Glitch Art Generation and Performance Using Musical Live Coding

Noriki Amano(✉)

Mukogawa Women's University, Nishinomiya 6338558, Hyogo, Japan
amnrk@mukogawa-u.ac.jp

Abstract. A glitch means an error in image, video, or sound, and glitch art is art in which glitches are intentionally generated by destroying digital data or physically manipulating electronic devices for artistic purposes. The effects of such glitches are often unpredictable and aesthetically interesting. As part of our exploration of glitching methods and performance, we have attempted to generate glitch art images using musical live coding. Specifically, we generated glitches in PNG image data in conjunction with sound effects using the Sonic Pi: musical live coding system. Furthermore, such glitch art images are generated in real-time. We have confirmed that our method can generate glitches with regularity assuming the filtering of PNGs. This research proposes a method of art expression across sound and image and is an attempt to explore the performance of glitch art.

Keywords: Glitch Art Images · Live Coding · PNG

1 Introduction

A glitch means an error in images, video, or sound. Glitch art is art in which glitches are intentionally generated by destroying digital data or physically manipulating electronic devices for artistic purposes. The effects of such glitches are often unpredictable and aesthetically interesting.

Live coding means writing and executing a program on the spot and has been a common practice in programming workshops. However, live coding as a performing art that considers the improvisational act of coding as a means of expression has been attracting attention in recent years.

We have been exploring glitching methods and their performance, and as part of this exploration, we have attempted to generate glitch art images using musical live coding. Specifically, we generated glitches in PNG image data in conjunction with sound effects using the Sonic Pi: musical live coding system. Furthermore, such glitch art images are generated in real-time. We have confirmed that our method can generate glitches with regularity assuming the filtering of PNGs. This research proposes a method of art expression across sound and image and is an attempt to explore the performance of glitch art.

A. L. Brooks (Ed.): ArtsIT 2023, LNICST 565, pp. 175–185, 2024.
https://doi.org/10.1007/978-3-031-55312-7_13

This paper is organized as follows: Sect. 2 describes the background of this research on glitch art and live coding as performing art, and related works. Section 3 describes the basic approach of this research. Section 4 outlines the prototype system and its implementation. Finally, Sect. 5 concludes the paper.

2 Research Background and Related Works

2.1 Glitch Art

A glitch means an error in images, video, or sound. Glitch art is art in which glitches are intentionally generated by destroying digital data or physically manipulating electronic devices for artistic purposes.

Such glitch art is classified as follows [1].

- **Data manipulation**: changing the data in a file and causing glitches
- **Misalignment**: opening a file from one app in another app
- **Hardware malfunction**: causes a machine to malfunction, producing sound or video
- **Misregistration**: physical noise in analog media
- **Distortion**: creating physical distortion with magnets, etc.

The effects of such glitches are often unpredictable and aesthetically interesting. In addition to the publication of a book on art using glitches [2], an international conference on glitch art such as GLI.TC/H [3] has been held, and glitching is already recognized as a form of expression in art.

Theoretical research has also been conducted on this kind of glitch art. Rosa Menkman uses information theory to propose an understanding of glitch art as a specific genre of contemporary art [4]. In 2010, an international conference on glitches, GLI.T/CH, was held by Roasa Menkman and others. Glitch art research has been also presented at the international conference evomusart, which has been held every year since 2011 [5].

2.2 Live Coding

Live coding as a performing art has been taking place worldwide since 2000. Events such as Algorave [6] have been held in various locations, led by the live coding community TOPLAP [7], and academically, international conferences such as ICLC [8] have been held annually since 2015.

Live coding as a performing art is dominated by coding performances that generate music and video improvisationally. In particular, music-based live coding is popular because it feels similar to live music performance.

Many tools have been developed for live coding as a performing art. Specifically, there are tools for live music coding such as Chunk [9], OverTune [10], TidalCycles [11], and Sonic Pi [12], as well as tools for live video coding such as Fluxus [13], LiveCodeLab [14], and Hydra [15].

