Es-Chord for Bela

Tim Strauch und Philipp Steigerwald
strauch1210@googlemail.com
p.steigerwald@campus.tu-berlin.de

Technische Universität, Berlin, Deutschland

0 Einleitung

Im Rahmen des Seminars ”Sound Synthesis with FAUST” von Henrik von Coler an der TU Berlin entstand der Es-Chord for Bela, ein digitaler String-Modelling Synthesizer im Eurorack-Format. Das Ziel war es, einen Synthesizer zu entwickeln, welcher bei monophonem Spiel zufällige Harmonien über den Grundtönen bildet und somit das Spiel des Musikers auf eine einzigartige Weise begleitet. Als Vorbild für den implementierten Zufallsgenerator wurde auf die Konzepte von analogen Synthesizern zurückgegriffen, wie beispielsweise Don Buchla’s ”Source of Uncertainty”. Die Umsetzung erfolgte in der Programmiersprache FAUST und der Code wurde auf einem Bela Pepper Eurorack-Modul implementiert.

1 Grundlagen

Im Folgenden werden die Grundlagen der Klangerzeugung und der Generation von Zufallswerten des Synthesizers näher erläutert.

1.1 Klangerzeugung

Physical Modelling ist eine Art der Klangsynthese, bei der mittels mathematischen Modellen versucht wird, den Klang von akustischen Instrumenten möglichst originalgetreu abzubilden. Die charakteristische Klangfarbe eines Instruments ist hauptsächlich abhängig von dem resultierenden Obertonspektrum bei der Anregung mit einer bestimmten Anregungsfrequenz. Die Komplexität der Abbildung eines Obertonspektrums variiert zwischen verschiedenen Instrumenten und kann über unterschiedliche Algorithmen realisiert werden. Bei diesem Projekt wurde die Modellierung mit der Standard-Faust-Funktion ”comb-String” durchgeführt, welche basierend auf einem Kammfilter, die Klangfarbe einer vibrierenden Saite modelliert. Um ein Kammfilterspektrum zu synthetisieren wird meist ein Karplus-Strong Algorithmus genutzt. Der von Faust genutzte Algorithmus könnte so aussehen:

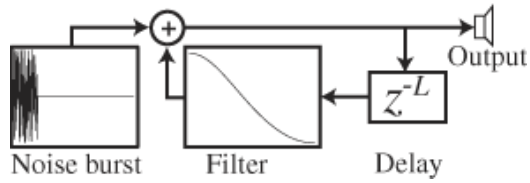

Fig. 1. Generischer Karplus-Strong Algorithmus zur Modellierung einer angezupften Saite [1]

Bei einem Karplus-Strong Algorithmus handelt es sich grundlegend um eine Feedback-Schleife. Ein Eingangssignal wird periodisch ausgelesen und bei jedem Durchlauf verändert. Bei dem Algorithmus in Abbildung 1 wird die Feedback-Schleife anfangs durch einen Noise Burst angeregt. Anschließend wird es unverändert ausgegeben und über einen rekursiven Signalweg um L Samples verzögert und tiefpassgefiltert. Das gefilterte Signal wird nun erneut ausgegeben und über den rekursiven Signalweg zurückgeführt. Diese Schleife wird so oft wiederholt, bis das gewünschte Abklingverhalten der Saite realisiert wurde. Das resultierende Übertragungsverhalten dieser Feedback-Schleife entspricht dem eines Kammfilters (vgl. Abbildung 2) und ähnelt dem Spektrum einer angezupften Saite ausreichend, um die Klangfarbe realistisch abzubilden. Deshalb wird dieser Algorithmus auch Plucked-String-Algorithmus genannt.

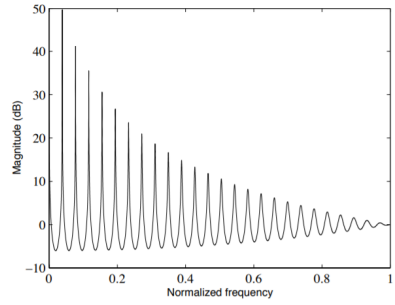

Fig. 2. Übertragungsfrequenzgang des Plucked-String-Algorithmus [2, S.111]

Das Klangbild des Es-Chord setzt sich aus den Grundtönen, synthetisiert durch die combString Funktion, sowie einer varrierenden Anzahl an zugehörigen Obertönen, bzw. Intervallen zusammen. Die Intervalle sind vielfache der Grundfrequenz und so können beispielsweise Terzen, Quinten und Oktaven passend zum gespielten Grundton erklingen. Um diese Zusammensetzung zu erreichen, wird die in der combString Funktion anzugebende Fundamentalfrequenz, mit den entsprechenden Faktoren multipliziert.

