BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

GRANULAR SYNTHESIS IN MUSIC PRODUCTION

GRANULÁRNÍ SYNTÉZA V HUDEBNÍ PRODUKCI

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **PAVEL PEČINKA**
AUTOR PRÁCE

SUPERVISOR **prof. Dr. Ing. JAN ČERNOCKÝ**
VEDOUCÍ PRÁCE

BRNO 2022

Department of Computer Graphics and Multimedia (DCGM) Academic year 2021/2022

Bachelor's Thesis Specification

25122

Student:	**Pečinka Pavel**
Programme:	Information Technology
Title:	**Granular Synthesis in Music Production**
Category:	Signal Processing

Assignment:

1. Get acquainted with the principles of HW and SW systems for music production.
2. Get acquainted with the principles of granular synthesis and perform a survey of commercial and open solutions.
3. Design an algorithm for granular synthesis. Create a prototype in a standard programming language.
4. Implement as a plugin capable of interfacing with music processing software.
5. Suggest one or several improvements to the current systems, implement and test.
6. Design, perform and evaluate user tests.

Recommended literature:

- TRKAL, Tomáš. Softwarový multiefekt pro postprodukci populární hudby. Brno, 2017. Master's Thesis. FIT BUT. 2017
- according to supervisor's recommendation

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Černocký Jan, prof. Dr. Ing.**
Head of Department:	Černocký Jan, doc. Dr. Ing.
Beginning of work:	November 1, 2021
Submission deadline:	July 29, 2022
Approval date:	November 1, 2021

Abstract

The focus of this thesis is on designing and creating a granular synthesis application for music production. The application is implemented as a VST3 plugin with the use of JUCE framework and C++ programming language and is capable of interfacing with Digital Audio Workstations, or DAWs for short, which are programs that serve as a central point for music producers, where they can create new sounds, mix songs and create audio recordings among many other things. The plugin is designed as a sampler, able to load a sample file containing audio data and use this data for further processing and playback. Evaluation of user tests indicates, that the application can be successfully used for music production as a granular synthesis instrument.

Abstrakt

Cílem této práce je navrhnout a vytvořit aplikaci pro granulární syntézu, využitelnou v hudbení produkci. Aplikace je implementována jako VST3 zásuvný modul s využitím frameworku JUCE a programovacího jazyka C++ a je schopná propojení s programy typu Digital Audio workstation, zkráceně DAW, což jsou programy sloužící jako centrální bod pro hudební producenty, kde mohou mimo jiné vytvářet nové zvuky, provádět mixování skladeb a vytvářet zvukové nahrávky. Tento zásuvný modul je navržen jako sampler a je schopen nahrát zvukový soubor obsahující audio data, která dále využívá pro další zpracování a následné přehrávání. Z vyhodnocení uživatelského testování vyplývá, že je tato aplikace použitelná pro hudební produkci jako nástroj pro granulární syntézu.

Keywords

granular synthesis, grain, envelope, music production, signal processing, DAW, Digital Audio Workstation, plugin, effect, synthesizer, sampler, VST, JUCE

Klíčová slova

granulární syntéza, zrnko, obálka, hudební produkce, zpracování signálů, DAW, plugin, zásuvný modul, efekt, syntetizér, sampler, VST, JUCE

Reference

PEČINKA, Pavel. *Granular synthesis in music production.* Brno, 2022. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor prof. Dr. Ing. Jan Černocký

Rozšířený abstrakt

Granulární syntéza je zvuková syntéza, ve které se používá koncept zrnka – malé jednotky zvuku, obvykle 1 až 100 milisekund dlouhé, k vytvoření dlouhých, rozsáhlých zvukových kulis. Na tato zrnka se aplikuje amplitudová obálka, která změní jejich obsah a umožní jejich rapidní přehrávání bez způsobení nechtěných zvukových artefaktů. Vše je dosaženo pomocí těchto zrnek, které samy o sobě nemusí mít žádný zajímavý sonický obsah, ale jejich skládáním na sebe, nebo naopak jejich rozprostřením v čase lze dosáhnout zajímavých zvukových textur. Takzvané granulizéry jsou virtuální hudební nástroje, které jsou schopné provádět granulární syntézu. Tyto programy obsahují četné ovládací prvky, kterými hudební producent může kontrolovat způsob přehrávání, parametry jednotlivých zrnek jako jejich velikost, amplitudovou obálku a jejich rozestupy.

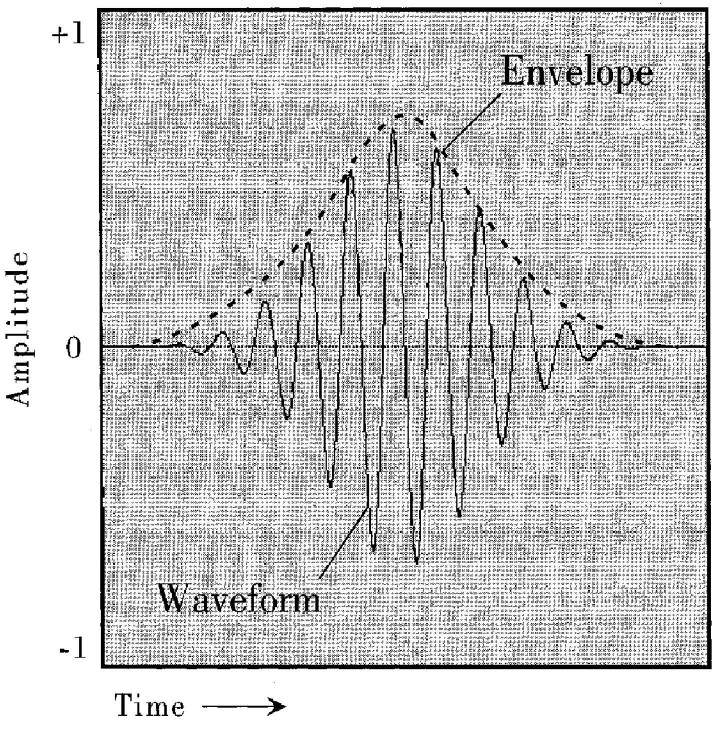


Figure 1: Příklad zrnka. Zdroj: [11]

Výstupem této práce je aplikace *Petaldance* – granulární syntetizér, implementovaný jako zásuvný modul ve formátu VST3, který je schopen propojení s hostitelskými aplikacemi pro hudební produkci, což jej umožňuje využívat během standardního procesu tvorby hudby. Během tohoto procesu pracuje hudební producent v takzvaném digial audio workstation, což je pracoviště obsahující komponenty pro vytváření skladeb, jejich mixování a editaci. Tato pracoviště obsahují mnoho dalších pluginů, jakým je výstup této práce, takže je zásadní, aby byl implementován jako zásuvný modul, který spolupracuje s ostatními programy a nerušil pracovní postup.

Na závěr byly navrženy a provedeny uživatelské testy, které měly za úkol zhodnotit výslednou aplikaci po grafické stránce, po stránce jednoduchosti a intuitivnosti ovládání a určit, zda je tato aplikace vhodná k použití jako nástroj v hudební produkci. Z výsledků uživatelských testů a recenzí vyplývá, že je aplikace vhodná jak pro začínající, tak pro

zkušené producenty a s menšími vylepšeními by mohla dle slov recenzentů konkurovat komerčním řešením, která jsou aktuálně dostupná na trhu.

Granular synthesis in music production

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of prof. Dr. Ing. Jan Černocký. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

. . . .
Pavel Pečinka
August 1, 2022

Acknowledgements

I would like to thank prof. Dr. Ing. Jan Černocký for his advice, consultations, guidance and help with this thesis and for allowing me to select this custom assignment. I would also like to thank my family for their support of my studies.

Contents

Chapter 1

Introduction

Becoming a music producer is more accessible than ever, thanks to the low cost of entry compared to few decades ago, together with the increasing amount of information that is shared on this topic. Physical instruments and hardware can be replaced with software variants that are cheaper or often free, allowing anyone that previously didn't have means or space to store the hardware to start this hobby.

This thesis deals with designing and implementing a granular synthesis plugin, that is capable of interfacing with digital audio workstations or DAWs for short. DAWs are host programs used by music producers where they can record, edit and master songs. The application is implemented in JUCE framework and C++ programming language as a VST3 plugin – a format used by most of the current digital audio workstations. It is designed as a sampler, able to load a sample file and use the loaded data for further processing and playback.

Granular synthesis is a synthesis method that uses the concept of grain – a small unit of audio data, usually around 1 to 100 milliseconds in duration, to create large and long soundscapes, sounding as if a reverberation effect was applied, all from these short, individually uninteresting grains. This is achieved by different ways of layering these grains on top of each other or spacing them out, depending on the desired resulting sound texture. Granular synthesis plugins, often called granulizers, can have numerous adjustable options, which control different aspects of the grain manipulation, such as the amplitude envelope applied to grains, amount of grains generated in a specific time frame, direction of the playback and many more. Capabilities of granular synthesizers are shown in chapter 4.

Chapter 2 focuses on music production, describing currently used digital audio workstations and plugins. The concept of granular synthesis is described in Chapter 3, along with different possible implementations. Plugin market research is conducted in Chapter 4, showing different granular synthesis plugins available on the market at the time of writing this thesis. Each of them is tested and analyzed to gain inspiration on which features should be picked up for the final application and to look out for any flaws that should be avoided in the final program. Chapter 5 deals with conceptual design for the final plugin and creation of a prototype in Python language. Chapters 6 and 7 focus on implementation, testing and user review of the final application.

Chapter 2

Music production

The increase in computing power, happening in the last two decades, enabled people to access tools that have been locked behind significant upfront costs. Owning or even renting studio used to be reserved mostly for professionals. Nowadays, anyone with a computer and interest in learning about music production can try this hobby, profession and passion. This chapter deals with describing tools for music production used in the current times – *Digital Audio Workstations* or *DAWs* for short and *plugins* used inside them.

2.1 Digital Audio Workstations

Digital audio workstations are host programs that together with other components, called *plugins*, create a virtual studio containing everything a music producer might need for recording, composing, editing and mixing of music.

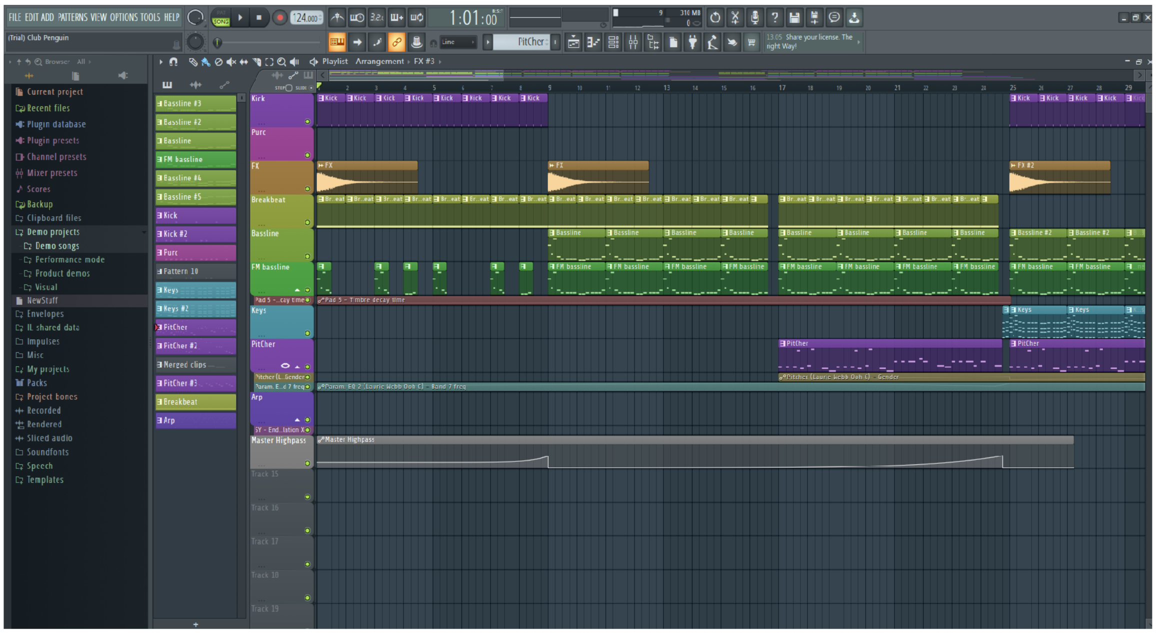


Figure 2.1: Digital Audio Workstation FL Studio 20

There are numerous DAWs on the current market, both paid and free, each with their own specific workflows, advantages and drawbacks. One of them is going to be analyzed and described – *FL Studio*[1] and *Ableton Live*[2] will be introduced as an alternative. These two DAWs were used during the development and testing of the final plugin as is described in Chapter 7.

