
SYNTH

Inhaltsverzeichnis

Einleitung

Der Software-Synthesizer „Synth“ ist ein VST-Plugin, welches im Rahmen der Vorlesung Audio-Video-Programmierung der HAW Hamburg entwickelt wurde.
Zur Verwendung kamen dabei die Entwicklungsumgebung Microsoft Visual Studio 2013 und das Framework JUCE. Laut Konzept vom 08. November 2015 soll das Plugin einen polyphonen Synthesizer mit einem Oszillator für die vier Basiswellenformen Sinus, Dreieck, Sägezahn und Rechteck, sowie die Einstellungen der Parameter Attack, Decay, Sustain und Release für die Hüllkurve bereitstellen. Neben diesen Funktionen ist in der fertigen Version des Plugins noch ein Delayeffekt und ein Rauschen verfügbar.

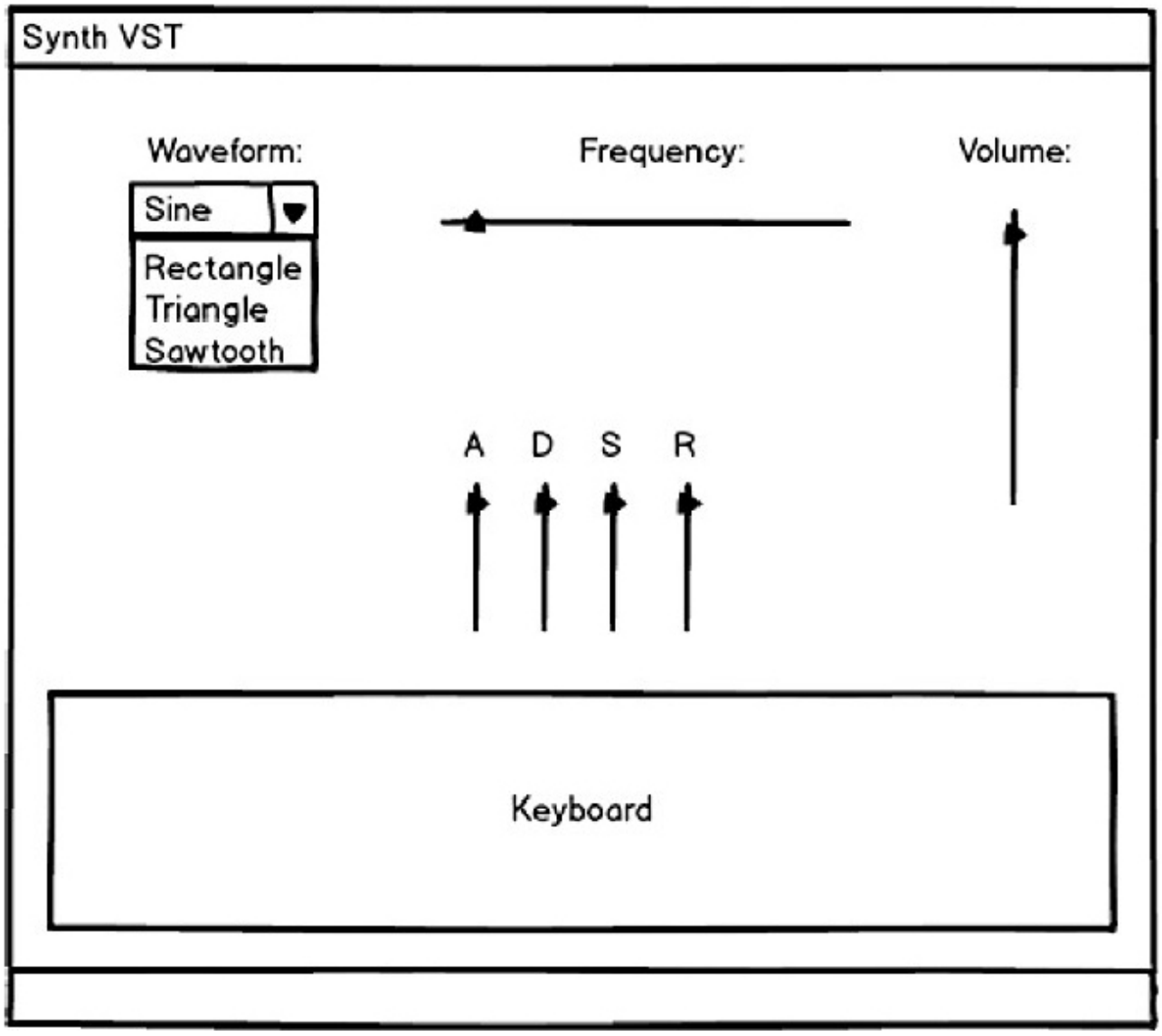

Abbildung 1: Konzept

Überblick von „Synth“

Zu Anfang des Signalflusses steht ein Oszillator, der die Basiswellenform generiert. Zur Auswahl stehen hier Sinus, Dreieck, Sägezahn und Rechteck. Die Sinuswelle hat keine Obertöne und klingt daher sehr dumpf und einfach. Die Dreieckwelle hat etwas mehr Obertöne als die Sinuswelle, die Sägezahnwelle wiederum mehr als die Dreieckwelle und die Rechteckwelle letztendlich mehr als die Sägezahnwelle.

Im nächsten Abschnitt wird die Hüllkurve der Welle geformt. Dabei kann mit dem Parameter Attack die Einschwingzeit der Transienten definiert werden. Eine Attackzeit von 10 ms bedeutet, dass der Ton nach dem Anschlag erst nach 10 Millisekunden die volle Lautstärke erreicht. Der Parameter Sustain beschreibt die Lautstärke, mit der ein Ton gehalten wird, während die Decayzeit jene ist, die verstreicht, bis der Ton nach der Attackphase die Lautstärke des Sustain erreicht. Nach Loslassen der Taste entscheidet die Releasezeit, wie lange es dauert, bis der Ton von der Sustainlautstärke aus leiser wird und verstummt.

