Informatik
Hauptcampus

H O C H
S C H U L E
T R I E R

Audio Plugin: Delay Effekt

Simon Steiner

Hausarbeit zur Vorlesung Tool- und Plugin-Programmierung

Trier, 28.2.2019

Kurzfassung

Wie werden Signale mit einem Audio-Plugin verzögert wiedergegeben? Dies wird in dieser Ausarbeitung genauer behandelt. Dabei wird eine neue Klasse erstellt die all dies übernimmt. Ein wichtiges Konzept ist dabei ein Kreisspeicher der am Ende wieder von vorne beginnt. Ein ansprechendes UI sowie angenehme Interaktionen damit sind auch ein wichtiger Punkt, den ein Produzent arbeitet bei fast jedem Lied mit Delayplugins.

Inhaltsverzeichnis

1

Einleitung und Motivation

Musik findet sich überall im Alltag. Sei es im Einkaufszentrum oder auf dem Smartphone. Doch wie wird eins dieser Lieder produziert. Jede Person mit einem Computer kann heutzutage Musik damit erzeugen. Anwendungen, die dies können, sind weit verfügbar und in verschiedenen Preisklassen erhältlich. Ein persönlicher Favorit ist Ableton Live. Diese digitalen Workstations enthalten oftmals mehrere digitale Instrumente und Effekte. Ein beliebter Effekt ist das verzögerte Wiedergeben von den erzeugten Tönen wie zum Beispiel das Rufen in den Bergen ("Wer ist der Bürgermeister von Wesel"). Dies wird oftmals als Delay Effekt bezeichnet. Ein bekanntes Lied, was diesen Effekt benutzt hat, ist *The Police - Every Breath You Take*.

Das Ziel dieser Arbeit ist es ein funktionierendes Delay Plugin für Ableton Live zu erstellen. Zum dem sollte es auch das verzögerte Audiosignal mehrmals wiederholen und dabei leiser werden. Mit Hilfe eines Reglers lässt sich das Mischverhältnis zwischen verzögertem Audiosignal und originalem Input verändern.

2

Allgemein

Um Musik am Computer zu produzieren werden verschiedene Synthesizer und Effekte benutzt. Diese sind oftmals in C++ programmiert. Wichtig ist hierbei die Unterstützung von dem Musikprogrammen (Ableton Live) und das Wissen über verschiedene Plugin Formate.

2.1 JUCE

Juce ist ein C++ Framework zur Erstellung von Desktop und mobilen Anwendungen. Das Framework ist Open-Source und kann von dem GitHub-Repository heruntergeladen werden. Dort sind verschiedene Wrapper-Klassen enthalten die es ermöglichen, Audio Anwendungen zu erstellen [JUC19].

Die Besonderheit von Juce bildet das Tool Projucer. Dieses Tool ist eine IDE zum Erstellen und Verwalten von Juce Projekten. Mit ein paar Klicken erstellt dieses Tool für jede gewünschte Plattform ein Projekt. Zum Beispiel kann eingestellt werden, dass ein Visual Studio 2017 und ein XCode Projekt erstellt werden kann. Bei Projekte verwenden den selben Sourcecode. In den Projekten selber werden, je nach Auswahl, weitere Projekte erzeugt abhängig davon, welche Plugin Formate erstellt werden sollen.

Der Projucer erzeugt ein kleines Programmcodeskelett mit nützlichen Standardfunktionen. Dies vereinfacht den Einstieg, so dass nicht viel Zeit mit der Einarbeitung von irgendwelchen SDKs verschwendet wird.

2.2 Plugins

In der heutigen Musikproduktion werden verschiedene Formatstandards für Plugins eingesetzt. Die Wahl für einen gewissen Standard ist abhängig von dem gewählten Betriebssystem sowie welche digitale Audioworkstation (DAW) verwendet wird. Um festzustellen, welche Formate von der jeweiligen DAW unterstützt wird, sollte die Herstellerwebseite bezogen werden. Die Formate unterscheiden sich in ihrer Funktion nicht großartig. Dazu eine kleine Liste von bekannten Formaten und unterstützenden DAWs [PF19]:

