Dorian D.

<u>Dodo's Drum Chopper</u>

<u>Introduction and Historical Context:</u>

"*The basis of computer work is predicated on the idea that only the brain makes decisions and only the index finger does the work.*"

- *Brian Eno*

For the past thirty years, musical genres and trends have become increasingly linked to new technologies. In 1980, the TR-808 Rhythm Composer was released and became the earliest programmable drum machine. Three years later, its little brother, the TR-909 was also released. Both machines became a pillar of the music produced throughout the 80's and 90's, and the foundation of new genres such as techno and house.

In the early 90's, a new aggressive musical trend called jungle started to gain momentum. Jungle originated in London and was derived from a popular genre at the time called breakbeat, also known as hardcore. At its heart, jungle can be described as very fast, often reaching 175 beats-per-minute. It is based on the sampling, chopping, and layering of drum loops. One break, the Amen Break notably became one of the most heavily used samples at the time. This break, which first appeared in The Winston's "Amen, Brother," became popular in hip-hop during the late 80's. Back in the day, it was slowed and pitched down to around 95 beats-per-minute, and then used as the basis of a track. Jungle, however, totally revolutionized this trend.

Prominent Jungle artists would often speed the loop up to around 175 beats-per-minute, and then cut it into 16 beats. These were then played in a different order, filtered, and transposed to create a very different drum break. Notable Jungle tracks that sample the Amen Break are LTJ Bukem's "Atlantis (I Need You)" and Aquarius' "Dolphin Tune." Popular samplers used to create the typical grungy Jungle drum sound were the Akai S900 and the Akai MPC 60.

<u>Motivation and Objectives for my Project:</u>

For the past five years, most of my time off has been spent creating music using Ableton Live. It is something I will probably never stop doing. Yet, when trying to create a Jungle track, I get bored and tired. I often find myself plotting a transposition curve and copying and pasting parts of a drum loop for hours on end, only to find myself with a semblance of a Jungle beat, and very little variation throughout the track. The following figure shows what I typically do with a sampled drum loop to create a Jungle-like drum pattern. This is only for one bar. Imagine repeating this process for every loop in a five to six-minute track!

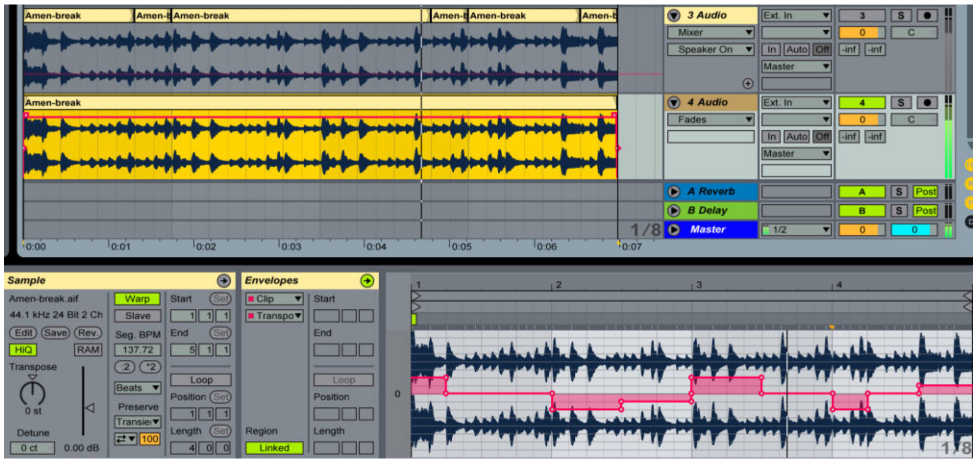
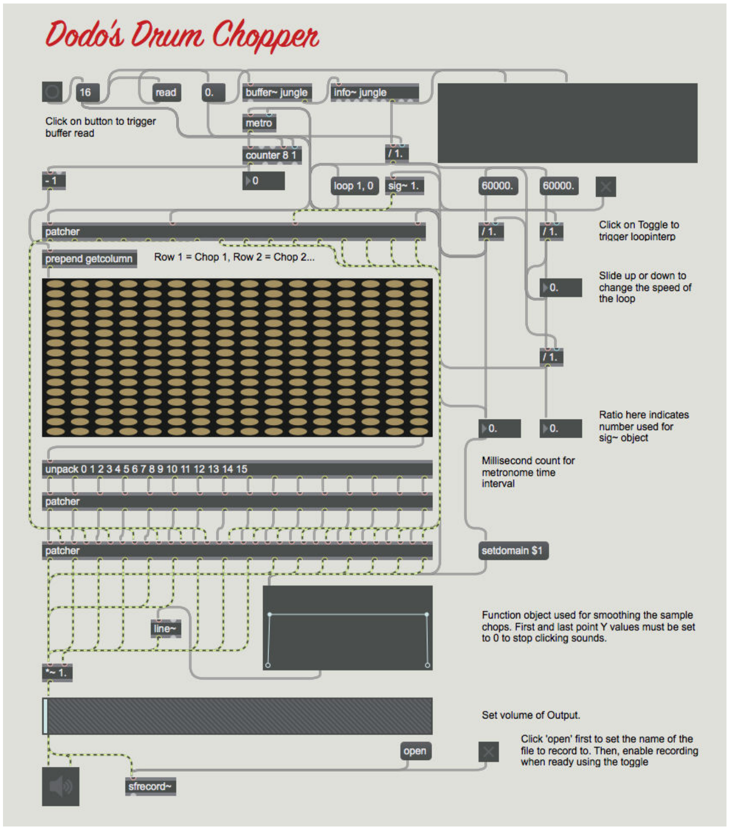


