Hochschule für Musik und Theater Hannover - Elektronisches Studio des Instituts für Neue Musik

# Entwicklung und Anpassung einer Computer-Peripherie an typische Situationen freier Soloimprovisation

Diplomarbeit vorgelegt von: Alex Hofmann
Studiengang: Musikerziehung
Studienrichtung: Jazz/Rock/Pop

1. Gutachter: Joachim Heintz
2. Gutachter: Prof. Dr. Herbert Hellhund

Hannover, den 31.10.2006

# Inhaltsverzeichnis

# 1. Einleitung

## *1.1 Einführung in die Thematik*

In meiner Arbeit werde ich die Erstellung meines interaktiven Musikprogramms beschreiben und anschließende Spielversuche vorstellen, in denen ich die Funktionsweisen dieser Software teste. Die Entwicklung fand auf einem handelsüblichen Computer aus dem aktuellen Jahr 2006 statt. Das zu programmierende Patch soll einen eigenen Spielcharakter aufweisen, Interaktion mit dem menschlichen Spieler anstreben und sich folglich in Echtzeit an der Gestaltung der so entstehenden gemeinsamen Improvisation beteiligen. Bei der Umsetzung werde ich von einem musikalischen Standpunkt an die Entwicklung herangehen und versuchen mathematische und informationstechnische Aspekte so einfach wie möglich zu lösen.

## *1.2 Inspiration*

Durch die gestiegene Leistungsfähigkeit handelsüblicher Computer seit dem Jahr 2000, ist es für mehr Menschen möglich geworden mit elektronischer Musik zu experimentieren. Hierbei sind Grenzen meist durch die Auswahl der Musiksoftware gesetzt. Viele Musikprogramme wie z.B. Logic oder Cubase, bieten durch die Architekturverwandtschaft zu herkömmlichen Mehrspuraufnahmegeräten einen guten Einstieg, sind aber für die Entdeckung neuer Klang- und Interaktionsmöglichkeiten im Gebiet der Live-Elektronik zu unflexibel. Für Live-Elektronik mit handelsüblichen PC oder Macintosh Computern hat sich, auf dem Prinzip der Klangschleife (Loop) basierend, ein neuer Markt entwickelt. Mit Software wie Ableton Live oder Fruity Loops kann der Musiker Audiospuren unkompliziert in Echtzeit aufnehmen, schichten und verfremden. Angeschlossene Midi-Steuerungen vereinfachen eine Veränderung von Parametern durch Fußpedale oder Drehknöpfe. Bei diesen Softwareprodukten ist ein deutlicher Bezug zu Musikstilen aus dem Bereich Techno, House etc. zu erkennen. Hier werden

Loops und Samples immer in Bezug zu einem festen Metrum und einer festgelegten Taktlänge verarbeitet.
Durch mein zunehmendes Interesse im Gebiet der Live-Elektronik, stieß ich jedoch auf die Grenzen dieser Software. Inspiriert durch Anthony Braxtons Solo-Saxophonkonzerte, die er mit Hilfe der von ihm dafür entwickelten „Language Types“[1] zusammensetzt, kam ich auf die Idee, mit einem Computerprogramm Solo-Konzerte spielen zu wollen.

Bei einer freien Soloimprovisation ist für mich die Takt/Metrum basierte Loop kein funktionierender Grundbaustein. Der Audiostream, als sich immer verändernder Fluss von Musik, soll nicht in einer Schlaufe gefangen werden und sich solange wiederholen bis Midi-Befehle seine Struktur verändern. Das Audiomaterial soll während der gesamten Soloimprovisation festgehalten und markante Elemente sollten in veränderter Form zurückgegeben werden. Die vom Spieler erzeugten Klänge und Strukturen sollen dabei gleichzeitig die Ausgabe des Patch beeinflussen und rückwirkend beeinflusst die Klangausgabe des Patch den Spieler.

# 2. Einordnung der Arbeit in ein historisches Umfeld

In diesem Kapitel werde ich Definitionen und Entwicklungen im Bereich der Computermusik anführen, die für das Verständnis meiner Arbeit nötig sind. Dabei erhebe ich keinen Anspruch auf Vollständigkeit. So werden von mir z.B. keine Formen der Computer-Klangsynthese besprochen, die in meiner Umsetzung keine Anwendung finden.

[1] Siehe P.N. Wilson: Anthony Braxton, S. 96

## *2.1 Computermusik*

„Computermusik bezeichnet Musik, für die ein Computer notwendig oder wesentlich ist. […] Der Computer wird hierbei zum Erstellen von elektronischen Klängen (Klangsynthese) oder zum Berechnen von Kompositionen, dass heißt zur algorithmischen Komposition (Partitursynthese) verwendet[…]."[2] Im Falle der algorithmischen Komposition werden Musikregeln in den Computer eingegeben und mit Wahrscheinlichkeiten verknüpft. Als erstes Stück wurde 1956 die viersätzige „Illiac Suite für Streichquartett" von Lejaren A. Hiller und Leonard M. Isaacson unter Verwendung eines Computers komponiert. Dieser hat die errechnete Musik als Notenschrift ausgegeben. Anschließend wurde es auf traditionellen Instrumenten gespielt.[3] Ein anderer Ansatz ist der von Gottfried M. Koenig, welcher sich 1963 von seinem Kompositionsprogramm „Projekt 1" mit Hilfe von Zufallsgeneratoren nach seriellen Kompositionsstrategien Grundmaterial liefern lässt. Dieses verarbeitet er dann selbst weiter, um das fertige Musikstück zu kreieren. Diese Methode entwickelt er aus der Feststellung heraus, dass „(…)eine Mechanik ohne spontane Eingriffe musikalisch unbefriedigend bleibt."[4] Was Koenig dabei schon im Kern erkannt hat, sind die komplexen menschlichen Denkprozesse, die bei der Erstellung einer Komposition eine Rolle spielen. Diese als Computerprogramme zu formulieren setzt erhebliches informationstechnisches Wissen voraus.

## *2.2 Live-Elektronik*

„Der Begriff Live-Elektronik entsteht zu Beginn der 60er Jahre und bezeichnet Konzerte, bei denen in Echtzeit elektronische Klangerzeugung oder elektronische Verfremdung von akustischen Instrumenten bzw. der menschlichen Stimme vorgenommen wird." [5]

---

[2] H.Wandler: Elektronische Klangerzeugung und Musikreproduktion, S. 39
[3] Siehe Ebda., S. 41
[4] Siehe Ebda., S. 42
[5] Ebda., S. 26

In der Literatur wird „Imaginary Landscape No.1" (1939) von John Cage als erstes Live-Elektronik Stück genannt. Dabei verwendet er zu den beiden Instrumenten Klavier und Becken zwei Schallplattenspieler. Auf den Schallplatten befindet sich Klangmaterial (Sinustöne), das durch Geschwindigkeitsänderungen beim Abspielen verändert wird.[6] Die ersten Echtzeit-Klangverfremdungsmöglichkeiten bietet der Ringmodulator (ab 1964) sowie etwas später der Harmonizer, der Klänge verschieben lässt um den Klangumfang des akustischen Instrumentes zu erweitern oder Mikrointervalle zu erzeugen. Weitere Möglichkeiten bieten dann Hallsimulationen sowie die Verwendung mehrerer Lautsprecher.[7] „Mixtur für Orchester, vier Sinusgeneratoren und vier Ringmodulatoren" (1964) von Karlheinz Stockhausen soll hier als Werkbezug genannt werden.[8] Wenn in einem Konzert ein Computerprogramm in Echtzeit Kompositionen nach algorithmischen Berechnungen erstellt und das Ergebnis über Lautsprecher ausgibt, kann von computergestützter Live-Elektronik gesprochen werden.[9] Bei der computergestützten Interaktion werden Klänge der Solisten so verändert, dass zusätzlich zum Originalklang elektronisch veränderte Variationen von diesen über Lautsprecher erklingen. So reagiert der Computer in „Répons" (1981) von Pierre Boulez beispielsweise auf Lautstärke oder Ausklingverhalten der gespielten Instrumente.[10] Gegenüber der Partitursynthese kann bei der computergestützten Interaktion in Echtzeit in das errechnete Ergebnis nicht eingegriffen werden. "Der Musiker/Komponist muss in der Konzertsituation „weitermachen", auch dann, wenn ihm eine gegenwärtige Aktion des Systems nicht zusagt."[11] Eines der wichtigsten Zentren für Live-Elektronik stellt das, ab 1972 unter Leitung von Hans Peter

[6] Vgl. M.Supper: Elektroakustische Musik & Computermusik S. 13
[7] Siehe H.Wandler:Elektronische Klangerzeugung und Musikreproduktion, S. 26
[8] Siehe M.Supper: Elektroakustische Musik & Computermusik S. 16
[9] Vgl. H.Wandler:Elektronische Klangerzeugung und Musikreproduktion, S. 45
[10] Siehe ebda., S. 27
[11] Ebda. S. 92

Haller aufgebaute, Experimentalstudio des Südwestfunks Baden-Baden in Freiburg dar.[12]
Eine weitere Entwicklung im Bereich der Computermusik ist das Sampling, welches erstmals 1979 von der Firma CMI-Fairlight in Form eines Samplers vorgestellt wird.