2.3 Related Works

The use of glitches in artworks is not new, as seen in Nam June Paik's work "Magnet TV" [16], and as described in Sect. 2.1, glitches are already recognized as a form of expression in art. Some artists have published works on glitching in PNG, and we have referred to them in this study [17]. There is also a technique called data moshing, which causes errors during video playback, and research has been conducted on its effects and methods [18]. An installation work of glitches using live coding [19] has also been produced. This work is very interesting because it includes not only sound and images but also glitching by physical devices. Although not competing with this research, there is an attempt to use glitch art in contemporary dance [20]. This is very interesting because it uses glitch art based on sensor error to obtain dance motions as images for mixed reality.

The differences between these glitch artworks and this research are as follows.

- Manipulate (control) image glitches based on audio processing
- Intentionally cause glitches that match sound effects

In other words, the trigger for the glitch is audio processing (sound effects), which generates glitches in the image in real-time while enjoying the effect applied to the music. Live coding that generates video in real-time is not uncommon. Live coding systems that generate music have also been realized. Some live coding systems, such as Gibber [21], can generate music and video simultaneously. However, existing live-coding systems basically do not generate glitches, but rather generate video and music from scratch, and can be said to belong to the category of generative art [22]. Generative art and glitch art are distinctly different.

Another unique aspect of this research is the generation of regular glitches. This is related to the generation of glitches that are mapped to audio processing. That is, it is intended to generate glitches that match specific audio processing, not to generate glitches that are unpredictable. Although this point is debatable, one of the goals of this research is to use glitches to visualize audio processing. In other words, this study aims to enjoy music and glitch art at the same time, with the assumption that the two are interrelated.

3 Research Goals and Basic Approaches

3.1 Research Goals

In this study, our goals are as follows.

- Propose a method of expressing art across sound and image
- Create glitch art in an elegant method
- Performance of glitch art

The first goal is to propose a method of art expression in which sound and image processing are linked. Then, we will use an elegant method for creating glitch art. It's not difficult to create glitches. However, we are also concerned with the beauty of the method itself. We are not attracted to methods that are unreproducible, or just random. Nor do

we feel sympathy for using image processing software to create glitch art. Glitches are inherent errors, and glitches are distinctly different from image processing. In addition to the above, this research aims to make the creation of glitch art itself a kind of performance.

3.2 Basic Approaches

The basic approach of this study is as follows.

- Linking sound and image processing
- Image format should be PNG [23]
- Mapping audio processing to glitching
- Generate real-time glitch art images with live coding

First, the image is glitched by audio processing to generate a glitch art image. We assume that the format of this image is PNG. This requires a mapping between the audio processing and the glitching. The above is achieved in real-time using live coding. This is the basic approach taken in this research. We regard this approach as an elegant method to generating glitch art through meaningful audio processing, rather than generating glitch art through random destruction of data. In addition, by linking it with live coding, we will be able to see the generation of glitch art in real-time, aiming to turn glitch art into a performance. In order to generate glitch art through audio processing, we use a music-based live coding system in this research.

There is a reason why we chose PNG as the image format. Before that, the characteristics of PNG are listed below.

- File format that compresses and records image data
- Compressible without degradation
- Compression encoding of bitmap images
- Lossless compression and decompression
- Pre-filtering to improve compression efficiency by using image characteristics

In the PNG feature, we focused on filtering. We focused on the filtering process because we thought it would be possible to achieve regular glitches corresponding to the processing of each filter. The processing of each filter is as follows.

- **None**: Do nothing.
- **Sub**: Subtracts the byte to the left.
- **Up**: Differentiate from the byte directly above.
- **Average**: Differs from the average of the bytes directly above and to the left.
- **Paeth**: Calculates the Paeth value from the left neighbor, directly above, and above the left byte, and subtracts the difference.

Figure 1 shows the basic mechanism of glitching in this study.

In this study, glitching is used to display PNG images, based on the basic mechanism of PNG. This is based on the basic mechanism of PNG, which uses the characteristics of an image to perform filtering to improve compression efficiency before the ZIP compression process. This process is described above. When displaying a PNG image, the reverse process (reverse filtering) is performed according to each filter.