Figure 1: My Ableton Jungle Sampling

I therefore decided that I would try to solve this problem for my Final Project, especially since Max MSP seemed to be an ideal platform to create a sampler tailored to my needs.

The following figure is a screenshot of what this sampler now looks like. It can be downloaded in this Github repository.


Dodo's Drum Chopper

Click on button to trigger buffer read

Row 1 = Chop 1, Row 2 = Chop 2...

Click on Toggle to trigger loopinterp

Slide up or down to change the speed of the loop

Ratio here indicates number used for sig~ object

Millisecond count for metronome time interval

Function object used for smoothing the sample chops. First and last point Y values must be set to 0 to stop clicking sounds.

Set volume of Output.

Click 'open' first to set the name of the file to record to. Then, enable recording when ready using the toggle

Figure 2: Dodo's Drum Chopper

Dorian D.

The Project Design:

Dodo's Drum Chopper, which was named after the nickname I was given at birth, is a sampler that can be used with Max MSP or Max4Live. My goal with this project was to create a drum sampler that would be extremely easy to use and visually appealing.

To start chopping, one clicks the button object at the top of the screen. This button triggers a buffer read, which lets the users pick the loop they want to chop up. Once this has been done, the Max Patch automatically takes care of chopping the sample accordingly. The patch notably calculates the start and stop cue points for each drum chop, the metronome time interval, and the domain of the function object at the bottom of the patch. All of these parameters are essential to this patch. Figure 3 shows the sub patch which is then used for storing all 16 drum chops. The groove objects continuously loop a designated section of the input (1/16th of the input) which is then sent to the gates in Figure 4. These are manually activated by the user using the matrixctrl object.

