	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 (54)

I'm Marco Rademan, and my supervisor is Prof Jaco Versfeld. I will be presenting my final year project titled: "Designing Software for a Wavetable Audio Synthesiser". 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.

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.

The system can play up to a fixed number of notes, each with an arbitrary frequency, by using on/off note triggers, ideal for MIDI integration.

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

Overview (14)

After an audio demo, I will describe modular synthesis and how it is relevant to this project.

I will then move on to each software component and summarise the design procedures, and the test results from each component.

Finally an overall system test will be detailed, and the results concluded.

Audio (9)

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, to demonstrate the system.

Modular parts + LIT (56)

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, 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 (48)

I will now 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. Afterwards, the stereo signal is volume-modulated according to the MIDI velocity information and the ADSR volume envelope. It is finally stereo filtered by a user-chosen filter, with an ADSR-modulated cut-off frequency, set relative to the fundamental frequency of the note.

Each note is played by a generator component, which is efficiently managed by a generator manager component. The other components are shown on screen.

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

LUTs, lerp, and wavetables (55)

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

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. A pointer can be swept across the table to play back an arbitrary frequency. The samples are linearly interpolated. 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.

To validate correct operation a variety of inter-wavetable interpolated waveforms were generated, and plotted to ensure correct operation. A square-wave chirp signal was also generated, and a spectrogram was produced, to ensure that no aliasing occurs over a 20 Hz to 10 kHz range.

As can be seen from the figures, correct operation is achieved.

ADSR (20)

An ADSR envelope with retriggering capabilities was designed as a state-machine. To test envelope operation, 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 (60)

The bilinear transform was used to derive the discrete versions of 5 analogue filters, using a method known as filter prototyping. This yields 5 IIR filters and equations for their coefficients.

An efficient way of calculating filter coefficients was devised, since coefficients must be recalculated for each sample, if a filter-cutoff ADSR envelope is applied. A discrete 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. A 3D spectrogram was created, 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 expected cut-off frequency is also plotted. The results show correct operation of the filters.

Waveshaping (60)

The audio input of this component is scaled by a gain factor, and then fed through waveshaping functions. Both functions will add odd harmonics into the audio, and thus can cause aliasing for high gains and input frequencies. The considered functions are the hyperbolic tangent and the sinusoid. 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 last harmonic exceeding this threshold was recorded, which yields an estimate of the bandwidth required for a specific waveshaping gain.

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 technique, a sinusoidal chirp was waveshaped 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.

Generator (45)

The generator component is the component that comprises the DSP chain for every note. This generator is what brings it all together. Its operation has already been discussed in the system overview section of this presentation.

Vibrato FM was tested and analysed using a spectrogram, which demonstrated proper operation.

Finally, the stereo width was tested by using an industry-standard plot to visualise audio width, known as the lissajous plot. The mid and side content of the audio are treated as y and x coordinates, resulting in a parametric plot. The wider the audio, the more the parametric plot will tend towards the side-channel axis.

Detune values were chosen carefully, and the stereo width parameter was swept, which shows a clear increase in width on the lissajous plot, indicating correct operation.

Generator manager (45)

The generator manager is the final top-level component of the system. It assigns note-on triggers to appropriate available generators, and makes them available once per audio buffer request. It manages the generators in O(N) time complexity, where N is the number of available generators.

Using array-implementations of a queue, stack and hashtable, this component can efficiently manage the generator, and keep track of generator note associations. It can also retrigger the oldest generator if no generators are available for a new incoming note. Freeing the generators that have finished playing makes use of a simple element shifting algorithm.

Normal operation and overload operation was tested, by having the generators set sinusoidal outputs. A spectrogram was generated, and clearly shows correct normal and overload operation.

Full system test (20)

To verify full system operation, strategic values were chosen for system parameters so that their effect can clearly be seen in the spectrogram and time data.

The filter envelope, vibrato, detune and volume envelopes can be clearly and correctly observed for the chosen system parameters, indicating correct system operation.

Conclusion (35)

Implementation on a microcontroller, would be the next logical step for this project. This project can be further used to develop a variety of complex products, which could have better integrated MIDI support, such as mod-wheel, pitch wheel, and hold-pedal support, along with

parameter-mapped MIDI control. A system that allows for a user-specified signal chains can also be developed, using the component-based approach this system was built with.

All in all, this system forms a good software basis for any hardware synthesiser product, and successfully achieved its goals, for being a foundational design and exploration of sound synthesis, inspired by the synthesis style of eurorack modules.