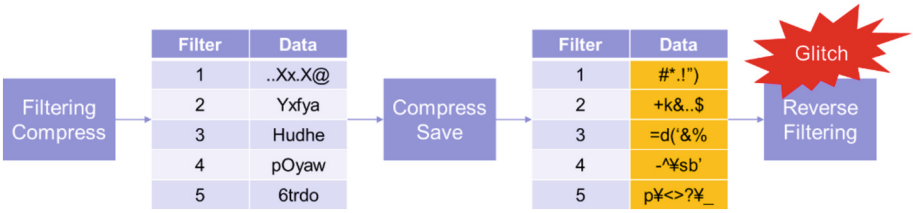

Fig. 1. Basic Mechanism of Glitch

Care must be taken when creating glitches in PNG images. That is, the PNG format must not be destroyed. In other words, if a PNG file is corrupted in a random way, the image itself cannot be displayed at all. Therefore, in this study, PNG images are loaded, the format is not broken, only the data portion is compressed simply, and the image is saved again. The key point is to skip the filtering process. This causes glitches when inverse filtering is performed. This mechanism generates glitches by inverse filtering even though the filtering process is skipped. In other words, the glitch art in this study is based on the data manipulation and misalignment described in Sect. 2.1.

The above glitching mechanism, which focuses on the filtering process of PNG, makes it possible to generate glitches with regularity as follows.

- **Sub filter action**: causes horizontal glitching
- **Up filter action**: causes vertical glitching
- **Average filter action**: causes diagonal glitching
- **Paeth filter action**: causes complex glitches

The next step is to map these glitches to audio processing. The purpose of this research includes making glitch art performances. Instead of viewing glitch art in a frame, we envision a performance in which glitches are continuously generated in the original image while the song is being played through live coding of music, and the glitches are shown in real-time. Sound and images seem to have nothing to do with each other, but if we consider images as patterns of color recognition based on the reflection of light, they have something in common. This is because both sound and light have the characteristics of waves, and both can be displayed as waveforms.

Based on the above, this study focuses on typical audio processing and links it to the glitching mechanism. We will employ the following audio processing.

- **Reverb**: creates a sense of spatial depth and spaciousness
- **Echo**: creates atmosphere and broadens the range of expression
- **Distortion**: creates a powerful, thick sound
- **Compressor**: makes overly loud sounds more consistent and easier to hear.

The above audio processing is not uncommon and is implemented in many music production software.

In this study, these audio processes are linked to the PNG filtering process, and glitches are generated at the time of audio processing. Table 1 shows the correspondence between PNG filtering and audio processing.

Table 1. Audio processing and PNG filtering corresponding and glitching

Filter	Audio Process	Glitch
Sub	Reverb	Horizontal glitching
Up	Echo	Vertical glitching
Average	Distortion	Diagonal glitching
Paeth	Compressor	Complex glitches

4 Implementation

In this study, a prototype system that generates glitches in PNG images in conjunction with Sonic Pi, a musical live cording system, was developed. The system configuration is shown below (Fig. 2).

