**Nonlinear Quasistatic Quintet**

Prompt 3: Subtractive Synthesizer

Graphical user interface, website

Description automatically generated

Group members: Marco Cristofalo, Marco Berasconi, Francesco Barraco, Bram Steenbergen, Geoffrey Horn

11/05/2022

Department of Music and Acoustic Engineering

Politecnico di Milano

The group known by the moniker of Nonlinear Quasistatic Quintet (NQQ) has been tasked with the creation of a subtractive synthesizer utilizing the Juce framework. Given the lengthy history of synthesizers and how ubiquitous the synthesizer is in modern music, NQQ got to work researching the functionality of the modern synthesizer and what could make the project unique.

The term “synthesizer” is used to describe any program or physical machine that generates sound by producing a variety of signals. In the modern musical world, synthesizers are used to produce nearly all modern music, especially in the pop and electronic music genres. Unfortunately, their ubiquity has also let to the common misconception that a “synth” is a keyboard in which you press a key, and a note is magically generated. While to the untrained eye this may seems to be the case, there is heaps taking place behind the scenes within the machine to get from the user interface to the final tone that the user hears.

As for defining “subtractive synthesis”, as is directed in the assignment, the distinction has to do with the methods used to filter the original tone as it works its way to becoming the output signal. Every synthesizer is composed of four main parts: a signal generator, filters, low-frequency oscillators (LFOs), and an ADSR envelope (which is an acronym for attack, sustain, decay, and release). In the creation of a subtractive synthesizer, the initial signal contains a substantial amount of information which is then filtered down as the signal goes through processing. Typically, this initial signal is a raw square or saw wave, as they are easy to generate with an oscillator. However, a signal such as this produces only a tone, and is in no way considered “musical”, and requires processing to get the wave shape to resemble a musical signal.

Additionally, many synthesizers contain more than one input waveform, allowing the user to create a unique input sound and giving the user more freedom in the synthesizer process. These signals are combined before being sent to the processing portion of the synthesizer, which adds complexity to the waveform before processing even begins.

Next, the filter portion of the synthesizer performs the first alterations to the input signal. The “subtractive” part of a subtractive synthesizer begins at this stage. Three main types of filters exist for an application such as this one, and their names effectively describe the actions that they perform; low-pass, high-pass, and band-pass filters allow all specified frequencies of the initial signal to pass through to the next stage in the process while cutting the frequencies which do not meet the requirements of the filter. For a high-pass filter, this is everything above a certain frequency threshold, whereas a low-pass filter allows all frequencies to pass below the given threshold. Band pass cuts everything except for a “band” around the specified center frequency.

The next step in the process is to amplify the signal from the filters using an envelope generator. As the signal has just been heavily modified, it is then required to modulate the amplitude of the wave to make it sound like a musical note, and to do this there are four parameters that craft the shape of the sound wave. This is known as the ADSR, and it is made of the attack, decay, sustain, and release portions of the signal. The first part, the attack, dictates how the initial moments of the sound is perceived. It is the onset of the note, and it controls how long the sound takes to achieve its maximum intensity. The decay follows up the attack, and it controls how long it takes the signal to come down to the level that the note will be played at. Third is the sustain, which controls the level that the sound is played at as it is held. The release then dictates how quickly it fades to silence once the note is released. With these four parameters, an effective envelope of the overall note is formed. Each level in the process can be tweaked and gives the user endless flexibility to fine tune the way that a note behaves.

Another element that is common in conjunction with the ADSR is a low-frequency oscillator, also known as an LFO. An LFO, just like the original oscillator signal, generates a wave of a specified shape. However, the LFO only generates very low frequency tones, which can be applied to the various aspects of the synth to modulate the final sound. Typically, an LFO is utilized to create effects as opposed to changing the envelope of the sound.

Once all these elements are combined, a true “note” is formed. Small tweaks to any part of the process can have major ramifications to the overall sound of the note. Therefore, it is important when creating a subtractive synthesizer to ensure that the user interface is intuitive, and enough parameters are included to allow the user to express their creativity and dial in the exact sound that the user is attempting to create.

