Ministry of Education and Science of Ukraine
NATIONAL UNIVERSITY OF KYIV-MOHYLA ACADEMY
Department of Informatics Faculty of Informatics

Euclidean Algorithm for Sound Generation

Supervisor
Head of the Informatics Department,
associate professor, candidate of physical and mathematical sciences
Horokhovskyi S.S
(signature)
“____” _____ 2021 p.

By student 1st year
Master Degree Program
122 «Computer Science»
Laiko A.V
“____” _____ 2021 p.

Kyiv 2021

Ministry of Education and Science of Ukraine
NATIONAL UNIVERSITY OF KYIV-MOHYLA ACADEMY
Department of Informatics Faculty of Informatics

CERTIFY
Head of the Informatics Department, associate professor, candidate of physical and mathematical sciences
_____ Horokhovskyi S. S.
(signature)
„____”_____2021 p.

PERSONAL ASSIGNMENT
for course work

To student Laiko Artem Volodymyrovych faculty of informatics 1-year master programme
Thesis «Euclidean Algorithm for Sound Generation»

Output:
Text part content of coursework:
An individual task
Calendar plan
Annotation
Introduction
Chapter 1: Music Theory and Sound Generation
Chapter 2: Euclidean Algorithm
Chapter 3: Euclidean Algorithm for Sound Generation
Chapter 4: Practice
Conclusion
References

Issue date „___” _____ 2021 p. Supervisor _____
(signature)
Task received _____
(signature)

Thesis: Euclidean Algorithm for Sound Generation

Calendar plan of coursework execution:

№	Stage name	Deadline	Note
1.	Assignment received	29.10.2020	
2.	Literature lookup	30.10.2020	
3.	Introduction into resources and following development of a draft intro	09.11.2020	
4.	Introduction part written	11.11.2020	
5.	Familiarising with Music theory relationship to math and further development of the first chapter	17.12.2020	
6.	Lookup for Euclidean algorithm development history and possible ways of application	8.01.2021	
7.	Testing the Euclidean algorithm for rhythm generation	25.02.2021	
8.	Further development of second and third chapters	29.02.2021	
9.	Testing and development of VST3 plugin using the Euclidean algorithm as wave shaping helper	17.03.2020	
10.	Course work amendments according to the supervisor feedback	15.04.2021	
11.	Created final presentation and formatted layout of course work	9.05.2021	
12.	Presentation of the course work	17.05.2021	

Student Laiko A.V.

Supervisor Horokhovskyi S.S.

“_____”_____

ANNOTATION

This course work aims to research possible ways of Euclidean algorithm application and influence for sound generation process, as well as mathematical basis of sound and sound waves.

The practice part applies obtained knowledge to develop a VST3 plug-in for sound wave morphing with the Euclidean algorithm application.

Contents

INTRODUCTION

The Euclidian algorithm has been known by humanity for more than 2000 years so far, and its further development, together with understanding the ways it could be used, lead to many new inventions and improvements in various sciences.

In 2005 the application of the Euclidean algorithm found its way in music, especially, explaining and generating traditional musical rhythms, using the simple concept of finding the greatest common divisor.

1. Music Theory and Sound Generation

To get an understanding of what music theory is and how sounds could be generated, let's first start with the definition of sound. Sound is a vibration of any surface that creates an acoustic wave through a medium which usually is air, but could be any liquid or even soil. We as human beings are given the ability to decode the changes in air pressure in a wide range from 20 Hz up to 20 kHz, and depending on oscillation speed recognise those as continuous sounds.

1.1 Pitches and Notes

The frequency of the sound wave is called «pitch», some of those pitches are more likely to be caught by our sensory system and we know them as notes. Those notes in western music notation have names as follows:

Note	Name	Base Frequency
A	La	13.75
B	Si	15.43
C	Do	8.17
D	Re	9.17
E	Mi	10.3
F	Fa	10.9
G	Sol	12.24

Table 1. Notes with names and base frequencies.

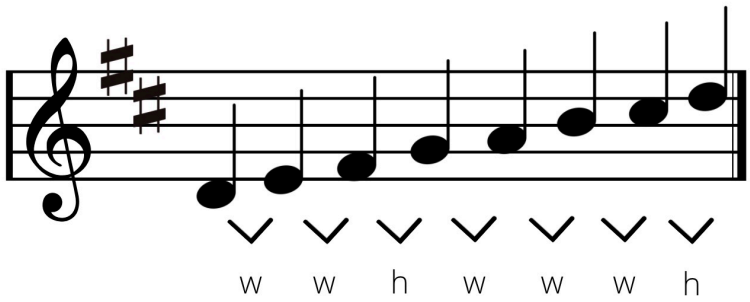
Each of those notes has multiple pitches, which repeat themselves and called «octaves». The process of calculating the frequency of a certain note in a certain octave could be described as:

$$x = n \times 2^{b+1}$$

Where x is the pitch of the note in the desired octave, and n is a base frequency for this note, and b is the desired octave. For example, note A (la) fourth octave will have a frequency of 440 Hz.

