Fakultät Informatik, Mathematik und
Naturwissenschaften
Studiengang Informatik Master

Projektarbeit zur Vorlesung Computermusik

BrandtBrauerFrick.hs

Autoren: Nico Mehlhose, Raphael Drechsler
Abgabedatum: 03.02.2019

INHALTSVERZEICHNIS

1 ABSTRACT

Abschnitt bearbeitet von: Raphael Drechsler

BrandtBrauerFrick.hs
Brandt Brauer Frick ist ein Techno-Projekt aus Berlin. Die Basis des Projekts bilden Klänge aus dem Instrumentarium der klassischen Musik, welche anfangs gesampelt, später in einem zehnköpfigen Ensemble auch live vorgeführt wurden.[1]

Ziel des Projektes:
Die Umsetzung des Songs "Pretend" von Brandt Brauer Frick entweder in Tidal oder Euterpea. Eine online verfügbare Live-Aufführung [2] soll dabei als Referenz dienen. Bei der Umsetzung soll auch Wert auf die Nachbildung der echten Instrumente und deren teilweise Zweckentfremdung gelegt werden.

Herausforderungen:

- Evaluation ob Tidal[3] oder Euterpea[4] genutzt werden soll:
- Untersuchung der Frage ob klassische Klänge am ehesten in Euterpea oder Tidal nutzbar sind. (Durch repetitiven Charakter des Liedes würde sich Tidal zur Live-Vorführung eignen)
- Analyse der einzelnen musikalischen Bausteine und deren Implementierung.
- Zusammenfügen der erarbeiteten Bausteine zu einer Performance.

2 UMSETZUNG IN TIDAL ODER EUTERPEA

Abschnitt bearbeitet von: Nico Mehlhose

Dieses Thema soll sich um die Evaluation zwischen Tidal und Eutherpea handeln. Bevor es zu der Evaluation kommt, werden die eben genannten Programme kurz erklärt.
Tidal ist eine Open Source Software mit deren Hilfe es möglich ist Musikpatterns per Code zu generieren. Tidal benutzt den Synthesizer *SuperCollider*.[3]
Der SuperCollider, ebenfalls eine Open Source Software, ist ein Audio-Synthesizer. Dabei können eigene Klänge oder Instrumente mithilfe von unterschiedlichen Oszilatoren, Filtern und anderen Hilfsmitteln des SuperColliders erstellt werden. [5]
Euterpea ist ein in Haskell eingebettetes Programm, welches durch Code u. a. Musik erzeugt, algorithmische Komposition und Sound Synthese ermöglicht.[4]
Im Endeffekt ist unsere Auswahl des Programmes auf Tidal gefallen. In diese Entscheidung ist der Programmieraufwand, vorhandenen Informationen und die Möglichkeit den Synthesizer zu erweitern mit eingeflossen.
Bei dem Programmieraufwand wird sehr schnell klar, dass das sehr repetetive Lied *Pretent* von BrandBrauerFrick für Tidal besser geeignet ist als für Euterpea, da in Tidal die Musikpatterns immer in einem loop abgespielt werden. In Eutherpea kann dieser Effekt nur durch zusätzlichen Programmieraufwand erreicht werden.
Bei den vorhandenen Informationen zu dem Musikstück stellt sich heraus, dass keine offiziellen Notenblätter für das Lied Online existieren, wodurch Euterpea etwas an Bedeutung verliert, da Euterpea für genaue Notenbestimmungen perfekt geeignet wäre. Durch die absenz dieses Faktors, kann das selbe Maß an Genauigkeit auch mit Tidal erreicht werden.
Der Letzte und für uns wichtigste Punkt war die Erweiterbarkeit der Sounds. Die Wichtigkeit darin besteht in der entfremdeten Benutzung der Musikinstrumente in dem Lied. In Eutherpea haben wir nach einiger Recherche keinen weg gefunden

Sounds hinzuzufügen um diese später zu verwenden. In Tidal existiert diese Möglichkeit mittels dem Befehl `~dirt.loadSoundFiles("full/path/to/directory/*")`. Mit diesem Befehl lässt sich ein Verzeichnis in Tidal integrieren. Anschließend können die Sounds beim Programmieren mit dem Ordnernamen benutzt werden.[6]

3 EIGENSCHAFTEN DES STÜCKS PRETEND UND DESSEN GLOBALE STRUKTUR

Abschnitt bearbeitet von: Raphael Drechsler

Die in der Live vorgeführte Version [2] hat eine ungefähre Dauer von 7 Minuten, 15 Sekunden. Die Angabe erfolgt ungefähr, da die Aufnahme nicht mit dem ersten Takt beginnt
Per Gehör ließ sich feststellen, dass das Stück in der Tonart Gm steht.
Über ein BPM-Measuring-Tool [7] wurde ein Tempo von 130bpm ermittelt. In Tidal wird somit der folgende Code zur Tempo-Einstellung benötigt.

```
setcps (130/60/4)
```

Die Globale Struktur des Liedes, also die Zeitliche Abfolge der Figuren der einzelnen Instrumente, wurde per Gehör analysiert. Dabei wurde ebenfalls die Live-Version des Liedes als Untersuchungsgegenstand verwendet. Um das Ergebnis zu visualisieren, wurden in Logic Pro[8] (einer digital Audio-Workstation der Firma Apple) für die jeweiligen Figuren leere MIDI-Regionen innerhalb der 237 Takte erzeugt. Anschließend wurde das Resultat per Screenshot aufgenommen und die einzelnen Figuren mit F1,F2,... für die jeweilige Figur beschriftet. Zur besseren Verständigung darüber, wo man sich innerhalb der globalen Struktur befindet wurde das Lied in 10 Parts unterteilt. Diese wurden mit P1,P2,...,P10 beschriftet.
Für die ersten vier Takte wurde bei der Erstellung der MIDI-Regionen eine Annahme getroffen.

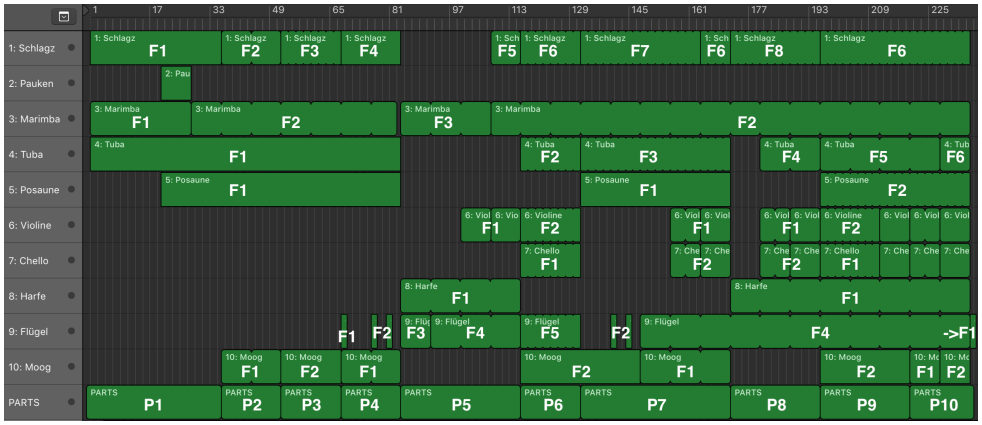

Abbildung 1: Globale Struktur dargestellt als leere MIDI-Regionen in Logic Pro mit Beschriftung der Figuren

Es wurde ebenfalls versucht die Abfolge mithilfe eines freien Notations-Programmes in einer Partitur darzustellen. Jedoch erwies sich die obige Darstellung als kompakter und ausreichend.

4 ANALYSE UND SYNTHESE DER EINZELNEN INSTRUMENTE

Abschnitt bearbeitet von: Raphael Drechsler

Im folgenden Abschnitt sollen neun der zehn Instrumente synthetisiert werden. Da die Pauken auf einer ähnlichen Frequenz klingen wie die Bass-Drum und Moog-Synthesizer sind diese nur sehr schwer aus dem Stück herauszuhören. Durch diesen Umstand und mit der Absicht das fertig synthetisierte Stück nicht zu überladen wurde daher beschlossen auf Analyse und Synthese der Pauken zu verzichten.

Wie in der globalen Struktur (siehe Abb. 1) zu erkennen ist, existieren in den meisten Fällen pro Instrument mehrere Figuren. Die folgenden Arbeitsschritte sollen daher pro Instrument und Figur erfolgen.

