particular amplitude exceeds the threshold stated above, then the algorithm replaces that particular amplitude with the threshold instead, causing clipping to occur.

Algorithm 1 Simple Distortion

Require: array of amplitudes in, array of outputs out
simple(in , out)
for $i = 0$, $i <$ sizeof in **do**
  if $in > 0.07$ **then**
    $out[i] \leftarrow 0.07$
  else if $in < -0.07$ **then**
    $out[i] \leftarrow -0.07$
  else
    $out[i] \leftarrow in[i]$
  end if
end for

IV. Complex Distortion

The complex distortion effected is a more complicated one, first developed by R. Bendiksen of the Norwegian University of Science and Techology. In his thesis statement, instead of only clipping the amplitudes that exceed a certain threshold, each sample is effected and clipped such that every single amplitude is distorted in the transformed audio signal. This method is shown as followed.

Definition IV.1. *Complex Distortion*

$$f(x) = \frac{x}{|x|}\left(1 - e^{\frac{x^2}{|x|}}\right)$$

Where x is the amplitude of a particular sample and $f(x)$ is the altered sample. This function is altered slightly in implementation to allow a gain and blend feature. The gain allows the user to add in how much of a distortion effect do they want to add to their signal. The higher the gain, the more distortion there will be. The blend feature adds to how much of the distortion will blend in with the original sample, with a blend of 1 being a completely wet signal. The resulting function is as follows.

Definition IV.2. *Complex Distortion with gain and blend*

$$f(x) = blend * \frac{x}{|x|}\left(1 - e^{gain * \frac{x^2}{|x|}}\right)$$

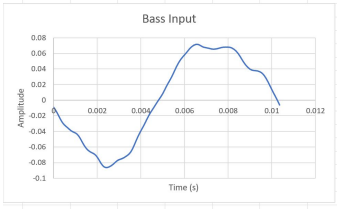
V. Implementation of the Complex Distortion Algorithm

VI. Visual Implementation

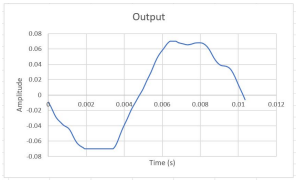
Below here is the unaffected input signal, showcasing the wave of the original signal.

Algorithm 2 Complex Distortion Algorithm

Require: array of amplitudes in, array of amplitudes out, gain, blend
complex(in, out, gain, blend)
for $k = 0$, $k < sizeofin$ **do**
$x = in[i] * abs(in[i])$
$out[i] \leftarrow blend * x * (1 - e^{gain * \frac{x^2}{|x|}}$
end for

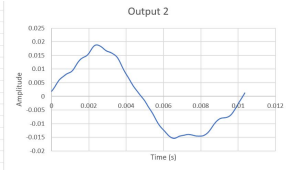


The next graph is the implementation of the simple distortion, showcasing the effected plot points.



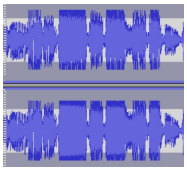
As is shown by the graphs, all of the points that were passed the threshold in the input signal now have been clipped and replaced by the new threshold of 0.07 and -0.07 respectfully.

The last graph showcases the transformation of the entire signal using the complex distortion algorithm.



Showcased by the graph, the complex distortion pedal completely changes the original signal, changing the make up of each sample by different values. As such, the sample sounds much grittier and much heavier as to compared to the input, showcasing how distorted the signal got. The sound however was actually less distorted than of the simple distortion algorithm from above.

In comparison to audacity's distortion effect, this method is similar in practice but one downfall remains that audacity has perfected. In both of the algorithms above, there are still samples that are out of position and result in clicking sounds to be heard in the recording. Audacity's algorithm has removed this issue, thus is still a superior distortion system.



Shown here, the Audacity algorithm does not allow any single clicking amplitudes remain after the signal has been distorted

VII. Conclusion

Distortion effects were originally invented by accident when guitar players would accidentally rip their speakers while recording in the studio. What followed was years and years of trying to prefect the art of creating the best distortion sound, in an analog and in a digital aspect. While the formulas we use here can definitely provide a distorted sound, there is still more implementation to be done in order to perfect the algorithms.

Acknowledgment

R. Bendiksen. Digitale Lydeffekter. Master's thesis, Norwegian University of Science and Technology, 1997.

DAFX: Digital Audio Effects Udo Zolzer

Copyright q 2002 John Wiley & Sons, Ltd ISBNs: 0-471-49078-4 (Hardback); 0-470-84604-6 (Electronic)