

Name: Olivier Skrzypek

Student Number: 1904614

Module Title: Development (ADAD7005_LON)

Assignment Title: Foundation Assessment

Module Lecturer: Ciarán Robinson

Introduction

The goal of this essay is to analyse the approach, methodology and reflection which led to the creation of my own software instrument, with reference to research, discussion and feedback gathered in the process. The instrument in question is a granular synthesizer and sampler which allows for complex manipulation of pre-existing audio files, most notably producing “grains”. These are short pieces of audio (usually 1 to 50ms) which can be triggered simultaneously at different speeds, volume and frequencies, among other parameters (Roads, 2001).

Depending on the playback speed, granular synthesis can be used to create anything from lush, evolving soundscapes to clusters of audio information, complex rhythms and sound effects (Eckel, Rocha and Becker, 1995). Many electronic music producers take advantage of granulation and its nearly endless sound design capabilities. One of the most notable recent works, almost exclusively based on granular synthesis, is Flume’s Grammy-nominated album “Hi! This Is Flume”. The Australian producer tastefully warps melodies and meddles with scattered drums and basslines to blend them into a glitchy symphony of sonic chaos and mayhem (Davies, 2019).

This technique dates back to the 1960s, with Iannis Xenakis being credited as the inventor of granular synthesis (Xenakis, 1992). Roads (1996) reports: “the composer Iannis Xenakis was the first to explicate a compositional theory for grains of sound. All sound, even continuous musical variation, is conceived as an assemblage of a large number of elementary sounds adequately disposed in time. In the attack, body, and decline of a complex sound, thousands of pure sounds appear in a more or less short interval of time”.

Despite the mathematical complexity of some of the granular models used today, granular synthesis is appealing to producers because of its conceptual simplicity: small fragments of sounds are overlaid to construct more complex sound material. All complexity of a particular

granular synthesis application lies in the way this basic concept is applied and how it is controlled by the composer (Eckel, Rocha and Becker, 1995).

Before starting to work on my instrument, I have outlined the key processes and tasks I wanted it to perform: grain production and manipulation, MIDI integration, polyphony, slicing and transposition of audio files. This provided me with a basic framework to follow and gave me a point of reference when approaching and delving deep into the creation process.

Approach & Methodology

1. Grain engine

My initial goal was creating a simple engine capable of generating grains from audio samples. To begin with, I needed the means of loading and storing audio information. The `buffer~ sound` and `dropfile` objects helped with that task. The combination of the two allowed me to drag and drop any audio file and `replace` it at any given moment. The `groove~ sound` object is *the core* of the patch. It is a variable rate, looping, sample-playback engine which references the audio information stored in the `buffer~` object with the same name (Cycling '74, 2020). It allows for playback speed, position and loop point manipulation, which was exactly what I needed to produce and process grains.

In order to be able to select a portion of the audio file to sample, I used the `waveform~` object and connected the third and fourth outlets (selection start and end) to the second and third inlets of the `groove~ sound` object (loop min and max). I introduced the `edge~` object, which deals with zero and non-zero information, to send a `bang` out to the `stop` object when the loop ended (Cycling '74, 2020). This allowed for precise selection within the audio file which would play out once, from selection start to selection end. It definitely was progress, but I was still very far from granulation. At this point, I ran into an issue with clicks. Knowing I aimed to trigger several grains simultaneously, the clicks would become unbearable and would compromise the

sound quality of the instrument. My fix was using a `trapezoid~` object to drive the amplifier and act as a simple envelope to fade audio in and out (Cycling '74, 2020).

With granulation, rather than determining the start and end points of the grain, the selection sets the start and range of the grain, with the length being controlled by another parameter (Batchelor, 2018). A *random offset engine* is a very common technique in granular synthesis, which facilitates this kind of processing (Roads, 1988). It uses a range of values to generate random numbers within that range. These values can stand for a variety of parameters which can then change and randomize over time. Random offset engines are definitely a backbone of my patch and I had used them repeatedly in the creation of my instrument. The main driving force behind such engine is the `random` object which simply outputs random numbers. The `!-` (bang, subtract) object makes it possible to calculate the exact selection range by subtracting the selection start value from the selection end value. In order for the engine to only produce values higher than the selection start point, I had wired the selection start value to the `+` object and added it to the `random` object's outlet. This ensured that the lowest, randomly generated number would be *selection start value + 1*, thus making the random offset engine work precisely within the limits of the selection range.