Der Delayeffekt ist ein Effekt, der das Eingangssignal verzögert erneut ausgibt. Die Verzögerungszeit in Millisekunden lässt sich über den Parameter der Delaylänge einstellen. Zudem kann mit den Parametern Dry und Wet die Lautstärke des Originalsignals und des verzögerten Signals eingestellt werden.

Mit dem Rotary-Slider für das Rauschen kann man dem Signal einen beliebigen Anteil zufällig verteilter Samples beifügen und hat damit etwas mehr Vielfalt in der Klangbearbeitung.

Die grafische Benutzeroberfläche (GUI) wurde zum größten Teil mit dem Introjucer von JUCE konstruiert. Dabei fanden Labels, Rotary-Slider und Combo-Boxen Verwendung. Der Hintergrund des Plugins dagegen wurde in Adobe Photoshop erstellt, um das Plugin etwas interessanter zu machen. Dabei bestand die Motivation darin, die Oberfläche etwas verstaubt und zerkratzt wirken zu lassen, sodass es an alte analoge Vintage-Geräte erinnert. Dies liegt nahe, da dieser Synthesizer nur grundlegenden Funktionen beinhaltet, ähnlich wie bei den alten Hardwaremodellen.

Bislang gibt es nur ein Problem mit dem Delay, welches selten nach nicht eingrenzbaren Parametereinstellungen ausfällt und bis zum Neustart des Plugins nicht mehr funktioniert.

Im Detail: Die Hüllkurve

Im Folgenden wird die Funktionalität der Hüllkurve genauer erklärt:

Die Hüllkurve (engl.: Envelope) dient zur Beschreibung der Ein- und Ausschwingphase von Signalen. Dabei hat sich ein Modell für die Verwendung im Bereich der musikalischen Signalbearbeitung durchgesetzt, das ADSR-Modell. Die Initialen A, D, S und R stehen dabei für die Parameter, die in dem Modell verwendet werden. A steht für Attack, D für Decay, S für Sustain und R für Release. Attack und Decay bilden die Einschwingphase, welche auch Transiente genannt wird. Sustain definiert die Lautstärke, auf der ein Ton gehalten wird und Release bestimmt die Zeit, die der Ton brauch um auszuklingen.

