Fakultät Informatik, Mathematik und
Naturwissenschaften
Studiengang Informatik Master

Projektarbeit zur Vorlesung Computermusik

BrandtBrauerFrick.hs

Autoren: Nico Mehlhose, Raphael Drechsler
Abgabedatum: 01.02.2019

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.
TODOWas ist Tidal[3] (dazu SuperCollider[5] erklären) , was ist Euterpea[4]?
Unsere Entscheidung Tidal zu nehmen beruht gewiss nicht auf einer zufälligen Entscheidung. In diese Entscheidung ist der Programmieraufwand, vorhandenen Informationen und die Möglichkeit den Synthesizer zu erweitern.
Bei dem Programmieraufwand wird sehr schnell klar, dass durch das Lied *Pretent* von BrandBrauerFrick Tidal besser geeignet ist als Euterpea. Der erste Gesichtspunkt der betrachtet wurde ist die Repetetivität des Songs, welcher in Euterpea zwar auch umsetzbar ist aber in Tidal von Anfang an gegeben ist, da Tidal die Sounds immer in einem Loop abspielt. Bei den vorhandenen Informationen stellt sich heraus, dass es keine Offiziellen Notenblätter für das Lied Onlinegibt, wodurch Euterpea etwas an Bedeutung verliert, da Euterpea für genaue Notenbestimmungen perfekt geeignet wäre. Da dieser Fakt aber nicht vorliegt, 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 allerdings existiert diese Möglichkeit mittels des Befehl . Mit diesem Befehl lässt sich ein Verzeichnis in Tidal integrieren.

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 [6] 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[7] (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.
Für die ersten vier Takte wurde bei der Erstellung der MIDI-Regionen eine Annahme getroffen.

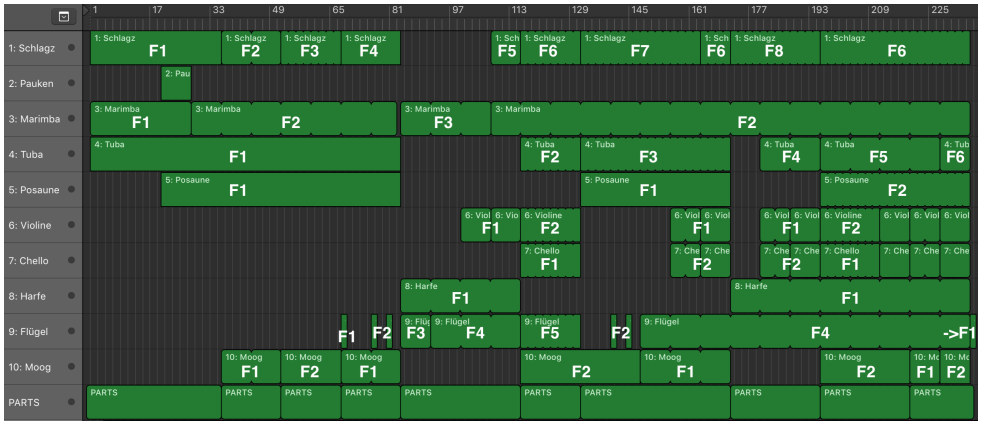

Abbildung 1: Globale Struktur dargestellt als leere MIDI-Regionen in Logic Pro mit Beschriftung der Figuren

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 die zehn Instrumente synthetisiert werden. 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 Parts, die besonders schwierig herauszuhören sind, da z.B. das Instrument nur sehr schwer hörbar ist, werden zusätzlich eine ähnliche Studio-Version[8] sowie eine früher Variante[9] 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, die 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 Samples sowie 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 6 - 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 Base-Drum, Note *Dis* für die High-Hat.

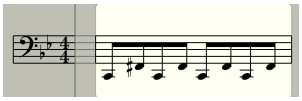

Abbildung 2: Schlagzeug Figur 1

Um in Tidal pro Takt eine Figur mit 4 Schlägen auf die Base-Drum und 4 Schlägen auf die High-Hat im Wechsel zu realisieren, lässt sich eine Kombination von Gruppierung und Wiederholung[10] 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.