1.2 Scales and Modes

After defining what notes are, we able to assume, that they can be arranged in a variety of systems, with intervals in frequency and relations. Those arrangement systems called «scales». In western music theory octave is usually consists of 12 tones and named «chromatic scale». Two notes (tones) usually have a middle tone which is half step away from them and can be named either «sharp» or «flat». Creating different patterns among the set of 12 tones, arranging them by whole-tone or semi-tone creates different scales, which utilising same notes, but having different relation between them creates distinct sound.



Picture 1. D major key with steps between notes.

Each scale that is used for a composition is usually determined by a key signature, which helps to define all the pitches that define a certain scale. During the course of composition set of pitches could change and introduce new

one's, that are not part of the initial scale, this called «transposition». Sometimes transposition used as an artistic function, but sometimes to accommodate composition for a certain instrument or artist, which is not able to perform a piece of music in the original key.

1.3 Consonance and dissonance

Consonance and dissonance are the qualities based on the subjective judgement of intervals between tones. They define relations inside and outside the key. Each note in key can be marked as a «step» and depending on the interval if it's a whole-step or half-step between two neighbour notes, dissonance or consonance will be heard.

Consonant interval is the one, that stable and gives a subjective feeling of calmness and completion. On the other hand, the dissonant interval doesn't have such stability and wants to be resolved back into the consonant interval. Those two fundamentals are used in any music to create tension and release, which creates movement inside of a composition.

1.4 Rhythm

The music consists of several pitches, which have relations and also keys to which those pitches belong, and any given composition also has rhythm. Rhythm is the movement of the musical piece and consists of sets of sounds and silences during the course of composition.

Time measurement of a complete music composition begins with the definition of BPM (beats per minute). As well as having the definition of BPM the time signature for the music piece is important as well because it defines how many pulses the bar consist of. Common time signatures are next:

- $\frac{4}{4}$

- $\frac{5}{4}$
- $\frac{6}{8}$
- $\frac{7}{8}$

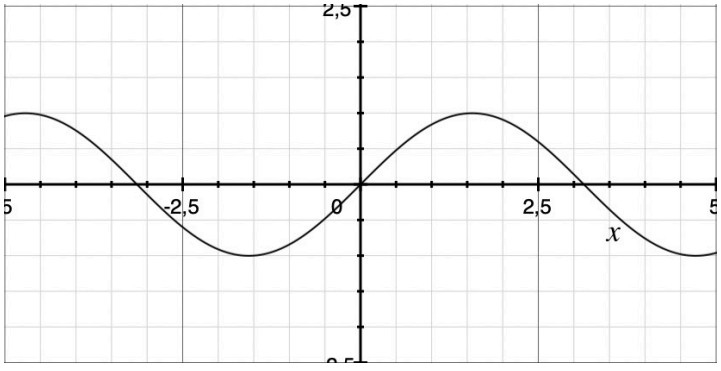
Those time signatures could be read as 4 occurrences of quarter-note in one bar, which means, that one bar can have 2 4th notes and 4 8th notes, and in this way, using different note-length together with certain time-signature allows for the composer to achieve unique sounding of a music piece, due to variation in rhythmical patterns and change in pulses.

1.5 Sound Generation

All the traits of a sound and knowledge of music theory based on the basic concepts of what sound is, how it could be generated, and the physics behind each sonic component in music.

1.5.1 Sound waves

Sounds are the collection of waves that create unique sounding, depending on the shape of the wave. Each wave could be treated as a function, the basic sound wave available is $sin(x)$, because of its repetitive shape and predictable period of oscillation it creates a plain buzzing sound, without any sub-harmonies.



Picture 2. Sin(x) waveform.

Having a rough look at the $sin(x)$ function, we can determine *x* axis as Time measurement, and *y* axis amplitude as the change in pressure. Change in pressure determines by the difference of average local pressure in a given medium and sound wave influence on it.

1.5.2 Sound waves and Pitch

Pitch in human perception is being perceived as a «low» or «high» sound and represents the repetitive nature of the vibration that defines the sound. When it comes to the relation between pitch and frequency of a sound wave, it is being defined as the slowest vibration in the given sound and usually called a fundamental harmonic, for complex sounds pitch perception can be different, due to individual perception.

Pitch is being decoded by the human brain in the pre-conscious examination of sound waves and balance in frequencies and loudness between them. Most of the attention usually being drawn by recognised harmonics that build the sound.

When multiple sound waves persist in given composition, they will sum up and the resulting sound will consist of the unified and continuous sound wave, because of the wave summing, some of the instruments that were used to produce the sound might be cancelled by the difference in phases of given sound waves. Usually, when this occurs, the loudest or «highest» pitch will dominate over the weaker ones and will cancel most or all of the characteristics that are created by weak pitch.