## *2.3 Sampling*

Beim Sampling werden Klänge digital aufgezeichnet und wiedergegeben. Hierbei wird das akustische Signal in regelmäßigen Zeitabständen durch einen Analog/Digital-Wandler abgetastet und die gewonnen Werte werden als Daten abgespeichert. Die Abtastfrequenz beeinflusst dabei maßgeblich die Klangqualität. Ein heutzutage etablierter Standard durch die Audio CD ist, 44.1 kHz Abtastfrequenz mit einer Dynamikauflösung von 16 Bit.[13]

## *2.4 MAX*

Um Computern Anweisungen zu geben benutzt man Programmiersprachen. In diesen werden Programme geschrieben, die es dem Anwender erleichtern Aufgaben vom Computer ausführen zulassen, da sie in einem speziellen Bereich Arbeitswerkzeuge zur Verfügung stellen. Beispielsweise gibt es in einer Textverarbeitungssoftware einen Menupunkt zum Ausdrucken des Textes.
Das von mir verwendete Programm „Max" ist ein Grenzfall zwischen Programm und Programmiersprache. Auf der einen Seite ist es dadurch Programm, dass es als Programm mit eigener Oberfläche ausgeliefert wird und vorwiegend Werkzeuge zur Klangverarbeitung zur Verfügung stellt, andererseits können diese Werkzeuge in beliebiger Art zusammengesetzt werden um neue Aufgaben zu erfüllen. So hat diese, ab 1986 von Miller Puckette am IRCAM entwickelte, Software schnell einen Anwenderkreis gefunden, welcher damit musikalische Anwendungen gestaltet. Ein Grund dafür ist die vergleichsweise einfache Bedienung durch graphische Objekte,

[12] Vgl. M.Supper: Elektroakustische Musik & Computermusik S. 15
[13] Siehe H.Wandler: Elektronische Klangerzeugung und Musikreproduktion, S. 95

welche durch virtuelle Kabelverbindungen einander Botschaften übermitteln. Ein daraus entstandenes Gebilde wird als „Patch" bezeichnet. Die Möglichkeit zusätzliche Objekte, meist in „C" programmierte Bausteine die eine konkrete Aufgabe erfüllen, später hinzuzufügen, stellt eine Umsetzung nahezu sämtlicher interaktiver Musikanwendungen in Aussicht. Max selbst setzt sich aus einer Vielzahl solcher Bausteine zusammen, sodass diese in Kombination immer komplexere Datenverarbeitungen ausführen können. Die Audioverarbeitungsbibliothek MSP wird 1997 hinzugefügt und funktioniert auf handelsüblichen Computern.[14]

## *2.5 Freie Soloimprovisation*

Der Begriff freie Soloimprovisation umfasst drei Komponenten. Die Erste ist, dass es sich um einen Solovortrag handelt. Dadurch ist die Anzahl der Spieler auf eine Person begrenzt. Durch die in meinem Fall geplante Verwendung der computergestützten Live-Elektronik könnte zwar eine Duo-Situation entstehen, dies wird aber erst die Auswertung zeigen.
Bei einer Improvisation ist der musikalische Urheber auch gleichzeitig Darbietender und Interpret. Diese stellen daher bei der Musikproduktion eine Einheit dar. „Diese Unmittelbarkeit schließt die weitgehende Unvoraussagbarkeit des musikalischen Improvisationsprozesses(…) ein."[15]
Der Zusatz „freie" Soloimprovisation lässt dem Spieler alle musikalischen Freiräume über das zur Verfügung stehende Material. Eine „typische Situation" dafür wäre der Vortrag einer Soloimprovisation im Stile des zeitgenössischen Jazz.
"Kaum eine kulturelle Aktivität erscheint flüchtiger als die musikalische Improvisation. Einer Folge gelebter Augenblicke gleichend, widerstrebt sie dem Versuch einer Festschreibung."[16] So beginnt Noglik sein Buch „Klangspuren - Wege improvisierter Musik", in dem er einige Musiker

[14] Vgl. H.Wandler: Elektronische Klangerzeugung und Musikreproduktion, S. 46
[15] B. Noglik: Klangspuren, S. 332
[16] B. Noglik: Klangspuren, S. 6

portraitiert und anschließend versucht, musikalische Bewegungen und Zusammenhänge der Entwicklungen im zeitgenössischen Jazz/improvisierter Musik aufzuzeigen. Weiter schreibt Noglik: „Die Jazztradition erwies sich als eine wichtige Triebkraft auch für musikalische Entwicklungen und Bereiche, die heute nicht mehr ausnahmslos dem Jazz zuzuordnen sind.[…] Im Grunde gibt es so viele [freie Improvisations-] Stilrichtungen wie es Musiker und Gruppen gibt." Auf Grund dessen gestaltet sich die Programmierung meines Patch besonders interessant, denn eigentlich kann man nicht wissen worauf man es vorbereiten soll, wenn jeder Spieler sein eigenes Konzept verfolgen kann. Eine Möglichkeit dem Spieler Freiräume zu lassen, aber dennoch Vorbereitungen treffen zu können zeigt Matthias Ockert auf, welche ich im Folgenden beschreiben werde.

# 3. Betrachtung unterschiedlicher Programmmodelle

Da sich diese Arbeit vorwiegend mit der Entwicklung meines Patch beschäftigt, möchte ich nur zwei andere Möglichkeiten vorstellen, dessen Entwicklungen mein Interesse in besonderer Weise auf sich gezogen haben und dadurch meine Arbeit beeinflussen.

## *3.1 Matthias Ockert*

Sein Stück Inbild/Abbild 3d v3 (2005), was er selbst als „work in progress" bezeichnet zielt darauf ab Improvisation formal zu gestalten, ohne auf die Harmonieschemata des Jazz zurückgreifen zu müssen.
Das Stück ist für Trompete, Akustikgitarre, Schlagzeug und Live-Elektronik geschrieben. Die Gesamtdauer von 12 Minuten ist in 3 Einzelsätze unterteilt, deren jeweiliger Charakter durch ausnotierte Abschnitte sowie die unterschiedliche Wiedergabe des Gespielten durch die Live-Elektronik geprägt wird. (1. Satz „weit und ruhig", 2. Satz „zurück und vor" 3. Satz „dicht und nah") Während der Improvisation mit dem als Inspirationsquelle gedachten Material, werden die drei Instrumentalisten vom Computer

aufgenommen. Die sofort einsetzende und transformierte Wiedergabe der Samples erfolgt nach auskomponierten Listenstrukturen, welche die Abstände zwischen den Impulsen, deren Tonhöhen und die Lautstärke bei der Wiedergabe bestimmen.[17] Spielanweisungen wie: „Improvisation mit Material nur dann, wenn eigenes Instrument in der Live-Elektronik nicht hörbar"[18] ist ein Beispiel dafür, wie die Formstrukturen innerhalb der Sätze durch die Live-Elektronik bestimmt wird. Durch die festgelegten Klangtransformation über Listen, ist es möglich den Ablauf in einer Partitur festzuhalten. Im Gegensatz zu meinem Konzept wird ein Verschmelzen zwischen Instrumentalisten und Live-Elektronik angestrebt. Das soll dadurch erreicht werden, dass das Originalsignal des Spielers, sowie die Samples über den selben Lautsprecher ausgegeben werden.
Obwohl der Bereich der freien Soloimprovisation durch die Vorgabe von Notenmaterial und die Festlegung der Berechnungsalgorithmen für den Computeroutput verlassen wurde, hat mich an diesem Stück die Möglichkeit der Formgestaltung durch eine transformierte Wiedergabe des Gespielten beeindruckt. Besonders die Verwendung von Buffermaterial des 1. Satzes „weit und ruhig" als Rückführungselement im 2. Satz „zurück und vor" machte mir musikalische Wirkung von zeitlicher Verschiebung aufgenommenem Klangmaterials bewusst.
Diese Erkenntnis spiegelt sich in der späteren Entwicklung des Formteilgenerators meines Patch wieder, der im Kapitel „Formabschnitte und Formteile" besprochen wird.