1.2 Zufallsgenerator

Um zufällige Noten oder Modulationen zu erzeugen, ist in vielen analogen Synthesizern ein Zufallsgenerator verbaut. Die meist genutzte Methode um zufällige Spannungen zu erzeugen, ist die Verwendung einer Sample & Hold-Schaltung. Diese ist in der Lage ein am Eingang anliegendes Signal (meistens weißes Rauschen) für eine bestimmte Zeit zu speichern. Die Zeit wird hier durch ein anliegendes Taktsignal (Clock) vorgegeben: Bei jedem neuen Taktsignal am Clock-Eingang wird ein neuer Wert des Eingangssignals gespeichert und solange gehalten, bis ein neues Taktsignal am Eingang registriert wird. Das hierbei resultierende Signal ist eine quantisierte Abbildung des ursprünglichen Eingangssignals (siehe Abb.3).

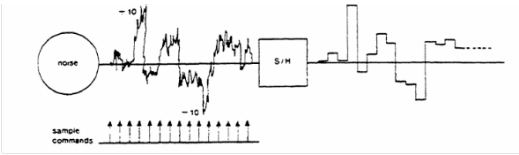

Fig. 3. Sample&Hold Mechanismus erzeugt Zufallswerte mit Hilfe eines Rauschsignals [3, S.83]

Werden diese Spannungen nun beispielsweise an einen exponentiellen FM-Eingang eines Oszillators gelegt, so erhält man eine zufällige Abfolge von Noten. Da die, nun quantisierten Spannungswerte eine sehr große Reichweite besitzen, (in Abb.3 -10 bis +10V) muss diese Spannung in der Regel kleiner skaliert werden, um beispielsweise Notenwerte innerhalb einer Oktave zu generieren. Dies wird bei einem analogem Synthesizer mithilfe eines Attenuators umgesetzt. Möchte man anstelle von diskreten Spannungswerten, kontinuierlich von einer Spannung zur nächsten “gleiten“ so wird in der Regel die Spannung durch einen Slew-Limiter tiefpass-gefiltert [4, S.252f.].

Die aus abgetastetem weißem Rauschen resultierenden Zufallswerte, besitzen eine gaußähnliche Verteilung, daher werden sehr kleine und sehr große Spannungswerte seltener generiert. Einige Zufallsquellen, wie die Buchla ”Source of Uncertainty”, besitzen die Möglichkeit die Verteilung der generierten Spannungen durch den Musiker zu beeinflussen. Somit kann eine gleichmäßigere Verteilung der Spannungswerte erreicht werden oder die Verteilung gezielt auf einen Bereich eingeschränkt werden [3, S.85].

2 Umsetzung in FAUST

Der Signalfluss des Es-Chords besteht aus zwei grundlegenden Signalwegen, die sich vor dem VCA summieren und zusammen eine Melodie und deren Begleitung bilden. Die Fundamentalfrequenz der Melodie, bzw. der Abfolge von Tönen, wird über externes CV gesteuert. Sie wird in den in den Summationsblock geführt und gibt gleichzeitig auch die Fundamentalfrequenz f0 für die harmonischen Obertöne vor. Die Obertöne bilden den zweiten Signalweg. Deren Anzahl und Zusammensetzung wird abhängig von den Ausgangswerten einer Sample & Hold Schaltung bestimmt, welche einen bestimmten Case im Case Picker triggern. Der ausgewählte Case bestimmt dann eine fest zugeordnete Begleitung.