2.1.1 FL Studio

FL Studio, formerly known as FruityLoops, is a DAW developed by Image-Line[3]. FL Studio comes in four different editions, costing from € 99 up to € 499, but comes with free lifetime updates [5]. FL Studio 20 can be seen in Figure 2.1. Three core components of this DAW are a mixer, playlist and piano roll.

The mixer contains tracks into which music producer can assign plugin instruments, samples or external audio inputs such as microphones. Mixer tracks have slots for effect plugins such as delay, filter or reverb. Typical examples for a mixer track can be a group of drums or a vocal. Producer would then apply effects on this track in a specific order, modifying the resulting output in the final mix [2]. Mixer can be seen in Figure 2.2.

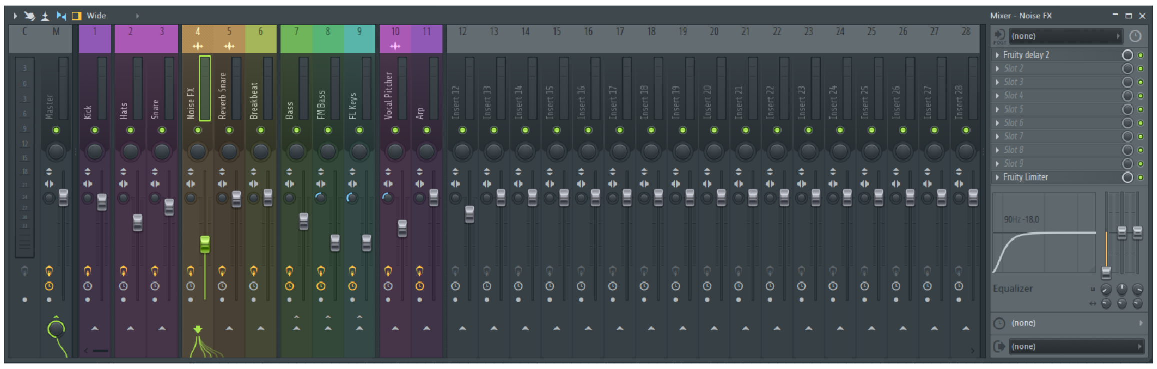


Figure 2.2: Mixer from FL Studio 20

Another part of this DAW is playlist with tracks. Playlist is where the music producer can arrange the song, specifying when each element starts. Each track can contain a pattern with MIDI notes, a sample like a vocal track or automation clip, used to automate different values, such as plugin parameters or sample file attributes like pitch. Playlist can be seen in Figure 2.3. Individual tracks can be color coded and named for better orientation in the playlist, allowing users to create organized projects that can then be shared for collaboration with other artists or producers without the other party having to decipher the contents of the project [4].

Last notable part of this DAW is a piano roll, which is used to store note information for playback, as can be seen in Figure 2.4. Music producer manually inserts the notes which are then triggered by playback, making the plugin, that the notes are drawn for, play with the corresponding pitch. Additional options include controlling the note velocity, setting up note sliding and setting portamento effect [3].

FL Studio comes with a wide range of plugin instruments, ranging from virtual pianos, synthesizers and samplers. FL Studio also comes with a range of built-in effects such as filters, delays and compressors, built-in instruments, one of which is going to be described in Chapter 4.2 and other components like Edison – a fully host integrated audio editing,

[1]FL Studio - https://www.image-line.com/fl-studio/
[2]Ableton Live - https://www.ableton.com/en/shop/live/
[3]Image-Line - https://www.image-line.com/

analyzing and recording tool, useful also in development for inspecting the audio output of our plugin.

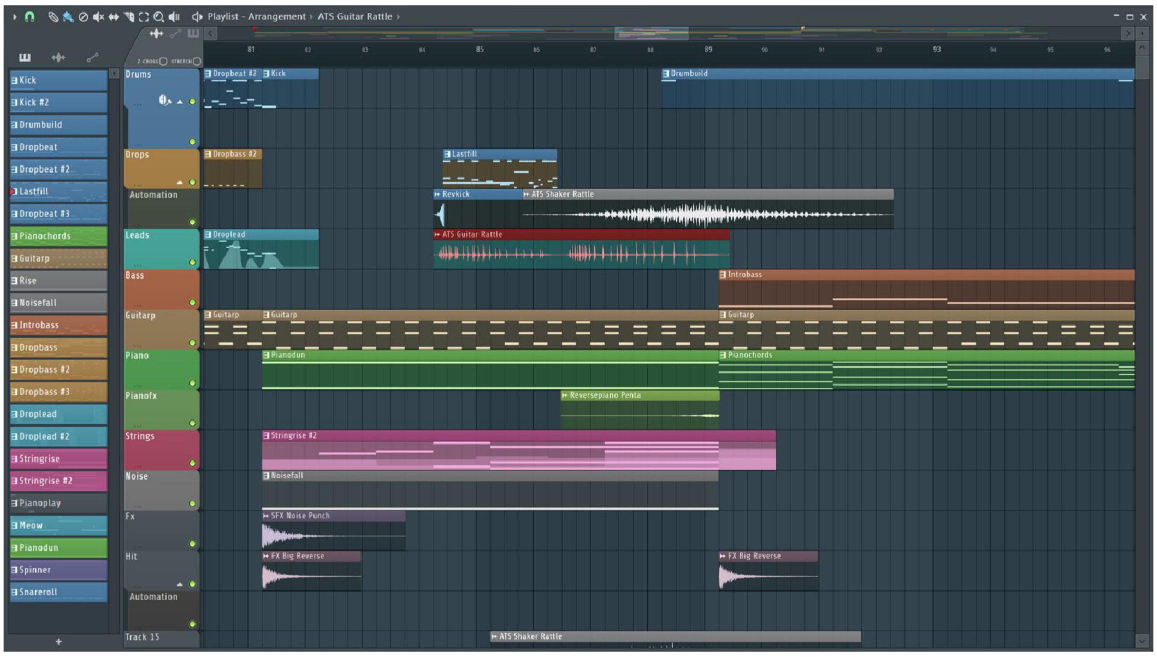


Figure 2.3: Playlist from FL Studio 20

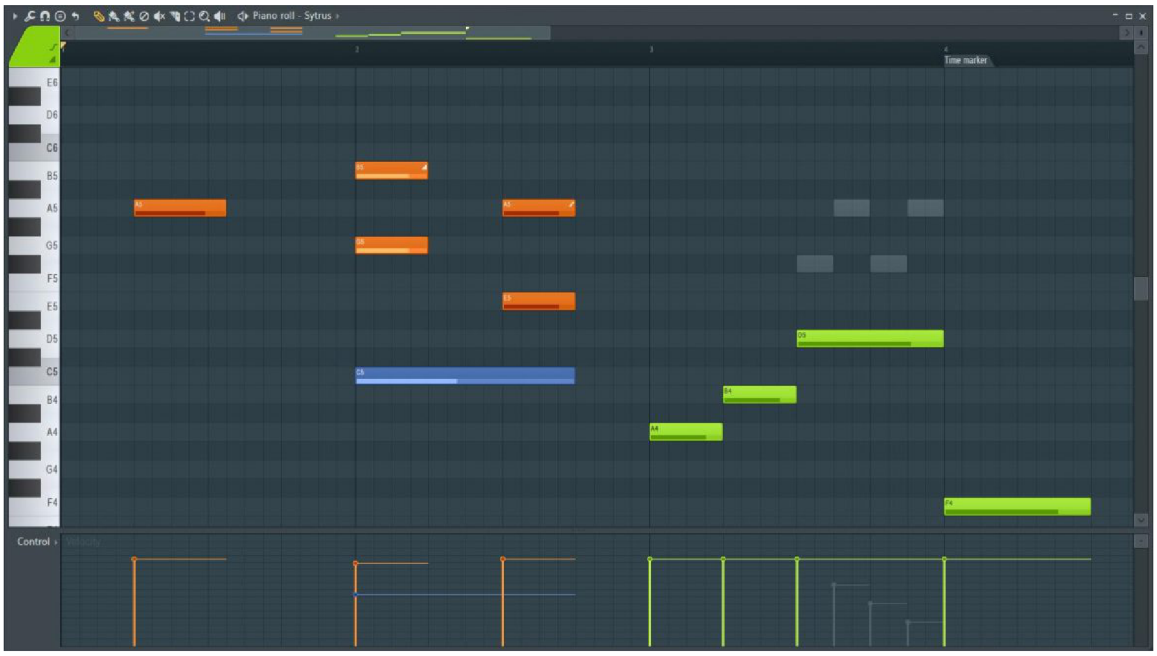
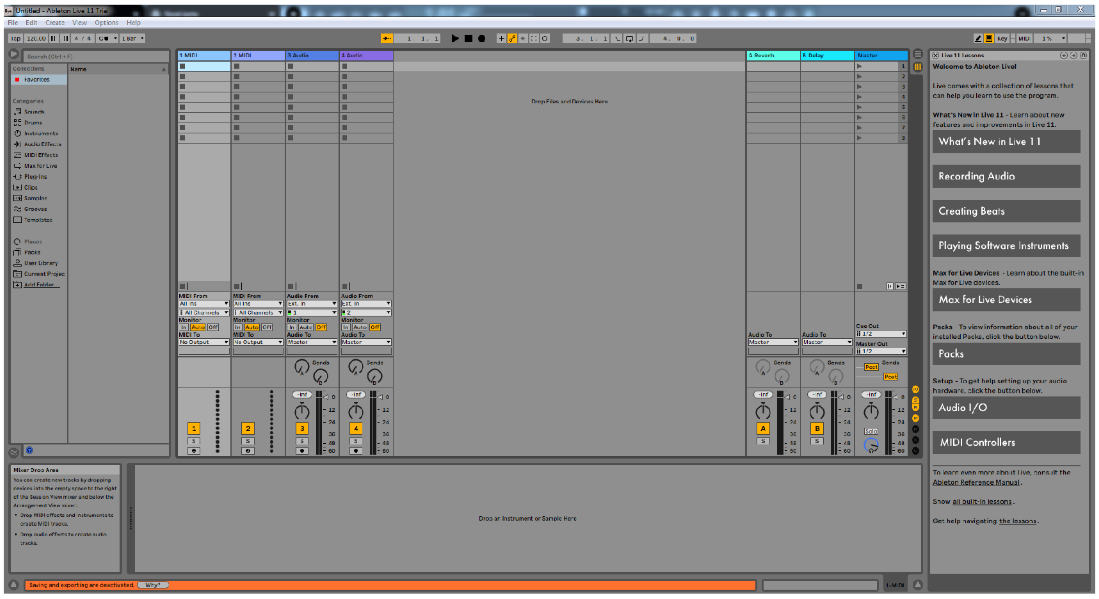


Figure 2.4: FL Studio 20 with a pianoroll open, containing notes. Source: [3]

2.1.2 Ableton Live

Ableton Live is a digital audio workstation developed by Ableton AG[4]. In comparison to FL Studio, each new version of this DAW has to be bought again, whereas FL Studio guarantees lifetime of free updates after one time purchase. Ableton Live can be seen as a MacOS alternative to FL Studio, both of them being popular and essentially rival DAWs. Ableton Live has its own version of FL Studios mixer, playlist and piano roll, with a different workflow behind it. Ableton Live 11 can be seen in Figure 2.5.



Welcome to Ableton Live!
What's New in Live 11
Recording Audio
Creating Beats
Playing Software Instruments
Max for Live Devices
Packs
Audio I/O
MIDI Controllers
Drop Files and Devices Here
Drop an Instrument or Sample Here
Saving and exporting are deactivated.

Figure 2.5: Ableton Live 11

Figure 2.6 shows *Arrangement* view, the Ableton Live 11 oversion of the *Playlist* from FL Studio. The individual tracks can be assigned to plugin instruments and MIDI note data entered on the track, making the instrument play the defined notes during track playback.