1.5.3 Basic waveforms

Most of the instruments and ways to create the sound are exploiting 4 basic waveforms, which usually are oscillated using the basic waveform of $sin(x)$.

Those waveforms are next:

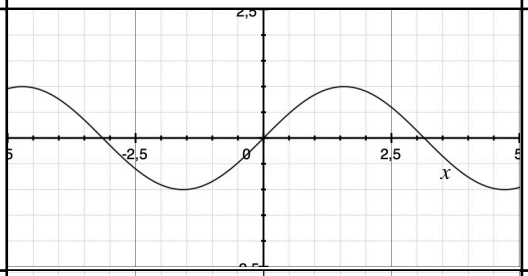
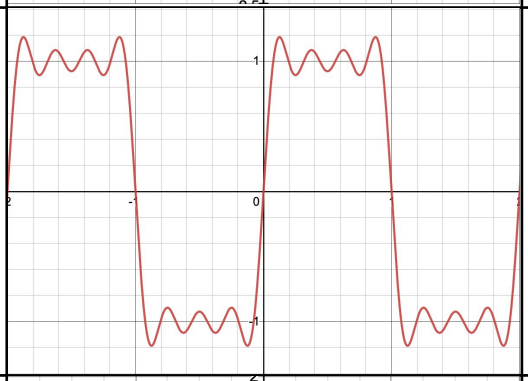
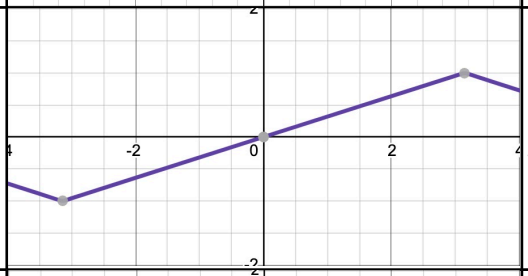
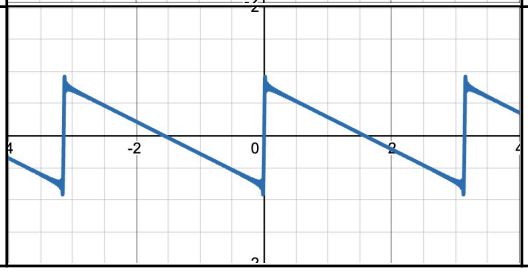
Name	Image	Formula
Sine wave		$y(t) = sin(t)$
Square wave		$y(t) = \frac{4}{\pi}\sum_{k=1}^{\infty}\frac{\sin\left(2\pi(2k-1)ft\right)}{2k-1}$
Triangle wave		$y(t) = \frac{2a}{\pi}\arcsin\left(\sin\left(\frac{2\pi}{p}t\right)\right)$
Sawtooth wave		$y(t) = \sum_{n=1}^{\infty}\frac{\sin\left((2n)\,t\right)}{2n}$

Table 2. Waveforms and formulas.

As we see in the given table, the basic sine function could be used as part of the Fourier series, to modulate any other given waveform. In that way,

many modern digital instruments reproduce sound, by taking the base sine waveform and modulating the waveform that is defined by a certain shape. Due to the modulation nature, the sine wave introduces harmonics, which are the glitches in a waveform that are not initially present in a given shape, but acquired due to modulation accuracy. Usually, harmonics are used as a technique to bring new sounding to the instrument and morph the sound.

2. THE EUCLIDIAN ALGORITHM

The Euclidian algorithm is the algorithm first described by the ancient Greek mathematician Euclid in his work «Elements». This algorithm is used to calculate the greater common divisor of two natural numbers. The result of its work is the largest natural number, which divides two numbers without remain.

This algorithm can also be applied for three or more numbers, in this case GCD will be the product of prime factors common to all the numbers:

$$gcd(a, b, c) = gcd(a, gcd(b, c)) = gcd(gcd(a, b), c) = gcd(gcd(a, c), b)$$

2.1 Algorithm description

Euclidean algorithms work in a series of steps in which the output of the step is used as an input for the next one. Let's assume that i is an integer that will count steps taken by the algorithm, which starts with zero value, where the initial step corresponds to $i = 0$ and the following steps are $i = i + 1$.

Steps start with two remainders r_{i-1} and r_{i-2} with a value > 0. Due to the requirement of the algorithm to steadily decrease with each step, the remainder r_{i-1} has to be less than r_{i-2}, with the goal for i-th iteration to discover the quotient q_i and remainder r_i that will satisfy the following equation:

$$r_{i-2} = q_i r_{i-1} + r_i$$

And that following clause is true: $0 \leq r_i < r_{i-1}$.

Initial step $i = 0$ has both remainders r_{i-2} and r_{i-1} equal to given a and b, but next step $i = 1$ remainders equal b and the remainder r_0 of the first step.