With research now complete, the next step was to begin crafting a new subtractive synthesizer within Juce. As many of the group members of the Nonlinear Quasistatic Quintet play guitar and have a deep appreciation for the aesthetic and music of the 1960’s and 1970’s. Therefore, this synthesizer will be tuned towards that overall sound, which is a unique direction to take a synthesizer in 2022. However, this is what makes the project fun.

As discussed previously, the first part of the synthesizer is to create an oscillator. Included in Juce are a variety of built-in oscillators, which were used as the framework of the first iteration. However, it was quickly discovered that these built-in oscillators had major limitations that restricted the ability to control the next aspects of the synthesizer. The built-in synthesizers were scrapped, and construction was to begin on a custom oscillator format.

The NQQ synth utilizes two oscillators, as with many synthesizers. However, the creation of a new oscillator was necessary given the uniqueness of the architecture. Instead of having n-predefined waves to choose from, the oscillators can have their shape changed dynamically. The first oscillator changes from a sine wave to a triangle wave; the other one from a sawtooth wave to a square wave. This behavior is achieved by defining the functions which constitute the actual oscillator as a mathematical series of functions which depend on one parameter. By changing a single parameter, a morphing effect on the waves is achieved. This design was inspired by the oscillators you can find in a Buchla modular synthesizer.

Taking the more difficult route of creating a new oscillator is no easy task and consumed a majority of the programming labor hours. The main difficulty in the creation of the new oscillators was the polyphony aspect. As the oscillators are hand-made, a new polyphony system was required to be created. This system consists of two main pieces of data for each oscillator: which MIDI note is being played, and how much time that note is being played for. For example, when a note is being played on a keyboard, the program looks for searches for a free oscillator which is currently not playing anything, or it finds an oscillator which is already playing the specified note. If a free oscillator is discovered, it will be assigned a frequency and a note number to use as an identifier. Simultaneously, an array is referenced that specifies how much many buffers have passed since that specific oscillator was last triggered. This array is increased each time that a new buffer is computed. If a new note is played but there is no free oscillator, the program finds oscillator has been playing the longest and reassigns it the new oscillator value for the new note. Its age is therefore reset. When a NoteOff message is received, the program scans for the oscillator which has the same frequency as specified in the midiMessage. At this moment, the release stage of the ADSR is triggered. When the ADSR reaches zero, the oscillator is no longer considered in the computations until it is triggered once again.

Having completed the oscillator portion of the programming, the next logical step was to begin programming the filter. Fortunately, this step required significantly less customization than the creation of the oscillators. The filter is a State Variable filter found in the digital signal processor (DSP) module. This filter already contains the three previously discussed filter types of low-pass, high-pass, and band-pass. It has a slope of 12dB/octave, which is standard for a filter in these applications.

Additionally, the pre-made Juce ADSR worked seamlessly when implemented within the NQQ synthesizer. Having no need to build the ADSR saved the team a substantial amount of programming time. Therefore, two ADSRs were in fact implemented. The first ADSR controls the amplitude of the output for each voice, allowing the user to craft the exact shape of the voice envelope. The second ADSR dynamically controls the cutoff frequency of the filter. This type of customizability gives the user the ability to create a “wah-wah” effect, typically common in the psychedelic rock music of the 1960’s and 1970’s.

To double down on the psychedelic flavor of the synthesizer, a phasor and LFO were also added to the program. The purpose of a phasor is to create a “sweeping” effect, which was popularized in classic rock. It can be applied to a human voice to achieve a robotic tone, or it is more commonly used to modify a guitar signal to make the note feel expansive.

The LFO is implemented to allow the user to modulate the signal, giving the note a pulsating effect. This is accomplished in two different manners. The first is controlling the pitch of the note to contain a vibrato effect. The second effect controls the morph component of the two waves, which creates the effect of movement of the sound as the note is sustained. Regardless of which manner the user chooses to implement the LFO, the sounds achieved are essential to creating a sound unique to the user. After putting all the previously listed elements into one functioning synthesizer, the result is quite unique.