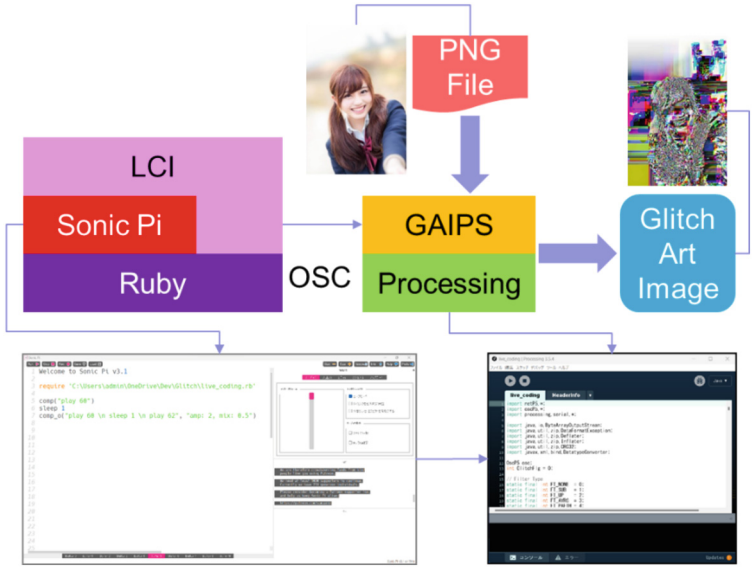

Fig. 2. System Configuration

Processing is used to generate glitch art images. The commands for generating glitch art images are implemented on Sonic Pi, a musical live coding system that is linked to, and Ruby, the language in which Sonic Pi is implemented, is also used for some of the commands. OSC communication is used to control glitching through live coding.

Live coding is performed on the Sonic Pi by writing a program in the Sonic Pi workspace similar to the Sonic Pi program and pressing the Run button (Fig. 2). To perform live coding for glitch control, the definition file (live_coding.rb) is first loaded with the require function of the Ruby language.

The definition file contains the instructions necessary for live coding to manipulate glitches. The instructions for controlling glitches are as follows (Table. 2).

Table 2. Glitch Control Commands and Usage Examples

Commands	Syntax	Example
revrb	revrb(cmd)	revrb(“play 60”)
revrb_o	rever_o(cmd, op)	revrb_o(“play 60 \n play 62”, “amp: 2, room: 0.5”)
ech	ech(cmd)	ech(“play 60”)
ech_o	ech_o(cmd, op)	ech_o(“play 60 \n play 62”, “amp: 2, decay: 2”)
dist	dist(cmd)	dist(“play 60”)
dist_o	dist_o(cmd, op)	dist_o(“play 60 \n play 62”, “amp: 2, distort: 0.5”)
comp	comp(cmd)	comp(“play 60”)
comp_o	comp_o(cmd, op)	comp_o(“play 60 \n play 62”, “amp: 2, mix: 0.5”)

These commands are written in the Ruby language and are read by the require function when Sonic Pi is run. Each command is a wrapper function for a Sonic Pi effect; Sonic Pi has a with_fx command for effects, which can generate sound effects.

The following is a simple program for Sonic Pi.

```
with_fx :reverb do      # reverb effect
   play 60              # make C sound
end
```

This program applies a reverb effect to midi note number 60, the C sound in the middle of the keyboard. The implementation of the glitch control instruction above is a function for wrapping such a Sonic Pi effect. In other words, the above code is equivalent to the following code.

```
revrb(“play 60”)
```

The effects by Sonic Pi have many options; the revrb command is a simple command with default options, but if you want to specify detailed options, use the revrb_o command.

```
revrb_o(“play 60 \n play 62”, “amp: 2, room: 0.5”)
```

The above code plays C sound and D sound at the same time, applying reverb with specified volume (amp option) and room size (room option), which is a measure of reverb. The above code is equivalent to the following Sonic Pi code.

```
with_fx :reverb, amp: 2, room: 0.5  do      # reverb with options
   play 60                # make C sound
   play 62                # make D sound
end
```

We show examples of glitch art generated in real-time using the above glitch control instructions below.

Fig. 3. Original image

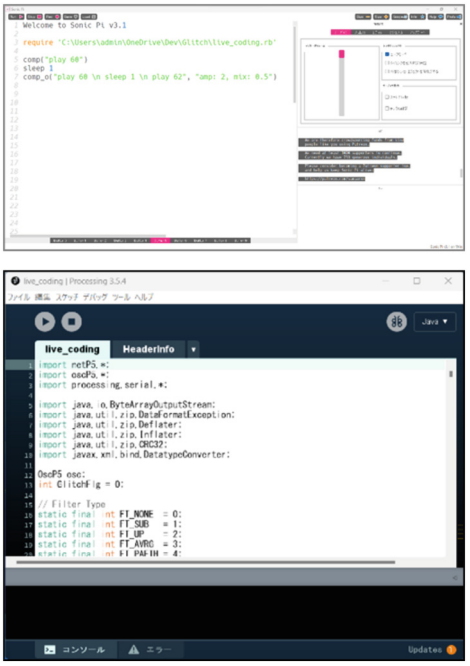

Fig. 4. Capture screen

Figure 3 shows a PNG image of the sample used in the experiment, and Fig. 4 shows a captured image of the execution screen of the system implemented in this study.