Due to the fact, that remainders decrease with every step, but cannot be < 0, the last remainder r_N has to be zero and it will be treated as a stopping

point for the algorithm. Final $\neq 0$ remainder r_{N-1} is being used as a result and defined as the greatest common divisor of *a* and *b*.

2.2 The Euclidean algorithm application for rhythm generation

The Euclidean algorithm can be used as the way to define rhythm in music composition by creating patterns for hits and silences in a given size, as been described in the paper «The Euclidean algorithm Generates Traditional Musical Rhythms» by Godfried Toussaint.

The idea is simple to define the musical size for cycle rhythm and place the pulses of music (tones that will introduce most tension) by application of Euclidean algorithm. In this case, number of pulses has to be less than number of steps in cycle. Let's assume we want to place the 5 pulses over the size of 16 steps, this will give us musical size combining $\frac{4}{4}$ and $\frac{3}{4}$, and by application of Euclidean algorithm we get resulting pattern like this (1 is moment of pulse, and 0 is moment of silence):

1 step: [1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0]

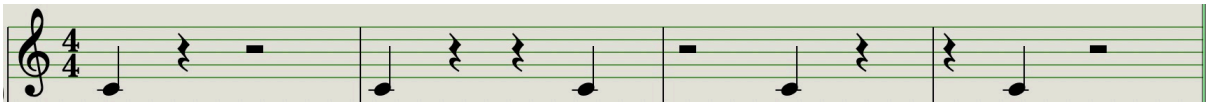
2 step: [1 0] [1 0] [1 0] [1 0] [1 0] [0] [0] [0] [0] [0] [0]

3 step: [1 0 0] [1 0 0] [1 0 0] [1 0 0] [1 0 0] [0]

4 step: [1 0 0 0] [1 0 0] [1 0 0] [1 0 0] [1 0 0]

Final result: [1 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0]

We can put this series of pulses onto the stave and we will have next pattern:



Picture 3. Euclidean rhythm generated over E(5, 16).

This rhythm combines multiple sizes, with 4/4 part over the first bar, and the following 3 bars are creating unstable movement due to stepping over 3/4 size.

In general, the Euclidean algorithm could be in hand for application in many parts of the sound generation process, coming first from the definition of pulses inside music composition, but in the same fashion, it applies for more complex signal tuning for oscillators and LFOs. Those ways of application and examples are described in the following chapters.

3. The Euclidean algorithm for Sound Generation

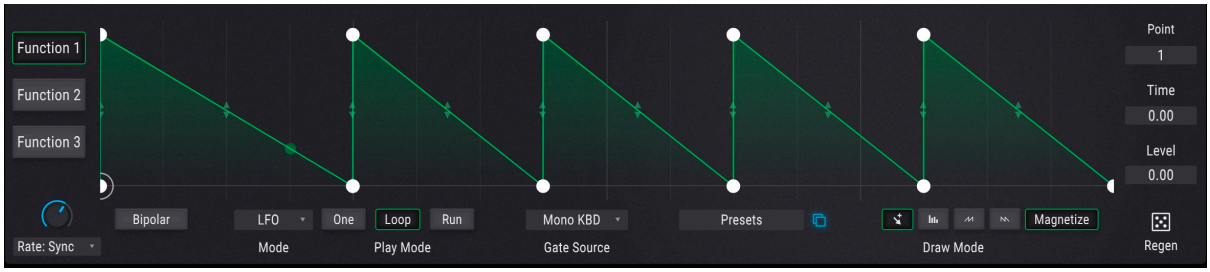
Given the ability of the Euclidean algorithm to generate rhythm patterns for music pieces, the same approach can be utilised to create the sound waves and sound automation with pulses (zones of high pressure) defined by the resulting value from the Euclidean algorithm.

3.1 Waveshaping with the Euclidean algorithm

Apart from creating the rhythm for percussion elements and actual movements in sound, we can take the same approach of defining stepping of 1 and 0 over a given number of pulses, to develop the pulse patterns inside of the wave shapes, transforming them into more complex sounds.

We can take the basic LFO (low-frequency oscillator) and use it as our envelope during the process of waveshaping. The base function for wave folding will be one of 4 waveforms we've defined previously (sine, square, triangle, sawtooth). From there we will take peak values of those functions and

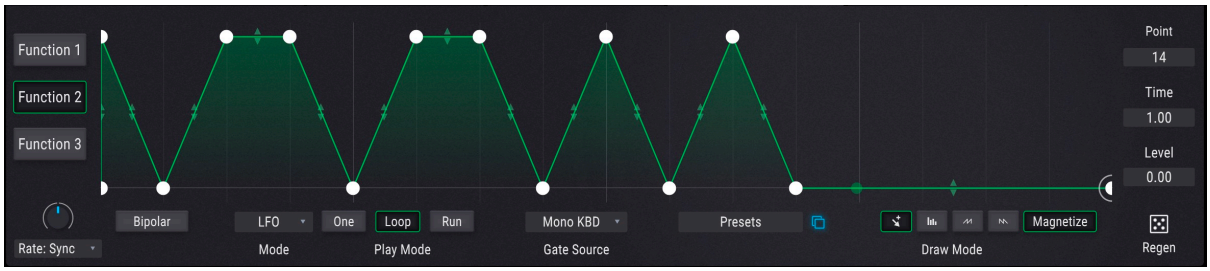
Picture 4. Waveform oscillation shaping with Euclidean algorithm