## *3.2 Robert Rowe*

Robert Rowe entwickelte im Rahmen seines Ph.D. am MIT Media Laboratory um 1990 das Program Cypher. Leider liegt mir keine spielbare Software-Version von Cypher vor. Ich werde die Programmfunktionen daher der Dokumentation von Robert Rowe entnehmen. Eine detaillierte Beschreibung

[17] M.Ockert: Aufführungsvorgaben in der Partitur zu Inbild/Abbild 3D v3 (2005)
[18] Ebda. , II 3, (2. Satz, Minute 1:00)

über die Entwicklung seines interaktiven Musiksystems befindet sich in dem von ihm verfassten Buch „Interactive Music Systems – Machine Listening and Composing". An dieser Stelle meiner Arbeit soll lediglich der Ansatz dieser Software beschrieben werden, um Parallelen und Unterschiede zu meiner Entwicklung aufzuzeigen.
Cypher ist ein interaktives Computermusikprogramm welches für Komposition und Performance genutzt werden kann. Das Programm besteht aus zwei Haupteinheiten: Dem Hörer (Listener) und dem Spieler (Player). Der Player hat keine vorher gespeicherten Noteninformationen oder rhythmische Pattern, es gibt ausschließlich algorithmische Berechnungen für die musikalischen Antworten. Da dieses Programm bis 1991 entwickelt wurde, basiert der Datenaustausch auf dem seinerzeit üblichen Midi-Standard. Akustische Inputsignale müssen mit Hilfe eines Pitch-to-MIDI Konverters umgewandelt werden, dessen Ausgereiftheit und Präzision von Rowe in Frage gestellt wird.[19] Deshalb ist zur Eingabe von musikalischen Informationen die Verwendung von Midi-Instrumenten am besten geeignet. Rowe bezeichnet sein Programm als „Partner für ein Duett".[20] Er geht dabei soweit, das Spielen mit seinem Programm als Improvisation mit einem zweiten Spieler zu bezeichnen. Der Spieler soll folglich den Output seines Programms als absolut komplementär zu seinem eigenen Spiel erleben und nicht als Erweiterung oder Fortführung.
Dazu entwickelte Rowe in jahrelanger Arbeit ein komplexes Regelsystem. Eine graphische Programmoberfläche fordert den menschlichen Spieler vor Improvisationsbeginn auf, Verknüpfungen zwischen Eingangsereignissen und möglichen Programmreaktionen zu treffen. Diese Verknüpfungen können abgespeichert und zu einem späteren Zeitpunkt geladen werden. Außerdem besteht die Möglichkeit, diese von einem zweiten Mitspieler während der Performance zu verändern.[21]

[19] Vgl. R.Rowe: Interactive Music Systems: S. 15
[20] Siehe ebda.: S. 41
[21] Siehe ebda.: S. 73

Die Idee einen „Duettpartner" zu entwickeln soll auch in meinem Patch verwirklicht werden. Die aktuellen technischen Möglichkeiten bieten ein nahezu zeitlich unverzögertes Arbeiten mit echten Audiosignalen. Latenzzeiten zwischen 3-10 ms ermöglichen Audiosignale nahezu gleichzeitig aufzunehmen, zu analysieren  und musikalische Reaktionen als Audio-Output auszugeben. Auf die Möglichkeit für den Anwender im vorhinein in die Reaktionsroutinen des Programms einzugreifen habe ich verzichtet, um eine einfache Handhabung zu gewähren. Die Programmreaktionen werden durch den Audio-Input gesteuert, die Verknüpfungen zwischen Audio-Input und Patchsteuerung werden durch Zufallsberechnungen bestimmt.

# 4. Beschreibung des von mir entwickelten Patch

## *4.1 Vorüberlegungen*

Bei meinem Patch möchte ich keine Loops im Verhältnis zu einem Metrum aufnehmen, sondern einen konstanten Audio-Stream. Dieser wird während der Aufnahme nach den Parametern Tonhöhe, Lautstärke, Obertonspektrum, Geräuschanteil und auf perkussive Elemente hin analysiert. Die gewonnenen Ergebnisse werden dann genutzt um gespeicherte Sampleregionen in Abhängigkeit von neuem Inputmaterial verändert abzuspielen. So wäre es dann zum Beispiel möglich eine Einstellung zu treffen, dass wenn Töne über 800 Hz gespielt werden, nur Regionen mit hohem Geräuschanteil des im Buffer gespeicherten Audiostreams abgespielt würden. Da diese nicht unmittelbar nebeneinander liegen müssen, werden kleinste Audioeinheiten hintereinander abgespielt und es ergibt sich eine neuartige Struktur mit sehr vielen Schnitten.

### *4.2 Vorgehensweise bei der Umsetzung*

Für die Realisierung dieser Idee habe ich mich für die Audio-Programmiersprache Max/Msp (Abk.: Max) entschieden. Grund dafür ist, dass es innerhalb der Sprache selbst, und in der im Internet öffentlich und frei zugänglichen Objekt-Bibliothek[22] einige wichtige Grundbausteine gibt, die eine Umsetzung in einem vorhersehbaren Zeitraum möglich machen sollten. Das Max eigene Objekt „fiddle~" von Miller Puckette war für mich ein überzeugender Ansatz für die Übersetzung des Live-Audiomaterials in Datenpakete, welche notwendig sind zur musikalischen Analyse des Audio-Input und zur späteren Steuerung des Patch. Eine Recherche bei www.maxobjects.com zeigte, dass das Objekt „analyzer~" von Tristan Jehan, welches auf „fiddle~" basiert, die von mir gesuchten Informationen sofort in brauchbarer Weise ausgibt.
Für die Audio-Aufnahme, Speicherung und Wiedergabe bietet Max jede Menge brauchbare Werkzeuge (Objekte).
Bei der Verarbeitung des erheblichen Datenaufkommens durch die Analyse des Eingangssignals ist es notwendig geworden eine externe Objekt-Bibliothek zu benutzen. Hierbei verwende ich FTM 1.7.7, entwickelt am IRCAM Institut Paris. FTM bietet eine Max-Erweiterung zum Erstellen, Verarbeiten und Speichern von Tabellen und Matrizen an. FTM ist nach kurzer Installation komplett in Max integriert und ermöglicht zudem einen einfachen Datenaustausch, auch mit Max eigenen Objekten.[23] Mit den vorhandenen Objekten habe ich begonnen, das Patch zu erstellen.

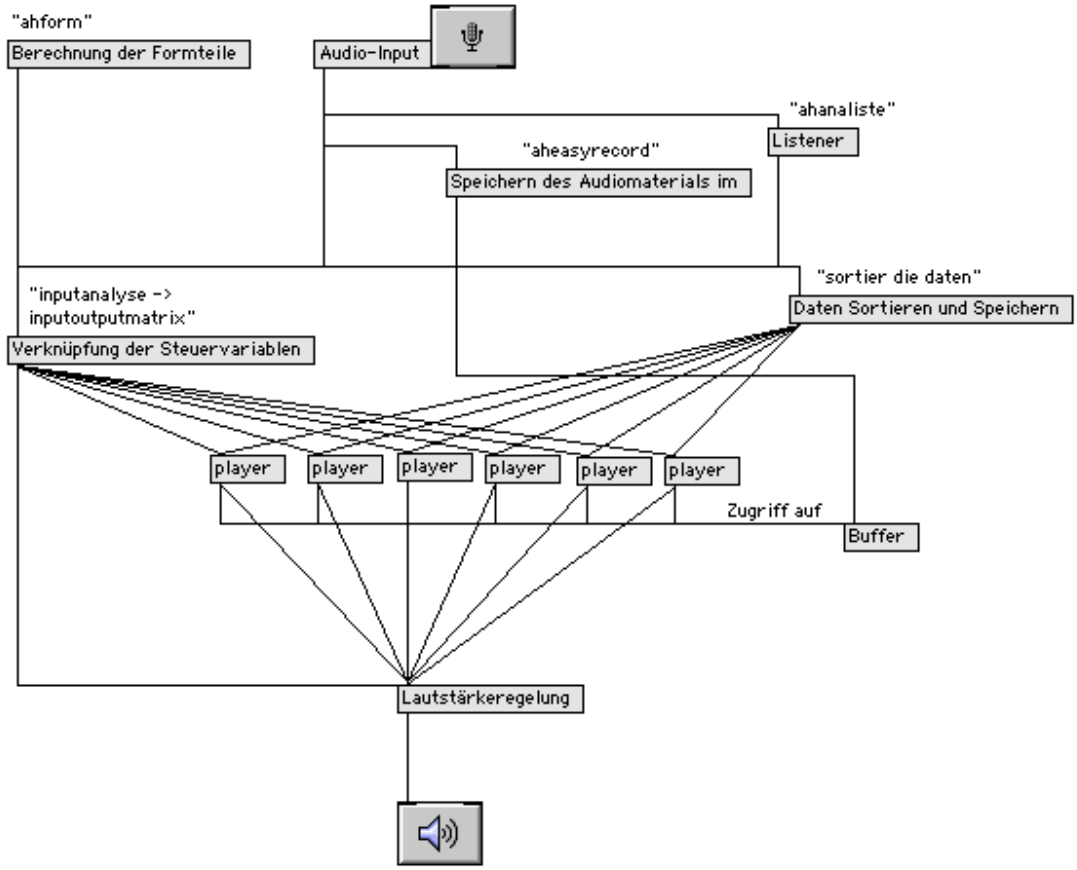
Um mathematische Zusammenhänge einfach zu gestalten und nicht zu tief in das Gebiet der Wahrscheinlichkeitsrechnung einzutauchen, habe ich mir bei Auswahloperationen zwei einfache Modelle ausgesucht, die in dem gesamten Patch Verwendung finden. Die „if..then.." Bedingung, eine eindeutige Zuordnung. „Wenn" etwas zutrifft, „dann" reagiere so. Die andere einfache

[22] http://www.maxobjects.com
[23] Auf der CD-Rom im Anhang befindet sich eine Beschreibung der Bibliothek in Ftm.longer2005.pdf

Möglichkeit ist die des Auslosens. „random 10" bedeutet: Ziehe per Zufall (an die interne Systemuhr gekoppelt, dadurch für uns nicht vorhersehbar) eine Zahl zwischen 0 und 9. Einer dieser 10 Wege wird dann eingeschlagen. Die Wahrscheinlichkeit für das Eintreten jedes Events ist gleich hoch. Diese Beschränkung bei einer Weiterentwicklung des Patch aufzuheben, bietet aus meiner Sicht ein hohes Entwicklungspotential.