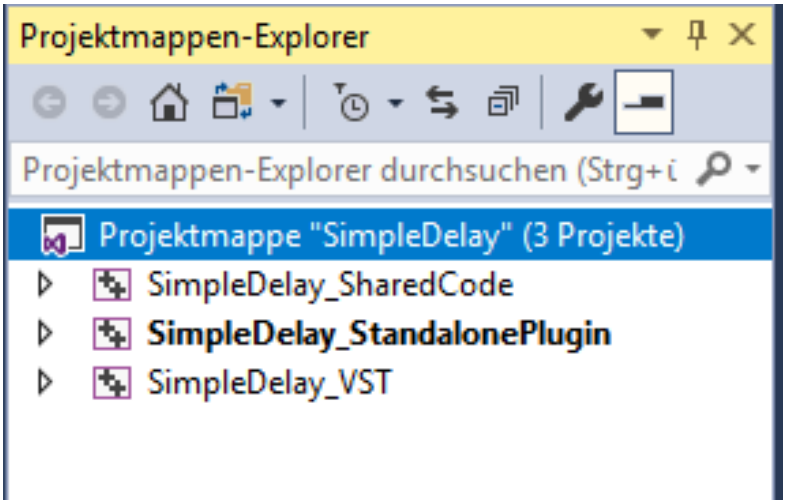

Abb. 2.1. VS '17 Projektmappen-Explorer. Erstellen des Projekts erzeugt hier ein .exe und ein VST2 Plugin (.dll)

- VST (Virtual Studio Technology): Ableton Live, Fl Studio, Cubase, Reason, Reaper
- AU (Audio Unit)[Nur Mac]: Ableton Live, Logic Pro, Studio One
- RTSA (Real Time AudioSuit): Pro Tools
- AAX (Avid Audio eXtension): Pro Tools

Steinberg stellt zur Erstellung von VSTs ein SDK bereit. Im Moment beginnt bei Steinberg eine Umstellung von dem VST2 Standard auf den VST3 Standard. In der aktuell verfügbaren SDK gibt es keine Unterstützung mehr für die Erstellung von VST2 Plugins. Manche Hersteller haben sich noch nicht angepasst wie zum Beispiel Ableton, diese akzeptieren bislang noch keine VST3 Plugins. Ältere Versionen von Juce haben noch einen VST2 Support zum Erstellen von Plugins. Die aktuelle Version kann jedoch nur noch den neusten VST Standard erzeugen. Da zum Testen stand nur Ableton zu Verfügung, weshalb das Plugin mit einer älteren Version von Juce erstellt werden musste. Das Plugin selber ist eine .dll-Datei, welche sich im VST-Plugin Ordner der DAW befinden muss. Wichtig ist, dass das Plugin und die DAW die selbe Bit-Architektur haben (32/64).

3

Konzept und Theorie

Die Umsetzung des Delay Plugins besteht aus zwei Teilen: Das Delay und die Steuerung der Parameter mit Hilfe einer GUI.

3.1 Delay Effekt

Der Delay Effekt ist im Grunde eine Wiederholung des Eingangsignals zu einer späteren Zeit (siehe 3.1: Spur 2). Da zu einem späteren Zeitpunkt dieses exakte Signal nicht mehr da ist muss es in einem Buffer gespeichert werden. Zur Vereinfachung wird der Buffer wie ein kreisförmiger Speicher gesehen. Mit Hilfe von dem Modulo-Operator ist dies programmiertechnisch möglich. Sollte zu einem bestimmten Zeitpunkt das vergangene Signal zum aktuellen Input hinzugefügt werden, so muss eine gewisse Anzahl an Samples im Speicher zurückgesprungen werden. Dies lässt sich mit Hilfe der Samplerate berechnen. Um dieses Signal nicht nur einmal wiederzugeben sondern langsam abschwächen zu lassen (siehe 3.1: Spur 3) muss das verzögerte Signal wieder in den Delaybuffer eingespeist werden.