Picture 5. Function 1 with E(5,16) and sawtooth OSC waveform.

treat them as *true* or 1 in our euclidean algorithm result, and 0 values as the release ones. The number of LFO pulses per second could be defined by the size of note and number passed in Euclidean algorithm function, in this example, we're passing 16 pulses with 5 «hits» over one second of LFO oscillation.

The parameters that being morphed by the LFO are waveform position, meaning the wave shape transform from sine to sawtooth, with the 1 position as sawtooth and 0 as a sine wave. Oscillation takes the steady decline back from sawtooth to sine wave triggering shapes such as triangle and square and cause of this, sound becomes dirtier and could be used as composer choice to bring new characteristic to the defined synthesiser.



Picture 6. Function 2 with E(7, 12) and triangle OSC waveform.

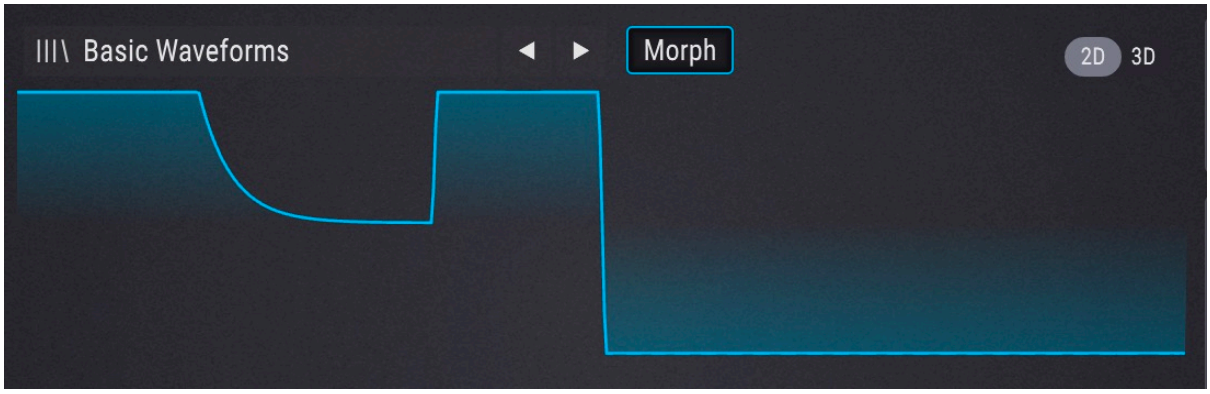
3.2 The Euclidean rhythm as oscillation pattern

We've taken a look at the Euclidean algorithm oscillation through LFO to change sonic characteristics of a generated waveform, but we proceed further with Euclidean rhythm application to blend multiple patterns from separate

LFOs and with such approach push the boundaries of possible creativity, designing waveshapes that will be more complex, compared to 4 basic ones.

By re-arranging the behaviour of the oscillation function in the synthesiser editor, we can achieve a blending of multiple functions and their separate influence on the resulting sound wave.

On the two pictures (5 and 6) we're able to see the oscillation patterns with peak points being defined by 1's in resulting Euclidean output, and decay parts as 0.



Picture 7. Resulting waveform with blended OSC outputs.

Setting those oscillators output for wave shape change, allows us to achieve unique, and non-basic waveform, which is being modulated using the Fourier series, adding more sub-harmonies to the resulting sound.

3.3 The Euclidean rhythm oscillation for FX control

The beauty of LFO oscillators is in their simplicity, it generates the signal value that is being modulated to the given function form, and can be used not only for wave shaping and morphing waveforms but also as an FX controls unit, making possible the control of FX values like cutoff for filtering sound or resonance and panning with ease.

3.3.1 FX types

Each synthesiser that creates sound based on OSC and waveshaping, depends on few key elements.

First, is the oscillator itself, and the wave function that is being used as a

Picture 8. FX Cutoff being oscillated by Function 2.

sound form. The oscillator passes the value at a given moment throughout the pipeline, which goes through filters or other effects, that can drastically change the resulting waveform.

The second is the FX. FX is short for «effect» and usually means a single function for sound wave morphing, it could be a filter, cutting frequencies above or below a certain threshold.

Apart from the common filter, there exist other FX, they are as follows:

- LowPass Gate – the function that ensures that all the frequencies below the defined threshold will be cut off with a given intensity
- Phaser – the function that inverts phase momentum in the given frequency range, allowing a switch to phase.
- Comb – morphs frequencies(cutting or boosting them) based on the provided waveform or multiple waveforms and given the intensity of the FX that has to be applied.