ANALYSE DER GESPIELTEN TONHÖHEN UND RHYTHMEN. Diese Analyse erfolgt per Gehör. Untersucht wird dabei die Live-Version[2] des Stückes. Für Figuren, die besonders schwierig herauszuhören sind, da z.B. das Instrument nur sehr schwer hörbar ist, werden zusätzlich eine ähnliche Studio-Version[9] sowie eine früher Variante[10] des Liedes für die Untersuchung herangezogen. Das Ergebnis der Analyse soll hier als Text, welcher die Figur beschreibt und/oder mithilfe von Noten dargestellt werden.

SYNTHESE VON TONHÖHE UND RHYTHMUS Die analysierten Tonhöhen und Rhythmen sollen als Tidal Code umgesetzt werden. Dabei wird kein gesteigerter Wert auf die Auswahl eines passenden Klanges gelegt. Im Vordergrund der Betrachtung steht, dass der umgesetzte Code den analysierten Tonhöhen und Rhythmen entspricht. Vereinzelt soll hierbei auf zu überwindende Herausforderungen und genutzte Funktionen eingegangen werden.

KLANGANALYSE Per Gehör soll das Klangbild des Instrumentes in der speziellen Figur sowie die Wirkung, welche durch die Figur beim Hörer erzeugt wird, untersucht werden.

ANPASSUNG DER KLANGSYNTHESE Unter Berücksichtigung des analysierten Klangbildes und des bereits vorliegenden Tidal-Codes soll nun die Art der Klangsynthese derart angepasst werden, dass die beschriebene klangliche Wirkung erzielt wird. Mögliche Arten der Anpassung sind dabei die Auswahl von Tidal-Instrumenten, Einbinden von fremden und selbst aufgenommenen Samples sowie das Erstellen eines SuperCollider-Instrumentes.

Auf diesem Wege sollen die Figuren aller Instrumente als ausführbarer Tidal-Code mit erwünschter klanglicher Wirkung entstehen. Das Verbinden der einzelnen Figuren zu einem live vorführbaren Stück soll in *Kapitel 5 - Performance* beschrieben werden.

4.1 Instrument 1: Schlagzeug

4.1.1 *Figuren*

Abschnitt bearbeitet von: Raphael Drechsler

Figur 1
Herausgehört wurde das folgende Muster. Die Note *A* steht dabei für die Bass-Drum, Note *Dis* für die High-Hat.

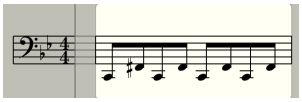

Abbildung 2: Schlagzeug Figur 1

Um in Tidal pro Takt eine Figur mit 4 Schlägen auf die Bass-Drum und 4 Schlägen auf die High-Hat im Wechsel zu realisieren, lässt sich eine Kombination von Gruppierung und Wiederholung[11] verenden:

```
d1 $ sound "[bd hh]*4"
```

Figur 2
Herausgehört wurde eine Figur über 8 Takte. Dabei werden in Takt 3,7 und 8 wie nachfolgend notiert Fills auf der High-Hat gespielt.

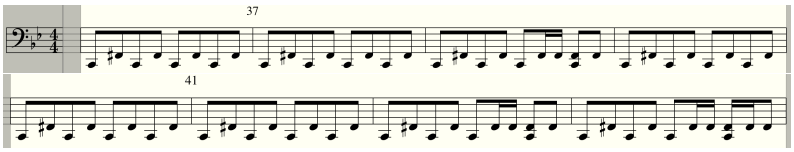

Abbildung 3: Schlagzeug Figur 2

Die Umsetzung in Tidal der einzelnen Takte ohne Fills erfolgt analog zu Figur 1. In den Takten mit Fills werden auf einige Zählzeiten Bass-Drum und High-Hat gleichzeitig gespielt. Dies kann in Tidal durch die Nutzung von `stack` [12] umgesetzt werden. Über den `cat`-Ausdruck[13] werden die acht Figuren hintereinander gespielt, was den Cycle auf die gewünschte Länge von acht Takten verlängert. Es ergibt sich folgender Code.

```
d1 $ cat [
sound "[[bd hh]*4]",
sound "[[bd hh]*4]",
stack [ sound "[bd ~]*4", sound "[~ hh ~ hh][~ [hh hh] hh hh]" ],
sound "[[bd hh]*4]",
sound "[[bd hh]*4]",
sound "[[bd hh]*4]",
stack [ sound "[bd ~]*4", sound "[~ hh ~ hh][~ [hh hh] hh hh]" ],
stack [ sound "[bd ~]*4", sound "[~ hh ~ hh][~ [hh hh] [hh hh] hh]" ]
]
```

Figur 3
Herausgehört wurde die folgende Figur mit zwei Takten Länge. Die Note *Fes* steht dabei für eine geöffnete High-Hat.

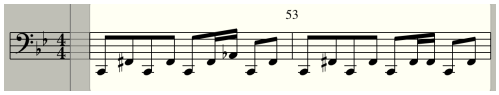

Abbildung 4: Schlagzeug Figur 1

In Tidal wurde die Figur hintereinander in Gruppierungen geschrieben und per `slow 2`[14] auf die Länge von zwei Takten gestreckt.

```
d1 $ slow 2 $ sound "[bd hh bd hh][bd [hh ho] bd hh] [bd hh bd hh][bd [hh hh]
    bd hh]"
```

Figur 4
Analog zu Figur 1. Dazu kommen zyklische Bewegung auf der Rim (Rand der Snare-Drum). Die Figur ließ sich nur schwer durch Heraushören bestimmen. Es wurden daher 5 Schläge auf die Rim pro Takt als Annahme getroffen, wobei aller 2 Takte der letzte Schlag ausgelassen wird.

Dies realisiert der folgende Tidal-Code.

```
d1 $ stack [
sound "[bd hh]*4",
cat[sound "[rm rm rm rm rm]", sound "[rm rm rm rm ~]"]
]
```

Figur 5
Nur die zyklische Rimclick-Bewegung aus Figur 4.

```
d1 $ cat[sound "[rm rm rm rm rm]", sound "[rm rm rm rm ~]"]
```

Figur 6 und 7
Figur 6 wie Figur 4 und Figur 7 wie Figur 3. Jeweils kräftiger gespielt. Dies wird später in der Performance mit einem `# gain`-Ausdruck[15] zum Regulieren der Lautstärke des gespielten Samples umgesetzt.

Figur 8
Wie Figur 4 aber ohne Bass-Drum.

```
d1 $ stack [
sound "[~ hh]*4" ,
cat[sound "[rm rm rm rm rm]", sound "[rm rm rm rm ~]"]
]
```

4.1.2 *Klangbild*

Abschnitt bearbeitet von: Nico Mehlhose

Art der Synthetisierung: Da Bass Drum und High Hat normal gespielt werden können die Sounds aus dem Supercollider mit minimaler Anpassung benutzt werden. Lediglich die Rim aus Figur 4 muss, wegen ihres hölzernen Sounds, selbst aufgenommen werden.

Figur 1
Sound BD: Die Bass Drum wird mit zunehmender Dauer lauter gespielt.
Sound HH: Auch die HH wird Anfangs nur sanft angespielt aber mit zunehmender Zeit lauter.
Problem: Bass Drum und High Hat müssen mit zunehmender Vorführungszeit lauter werden.
Lösung: Die Lautstärkensteigerung der Schlagzeugs wird live, mittels der Erhöhung des *gain*-Parameters, realisiert.

```
d1 $ sound "[bd hh]*4" #midinote 58 #gain 0.9
```

Figur 2
Sound: Das Schlagzeug wird in dieser Figur, bis auf die Lautstärkensteigerung, analog zu Figur 1 gespielt. In dieser Figur ist die Lautstärke gleichbleibend.
Figur 3
Sound: Die Bass-Drum wird etwas schwächer angespielt um einen zu starken Bass zu vermeiden und um nicht dominant zu wirken. Selbiges gilt für die High Hat.