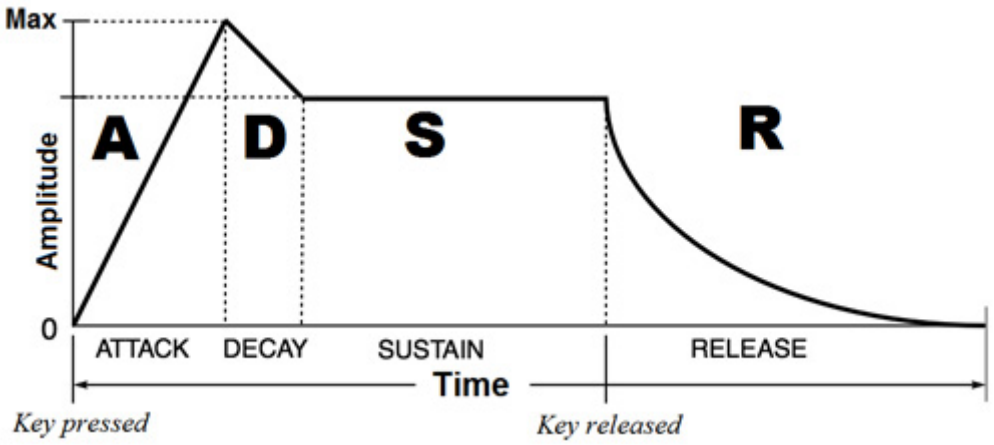

Abbildung 2: ADSR-Modell (Quelle: http://www.songsofthecosmos.com/images/ADSR.jpg)

Der genaue Ablauf des Modells kann wie folgt beschrieben werden: Sobald ein Ton ausgelöst wird (Note On), zum Beispiel durch eine Taste, befindet sich der Ton in der Attackphase. Die Attackzeit definiert die Zeit in Millisekunden, die der Ton benötigt, um die Lautstärke vom Minimalpegel (Gainlautstärke 0) auf Maximalpegel (Gainlautstärke 1) zu erhöhen. Bei einer Attackzeit von 0 ms beginnt der Ton sofort bei dem Maximalpegel. An die Attackphase schließt die Decayphase an. Die Decayzeit bestimmt die Zeit, in der der Ton vom Maximalpegel auf den Sustainlautstärke sinkt. Die Sustainlautstärke ist ein Wert zwischen Minimalpegel und Maximalpegel. Während die Taste gehalten wird, bleibt der Ton auf der Sustainlautstärke bis sie losgelassen wird (Note Off), woraufhin die Releasephase beginnt. Sollte der Sustainwert dem Minimalpegel entsprechen, wird die Sustainphase übersprungen und die Releasephase setzt am Ende der Decayphase ein. Die Releasephase senkt die Lautstärke in Releasezeit von der Sustainlautstärke auf den Minimalpegel.

Realisiert wurde das Modell folgendermaßen:

In JUCE bildet die Klasse AudioProcessor die Steuereinheit, während AudioProcessorEditor die Elemente der grafischen Benutzeroberfläche (GUI) strukturiert und Listenerfunktionen verwaltet. Sobald das Plugin gestartet wird beginnt die Methode `void SynthAudioProcessor::processBlock (AudioSampleBuffer& buffer, MidiBuffer& midiMessages) {…}` des Audio-Prozessors in einer Dauerschleife Bufferblöcke zu verarbeiten und auszugeben. Sofern keine MIDI-Informationen vorliegen, ist nichts zu hören. Wenn aus der Combo-Box eine Basiswellenform gewählt wurde und MIDI-Informationen vorhanden sind – sei es wenn live eine MIDI-Taste in der Digital Audio Workstation (DAW) gedrückt wird oder dort ein MIDI-Objekt abgespielt wird – werden diese in der Klasse SynthVoice indirekt durch die Klasse Synth Block für Block gerendert.