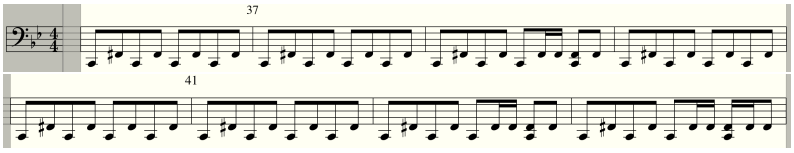

Abbildung 3: Schlagzeug Figur 2

Die Umsetzung in Tidal der einzelnen Takte ohne Fills erfolgt analog zu Figur 1. In den Takten mit Fills werden auf einige Zählzeiten Base-Drum und High-Hat gleichzeitig gespielt. Dies kann in Tidal durch die Nutzung von `stack` [11] umgesetzt werden. Über den `cat`-Ausdruck[12] 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.

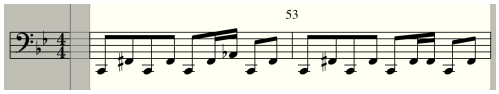

Abbildung 4: Schlagzeug Figur 1

In Tidal wurde die Figur hintereinander in Gruppierungen geschrieben und per `slow 2`[13] 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 Raushören 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[14] zum Regulieren der Lautstärke des gespielten Samples umgesetzt.

Figur 8
Wie Figur 4 aber ohne Base-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

Bestandteile des Schlagzeuges: herkömmliches Schlagzeug
Art der Synthetisierung: Da Bass Drum und High Head 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: Base Drum wird mit zunehmender dauer lauter gespielt
Sound HH: wird Anfangs nur sanft angespielt aber mit zunehmender Zeit etwas lauter
Problem: Base Drum und High Hat müssen mit zunehmender Vorführungszeit lauter werden
Lösung: Die Lautstärkensteigerung der Schlagzeugs wird live mittels der Erhöhung des *gain*-Parameters realisiert.

```
Code: p "i1" $ sound "[bd hh]*4" #gain 0.9 # midinote 58
```

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 und erhöht sich nicht.
Lösung:

```
Code:\\
p "i1" $ 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]" ]
] #midinote 58
```

Figur 3

TODO Klangbild 3-8

Fig4:
Sound Rim:
Nico: selber bauen da keine hölzernen klänge vorhanden sind
Fig6:
irgendwas mit gain?

Töne: hh, bd (dumpf, wenig knackig)
rim

4.2 Instrument 2: Pauken

4.2.1 *Figuren*

OFFEN
TODO (Hierzu Studio-Version hören)

4.2.2 *Klangbild*

Bestandteile der Pauke: 3 Kesselpauken
Art der Synthetisierung:?

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.

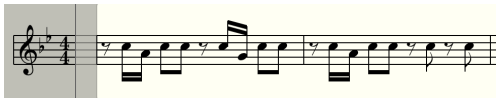

Abbildung 5: Marimba Figur 1

Die Tonhöhe des Sounds `glasstrap` wurde mithilfe des `midinote`-Ausdrucks [15] auf die jeweils gewünschte Tonhöhe geändert.

```
p1 $ 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.

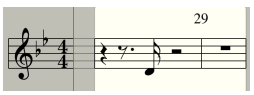

Abbildung 6: Marimba Figur 2

Der Ausdruck `irand 10` [16] liefert einen zufälligen int-Wert von 0 bis einschließlich 9 zurück. Dieser Wert verändert in Verbindung mit dem `# speed`-Ausdruck[17] die Abspielgeschwindigkeit des `can`-Samples und damit dessen Tonhöhe. So werden 9 verschiedene Kuhglocken simuliert.

```
p1 $ 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.

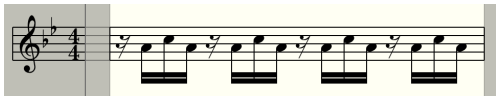

Abbildung 7: Marimba Figur 3

Umsetzung in Tidal:

```
p1 $ midinote "[~ 49 51 49]*4" # s "glasstap"
```

4.3.2 *Klangbild*

TODO Raphas antwort+ Figuren 1,2,3 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 aufgenommen werden. Dies geschieht mittels Holzsticks die auf einen hölzernen Untergrund geschlagen werden.
Figur 1
Sound Marimba: Die Marimba wird am Anfang des Stücks relativ dominant, im gegensatz zu den restlichen Instrumenten gespiel
Lösung:

Figur 2
Sound Marimba: Wird nur schwach angespielt wodurch sie in diesem Teil in den Hintergrund tritt. Holzsticks auf Marimba in verschiednenen Tonhöhen wobei eher rhythmisch als melodisch eingesetzt, dazu Kuh-Glocken bereitstellen für Random-Funktion

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.