Dies wird in der Performance berücksichtigt. **Figur 4**
Sound: In Figur 4 wurde eine hölzern klingende Rim hinzugefügt. Diese ist nur sehr dezent im Klangbild zu erkennen. Lösung: Die hölzern klingende Rim wurde durch einen Schlag eines Bleistiftes auf eine kleine hölzerne Kiste erzeugt. `room` soll der Rim noch einen hallenden Charakter geben, da der aufgenommene Sound diesen nicht besessen hat.

```
d1 $ stack [
sound "[bd hh]*4" #midinote 58 #gain 1.0,
cat[
  sound "[Rim Rim Rim Rim Rim]",
  sound "[Rim Rim Rim Rim ~]"
  ] #room 0.15 #orbit 1 #gain 1.5
]
```

Figur 5
Sound: In dieser Figur wird lediglich die hölzerne Rim gespielt. Die Rim wird analog zu Figur 4 gespielt.
Figur 6 und 7
Sound: Figur 6 wird wie Figur 4 gespielt und Figur 7 wie Figur 3. In beiden Fällen sind 6 und 7 nur lauter gespielt, was über den `Gain`-Befehl realisiert wird.
Figur 8
Sound: Die letzte Figur des Schlagzeugs ist für "die Ruhe vor dem Sturm"zuständig, welche kurz vor dem zweiten Hauptteil zu hören ist. Dabei sieht diese Figur so aus wie Figur 4, jedoch fehlt die Bass Drum was für die "Ruhe"sorgt.
Lösung:

```
d1 $ stack [
  sound "[~ hh]*4" #midinote 58 #gain 1.3,
  cat[
    sound "[Rim Rim Rim Rim Rim]",
    sound "[Rim Rim Rim Rim ~]"] #gain 1.0
]
```

4.2 Instrument 2: Pauke

Wird, wie oben erwähnt, nicht synthetisiert.

4.3 Instrument 3: Marimba

4.3.1 *Figuren*

Abschnitt bearbeitet von: Raphael Drechsler

Figur 1
Die Tonhöhe war per Gehör nicht genau differenzierbar. Es wurde folgende Annahme getroffen.

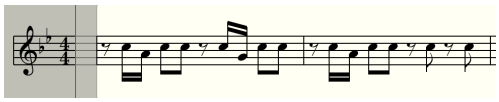

Abbildung 5: Marimba Figur 1

Die Tonhöhe des Sounds `glasstrap` wurde mithilfe des `midinote`-Ausdrucks [16] auf die jeweils gewünschte Tonhöhe geändert.

```
d1 $ slow 2 $ midinote "[~ [51 49] 51 51][~ [51 46] 51 51][~ [51 49] 51 51][~
    51 ~ 51]" # s "glasstap"
```

Figur 2
In der Live-Version des Liedes ist im Video zu erkennen, dass auf der Marimba Kuhglocken (8 oder 10 Stück) in verschiedenen Tonhöhen liegen. Diese werden in der zweiten Figur auf die in der folgenden Abbildung gezeigten Zählzeit angespielt. Dabei wird Immer eine andere Tonhöhe gespielt. Das Muster in welcher Reihenfolge die Glocken gespielt werden wurde nicht weiter analysiert. Stattdessen soll das Anspielen der Glocken zur Vereinfachung randomisiert werden.

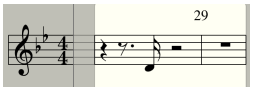

Abbildung 6: Marimba Figur 2

Der Ausdruck `irand 10` [17] liefert einen zufälligen int-Wert von 0 bis einschließlich 9 zurück. Dieser Wert verändert in Verbindung mit dem `# speed`-Ausdruck[18] die Abspielgeschwindigkeit des `can`-Samples und damit dessen Tonhöhe. So werden 9 verschiedene Kuhglocken simuliert.

```
d1 $ stack [
slow 2 $ midinote "[~ [51 49] 51 51][~ [51 46] 51 51][~ [51 49] 51 51][~ 51 ~
    51]" # s "glasstap",
slow 2 $ midinote "[~[~~~60]~~][]" # s "can" # speed (1 + (irand 10)*0.2)
]
```

Figur 3
Es wurde folgende Figur per Gehör ermittelt. Dabei sind die Tonhöhen wie in Figur 1 angenommen.

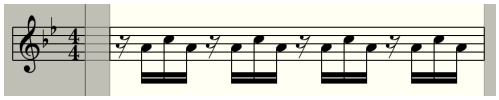

Abbildung 7: Marimba Figur 3

Umsetzung in Tidal:

```
d1 $ midinote "[~ 49 51 49]*4" # s "glasstap"
```

4.3.2 *Klangbild*

Abschnitt bearbeitet von: Nico Mehlhose

Bestandteile der Marimba: Eine Marimba, wobei Kuhglocken zum zufälligem Anspielen enthalten sind.
Art der Synthetisierung: Da keine hölzernen Klänge im SuperCollider enthalten sind müssen diese selbst erstellt werden. Dies geschieht mittels einem von *SuperCollider Code* stammenden Code.[19]
Folgender Code wurde leicht abgewandelt, sodass er unseren Anforderungen entspricht:

```
(
SynthDef(\bell, {
  |fs=1, t60=1, pitchy=1, amp=0.25, gate=1|
  var sig, exciter;
  //exciter = Impulse.ar(0);
```

```
  exciter = WhiteNoise.ar() * EnvGen.ar(Env.perc(0.001, 0.05), gate) * 0.25;
  sig = Klank.ar(
    `[
      [1, 2, 2.803, 3.871, 5.074, 7.81, 10.948, 14.421], // freqs
      [1, 0.044, 0.891, 0.0891, 0.794, 0.1, 0.281, 0.079], // amplitudes
      [1, 0.205, 1, 0.196, 0.339, 0.047, 0.058, 0.047]*t60 // ring times
    ],
    exciter,
    freqscale:fs*pitchy);
  sig = FreeVerb.ar(sig) * amp;
  DetectSilence.ar(sig, 0.001, 0.5, doneAction:2);
  Out.ar(0, sig!2);
}).add
)


(
Pbind(
  \instrument, \bell,
  \fs, 60.midicps,
  \t60, 0.45,
  \pitchy, 1.75,
  \dur, 0.25
).play;)
```

Wobei der erste Teil des Codes einen Ton erzeugt welcher Vergleichbar ist mit dem Kratzen auf einem Mikrofon.
Der zweite Teil ändert den ersten Sound so ab, sodass dieser mehr hölzern klingt.
Als nächstes muss dieser Sound aus dem SuperCollider aufgenommen werden. Dies wird mit dem Befehl *Server.default.record;* umgesetzt.[6]
Der dritte Schritt war das Zuschneiden der Audisamples. Dazu wurde Audacity benutzt. Audacity ist ein kostenlosen Audioschnitt-, Editierungs- und Aufnahmeprogramm.[20]
Im Anschluss kann dann der Code für die einzelnen Figuren angepasst werden.

Figur 1
Sound : Die Marimba wird am Anfang des Stücks relativ dominant, im Gegensatz zu den restlichen Instrumenten, gespielt.Dieser Effekt entsteht durch den `Gain`-Opperator in der Schlagzeugfigur 1.
Lösung:

```
d1 $ slow 2 $ midinote "[~ [65 63] 65 65][~ [65 60] 65 65][~ [65 63] 65 65][~
    65 ~ 65]" # s "Marimba" # room 0.1 #orbit 1
```

Figur 2
Sound: In dieser Figur wird die Marimba eher rythmisch als melodisch eingesetzt. Hinzu kommt das die Marimba nur schwach angespielt wird, wodurch sie in diesem Teil in den Hintergrund tritt. Des weiteren liegen neben der Marimba noch Kuh-Glocken. Die Kuh-Glocken werden zufällig angespielt.
Figur 3
Sound: In der dritten Figur ist die Marimba nur sehr dezent im Hintergrund zu hören. Sie dient in dieser Figur lediglich als Taktgeber.

4.4 Instrument 4: Tuba

4.4.1 *Figuren*

Abschnitt bearbeitet von: Raphael Drechsler

Figur 1
Es wurde eine Figur über einen Takt ermittelt, in der ein Schlag auf das Mundstück der Tube als rhythmisches Element auf zweite Zählzeit im Takt erfolgt.

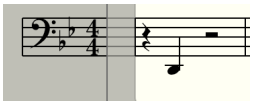

Abbildung 8: Tuba Figur 1

```
d1 $ sound "[~ sn ~ ~]"
```

4.4.2 *Figuren*

Figur 2
Es wurde eine Figur über 2 Takte ermittelt. Instrumentalist bläst in dieser Figur in die Tuba ohne dass die Lippen vibrieren, um ein Rauschen zu erzeugen. Am Ende der Figur wurde eine Pause als Atempause angenommen.

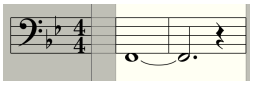

Abbildung 9: Tuba Figur 2

Für die Umsetzung wird ein Sample benötigt, welches über die Dauer von zwei Takten reicht. Um dieses nur aller 2 Takte abzuspielen damit sich mehre Instanzen des Samples nicht überlagern kann der Ausdruck `loopAt 2` [21] verwendet werden.

```
d1 $ loopAt 2 $ sound "blasesoundTuba"
```

Figur 3
Analyse-Ergebnis: Wie Figur 1, hier allerdings kurzes tonloses Pusten stoßweise gespielt anstelle von Schlag auf Mundstück. Code analog zu Figur 1.