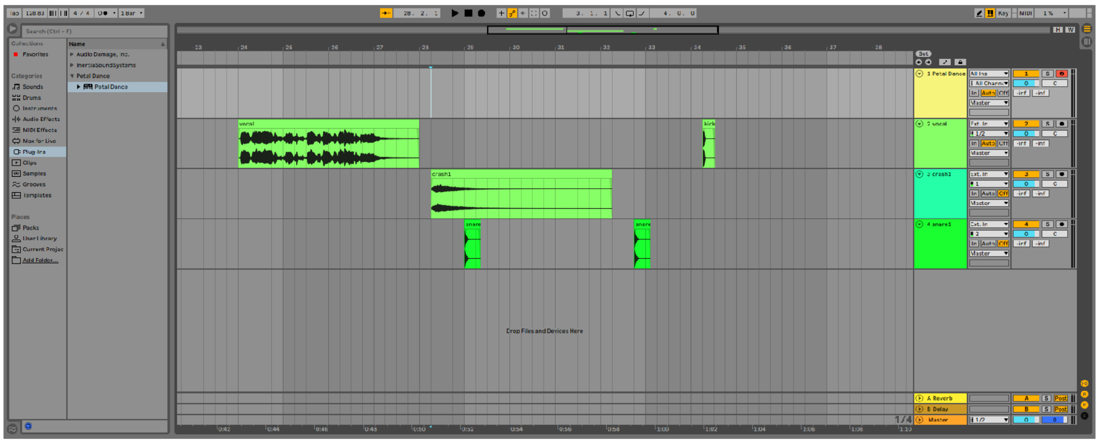


Figure 2.6: Ableton Live 11 Arrange View

[4]Ableton AG - https://www.ableton.com/en/

2.2 Plugins

Plugins, in the context of music production and digital audio workstations, are applications that run inside the host DAW and can be one of various categories. One category of plugins are instruments – this includes software versions of real-life instruments such as guitars, pianos, drums, software versions of analog synthesizers or instruments that exist purely as a software, not having a physical, real-life variant such as granulizers. Another category are effect plugins. Effect plugins are used to modify audio from different sources, examples of such plugins being *filters*, *delays*, *reverberation* effects and plugins used for mastering and mixing of tracks, such as *compressors* and *limiters* that compress dynamic range. Third category of plugins could be called utility plugins, or analyzers. These plugins can be used to inspect the audio quality by displaying the waveform or a spectrogram of the current playback, record audio or otherwise assist the music producer with tasks different from general music production.

2.2.1 VST

Virtual Studio Technology, or VST for short, is an interface and a plugin format for both Windows and MacOS developed and maintained by *Steinberg Media Technologies*[5]. It allows the developer to focus on the plugin functionality and handles the interfacing with the host application. Currently, VST plugins are supported by almost every DAW on the market, including FL Studio and Ableton Live, so it is a prudent choice as a baseline for a plugin, since an application built on this technology will have far reach and compatibility from the start. Currently supported and most used version is VST3, which is a rework of the older versions, with improved performance and new features [13]. These changes made the VST3 version not backwards compatible, but at the time of writing this thesis, there is already announcement about discontinuing of support for VST2 in the future, meaning developers should focus on VST3 plugin development only [14]. This is the reason VST3 plugin format was chosen for the granular synthesis plugin created in this thesis.

Figure 2.7: VST logo. Source: [12]

[5]Steinberg Media Technologies - https://www.steinberg.net/

2.2.2 AU

Audio Units is a plugin format for plugins on MacOS. It is often times considered as MacOS counterpart to the VST format. This format is supported by numerous DAWs available on MacOS, such as Ableton Live, *Logic Pro*[6] or *GarageBand*[7].

Figure 2.8: AudioUnits logo. Source: [1]

[6]Logic Pro - https://www.apple.com/logic-pro/
[7]GarageBand - https://www.apple.com/mac/garageband/

Chapter 3

Granular synthesis

This chapter deals with description of granular synthesis. Basic concepts are described, together with different kinds of methods of implementation.

3.1 Concept of grains

The adjective granular in granular synthesis means the signal is processed with a technique called *granulizing*. Granulizing is a process where the input signal, in music production typically audio signal from a DAW or a sample file, is split up into small bits called *grains*.

Grains are small parts of the original signal, typically in range from 1 to 100 miliseconds long, but even multiple seconds long grains can be considered, when they have the right parameters. These grains are then processed further – an amplitude envelope is applied to each grain along with possible other effects, such stereo *panning* and then they are played back [10, 11]. Example of a grain can be seen in Figure 3.1. Chapter 3.2 adresses what grain envelopes are and which envelope shapes can be considered when trying to select one.

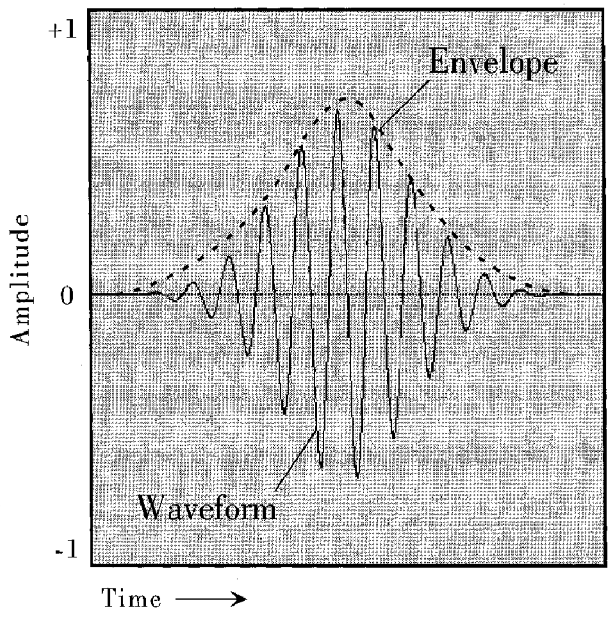


Figure 3.1: Example of a grain. Source: [11]

3.2 Choosing a grain envelope

Grain envelope, often times called a window, is an amplitude envelope that is applied to each generated grain. The process of applying the envelope to a grain is often called windowing. In general, it can be interchangeably said that the envelope also determines the length of the grain or that the envelope is created to fit the size of the grain.

In theory, grain envelopes can have any shape, ranging from the shape of a Dirac delta function:

$$\delta(x) = \begin{cases} +\infty, & x = 0 \\ 0, & x \neq 0 \end{cases} \tag{3.1}$$

all the way up to a rectangular window also known as Dirichlet or a boxcar window, which doesn't change the amplitude of the grain at all:

$$w[n] = 1 \tag{3.2}$$

Since the process of granulizing is splitting the input signal into grains, it is important that the envelope starts and ends in a zero value, to prevent non-zero start and end of playback of the grains. If there was no such amplitude envelope applied (or an envelope not starting and ending in zero), the playback of such grains could create output full of unwanted artifacts, such as clicking.

Some simpler envelopes can be described by an *ASR* (attack, sustain, release) envelope, which can be seen in figure 3.2.

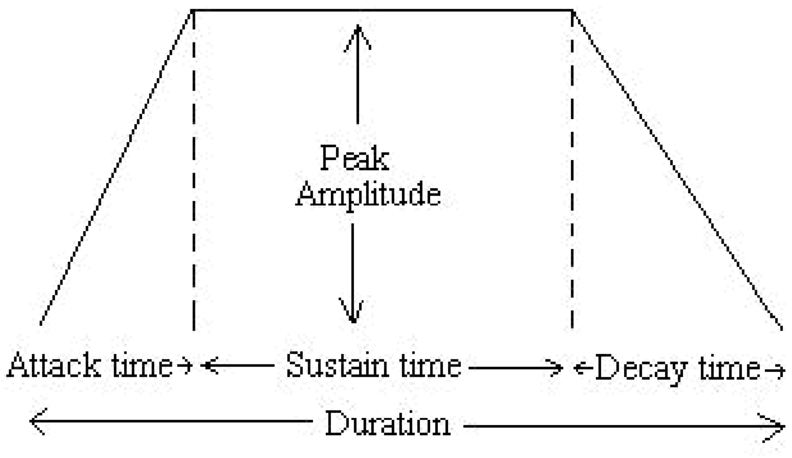


Figure 3.2: Example of an ASR envelope. Source: [9]

Two other grain envelopes frequently appear in literature on the topic of granular synthesis – Hann, or Hanning and Gaussian envelope. Hann envelope can be described as follows:

$$w(n) = \frac{1}{2} * \left(1 - \cos\left(\frac{2\pi * n}{N}\right)\right) \tag{3.3}$$

Where N is the size of the window. Hann window can be seen in Figure 3.3.

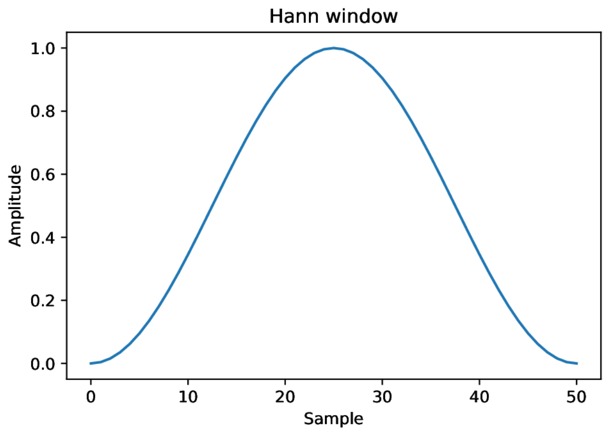


Figure 3.3: Hann envelope

Gaussian envelope can be described as follows:

$$w(n) = e^{\frac{-1}{2}(\frac{n}{\sigma})^2} \tag{3.4}$$

Where sigma is the standard deviation.

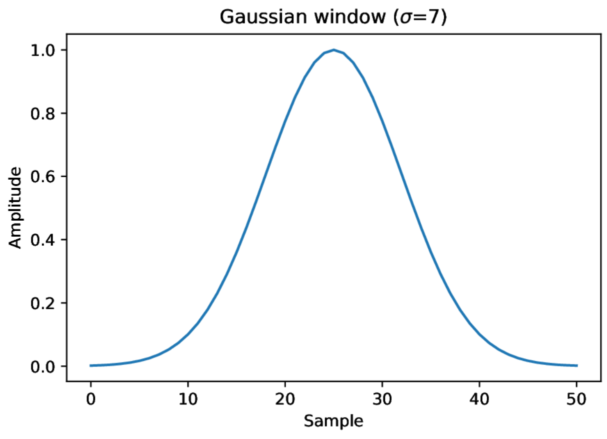


Figure 3.4: Gaussian envelope

Boxcar window is not of use, at least not in this form. In case of playback of grains with this envelope, there would be unwanted artifacts such as clicking. The sudden jump from zero to some is what causes the artifacts and this would happen for each and every grain. With grains of small sizes, such as 10 milliseconds, this could mean hundreds of unwanted artifacts per second. Boxcar envelope can be seen in Figure 3.5.

A solution would be to an envelope such as Turkey window, also known as tapered cosine window – an envelope with smoothed out start and end parts. Here the value of the envelope starts and ends in amplitude 0 and slowly ramps up to maximum amplitude, before ramping down again. This prevents the creation of artifacts. Turkey window can be seen in Figure 3.6.

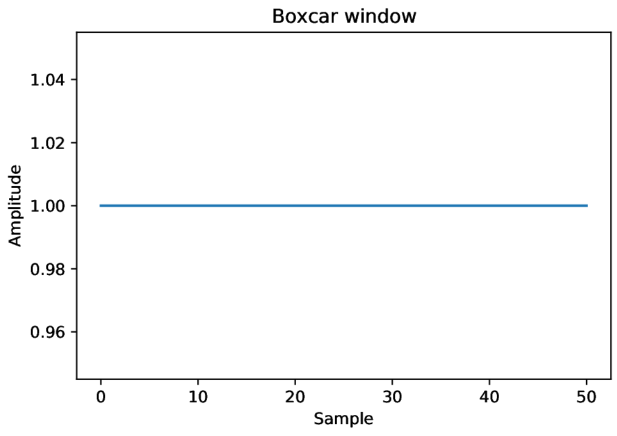


Figure 3.5: Boxcar envelope

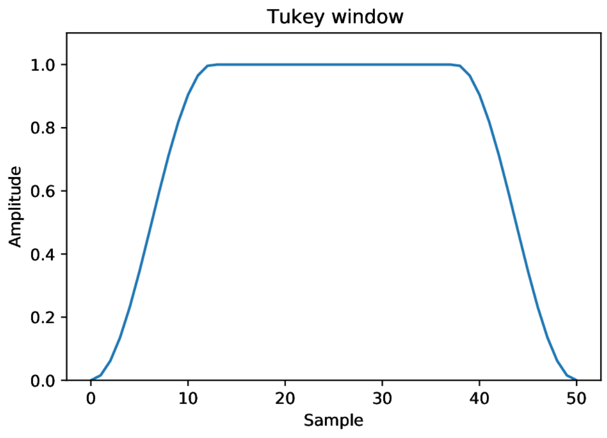


Figure 3.6: Turkey envelope

Virtually any grain envelope can be used, as long as it starts and ends in zero amplitude. The smaller the grain size is, the more pronounced is the effect of the grain envelope on the resulting sound texture.