Figure 5 shows glitch art generated in real-time in conjunction with sound effects by the implemented system. The effects of the effects are clearly shown.

① Generation by **revrb** command: Horizontal glitching by **Sub** filter
② Generated by **ech** command: Vertical glitching by **Up** filter
③ Generated by **dist** command: Diagonal glitching by **Average** filter
④ Generated by **comp** command: Complex glitching by **Paeth** filter

The numbers ① through ④ are for convenience only.

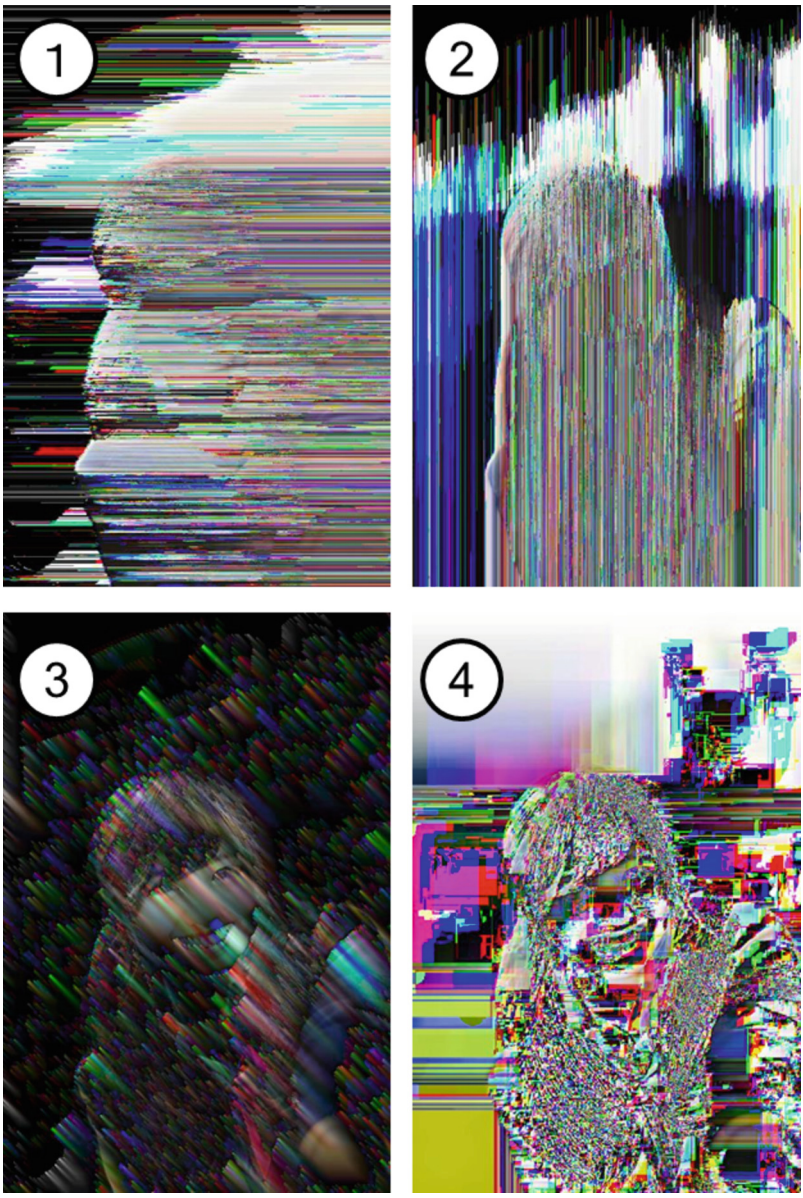

Fig. 5. Generated glitch art images

5 Conclusion

In this study, we explored the generation and performance of glitch art images using musical live coding. Specifically, we implemented and experimented with a prototype system that generates glitched art images in real-time by generating glitches in PNG

image data in conjunction with sound effects using the Sonic Pi: musical live coding system. In the experiments, we confirmed that our method can generate glitches with regularity, assuming the filtering of PNGs. This regular glitch generation is based on audio processing and is a result of the basic approach of this research. This research proposes a method of art expression across sound and image and is an attempt to explore the performance of glitch art.