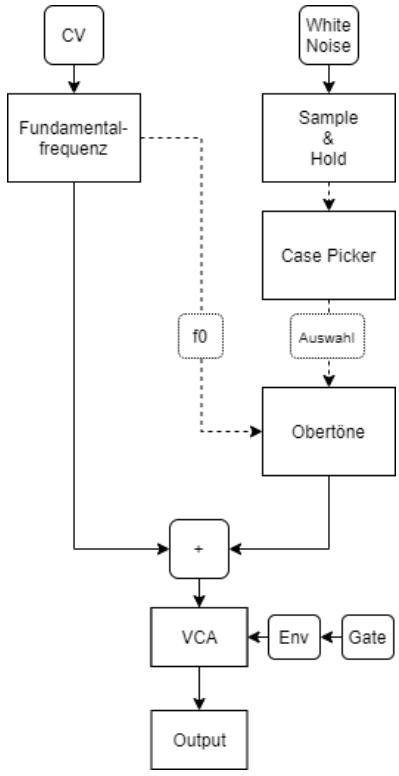

Fig. 4. Signalfluss des Es-Chords

Der Case Picker ist wie folgt aufgebaut:

```
case1 = ba.if(sh_abs>=(0),case1_1,0);
case1_1 = ba.if(sh_abs <(0.16),1,0);
case2 = ba.if(sh_abs>=(0.16) ,case2_1,0);
case2_1 = ba.if(sh_abs<(0.29),1,0);
case3 = ba.if(sh_abs>=(0.29),case3_1,0);
case3_1 = ba.if(sh_abs <(0.39),1,0);
```

Wird der S&H getriggered, gibt er einen absoluten Wert zwischen 0 und 1 aus. Da die Verteilung der Werte des FAUST-S&H gaußähnlich ist, müssen die Wertebereiche dementsprechend angepasst werden. Deshalb werden die Bereiche, in die ein Wert fallen muss, um einen zugehörigen Case zu triggern, zur Mitte hin exponentiell kleiner. Über diese Anpassungen ist es möglich, die Funk-

tion der Source of Uncertainty nachchzubilden. Die Logik für case1 lautet: Wenn sh_abs größer als 0 ist, nimm den Wert von case1_1 an. Case1_1 hat nur einen Wert ungleich 0, nämlich 1, wenn sh_abs einen Wert zwischen 0 und 0,16 ausgegeben hat. In diesem Fall wäre case1 = 1 und triggert so eine Case spezifische Begleitung, hier die erste Harmonische.

```
firstharm =  case1 * sy.combString(f0*2,rtc,triggerE);
```

Um den Synthesizer zu spielen wird der Master-VCA über einen Attack/Release- Envelope geöffnet, welcher wiederum durch ein externes Gate aktiviert wird. Die Grundfrequenz kann ebenfalls extern vorgegeben werden, beispielsweise mithilfe eines Keyboards oder Sequenzers. Über zwei verschiedene Clocks wird zum einen die Wiederholung der Begleit-Intervalle (Chord-Rate) eingestellt und die zweite Clock (SH Rate), bestimmt die Taktung der SH-Schaltung. So ist es möglich die Geschwindigkeit der Auswahl der Begleit-Intervalle unabhängig ihrer Spielgeschwindigkeit einzustellen.

3 Implementierung als Eurorack-Modul

Als Basis für die Hardware-Implementierung des Algorithmus diente ein Bela-Audioprozessor. Das Bela-Board basiert auf einem BeagleBone Black, dessen Ein- und Ausgänge durch ein Bela-Cape Adapter zugänglich gemacht werden. Das Bela-Board besitzt eine sehr geringe Latenz und ist somit optimal für verschiedene Audio-Implementierungen geeignet.

