APHASIA: ARTICULATED GLITCH EFFECT PLUG-IN

Craig A. Hanson

Stanford University
CCRMA, Stanford, CA
chanson9@ccrma.stanford.edu

ABSTRACT

In this paper, the author provides a brief historical context of the glitch effect and a musical perspective on the traditional uses of this effect in contemporary electronic music. This historical perspective is used as motivation for the development of the VST plug-in, Aphasia, which addresses issues of expressiveness in the effect. Following is a discussion of the plug-in design in Matlab and its real-time implementation in VST format. Overall project results are then presented.

1. INTRODUCTION

The glitch effect has become a standard in todays electronic music industry. The effect can be used as a startling, haunting and sometimes beautiful addition to both pre-composed and live musical material. Indeed, this musical alteration has launched an entire genre of music. However, as it is used in most applications, glitch lacks expressive quality. Many plug-ins use randomized stuttering or re-triggering algorithms that disregard musical articulation and rhythmic motive. The plug-in presented here attempts to apply programmed musical gesture to the glitch effect.

2. THE ARTFORM OF GLITCH

2.1. Origins

The artistic use of stuttered sound for musical fulfillment can be traced back to the origins of rhythm in music itself. While rhythm began without thought to computerized, metronomic precision, the earliest musicians formulated subdivisions of time through tempo and developed repetitive motives that formed the basis for their music [5]. The use of minute rhythmic alterations and repetition of sound have always been a staple of the musical mind.

2.2. Computerized Musical Control

Artists, such as Oval, began experimenting with repetitive electronic music by scratching the surface of compact discs to induce skipping and create some of the first glitched musical material [1]. It took great amounts of effort to control the skips of a compact disc in a musical manner. The material had to be played back, re-recorded and played back again several times to attain the desired result. Nevertheless, the effect was captivating and was soon adopted by many artists.

With the advent of computer technology, it became possible to have control over musical articulations below the threshold of perception. This fact was exploited by computer programmers and artists who developed applications for the manipulation of micro-timing in musical material.

While the glitch effect found its home in experimental electronic music, it soon proliferated to commonplace in such genres as techno, trip-hop, drum and bass, and dance music. This was facilitated by the use of digital audio in computers. Computer programs were devised to provide the same kinds of glitched sound generated by skipping a cd or record directly in digital audio workstations. Granular synthesizers were also developed which could produce similar effects, but also generated their own audio. These initial developments were based mainly on randomized re-triggering of input sounds.

As VST plug-ins and other general-purpose DAW effects processors became commonplace, individuals began exploring the glitch effect plug-in. Nick Collins developed the BBCut library in SuperCollider [4]. This library was the basis for the LiveCut plug-in. LiveCut offers many parameters for glitch effect manipulation, but little control over musical articulation of the effect. Kieran Foster developed one of the first controlled glitch effect plug-ins, Glitch [2]. This plug-in offered a tempo-synced glitch with a myriad of parameters for the artist to take control of their sound. It was a great first step towards a more musical use of glitch.

In 2006, the popular dance music artist BT released an album entitled This Binary Universe [6]. In the authors opinion, this album presented the first use of glitch as a truly musical aesthetic. The album featured dream-like soundscapes and the micro-manipulation of both acoustic and electronic sounds. These manipulations were achieved through the use of the Beat Stutter plug-in, a program developed by BT in CSound to sculpt the feel of the album. The sounds on this album have become a standard by which I will measure the effectiveness of the plug-in presented here.

In 2008, the electronic musician Barry Lynn released "Balancing Lakes" [3]. The album presented a sophisticated use of glitch in well-programmed intelligent dance music.

He used glitch for both subtle and outlandish musical statements throughout the album. It featured usage of rhythmic articulations to convey tension and repose in an artistic manner over the course of the musical material.

3. THE GLITCH EFFECT DEMYSTIFIED

There are many ways to achieve an effective glitch effect. The use of glitch as an artform may sometimes call for smooth transitions between glitched segments or it may call for jarring irregularities. This glitch effect most often uses windowing of the glitched waveform segments to eliminate harsh artifacts that are created when a waveform is chopped off. However, sometimes it is desirable to eliminate windowing in the effect and cause that kind of disruption. The following sub-sections provide descriptions of popular methods for achieving the glitch effect.