3.3 Grain panning

Another parameter of grains is panning. Based on the parameter, grains will output varying amplitude to either left or right channel. Extreme case of this would be outputting normal volume to one channel – for example left – and 0 amplitude, so silence, to the right one. By switching this parameter from grain to grain, either randomly or in an organized manner, spatial effect occurs. With small sizes of grains and rapid switching of panning parameter on each of them a wide, surrounding texture of sound happens.

3.4 Grain playback

Apart from the size, shape and the content of the grains, the only component missing is the way of playing the grains. The way the grains are played back is the essence of granular synthesis. They can be played in the same order as was the original signal or sample, but for sonically interesting results it's typically in a different order, examples being reversed and/or randomised. Additionally, the grains can be layered on top of each other and cross-faded into each other in order to create interesting sound textures.

There are multiple types of granular synthesis, depending on how they organize the grains. The focus will be on *synchronous* granular synthesis.

3.5 Synchronous granular synthesis

Synchronous granular synthesis manages playback in a way, where each grain follows another one, with a specified delay between them. This delay can make the grains stack on top of each other or make them spaced out from one another, with silence in between. This delay can be described by a parameter called grain *density*.

Grain density can be represented as a number of grains per second. The sonic content of the individual grains is added together, resulting in increased amplitude, change in its timbre and pitch of the texture. Grain density is affected by the length, or duration of the individual grains. The greater the grain size, the larger the overlap of individual grains will be, resulting in completely different sound texture from a setting where small sizes of grains would be used.

Chapter 4

Plugin market research

This chapter contains a description of several granular synthesis plugins that are available on the market at the time of writing this thesis. The plugins are selected to show what granulizers can, but don't have to be capable of, in terms of manipulating the sound and how they can be controlled. In general, these plugins can have many functionalities, such as filters, FM synthesizers, LFOs for playback volume modulation and many more built into them, but the focus will be on the granulizing abilities, omitting the other functions.

The first plugin – The Mangle – will be described in detail, since it is the most unique out of the selected granulizers and provided the most inspiration for designing the plugin created in this thesis.

Testing was done by inserting samples of a vocal recording, snare drum hit, sine wave and a special "ramp" signal into the plugins and observing the output. Description and waveforms of these samples are in Section 5.2.

4.1 The Mangle

The Mangle[1] is a VST plugin made by Tom Maisey (Sound-Guru). It has a free, limited demo version and a paid, unrestricted version which costs £19.99 This plugin works as a *sampler* – user selects a sample file, drags and drops it onto the plugin, the sample file gets loaded and user can start working with the plugin.

The Mangle doesn't play back the entire sample, instead it generates grains from a static, selected position in the sample. The plugin displays a waveform of the loaded sample as can be seen in figure 4.1. There is a horizontal and vertical line creating a crosshair pointing to a specific spot on the sample. The vertical line signals the position in the sample where the grains will be generated from. The horizontal line determines volume amplitude of the grain ranging from 0% to 100%. The plugin also provides visual feedback for each generated grain as a dot which travels in the direction of the playback and has a size corresponding to the amplitude of the generated grain, bigger meaning louder, smaller meaning quieter. Focusing on the Modulators tab, there are five knobs that control the way the grains are played back:

1. **Rate** - rate of generating grains, ranging from one grain per 4 seconds to 250 grains per second

[1] The Mangle - http://sound-guru.com/software/mangle/

2. **Pitch** - changes the pitch of the grains, ranging from 25% to 1600% of the original pitch – the change in pitch also alters the playback speed, lower pitch making the playback slower, higher pitch higher

3. **Release** - release time of the grain, ranging from 0 to 3 seconds (can be also seen on the grain envelope at the bottom of figure 4.1)

4. **Reverse** - a percentage chance that a generated grain will be played in reverse, ranging from 0% to 100%

5. **Pan** - ranging from values -1 to +1, it indicates the spatial panning of the played grains, where negative values mean panning to the left, positive values mean panning to the right

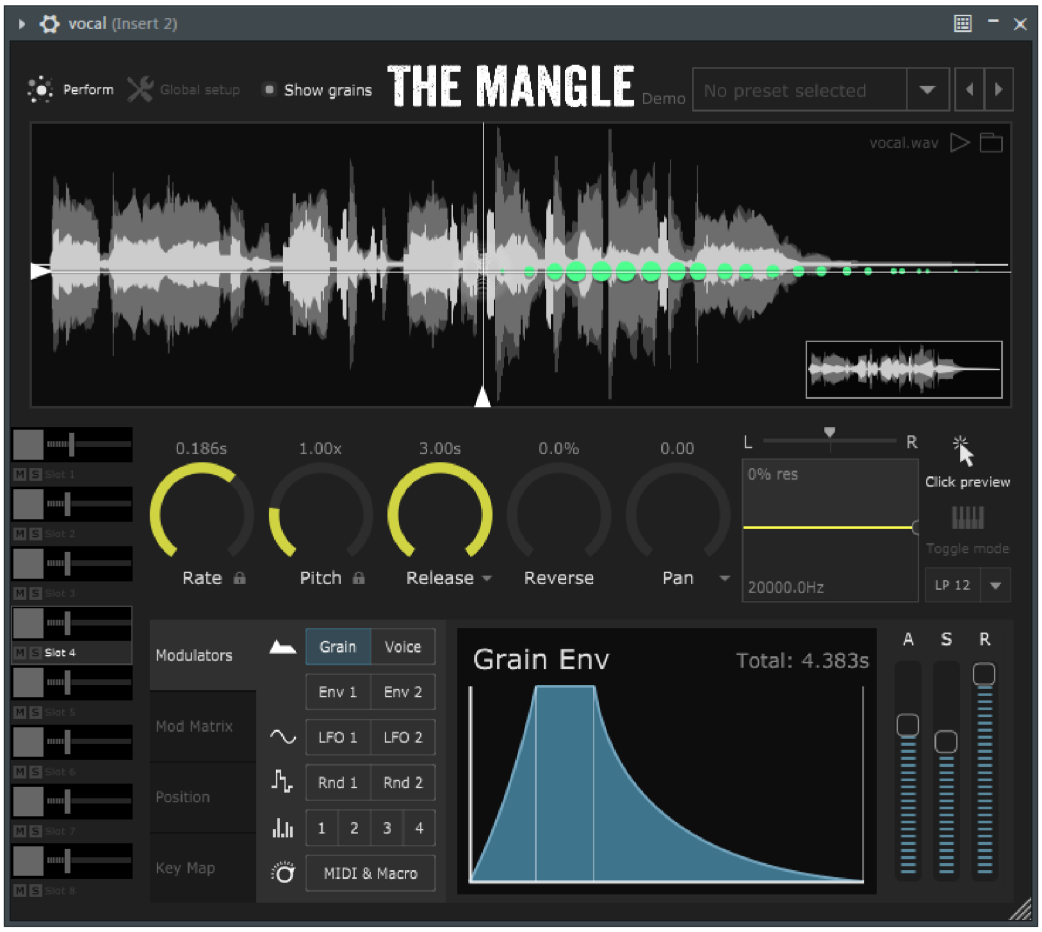


Figure 4.1: The Mangle VST plugin with a loaded sample

Lastly, at the bottom of the figure 4.1 is a grain envelope, which is composed of attack, sustain and release stages. Each of these stages can be adjusted by sliders located right of the grain envelope. Attack and release can be set anywhere from 0 seconds to 3 seconds and sustain can be set from 0 seconds to 4 seconds. Therefore, this plugin can create grains that are up to 12 seconds long, which further sets it apart from the other plugins, which usually have much shorter grain sizes.

Disadvantages

A few shortcomings of this plugin arised when testing this plugin. One imperfection of this plugin is in the grain envelope. When one or both of the attack and release parts of the envelope are set to 0 seconds, the envelope turns into a (half) square wave, which causes unwanted artifacts in the output signal because the crude envelope distorts it, making the output sonically unpleasant, which can be described as “clicking“.

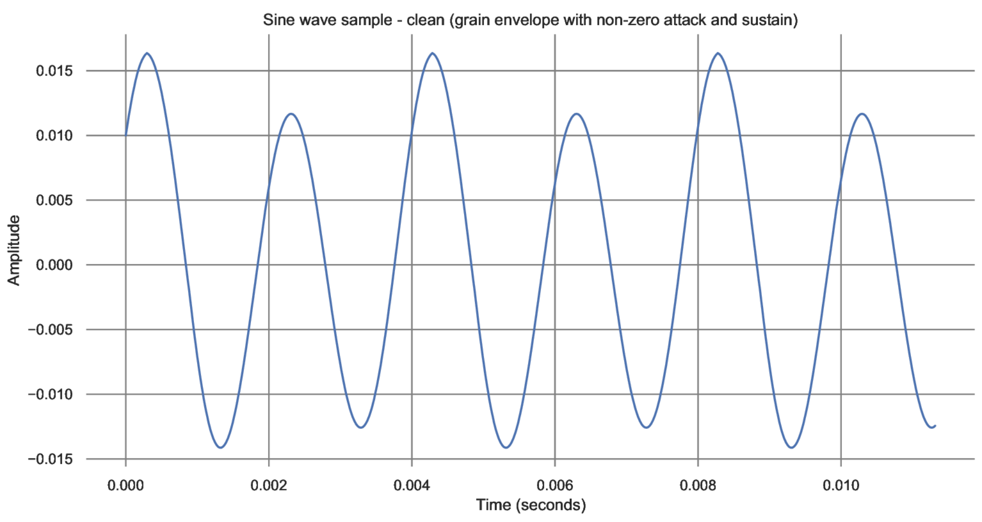


Figure 4.2: Clean output of the plugin with non-zero attack/release

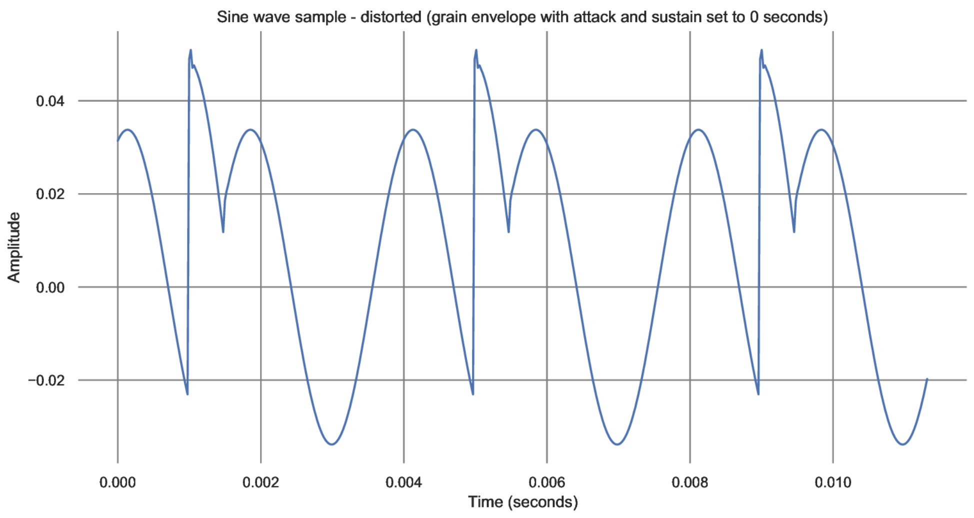


Figure 4.3: Distorted output of the plugin with square wave grain envelope

Another imperfection arises when setting the *Rate* knob to produce more than ~100 grains per second. After passing this threshold, the output of the plugin becomes sonically unpleasant for speech and drum samples. The plugin doesn’t create unwanted artifacts because of the amount of grains, rather it doesn't produce satisfying output when the grain rate is set to such speed, at least not with samples that a producer would typically consider using in music production. Examples of such samples are a recording of speech, singing, drum hits and natural sounds like dripping water.

