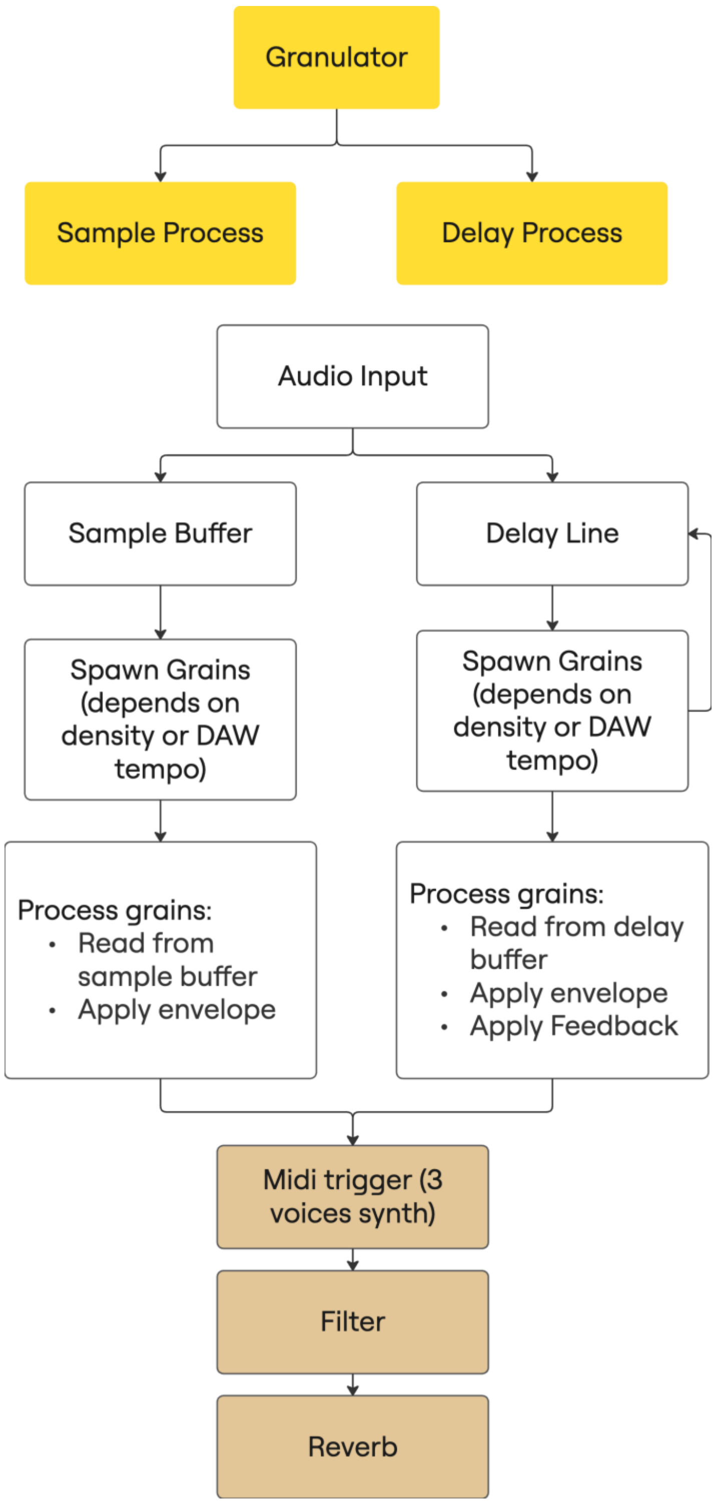
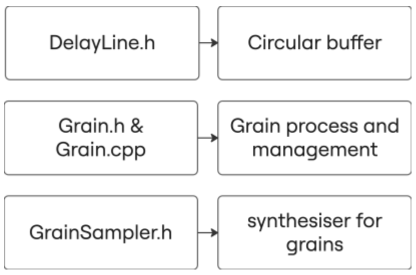
Hybrid Granulator

Shreya Gupta





The flowchart illustrates the signal and control path of the hybrid granulator, detailing how audio input is routed through either a sample buffer or a tapped delay line, leading to grain spawning, processing and final output through MIDI triggered polyphonic synthesiser with filtering and reverb effects.

Introduction:

Granular synthesis is a powerful method of sound design by manipulating audio at a micro level. This allows the user to create rich and evolving textures. This project explored the creative and technical possibilities of granular synthesis through the development of a hybrid granulator plugin – Tapped Delay Line Granular Synthesis and Stored Sample based granular synthesis. The aim was to design a versatile plugin capable of generating both rhythmic and ambient soundscapes.