Figur 4
Analyse-Ergebnis: Tiefe Töne durch Tuba. Die Tonhöhe ist nahezu nicht erkennbar. Die Tonhöhe wurde jedoch durch die Analyse in Figur 6 ableitbar. Die gespielten Töne werden über den Verlauf von zwei Durchläufen der Figur langsam lauter.

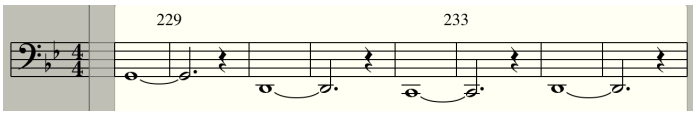

Abbildung 10: Tuba Figur 4

Das lauter Werden der Töne über die Dauer von zwei Durchläufe der Figur wurde mit einem Pattern realisiert, welches dem `# gain`-Effekt übergeben wurde.[22] Um die lang ausgehaltenen Noten umzusetzen, wurde auf den Ausdruck ein Hall angewendet, wobei `# room` die Lautstärke des Halls und `# sz` die Größe des hallenden Raumes bedingt. `# orbit 1` wird genutzt, um die ansonsten globale Wirkung des Hall-Effektes auf den vorangegangenen Ausdruck zu beschränken. [23]

```
d1 $ slow 16 $ midinote "[43 38 36 38]*2" # s "superpiano" # room 0.85 # sz 0.8
    # orbit 1 #gain "0.4 0.45 0.5 0.55 0.6 0.7 0.8 0.9"
```

Figur 5
Herausgehört wurde eine Figur analog zu Figur 4, jedoch kräftig gespielt.

Figur 6
Herausgehört wurde eine Figur analog zu Figur 5, jedoch maximal kraftvoll ausgespielt. Dabei werden die Töne jeweils eine Oktave höher gespielt, was die Tonhöhe der einzelnen Töne gut erkennbar macht.

Umsetzung in Tidal:

```
d1 $ slow 8 $ midinote "55 50 48 50" # s "superpiano" # room 0.85 # sz 0.8 #
    orbit 1
```

4.4.3 *Klangbild*

Abschnitt bearbeitet von: Nico Mehlhose
Bestandteile der Tuba: Eine Tuba, welche entartet, wie in den folgenden Figuren beschrieben, benutzt wird.
Art der Synthetisierung: Da die Tuba ein reales Musikinstrument ist, welches in dieser Form nicht im Supercollider enthalten ist, werden hierfür selbst aufgenommene Samples benutzt. Die Samples werden für die entartete Benutzung in Tidal so manipuliert, dass sie die Sounds nachempfinden. Des weiteren wurden Onlinesamples für die Figuren 4-6 benutzt.[25]

Figur 1
Sound: In dieser Figur wird auf das Mundstück der Tuba geschlagen. Der erzeugte Ton hört sich in etwa an wie eine Bass Drum ohne Bass.
Lösung: Um den eben beschriebenen Ton zu erzeugen, wurde die Bass Drum in Tidal mit `Midinote` langgezogen. Damit nun dieser langgezogene Sound nicht über den gesamten Loop spielt, wurde eine High Hat mit `Gain 0` eingefügt damit die HH keinen Sound macht. Anschließend muss der Ton der BD mit `cut 1` geschnitten werden damit dieser bei der stummen HH aufhört zu spielen.

```
d1 $ sound "~ bd hh ~" # cut 1 # midinote 15 #gain "1.3 0"
```

Figur 2
Sound: Bei Figur 2 ist das Klangbild vergleichbar eines Reifens der Luft verliert. Der Sound wurde selbst aufgenommen. Um den eben beschriebenen Ton nachzustellen wurde mit weit gespreizten Nasenflügeln ausgeatmet.
Lösung:Der `room` sorgt dafür, dass sich der Sound nach einem größerem Gegenstand und nicht nach einer Nase anhört.

```
d1 $ loopAt 2 $ sound "Atmen" #room 1 #orbit 1 # gain 1.3
```

Figur 3
Sound: Der Sound ist wie aus Figur 2 aber er wird mit mehr Druck und nicht durchgehend gespielt. Erstellt wurde der Sound durch das anspielen von einem Handscheibenwischer mit hohlem Griff.
Lösung: `Room` und `Gain` haben hier den selben Effekt wie in Figur 2

```
d1 $ sound "[~ Tuba ~ ~]" # room 0.4 #orbit 1 # gain 1.1
```

Figur 4, 5, 6
Sound: Die Tuba wird in Figur 4 mit absteigender Lautstärke gespielt. In Figur 5 und 6 wird sie mit gleichbleibender Lautstärke gespielt wobei jedoch die Lautstärke zwischen den Figuren ansteigt.
Lösung Figur 4:

```
d1 $ slow 16 $ midinote "[55 50 48 50]*2" # s "TubaOrch:20" # cut 1 #gain " 1
    0.95 0.9 0.85 0.8 0.7 0.7 0.7"
```

Lösungen für Figuren 5 und 6 sind oben zu sehen.

4.5 Instrument 5: Posaune

4.5.1 *Figuren*

Abschnitt bearbeitet von: Raphael Drechsler

Figur 1
Herausgehört wurde eine Figur über einen Takt. In der Figur erfolgt ein kurzes, tonloses Pusten in die Posaune. Dieses wird Stoßweise gespielt und als rhythmisches Element auf letzte Achtelnote im Takt eingesetzt.

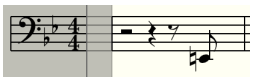

Abbildung 11: Posaune Figur 1

Umsetzung in Tidal:

```
d1 $ sound "[][[][~ sn]]"
```

Figur 2
Zu hören ist, dass die Figur ein Rauschen analog zur Figur 2 der Tuba umfasst. Der Code gestaltet sich entsprechend.

```
d1 $ loopAt 2 $ sound "blasesoundPosaune"
```

4.5.2 *Klangbild*

Abschnitt bearbeitet von: Nico Mehlhose

Bestandteile der Posaune: Gewöhnliche Posaune, welche aber entartet benutzt wird. Art der Synthetisierung: Da die Posaune ein reales Musikinstrument ist, welches in dieser Form nicht im Supercollider enthalten ist, werden hierfür Samples benutzt. Die Samples werden eigens dafür aufgenommen. Die Sound werden wie bei der Tuba erzeugt.

Figur 1
Sound: In dieser Figur wird auf das Mundstück der Posaune stoßartig angespielt.
Lösung:`room` bzw. `gain` werden hier benutzt damit die Sounds sich nach einem größeren Hohlkörper und nicht nach einem Scheibenwischer anhören.

```
d1 $ sound "[][[][~ Posaune]]" # room 0.3 #orbit 1 # gain 1.2
```

Figur 2
Sound: Ähnlich zu Figur 3 der Tuba allerdings ist dieser Sound weniger tief.
Lösung: Für die geringere Tiefe des Sounds wurde der Hall der Posaune angepasst.

```
d1 $ loopAt 2 $ sound "Atmen" #room 0.6 #orbit 1 #gain 1.1
```

4.6 Instrument 6: Violine

4.6.1 *Figuren*

Abschnitt bearbeitet von: Raphael Drechsler

Figur 1
Herausgehört wurde die folgende Figur über 16 Takte.

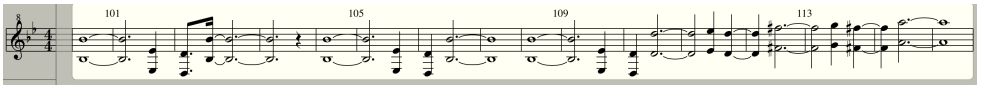

Abbildung 12: Violine Figur 1

Die Noten der Figur werden oktaviert gespielt. Der Einfachheit halber wurden nur die gut hörbaren hohen töne umgesetzt. Um jeweils einen Ton zu einem Zeitpunkt zu hören, werden die abgespielten Samples mithilfe des Ausdrucks `# cut 1` immer dann gestoppt, sobald sie für den nächsten Ton abgespielt werden.[24]

```
d1 $ slow 4 $ cat [
midinote "[82][][][][][][][][75][74][82][][][][][][][]" # s "gtr",
midinote "[82][][][][][][][][75][74][82][][][][][][][]" # s "gtr",
midinote "[82][][][][][][][][75][74][86][][][][][][87][86]" # s "gtr",
midinote "[][90][][][][][][91][90][][93][][][][][][][]" # s "gtr"
] #cut 1 # room 0.85 # sz 0.8 # orbit 1
```

Figur 2
Analyse-Ergebnis:

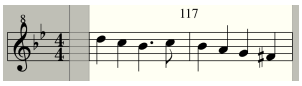

Abbildung 13: Violine Figur 2

Umsetzung in Tidal:

```
d1 $ slow 2 $ midinote "[[86][84][82][[][84]]][82 81 79 78]" # s "gtr" #cut 1 #
    room 0.85 # sz 0.8 # orbit 1
```

Weitere Figuren
Im hinteren Teil des Stückes treten weitere Figuren der Violine auf. Da deren Umsetzung jedoch keinen signifikanten Unterschied in der Performance ausmacht wurde auf die Analyse und Synthese dieser Figuren verzichtet.