In der Methode `void processBlock(AudioBuffer<FloatType>& outputBuffer, int startSample, int numSamples) {…}` der Klasse SynthVoice wird die entsprechende Basiswellenform Sampleweise berechnet. Auf die Samples wird nun zugegriffen, um die Lautstärkeanpassungen für die Phasen der Hüllkurve vorzunehmen, noch bevor sie in den Buffer geschoben werden. Im nächsten Schritt sind die Daten der Basiswellenform Blockweise im Buffer mit Floats für jeden Kanal gespeichert. Der jeweilige Laustärkewert wird beim Übergeben in den Buffer auf das Sample angewendet. Um herauszufinden wann jede Stimme des Synthesizers angeschlagen wurde, wird die Methode `void startNote(int midiNoteNumber, float velocity) {...}` der Klasse SynthVoice ergänzt. Sobald eine MIDI-Information vorhanden ist, wird diese Methode von der DAW aufgerufen und der Status, welcher in Form eines Enumerators implementiert wurde, wird auf NOTEON gesetzt. Anschließend wird in der Schleife die Methode `void processBlock(AudioBuffer<FloatType>& outputBuffer, int startSample, int numSamples) {…}` der aktuelle Status und die Variablen für die Hüllkurve überprüft. Damit die Variablen immer auf dem aktuellen Stand des GUI sind, gibt es in der Klasse AudioProcessor die Methode `void setParameter(int index, float newValue) {...}`. In der Klasse AudioProcessorEditor wird beim Einsatz eines Listeners eine Instanz der Klasse AudioProcessor erzeugt. Diese Instanz wird genutzt, um die aktuellen Werte, die der Listener über die Elemente des GUI überwacht, mittels `void setParameter(int index, float newValue) {...}` dem Objekt des AudioProcessor zu übergeben. Dabei ist i der Index von `enum Parameters {...}`, in der alle Parameter enthalten sind. Dadurch kann ein Parameter selektiert und der Wert des GUI-Elements in der Klasse AudioProcessor und anschließend auch die Variablen der vorhanden SynthVoice - Objekte durch f ersetzt werden.

Nachdem der Status NOTEON gesetzt wurde liegt der aktuelle Lautstärkewert bei 0 und es wird überprüft, ob die eingestellte Attackzeit über 0 ms liegt und der Status auf ATTACK gestellt werden kann, anderenfalls wird diese Phase übersprungen. Während der Attackphase wird in Schleife die Funktion `void attack() {…}` ausgeführt. In dieser wird zuerst anhand der Attackzeit und der Samplerate die Anzahl der Samples berechnet, in denen die Steigung der Lautstärke von Minimalpegel auf Maximalpegel erfolgen soll. Anschließend wird für jedes Sample nur der berechnete Teil der Lautstärke angehoben, bis der Maximalpegel erreicht ist. Dann wird, sofern die Decayzeit höher als 0 ms ist, der Zustand DECAY aktiviert. Die Funktion `void decay() {…}` ist der Attackfunktion sehr ähnlich. Allerdings wird hier eine Lautstärkekurve berechnet, die von dem Maximalpegel auf die Sustainlautstärke fällt. Nachdem diese erreicht wurde, wird der Status auf SUSTAIN gestellt. Hier wird durch die Funktion `void sustain() {…}` nur eine Lautstärkeeinstellung des Sustainwertes auf die Samples angewendet. Nachdem der Ton losgelassen wird, stellt die Synthesizer-Stimme den Status NOTE OFF ein. Sobald dieser aktiv ist, wird er durch RELEASE ersetzt, wenn die aktuell eingestellte Releasezeit höher als 0 ms ist. In dieser Funktion `void release() {…}` wird wie schon bei der Decayfunktion die Lautstärke abgesenkt. Und zwar von der Sustainlautstärke auf den Minimalpegel. Nachdem diese Phase beendet ist, wird der Status OFF eingestellt und beendet den Ton.

Ausblick

Der Synthesizer besitzt die grundlegenden Funktionen und sogar noch einen Delayeffekt, der die Sounds wesentlich interessanter macht. Mit diesem Plugin kann man musikalisch schon eine Menge machen, kann aber noch durch weitere Effekte oder Filter erweitert werden. Interessant wäre auch die Umsetzung mehrere Oszillatoren, die man dann beliebig untereinander und mit verschiedenen Effekten kombinieren könnte.

Ich hatte auf alle Fälle viel Spaß an der Programmierung dieses Plugins und habe vor, mich in Zukunft weiter diesem Bereich zu widmen.