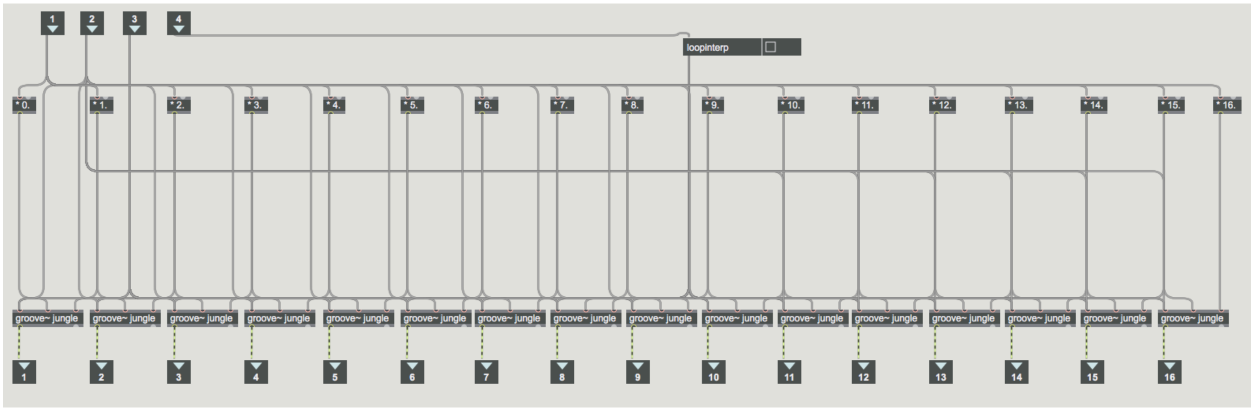


Figure 3: Sections used for each Drum Chop

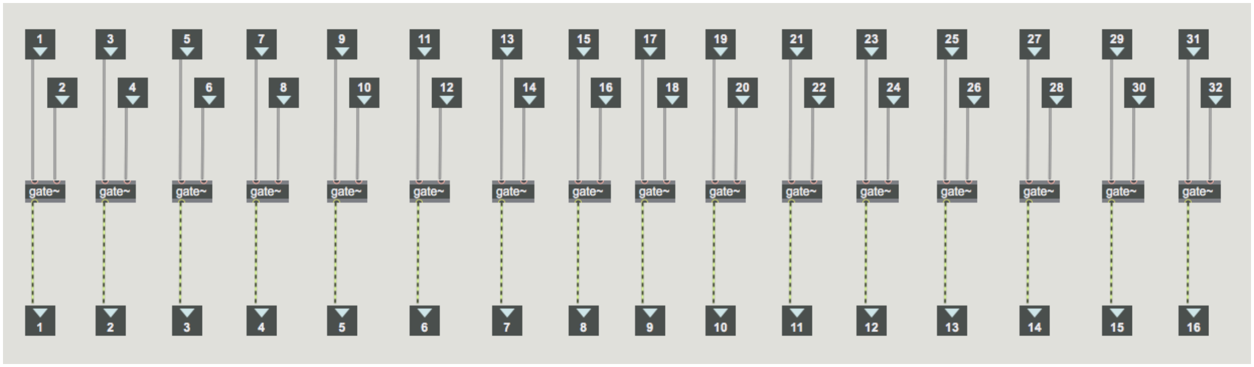


Figure 4: The Selection Process of each Drum Chop

Each row of the matrixctrl object controls a particular drum chop (row 1 controls drum chop 1, row 2 controls drum chop 2…).

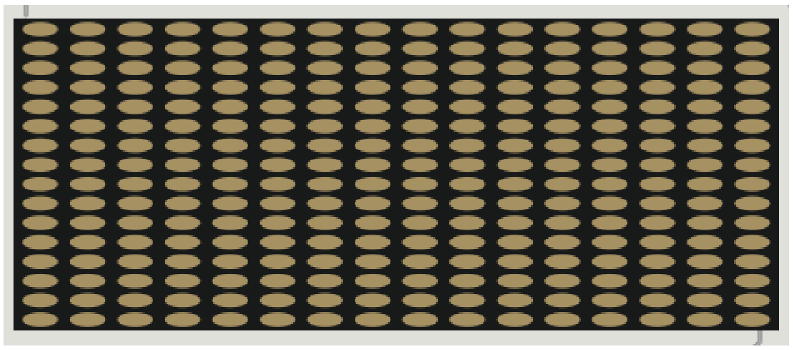


Figure 5: The Matrixctrl object

The count in the counter object is incremented each time the metronome outputs a bang. This count is then used to output a column of the matrix object. The counter is bounded by 0 and 15 in order to be in sync with the 16 chops used by this patch. All users then need to do is click on the row / column combination that controls the chop they want to play on each 16th note. The patch then continuously plays the loop generated by the user.