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

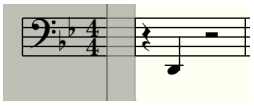

Abbildung 8: Tuba Figur 1

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.

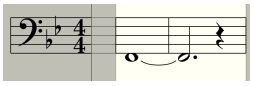

Abbildung 9: Tuba Figur 2

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` [18] 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.

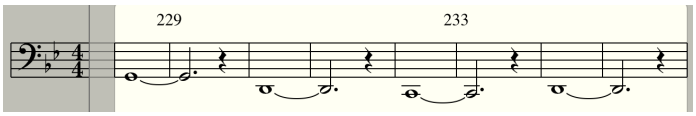

Abbildung 10: Tuba Figur 4

Das lauter Werden der Töne über die Dauer von zwei Durchläufe der Figur wurde mit einem Pattern realisiert, welches dem `# gain`-Effekt übergeben wurde.[19] 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. [20]

```
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
TODO Figur 2+3 Raphas antwort Restlichen Figuren machen
Bestandteile der Tuba: normale Tuba, welche aber entartet benutzt wird
Art der Synthetisierung: Da die Tuba ein reales Musikinstrument ist, welches in dieser Form nicht im Supercollider enthalten ist, werden hierfür Samples benutzt. Die Samples werden für die entartete Benutzung in Tidal so manipuliert, dass sie die Sounds nachempfinden.
Für Figur 2 werden eigene Samples aufgenommen. Den Sound soll ein Handscheibenwischer mit einem hohlen Griff erzeugen.
Figur 1
Sound: In dieser Figur wird auf das Mundstück der Tuba geschlagen. Der erzeugte Ton hört sich in etwa an wie eine Base Drum ohne Bass.
Lösung:

```
d1 $ sound "~ bd ~ ~" # midinote 15
```

Figur 2
Sound: ähnlich eines Reifens der Luft verliert
Lösung:

```
Nico: d1 $ sound "[sax ~ ~ hh]" # speed 0.35 # midinote 55 # gain "[1 0]" # cut
    1 \\
d1 $ sound "[trump ~ ~ hh]" # speed 0.05 # midinote 55 # gain "[0.7 0]" # cut
    1\\
```

Figur 3
Sound: Wie Sound aus Figur 2 aber mit mehr Druck und nicht durchgehend.
Lösung:

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.

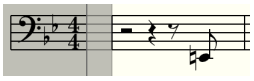

Abbildung 11: Posaune Figur 1

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*

TODOFigur 1+2 antwort Rapha warten
Bestandteile der Posaune: normale 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 für die entartete Benutzung in Tidal so manipuliert, dass sie die Sounds nachempfinden.
Für Figur 2 werden eigene Samples aufgenommen. Den Sound soll ein Handscheibenwischer mit einem hohlen Griff erzeugen.
Figur 1
Sound: In dieser Figur wird auf das Mundstück der Posaune stoßartig angespielt.
Lösung:

```
\\
```

Figur 2
Sound: ähnlich zu Figur 3 der Tuba allerdings mit weniger tief.
Lösung:

```
\\
```

4.6 Instrument 6: Violine

4.6.1 *Figuren*

Abschnitt bearbeitet von: Raphael Drechsler

Figur 1
Herausgehört wurde die folgende Figur über 16 Takte.

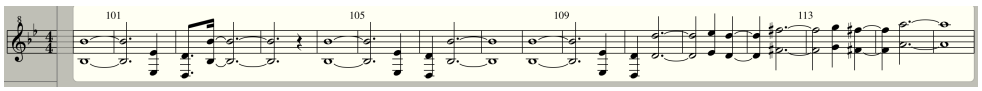

Abbildung 12: Violine Figur 1

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.[21]

```
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:

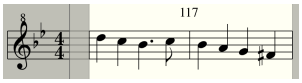

Abbildung 13: Violine Figur 2

Umsetzung in Tidal:

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

Figur ?
TODO weitere Figuren im hinteren Teil des Stückes

4.6.2 *Klangbild*

TODOBeispiel für Rapha ausarbeiten, Art der Synthetisierung ändern, nach anderen violinensounds suchen in den samples Bestandteile der Violine: Violine
Art der Synthetisierung: Hierfür wurden Samples Online gesucht da es keine vergleichbaren SuperCollider Sounds gibt oder jemand eine Violine besitzt.
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. Im zweiten Teil wird sie hektisch und sauber gespielt.
Problem: Die Violine muss in unserer Vorführung diesen gleichmäßigen Anstieg der Lautstärke vollführen, ohne merkliche Sprünge zu machen.
Lösung:

```
\\
```

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:

```
\\
```

4.7 Instrument 7: Chello

4.7.1 *Figuren*

Abschnitt bearbeitet von: Raphael Drechsler

Figur 1
Analyse-Ergebnis:
Umsetzung in Tidal:

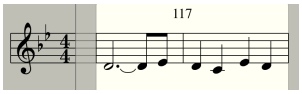

Abbildung 14: Chello Figur 1

```
p1 $ 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:

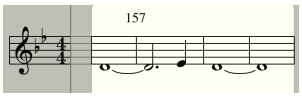

Abbildung 15: Chello Figur 2

Umsetzung in Tidal:

```
p1 $ slow 4 $ midinote "[[62][][][][][63]]][62]" # s "gtr" #cut 1 # room 0.85 #
    sz 0.8 # orbit 1
```

Figur ?
TODO weitere Figuren im hinteren Teil des Stückes

4.7.2 *Klangbild*

TODOFiguren einarbeiten, Samples? Ruhige Parts, Hektik (schnell gespielte Töne)

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.

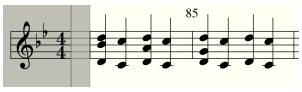

Abbildung 16: Harfe Figur 1

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*

TODO Art der Synthetisierung anpassen Bestandteile der Harfe: Harfe
Art der Synthetisierung: Die Harfe wird mit Samples synthetisiert da diese nicht über die SuperCollidertöne synthetisierbar ist.
Sound: Die Harfe wird in ihren Teilen sehr deutlich gespielt. Jedoch schwankt die Lautstärke mit der sie gespielt wird von leise zu laut und wieder zurück.
Problem: Die schwankende Lautstärke ist das Hauptproblem bei der Aufführung.
Lösung:

\\

4.9 Instrument 9: Flügel

4.9.1 *Figuren*

Abschnitt bearbeitet von: Raphael Drechsler

Figur 1
Analyse-Ergebnis: Umsetzung in Tidal:

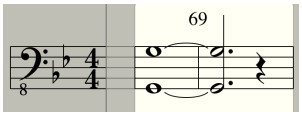

Abbildung 17: Flügel Figur 1

```
p1 $ slow 2 $ stack[midinote "43 " #s "superpiano", midinote "55 " #s
    "superpiano"]
```

Figur 2
Analyse-Ergebnis:

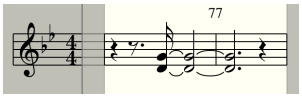

Abbildung 18: Flügel Figur 2

Umsetzung in Tidal:

```
p1 $ slow 2 $ stack[
  midinote "[[][][][][][67]][][][]" #s "superpiano",
  midinote "[[][][][][][62]][][]] []" #s "superpiano"
] # room 0.5 # sz 0.83 # orbit 1
```

Figur 3
Analyse-Ergebnis:

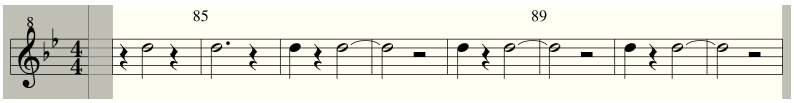

Abbildung 19: Flügel Figur 3

Umsetzung in Tidal:

```
p1 $ 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:

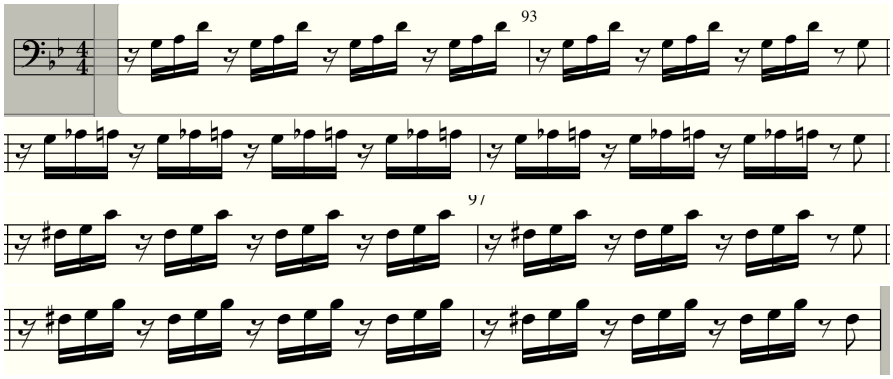

Abbildung 20: Flügel Figur 4

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:

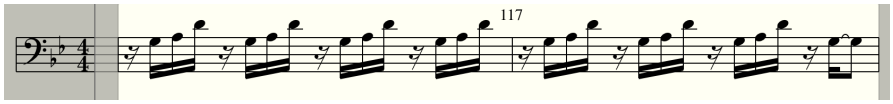

Abbildung 21: Flügel Figur 5

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*

TODOvllt auch aufnehmen? Bestandteile des Flügels: Flügel
Art der Synthetisierung:?
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:

\\

Figur 2
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.
Lösung:

\\

Figur 4
Sound: normal angespielter Flügel
Lösung:

\\

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.

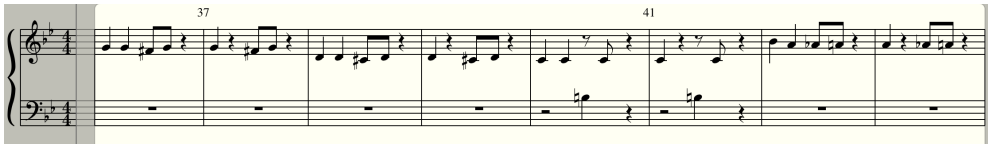

Abbildung 22: Moog Figur 1

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.

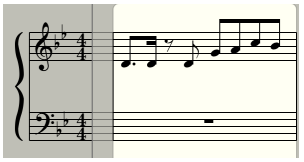

Abbildung 23: Moog Figur 2

Umsetzung in Tidal:

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

4.10.2 *Klangbild*

Bestandteile des Moogs: Keyboard, PC, Mischboard
Art der Synthetisierung: Der Moog wird von uns selbst im SuperCollider Programmiert, da ein Moog relativ leicht selbst zu Programmieren ist. Des weiteren wollen wir uns damit die Option offen halten anstatt eines Moogs eine Bassline zu benutzen.

Für die Synthetisierung des Moogs mussten einige kleine Teilschritte unternommen werden. Zuerst die Syntetisierung des Moogs im SuperCollider.
Dazu wurde folgender Code geschrieben:

```
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
)
```

Als nächstes muss dieser Sound aus dem SuperCollider aufgenommen werden. Dies geht mit dem Befehl *Server.default.record;*.
Der dritte und letzte Schritt war das zuschneiden der Audisamples. Dazu wurde Audacity benutzt. Danach kann dann der Code für die einzelnen Figuren angepasst werden.
Figur 1
Lösung:

```
p "i10" $ slow 8 $ fastcat [sound "[[BBFMoog:5 BBFMoog:5][BBFMoog:4 BBFMoog:5 ~
    ~]]*2" # cut 1 ,
sound "[[BBFMoog:3 BBFMoog:3 ][BBFMoog:2 BBFMoog:3 ~ ~]]*2" # cut 1,
sound "[[BBFMoog:1 BBFMoog:1 ][BBFMoog:0 BBFMoog:1 ~ ~]]*2" # cut 1,
sound "[[BBFMoog:7 BBFMoog:6 ][BBFMoog:5 BBFMoog:6 ~ ~]] [[BBFMoog:6
    ~][BBFMoog:5 ~ BBFMoog:6 ~ ~]]" # cut 1
] #gain 1.5
```