4.2 Fruity Granulizer

Fruity Granulizer[2] is a granular synthesis plugin native to FL Studio[3] DAW. It is available for free as a part of all paid versions of FL Studio, starting at € 89 for the "Fruity" version, but can be tried out for free in the FL Studio trial version.

It works as a sampler and unlike The Mangle, it plays back the entire sample, playing grains one after each other, until the entire sample has been played in this way. It is quite simple in comparison to the other selected plugins, as it doesn't have any visual aspects apart from displaying the loaded sample waveform and contains just a few control knobs. Fruity Granulizer also has the ability to perform real-time time stretching.

In comparison to The Mangle, there is no problem with unwanted artifacts being created by setting "ATTACK" knob to the lowest value. This knob sets both the attack and decay stages of the grain envelope and because the minimum value that a user can set is 5 miliseconds, this allows the grains to fade in and fade out smoothly, not causing any artifacts.

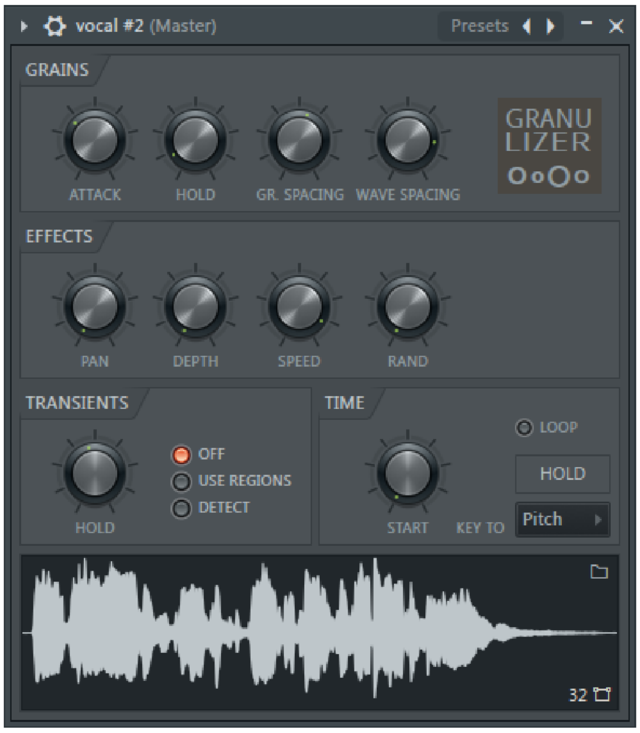


Figure 4.4: Fruity Granulizer plugin with a loaded sample

[2]Fruity Granulizer - https://www.image-line.com/fl-studio-learning/fl-studio-online-manual/html/plugins/Fruity%20Granulizer.htm

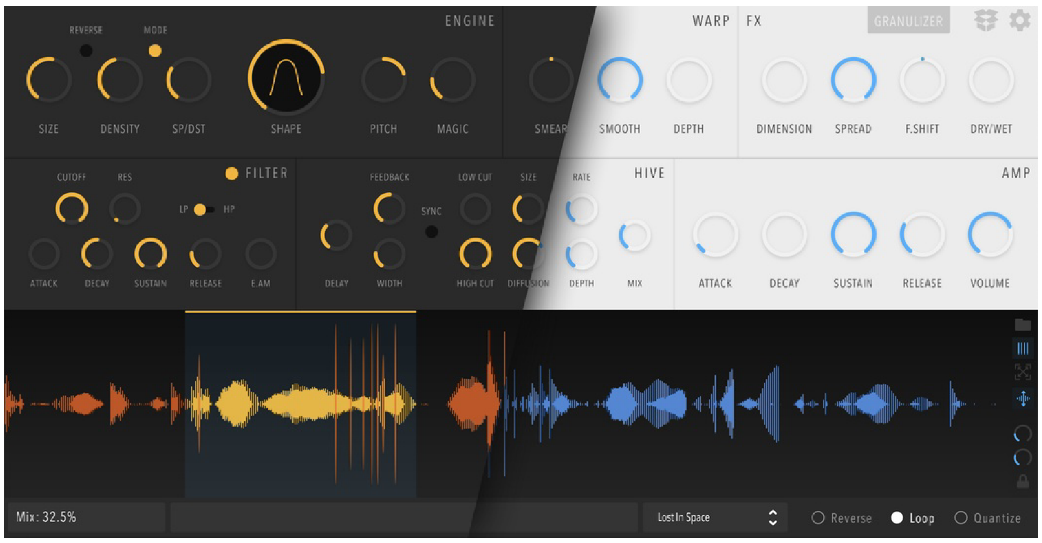
[3]FL Studio - https://www.image-line.com/fl-studio/

Last thing to note about this plugin is the “Transients“ section. This section enables aligning of grains to transients, which can improve output quality on certain sample categories, like drum samples. User can turn this feature on or off, with the option to autodetect or process it from the sample slices if there are any [6].

When testing with a snare drum, vocal and a sine wave samples, no measurable artifacts or sonically unpleasant output was being generated by the plugin, even when trying to set the plugin controls to extreme values.

4.3 Granulizer 2

Granulizer 2[4] is a granular synthesis plugin available in VST, VST3 and AU formats, created by Inertia Sound Systems[5]. It costs € 80 and has a free trial version that works for a limited time.



Mix: 32.5%

Lost In Space

Reverse [✓ Loop] Quantize

Figure 4.5: Granulizer 2 plugin. Source: [7]

Granulizer 2 is a sampler and similar to Fruity Granulizer, it plays back the entire sample, unless changed with the added option to select a specific part of the sample only. This gives the user the advantage of not having to splice the sample if there is a specific part of the sample that the user wants to generate grains from. Grain envelope can be changed by adjusting the “Shape“ knob, which smoothly transitions from a sharp envelope shape to a more rounded one and finally a flatter, almost rectangular shape, as can be seen in figure 4.6. Grain sizes can vary from 10 miliseconds to 300 miliseconds.

During testing, wide sounding, long duration sounds were easy to achieve with sonically satisfying results. Nevertheless, when trying to create rapid sounds with small grain lengths and high grain density, the output of the plugin wasn’t satisfying.

[4]Granulizer 2 - https://www.inertiasoundsystems.com/store/products/granulizer-2/
[5]Inertia Sound Systems - https://www.inertiasoundsystems.com/

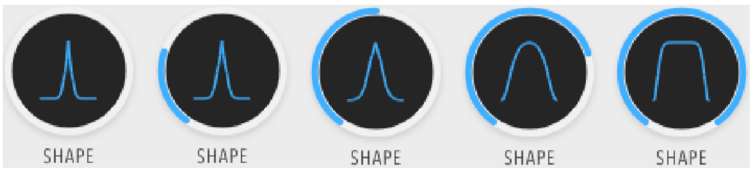


Figure 4.6: Examples of grain envelopes available in Granulizer 2

4.4 Granulator II

Granulator II[6] is a plugin made by Robert Henke[7]. The plugin itself is free, but comes exclusively for owners of Ableton Live[8] DAW (version 9.1 or higher) and Max for Live[9], which is included in a Suite version of Ableton Live and costs € 599, but it can be tried out for free together with Ableton Live trial version.

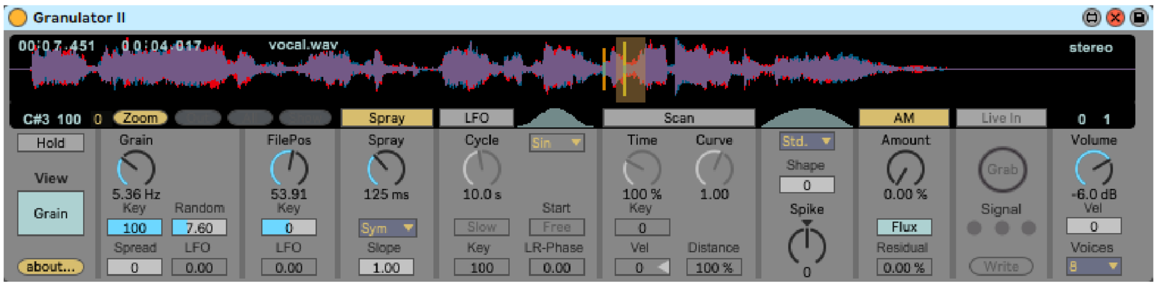


Figure 4.7: Granulator II plugin

Similar to The Mangle, Granulator II generates grains at a certain rate from a selected position in the sample, with the added option of a "Spray" knob, which makes the starting position of the grains to be selected at random from a specified interval, left or right to the marked position in the sample. Just like the previous plugins, it contains controls that change the grain envelope. User has an option to select from four predefined envelope shapes: standard, decaying/falling, rising and "Noiz" shape, which is a noise waveform. These four envelopes can be then modified by "Shape" and "Spike" parameters, to create a new, unique envelope shape.

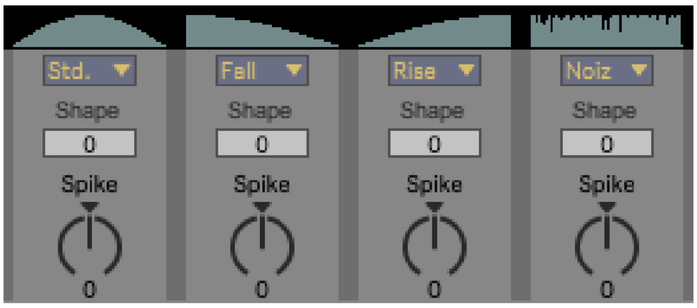


Figure 4.8: Examples of Granulator 2 grain envelopes

[6]Granulator II - https://roberthenke.com/technology/granulator.html
[7]Robert Henke - https://roberthenke.com/interviews/bio.html
[8]Ableton Live - https://www.ableton.com/en/shop/live/
[9]Max for Live - https://www.ableton.com/en/live/max-for-live/

The plugin provides visual feedback to the user by displaying the selected position in the sample together with a playhead, signalizing current playback position. Despite having means of creating rapid-firing grain clouds, the audio output doesn't get unpleasant and no clipping occurs.

4.5 Conclusion

In conclusion, key features that should be included in the final program were identified thanks to the market research. Mainly on the visual side, seeing a playhead and being able to keep track of the current playback position is crucial in deciding which parameters to change next. All of the problems with researched plugins that surfaced during testing were taken into consideration and served as indications of what to be wary of during implementation of the final program.

Chapter 5

Conceptual design and prototyping

This chapter will cover description of conceptual design for granular synthesis application and creation of a static, non-realtime prototype for a granular synthesis application created in Python programming language.

5.1 Conceptual design

Based on the plugin market research, decision to create a sampler based application was made. This allows for freedom of choice of the sonic content used for synthesis, by being able to move to a different position in a file, selecting the part of the sample with the most interesting resulting playback. Also, by having the entire sample stored and available, processing or analysis can be done prior to playback.

Synchronous granular synthesis was chosen as a baseline for the algorithm. Our resulting application should be able to start generating grains from a specific specific starting point and move forwards or backwards in the sample file, essentially moving forwards or backwards in time. The grains should be generated in a linear, synchronized manner, one after the next with a set delay between them, called spacing. The size and shape of the envelope should be modifiable, together with option to change the stereo panning of the individual grains, to allow for spatial effects.

The parameters are as follows:

- **Grain size** - 1 to 100 milliseconds in length
- **Grain shape** - Hann or Gaussian window
- **Grain panning** - able to set 0% - 100% of amplitude to either left or right channel, keeping the other channel intact
- **Grain spacing** - either spacing set in milliseconds or as a percentage of grain length, allowing overlapping for many grains (will be decided exactly by testing)

As mentioned in the plugin research conclusion, for the final application a waveform display along with a playhead is crucial, to allow the user to see where the grains are currently being generated from.

5.2 Static granulizer prototype

The prototype was implemented in Python as a *jupyter notebook*. The goal of creating the prototype being the testing of different concepts of granular synthesis and evaluating whether they were understood correctly. By doing so, correct implementation in a different language could be achieved, regardless of the differences in implementation details.

