Assignment #1: Monophonic Sonic Challenge

Pedro Esteves

Faculty of Engineering of the University of Porto

up201705160@fe.up.pt

I. Introduction

Sound synthesis techniques can be used to create music or parts of them. One great example of these techniques in the industry can be seen in On the Run from Pink Floyd's Dark Side of the Moon.

This report describes a way to reproduce a snippet from this music, as well as other experimentations made to reach the final result.

II. Music Notes

One of the first steps to reproduce a piece of music is to find the notes used to make it. In this case, it was not different.

A. Find the Notes

On the Run is composed of 8 notes, E2 G2 A2 G2 D3 C3 D3 E3, or, in MIDI values, 40 43 45 43 50 48 50 52.

Since the *tempo* of the music is fast, it is almost impossible to find these notes by hearing the song. So, to find them, I tried some different paths:

- Analyse the frequencies of the song, by creating a sub-patch, inside the main patch, to read the WAV file and display its spectrum (sub-patch *open_original_file)*;
- Analyse the frequencies of the song by importing the WAV file to Reaper and using different plug-ins to display its spectrum;
- Search online for the song's notes;
- Search online for a MIDI Sequence of the song, and analyse its values.

All these approaches led more or less to the same notes, with the first two being harder to decipher.

B. Create Sequences with Pure Data

I am very new to Pure Data, so concepts such as creating sequences in this language were new to me. I knew that these shouldn't be very different from other languages, so I started looking for loops in Pure Data.

I found the object *table* and *tabread* that could serve this purpose. I created a new sub-patch, called *notes*, with a *table*, which received a message with the sequence of notes, a counter made-by-hand (defining variable, incrementing it, and using the remainder operator to loop), and a *tabread* object to get the MIDI values from the *table*. This loop was also triggered using a *metro* with a 91ms interval. By hearing the song, I discovered the interval was around 90ms, but then I found online that the song's *tempo* was 165 bpm ((165 bpm / 60 s) / 4 notes per beat ≃ 91ms), which made me increment this value by 1.

After learning more about Pure Data *arrays*, I restructured the sub-patch *notes* to use arrays instead of the table. In this refactoring, I also changed my counter to the *counter* object.

III. Waveform

A sound signal can have one of the following waveforms: (1) sine, (2) sawtooth, (3) square, and (4) triangle.

Since this is exploratory work, I decided to play around with all four waveforms, trying to approximate my work from the original one. After exploring them, I realized that the square waveform was the most similar to the one used to record On the Run, so I chose that for my work.

IV. Synthesis Techniques

Through this work, I explored all the synthesis techniques discussed in the classes so far. In the following sections, I explain my experimentations with these techniques and their impact on the final result.

A. Subtractive Synthesis

This technique can be applied using filters (low-pass, high-pass, band-pass, or notch). In this

case, I explored the first three filters, with emphasis on the band-pass filter.

Using the band-pass filter, I could recreate the second effect (the final one) by increasing its center frequency or the Q value. There are two toggles to enable this effect, the red and yellow ones, and this work can be seen inside the *filters* sub-patch.

B. Amplitude Modulation

Amplitude Modulation was one of the first techniques I explored by adding a slider with values between 0 and 1 and multiplying its value by the result of the square waveform. This way, I could control the volume of the sound by modulating its amplitude.

Later, I added a *line* object to generate a ramp, hence avoiding the start buzzing. The line object is triggered by a toggle that receives *start* or *stop* messages.

When I was exploring the first effect of the sound snippet, I remembered I could explore the Amplitude Envelope Shape by changing the values for Attack, Decay, Sustain, or Release. This was when I realized that decreasing the release and increasing the decay could produce a similar effect to the one present in the snippet. So, I created two green toggles to control this effect.

To make the code cleaner, I separated the amplitude modulation into a new sub-patch, which receives the output from the *square* object and sends the modulated signal.

C. Ring Modulation

Ring Modulation consists of the multiplication of two signals, a carrier signal, and a modulating signal. Assuming the previous signal passed through the filters, as the carrier signal, I tried to recreate the first effect from the snippet.

Although it produces a similar effect, it was not as satisfying as the envelope effect, which led me to exclude this effect from the final attempt but keep a violet toggle to control it.

D. Frequency Modulation and Additive Synthesis

Frequency Modulation (FM) was the second technique I explored. Usually, this modulation works with a constant modulation frequency, so I tried different values without success. Finally, I decided to get the carrier frequency and multiplied it by the deviation value, which turned the frequency modulation more into an additive synthesis since the carrier signal doesn't change according to the modulation frequency.

V. Record The Attempt

To record my attempt, I decided to learn more about how to write files in Pure Data. I already knew that I could read files using the *readsf~* object, so writing to files shouldn't be very different.

While researching, I found the *writesf~* object, which could record my attempt. This object requires an open message to open the file for writing purposes and *start* or *stop* messages to control this operation. This object also has one integer parameter to specify the number of audio channels, requiring inputting the audio signal to each inlet as the *dac~* object.

The final recording uses a combination of the envelope, FM, and center frequency effects. Other attempts can be found in this folder.

VI. Conclusions And Future Work

Although I haven't made an exact replica of the sound snippet provided, mainly due to the snippet's first effect, with the help of this assignment, I have learned a lot more about Pure Data and Sound Synthesis techniques. After spending several hours working on this assignment, I think I achieved a decent result and a good understanding of these concepts. With the help of this work and the concepts learned, I can now create sounds for several applications.

In future work, I could improve the first effect of the snippet and create the rhythmic component. Working with an analog synthesizer could also help understand these concepts better and reproduce *On The Run* more accurately.