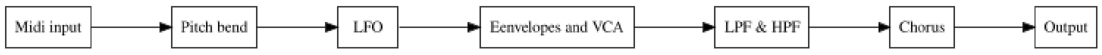
Homework 2

Analysis of ATLAS-06-Synthesizer

Group12

1 Introduction

This report analyzes and explains the implementation of a synthesizer plug-in called ATLAS-06-Synthesizer. Below is the block diagram of the audio chain of the plug-in. In the following sections, each module will be discussed one by one.



2 Modules

2.1 LFO

2.1.1 Working Principle

A low-frequency oscillator (LFO) is an electronic signal generator that produces waveforms at frequencies below the audible range (usually less than 20 Hz). The main principle behind LFOs is to create a repeating waveform that can be used to modulate other audio signals, adding a sense of movement and animation to the sound.

The waveform generated by an LFO can be of different types, such as sine, triangle, square, sawtooth, or random. Each waveform has its own unique character and can be used to achieve different effects. For example, a sine wave produces a smooth and gentle modulation, while a square wave produces a more abrupt and choppy modulation.

2.1.2 Implementation

The constructor takes a reference to a ***'SynthFrameworkAudioProcessor'*** object as an argument, which is the main processor object of the plugin.

SynthFrameworkAudioProcessor is a class that represents the main processor in a virtual synthesizer built using the JUCE framework. The ***'SynthFrameworkAudioProcessor'*** class is responsible for processing the audio data and generating the sound output. It also manages the parameters and state of the synthesizer. This class typically interacts with other classes and components of the synthesizer, such as the filter, envelope, LFO, and GUI components, to implement the desired behaviour of the synthesizer.

The '***LFO'*** class has several member variables, including ***'Slider'*** objects for controlling the LFO rate and delay parameters, and ***'Label'*** objects for displaying numerical values next to the sliders. There are also ***'AudioProcessorValueTreeState::SliderAttachment'*** objects for linking the sliders to corresponding parameters in the plugin's '***AudioProcessorValueTreeState'*** object.

2.1.3 GUI

- ***'lfoRate'*** controls the frequency or rate of the LFO waveform, which determines how fast it oscillates or modulates the audio signal. In this case, the ***'lfoRateSlider'*** is the graphical slider that allows the user to adjust the LFO rate in the range of 0 to 10 Hz.

- ***'lfoDelay'*** controls the amount of time before the LFO modulation starts, which can be used to create a gradual or phased-in effect. In this case, the ***'lfoDelaySlider'*** is the graphical slider that allows the user to adjust the delay time in the range of 0.1 to 50000 milliseconds.

2.2 DCO

2.2.1 Working principle:

DCO (Digitally Controlled Oscillator) is a type of oscillator that uses digital circuitry to control the frequency and waveform of the oscillator. They were designed to overcome the tuning stability limitations of early VCO (Voltage Controlled Oscillator) designs. VCO typically consists of a core oscillator circuit that generates a waveform, such as a sine wave, triangle wave, sawtooth wave, or square wave. The frequency of the waveform is determined by a control voltage in the oscillator circuit.

Compare to VCO, DCO produces more stable tuning and reliability compared to VCO and the frequency of a DCO is controlled by the digital circuit.

2.2.2 Implementation

Most of the algorithms used in this project are from an audio library called Maximilian.

```
double dcoSound = getSawOsc() + getSquareOsc() + getSubOsc() +  noiseValue;
```

- noiseValue: use rand() to generate a number between 0 to 1 and map it between -1 to 1, and multiply it by the value read by the noise slider.
- getSawOsc() and getSquareOsc() return the Sawtooth generator and square wave generator from the external library.

2.2.3 GUI

- 4' 8' 16' These buttons are used in the function getPitchRangeSetting(), which will divide the notes' pitch frequency by 1, 2, and 4 respectively.
- LFO changes the amount of the modulation, while the LFO in the LFO block changes its rate.

- PWM controls the amount of Pulse Width Modulation (PWM) applied to the DCO's pulse wave. By modulating the pulse width, a wide range of timbres and effects can be created.
- The next two buttons are the switches of the square and saw oscillator.
- SUB adds a square oscillator with a pitch 2 octaves lower.
- Noise adds white noise.

2.3 ENV

2.3.1 Working Principle

An envelope (ENV) is a module that shapes the volume of a sound over time. An envelope generator produces a voltage that changes over time, usually in response to the pressing of a key on a keyboard, a trigger signal, or a gate signal. The four main stages of an envelope are Attack, Decay, Sustain, and Release, commonly referred to as ADSR.

- Attack: the time taken for the volume to reach its maximum level after the key is pressed or the gate signal is received.
- Decay: the time taken for the volume to drop from its maximum level to the level set by the Sustain parameter.
- Sustain: the level at which the volume remains as long as the key or gate signal is held down.
- Release: the time taken for the volume to return to zero after the key is released or the gate signal ends.

2.3.2 Implementation

The ‘ENV’ class has four sliders and labels for setting the attack, decay, sustain, and release parameters of the envelope. The constructor takes a reference to a ‘SynthFrameworkAudioProcessor’ object and initializes the sliders with default values, ranges, and listener callbacks. The ‘AudioProcessorValueTreeState::SliderAttachment’ class is used to link the values of the sliders to a value tree in the processor object.