This research has only just begun, and the following are planned for future development.

- implementation of glitches corresponding to audio processing options
- implementation of audio processing and corresponding glitching
- Practice and evaluation of glitch art performance

The prototype implemented in this study does not implement glitches corresponding to audio processing options completely. For example, Sonic Pi's reverb has an option to specify the size of the space, which allows the reverb to change, but the prototype does not implement the corresponding glitching. A possible effect would be to change the size of the glitch in the horizontal direction, but we will consider appropriate glitches for such audio processing options in the future.

Moreover, the audio processing implemented this time and the corresponding glitches are at most 4 types, and the performance of glitch art lacks impact. Sonic Pi has various sound effects, and we plan to design and implement glitches corresponding to them.

In addition to the above, we would also like to conduct a live-coded glitch art performance in front of a large number of people and evaluate the results. Based on the evaluation results, we will consider whether the way of enjoying glitch art proposed in this study can be established as performance art.

References

1. Betancourt, M.: Glitch Art in Theory and Practice: Critical Failures and Post-Digital Aesthetics. Routledge, New York and London (2016)
2. Moradi, I.: Glitch: Designing Imperfection. Mark Batty Publisher (2009)
3. McCormack, T.: Code Eroded: At GLI.TC/H (2020). https://rhizome.org/editorial/2010/oct/13/code-eroded-at-glitch/
4. Menkman, R.: The Glitch Moment(um), Institute of Network Cultures, No. 04. networkcultures.org (2011)
5. Heijer, E.: Evolving glitch art. In: Machado, P., McDermott, J., Carballal, A. (eds.) EvoMUSART 2013. LNCS, vol. 7834, pp. 109–120. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36955-1_10
6. Algorave (2012). https://algorave.com/
7. TOPLAP (2004). https://toplap.org/
8. International Conference on Live Coding (2015). https://iclc.toplap.org/
9. Chunk (2003). http://chuck.stanford.edu/
10. OverTone (2018). https://overtone.github.io/
11. TidalCycles (2009). https://tidalcycles.org/
12. Sonic Pi (2012). https://sonic-pi.net/
13. Fluxus (2005). https://monoskop.org/Fluxus_(software)
14. LiveCodeLab (2014). https://livecodelab.net/

15. Hydra (2019). https://hydra.ojack.xyz/
16. Magnet TV (1965). https://whitney.org/collection/works/6139
17. The Art of PNG Glitch (2015). https://ucnv.github.io/pnglitch/
18. Yuichi, I., Carl, S., Masashi, Y., Shinya, M.: Datamoshing technique for video art production. J. Soc. Art Sci. **13**(3), 154–168 (2014)
19. Witchcraft at Livecodera (2022). https://www.youtube.com/watch?v=XyUGDKd3lME
20. Stephan, J., Nuno, N.C., Raul, M.: Designing glitch procedures and visualisation workflows for markerless live motion capture of contemporary dance. In: Proceedings of the 7th International Conference on Movement and Computing, MOCO 2020, pp.1–8 (2020)
21. Roberts, C., Kuchera-Morin, J.: GIBBER: LIVE CODING AUDIO IN THE BROWSER. In: Proceedings of the International Conference on Mathematics and Computing, ICMC 2012, pp.64–69 (2012)
22. Matt, P.: Generative Art: A Practical Guide Using Processing. Manning Publications (2011)
23. Roelofs, G.: PNG: The Definitive Guide, 2nd edn. O'Reilly & Associates (2003)