Four samples/signals were used when working with the prototype. First sample is a ~7 second long recording of a female singer. Waveform of this sample can be seen in figure 5.1. Speech and singing samples are often used in granular synthesis, since there can be many interesting textures found in human speech.

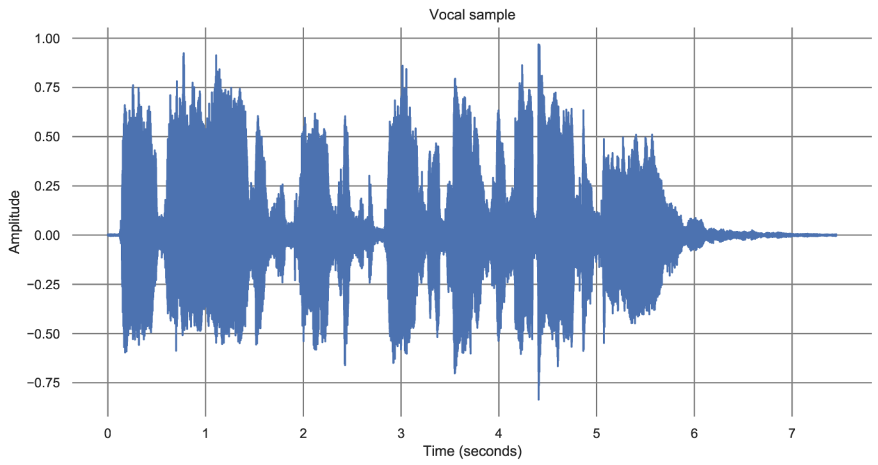


Figure 5.1: Vocal sample

The second sample that is used is a ~0.7 seconds long snare drum sample, with the actual snare drum part being ~0.2 seconds long. Waveform of this sample can be seen in figure 5.2. Drum samples are often used in granular synthesis with settings of low grain size together with high grain density in order to create rapid firing sounds. They can also be used by changing the panning of the grains, to create a feeling of the drum moving around the listener and being hit rapidly in succession. Most of the content of the sample is in the first 0.1 seconds, which shows that granular synthesis can make use of really short sounds.

Third sample is a simple ramp signal, shown in figure 5.3, used for better visualization of changes to a signal made by the granulizers. It was used when testing the granulizers mentioned in chapter 4, to better understand how exactly some controls work.

Overlap and add methods were tested at first, but after realizing that the type of synthesis that was the aim of this thesis could be done without this analysis and synthesis process, this approach was abandoned. Parameters like grain density were tested, grain overlapping, position randomizing and were also tested.

The last sample is a two second long sine wave, used to detect possible unwanted distortion/artifacts in the output signal.

Windowing functions from *scipy* library[1] were used as envelopes. By multiplying the grain envelope with a slice of a loaded sample file using Python *slicing*, grains were created.

[1]Window functions - https://docs.scipy.org/doc/scipy/reference/signal.windows.html

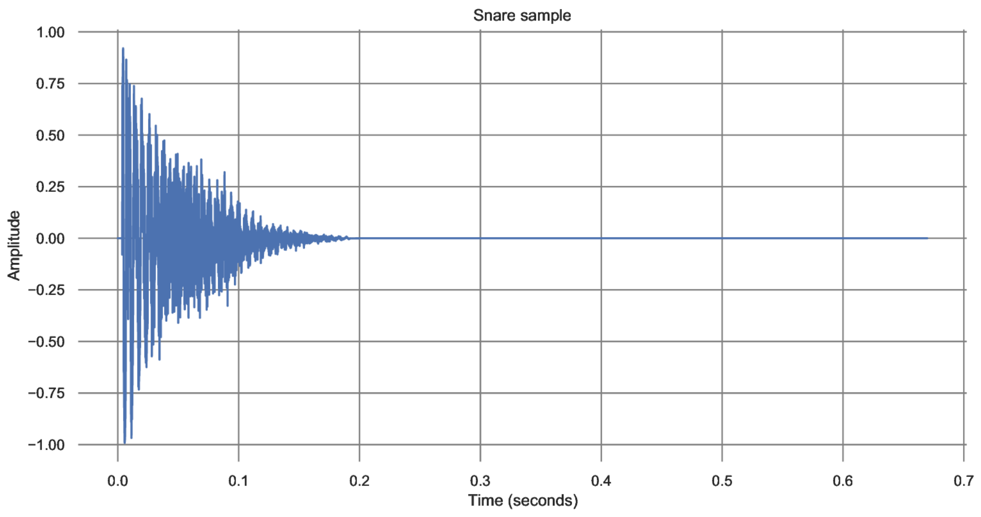


Figure 5.2: Snare drum sample

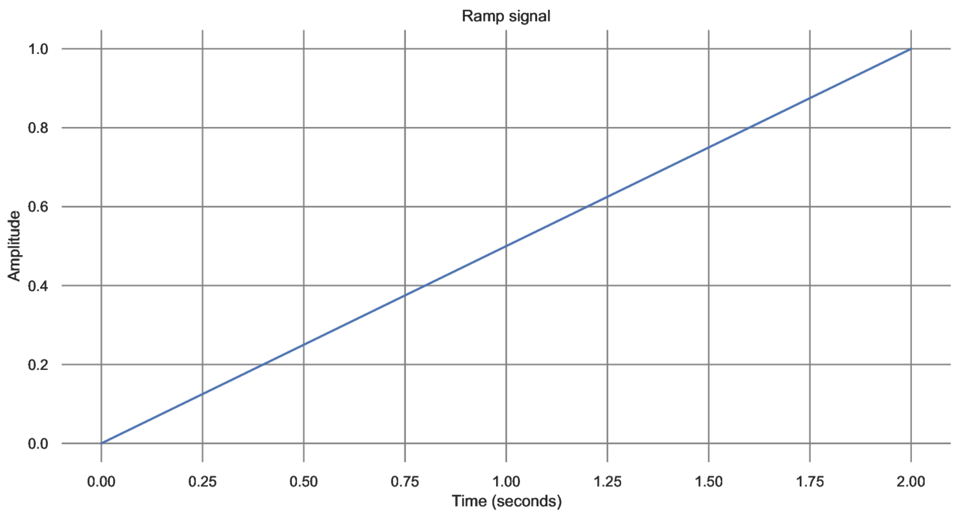


Figure 5.3: Ramp signal

Variable length of these envelopes was used to create grains for all of the samples and the sound output was both plotted and listened to, to analyze the outcome.

5.3 Conclusion of prototyping

Static prototype cleared up some initial misconceptions about the process of granulizing, allowing easier implementation of the final application. Block by block (or grain by grain) approach was envisioned as the solution to this problem and the parameters like randomization of position were selected as candidates for implementation.

Chapter 6

Implementation

This chapter deals with describing implementation of a granular synthesis plugin in JUCE framework using C++ programming language and the VST standard. Firstly, the JUCE framework itself is briefly introduced, followed by the description of the application design. At the end of the chapter, suggestions for further improvements on the currently available systems are given.

6.1 JUCE framework

JUCE is a framework for development of multi-platform audio applications, containing support for *Windows*, *MacOS*, *Linux*, *Android* and *iOS* operating systems. The JUCE framework also supports development of plugins with VST, AU and AAX formats. The framework provides the programmer with many different graphical components[1] to choose from without the need to design one's own, such as sliders, scroll bars, buttons, progress bars and others.

Figure 6.1: JUCE framework logo. Source: [8]

One of advantages of JUCE framework is that it has well written and updated documentation[2], with examples often included. There is also a page with numerous tutorials[3] as a part of the documentation, showing different concepts and their execution inside the framework, which help developers new to this framework to quickly get oriented, since there are numerous projects that one can use as a baseline. Another valuable resource is

[1]JUCE: Component Class - `https://docs.juce.com/master/classComponent.html`
[2]JUCE documentation - `https://docs.juce.com/master/index.html`
[3]JUCE Tutorials - `https://juce.com/learn/tutorials`

the JUCE Forum[4]. Apart from the mentioned JUCE resources, another valuable resource was *KVR audio* forum[5].

6.2 Petaldance plugin

The main focus of this thesis was creating this application – VST3 plugin called *Petaldance*. It is a sampler based plugin, which means that it takes a sample file and the playback data comes entirely from this loaded sample file. No other input such as microphone or some other source from DAW is inserted into the plugin.

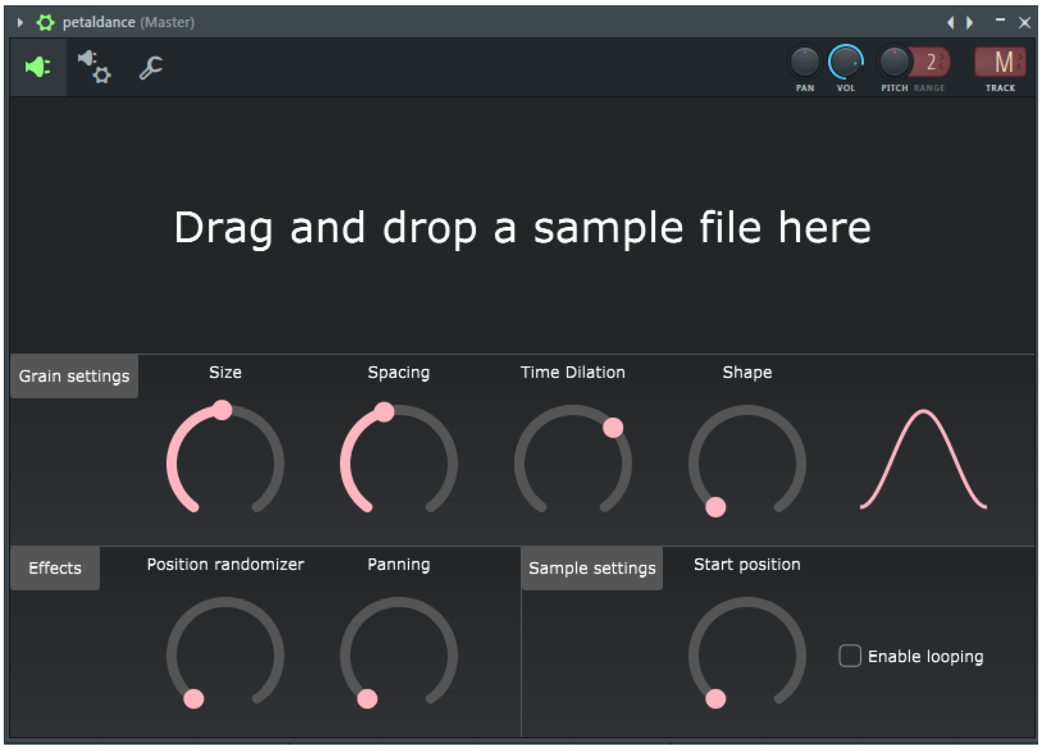


Figure 6.2: Petaldance plugin loaded in FL Studio DAW

Upon opening the plugin, the user is greeted by graphical user interface shown in Figure 6.2. The plugin waits for user to drag and drop a .wav or .mp3 sample file on the top section of the plugin with the text „Drag and drop a sample file here“. After user drags and drops a sample file, it will get loaded and stored inside the plugin. After this step, plugin will look like in Figure 6.3.

At this stage, the plugin is ready for playback, but nothing is happening yet. The plugin awaits for midi input from the host DAW, provided by the user. This can be achieved either by pressing keys on computer keyboard (keybinds are specific for each DAW) or by having connected and configured midi keyboard.

