Cellular Automata Musification Using Python and Magenta

Austin Franklin
Louisiana State University
afran84@lsu.edu

ABSTRACT

Life Like is an audiovisual project exploring the generation and musification [1] of Life-Like Cellular Automata modeled after John H. Conway's Game of Life using Magenta's GANSynth Colab Notebook. This cloud-based Jupyter notebooks contains pre-trained models for synthesizing timbre and audio files. A video and MIDI file of "live" cells are created from variable birth and survival conditions set before generation in Python.

1. INTRODUCTION

Cellular Automata is a mathematical model consisting of an array (usually two-dimensional) of cells that "evolve" according to the state of neighboring cells and a set of birth and survival conditions. The utility of Cellular Automata is their ability to produce complex patterns and shapes using simple rule sets, and these models can be used to simulate various complex real-world processes. "They were invented in the 1940's by American mathematicians John von Neumann and Stanislaw Ulam at Los Alamos National Laboratory [2]."

A cellular automaton is considered life-like if it meets the following criteria:

- The array of cells of the automaton has two dimensions.
- Each cell of the automaton has two states (conventionally referred to as "alive" and "dead").
- The neighborhood of each cell is the Moore Neighborhood consisting of the eight adjacent cell surrounding the one under consideration.
- In each step of the automaton, the new state of a cell can be expressed as a function of the number of adjacent cell, that are in an alive state and of the cell's own state.

[1] "The methodology of musification is concerned with both the abolute and programmatic elements of a work. It is a tool employed to assist with translation and organization of the symbols of a data set into organized musical sound. [1]"

2. VIDEO ANIMATION AND PYTHON

The video component of *Life Like* is written in Python and visualized using the matplotlib library to display, iterate, and animate subsequent generations. [2] It consists of a binary matrix representing the cells where 1 stands for alive and 0 for dead. The rules of the game can be specified in the format 'BXSY' where X and Y are the birth and survival conditions. Once the game begins, the number of living neighbors of each cell are returned before the game iterates one time. This is done for each 'frame', or iteration of the game (Figure 1).

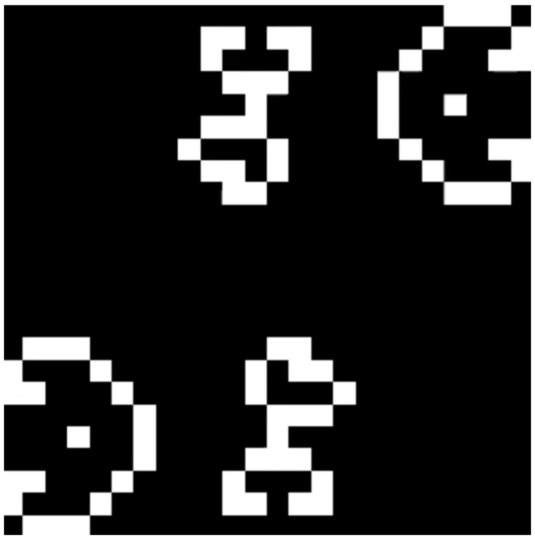

Figure 1. A single generated frame from the Life Life video

The original Game of Life as well as *Life Like* have a rule set of 'B3S23', meaning that if the game is run according to this rule set the outcome will be the same provided the grid size does not change. Several other rule sets were explored, including common alternative rule sets such as Day and Night (B3678S34678) and Fractal-Like (B1S123), but eventually abandoned due to their increased likelihood of becoming a combination of still lifes, oscillators, and spaceships.

The color map 'cubehelix' was used for each frame of Life Like because of its minimal and simplistic aesthetic. It also offers incredible contrast between the live and dead cells.

3. AUDIO AND MIDI

The gridSize, frameNumber, and frameSpeed in ms of the video are settings that can be changed to produce a greater

[2] https://matplotlib.org/

number of possibilities. For *Life Like* however, a grid size of 24 is used (the number of cells in both the X and Y dimensions) to map a pitch range of three octaves using an A Lydian scale [1]. For each iteration, the number of 'live' cells in each row are counted and their index positions are stores in a new array using the numpy.where() function. [3] This is done for all rows in each frame and for each frame in the entire game.