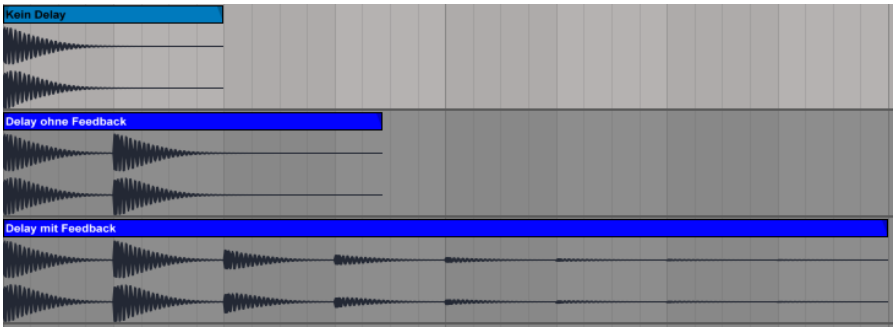

Abb. 3.1. Ausschnitt aus Ableton Live bei Anwendung des Delay Plugins. 3 verschiedene Audiospuren. 1. Spur: Nur das original Signal. 2. Spur: Signal wird nach 500ms wiederholt. 3. Spur: Wiederholung des Signals mit Einspeisung des verzögerten Signals aber immer leiser werdend. [Siehe Anlage Audio 1 bis 3]

Als Veranschaulichung dient hierbei Abbildung 3.2. Hier werden N Samples direkt vom Input an den Output weitergeleitet als auch im DlyWBuf gespeichert.

Das verzögerte Signal g wird zusammen mit dem Input addiert und befindet sich dann im Output. *fb* entspricht dem selben Signal wie g und wird gleichzeitig mit dem aktuellen Input im Delaybuffer gespeichert.

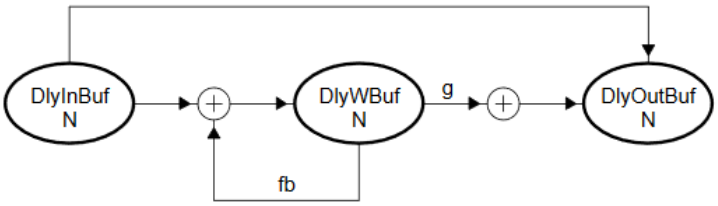

Abb. 3.2. Ablaufdiagramm einer typischen Delayline mit Feedback. Input landet direkt im Output. Input wird gleichzeitig im Delaybuffer gespeichert. Verzögertes Signal wird zusammen mit Input im Output zusammengeführt. Gleichzeitig wird das verzögerte Signal wieder im Delaybuffer eingespeist. [HTC19]

3.2 GUI

Es gibt bei dieser Art von Delay 3 Werte die verändert werden könnten. Eine mögliche Umsetzung wäre das Nutzen von 3 Slidern. Einen für den Verzögerungswert (Delaytime), Feedback und Mischverhältnis. Für zukünftige Plugins ist jedoch die Verwendung eines Felds, wo Werte abhängig von einer Position auf der x- und y-Achse sind, von nutzen. Slider sind in dem Framework schon standardmäßig dabei, zusammen mit dazugehörigen Observer-Pattern. Zur Anzeige der Werte kann bei dem Slider eine zugehörige TextBox (TextEditor) aktiviert werden.

Das ControlPad muss jedoch selbst umgesetzt werden. Es besteht aus einem Kontrollpunkt (Trackball). Der Mittelpunkt des Trackballs darf nicht das darunterliegende Feld verlassen. Um die Werte anzuzeigen, welche die jeweilige Achsenposition repräsentieren, gibt es noch zwei TextBoxen (xValue, yValue). Alles zusammen ergibt das Element *ControlPad* .

Das Hintergrundbild wirkt als Anlehnung an das Echo in den Bergen.

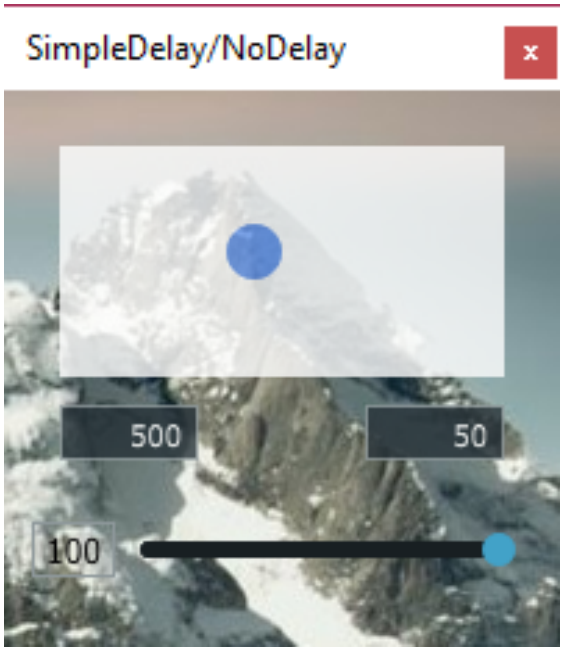

Abb. 3.3.