Influences:

In this project, both Tapped Delay Line Granular Synthesis and Stored Sample Granular Synthesis are directly inspired by Ross Bencina's paper Implementing Real-Time Granular Synthesis (Bencina, 2001). This foundational interest was further deepened through the study of the Microcosm pedal by Hologram Electronics, whose presets—particularly Mosaic and Haze—appear to exhibit characteristics of delay-line-based granulation. Building on this concept, the project was initially conceived as a looper or granular sampler. However, it later evolved into an expressive grain synthesizer, significantly influenced by Robert Henke's Granulator II for Ableton Live, which demonstrated the creative potential of polyphonic, pitch-based granular synthesis.

From a coding and structural perspective, the architecture for grain management was inspired by ChapaChapa's Freeze Granulator on GitHub. Studying this implementation of grain generation and control provided a strong foundation for developing dynamic grain parameters, including envelope type, spatial spread, playback rate, and jitter. The final outcome is a hybrid granulator that performs traditional granular synthesis while introducing a novel feedback mechanism—routing grains back into the delay line—to produce evolving and abstract sonic textures.

Coding highlights:

The plugin adopts a modular design approach where each grain is treated as an independent entity with a well-defined parameter set. These parameters include:

- Onset: When the grain begins (in samples, relative to process time)
- Length: Duration of the grain in milliseconds
- Rate: Playback speed (Middle C = normal; above = faster, below = slower)
- Level: Amplitude multiplier (0.0-1.0)
- Position: Start point in the source or delay buffer (0.0-1.0)

This has been implemented via a Grain class (defined in Grain.h and Grain.cpp) which allows for each grain to be instantiated, processed, and erased independently, offering fine-grained control over behaviour.

Grains are then managed by a custom sub class of juce::SynthesiserVoice, enabling polyphony. The grain scheduler is implemented in renderNextBlock() method, where grains are spawned based on density. The density can be quantised to the host DAW's tempo when synchronisation is enabled. MIDI integration allows for pitch mapping through startNote(). To ensure thread safety, all parameters use std::atomic<float> and to interpolate changes over time, JUCE’s SmoothedValue class is used, preventing audio artifacts to abrupt parameter transitions.

To introduce greater variety in textures and sounds, parameters such as grain length, and position include controlled randomisation/jitter. This allows users to explore a sonic playground of evolving and unpredictable textures. Additionally, global effects such as filtering ((low-pass, high-pass, or band-pass) and reverb (implemented using JUCE’s built-in DSP modules) help to shape and unify the overall output.

Context of Use:

Right now, the delay line has been intended to be used like a looper pedal (which is triggered on the MIDI by holding onto one note). Delay line seems to be especially suited to live performance contexts, where the granulated output dynamically follows the real-time input. The stored sample mode is more flexible, allowing for both live performance and composition use cases, particularly for building rich, polyphonic, choral textures.

Significant Challenges:

A significant challenge was scheduling and rendering grains under high density or polyphonic conditions. Initially, grain overlaps caused CPU spikes. Each grain is assigned an onset (start time) and a length (duration) and it’s existence is checked against the global time reference during each render cycle. Grains whose age exceeds their lifespan (onset + length < time) are erased, ensuring the memory is freed in real time. Additionally, the number of voices are kept to 3 to limit the CPU spikes. For users working on high-performance systems, the voice count can be increased, enabling denser and more immersive soundscapes.

Further explorations:

Given the current synthesiser-based architecture, the code is prepared for future enhancements. Potential extensions include per-grain LFO modulation, tremolo effects via pitch wheel and pitch glides triggered on note events. Presently, the input stream for both the delay and sample processes is sourced from an audio file within the plugin’s resources folder. A key objective for future development is to implement support for real-time audio input, enabling the plugin to function as a live processing tool—particularly in the tapped delay line mode, similar to the behaviour of performance-oriented pedals.

Conclusion:

This project represents an exploration of granular synthesis through the design and implementation of a hybrid granulator plugin. Drawing on a diverse range of inspiration – from Bencina’s foundational work to contemporary tools like Microcosm and Granulator II – this project merges the conceptual strengths of both tapped delay line and stored sample synthesis. The resulting plugin offers a flexible sonic palette capable of producing everything from rhythmic micro-loops to lush ambient textures. The integration of feedback in the delay line introduces a dynamic layer of sonic evaluation, allowing users to sculpt evolving textures. Ultimately, it contributes to the ongoing evolution of granular synthesis as both a compositional and performative tool, blurring the boundaries between instrument, effect, and texture generator.