2. Grain parameters

At this point, the grain would trigger from a random place within the selection but would always play out until selection end. This was not ideal, as it still sounded very much like a loop, instead of a randomly generated grain. My approach was to introduce another random offset engine to manipulate the *grain length*. Since I was dealing with a separate parameter, I needed a new object to provide a range of values for the engine to work with. The `rslider` seemed like the simplest and best tool for that. I set its range to go from 0 to 2001ms, meaning the maximum grain length was 2 seconds. `Random` objects only deal with values one higher than the lowest and one lower than the highest in the range. I had connected the randomly generated start and

end values to the loop min and max inlets of the `groove~` object respectively. This meant the grain engine would play each grain from a slightly different position, for a slightly different duration anytime it was triggered.

The next parameter I planned to introduce was *grain transposition*. The principle was the same: using the random offset engine to generate values within a specified range. I had changed the resolution of the `rslider` to contain values from 0 to 49. My goal was to allow for transposition of 2 octaves up or down from the original sample pitch. In the MIDI world, C3 has a value of 60, while the `rslider`'s middle value was 24 (Batchelor, 2018). Since I wanted the middle C to correspond to the original playback speed, I had to add +36 to both outlets of the `rslider` object. Nonetheless, I could not feed this information to the `groove~` object, as it would not understand it. I needed a way of translating the MIDI note number to frequency, which was exactly what the `mtof` object (MIDI to frequency) assisted with. In order to determine the transposition amount – the change in playback speed – I divided the resulting frequency by the reference frequency of the middle C, which is 261.62558Hz (New York University, n.d.). Simply speaking, if the user played the note C5, the engine would calculate that its frequency is two times higher than C3, it would play back the sample at twice the speed, thus pitching the audio two octaves up.

The last parameter I manipulated in a similar way was the *grain speed*, which is the rate at which grains are played back. At this point, I had introduced the `metro` object as the master trigger. It is a metronome object which outputs a `bang` message at regular time intervals (Cycling '74, 2020). To make these intervals differ for every grain, I used yet another random offset engine to supply randomized values.

3. Polyphony

With all the parameters set up, my next objective was making the instrument polyphonic. I had to allow for multiple grains to be triggered at the same time to truly capture the sound of granular synthesis. Obviously, the main object for creating polyphony is the `poly~` object, but there was a number of steps I had to take before introducing it. First, I contained all the grain parameter information within a single `pack` object. This would help with efficiency and organization later on in the process. I had encapsulated the main engine and changed the default inputs and outputs to its polyphonic counterparts - `in 1` and `out~ 1` (Dehaan, 2016). I saved the grain engine as a separate patch (FRAGMENTS_ENGINE.maxpat) which allowed me to load it with the `poly~` object and increase the number of desired voices. I have found 100 voices to provide stable CPU performance without sacrificing the audio quality. The `prepend note` object, which tells the `poly~` object to find a spare instance of the loaded patch for playback (Cycling '74, 2020), was another vital element of the chain. I also needed a way of allowing the `poly~` object to identify whether a voice was busy or not, and whether it should be muted. I came up with a simple chain to help out with this process: I would send the “mute off” message to `thispoly~` whenever a new voice was triggered, and a “mute on” message whenever a voice ended (Cycling '74, 2020). I had also added voice stealing, which would turn off the oldest voice to make space for the new one if the `poly~` object received more notes than it had voices (Reid, 2001). This would prove beneficial when implementing the next function – *grain cluster*. The `uzi` object would generate multiple grains in rapid succession for each bang message from the `metro` object. This is sometimes referred to as *grain density* (Price, 2005). It creates the desired audio cluster effect, although leads to volume spikes if the output gain is too high.

4. MIDI integration

MIDI integration was a very important aspect in the creation of my instrument. I believed that allowing the use of a MIDI controller to interact with the patch would greatly improve its workflow speed, friendliness and intuitiveness. The `notein` object provided me with two types of crucial information: velocity and pitch. The first would send a simple on and off message to the master trigger and the `metro` object. The latter would have two interchangeable functions, controllable with a `switch`.