### *4.3 Hauptkomponenten des Patch:*

Das von mir entwickelte Patch ist ein Sampler mit interaktiver Ausgabefunktion. Es kann grob in zwei Teile gegliedert werden. Einem Hörer (Listener), der den Audioinput analysiert und den Spielern (Player), die aus dem aufgenommenen und analysierten Material neue Strukturen formen.



Die Interaktion zwischen Sampleplayern und Eingangssignal ist abhängig vom derzeit aktiven Formteil. Der Formablauf wird im Objekt (Obj.) „ahform" bestimmt. Das eingehende Audiomaterial wird im Obj. „aheasyrecord" in einen Buffer mit maximaler Länge von 15 min. geschrieben. Die Echtzeitanalyse des Audiomaterials findet im Obj. „ahanaliste5" statt. Die festgehaltenen Werte werden in eine Haupttabelle geschrieben.
Zur Auswertung müssen diese im Subpatch (Subp.) „fmatsplitter" aufgeteilt werden. Die Positionen und Charakteristika der herausgefilterten Klangereignisse werden im Subp. „sortier die daten" in neuen Tabellen kopiert. Daraus ergeben sich Tabellen mit Informationen über Tonhöhenänderungen oder hervorstechende Attack-Positionen im Buffer. Aus diesen Tabellen können die verschiedenen Player (Subp. „hackplayer", „loopplayer", „attacplayer","kurzzeitplayer","cdhangplayer" und „pitchplayer) ihr Audiomaterial auswählen.
Die Steuerung der Player sowie deren Lautstärke ist zum Einen vom Formablauf („ahform") und zum Anderen vom eingehenden Audiomaterial (Subp. „inputanalyse") abhängig. So legt die Form eine Gewichtung der Player fest, jedoch ist das Lautstärkevolumen in einem gewissen Rahmen noch veränderlich. Variablen, die aus dem eingehenden Audiomaterial Steuerungsfunktionen ableiten, verändern ihre Bezugsquelle mit jedem Formteil (Subp „inputanalyse-> „inputoutputmatrix"").

Die Beschreibung der einzelnen Datenverarbeitungsgruppen findet nun in der Reihenfolge ihrer Entwicklung satt.

### 4.3.1 Die Echtzeitanalyse des Audiomaterials (ahanaliste5)

Grundlage für die Analyse des eingehenden Audiostreams ist das Objekt analyzer~ von Tristan Jehan. Analyzer~ kann in Echtzeit folgende Informationen über den eingehenden Audiostrom augeben.: Pitch (Tonhöhe in Hz), Loudness (Amplitude in dB), Brightness (Gewichtung des

Obertonanteils), Noisiness (Geräuschanteil) und Attack (plötzlicher Anstieg der Amplitude um 10 dB). Die Anzahl der ausgegebenen Werte pro Sekunde ist abhängig von der an den Computer angeschlossenen Audiohardware und dem Computer eigenen Prozessor. Die Zeitintervalle zwischen den Messungen sind nicht konstant, sodass immer ein Timecodebezug zum Audiomaterial hergestellt werden muss. Eine genaue Beschreibung des „analyzer~" befindet sich in der analyzer~.help, die mit dem Objekt zusammen auf der Hompage zum freien Download steht.[24]

Die Analysewerte werden in einer Tabelle (*fmat* ori) festgehalten werden.

| | timecode | pitch | loudness | brightness | noise | reg.attack | cal.bpm | bpm in ms |
|---|---|---|---|---|---|---|---|---|
| fmat | | | | | | | | |
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 722 | 8660.0 | 798.29047 | -42.813717 | 2268.4773 | 0.6566416 | 0.0 | 50.0 | 1200.0 |
| 723 | 8672.0 | 798.29047 | -43.419247 | 2328.7559 | 0.6788318 | 0.0 | 50.0 | 1200.0 |
| 724 | 8683.0 | 409.02524 | -43.613335 | 2514.0237 | 0.69183975 | 0.0 | 50.0 | 1200.0 |
| 725 | 8695.0 | 409.02524 | -44.807793 | 2864.435 | 0.7364321 | 0.0 | 50.0 | 1200.0 |
| 726 | 8707.0 | 387.7001 | -44.633705 | 2421.7693 | 0.6644037 | 0.0 | 50.0 | 1200.0 |
| 727 | 8718.0 | 387.7001 | -45.36826 | 2852.6409 | 0.7023559 | 0.0 | 50.0 | 1200.0 |
| 728 | 8730.0 | 387.7001 | -45.90845 | 2626.4814 | 0.7030237 | 0.0 | 50.0 | 1200.0 |
| 729 | 8739.0 | 387.7001 | -45.074093 | 2451.288 | 0.67822456 | 0.0 | 50.0 | 1200.0 |
| 730 | 8753.0 | 387.7001 | -45.727177 | 2641.0134 | 0.6783308 | 0.0 | 50.0 | 1200.0 |
| 731 | 8765.0 | 387.7001 | -40.355843 | 1940.5741 | 0.6365836 | 0.0 | 50.0 | 1200.0 |
| 732 | 8776.0 | 387.7001 | -23.458778 | 742.76807 | 0.43180096 | 0.0 | 50.0 | 1200.0 |
| 733 | 8788.0 | 173.3786 | -17.130613 | 660.00494 | 0.3771813 | 1.0 | 50.0 | 1200.0 |
| 734 | 8799.0 | 348.79144 | -16.248983 | 724.0737 | 0.34927803 | 1.0 | 50.0 | 1200.0 |
| 735 | 8811.0 | 348.79144 | -16.126886 | 758.9321 | 0.36714244 | 1.0 | 50.0 | 1200.0 |
| 736 | 8823.0 | 348.79144 | -15.75513... | 786.72003 | 0.37879223 | 1.0 | 50.0 | 1200.0 |
| 737 | 8834.0 | 348.79144 | -16.024075 | 784.4511 | 0.3709367 | 1.0 | 50.0 | 1200.0 |
| 738 | 8846.0 | 348.79144 | -15.982635 | 742.62555 | 0.36646438 | 1.0 | 50.0 | 1200.0 |
| 739 | 8858.0 | 348.79144 | -16.231209 | 748.27405 | 0.35572225 | 1.0 | 50.0 | 1200.0 |
| 740 | 8869.0 | 348.79144 | -16.191673 | 756.4996 | 0.3580851 | 1.0 | 50.0 | 1200.0 |
| 741 | 8879.0 | 348.79144 | -16.567442 | 782.6318 | 0.3769737 | 0.0 | 50.0 | 1200.0 |
| 742 | 8893.0 | 348.79144 | -16.809645 | 739.9366 | 0.36897552 | 0.0 | 50.0 | 1200.0 |
| 743 | 8904.0 | 348.79144 | -17.102194 | 720.91064 | 0.357741 | 0.0 | 50.0 | 1200.0 |
| 744 | 8916.0 | 348.79144 | -17.564163 | 716.0678 | 0.35883158 | 0.0 | 50.0 | 1200.0 |
| 745 | 8927.0 | 348.79144 | -17.986238 | 753.0524 | 0.35860044 | 0.0 | 50.0 | 1200.0 |
| 746 | 8939.0 | 348.79144 | -18.256918 | 694.08765 | 0.34054655 | 0.0 | 50.0 | 1200.0 |
| 747 | 8949.0 | 348.79144 | -18.51436 | 685.01013 | 0.33756965 | 0.0 | 50.0 | 1200.0 |
| 748 | 8963.0 | 348.79144 | -18.651947 | 726.90594 | 0.3484214 | 0.0 | 50.0 | 1200.0 |
| 749 | 8974.0 | 348.79144 | -18.905338 | 698.082 | 0.34409958 | 0.0 | 50.0 | 1200.0 |
| 750 | 8986.0 | 348.79144 | -19.079548 | 686.79205 | 0.33991587 | 0.0 | 50.0 | 1200.0 |

[24] download: http://web.media.mit.edu/~tristan

Parallel zur Analyse wird der Audio-Input im Subp. „aheasyrecord“ in den Buffer geschrieben.

### 4.3.2 Das Sortieren der gesammelten Daten

Da es innerhalb der *fmat* (FTM 1.7.7) nicht möglich ist, nach bestimmten Werten zu suchen und sich die dazugehörigen Koordinaten der Tabelle ausgeben zu lassen, muss die Tabelle in Listen zerlegt (Subp. „fmatsplitter“, und diese mit dem Max eigenen Werkzeug "zl sub" untersucht werden. Bei Listen ist die Beschränkung auf 256 Events zu beachten. Deshalb wird alle 256 Analysewerte (ca. alle 3 sek.) eine Auswertung der angesammelten Daten vorgenommen.