Initially, it was suspected that this project would not be as programming heavy as the first project in Supercollider, as Juce has built-in elements for a majority of the parts of a subtractive synthesizer. However, when it was realized that more customization was required to achieve an optimal synthesizer and the oscillator would need to be created from scratched, this changed the group’s approach. In Juce, the Synth class can manage different voices and polyphony. However, due to the group’s ambitions, it became easier to create unique synths as opposed to altering the available synths classes. Due to the group’s limited knowledge of object-oriented programming, a modular approach to the program took place in which the different members of the group crafted the part of the code that was most familiar to them. With certain elements, such as the LFO and the filters, the group was comfortable integrating these parts into the homemade elements of the code, and therefore did not need to start from nothing with these elements. However, of course, the group needed to be conscious of time and capability limitations, especially with a hard deadline. Adding the LFO and filter were ambitious but were deemed necessary to tailor the synthesizer to the psychedelic sound that was the goal all along.

The user interface of the program gives a beautiful fascia to the brilliant brains of the Nonlinear Quasistatic Quintet’s subtractive synthesizer. Building upon the theme set in the first Supercollider project’s GUI, the trippy 1970’s color wheel design has been substituted for a more simplistic, muted color scheme. Still continuing with the same color scheme, it was ultimately decided that due to the sheer amount of buttons and dials and a logical signal flow needing to be conveyed, simplicity trumped flamboyance.

The GUI is divided into 6 segments, each commanding a step of the process as discussed above. Three segments are in the top half, while the other three are located in the bottom half of the window. Starting in the top left is the first setting for the user, which is the selection of the shape of the first oscillator. The shape of this oscillator is controlled by the knob labelled “triangleness” and is attenuated by its volume knob. This is followed by the second section, containing the “squaredness” knob and its volume knob. The third section contains the ADSR, along with the buttons allowing the user to decide whether the ADSR controls the Amp, Filter, or LFO. Following this is the filter selection panel, in which the user is prompted to select low-pass, high-pass, or band-pass, while also allowing the user to modify the mod depth, cutoff, and resonance for the selected filter. The fifth section contains knobs for the speed and mix of the phasor, which then ultimately leads to the output master volume and oscilloscope in section six. Ultimately, this setup is intuitive to use, and is simple to visualize the output thanks to the oscilloscope.

While the team is more than happy with the result of the program, there is certainly plenty that could be improved upon. To meet deadlines, the different aspects of the code were created by various members of the team, all of which have unique coding backgrounds and coding styles. The various pieces were then patched together, creating an effective result, but certainly not a seamless one. Additionally, the limited knowledge of coding made it so the code is certainly not as efficient as it could be were it written by a team of professional software engineers. Additionally, if the process were to be done again, implementing the pre-built synths and oscillators could save a lot of time during the development stage.

While the creation of unique oscillators allowed the code to have lots of customization, there were certainly more wave shapes that could have been experimented with to achieve optimal input. Due to the functions that were used for the programming of the wave shape, and the mode at which the oscillators fade between different wave shapes, some of the shape is muddled. For example, being able to select the “triangleness” of the wave certainly makes the sound extra-customizable. However, it also makes it difficult to ensure that a triangle wave set to 100 percent “triangleness” achieves a pure triangle wave every time, and the same goes for the square wave setting. Perhaps giving the user less flexibility in this case would lead to a more polished final product if the synthesizer were to go to market. Ultimately, it would be for the user to decide what kind of sound most appeals to their musical style.

Additionally, if given more time, even more effects could have been introduced for the user to tinker with. Some examples are delay, reverb, and chorus. However, as these were far outside of the scope of the current subtractive synthesizer project description, they were all wish list items and were deemed unnecessary for the sake of time.

Ultimately, despite perhaps doing more than was necessary, NQQ has produced a subtractive synthesizer to be proud of. The final product is a balanced mix of an abundance of tools for full musical freedom while not making the program too confusing to use efficiently. Utilizing a healthy mix of built-in Juce tools and hand-made elements, the subtractive synthesizer is perfect for the experienced music producer looking for their next new retro sound, and approachable enough for a rookie musician to unlock a variety of new sounds.