The default instrument setting is the “FRAGMENTS” mode. It automatically divides the loaded audio file into equal parts and maps them out across the keyboard. This allows for quick sample navigation, as every note is assigned to a different selection point within the sample. I obtained that by automatically sourcing the sample length from the `buffer~` object and dividing that by set resolution. The default is 88, which is the size of a full keyboard, but it can be changed to user’s preference. The higher the resolution, the more precise and smaller the audio slices.

I aimed to give the user the option to control the *selection length* by pressing two keys simultaneously, instead of simply jumping between time markers. The lowest note would set the selection start while the highest would mark its end. For example, with slicing resolution set to 12, pressing C1 and C2 at the same time would select the entire sample. Pressing F1 and A1, however, would only create a small selection in the middle of the sample. Such workflow makes a lot of sense with samples which change a lot over time, like loops, vocals or entire arrangements, as you can easily experiment with resampling various slices of audio.

The trickiest thing I had to programme was a way of making the instrument know when multiple notes were held and giving it an option to organize them from lowest to highest. I used the `counter` object to determine how many notes were being played at the same time and I introduced the `if` object with the following condition: `if $i3 > 1 then $i2 else $i1`. This meant

that if the number of pressed keys was smaller than 2 (single voice), the patch would make an arbitrary selection of 100ms from the selection start point. However, if multiple notes were pressed, it would find the highest note, calculate its value in relation to its position within the audio file, and send the `selectionend` information back to the `buffer~` object.

The other mode – “PITCHES” – overrides this chain and links the pitch value of the `notein` object to the grain pitch `rslider`. This is a very temporary way of portraying the main principle behind the mode, as it only allows for monophonic transposition. Ideally, it would be possible to play overlapping notes and chords. This would require a similar chain to the one used in “FRAGMENTS” – one able to identify and organize multiple notes being played. It would also need the ability to send the information about each note to an independent transposition tool to process each voice separately.

5. FX

Providing audio processing capabilities within the plugin is common practice in software instrument design. A number of leading synths, like Serum or Massive, offer that functionality. It gives the user the option to tailor and shape the sound even more, before the signal reaches the amplifier. It also saves on CPU power, as it limits the necessity of introducing additional plugins to carry out the basic processing (Price, 2005).

In order to provide context of what my finished instrument could sound like, I had introduced a couple of premade FX units, namely a filter and a chorus. The effects are very basic, but they can prove useful, especially when the sample sounds harsh and the grains are unpleasant. They can also be easily bypassed. I planned to introduce a wider range of effects, like reverbs or delay, but I was not happy with the sound of most units. Ideally, I would programme them myself, but it simply was not the focus of this project. Nonetheless, I am not crossing out the possibility of undertaking this challenge in the future.

Capabilities and limitations of instrument

“FRAGMENTS” offers a lot of sound design capabilities. It operates on pre-existing audio recordings, so its sound is hugely dependant on the quality and type of these samples. There is a number of parameters which can be tweaked to modulate the sound in a variety of unpredictable ways, but even the default settings provide immediate results. The instrument deals well with musical samples, drum loops or even full songs. It offers the ability to process these sounds to create complex textures, sound effects, pads, chord progressions and anything else in between. Thanks to the samples automatically being mapped out across the whole keyboard, it is very easy to navigate it and experiment with different loops and selections. The other parameters allow for in-depth modulation of the sound sources, from subtle alterations to granular sonic mayhem.

The “PITCHES” mode changes the role of the MIDI keyboard from slicing the audio files to transposing them. This works well with short, one-note samples, as it provides the user with the ability to play melodic parts. Of course, one could also try to experiment with different types of audio recordings to obtain unique results. The sample range selection is made manually, and there is no direct way to manipulate it with the MIDI controller. Overall, this mode could use the most improvement, because at the moment, it only allows for monophonic transposition. One technique which I found particularly efficient was blending both modes together. I would use “FRAGMENTS” to quickly scan across samples for interesting loop points and switch to “PITCHES” to manipulate their pitch and speed.