Folgende Untersuchungen finden in dem Subp. „sortier die daten“ statt.:

a) Die durchschnittliche Lautstärke (innerhalb der letzten 256 Events) wird ermittelt.

b) Werte die darüber liegen werden als "laut" deklariert und ihre Position in die *fmat* "anloud" eingetragen. Diese Tabelle enthält die Lautstärkeangabe in dB sowie dessen Position im Sample.

c) Die lauten Stellen werden daraufhin auf weitere Eigenschaften untersucht. So wird z.B. der Geräuschanteil unterschieden. In die *fmat* "annoise1" werden Angaben über geräuscharme und in die *fmat* "annoise3" geräuschintensive Sampleposition eingetragen. Übergänge werden in der *fmat* „annoise2“ festgehalten.

d) Des weiteren wird untersucht ob sich die Tonhöhe verändert hat. Diese Veränderungen werden in "anpitchbox" notiert. Genauso wird mit dem Bright-Faktor, der Auskunft über den Obertonanteil des Eingangsignals gibt, verfahren.

e) Perkussive Stellen werden direkt aus der *fmat* „ori" ausgelesen und in Verknüpfung mit der Sampleposition in "anattacbox" gespeichert.

### 4.3.3 Die Player

Zur Zeit ist das Patch auf sechs Player zugeschnitten. Dies lässt sich ohne Probleme erweitern.

Eine wichtige Eingrenzung muss zunächst vorgenommen werden: Aufgrund der vielen verschieden Formen von Klangsynthese, die mit dem Computer möglich sind, könnte man sehr unterschiedliche Player entwickeln.[25] Weil diese den Output erzeugen, beeinflussen sie den Sound des Patch grundlegend. Ich habe mich entschlossen, nur mit den Samples des eingehenden Audiomaterials als solches zu arbeiten. Meine Player sind alle Sampleplayer und greifen ausschließlich auf den sich in diesem Moment füllenden Buffer zu. Selbst unter dieser Beschränkung eröffnet sich ein großes Feld an Ausdrucksmöglichkeiten.

#### hackplayer

Der „hackplayer" ist der von mir zu erst entwickelte Player. Sein Aufbau ist sehr einfach gehalten. Dadurch, dass er von Anfang an in dem Patch integriert war, ist sein Sound so prägend gewesen, dass ich ihn nahezu in seiner ursprünglichen Form belassen habe. Sein Samplematerial sucht der „hackplayer" per Zufall innerhalb eines bestimmten Ambitus aus. Dazu greift er auf die analysierten Tonhöhen aus der *fmat* „anpitch" zu. Bei einem erkannten Attack am Audio-Input werden  aus einem Drittel der in der Tabelle stehenden Werte durch eine Randomfunktion beliebig viele ausgewählt. Durch die Variable „oo2", die als einzige nachträglich hinzugefügt wurde, wird innerhalb dieser Auswahl ein Sample abgespielt. Diese Samples befinden sich immer in einem bestimmten Ambitus, da sie in der nach Tönhöhen geordneten Tabelle „anpitch" nahe beieinander liegen. Durch Veränderung der Lautstärke des Eingangssignals verschiebt sich der Auswahlbereich in der Tabelle und somit die Tonhöhe. Die Länge der

[25] M.Neukom: Signale, Systeme und Klangsynthese

gespielte Sampleausschnitte sind an den Geräuschanteil des Eingangssignals geknüpft. Die Länge liegt zwischen 0 ms und 6000 ms.

### loopplayer

Die Grundidee zum „loopplayer" basiert auf der Idee sehr kurze Loops (20-30ms Länge) zu überlagern. Da ich hier eine große Veränderung des Klanges beim Abspielen des Samples erreichen wollte, habe ich drei Granular-Player in den „loopplayer" integriert. Die Granular-Player basieren auf dem Prinzip, dass ein relativ kleiner Sampleabschnitt (Grain) als Loop abgespielt wird. Um einen dichteren Klang zu erreichen, wird der gleiche Bereich nochmals um die Hälfte der Zeit versetzt darüber gelegt. Per Hüllkurven wird zwischen beiden Signalen übergeblendet (Subp.„loopfadeplayer"). Die Player suchen ihre Samplepositionen in der Tabelle „annoise2", welche die Übergangsbereiche von Ton zu Geräusch innerhalb des Buffers enthält.
Die Auswahl der Samplepositionen ist hier wie beim „hackplayer" konstruiert, jedoch wird die Länge der Grains per Zufall in einem Bereich zwischen 20 und 30 ms gewählt.
Eine Besonderheit in diesem Player spiet das Subpatch „voicevol". Der „loopplayer" ist der einzige Player, der in sich eine Lautstärkesteuerung eingebaut hat. Diese ist dafür da, die drei Granular-Player immer neu zu gewichten. Hierbei ist eine direkte Verknüpfung mit dem Eingangssignal hergestellt. Der linke Player ist an die Lautstärke gekoppelt: Bei Überschreitung der Grenze von −15 dB wird per Zufall im Rahmen von 50 Schritten eine neue Lautstärke (Skala: 0-127) festgelegt. Die Fadezeit wird proportional zum Geräuschanteil des Eingangssignals verlängert. Der mittlere Player wählt seine neue Lautstärke in vollem Umfang zufällig bei Analyse eines sehr obertonarmen Eingangssignals. Die Fadegeschwindigkeit ist hierbei an den selben Faktor (bright) gekoppelt. Der rechte Player wählt bei einem Attack im Eingangssignal einen neuen Pegel, der sich immer im

oberen Lautstärkedrittel befindet. Die Fadegeschwindigkeit ist wie bei dem mittleren Player an den „bright" Faktor gekoppelt.
Das Subpatch „mover", in Verbindung mit dem Objekt „ahsizemod", sorgen zusätzlich für eine Bewegung innerhalb des Buffers. Ohne diese Bewegung wäre der Klang sehr statisch, denn trotz der Lautstärkeverschiebungen unter den Granular-Playern würde es keine Entwicklung geben. Die Tonhöhe sowie der ausgegebene Geräuschanteil blieben konstant. Ein wirklicher Bezug zum aufgenommenen Sample wird erst dadurch hergestellt, dass das Sample über längere Strecken durchfahren wird. So entsteht eine zeitlich gedehnte Wiedergabe des Bufferinhalts. Das Subp. „mover" gibt das Tempo, sowie die Richtung vor, in der sich der Player durch das Sample bewegen soll. In „ahsizemod" wird dies in Steuerungsbefehle für den „loopfadeplayer" umgesetzt.

**attacplayer**

Der „attacplayer" stellt den Versuch eines auf Rhythmuspattern basierenden Players da. Dieser Player zieht per Zufall jeweils zwei Samples aus der *fmat* „anattac" und der *fmat* „annoise3". Deren Abspieltonhöhe ist an die Variablen ubd1 bis ubd8 geknüpft, welche in der „inputoutputmatrix" Eingangsereignissen zugeordnet werden.(Siehe: 4.3.5 Die Inputanalyse) Die Impulse zum Abspielen werden im Sup. „rhythmus -> ah4slotseq2" generiert. Das darin enthaltende Objekt „ahgeneratepatt01" fungiert als Patterngenerator. Hier werden aus sechs möglichen Zeitabständen der Events zueinander per Zufall Listen (Grooves) zusammengestellt. Die möglichen Abstände betragen 1 (wie ¼ Note), 2 (wie ½ Note), 0.5 (1/8 Note), 0.25 (1/16 Note) und 0.125 (1/32 Note). Das in „ahanaliste5" ermittelte Tempo kann mit diesen Faktoren multipliziert werden und das Pattern bekommt so einen aktuellen Metrumbezug. Schwachstelle hierbei ist bisher die Tempoerkennung des Audio-Input, welche zur Zeit einfach an die analysierten Attacks gekoppelt ist. Damit wird davon ausgegangen, dass der

Audioinput immer nur ¼ Noten spielt bzw. betont. Das Festhalten der Attacks und der ermittelten BPM (Beats per Minute) in der *fmat* „ori" soll bei Weiterentwicklung des Patch, eine bessere Metrumerkennung möglich machen. Das Abspielen und die Erstellung der neuer Pattern wird im „attacplayer" durch die Variable oo2 ausgelöst, das Stoppen durch oo7. Damit ein Groove auch für längere Zeit konstant bleibt gibt es eine Sperre, die eine Veränderung durch die Variablen verhindert. Diese wird nach 20 ausgegebenen Events eingestellt und bei Erreichen eines neuen Formteils aufgehoben. Dann werden auch die bisher verwendeten Rhythmuspattern gelöscht, sodass bei einem erneuten Start des Players neue Rhythmen verwendet werden können.