This new array is used to create a .mid file consisting of all of the 'live' cells each row in the matrix as a single chord. The MIDI file is written entirely in python using the MIDIUTIL library [3]. The tempo of the MIDI file is calculated in beats per minute (BPM) from the initial settings of the game, where each chord is a quarter note, using this formula:

$$60000/(frameSpeed/gridSize + 1) \tag{1}$$

Creating MIDI files in python takes a fraction of the time it takes to generate audio files. Given that the duration of *Life Like* is approximately 6 minutes, the decision to generate .mid files as opposed to .wav file was made primarily with regards to convenience. MIDI is also an extremely flexible file format where note pitches and lengths and can be adjusted and manipulated much more easily than audio files after being created.

3.1 MIDI

The MIDI file is then imported into Logic Pro X and represented using the John Cage Prepared Piano Soundset by Big Fish Audio [4]. This is a sample library of the original prepared piano and the only samples ever to be authorized by the estate of world-renown composer, John Cage. This particular collection of sounds was created for use in his magnum opus composition, Sonatas & Interludes (February 1946-March 1948). Additionally, a copy of the MIDI file is also used with the built-in Steinway Grand Piano instrument (Figure 2).

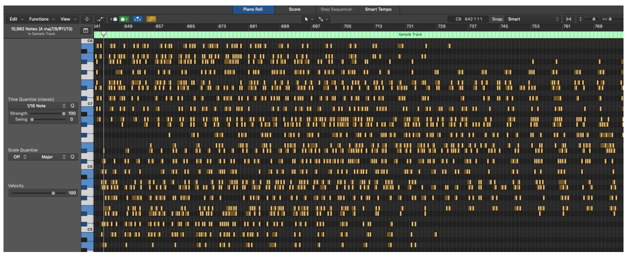

Figure 2. Piano Roll in Logic Pro X of the MIDI generated in Python

3.2 Audio in Magenta

The MIDI file was also used in Magenta's GANSynth Colab Notebook, which generates audio using Generative Adversarial Networks. [5] The model takes the pitch from the MIDI as a conditional attribute and learns to use its latent space to synthesize different instrument timbres by encoding numerous attributes related to that timbre [4]. These timbres can be constant or they can interpolate linearly over time to create smooth transitions and intermediary timbres [5].

In particular, the GANSynth demo was run using a random instrument interpolation set to 10 seconds per instrument. This means that every 10th second would generate a single instrument timbre and the time in between would be used to interpolate from the previous to the next instrument.

Interestingly, the final generated audio file did not contain every pitch from the file that was uploaded to the model. Many gaps, or 'rests', were present throughout the entirety of the file. This is most likely due to the fact that the MIDI file contains 10,962 individual MIDI events and the model was simply not able to either analyze or generate polyphonic density of this level.

4. FINAL EDITING

The video and newly created audio and MIDI are then imported into Reaper and layered together to create the finished product. The various audio tracks are layered together and faded in and out, causing the sounds to gradually transition between one another and each take a 'leading role' throughout one of the major sections of the piece.

The final step is rotating the video -90 degree inside of the DAW. This moves the 'live' cell indexes in each row to columns, allowing pitch register to be visualized low to high on the new Y axis, and time along the new X axis. This turns the created video with the layered audio into a beat sequencer.

5. CONCLUSIONS

Life Like is a project that explores the musification of John H. Conway's Game of Life. The video animation and MIDI were generated in Python according to the original rule set of the game (B3S23) and an A Lydian scale spanning three octaves. The MIDI file was then used with Magenta's GANSynth to generate and interpolate instrument timbres. The final product was created by layering all of the video, audio, and MIDI files together in Reaper.

6. REFERENCES

[1] A. Coop, "Sonification, Musification, and Synthesis of Absolute Program Music," 2016.

[2] T. E. of Encyclopedia Britannica, *Cellular Automat*. Encyclopedia Britannica, 2021.

[3] M. C. Wirt, "MIDIUtil Documentation," 2018.

[4] T. P. Van, T. M. Nguyen, N. N. Tran, H. V. Nguyen, L. B. Doan, H. Q. Dao, and T. T. Minh, "Interpreting the Latent Space of Generative Adversarial Networks using Supervised Learning," 2021.

[5] A. Brock, J. Donahue, and K. Simonyan, "Large Scale GAN Training for High Fidelity Natural Image Synthesis," 2019.

[3] https://numpy.org/doc/stable/reference/generated/numpy.where.html
[4] https://www.bigfishaudio.com/John-Cage-Prepared-Piano
[5] https://colab.research.google.com/notebooks/magenta/gansynth/gansynth$_{d}emo.ipynb$