The user can also play with the speed of the bar being generated by sliding the first number box under the waveform object. This number box shows the BPM of the loop that was used as input depending on the length of each chop, and can be used to increase or decrease the speed of the resulting loop. Note that the pitch will shift accordingly, and that the actual BPM is only shown if the input is one bar long.

The other number boxes should not be changed or played with by the user. They serve as an indicator of the important parameters used by some objects in the patch, and their comments indicate their purpose.

The last element that the user should change according to taste is the function object at the bottom of the patch. This object is essential as it gives the user the opportunity to drastically reduce the clicking that can be heard when various loops in each groove objects are played succinctly. The most important points to have in the function objects are the first and last one. Their y-values must be set to 0 in order to eliminate sudden amplitude discontinuities between successive sound chops. The x values for both points must be 0 and the maximum x value of the domain. Note that the domain is automatically set when the time interval output to the metronome is changed. Different speeds and samples require different sample smoothening to avoid clicks, which is why the user can change the function being output by the function object. Changing the function can also be highly creative, as one can change the attack and decay of the samples before output!

These are the important objects of the patch. Experimenting with different loop chops and speeds can lead to very interesting results and very different patterns that can then be used in any track. Recording is enabled at the bottom of the patch for future use of the sounds generated by the user!

Dorian D.

__Issues that Proved Challenging:__

A few elements of the patch proved to be very challenging.

First, finding the right objects for chopping up samples was a real nightmare. I started off using the waveform object as the main sound source as it provided me with real-time visuals of the portions of the loop being played. I managed to chop the sample quite well, but making the patch easy to use proved to be a nightmare afterwards. I therefore opted to use the groove object instead.

Once I had my samples loaded into various groove objects, I managed to set up the matrixctrl object pretty fast. By that time, I thought I was almost done, and quite happy with the results. Little did I know that I was about to encounter the two most difficult aspects of the patch.

The first problem I encountered had to do with time. Syncing the metronome, the sig~ object, and the user-set BPM proved to be very challenging, as I had to find a ratio for the sig~ object that would match the metronome's time interval changes. This part of the project was marked by lots of debugging, and the extensive use of print objects and number boxes.

The second problem I encountered was linked to sound clicking, as the groove objects all have very different amplitudes at the start and end of their loops. I needed to find a function that would briefly and inaudibly set the amplitude of each groove object to 0 at the start and end of each loop. I tried numerous functions as inputs for the line~ object, but most did not prove to adapt to speed changes very well (attack and decay would not shorten accordingly when increasing the speed of the samples). In the end, the function object does this to perfection, and is adaptable and easy to use. Users can change the attack and decay of each groove object depending on the clicking they hear in real-time, allowing for better control of the sound being output.

Overall, however, this patch was extremely fun to create, and these problems proved to be extremely beneficial for my understanding of Max MSP and sampling.

__Moving Forward:__

This patch already provides me with a solid base for chopping samples in a Jungle-style way. It is, however, not finished as there are still a few things left to do in order to make it the ultimate Junglist Max patch.

Before reading the rest of the wrap-up, one might want to give Baraka's "I'll be there" a listen. In my opinion, this track is the apex of Jungle style drum chopping. The drum chops are a lot shorter than those that can be created by my patch, and they are all transposed and filtered heavily for a unique result.

The following features must therefore be added to my Max Patch. First, I need to create a 32-chop version of the Max Patch. This should be very easy, as the groundwork is the same as the current sampler. Secondly, I need to add filtering, transposition, and gate effects to the patch. The patch should be able to generate different filter and transposition parameters depending on the count index, so as to have lots of variation in the loop being generated. The gate can then help with the conciseness and sharpness of the peaks of each chop, and potentially enhance the swing and flair of the new loop! This technique was heavily used by artists such as LTJ Bukem (in his "Journey Inwards" album) when sampling old jazz records. These are the two ideas I have at the moment to better my Max Patch, although the feature possibilities are pretty endless! Why not add a delay effect, or randomize the order of the chops for even more surreal results?

With that being said, I will leave the reader to experiment with this patch. I hope you have fun like I do every time I open it! I have also included the Amen Break in the Github repository for the reader to experiment with the patch. I recorded it myself.