|  |  |  |  |
| --- | --- | --- | --- |
|  | secs MAX | WPM=140 | num words |
| **Intro** | 40 |  | 90 |
| **Audio** | 30 |  | 70 |
| **Modular parts + LIT** | 60 |  | 140 |
| Overall system | 50 |  | 90 |
| LUTs and lerp | 50 |  | 120 |
| Wavetable | 50 |  | 120 |
| Filters | 50 |  | 120 |
| Waveshaping | 60 |  | 140 |
| ADSR | 40 |  | 90 |
| Generator | 50 |  | 120 |
| GM | 40 |  | 90 |
| **Overall tests** | 40 |  | 90 |
| **Conclusion** | 40 |  | 90 |
|  | 9.833333 | mins | 1370 |

**Intro**

Synthesisers play an important role in all music today since the inception of analogue sound synthesisers in 1928. A synthesiser refers to a system that generates audio signals.

The aim of this project is to design and implement wavetable-based audio generation software, targeted for microcontrollers. The system can play up to a fixed number of notes, each with an arbitrary frequency, by using on/off note triggers.

It produces high-quality stereo audio, while taking computational speed and memory consumption into account, and implements all the basic synthesis features: volume modulation; filtering and frequency cut-off modulation; ADSR envelope control signals; FM; waveshaping.

This thesis can form the basis for a complex commercial product.

**Audio**

I will now play 5 different sections from Beethoven’s Moonlight Sonata, generated by the designed system from a MIDI file, each with different configuration parameters.

**Modular parts + LIT**

Modular synthesis is a form of sound synthesis that uses eurorack modules, which each perform a basic function, that produce control signals to modulate other modules’ parameters.

A basic monophonic modular setup can be seen the following diagram.

The voltage-controlled oscillator generates the audio signal at a specific frequency, and usually operates at a 1V/octave standard. A voltage-controlled filter has a similarly controlled cut-off frequency.

A voltage-controlled amplifier can be used to act as a signal multiplier to control the volume of the oscillator, which is often controlled by an ADSR envelope. The ADSR envelope can also modulate the VCF’s frequency.

With the addition of waveshaping, which refers to running the audio through a function such as the hyperbolic tangent, all functionality of these modules were used to create the basis for the designed system.

The wavetable aspect refers to how oscillator sound is generated. This is done by storing a waveform in a look-up table (or LUT) and using a pointer into this table to linearly interpolate between samples.

**Overall system**

I will now be detail the signal chain for every generated note.

3 wavetable oscillators generate the audio, of which 2 are detuned, with vibrato FM applied by an additional oscillator. The oscillators are then mixed to create a stereo signal. Each stereo channel is then optionally waveshaped, with a specified gain into the function. It is then stereo filtered by a user-chosen filter, with an ADSR-modulated cut-off frequency, set relative to the fundamental frequency of the note. Afterwards, the stereo signal is volume-modulated according to the MIDI velocity information and the ADSR volume envelope.

Each note is played by a generator component, which is efficiently managed by a generator manager component.

All code is implemented C, with some C++ compiler directives, and is functional AND NOT OBJECT ORIENTED, so that assembly code is more predictable.

**Focused sections**

Each software component present in the designed system will now be briefly discussed. There are many design choices, variables and other details that were considered for each component. There is too much detail to discuss in this video. Please refer to the report for any clarification.

All audio and system test data was generated by the C implementation, and then analysed and plotted in MATLAB.

**LUTs, lerp, and wavetables**

LUTs are generated for the 4 basic waveforms, using the Fourier series expansion. A 3D array is used to store copies of a single waveform with varying harmonics, so that frequency scaling will not cause aliasing.

The ability to linearly interpolate between samples in the LUT allows for accurate recreation of the signals. The 3D array further allows for interpolation between the waveforms as well, thus allowing for standard wavetable-based synthesis, such as seen in the Serum VST.

A way of storing the required amount of harmonics in the waveform was devised, with an efficient way to index to the correct waveform, so that aliasing is prevented.

To validate correct operation, a square-wave chirp signal was generated, and a spectrogram was produced, to ensure that no aliasing occurs over a 20 Hz to 10 kHz range.

A variety of inter-wavetable interpolated waveforms were generated, known as the wavetable position, and plotted to ensure correct operation.

As can be seen from the figures, correct operation is achieved. (MAYBE MORE?)

**ADSR**

An ADSR envelope with retriggering capabilities was designed as a state-machine. To test envelope operation, a two subsequent trigger-on events, followed by a trigger-off event was performed on the envelope component. This simultaneously shows correct trigger and re-trigger capabilities.

The result was plotted and shows successful implementation.

**Filters**

The bilinear transform was used to derive the discrete versions of 5 analogue filters, using a method known as filter prototyping. The designed filters were the 12 and 24 dB/octave low pass and high pass filters, along with a 2-pole bandpass filter. This method yields 5 IIR filters and equations for their coefficients.

An efficient way of calculating filter coefficients was devised, since coefficients must be recalculated each sample, if a filter-cutoff ADSR envelope is applied. A discreet time difference equation is used to perform filtering.

To test the filters, Gaussian white noise was filtered, with a cut-off frequency that is controlled by an ADSR envelope. The expected ADSR cut-off frequency was also recorded separately. This test simultaneously shows correct filter operation and cut-off modulation. A Q of 3 is chosen, so that the cut-off can be clearly visible, if applicable.

A 3D spectrogram was created in MATLAB, with the plotted surface having a Guassian blur applied to it, typically used for digital image blurring. This was to remove the sharp edges from the spectrum of the white-noise, and does not affect the shape of the filtered data in a significant way, since white noise frequency data is inherently noisy.

The recorded ADSR output was also plotted.

The results show correct operation of the filters. Listening to the audio of the results also show correct operation, but this is qualitative, and was not discussed in the report.

**Waveshaping**

The audio input of this component is scaled by a gain factor, and then fed though waveshaping functions. The considered functions are the hyperbolic tangent and the sinusoid. Both functions will add odd harmonics into the audio, and thus have potential for aliasing. Generally, a higher gain causes more aliasing.

Analytical solutions to the generated harmonic amplitudes are either not easily computable or impossible, so a numerical approach was followed. A threshold of -40 dB to the fundamental was chosen, and a sinusoid was passed through the functions at various gains. The location of the first harmonic exceeding this threshold was recorded.

This information was then used to split the incoming signal into low and high passed versions, with the low-passed signal being waveshaped. Oversampling would be best used in combination with this technique, but was neglected due to computational considerations.

To test the anti-aliasing filtering, a sinusoidal chirp was used at varying gains, with and without the filtering. The spectrograms of the results show a clear reduction in aliasing for both waveshaping functions, which is especially visible at higher gains.