4

Implementierung

4.1 Delay Processing

Das wichtigste in der digitalen Signalverarbeitung ist die Methode *processBlock()*. In dieser Methode findet die ganze Signalbearbeitung statt. Im AudioBuffer Objekt stehen die aktuellen Eingangsamples. Diese werden immer blockweise geladen. Dieser AudioBuffer ist zeitgleich auch der Ausgangsbuffer.

Der Nachfolgende Code ist des Herzstück des Delay Plugins. Hier werden die aktuellen Samples in den Delaybuffer des Objekts *d* gespeichert. Dann wird in ein AudioBuffer-Objekt die vergangenen Samples aus dem Delaybuffer kopiert. Die darin enthaltenen Samples werden mit einer gewissen Lautstärke (Feedbackwert) zu den vorhandenen Samples in dem Delaybuffer aufaddiert (nicht ersetzt!) aber an der selben Stelle wie vorher die Eingangssamples gespeichert wurden. Alle vergangen Samples erhalten noch eine gewisse Lautstärke (Mixwert). Am Ende werden die vergangenen Samples im AudioBuffer-Objekt zu den Eingangssamples aufaddiert.

Listing 4.1. Ausschnitt aus processBlock()

```
1  for (int channel = 0; channel < totalNumInputChannels; ++channel)
2  {
3  float* inputData = buffer.getWritePointer (channel);
4
5  d.setBufferData(channel, inputNumSamples, inputData);
6
7  float delayTimeVal = *parameters->getRawParameterValue("timeID");
8
9  d.getBufferData(channel, inputNumSamples,delayTimeVal, delayedSignal);
10
11 float feedbackVal = *parameters->getRawParameterValue("feedbackID");
12
13 d.addFeedbackData(channel, inputNumSamples,
14    feedbackVal/100.0f, delayedSignal.getReadPointer(channel));
15
16 float mixVal = *parameters->getRawParameterValue("mixID");
17 delayedSignal.applyGain(mixVal/100.0f);
18
19 buffer.addFrom(channel, 0, delayedSignal, channel, 0, inputNumSamples);
20
21 }
```

Nach der Schleife ist es nur wichtig mit der Methode *updateBufferPos()* aufzurufen um die Schreibposition zu verschieben.

Alle Operationen die auf dem Delaybuffer (Kreisspeicher) arbeiten befinden sich im Delay-Objekt. Jede diese Operationen verschiebt aber nicht die Samples einzeln in den Delaybuffer sondern mit Hilfe von Vektorfunktionen. Andernfalls wird die Laufzeit möglicherweise beeinflusst und zwar auf die schlechte Art.

Das einzige Problem an dieser Implementierung ist, wenn die Delaytime geändert wird, entstehen Artefakte. Dies kommt daher, dass sich die Position zum lesen der vergangenen Samples geändert hat. Damit entsteht zwischen dem alten und neuen Wert ein zu großer Unterschied den der Lautsprecher nicht natürlich wiedergeben kann. Daraus folgt ein Knackgeräusch. Dies könnte mit Hilfe von Interpolation entfernt werden.

4.2 ControlPane

Die ControlPane-Klasse war als langfristige Erweiterung geplant. Diese könnte zum Beispiel für eine Implementierung eines Filters mit Resonanz genutzt werden. Dafür wurden Vorkehrungen bei der Implementierung von Methoden gewählt.

Die Wertebereichslimits können auch so gesetzt werden, dass der Minimalwert negativ sein kann. Die Methoden zur Umrechnung von Achspositionen zu Absolutwerten (*convertAxisToValue()*) und umgekehrt von Absolutwert zu Achsenposition (*convertValueToAxis()*) berechnen korrekte Werte egal ob der Anfangswert negativ ist. Solche Berechnungen, welche Werte von einem bestimmten Bereich auf einen anderen ableitet, sind ein persönlicher Favorit von mir.