Figur 2
Lösung:

```
p "i10" $ sound "[[[[BBFMoog:3 ~ ~ BBFMoog:3]][~ BBFMoog:3]][BBFMoog:5
    BBFMoog:6 BBFMoog:8 BBFMoog:7]]" # cut 1 # gain 1.5
```

5 PERFORMANCE

Abschnitt bearbeitet von: Raphael Drechsler

Bis jetzt nur alle instrumente einzeln. Soll: Verbinden dass Performance analog Globale Struktur Abb 1 rauskommt. Umsetzung erster Ansatz: jedes Instrument

bekommt eigenen Kanal. Mit `p "i10"` möglich. Problem: Zu einem Zeitpunkt der Ausführung nur ein Instrument beeinflussbar. Nicht cool für wenn mehr umbricht als nur 1 instrument.

Also: Parts nachempfinden. Dazu einzelne Blöcke bauen. in Blöcken spielbar mit Parametern.

Zur Ausführung im Block wenn sich mehr ändert: Mit Kommentar-Funktion siehe mute instruments [22] Dazu alles was mut-bar sein soll in eigene Zeile. Um alles zu sehen in Atom Einstellungen - Editor - Soft Wrap aktivieren [23].

Exemplarisch sollen erste zwei Blöcke beschrieben werden.

Part 1

```
---PART1---
d1 $ stack [
--Drums1
sound "[bd hh]*4" #gain 0.4
--Marimba1&2
,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)
]
--Tuba1
,sound "[~ cp ~ ~]" #pan 0.2
--Posaune1
-- ,sound "[][[][~ cp]]" #pan 0.8
]
```

Innerhalb des Parts an gain drehen um Lautstärke-Anstieg BD von Drums live zu reaalisien. Einkommentieren von Tuba. Einkommentieren von zweitem Zeil. Daher dieser Teil

Part 2 TODO: Update

```
---PART2,3,4---
d1 $ stack [
  --Drums2,3,4
  --f2
  slow 8 $ fastcat [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]" ]]
  --f3
  -- slow 2 $ sound "[bd hh bd hh][bd [hh ho] bd hh] [bd hh bd hh][bd [hh hh]
      bd hh]"
  --f4
  -- slow 2 $ stack [sound "[bd hh]*8" #gain 1.0,fastcat[sound "[rm rm rm rm
      rm]", sound "[rm rm rm rm ~]"] #gain 0.9]
  --Marimba2
  ,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)]
  --Tuba1
  ,sound "[~ cp ~ ~]" #pan 0.2
  --Posaune1
  ,sound "[][[][~ cp]]" #pan 0.8
  --MOOG1,2
  ,slow 8 $ fastcat [midinote "[[55 55][54 55 ~ ~]]*2" # s "moog" # cut 1,
      midinote "[[50 50][49 50 ~ ~]]*2" # s "moog" # cut 1, midinote "[[48
      48][47 48 ~ ~]]*2" # s "moog" # cut 1, midinote "[[58 57][56 57 ~ ~]]
      [[57 ~][56 57 ~ ~]]" # s "moog" # cut 1] #gain 1.0
```

```
  --2
  -- ,midinote "[[[[50 ~ ~ 50]][~ 50]][55 57 60 58]]" # s "moog" # cut 1
  --Piano1+2
  -- ,slow 8 $ stack[midinote "43 " #s "superpiano", midinote "55 " #s
      "superpiano"]
  --2
  -- ,slow 8 $ stack[midinote "[[[][[][][][67]][][][]] []
      [[[][[][][][67]][][][]] []" #s "superpiano" ,midinote
      "[[[][[][][][62]][][][]] [] [[[][[][][][62]][][][]] []" #s
      "superpiano"] # room 0.5 # sz 0.83 # orbit 1 #gain 0.9
]
```

Beschreiben Toggle Ablauf.
TODO:
Spuren per Stack verbinden?
Konzept für Ablauf Performance?
Wie mehrtaktige Figuren mit every zusammenfassen? Link zu den Soundfiles: http://virtualplaying.com/vi
playing-orchestra/
https://rhythm-lab.com/moog-rogue-bass/

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: 22.01.2019.

[4] Euterpea - a haskell library for music creation. http://www.euterpea.com. Zugriff: 22.01.2019.

[5] Supercollider github-site. https://supercollider.github.io. Zugriff: 22.01.2019.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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