3.3.2 FX controls

Each FX has a set of controls that define the level of «presence» of the effect on the resulting waveform, custom controls specifically for FX, but usually the range of frequencies impacted by the FX and amount of signal that is being mixed into resulting one.

3.4 Macros mapping with the Euclidean rhythm

Macros in any synthesiser usually mean the quick access knob or slider, which provides numeric value which can be used to pass value to different parameters of the sound generator.

Macros could be automated with an LFO oscillator in the same fashion as any other digital parameter of a defined synthesiser, and this creates an additional layer of creativity, making possible the development of rhythmic patterns that influence the sound generation pipeline.

4. PRACTICE

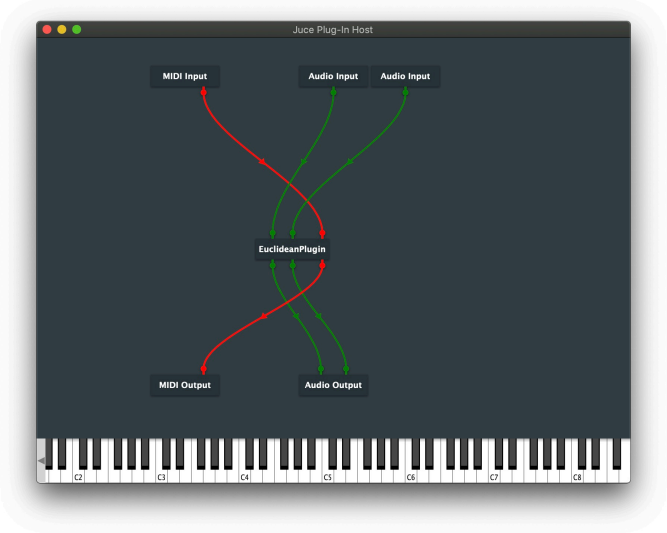
The practice part is dedicated to the development of the VST3 plugin for any host DAW(digital audio workstation) which will implement the synthesiser where sound waves are being generated with Euclidean algorithm stepping.

4.1 Tools

For plugin development author decided to take the JUCE framework written in C++, which defines the abstractions for DSP (digital signal processing), such as input and output channels, MIDI messages capturing and oscillation signal generation. Additional library Maximilian that provides additional DSP abstractions and oscillator engine classes was in use too.

As a host DAW has used Logic Pro X software, this audio workstation allows integration of the plugin into the sound generation pipeline, and captures the output signal, as part of the recording.

Apart from shaping sound waves inside of the custom-made plugin, another VST Pigments were used to test and compare possible output, and compare wave shaping quality to the one developed in this course work.



Picture 9. JUCE Plug-in Host used for plug-in debug.

For debugging purposes the environment provided by JUCE were used, it allows to customise signal and messages that are being sent over the plugin pipeline, but also generates MIDI signal.

4.2 Main abstractions and understanding of DSP pipeline

Any existing VST3 plug-in integrates into the host DAW application as a middle-man program, that receives the audio signal from DAW, does the calculation of a received buffer and outputs signal with introduced changes.

For further development of this plugin, the author decided to take a path with separate abstraction for the Euclidean rhythm generation, which will be triggered each time the function *processBlock* by VST3 processor will be called. In this case, the parameters which could be defined by knobs for size and number of pulses are sent to the *euclid()* function alongside the current step of the oscillator, to receive value based on the stepping.

JUCE framework came in hand with supporting types for oscillator signal generation. It provides pulses required for buffer synchronisation as well as the ability to utilise oscillator as LFO for more granular tuning and precision in the calculation.

4.3 Plug-in interface

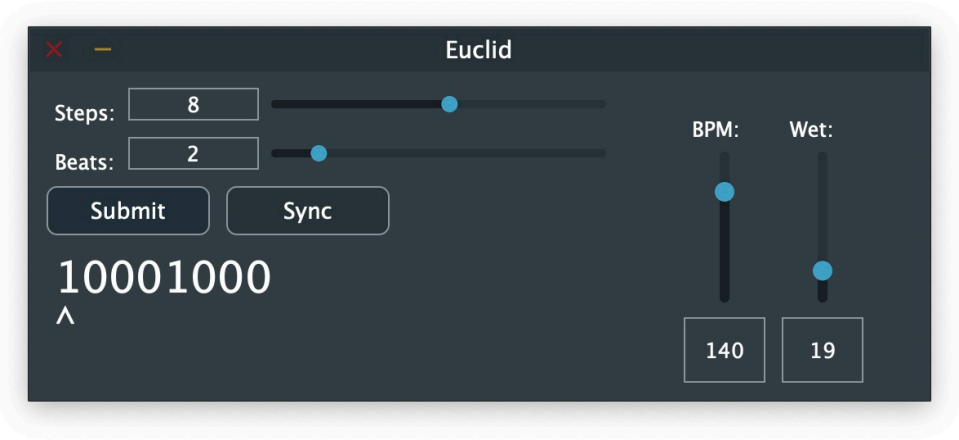
Plug-in interface requirements are as follows:

- Simple controls for the Euclidean rhythm generator
- Less obstructive and confusing elements
- Defined presets for quick-start work
- Number approximation control for sound quality control

Based on those requirements the control surface of the plugin was created and tested.