Caveats in plugin:

- When the feedback is more, avoid polyphony for managing CPU power in the case of Delay Process.ˆ
- There is a choice in the resources folder for other samples, you can change the sample in the loadfromMemory function.
- I have summed the grained output and passed it into the delay line again. I am a little unsure if I have done it correct and the result is as expected.

Audio Files made with Clair de Lune:

1. Feedback: based on the feedback and grain parameter
2. Groove: based on quantiser
3. Mosaic: based on density and length manipulation
4. MIDI Arp: based on MIDI arpeggiator going through granulator
5. Rumble: based on filters and sparse
6. Stutter: based on sparse and grain length jitter

Presets:

1. Spiral
2. Mosaic
3. Ant Parade
4. Textural (Add MIDI Arp)
5. Chops
6. Rumble
7. Refraction

References:

Bencina, R. (2001). Implementing Real-Time Granular Synthesis. http://www.rossbencina.com/static/code/granular-synthesis/BencinaAudioAnecdotes310801.pdf

Henke, R. (2012). Granulator II [Mac for live Device]. Ableton. https://www.ableton.com/en/packs/granulator-ii/

ChapaChapa.(2020). Freeze Granulator[JUCE Github Repository]. Retrieved from: https://github.com/ChapaChapa0/FreezeGranulator?tab=readme-ov-file

Holograom Electronics. (2020). Microcosm Pedal. Retrieved from: https://www.hologramelectronics.com/products/microcosm

References for samples in Resources folder:

- The ClairDeLune piano was grabbed from royalty free: https://motionarray.com/royalty-free-music/claude-debussy-clair-de-lune-945966/?utm_source=google&utm_medium=cpc&utm_campaign=14398494849&utm_content=149237462574&utm_term&keyword&ad=663242597649&matchtype&device=c&gad_source=1&gclid=Cj0KCQjwiLLABhCEARIsAJYS6um8Xco2Wtts7pZOzkwaAOpqOGOoH9e-2LiwR5H1yEUUWtSkOcz-fVwaAloDEALw_wcB
- PRAKSHI 2.wav – a self-produced track (unreleased)
- Beat_2.wav – self produced beat
- Beat.wav – self produced beat

Label and Parameter with their functions:

Label	Parameter	Function
Granular Mode	mode	Chooses Delay-based vs Sample-based grain source.
Grain Envelope	envelope	Selects grain envelope shape (e.g., Triangle, Hann, Trapezoid, etc).
Grain Length	grainLength	Controls the duration (in ms) of each grain.
Grain Length Jitter	grainLengthJitter	Adds randomness to grain length (0 = fixed, 1 = full variability).
Density	density or quantised	Controls time interval between grain spawns (lower = more frequent grains).
Activity	activity	Number of simultaneous active grains per voice (influences layering).
Grain Level	level	Base amplitude of grains.
Probability	probability	Chance (0–1) that a grain will spawn on trigger.
Position/Tap	position	For sample mode: position in waveform; for delay: tap offset.
Sparse	sparse	Randomness to the tap position or delay offset position.
Playback	playbackMode	Controls playback direction (Forward, Reverse, Random).
Mix	mix	Crossfade between dry (original) and wet (granulated) signal.
Stereo Width	stereoWidth	Spreads grains across stereo field (0 = mono, 1 = full width).
Quantise	quantiseParam	Enables syncing grain density to BPM from DAW.
Quantise Division	quantiseDivision	Chooses rhythmic grain interval (e.g., 1/4 = quarter note, 1/8 = eighth).
Filter Type	filterType	Selects post-grain filter: lowpass, highpass, bandpass, etc.
Filter Cutoff	filterCutoff	Sets cutoff frequency of post-grain filter.
Filter Resonance	filterResonance	Sets Q/resonance of the filter peak.
Reverb On	reverbToggle	Enables/disables post-grain reverb effect.
Reverb Mix	reverbMix	Wet/dry balance of reverb applied to grain output.
Feedback Amt	feedbackParam	Controls how much of the delay line’s output feeds back into itself. (only in Delay process)
Grain Feedback	grainFeedbackParam	Controls how much grain audio feeds into the delay buffer each sample. (only in Delay process)