4.6.2 *Klangbild*

Abschnitt bearbeitet von: Nico Mehlhose

Art der Synthetisierung: Hierfür wurden Samples Online gesucht da wir möglichst viele verschiedene Arten der Soundeinbindung abdecken wollten. Dabei wurde ein großer Datenbestand von Orchesterinstumenten gefunden.[25] Des weiteren wäre es mit zusätzlichem Zeitaufwand verbunden gewesen jedes Instrument über den SuperCollider zu synthetisieren, weshalb Sound Samples für dieses Instrument gewählt wurden.

Figur 1
Sound: In dieser Figur steigt die Violine leise in das Stück ein und wird über die Zeit immer lauter, wodurch sie schlussendlich im zweiten Teil zum Hauptinstrument des

Abschnittes wird. Im ersten Teil wird die Violine etwas quitschend und langsam gespielt. Teil zwei besteht aus einem hektisch aber sauber gespielt Part.
Problem: Die Violine muss in unserer Vorführung diesen gleichmäßigen Anstieg der Lautstärke vollführen, ohne merkliche Sprünge zu machen.
Lösung: Violine:5 ist ein Soundsample in dem eine Violine zu hören ist welche *sustain* gespielt wird.

```
d1 $ slow 4 $ cat [
midinote "[52][][][][][][][][45][44][52][][][][][][][]" # s "Violine:5",
midinote "[52][][][][][][][][45][44][52][][][][][][][]" # s "Violine:5",
midinote "[52][][][][][][][][45][44][56][][][][][][57][56]" # s "Violine:5",
midinote "[][60][][][][][][61][60][][63][][][][][][][]" # s "Violine:5"
] #cut 1 #speed 1.2 # sz 0.8 # orbit 1 # room 0.85 #gain 0.9
```

Figur 2
Sound: Wenn die Violine in diese Figur übergeht ist sie das Hauptinstrument des Stückes. Die Violine wird hierbei sehr hektisch aber im Gegensatz zu Figur 1 sauber gespielt.
Lösung: Violine:6 ist eine Violine die *tremolo* gespielt wird.

```
d1 $ slow 2 $ midinote "[[61][59][57][[][59]]][57 56 54 53]" # s "Violine:6"
    #cut 1 # room 0.85 # sz 0.8 # orbit 1
```

4.7 Instrument 7: Chello

4.7.1 *Figuren*

Abschnitt bearbeitet von: Raphael Drechsler

Figur 1
Analyse-Ergebnis:

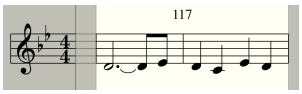

Abbildung 14: Chello Figur 1

Umsetzung in Tidal:

```
d1 $ slow 2 $ midinote "[[74][][][][][75]]][74 72 75 74]" # s "gtr" #cut 1 #
    room 0.85 # sz 0.8 # orbit 1
```

Figur 2
Analyse-Ergebnis:

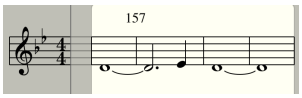

Abbildung 15: Chello Figur 2

Umsetzung in Tidal:

```
d1 $ slow 4 $ midinote "[[62][][][][][63]]][62]" # s "gtr" #cut 1 # room 0.85 #
    sz 0.8 # orbit 1
```

Weitere Figuren
Analog zur Violine existieren im hinteren Teil des Stücks weitere Figuren, auf deren Analyse und Synthese ebenfalls verzichtet wurde.

4.7.2 *Klangbild*

Abschnitt bearbeitet von: Nico Mehlhose

Art der Synthetisierung: Das Cello wird, wie auch die Violine, über die Samples synthetisiert[25], welches anschließend in den SuperCollider eingefügt werden.

Figur 1
Lösung:

```
d1 $ slow 2 $ midinote "[[50][][][][][51]]][50 48 51 50]" # s "Cello:1" #cut 2 #
    room 0.85 # sz 0.8 # orbit 1 #gain 0.7
```

Figur 2
Lösung:

```
d1 $ slow 4 $ midinote "[[50][[][][][51]]][50]" # s "Cello:2" #cut 2 # room
    0.85 # sz 0.8 # orbit 1 #gain 0.7
```

4.8 Instrument 8: Harfe

4.8.1 *Figuren*

Abschnitt bearbeitet von: Raphael Drechsler

Figur 1
Die Rhythmik sowie die Tonhöhe der zweitaktigen Figur ließen sich nicht ohne weiteres bestimmen. Zur Reduzierung des Arbeitsaufwandes wurde für die Figur die folgende Annahme getroffen.

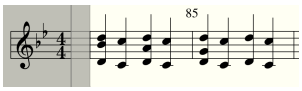

Abbildung 16: Harfe Figur 1

Umsetzung in Tidal:

```
d1 $ stack [
  midinote "[74 72]*2" # s "gtr",
  midinote "[62 60]*2" # s "gtr",
  cat [
    midinote "70 69" # s "gtr",
    midinote "67" # s "gtr",
    midinote "70 69" # s "gtr",
    midinote "66" # s "gtr"
  ]
]
```

4.8.2 *Klangbild*

Abschnitt bearbeitet von: Nico Mehlhose
Bestandteile der Harfe: Harfe
Art der Synthetisierung: Die Harfe wird mit Samples[25] synthetisiert da diese nicht über die SuperCollidertöne synthetisierbar ist.
Sound: Die Harfe wird in ihren Teilen sehr dominant gespielt.
Lösung:

```
d1 "i8" $ slow 2 $ stack [
```

```
  midinote "[57 55]*4" # s "Harp:13",
  midinote "[45 43]*4" # s "Harp:13",
  slow 2 $ fastcat [
    midinote "53 52" # s "Harp:13",
    midinote "50" # s "Harp:13",
    midinote "53 52" # s "Harp:13",
    midinote "49" # s "Harp:13"
  ]
] #speed 2 # room 0.5 # orbit 1 #gain 0.9
```

4.9 Instrument 9: Flügel

4.9.1 *Figuren*

Abschnitt bearbeitet von: Raphael Drechsler

Figur 1
Analyse-Ergebnis: Umsetzung in Tidal:

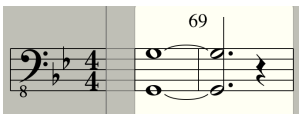

Abbildung 17: Flügel Figur 1

```
d1 $ slow 2 $ stack[midinote "43 " #s "superpiano", midinote "55 " #s
    "superpiano"]
```

Figur 2
Analyse-Ergebnis:

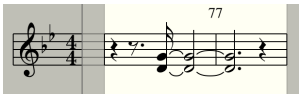

Abbildung 18: Flügel Figur 2

Umsetzung in Tidal:

```
d1 $ slow 2 $ stack[
  midinote "[[][][][][][67]][][][][]" #s "superpiano",
  midinote "[[][][][][][62]][][][] []" #s "superpiano"
] # room 0.5 # sz 0.83 # orbit 1
```

Figur 3
Analyse-Ergebnis:

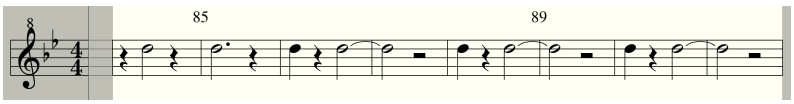

Abbildung 19: Flügel Figur 3

Umsetzung in Tidal:

```
d1 $ slow 2 $ midinote "[86 ~][86][~] [~] " # s "superpiano" # room 0.5 # sz
    0.83 # orbit 1 #gain "<0.65 0.7 0.75 0.8>"
```

Figur 4
Analyse-Ergebnis:

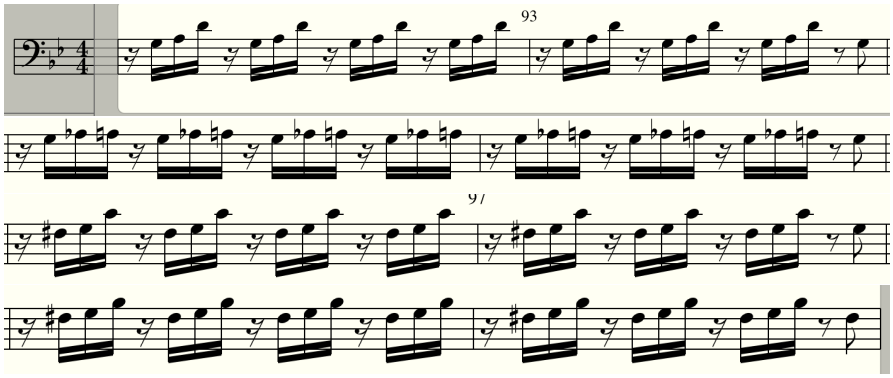

Abbildung 20: Flügel Figur 4

Umsetzung in Tidal:

```
d1 $ slow 2 $ cat [
midinote "~ 67 69 74 ~ 67 69 74 ~ 67 69 74 ~ 67 69 74 ~ 67 69 74 ~ 67 69 74 ~
    67 69 74 ~ ~ 67 ~ " # s "superpiano",
midinote "~ 67 68 69 ~ 67 68 69 ~ 67 68 69 ~ 67 68 69 ~ 67 68 69 ~ 67 68 69 ~
    67 68 69 ~ ~ 67 ~ " # s "superpiano",
midinote "~ 66 67 72 ~ 66 67 72 ~ 66 67 72 ~ 66 67 72 ~ 66 67 72 ~ 66 67 72 ~
    66 67 72 ~ ~ 67 ~ " # s "superpiano",
midinote "~ 66 67 70 ~ 66 67 70 ~ 66 67 70 ~ 66 67 70 ~ 66 67 70 ~ 66 67 70 ~
    66 67 70 ~ ~ 66 ~ " # s "superpiano"
]
```

Figur 5
Analyse-Ergebnis:

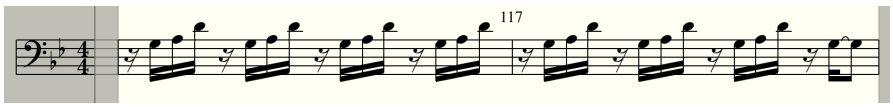

Abbildung 21: Flügel Figur 5

Umsetzung in Tidal:

```
d1 $ slow 2 $ midinote "~ 67 69 74 ~ 67 69 74 ~ 67 69 74 ~ 67 69 74 ~ 67 69 74
    ~ 67 69 74 ~ 67 69 74 ~ ~ 67 ~ " # s "superpiano"
```

4.9.2 *Klangbild*

Abschnitt bearbeitet von: Nico Mehlhose

Bestandteile des Flügels: Flügel
Art der Synthetisierung: Der Flügel wurde mittels dem SuperCollider syntetisiert. Dazu wurde die Funktion *OteyPianos* benutzt, welche einen Pianoartigen klang besitzt. Der restliche Teil der geschriebenen Funktion schneidet die Höhen etwas heraus und gibt dem Klang etwas mehr Tiefe.

```
(
SynthDef(\piano, { |out=0, freq= 391.995,gate=1, amp=0.5,rho=1|
   var son = OteyPiano.ar(freq, amp,
        rho:rho)*EnvGen.ar(Env.asr(0,1,0.1),gate,doneAction:2);
```

```
    Out.ar(out, Pan2.ar(son * 0.1,
        LinLin.kr(freq,36.midicps,90.midicps,-0.75,0.75)));
}).play(s);
)
```

Figur 1
Sound: Der Sound der Figur wird zur Einleitung in den ersten Hauptabschnitt benutzt. Dabei werden die Tasten des Flügels schnell gedrückt um einen möglichst lauten Ton hervorzubringen.
Lösung: In dieser Figur wurden die zwei Töne die stark angespielt wurden bereits per Audacity zusammengeschnitten um Code einzusparen.

```
d1 $ slow 2 $ sound "[K]" # room 0.2 #orbit 1 # gain 1.3
```

Figur 2
Sound: Der Sound der Figur wird zu der Hinführung in den ersten Hauptabschnitt benutzt. Dabei werden die Tasten des Flügels schnell gedrückt um einen möglichst lauten Ton hervorzubringen.
Lösung:

```
d1 $ slow 8 $ sound "[[][[][][][K:1]][][]] [] [] [] [[][[][][][K:1]][][]] [] []
    []" # room 0.2 #orbit 1 #gain 1.2
```

Figur 3
Sound: Der Flügel wird gefühlvoller angespielt um im Einklang mit der Harfe zu stehen. Lösung: `speed 2` wird hier benutzt damit das Soundsample des Flügels eine Oktave höher gespielt wird.

```
d1 $ slow 2 $ sound "[K:2 ~][K:2][~] [~] " #speed 2 #room 0.4 #orbit 1 #gain 1.5
```

Figur 4
Sound: Der Flügel wird in dieser Figur normal angespielt und führt den Zuhörer zu dem Höhepunkt des ersten Haupteiles welcher von der Violine gespielt wird.

Midinote Superpiano	Soundsample K:
66	7
67	3
68	6
69	4
70	9
72	8
74	5

Melodie: siehe Melodiepart Figur 4 Flügel

Figur 5
Sound: In der Figur wird der Flügel normal gespielt.
Lösung:
siehe Tabelle und Melodie Figur 5 Flügel

4.10 Instrument 10: Moog Syntheziser

4.10.1 *Figuren*

Abschnitt bearbeitet von: Raphael Drechsler

Figur 1
Herausgehört wurde der folgende Basslauf über acht Takte.

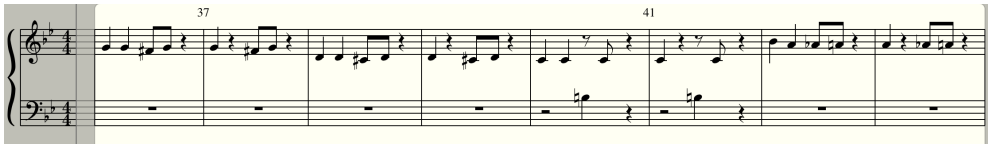

Abbildung 22: Moog Figur 1

Umsetzung in Tidal:

```
d1 $ slow 2 $ cat [
  midinote "[[55 55][54 55 ~ ~]]*2" # s "moog",
  midinote "[[50 50][49 50 ~ ~]]*2" # s "moog",
  midinote "[[48 48][47 48 ~ ~]]*2" # s "moog",
  midinote "[[58 57][56 57 ~ ~]] [[57 ~][56 57 ~ ~]]" # s "moog"
] # cut 1
```

Figur 2
Herausgehört wurde der folgende Basslauf über einen Takt.

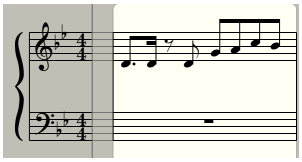

Abbildung 23: Moog Figur 2

Umsetzung in Tidal:

```
d1 $ midinote "[[[[50 ~ ~ 50]][~ 50]][55 57 60 58]]" # s "moog" # cut 1
```

4.10.2 *Klangbild*

Abschnitt bearbeitet von: Nico Mehlhose

Bestandteile des Moogs: Keyboard, PC, Mischboard
Art der Synthetisierung: Der Moog wird mittels dem SuperCollider programmiert. Der SuperCollider besitzt die Funktion, welche einen Moog synthetisieren kann. Dies ist die *BMoog*-Funktion welche im folgendem Codebeispiel zu sehen ist. Als nächstes müssen nurnoch die Sounds für die verschiedenen Frequenzen aufgenommen, geschnitten und eingefügt werden.

```
x=(
SynthDef(\moog, {
  arg freq=102, width=0.5, mul=0.5, freq2=300, q=0.2, mode=0;
  var moog ;

    moog=BMoog.ar(Pulse.ar(freq,width, mul),freq2, q, mode, mul:0.2);


    Out.ar(0, moog);
    Out.ar(1, moog);

}).play
)
```

Der eingefügte Moog Sound schwingt in beiden Figuren auf 66 Hz und wurde mittels dem `Midinote` Operator manipuliert.