### kurzzeitplayer

Der „kurzzeitplayer" ist bisher der einzige Player, der nicht auf die analysierten Werte in den Tabellen „an…" zugreift. Er bedient dich aus dem noch nicht analysierten Material der *fmat* „ori". Seine Aufgabe stellt eine Art Reflex dar, ein unüberlegtes (unberechnetes) Reagieren auf gerade eingegangene Impulse. Dabei funktioniert er wie ein variables Delay mit Pitch-Funktion. Diese drei veränderlichen Faktoren Delayzeit, Pitch-Up und Pitch-Down bergen aber erhebliche Ausdrucksmöglichkeiten.

### cdhangplayer

Die Idee zu diesem Player basiert auf einem akustischen Ereignis, welches entstehen kann, wenn eine zerkratzte Compact Disk in einem CD-Player abgespielt wird. Dieser kann beim Abspielen dann an einer Stelle „hängen bleiben". Bei dem Versuch an dieser Stelle vorwärts zu spulen, ändert sich oftmals nur die Länge der in Schleife gespielten Stelle. Dieses Phänomen hat

den prägnanten Charakter eines digitalen Fehlers, sodass ich versucht habe, dies in einen meiner Player zu integrieren. Der „cdhangplayer" sucht aus allen Werten innerhalb der *fmat* „loud" per Los eine Stelle, mit einer Länge zwischen 200 und 2400 ms aus und wiederholt diese. Bei Eintreffen eines Auslösers über die Variable oo2 wird die Länge ungefähr halbiert. Dies findet so oft statt, bis die Samplelänge unter 20 ms liegt. Dann beginnt der Vorgang von neuem.

**pitchplayer**

Die Steuerung des „pitchplayer" ist direkt mit dem analysierten Pitch des Audio-Input verbunden. Ein kurzer Sampleauschnitt (50 ms) soll durch ihn aus seinem ursprünglichen Zusammenhang gerissen und parallel zum Audioinput sequenziert werden. So entsteht eine Doppelung des Inputsignals mit einer anderen Klangfarbe.

### 4.3.4 Die Formabschnitte und Formteile

Im Laufe der Entwicklung des Patch hat sich gezeigt, dass der Zugriff auf alle analysierten Daten des Audio-Stream aus der bisherigen Spiel-Session für die Weiterentwicklung der Improvisation hinderlich sein kann. Grund dafür ist, dass Material und Motive (Sampleauschnitte) des Anfänglichen stark rückführend wirken. Folglich ging es darum einen Steuerungsmechanismus zu finden, der eine Weiterentwicklung der Improvisation zulässt. In dem Objekt "ahform" wird versucht, dies zu entwerfen.
Somit stellt „ahform" eine übergeordnete Einheit dar, die auf das bereits konstruierte Gebäude gesetzt wird, um der Stagnation entgegen zu wirkend. In meinem Objekt „ahform" werden beim Laden des Patch durch Randomfunktionen Zeiten ermittelt, welche die Längen der Formabschnitte und die Grenzen der gesamten Spiel-Session vorgeben. Da äußere Umstände

oft eine zeitliche Begrenzung vorgeben, kann an dieser Stelle der Anwender einen groben Zeitrahmen bestimmen. „ahform 6 4 70 30 8" bedeutet, es gibt maximal 9 Formabschnitte (mindestens 4 + random 6 ->0-5), welche maximal 99 Sekunden (mindestens 30 + random70 ). Die letzte Zahl gibt an, wie viele Formteile im Subp. „inputanalyse" zur Verfügung stehen. Formabschnitte geben dabei nur Zeitdauer vor, nicht was in ihnen geschieht! Der Ablauf kann in „coll" eingesehen werden. Dort befindet sich die Aufzählung der Formabschnitte mit der jeweiligen Dauer.
Beim Übergang von einem Formabschnitt zum Nächsten, wird erst per Zufall ermittelt, welche Verhaltensformen (Formteil) die Lautstärkeverhältnisse der Player annehmen werden. So kann es vorkommen, dass zwei verschiedene Formabschnitte die selbe Gewichtung der Player zeigen, da sie den selben Formteil zugewiesen bekommen haben. Jedoch wird anderes Material verwendet, da die Analyse-Listen bei einem Formabschnittwechsel stets gelöscht werden.  Das Löschen der Listen ist für mich keine endgültige Lösung. Bei einer Weiterentwicklung des Patch soll die Möglichkeit bestehen auch auf Material aus vergangenen Formteilen zuzugreifen. Daher habe ich die *fmat* „an..box" als Container von allem analysierten Material vorgesehen, auf die bis jetzt aber noch nicht wieder zugegriffen wird. Aus diesen könnten Werte zurück in die Listen „an.." kopiert werden, in denen die Player ihre Samplepositionen ablesen.
Außerdem führt ein Formabschnittwechsel zu einer neuen Verknüpfung der Steuerparameter für die Player innerhalb der „inputoutputmatrix".

### 4.3.5 Die Inputanalyse

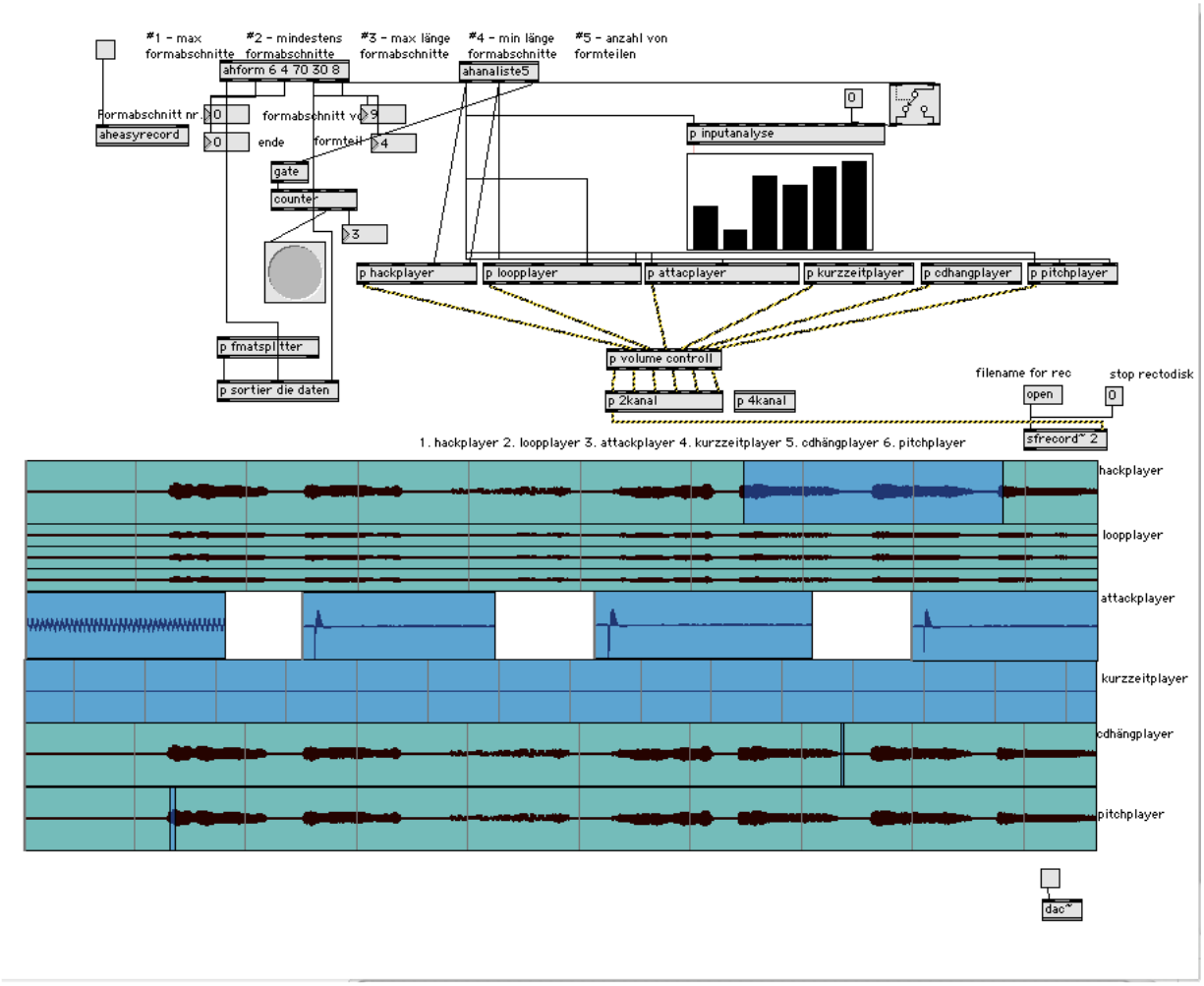
Das Subpatch „inputanalyse" enthält die Reaktionstypen der verschieden Formteile. Diese sind als Objekt "ahformteil" zu erkennen. Ihre veränderlichen Parameter sind die maximale und minimale Fadegeschwindigkeit der Lautstärkeregler am Hauptmixer, sowie die

Gewichtung der Player innerhalb des Formteiles. So bedeutet "ahformteil 3 1 4 2 1 4 1 3", dass die Lautstärkeveränderungen relativ langsam vor sich gehen, weil sie maximal 4 Sekunden (mindestens 1 Sek. + random3) und mindestens 1 Sekunde dauern. Die lautesten und somit dominatesten Player sind hierbei der erste und vierte am Mixer (hackplayer und kurzzeitplayer), gefolgt vom zweiten(loopplayer). Die Anderen sind innerhalb dieses Formteils nur sehr leise zu hören.
Die Berechnung neuer Lautstärken erfolgt zufällig bei Eintreffen eines Auslösers über die Variable oo3. Der Dynamikspielraum ist durch den Gewichtungsfaktor 1-4 vorgegeben. Auf einer Skala von 0 bis 127 (ein Relikt des Midi-Standard) entspricht die Gewichtung „1" 0-30, „2" 37-67, „3" 67-97, „4" 97-127. Die Variable oo3 ist innerhalb der „inputoutputmatrix" mit einem Event des eingehenden Audiomaterials verknüpft. So wird eine Rückbindung zum aktuellen Spielgeschehen hergestellt.