After pressing a key, playback starts happening from the specified position in the sample, with the playback settings decided by the values of the slider components. During playback, user can see a playhead indicating the current position in the sample where the grains are being generated from. The playhead can be seen in Figure 6.4

[4]JUCE Forum - https://forum.juce.com/
[5]kvraudio - https://www.kvraudio.com/

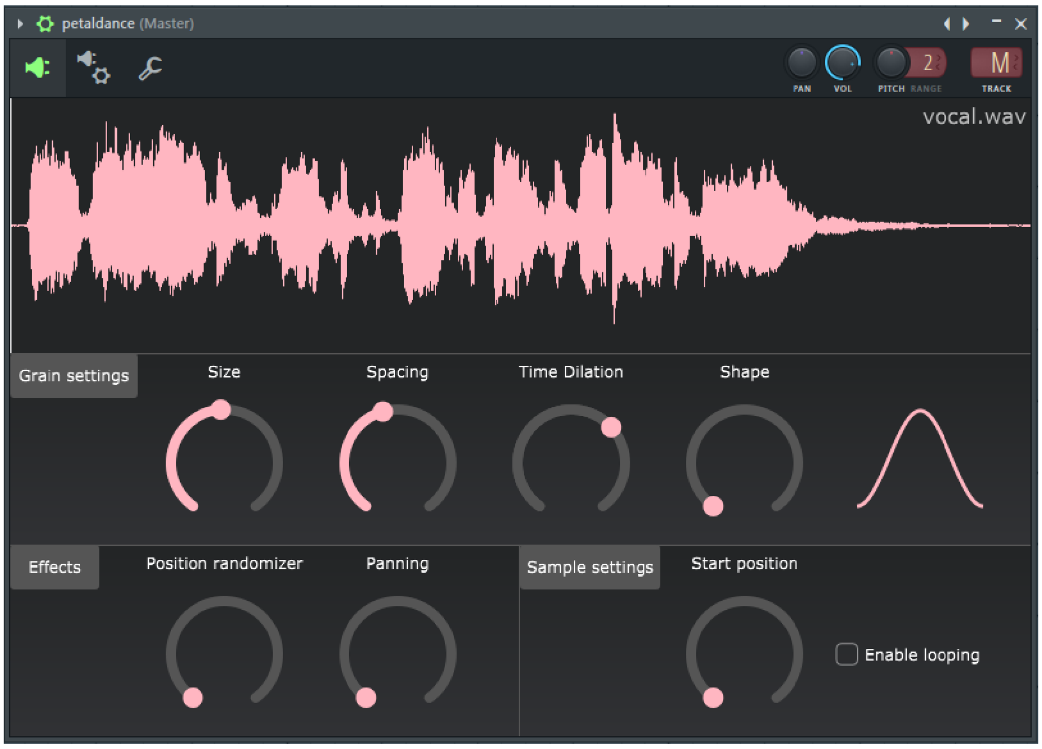


Figure 6.3: Petaldance plugin with a loaded sample file

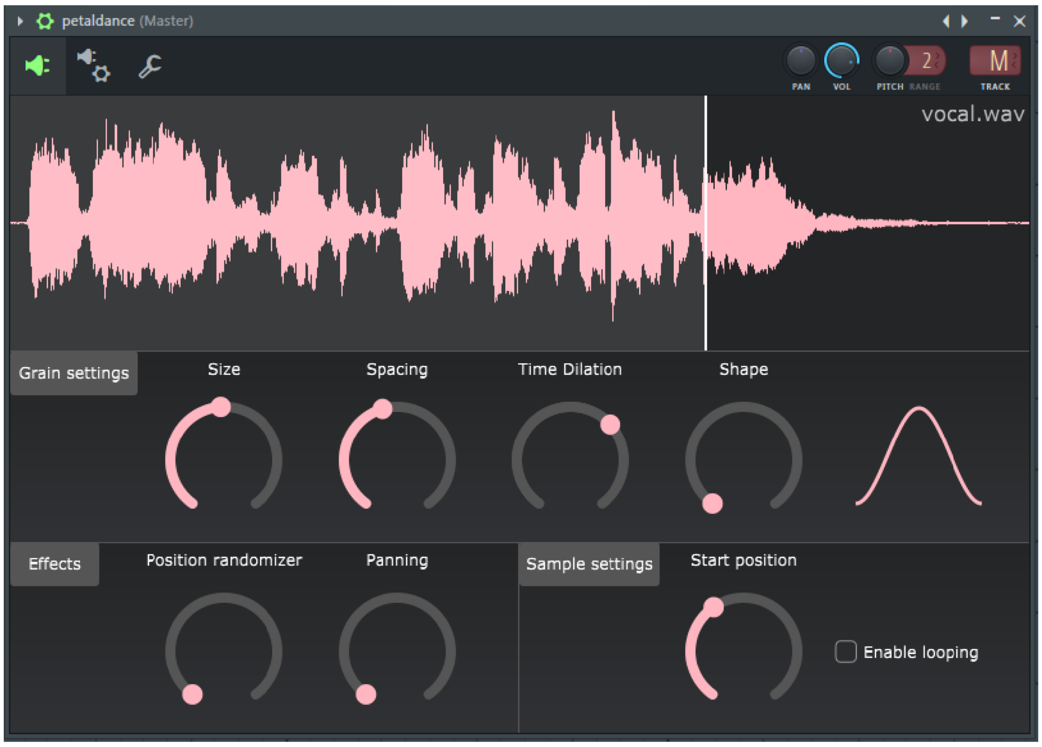


Figure 6.4: Petaldance plugin with a visible playhead during playback

Implementation details

Files with the prefix `UI*` are user interface component classes, containing GUI controls. Each of these classes has its own set of controls that it positions inside itself, such as sliders, plus their settings. These component classes are then contained within, and positioned in the `PluginEditor` class, which is the main JUCE graphical class, representing the entire application window.

For the granulizer functionality, classes `Grain` and `Granulizer` are implemented. `Grain` class represents grains and instances of this class get created with a specific lifetime, after which they are destroyed. The `Granulizer` class contains methods for granulizing the output.

JUCE's `PluginProcessor` class[6] is the main audio procesing class. Entire signal processing happens inside the `processBlock` method of this class. This method contains two parameters – `AudioBuffer<float> &buffer` and `MidiBuffer &midiMessages`. The `buffer` variable contains any input data, which is not the case of this application, since it doesn't accept any input such as a microphone. The second parameter contains *MIDI* messages, example being a message signalizing that a certain note was pressed or released. The MIDI message also contains other information, such as note velocity. This MIDI buffer needs to be processed in case working with MIDI input is desired, as is the case with this plugin. The `processBlock` method gets repeatedly called depending on the buffer size and the set sample rate that the playback is running on – if the sample rate is set to 44100 Hz and a block size to 441, the method gets called one hundred times a second. Each MIDI event gets included in the next `processBlock` call and should be processed inside it, otherwise it gets output by the plugin as output MIDI message.

In order to affect the sound output, the `buffer` needs to be filled with the desired audio data. This is done by looping over the individual samples of this variable and calling the `applyGrainToBuffer` method of the `Granulizer` class. This method takes the grain parameters, calculates the selected envelope for the current value and stores the result of the calculations into the buffer, essentially windowing that specific sample. This happens for every sample of the buffer, for every new buffer that gets created by the `processBlock` method as means of audio playback. The grain has to be currently active for it to make changes to the buffer, which is checked by `shouldGrainPlay` method of `Granulizer` class.

Generating of grains is done inside `run` method of `PluginProcessor` class. `Run` is a method implemented because of `PluginProcessor` inheriting from the JUCE's `Thread` class[7]. This allows for use of a thread inside the application. This is crucial, because no memory allocations, like creation of new grains, should be done on the main audio thread, for example in the `processBlock` method. The reason being it could hinder the playback and cause unexpected artifacts. Once playback starts, one grain is generated and time that the next grain should start is noted. Time is counted as a number of samples that have been processed so far and since the `AudioProcessor` processes exactly the amount of samples as is the currently set sample rate, this can be used as accurate timer. As mentioned, the grains are created on this thread. The thread is setup in a way such that it sleeps when not needed and is awakened by `notify()` when needed. If no playback was happenning, pressing a note causes the program to send notify signal to the thread, which starts the first grain and then sleeps for a duration based on the size of the grain and the grain spacing

[6]AudioProcessor class documentation - https://docs.juce.com/master/classAudioProcessor.html
[7]Thread class - https://docs.juce.com/master/classThread.html

parameter. In case the note gets released (and no new note started prior to this), the thread is stopped and all currently existing grains are deleted.

Midi notes are processed to determine whether a sample playback should be happening or not. This is done simply by incrementing or decrementing `numOfPlayedNotes` variable based on the captured note events – `.isNoteOn()` and `.isNoteOff()`. In case there is one or more notes being pressed, the playback of the sample starts from the currently selected position set by *Sample position* slider, in the currently chosen direction by *Time Dilation* slider, with the grain parameters of *Size* and *Spacing* sliders. Once the notes are released, the playback stops, any currently active grains get deleted and the playback position is set back to the value before playback.

6.3 Problems during development

During development, multiple problems arised from the differences in the host digital audio workstations. One such example is the different size of buffer that is processing the audio output inside `processBlock` method of JUCE's `AudioProcessor` class, which handles sound output. It is mentioned on the JUCE documentation page of `AudioProcessor` that the buffer size can be variable and depends on host. In Ableton 11, stable buffer size of 256 samples was always present. The buffer size in FL Studio was rapidly changing from 1 up to around 500 samples, each buffer having different size. This can be a non issue or potentially critical depending on the type of application that needs to be developed. My solution to this problem was to process the block sample by sample, instead of previously envisioned and planned processing on block basis, as was done in the prototype.

6.4 Improvements on the current systems

One of the possible improvements on the current granular synthesis systems, that are sample based or use samples in some way, would be sample file analysis. Since the sample file is static and the entire content of it is known, various analysis methods could be used to enrich the grain playback process. For sample containing some form of speech – either spoken or sung – detection of changes in fundamental frequency can be used. Fundamental frequency is a frequency the vocal cords are oscillating on thus determining the pitch of the voice. Decisions could be made with this information to alter the playback, for example changing the sample file position based on changes of the pitch, when a singer stops holding a note and changes to another. Such approach of course wouldn't work with non-speech samples such as drum hits.

Chapter 7

Testing

7.1 Application testing

During development, the plugin was continually tested on operating systems *Windows 7* and *Windows 10* and in two digital audio workstations – *FL Studio* and *Ableton Live.* FL Studio version 12 and 20 was used and Ableton Live version 11 was used. The developement was done using *Visual Studio Community 2019* with the option to attach to a running process, the process being Ableton Live 11. With Visual Studio attached in this way, it is possible to set breakpoints, see debug output, observe variables in the watch window and debug the plugin just like a standalone application. It would be almost impossible to debug the application without this option. For debug output, the `DBG()` macro is available to the programmer and there is a `FileLogger`[1] class for more robust logging, both available from the JUCE framework, so there is no need to write one's own methods for logging.

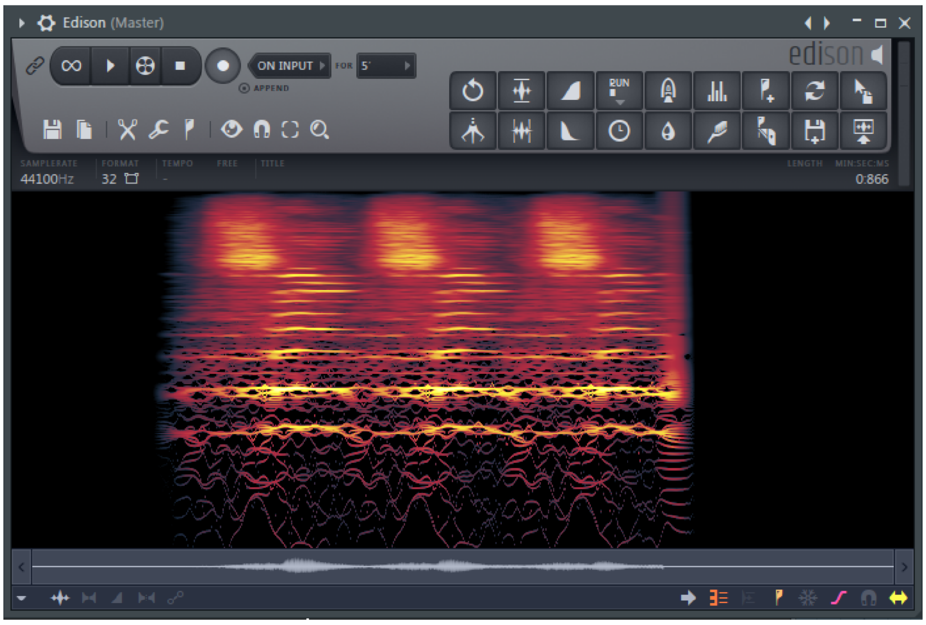


Figure 7.1: Plugin Edison

[1]FileLogger class documentation - https://docs.juce.com/master/classFileLogger.html

For the sound output analysis, plugin native to FL Studio called *Edison*[2] was used. This plugin is an audio recording and editing tool, with both spectral and waveform view, with the option to zoom all the way to individual samples. The plugin can be seen in Figure 7.1. This allowed inspection of the output of the plugin in cases when there were unwanted artifacts.