Figur 1
Lösung:

```
p "i10" $ slow 2 $ cat [
  midinote "[[55 55][54 55 ~ ~]]*2" # s "BBFMoog:6",
  midinote "[[50 50][49 50 ~ ~]]*2" # s "BBFMoog:6",
  midinote "[[48 48][47 48 ~ ~]]*2" # s "BBFMoog:6",
  midinote "[[58 57][56 57 ~ ~]] [[57 ~][56 57 ~ ~]]" # s "BBFMoog:6"
] # cut 1 #speed 2 #gain 0.9
```

Figur 2
Lösung:

```
p "i10" $ midinote "[[[[50 ~ ~ 50]][~ 50]][55 57 60 58]]" # s "BBFMoog:6" # cut
    1 #speed 2 #gain 1.0
```

5 PERFORMANCE

5.1 Umsetzung der Performance

Abschnitt bearbeitet von: Raphael Drechsler

Nachdem die einzelnen Figuren aller Instrumente durch das vorangegangene Kapitel als Tidal-Code umgesetzt sind, soll nun die vorführbare Performance aus den einzelnen Teilen zusammengesetzt werden. Dabei soll im Wesentlichen die in Abbildung 1 gezeigte globale Struktur des Liedes nachempfunden werden.
Ein erster gedanklicher Ansatz bei der Umsetzung bestand in der Nutzung von mehreren Tidal-Connections. Tidal stellt 16 Verbindungen zum SuperDirt-Synthesizer bereit[26], daher kann jedes Instrument auf einer separaten Verbindung gespielt werden. Durch das Auswerten einzelner Zeilen während der Vorführung lassen sich somit die Figuren pro Instrument um- und abschalten. Der Code der Performance würde sich wie folgt darstellen.

```
d1 $ --Tidal-Code Drum-Figur 1
d1 $ --Tidal-Code Drum-Figur 2
d1 $ --Tidal-Code Drum-Figur n
d1 $ silence
...
d2 $ --Tidal-Code Marimba-Figur 1
d2 $ --Tidal-Code Marimba-Figur 2
d2 $ silence
...
d9 $ --Tidal-Code Moog Figur 2
d9 $ silence
```

Dabei ergibt sich jedoch das Problem, dass pro Zeitpunkt immer nur eine Änderung für ein Instrument ausgewertet werden kann. In der angestrebten Performance finden jedoch zu einzelnen Zeitpunkten mehrere Änderungen statt. So wird zum Beispiel beim Übergang von Part 4 in Part 5 der Bass (Moog-Synthesizer) pausiert, während Piano, Harfe und Marimba zeitgleich eine neue Figur zu spielen beginnen.

Um mehrere Instrumente innerhalb einer Code-Auswertung dazuschalten oder muten zu können, wurde die Implementierung der Performance an die in der Tidal Userbase beschriebene Vorgehensweise für das Stummschalten einzelner Instrumente angepasst.[27] Dazu wurden für alle Instrumente dieselbe Tidal-Connection `d1` genutzt und die einzelnen Figuren mithilfe des `stack`-Ausdrucks miteinander ver-

bunden. Dabei wurden alle Parts, die abschaltbar sein sollen in eine eigene Zeile geschrieben. Über die Nutzung des Tasten-Kurzbefehls für das Aus- bzw. Einkommentieren einzelner Zeilen im Code-Editor und das erneute Auswerten des gesamten Befehls, lassen sich somit einzelne Bestandteile in kurzer Zeit muten und wieder aktivieren.
Als Code-Editor wurde Atom verwendet. Um in Atom den gesamten Code des Ausdrucks trotz langer Zeilen im Bild zu behalten, muss im Einstellungsmenü unter dem Menüpunkt *Editor* über das Setzen des Kontrollkästchens *Soft Wrap* der Zeilenumbruch aktiviert werden.[28].

Um das Vorgehen für das Dazu- und Umschalten einzelner Spuren zu illustrieren, soll zunächst der Code für den ersten Part gezeigt und dessen Verwendung beschrieben werden.

```
1  ---PART1---
2  d1 $ stack [
3  --Drums1
4  sound "[bd hh]*4" #gain 0.4
5  --Marimba1&2
6  ,stack [
7  slow 2 $ midinote "[~ [51 49] 51 51][~ [51 46] 51 51][~ [51 49] 51 51][~ 51 ~
       51]" # s "glasstap"
8  -- ,slow 2 $ midinote "[~[~~~60]~~][]" # s "can" # speed (1 + (irand 10)*0.2)
9  ]
10 --Tuba1
11 ,sound "[~ cp ~ ~]" #pan 0.2
12 --Posaune1
13 -- ,sound "[][[][~ cp]]" #pan 0.8
14 ]
```

Zunächst wird der Code ausgeführt und die Performance beginnt. Entsprechend der globalen Struktur (Abbildung 1) wird nach einigen durchlaufenen Takten Figur 1 der Posaune gestartet. Dazu wird die Kommentierung der Zeile 13 per Tastenkurzbefehl aufgehoben und der gesamte Ausdruck ebenfalls per Tastenkurzbefehl erneut ausgewertet. Nach weiteren durchlaufenen Takten wechselt die Marimba von Figur 1 zu Figur 2. Diese Figuren unterscheiden sich den gespielten Kuhglocken. Der Code für diese befindet sich in Zeile 8. Folglich muss zum gewünschten Zeitpunkt Zeile 8 ent-kommentiert und der gesamte Ausdruck erneut ausgewertet werden.
Im Code zu Part 1 findet sich für die Schlagzeugspur in Zeile 4 ein Ausdruck `# gain 0.4`. Dieser soll genutzt werden um über das Vergrößern des Parameters in den ersten Takten von Part 1 live die Lautstärke der Schlagzeug-Figur anwachsen zu lassen, um die ansteigende BD nachzuempfinden. Zudem finden sich in der Performance vereinzelt weitere genutzte Effekte. So werden zum Beispiel die Figuren der Tuba und der Posaune mithilfe des `#pan`-Effektes[29] im Stereo-Kanal nach links und rechts verschoben um den Gesamtklang der Performance zu verbessern.

Das hohe Maß von wiederkehrenden und gleichbleibenden Figuren kann genutzt werden, um mehrere Parts des Liedes in einem Ausdruck zusammenzufassen. Dies wird im nachfolgend skizzierten Ausdruck für Part 2,3 und 4 deutlich.

```
1  ---PART2,3,4---
2  d1 $ stack [
3    --Drums2,3,4
4    --f2
5    cat [sound "[[bd ... ]"]
6    --f3
7    -- slow 2 $ sound "[bd ... hh]"
```

```
8   --f4
9   --slow 2 $ stack [sound "[bd ... rm ~]"] #gain 0.9]
10  --MOOG1,2
11  , slow 2 $ cat [midinote "[[55 ... ~]]" # s "moog" ] # cut 1 #gain 1.0
12  --2
13  -- ,midinote "[[[[50 ~ ~ 50]][~ 50]][55 57 60 58]]" # s "moog" # cut 1
14  --Piano1+2
15  -- ,slow 8 $ stack[midinote "43 " #s "superpiano", midinote "55 " #s
        "superpiano"]
16  --2
17  --,slow 8 $ stack[midinote "[...[]" #s "superpiano"] # room 0.5 # sz 0.83 #
        orbit 1 #gain 0.9
18  --STATIC:
19  --Marimba2
20  ,stack [slow 2 $ midinote "[~ [51 ... 51]" # s "glasstap",slow 2 $ midinote
        "[~[~~~60]~~][]" # s "can" # speed (1 + (irand 10)*0.2)]
21  --Tuba1
22  ,sound "[~ cp ~ ~]" #pan 0.2
23  --Posaune1
24  ,sound "[][[][~ cp]]" #pan 0.8
25  ]
```

Dabei sind alle vorkommenden Figuren eines Instrumentes jeweils untereinander geschrieben. Entsprechend der globalen Struktur ist pro Wechsel der Parts die Kommentierung per Tastenkurzbefehl derartig anzupassen, dass nur die gewünschten Figuren als Code ausgewertet werden. Zur besseren Übersichtlichkeit während der Performance sind alle gleichbleibenden Figuren im unteren Teil (im Auszug ab Zeile 18) zusammengefasst.

Der vollständige Code der Performance findet sich in der digitalen Anlage.