Die größten Vorstellungsschwierigkeiten hatte ich bei dem Entwurf für die Steuerungsmechanismen des Patch durch den Audioinput.
Ich war anfangs häufig gezwungen, bestimmte Reaktionen dadurch hervor zu rufen (z.B. verändern der Player-Lautstärke), dass ihre Steuerung direkt an die Werte der Inputanalyse gekoppelt wurde. So entstand aber eine erhebliche Einschränkung bezüglich der Idee der gegenseitigen Beeinflussung. Denn eine Verknüpfung von einer Aktion des Spielers mit einer unmittelbar folgenden Reaktion des Patch wird schnell erkannt und kann mitunter zu einem unerwünschten Lerneffekt beim Spieler führen. Vorschnell kann eine Rollenverteilung entstehen, bei dieser der Spieler nur noch bemüht ist die Reaktionsmuster des Patch zu ergründen und so bald er diese erkannt hat, wird er versuchen das Patch zu manipulieren.
Ob es nun dem Spieler eine Macht über das Patch verleiht oder das Patch die Macht über den Spieler erlangt hat, kann nicht genau gesagt werden. Jedoch sehe ich darin eine Gefährdung für die Weiterentwicklung der Improvisation.

Um diese direkten Verknüpfungen zu vermeiden kam ich auf die Idee einer Input-Output-Matrix.
Die „inputoutputmatrix" ist die Steuerzentrale des Patch. In ihr wird das aktuelle Klanggeschehen vom Spieler in Steuersignale für die Player und deren Lautstärke gewandelt. Damit eine Änderung der Formabschnitte auch eine Veränderung des Patchreaktionsverhaltens beinhaltet, werden hier die steuernden Parameter stets neu verknüpft. Dabei werden die Eingangsereignisse (Subp. „meanbangs") durch Randomfunktionen den Variablen oo1 bis oo8 sowie ubd1 bis ubd8 zugeordnet. Diese Variablen steuern dann zum Beispiel Funktionen der Player. Ein sich ergebender Nachteil dessen ist, dass es auch zu unsinnigen Zufallsverknüpfungen kommen kann, welche Reaktionen des Patch blockieren oder sogar Programmabstürze verursachen können.
Da dieser Programmteil so spät entstanden ist, gibt es immer noch Player, in denen es eine direkte Verknüpfung von Werten der Inputanalyse mit Steuerungsmechanismen gibt. Diese Player weisen aber gerade dadurch ihre spezielle Charakteristik aus, welche bei einer zufälligen Verknüpfung wiederum gefährdet wäre. So entschied ich mich, den „loopfadeplayer" in seiner ursprünglichen Form zu belassen.

## 4.4 Die Oberfläche




## 4.5 Beschreibung der konstruierten Objekte

ahanaliste5: Dieses Objekt enthält das External „analyzer~". Der Audio-Input am Eingang „1" der Soundkarte wird in den analyzer~ geroutet. Ein Timer startet zeitgleich mit der Aufnahme des Buffers und gibt einen Zeitbezug zwischen Messwerten und der Bufferposition des Samples. Die Analysewerte werden als Paket „packed-analyzer" ausgegeben. Analysierte Attacks werden einzeln ausgegeben, da sie teilweise direkte Funktionen in den Playern steuern. Am dritten Ausgang wird für jeden abgeschlossen Analysedatensatz (ca. alle 10-20 ms) ein Signal (bang) ausgegeben. Diese sind notwendig um

eine Zählung der Werte durchzuführen, damit keine längeren Listen als mit 256 Events erstellt werden.

ahbpmcounter: Ein Objekt was die Zeitabstände zwischen eingehenden „bangs" ermittelt und in Beats per Minute umrechnet. Dafna Naphtali hat dieses Objekt entwickelt.[26]

aheasyrecord: Das eingehende Audiomaterial wird im Obj. „aheasyrecord" in einen Buffer mit maximaler Länge von 15 min. geschrieben. Die Buffergröße ist abhängig von der Größe des Arbeitsspeichers (RAM) des Computers.

ahfades1: Dieses Objekt sendet einen kontinuierlichen integer Datenstrom von einem int. Wert a zu einem int. Wert b innerhalb einer bestimmten Zeitdauer c in ms.

ahform: In meinem Objekt „ahform" werden beim Laden des Patch durch Randomfunktionen Zeiten ermittelt, welche die Längen der Formabschnitte und die Grenzen der gesamten Spiel-Session vorgeben.
Formabschnitte geben dabei nur Zeitdauern vor, nicht was in ihnen geschieht!

ahformteil: Regelt die Lautstärke am Hauptmixer. Enthält als wichtigsten Bestandteil „ahintensität".

ahintensität: Führt die Berechnung neuer Lautstärken für den Hauptmixer aus. Der Dynamikspielraum ist durch den Gewichtungsfaktor 1-4 vorgegeben. Auf einer Skala von 0 bis 127 entspricht Gewichtung: „1" 0-30, „2" 37-67, „3" 67-97 „4" 97-127. Innerhalb des sich ergebenden Spielraumes werden zufällige Werte (random 0-30) ermittelt.

[26] Siehe R Rowe: Machine Musicianship, S. 229

ahpassperc: Ein Objekt, welches einen bestimmten Prozentsatz an „bangs" durchlässt. Umgesetzt nach den Beschreibungen im Max Tutorial 29.

### *4.6 Funktionen der wichtigsten Subpatches*

fmatsplitter: Kopiert die *fmat* „ori" nach 256 Events in die *fmat* „smp11", löscht „ori" und gibt sie sofort wieder zum füllen frei. Anschließend wird „smp11" in seine einzelnen Spalten geteilt und als jeweilige *fmat* gespeichert. (pitch, loud, bright, noise, att) Diese *fmat* enthalten nur eine Spalte und 256 Zeilen und können dadurch ohne Datenverlust in Listen umgewandelt werden.

sortier die daten: Die Positionen und Charakteristika herausgefilterter Klangereignisse werden im Subp. „sortier die daten" in neuen Tabellen umgeschrieben. Daraus ergeben sich Tabellen mit Positionsangaben von überdurchschnittlich lauten Stellen (anloud), Tonhöhenänderungen (anpitch), Obertonanteilveränderungen (anbright) Attack-Positionen (anattc) und drei Abstufungen über Geräuschanteile (noise1 – wenig Geräuschanteil, noise2 – mehr Geräuschanteil, noise3- überwiegend Geräuschanteil) im Buffer. Diese werden anschließend sortiert. „anpitch" nach Tonhöhe, die anderen nach Intensität ihres jeweiligen Messwertes.

findattacs: Sucht in der *fmat* „attc" nach Attacks und gibt deren Sampleposition als Liste aus.

loudnesscheck: Ermittelt den durchschnittlichen Lautstärkewert der letzten 256 Events in dB. Und findet alle Samplepositionen in denen die Lautstärke darüber lag. Diese werden als „laute" Stellen bezeichnet.

pitchcheck: Ermittelt an den „lauten" Stellen ob eine Veränderung der Tonhöhe vorliegt.

brightcheck: Ermittelt an den „lauten" Stellen ob eine Veränderung des Bright-Faktors vorliegt.

noisecheck: Ermittelt an den „lauten" Stellen ob eine Veränderung des Geräuschanteils am Sample vorliegt und sortiert in drei Kategorien: noise1 – wenig Geräuschanteil, noise2 – mehr Geräuschanteil, noise3- überwiegend Geräuschanteil.

delete to forgetmaterial: In diesem Subp. wird bei Eintritt in einen neuen Formteil, eine Löschung der bereits gesammelten Analyseergebnisse vorgenommen. Somit kann kein Player gezielt auf ein zurückliegendes Event zugreifen. Dies ist eine Maßnahme um ein Datenüberangebot zu verhindern und eine mögliche Entwicklung der Improvisation zu unterstützen.

inputanalyse: Das Subpatch „inputanalyse" enthält die Reaktionstypen der verschieden Formteile.

inputoutputmatrix: Die „inputoutputmatrix" ist die Steuerzentrale des Patch. In ihr wird das aktuelle Klanggeschehen vom Spieler in Steuersignale für die Player und deren Lautstärke gewandelt.