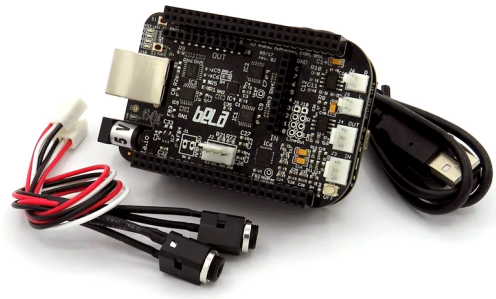

Fig. 5. Bela-Board inklusive Audio-Adapter

Die eigene IDE, welche direkt vom Gerät aus aufgerufen werden kann, ermöglicht es, das Bela-Board mit vielen verschiedenen Programmiersprachen schnell und unkompliziert zu programmieren. Somit können etwa Pure-Data Patches oder SuperCollider-Programme in wenigen Minuten auf das Bela-Board gespielt werden. Auch die für dieses Projekt verwendete FAUST-IDE ermöglichte durch eine eigene ”Export-to-Bela”- Funktion, eine einfache Möglichkeit den Algorithmus auf das externe Bela-Board zu spielen. Um das Bela-Board als ein Eurorack-Module zu verwenden, wurde für dieses Projekt ein weiterer Bela-Adapter benötigt, der Bela Pepper. Pepper ist ein DIY-Eurorack Modul mit einer Breite von 18HP, an dessen Rückseite das Bela-Board gesteckt werden kann. Somit werden die Ein- und Ausgänge des Bela-Boards auf Eurorack verträgliche Spannungen verstärkt und die Algorithmen können mithilfe von 8 Potentiometern und 4 Tastern gesteuert werden. Da es sich hier um ein DIY-Modul handelt, muss der Bela-Pepper selbst zusammengebaut und gelötet werden. Eine bereits fertig montierte Variante ist unter dem Namen Bela-Salt zu finden [5].

Fig. 6. Bela-Pepper Eurorack Modul

Die 8 Potentiometer des Bela Peppers wurden für dieses Projekt wie in Abb. 7 gezeigt, belegt.

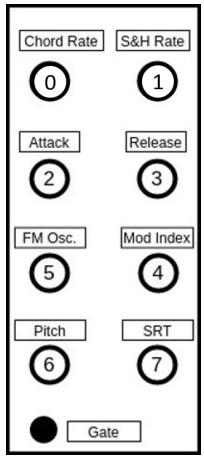

Fig. 7. Potentiometerbelegung des Bela-Pepper

Der Es-Chord kann manuell über den Gate Taster des Bela-Pepper getriggert werden. Der erste Potentiometer (Chord-Rate) gibt bei einem offenem Gate an, wie schnell die generierten Oberton-Intervalle gespielt werden. Über den zweite Potentiometer (SH-Rate) kann nun eingestellt werden, wie schnell diese Oberton-Intervalle neu gewählt werden. Die beiden nächsten Regler stellen den internen AR-Envelope ein. Um die Klänge noch weiter zu variieren, kann über die Regler FM-OSC und Mod-Index eine Frequenzmodulation mithilfe eines internen Sinus-Oscillators hinzugefügt werden. Der Pitch-Regler steuert die Tonhöhe der Fundamentalfrequenz und kann auch über Steuerspannungen geregelt werden. Der letzte Regler steuert die interne Nachhallzeit des StringModelling und kann als zusätzlicher Decay-Regler verwendet werden.

Eine Demonstration des fertigen Moduls ist als Video im Anhang zu finden.

4 Zusammenfassung

Die Arbeit am Es-Chord for Bela zeigt, dass es mit der Programmiersprache FAUST möglich ist, bewährte Prinzipien aus der analogen Synthese digital zu implementieren. Die Realisierung als Eurorack-Modul auf Basis des Bela Pepper war ohne große Komplikationen möglich und erweitert die Möglichkeiten der FAUST-Sprache enorm, da konzipierte Algorithmen nun nahtlos in ein Eurorack-System integriert werden können. Es-Chord for Bela entwickelte sich somit von einer ersten Idee in der FAUST-IDE, in ein vollwertiges Instrument, welches in Zukunft je nach Anwendungsgebiet problemlos erweitert und angepasst werden kann. Der zugehörige Code ist als Github-Repository im Anhang verlinkt .

5 REFERENCES

[1] "Karplus-Strong Algorithm," https://cnx.org/contents/ahcuwm3R@1.1:RRcC1ZNa@2/Karplus-Strong-Algorithm, zuletzt aufgerufen am 05. April 2021.

[2] T. Tolonen, V. Välimäki, M. Karjalainen, "Evaluation of Modern Sound Synthesis Methods," WorkingPaper 48, Helsinki University of Technology, Laboratory of Acoustics and Audio Signal Processing (1998).

[3] A. Strange, *Electronic Music* (W.C. Brown Co., Dubuque, Iowa) (1972).

[4] K. Bjørn, C. Meyer, *Patch & Tweak* (BJOOKS, Frederiksberg, Denmark) (2018).

[5] "Bela-Homepage," https://bela.io, zuletzt aufgerufen am 05. April 2021.

Anhang

GitHub:
https://github.com/philippsteigerwald/Es-Chord-for-Bela
Demonstrations-Video:
https://www.youtube.com/watch?v=H53izxFohtc