2.3.3 GUI

- ***‘AttackSlider’*** is the graphical slider that allows the user to adjust the attack value in the range of 0.1 to 5000 milliseconds.
- ***‘DecaySlider’*** is a graphical slider that allows the user to adjust the decay value in the range of 1 to 2000 milliseconds.
- ***‘SustainSlider’*** is the graphical slider that allows the user to adjust the sustain value in the range of 0 to 1.
- ***‘ReleaseSlider’*** is the graphical slider that allows the user to adjust the release value in the range of 0.1 to 5000 milliseconds.

2.4 HPF

2.4.1 Working Principle

A high-pass filter (HPF) is a type of electronic filter that allows higher frequency signals to pass through it while attenuating lower frequency signals. The HPF works by allowing signals above a certain cutoff frequency to pass through while attenuating signals below this frequency. The cutoff frequency is the frequency at which the filter starts to attenuate the signal, and it is typically measured in Hertz (Hz).

In practical applications, HPFs are used to remove low-frequency noise from an audio signal or to selectively remove bass frequencies from a sound source. HPFs are often used in combination with low-pass filters to create a bandpass filter that allows a specific range of frequencies to pass through.

2.4.2 Implementation

The ‘HPF’ class has one slider and several labels for setting the cutoff frequency. The constructor takes a reference to a ‘SynthFrameworkAudioProcessor’ object and initializes the sliders with default values, ranges, and listener callbacks. The ‘AudioProcessorValueTreeState::SliderAttachment’ class is used to link the values of the sliders to a value tree in the processor object.

2.4.3 GUI

- ***‘HpfSlider’*** is the graphical slider that allows the user to adjust the cutoff frequency in the range of 10 to 6000 Hz.

2.5 VCA

2.5.1 Working Principle

An amplifier (AMP) is an electronic device that is usually used to achieve amplification, which is the process of increasing the amplitude or strength of a signal. Opposite to amplification, attenuation is the process of reducing the amplitude or strength of a signal. The most common type of amplifier used in audio processing is the voltage-controlled amplifier (VCA).

The VCA gain is the amount of amplification or attenuation applied to an audio signal passing through a VCA. The gain of a VCA can typically range from 0 (fully attenuated) to 1 (fully amplified).

2.5.2 Implementation

The ***‘VCA’*** class constructor initializes includes a slider to control the VCA gain, a label to display the slider value, and another slider to toggle between two modes of operation (amplification and attenuation). The values of these components are linked to the application's ‘***AudioProcessorValueTreeState’***, which manages the state of the synth and communicates with the host application.

The ***‘sliderValueChanged method’*** is called when the ‘***ampModeSlider*** ‘ is modified, and ensures that its value is either 0 or 1, corresponding to the two available modes of operation.

2.5.3 GUI

- ***‘vcaSlider’*** is the graphical slider that allows the user to adjust to control the VCA gain.
- ***‘ampModeSlider’*** is the graphical slider that allows the user to toggle between two modes of operation (amplification and attenuation).

2.6 VCF

2.6.1 Working Principle

VCF stands for "Voltage-Controlled Filter". VCF is an audio signal processor typically used for changing timbre. It can manipulate the cutoff frequency and harmonic content of a filter based on a control voltage. VCFs are commonly used in synthesizers to adjust the richness and colour of timbre, creating a variety of sound effects.

The basic principle of a VCF is to adjust its cutoff frequency based on a control voltage, which can come from various sources such as the keyboard or other controllers of a synthesizer. By adjusting the cutoff frequency, a VCF can allow a certain range of audio signals to pass through while filtering out signals at other frequencies. Additionally, a VCF can adjust the bandwidth and resonance of the filter to create different sound effects.

2.6.2 Implementation

The std::unique_ptr<AudioProcessorValueTreeState::SliderAttachment> variable declaration creates 2 new SliderAttachment objects.

For Cut-off Frequency, the range of the resonanceVal will be between 30 - 4000Hz, and 4000Hz is the default value.

For the Resonance, the range of resonanceVal will be between 1-20, and 1 is the default value.

2.6.3 GUI

The SliderAttachment object created in the function is assigned to the filterVal pointer for Cutoff-Frequency and resonanceVal pointer for Resonance, which represents a slider GUI component controlling the resonance parameter of the VCF. The GUI element is created in another part of the code and passed as a reference to the SliderAttachment object.

2.7 Chorus

2.7.1 Working Principle

Chorus effect is an audio effect that enriches a sound by creating a time-varying delay effect. It works by adding a slightly delayed and modulated copy of the original audio signal to the original signal.

This part implements a chorus effect which is divided into 2 classes - Chorus and ChorusEffect.The Chorus class is responsible for the user interface and interaction with the ChorusEffect class. The ChorusEffect class implements the core functionality of the chorus effect.

2.7.2 Implementation

In the chorusEffect.cpp file, the main effect is based on delayed samples that create the time-varying delay effect of a chorus. The delay time is modulated by an LFO waveform generated using the sinebuf4 method of the LFO object, which has a frequency of the lfoRate. The delayed sample is then combined with the original sample and returned as the output of the chorus effect.

delayedSample = delay.dl(sample, 1 + 175*(1 + lfo.sinebuf4(lfoRate)), 0);

return sample + delayedSample;

In the chorus.cpp file, the user interface is implemented with a toggle button to turn the chorus effect on and off, and other UI elements. The ***AudioProcessorValueTreeState::ButtonAttachment*** class is used to bind the toggle button to the ***AudioProcessor***, enabling the relevant functions to be called automatically when the button is clicked.

2.7.3 GUI

- ***'chorusButton'*** contains 2 buttons which can control the chorus effect on and off. When the buttons clicked, the rates of the chorus effect changed.