7.2 User tests

User tests were designed to test the application and to assess satisfaction with the graphical design, design of the controls and to decide whether it is usable for music production as a granular synthesis instrument. The survey was done on 7 participants selected from a music production background – music producers, DJs, performers or hobbyists with interest in music production. The users were presented with a manual on how to set up the plugin in order for it to open in a digital audio workstation of their choice, short description on what the plugin does and description of the controls. An example of control configuration was given, together with a ~2 minute demonstration video, showing how to work with the plugin. Users were asked to work with the plugin just as they would when obtaining a new plugin from the internet and were then asked to provide their feedback by filling out a short form with questions. The form contained following questions that could be answered either yes or no:

- Have you worked with a granular synthesis plugin before?
- Do you find the graphical design of the application clean looking?
- Do you find the controls of the plugin intuitive?
- Is it obvious where in the sample are the grains currently generated from during playback? Meaning where is the current playback position?

The next part of the form asked users to rate the appearance, controls and usability of the plugin on the sound side. A point system was used, with a range from 5 to 1 points, the more points meaning better rating. The asked questions were:

- Please rate the appearance of the plugin.
- Please rate the control of the plugin.
- Please rate the usability of the plugin in music production on the sound side.

Arithmetic mean of the results of the point system questions can be found in Table 7.1.

Question	**Point range**	**Point score (mean)**
Appearance	5 – 1	3.86
Control	5 – 1	4.0
Usability	5 – 1	4.29

Table 7.1: Results of the user testing survey

[2]Edison - https://www.image-line.com/fl-studio-learning/fl-studio-online-manual/html/plugins/Edison_2.htm

The survey results indicate that the plugin, despite having room for improvements, is deemed as easy to use, clean looking granular synthesis instrument that can be used for music production. The suggestions for improvement from this survey, mentioned in Appendix A can be used as inspiration for future work.

Chapter 8

Conclusion

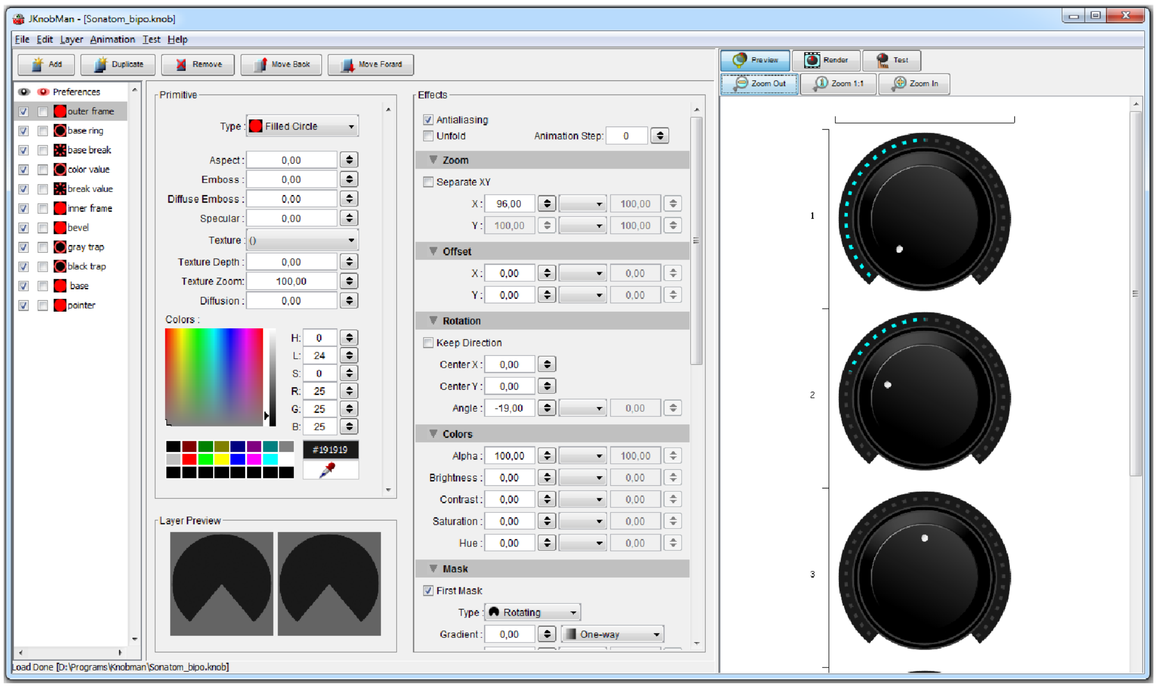
8.1 Summary

In this work, we have got acquainted with the principles of software systems used for music production. We have learned about the principles of granular synthesis and used this knowledge, together with the findings from market research and creation of a prototype, to create a virtual instrument that is able to perform granular synthesis. The virtual instrument was implemented as a sampler based VST3 plugin, capable of interfacing with digital audio workstations such as FL Studio and Ableton Live. A brief look into possibilities of improving on current granular synthesis systems was taken, with the focus on analyzing the sample itself. Lastly, user tests were performed and evaluated, resulting in the plugin being deemed as simple, clean and usable for actual music production. Suggestions for further improvement and additional features also arised from the user testing, giving inspiration for future work.

8.2 Future work

Many different ideas came to mind during the plugin market research and during the development itself. Other suggestions for improvement come from the performed user tests. One of the possible improvements that was mentioned in the user tests would be to create a set of plugin *presets*. A preset is a set of predefined values for the plugin controls, such as slider values, or checkbox state in the case of this application. The presets are usually themed for a specific use – example being a preset for vocal samples with predefined values that set up the plugin to produce interesting output for this category of sample files. Another potential improvement could be on the graphical user interface part – to use custom created slider images that map to the rotary value of the sliders such that they look and feel like knobs from a real life instrument or a piece of hardware. This could be done with Knobman[1]– a freeware image strip design tool used for creating knobs for virtual instruments. Knobman application with an image strip can be seen in Figure 8.1.

[1]Knobman by g200kg - https://www.kvraudio.com/product/knobman-by-g200kg



Type : Filled Circle

Aspect : 0,00
Emboss : 0,00
Diffuse Emboss : 0,00
Specular : 0,00
Texture : ()
Texture Depth : 0,00
Texture Zoom: 100,00
Diffusion : 0,00

Colors :
H: 0
L: 24
S: 0
R: 25
G: 25
B: 25
#191919

[✓] Antialiasing
[ ] Unfold
Animation Step: 0

Zoom
[ ] Separate XY
X : 96,00 100,00
Y : 100,00 100,00

Offset
X : 0,00 0,00
Y : 0,00 0,00

Rotation
[ ] Keep Direction
Center X : 0,00
Center Y : 0,00
Angle : -19,00 0,00

Colors
Alpha : 100,00 100,00
Brightness : 0,00 0,00
Contrast : 0,00 0,00
Saturation : 0,00 0,00
Hue : 0,00 0,00

Mask
[✓] First Mask
Type : Rotating
Gradient : 0,00 One-way

Load Done [D:\Programs\Knobman\Sonatom_bipo.knob]

Figure 8.1: Knobman application with an image strip on the right side

Bibliography

[1] *Audio Units - Licensing and Trademarks - Apple Developer* [online]. 2022 [cit. 2022-03-29]. Available at: https://developer.apple.com/licensing-trademarks/audio-units/.

[2] *Mixer Explained* [online]. 2022 [cit. 2022-03-20]. Available at: https://www.image-line.com/fl-studio-learning/fl-studio-online-manual/html/mixer.htm.

[3] *Piano roll* [online]. 2022 [cit. 2022-03-20]. Available at: https://www.image-line.com/fl-studio-learning/fl-studio-online-manual/html/pianoroll.htm.

[4] *The Playlist* [online]. 2022 [cit. 2022-03-20]. Available at: https://www.image-line.com/fl-studio-learning/fl-studio-online-manual/html/playlist.htm.

[5] *Lifetime Free Updates - FL Studio* [online]. 2022 [cit. 2022-03-20]. Available at: https://www.image-line.com/fl-studio/lifetime-free-updates/.

[6] *Fruity Granulizer - Instrument* [online]. 2022 [cit. 2022-02-09]. Available at: https://www.image-line.com/fl-studio-learning/fl-studio-online-manual/html/plugins/Fruity%20Granulizer.htm.

[7] *Granulizer 2 | Inertia Sound Systems* [online]. 2022 [cit. 2022-02-11]. Available at: https://www.inertiasoundsystems.com/store/products/granulizer-2/.

[8] *JUCE* [online]. 2022 [cit. 2022-04-07]. Available at: https://juce.com/.

[9] Opie, T. *Sound in a Nutshell: Granular Synthesis*. Melbourne, AU, 1999. BA Honours Thesis. La Trobe University.

[10] Roads, C. Introduction to Granular Synthesis. *Computer Music Journal*. The MIT Press. Summer 1998, vol. 12, no. 2, p. 11–13.

[11] Roads, C. *Microsound*. The MIT Press, 2004. ISBN 9780262681544.

[12] *Our Technologies | Steinberg* [online]. 2022 [cit. 2022-03-29]. Available at: https://www.steinberg.net/technology/.

[13] *VST 3 SDK: Introduction* [online]. 2022 [cit. 2022-03-29]. Available at: https://steinbergmedia.github.io/vst3_doc/vstsdk/index.html.

[14] *VST2 discontinued - Announcements - Steinberg Forums* [online]. 2022 [cit. 2022-03-29]. Available at: https://forums.steinberg.net/t/vst-2-discontinued/761383/2.

Appendix A

User review of the application

This appendix contains feedback collected from the most experienced users testing this application. They were asked to introduce themselves and provide feedback for the application along with any suggestions for improvements. As the feedback was provided in the Czech language, it was kept in the original form instead of being translated.

Radek Janků - music producer, artist

Vzhled na první pohled nabídne vše, co od pluginu potřebujeme. Rozmístění jednotlivých prvků je přehledné, v pluginu se při práci dá velmi jednoduše a rychle zorientovat. Oceňuji jednoduché drag-n-drop přidání samplu, se kterým chci pracovat. Jednotlivé ovládací prvky mi nicméně po chvíli přišly možná až moc velké, zabírající zbytečně moc prostoru a trochu vizuálně nudné.

Plugin jsem použil v DAW Ableton Live 11, kde šel spustit naprosto bez problémů. Ocenil jsem, že plugin s DAW spolupracuje a je možné jej například namapovat na knoby a ovládat pomocí hardware zařízení. Ovládání i díky tomuto mapování je jednoduché a intuitivní a s pluginem si jde po přidání vhodného samplu velmi pěkně pohrát.

Co se týče funckionality, nabízí plugin spíše základní možnosti. Chyběla mi zde možnost nastavit si více „start position“ bodů, tedy bodů, ze kterých se začne daný sample přehrávat. Pokud by bylo možné namapovat různé start position body na různé pady či klávesy na hardware zařízení, dokázal bych si jeho využití reálně představit například při živých vystoupeních. Dále by určitě zrychlení workflow a přípravy, například na zmíněné live vystoupení, pomohla možnost využít několika přednastavených presetů a hlavně možnost uložit si vlastní nastavení a toto nastavení si poté později jednoduše otevřít. Pokud by plugin nabízel tyto prvky, je dle mého bez problému srovnatelný s profesionálními placenými pluginy stejného zaměření.

I přes těchto pár nedostatků mi však plugin přijde reálně použitelný například při nahrávání různé elektronické muziky, ve které se hudebník nebojí experimentovat s vlastními samply.

Jakub Kopecký - music producer, artist, DJ

Vzhled pluginu je velmi čistý/minimalistický a z ovládacích prvků je víceméně jasné, k čemu slouží. I vizualizace samplu je přímočará, jen bych v ní ocenil znázornění grainů jako takových - ať si dokážu představit, jak bude výsledek znít. Dále bych také uvítal nějaké znázornění defaultních pozic všech sliderů + možnost syncu s tempem projektu, každopádně i přes tyto „neduhy“ si dokážu představit praktická využití tohoto pluginu, zejména při experimentaci s různými samply, automatizací apod.