3.1. Retriggering

Retriggering is the process by which a sound is stored in an audio buffer and played back, usually in a rapid fashion that is quantized to a host tempo. Retriggering can be used to generate granulated soundscapes or harsh, repetitive musical statements.

3.2. Stuttering

Stuttering is similar to re-triggering, except that a stutter effect does not necessarily use the same sample material for every glitch. A stutter may be more appropriate when attempting to glitch a live audio input of melodic material. The melodic material can be sampled and short segments of it may be stuttered while keeping the melodic phrasing relatively in tact.

3.3. Silence Insertion/Granulation

Silence may be inserted at quantized (or non-quantized) time intervals throughout a live or pre-recorded audio stream to achieve a glitch effect. When using silence insertion, it is typically desirable to window the audio just before the silence is presented.

3.4. Gating

Silence may be present in an audio stream unless the audio reaches a pre-specified threshold. The audio will then be let through the gate and played. This can be effective in glitching certain drum patterns, material with large variations in amplitude or material that has a consistent amplitude level close to the threshold.

4. MATLAB IMPLEMENTATION

4.1. Overview

The initial prototype of the glitch effect was done in Matlab. It was designed to glitch a waveform using retriggering or silence insertion. The user can choose the method of glitch along with the length of the glitch and spacing in between glitches by simply manipulating sliders in the GUI shown below.

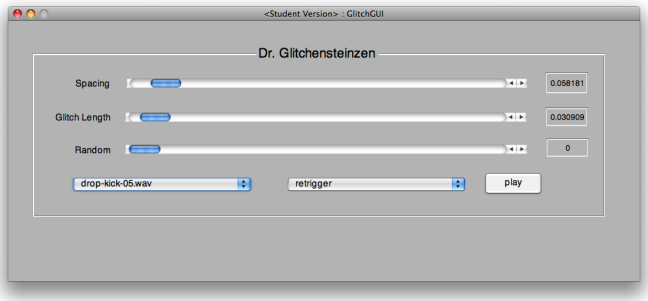

Figure 1. Dr. Glitchensteinzen Matlab Interface

4.2. Implementation

The initial Matlab implementation makes use of simple matrix manipulation for silence insertion effect and a single delay line for the retriggering effect (note: the VST plug-in will use a delay line to achieve silence insertion, retriggering and stuttering effects). The delay line is a circular buffer manipulated by read and write pointers. The write pointer moves on a sample by sample basis, writing audio into the buffer. The read pointer moves algorithmically through the buffer, pulling out audio and developing a final audio matrix, that is played via the play button. This Matlab implementation is compatible with a standard stereo wave file, therefore two delay lines are used (left and right).

4.2.1. Silence Insertion

Silence insertion is achieved by chopping out evenly spaced segments of silence throughout the input wave file. The glitch length slider controls how long audio will play in between the silent segments. The spacing slider controls how long the silence lasts in each silent segment. The silence is inserted through the use of an inverse hann window of the length called for by the length slider. By multiplying the inverse hann window (eq.1) with the incoming audio, a smooth segment of silence is introduced without any unwanted artifacts.

$$f(n) = 1 - \frac{1}{2}(1 + \cos(\frac{2\pi n}{N-1})) \quad (1)$$

4.2.2. Retriggering

Retriggering is achieved through a circular buffer. A segment of incoming audio (length determined by slider) is added to the delay line in evenly spaced intervals. The read pointer pulls out these evenly spaced segments of audio, multiplies them by a hann window (eq. 2) and places them into a matrix for playback. Through this method, the same segment of audio may be triggered over and over again, resulting in the desired effect.

$$f(n) = \frac{1}{2}(1 + \cos(\frac{2\pi n}{N-1})) \quad (2)$$

5. VST PLUG-IN

5.1. Overview

The VST Plug-In, Aphasia, is a real-time implementation of the stated glitch effect that can be used with any VST host. The author has used Steinbergs VST SDK 2.4 to develop the plug-in in the C++ programming language.

5.2. Features

Aphasia offers tempo-synced rhythmic glitching of an audio stream or sample and hold buffer. The features of each section are expanded below along with each user control parameter.