The main purpose of the instrument is also its main limitation. Since it works with pre-recorded audio, it does not offer the option to create sound from scratch. This might be uninviting for people who are passionate about other types of synthesis and enjoy having ultimate control over their sound. It also is not an accurate sampler for those looking to truthfully reproduce sound. It lies somewhere in the middle between sampling and synthesis.

In its present state, it also does not offer the option to modulate the parameters over time with envelopes or LFOs. All the grains have the same shape, set by the `trapezoid~` object, and there is no way to change it.

Reflection, adjustments and potential improvements

Overall, the instrument meets my initial goal. It has drag-and-drop functionality of loading and swapping audio files. It is capable of producing multiple grains of audio simultaneously, with randomized parameters like speed, pitch, position and length. It works with MIDI and allows for automatic slicing or transposition of audio files.

I have a number of ideas for potential future improvements. The biggest one is the development of the “PITCHES” mode and more advanced audio effects and processing capabilities. I believe introducing various slicing modes, like transient-based slicing, could be a great addition to the current patch. Adding LFOs and other time-based effects would provide the instrument with an extra layer of complexity. There are a couple bugs I am currently aware of. There seems to be an issue with automatic selection in “FRAGMENTS” mode which sometimes requires for a key to be pressed twice to trigger the engine. The `buffer~` object requires reloading on odd occasion, as it fails to provide the accurate sample length value and interferes with the slicing feature. Increasing the grain pitch resolution is one of the few smaller improvements that could be implemented. It could easily be done by multiplying the `rslider` values by 1000 to allow for fine-tuning, rather than operating in semitones only. Presets could also be added. A single *randomize* button for all the parameters could be an interesting addition as well. Automatic gain control, depending on the sample’s volume and grain density, would save the user’s time by automatically controlling the master fader for them. At this point, the process is about coming up with new ideas and optimizations and experimenting with them. The existing engine is a solid foundation that could easily be built upon in the future and it offers a lot of room for potential development.

Comparison to existing instruments

There are various granulators and granular synthesizers out there, but the market is not clearly dominated by a select few, like it is with other types of synthesizers. Some of the most notable mentions include The Mangle, Quanta, Granite and RIBS (despite being abandoned by its developer) (Brown, 2019). Granulator 2 is a free Max4Live synthesiser available in Ableton Live. Native Instrument's Kontakt offers some granulation features, like Grain Size and Smoothing, in addition to its sampling potential (Price, 2005). There are some third-party granular effect plugins, like Portal, Granulizer or Polygon, which modulate the incoming signal from audio or virtual instrument tracks. Despite having slightly different functions and capabilities, all of these devices are based on the same principle – producing grains. They work with pre-recorded samples and offer the option to manipulate, warp and stretch them. Granite and The Mangle offer the ability to control the *grain shape*, which is the envelope shape for each grain. Quanta brings other ideas forward, like a randomization knob for every parameter and a routing matrix to assign parameters to envelopes and LFOs. These are all appealing ideas which I could explore with my instrument in the future.

I believe what makes my instrument unique is the added sampler functionality and the ability to switch modes from audio slicing to audio transposition. Even at its demo stage, it translates well to MIDI and greatly speeds up the workflow. With further development, like transient-based slicing or polyphonic transposition, it could truly become an appealing marketing angle. Blending the worlds between synthesis and sampling is an idea I would like to push forward with my instrument.

One important thing, I would pay a lot of attention to, would be the visual side of the instrument. Most granular tools out there can appear quite daunting at first sight. Making sure the interface is easy to navigate and user-friendly could go a long way and encourage less-advanced producers to dive into the world of granulation. This is something the people at Output have

done extremely well with the release of Portal. Its GUI was deemed “stunning, clean and beautifully animated”, which inevitably led to the plugin’s quick rise in popularity. Having an inviting interface makes the users want to spend time with the instrument and explore its capabilities (Music Radar, 2019).

Conclusion

In conclusion, having conducted extensive research, I managed to create my own, original software instrument, based on the existing concept of granular synthesis. The process required critical thinking and evaluation to address the unforeseen issues and obstacles. There are definitely still issues to address and improvements to implement, but despite its imperfections, the instrument shows a lot of potential for future development. It brings forward some unique ideas and features, which I could not find in any existing granular synthesiser on the market.