4.4 Plug-in behaviour

The developed plug-in has predictable behaviour with controls for Euclidean rhythm generation, which define the number of steps and beats in a given sequence, outputting the resulting pattern in binary form, together with controls for BPM sync either from the host (in this case DAW) or manually defining one. Apart from controls of Euclidean rhythm generator, mix controls persist as well, they represent the amount of «wet» signal (a signal that has been impacted by plugin processing) is being mixed into an output signal and «dry» signal (a signal that has been initially sent to the plugin).

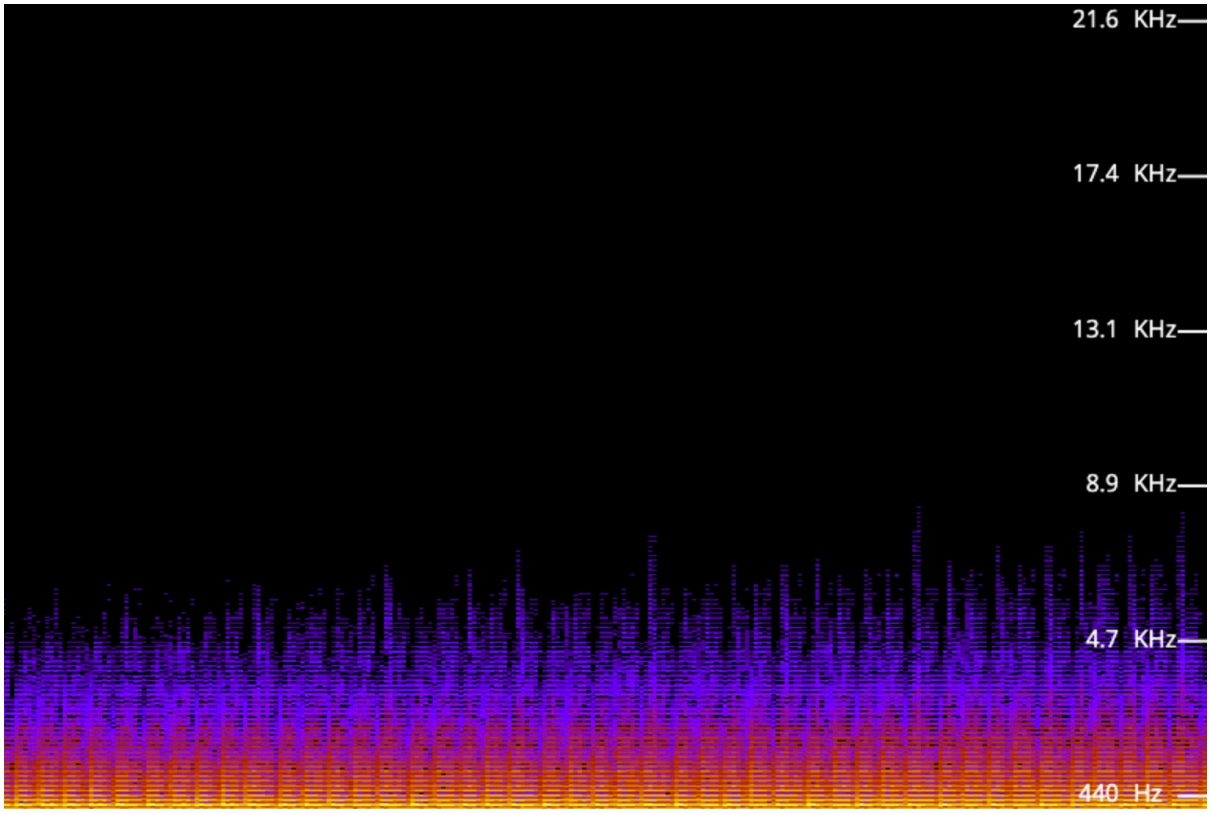


Picture 10. Plugin interface with stepping.