5.2 Abwesenheit von Sounds

Abschnitt bearbeitet von: Nico Mehlhose

In der Livevorführung sind nicht alle erzeugten Sounds zu hören, welche von uns eigens für die Klangsynthese aufgenommen wurden. Dieser Abschnitt soll erklären wieso einige Sounds von uns nicht verwendet werden.
In Figur 2 der Marimba werden Kuh-Glocken per Zufall angespielt. Die Glockensounds wurden im SuperCollider erzeugt mit folgendem Code:

```
-Code Marimba-
(
Pbind(
  \instrument, \bell,
  \fs, Pseq( (60..72), 1).midicps,
  \t60, 3,
  \pitchy, 2,
  \dur, 1
).play;
)
```

Durch eben gezeigten Code wurden 10 Glockensounds erzeugt. Bei dem zusammenfügen der Liveperformance fiehl uns aber auf das die Sounds das Klangbild zerstören, da diese viel zu hell waren und eher dumpf klingen müssen.
Die nächsten Sounds die nicht eingebunden wurden, waren die des Moogs. Für den Moog wurden extra die Frequenzen mittels dem *Sonic Visualiser*, welches ein kostenloses Programm zur Ansicht und Analyse von Audiodateien ist, ausgelesen.[30] Diese Frequenzen wurden dann im SuperCollider und dem im Moog gezeigten

Code erzeugt, geschnitten und in unseren Code geschrieben. Bei der Zusammenführung der Performance hatte der Moog nicht die richtigen Frequenzen getroffen, was daran liegt das der Moog einen ca. 10 Hz großen Bereich in der Visualisierung angezeigt hat. Dies führte zu dem schlechten Klangbild. Um dieses Problem zu lösen wurde sich ein Referenzton aus den erzeugten Samples gesucht, der mit der Liveperformance von *Brandt Brauer Frick* übereinstimmt. Mit diesem Referenzton wurde dann der Moog (Instrument 10) erzeugt. Die restlichen erzeugten Sounds für den Moog wurden als nächstes aus der Performance entfernt.
Alle eben genannten selbst erzeugten Töne sind unter *Sounds/Eigene/Instrumentename* zu finden.
Der letzte Sound der nicht in der Lifeperformance zu finden ist, ist die erste Figur der Tuba. Es ist uns nicht gelungen einen ähnlichen Sound wie diesen zu erzeugen, aufzunehmen oder Sounds in Tidal so zu entfremden das sie diesem Sound nahe kommen und eine akzeptierbare Qualität besitzen. Das nächste was an diesen Sound herangekommen ist, ist der aus Figur 1 der Tuba. Dieser Sound hat jedoch das Problem, dass die Qualität des Sounds unzureichend war.Aus diesem Grund benutzen wir eine einfache Clap um diesen Ton nachzustellen, da es der Grundidee des Originalsounds am nächsten kommt.

```
Benutzter Sound
p "i4" $ sound "[~ cp ~ ~]" #pan 0.2

Erstellter Sound
p "i4" $ sound "~ bd hh ~" # cut 1 # midinote 15 #gain "2 0"
```

6 GESANG

Abschnitt bearbeitet von: Raphael Drechsler

Neben der Umsetzung der einzelnen Instrumente wurden zusätzlich mit der Unterstützung einer Sängerin Gesangsaufnahmen angefertigt. Diese wurden ebenfalls SuperCollider als Sample zur Verfügung gestellt.
Hierbei galt es eine Herausforderung zu überwinden. Die Figuren des Gesangs sind acht Takte lang, starten jedoch nicht auf den ersten Schlag des ersten Taktes eines acht-Takte Intervalls; sie haben einen Auftakt.
Über das Berücksichtigen des Auftaktes bei der Aufnahme und mithilfe der folgenden Zeile Code können die acht Takte lange Samples für den Gesang mit dem Timing, welches dem Original entspricht, in der Performance verwendet werden.

```
d1 $ slow 8 $ sound "[[[][][][][VocP5]]]]"
```

7 ZUSAMMENFASSUNG

Bearbeitet von Nico Mehlhose, Raphael Drechsler
In diesem Dokument wurde zunächst aufgezeigt, weshalb sich für eine Umsetzung des Liedes *Pretend* von *Brand Brauer Frick* in Tidal anstatt in Euterpea entschieden wurde.
Die einzelnen Instrumente und deren Figuren wurden hinsichtlich ihrer Melodik, Rhythmik und ihres Klangbildes analysiert und entsprechend in Tidal Umgesetzt. Um ein originalgetreues Klangbild zu erzielen wurden dabei eigene Samples aufgenommen, Onlinesamples verwendet, Sounds des SuperColliders manipuliert sowie eigene Sounds in SuperCollider geschrieben.

Die einzelnen Figuren der Instrumente wurden entsprechend der ermittelten globalen Struktur zu einer live ausführbaren Performance zusammengefügt. Dabei wurden die Lautstärken der Instrumente aufeinander angepasst, weitere Effekte zur Verbesserung des Gesamtklanges eingebracht und die zuvor angefertigten Gesangssamples eingefügt.

Die Tidal-Dateien `BBF_instruments.tidal`, welche die Figuren der einzelnen Instrumente enthält und `BBFhs.tidal`, welche die Performance enthält, sind zusammen mit den Sounds unter der folgenden URL verfügbar.
https://github.com/MeisterXYZ/CM/tree/master/Code%2BSounds

LITERATUR/QUELLEN

[1] Wikipedia - brandt brauer frick. https://de.wikipedia.org/wiki/Brandt_Brauer_Frick. Zugriff: 15.12.2018.

[2] The brandt brauer frick ensemble feat. emika - pretend (live at concertgebouw brugge). https://www.youtube.com/watch?v=KCpLXpMB7F8. Zugriff: 17.01.2019.

[3] Tidalcycles userbase. https://tidalcycles.org. Zugriff: 26.01.2019.

[4] Euterpea - a haskell library for music creation. http://www.euterpea.com. Zugriff: 26.01.2019.

[5] Supercollider github-site. https://supercollider.github.io. Zugriff: 26.01.2019.

[6] Custom superdirt startup. https://tidalcycles.org/index.php/Custom_Samples. Zugriff: 26.01.2019.

[7] Beats per minute calculator and counter. http://www.beatsperminuteonline.com. Zugriff: 17.01.2019.

[8] Apple logic pro - produktwebsite. https://www.apple.com/de/logic-pro/. Zugriff: 22.01.2019.

[9] The brandt brauer frick ensemble feat. emika - pretend (official) !k7. https://www.youtube.com/watch?v=aBPEbF_u0cY. Zugriff: 22.01.2019.

[10] Brandt brauer frick - pretend. https://www.youtube.com/watch?v=bYTnFilh2EU. Zugriff: 22.01.2019.

[11] Tidal userbase - using * and / on groups. https://tidalcycles.org/index.php/Tutorial#Using_.2A_and_.2F_on_Groups. Zugriff: 22.01.2019.

[12] Tidal userbase - stack. https://tidalcycles.org/index.php/stack. Zugriff: 22.01.2019.

[13] Tidal userbase - cat. https://tidalcycles.org/index.php/cat. Zugriff: 22.01.2019.

[14] Tidal userbase - slow. https://tidalcycles.org/index.php/slow. Zugriff: 22.01.2019.

[15] Tidal userbase - gain. https://tidalcycles.org/index.php/gain. Zugriff: 22.01.2019.

[16] Tidal userbase - midinote. https://tidalcycles.org/index.php/midinote. Zugriff: 22.01.2019.

[17] Tidal userbase - rand, irand. https://tidalcycles.org/index.php/rand. Zugriff: 22.01.2019.

[18] Tidal userbase - speed. https://tidalcycles.org/index.php/speed. Zugriff: 22.01.2019.

[19] Modal synthesis of xilophone, marimba, glokenspiel, tubular bells. https://sccode.org/1-5ay. Zugriff: 30.01.2019.

[20] Professionelles aufnehmen und editieren. https://www.audacity.de/. Zugriff: 30.01.2019.

[21] Tidal userbase - loopat. https://tidalcycles.org/index.php/loopAt. Zugriff: 23.01.2019.

[22] Tidal userbase - control values are patterns too. https://tidalcycles.org/index.php/Tutorial#Control_values_are_patterns_too. Zugriff: 23.01.2019.

[23] Tidal userbase - room. https://tidalcycles.org/index.php/room. Zugriff: 23.01.2019.

[24] Tidal userbase - cut. https://tidalcycles.org/index.php/cut. Zugriff: 23.01.2019.

[25] Virtual playing orchestra: "it covers all the needs" – vitalker. http://virtualplaying.com/virtual-playing-orchestra/. Zugriff: 30.01.2019.

[26] Tidal userbase - play a single sample. https://tidalcycles.org/index.php/Tutorial#Play_a_Single_Sample. Zugriff: 25.01.2019.

[27] Tidal userbase - muting functions. https://tidalcycles.org/index.php/Muting_functions. Zugriff: 23.01.2019.

[28] Atom discuss: Turn on line wrap. https://discuss.atom.io/t/turn-on-line-wrap/1627. Zugriff: 23.01.2019.

[29] Tidal userbase - pan. https://tidalcycles.org/index.php/pan. Zugriff: 25.01.2019.

[30] Sonic visualiser. https://www.sonicvisualiser.org/. Zugriff: 30.01.2019.