Das der Trackball über das Feld hinausgehen kann wirkt auf den User besser. Könnte der Trackball sich nicht hinaus bewegen, so wirkt der Trackball nicht als rundes Objekt sondern eher als Quadrat. Damit bleibt die Illusion erhalten das es wirklich ein rundes Objekt ist. Zu dem wirkt es schöner wenn der Mittelpunkt des Trackballs an der Mausklick Position ist.

5

Bedienungsanleitung

Es lassen sich Delaytime (Zeit, die verstreicht in Millisekunden bis das Signal wiederholt wird), Feedback (Lautstärkewert, mit der das wiederholte Signal wieder in den Delaybuffer eingespeist wird) und das Mischverhältnis im Output zwischen Inputsignal und dem Signal aus dem Delaybuffer.

Ist das Mischverhältnis auf 0 Prozent, so wird nur das Eingangssignal ausgegeben. Je höher das Mischverhältnis des mehr wird von dem verzögerten Signal gehört. Bei 100 Prozent ist Eingangssignal und das verzögerte Signal gleich laut.

Das Verschiebung des Trackballs auf der x-Achse verändert die Zeit bis das Signal wiederholt wird. Mit diesem Plugin ist eine Zeit zwischen 100ms und 1000ms möglich.

Das Verschieben des Trackballs auf der y-Achse verändert die Lautstärke des wiederholenden Signals. Es ist nicht möglich den Wert auf 100 zu setzen, da das Signal sonst ewig wiederholt wird.

Im Ordner *Audio* liegen 3 Beispiele. Eines ohne Delay, eines mit einem 500ms Delay und das selbe noch einmal nur mit Feedback.

Im Ordner *Dateien* liegen eine eigenständige Anwendung und eine .dll zur Verwendung in DAWs.

5.1 Plugin

Um das erstellte Plugin in einer DAW zu verwenden:

- Projekt erstellen. Damit wird eine .dll erzeugt
- Aus ../Projektordner/Builds/VisualStudio2017/Win32/Debug/VST die Datei mit der Endung *.dll* in den VSTPlugin Ordner der DAW ziehen und neu scannen lassen
- Sollte es dort nicht gefunden werden, kann die Bit-Architektur des DAW/Plugin ungleich sein. Lösung: Plugin mit einer anderen Architektur erstellen
- Plugin nach einem digitalem Instrument platzieren und an den Reglern die Werte verändern

5.2 Standalone

Achtung: Bei Verwendung einer Lautsprecheranlage kann es zu Rückkopplungen kommen.

Die eigenständige Datei (.exe) starten. Es können möglicherweise Dateien (.dll) fehlen um die Datei auszuführen. Nach Installation der fehlenden Datei sollte auch die Standalone Variante funktionieren.

6

Zusammenfassung und Ausblick

Mit dem JUCE Framework ist es möglich ein Delay Plugin zu erstellen. Dieses funktioniert ohne Knackgeräusche wenn während der Nutzung der Delaytimewert nicht verändert wird. Folglich empfiehlt sich also nicht den Wert in einer DAW zu automatisieren.

Die Erstellung eigener UI-Komponenten ist auch relativ einfach möglich. Diese können entweder aus weiteren vorgefertigten Elementen bestehen oder eigene zeichnen wie in diesem Fall den Trackball und das Begrenzungsfeld.

Diese Variante könnte in der Zukunft weiter optimiert (Interpolation) oder um weitere Funktionen erweitert werden. Eine mögliche Erweiterung ist ein unterschiedliches Delay pro Kanal, so dass einer eine Verzögerung von 500ms und der andere von 550ms hat.

Literaturverzeichnis

HTC19. *HowtoCreateDelay-basedAudioEffectsontheTMS320C672xDSP*, 28.2.2019.
`http://www.ti.com/lit/an/spraaa5/spraaa5.pdf`.

JUC19. *JUCE*, 28.2.2019.
`https://en.wikipedia.org/wiki/JUCE`.

PF19. *Plugin Formats Explained*, 28.2.2019.
`http://support.pluginboutique.com/knowledgebase/articles/51119-plugin-formats-explained-vst-au-aax-etc`.