5.2.1. Silence Insertion

Silence Insertion works the same way in Aphasia as it does in the matlab prototype. The user specifies a glitch length and silence length in milliseconds and may activate the process by turning on the plug-in process. The silence is achieved in the plug-in by implementing a raised cosine window (eq. 3) on the audio. The equation below describes this window using n as the current sample and λ as the length variable.

$$-.5 * cos(2 * \pi * n/\lambda) + .5 \quad (3)$$

The user control parameters for Silence Insertion are as follows:

- On/Off - Simple binary switch to turn the effect on or off.
- Length (ms) - Controls the milliseconds of audio that will play during each glitch.
- Spacing (ms) - Controls the milliseconds of silence inserted in between audio segments.

5.2.2. Retriggering

Retriggering provides a quantized glitch of pre-sampled audio that repeats at user specified intervals. This function works by sampling the input audio into a retrigger buffer which is then used as source material when the function is turned on. The retrigger buffer is independent of the main audio buffer and allows for sample and hold control over the audio. This particular function is quantized using the global tempo and PPQ (parts per quarter) provided by the VST host using the getTimeInfo() function.

- On/Off - Simple binary switch to turn the effect on or off.
- Length (Note) - Controls the quantized note length of audio that will play during each glitch, ranging from a whole note to 1/512th note.
- Spacing (Note) - Controls the quantized note length of silence inserted in between audio segments, ranging from a whole note to 1/512th note.

5.2.3. Articulation

Articulation provides a means by which the user can have automated control over rhythmic statements using glitches in the audio stream or a retrigger buffer. The articulations are calculated by calling a function which evaluates the number of samples of audio needed by each portion of the requested articulation and storing that data in an array. As required, the array is indexed, causing the raised cosine window to be calculated appropriately in the desired sequence.

- On/Off - Simple binary switch to turn the effect on or off.
- Sample and Hold - Binary switch to use the retrigger audio buffer and repeat the previous audio heard rather than the input audio stream.
- Length (Note) - Controls the quantized length of the entire glitch articulation, ranging from 2 full bars to a 1/64th note.
- Articulation Center - Controls the center of the articulation. 0% will play an articulation of increasing speed. 100% will play a glitch of decreasing speed.

5.3. Development

The VST plug-in was written and re-written many times over. During the course of this project, I learned many new things about the VST SDK and coding for real-time audio which caused me to rethink my design. The final code is apportioned into six main classes. These provide all the functionality that Aphasia needs to operate and communicate with the VST host.

- AudioEffectX - This class is provided by the VST SDK and is inherited by Aphasia to provide the functions necessary to hand off audio buffers and to provide a basic UI.
- Aphasia - The main class containing all buffer handling and UI design.
- DelayLineEffect - Parent class of Articulation, Retrigger and Silence Insertion. This contains general parameters for all delay line based effects.
- Articulation - Contains required variables and functions for rhythmic articulations.
- Retrigger - Contains required variables and functions for retriggering.
- Silence Insertion - Contains required variables and functions for silence insertion.

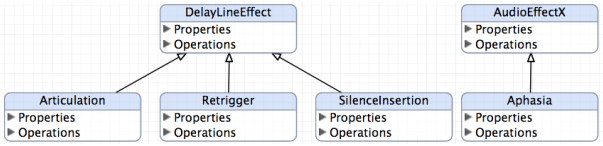

Figure 2. Aphasia Class Model

6. RESULTS

This project yielded some nice results, including a short composition which features the use of only royalty-free samples and Aphasia. The initial project goals were met, most notably an automated means of musical rhythmic articulation through the use of glitch. The VST implementation provides a versatile platform for the use of Aphasia, since it can be used directly in the VST host of your choice. Please refer to the Aphasia website to download the composition, the plug-in and supporting documentation:

http://ccrma.stanford.edu/∼chanson9/220C/220C.html.

7. REFERENCES

[1] K. Cascone, "The aesthetics of failure: "post-digital" tendencies in contemporary computer music," *Comput. Music J.*, vol. 24, no. 4, pp. 12–18, 2000.

[2] K. Foster, "dblue glitch plug in," http://illformed.org/plugins/glitch/.

[3] B. Lynn, "Balancing lakes," compact disc, March 2008.

[4] N. C. Rm and N. Collins, "The bbcut library," in *In Proc. Int. Computer Music Conference*, 2002, pp. 16–21.

[5] C. Sachs, *Rhythm and Tempo: a study in music history*. New York: W.W. Norton, January 1953.

[6] B. Transeau, "This binary universe," compact disc, February 2006.