This project has opened me up to the vast world of software instrument design. It exposed me to the basic principles and processes behind the creation of plugins and helped me identify the areas which I need to improve on. Although the possibilities are nearly limitless, they are very much grounded in technical skill and knowledge. Knowing how much more there is to explore, I am excited learn more about the topic and possibly develop my instrument more in the future.

Bibliography

Journals

Eckel, G., Rocha, M. and Becker, B., 1995. The Development of GiST, a Granular Synthesis Toolkit Based on an Extension of the FOF Generator.

Roads, C., 1988. Introduction to Granular Synthesis. Computer Music Journal. Cambridge: MIT Press.

Roads, C., 1991. Asynchronous Granular Synthesis of Sound. Representations of Musical Signals. Cambridge: MIT Press.

Roads, C., 1996. The Computer Music Tutorial. Cambridge: MIT Press.

Roads, C., 2001. Microsound. Cambridge: MIT Press.

Books

Xenakis, I., 1992. Formalized Music: Thought and Mathematics In Composition. 6th ed. New York: Pendragon Press.

Manuals and tutorials

Cycling '74. 2020. Max 8: edge~. [manual] [Accessed 21 March 2020].

Cycling '74. 2020. Max 8: groove~. [manual] [Accessed 21 March 2020].

Cycling '74. 2020. Max 8: mtof. [manual] [Accessed 21 March 2020].

Cycling '74. 2020. Max 8: metro. [manual] [Accessed 21 March 2020].

Cycling '74. 2020. Max 8: notein. [manual] [Accessed 21 March 2020].

Cycling '74. 2020. Polyphony Tutorial 1: Using the poly~ object. [online] Available at: https://docs.cycling74.com/max7/tutorials/11_polychapter01 [Accessed 23 March 2020].

Cycling '74. 2020. Polyphony Tutorial 2: Granular Synthesis. [online] Available at: https://docs.cycling74.com/max8/tutorials/11_polychapter02 [Accessed 23 March 2020].

Cycling '74. 2020. Max 8: poly~. [manual] [Accessed 21 March 2020].

Cycling '74. 2020. Max 8: thispoly~. [manual] [Accessed 21 March 2020].

Cycling '74. 2020. Max 8: trapezoid~. [manual] [Accessed 21 March 2020].

Videos

Batchelor, P., 2018. Plode - Build A Granulator In Max 7. [video] Available at: <https://www.youtube.com/watch?v=yfQfJQoGES4&t=184s> [Accessed 21 March 2020].

Dehaan, D., 2020. Max For Live: Building A Polyphonic FM Synth. [video] Available at: <https://www.youtube.com/watch?v=C9yj-J3V21o> [Accessed 23 March 2020].

Jackson, H., 2009. Max/MSP Tutorial 1 - The Very Basics. [video] Available at: <https://www.youtube.com/watch?v=nDBbxRvVwxw> [Accessed 20 March 2020].

Suzuki, C., 2017. Granular Synthesis (Prototype In Max MSP). [video] Available at: <https://www.youtube.com/watch?v=Cv9-TzzGAh8> [Accessed 26 March 2020].

Online

Brown, G., 2019. The Basics Of Granular Synthesis. [online] iZotope. Available at: <https://www.izotope.com/en/learn/the-basics-of-granular-synthesis.html> [Accessed 29 March 2020].

Davies, H., 2019. Three Takeaways from Flume's Returning Mixtape. [online] PILERATS. Available at: <http://pilerats.com/music/electronic/flume-hi-this-is-flume-mixtape/> [Accessed 4 April 2020].

Music Radar. 2019. Output Portal - Review. [online] Available at:

<<https://www.musicradar.com/reviews/output-portal>> [Accessed 5 April 2020].

New York University. n.d. Frequencies of Notes. [online] Available at:

<<https://cs.nyu.edu/courses/fall03/V22.0201-003/notes.htm>> [Accessed 22 March 2020].

Price, S., 2005. Granular Synthesis. [online] Sound On Sound. Available at:

<<https://www.soundonsound.com/techniques/granular-synthesis>> [Accessed 1 April 2020].

Reid, G., 2001. From Polyphony to Digital Synths. [online] Sound On Sound. Available at:

<<https://www.soundonsound.com/techniques/polyphony-digital-synths>> [Accessed 2 April 2020].