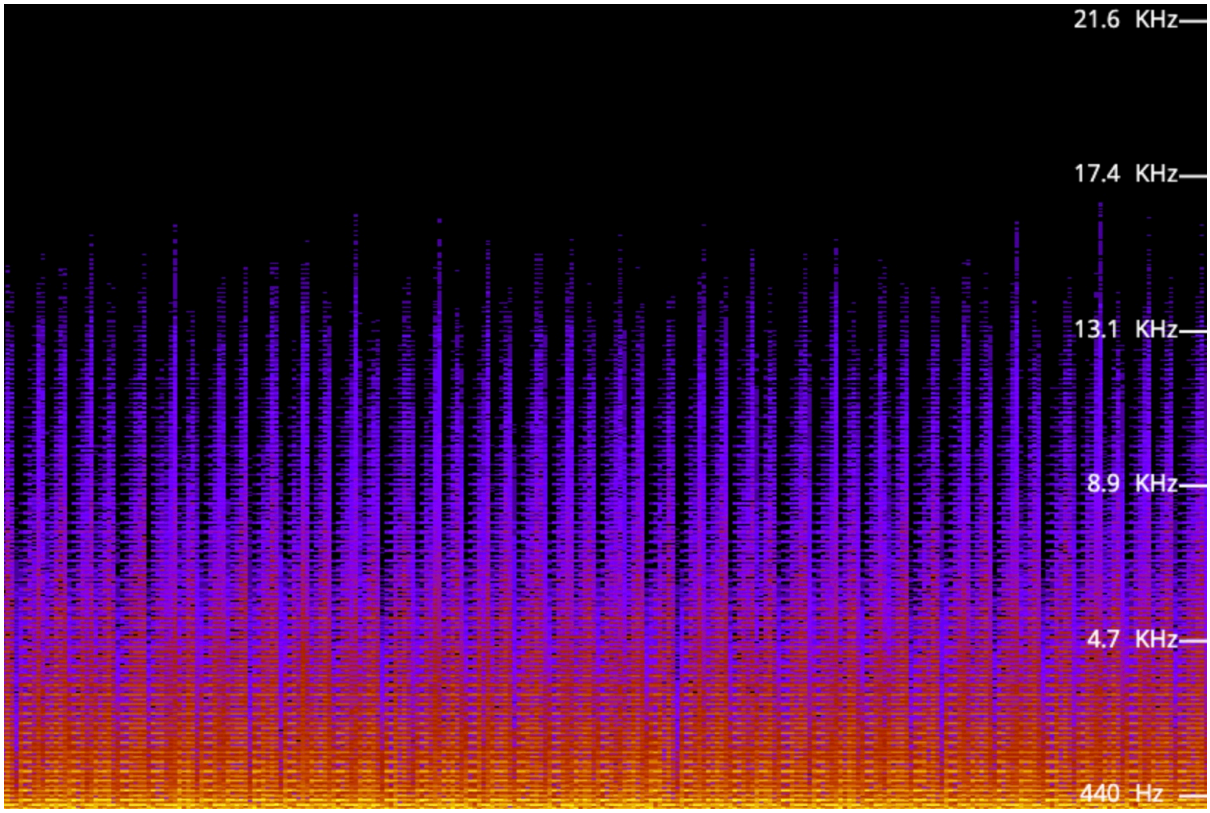
4.5 Plug-in application

Developed plugin, was used to process the synthesiser signal, and the result of processing saved in *processed.mp3* file attached with other sources for this course work, the clean example provided too.

As well as sonic examples the spectrogram for both clean and processed examples provided below, where we able to see, how in the same pice, the peaks of high frequencies are correlated with defined pattern (for examples the rhythm of E(5, 16) and E(7, 12) were used).



Picture 11. Spectrogram of the clean signal.



Picture 12. Spectrogram of the processed signal.

Due to the waveshape phase change, with two different rhythms defined, we are able to observe, the overall high-frequency range increase, with peaks located at Euclidean algorithm 1's location. This behaviour was expected, and the sound blending gives an additional level of creativity for music production.

CONCLUSION

This course work gave an ability to test author knowledge in different fields, such as music theory, digital signal processing and the mathematical basis of music. Given the example of sound generation that uses Euclidean algorithm as the main source for rhythmical and waves folding development, helped to understand the nature and physics of the sound better and combined with practice plug-in for digital signal processing.

As well, plug-in development experience leveraged and tested the knowledge of C++ language and frameworks such as JUCE. The time spent playing with different settings for the Euclidean plug-in gave a positive experience and showed new ways for sound production, that was not known by the author before.

REFERENCES

1. Godfried Toussaint, «The Euclidean Algorithm Generates Traditional Musical Rhythms», 2005
2. Cogan Robert, «Sonic Design The Nature of Sound and Music», 1976
3. David Byrne, «How Music Works», 2012
4. Shah Saloni, «An Exploration of the Relationship between Mathematics and Music», 2010
5. Jonathan M. Blackledge, «Digital Signal Processing: Mathematic and Computational Methods, Software Development and Applications», 2006
6. Olson, Harry F., «Music, physics and engineering», 1967
7. Arturia Pigments User Manual [https://downloads.arturia.net/products/pigments/manual/pigments_Manual_2_0_EN.pdf]
8. JUCE documentation [https://docs.juce.com/master/index.html]
9. Maximilian DSP library [https://github.com/micknoise/Maximilian/]