2kanal: Dort wird der Output des Patch an die Ausgänge 1 und 2 der Soundkarte geroutet.

# 5. Diskussion der Ergebnisse

## *5.1 Beschreibung der Spielsituation*

Um unabhängig von Selbstversuchen Eindrücke über das Spielverhalten des Patch zu gewinnen habe ich drei Musiker aus der Musikszene des Zeitgenössischen Jazz zum Spielen mit meinem Patch eingeladen.

Almut Kühne – Stimme
Berit Jung – Kontrabass
Matthias Schubert – Tenorsaxophon

Damit vergleichbare Ergebnisse entstehen können, haben alle drei Spieler das selbe Patch (Version 0.58c) benutzt. Außerdem haben alle die selben technischen Geräte zur Verfügung gestellt bekommen, welche ich vor dem Spielen eingerichtet habe.

| | |
|---|---|
| *Mikrofon*: | AKG SE 300B mit der CK91 Kapsel (C 391 B) |
| *Audiointerface*: | Motu-828-MKII, dessen Vorverstärker eingesetzt wird |
| *Computer*: | Intel MacBook, Duo 2 Ghz, 1 GB RAM, Max/MSP 4.6.1 FTM 1.7.7 Beta |
| *Lautsprecher*: | Klein & Hummel O 110 Aktiv-Monitor |

Aus organisatorischen Gründen war es nicht möglich, alle Aufnahmen am selben Ort durchzuführen. Außerdem weicht durch die unterschiedliche Bauart und Spielweise der Instrumente, der Mikrofonabstand zur Klangquelle ab. Dadurch kann der Raumanteil auf den Aufnahmen variieren.

Die beiden Audiosignale, das vom Instrumentalisten und das vom Patch, können vom Patch selbst simultan mitgeschnitten und direkt auf der Festplatte des Computers gespeichert werden. Die Stereo-Audiodatei enthält

links das akustische Originalsignal des Spielers, und rechts den generierten Audio-Output des Patch.
Während der Aufnahmesession ist es leider auch zu Programmabstürzen gekommen. Unerwartet beendet wurden die Aufnahmen almut02, mat05. Sofort nach Programmstart kam es bei den Aufnahmen mat02, mat04, mat07 zu einer Programmbeendung. Ich habe diese Aufnahmen dennoch in meiner Zählung behalten, auch wenn sie kein Audiomaterial enthalten und deshalb nicht auf der CD im Anhang zu finden sind. Programmabstürze sollen in dem Rahmen dieser Untersuchung erwähnt werden, auch wenn ich diese auf ein Hardwareproblem zwischen MacBook und Motu828 zurückführen würde. Grund dafür ist, dass diese Fehler nicht nachvollziehbar an einer bestimmten Programmroutine auftreten und bei Verwendung der MacBook internen Soundkarte bisher ausblieben.

Jeder Spieler hat mindestens drei Improvisationen mit dem Patch gespielt. Anschließend habe ich mit jedem Spieler ein Interview geführt.

### *5.2 Auswertung der Mitschnitte*

Auf folgende Kriterien hin sollen die Aufnahmen untersucht werden:

- Kommunikation zwischen Spieler und Patch
- Die musikalische Eigenständigkeit des Patch
- Erkennbare Reaktionsmuster
- Entwicklung der Improvisation
- Verwandtschaft des Patch-Output mit dem Inputmaterial
- Das Verhältnis des menschlichen Spielers zur Maschine

Die herausgegriffenen Stellen sollen dabei in einen Zusammenhang mit den Aussagen der Spieler in ihrem Interview gebracht werden.

### *5.3 Kommunikation zwischen Spieler und Patch*

Die Untersuchung der Kommunikation macht in erster Linie nur Sinn, wenn Bezug zu meiner anfänglich geäußerten These genommen wird: „Die vom Spieler erzeugten Klänge und Strukturen sollen dabei gleichzeitig die Ausgabe des Patch beeinflussen und rückwirkend beeinflusst die Klangausgabe des Patch den Spieler."

Zu Beginn jeder Improvisation mit meinem Patch kann keinerlei Kommunikation stattfinden, da der Vorrat an Audiomaterial, mit dem das Patch spielt, zuerst aufgenommen und analysiert werden muss. Dies betrifft mindestens die ersten ca. 2-3 Sekunden (abhängig von der Hardware), nach denen die Analyse der bisher 256 empfangenen Daten stattgefunden hat. Dann steht Spielmaterial für die Player zur Verfügung. Als Hörbeispiel eignen sich in diesem Fall alle Anfänge der Improvisationen, da das Patch auf dem Prinzip der Verarbeitung des Eingangssignals aufbaut. Folglich kann es nie passieren, das der Spieler zu Beginn eine Idee bringt und das Patch in dem Moment etwas völlig anderes spielt und somit eine musikalische Annäherung statt finden muss. Bei allen Aufnahmen beginnt das Patch programmbedingt nach dem Spieler und konfrontiert ihn gezwungenermaßen mit sehr jungem, gerade gespieltem Material. Eine Ausnahme stellt der „kurzzeitplayer" dar, da er nicht auf die Ergebnisse der Analyse angewiesen ist, er kann schneller reagieren. Am Beispiel almut01 lässt sich das gut nachvollziehen. Almut beginnt mit einem lang ausgehaltenen Ton, der vom „kurzzeitplayer" nach 2 Sekunden sofort, aber ohne die eigentliche Attackphase repetiert wird. Im Anschluss hört man einen Fade-In vom „cdhackplayer". Almut singt weiter lange Töne. Diese werden vom „cdhackplayer" aufgegriffen, die original Tonhöhen bleiben erhalten. Das Material wird bis Minute 0:40 konsequent in dieser Form verarbeitet. Analogien dazu finden sich am Beginn von Improvisation mat09. Hier wird das bisher gespielte Material bis Minute 2:00 immer nur  geschnitten und dynamisch verändert wiederholt. Nachteil dessen ist, dass der Spieler darin seinen eigenen Input erkennt und sofort

beginnt über die Materialverarbeitung im Patch nachzudenken. Wie die Sängerin Almut Kühne später erklärt.: „Man möchte herausfinden, was er mit dem Material macht." In den Hörbeispielen mit Matthias Schubert wird deutlich, wie Schubert bei jeder Improvisation streng sein Material limitiert, um die Reaktionen und Verarbeitungsmöglichkeiten des Patch zu ergründen. Bei der ersten Aufnahme (mat01) verwendet er nur sonisches Spielmaterial wie Rauschen oder Gurgeln, bei mat03 benutzt er nur Spaltklänge und Flageoletttöne, bei mat05 verarbeitet er vorwiegend motivisches Tonmaterial. Die Erkenntnis, dass er mit diesen sich selbst gesetzten Materialbeschränkungen womöglich versucht das Patch zu erkunden, kam mir erst während des Folgegesprächs, als wir über rhythmische Aspekte sprachen und er daraufhin noch einen Spielversuch zu dieser Thematik starten wollte. Die Gefahr, dass das Patch dadurch von Anfang an als rezessiver Partner betrachtet wird ist nun in Version 0.58c leider gegeben. Um von Anfang an eine gleichberechtigte Kommunikationsbasis herstellen zu können, müsste es einen Weg geben, ein musikalisches Thema zu entwerfen und den Spieler damit zu konfrontieren.

Jedoch ist ein Anfang nicht maßgeblich bindend für den weiteren Verlauf der Improvisation. Ein Ausschnitt aus der Improvisation almut03 bei 3:50 zeigt, wie sich das Patch dem Gestus der von der Sängerin vorgegebenen Improvisation anpasst. Jedoch bei Minute 5:07 findet ein klarer Rollenwechsel statt. Die Sängerin übernimmt eindeutig einen Ton vom Output des Patch. Zu Beginn der selben Aufnahme (1:06) hört man einen Rhythmus der vom Patch schon seit einigen Sekunden gespielt wird. Hier versucht die Sängerin darauf einzugehen und ein neues Tempo vorzugeben. Dies gelingt ihr nicht. Sie lässt davon wieder ab und beginnt mit langen fallenden Tönen zu experimentieren.

Obwohl Matthias Schubert im Interview angegeben hat, er selbst spiele zu dominant und würde nicht genug auf das Patch eingehen, finden sich bei seinen Aufnahmen am ehesten Stellen, an denen er deutlich versucht einen Player zu imitieren. Im Solo mat09 bei 1:30 bis Minute 2:00 greift er erkennbar die vom Patch gespielten rhythmischen Impulse auf und entwickelt auf diesen einen sich aufbauenden Spaltklang. Abschluss findet diese Form der rhythmischen Imitation bei 2:04, wo das Patch einen einzelnen Ton spielt und Schubert diesen kurz später mit einem einzelnen Ton beantwortet. Der derzeitige Aufbau des Patch kann aber leicht dazu führen, dass der Spieler in einem bestimmten Gestus zu improvisieren beginnt, und dieser durch die ganze Improvisation zu verfolgen ist, da keine neuen oder stark veränderten Sequenzen eingespielt werden. Jedoch sind generierte Sequenzen nicht mit Ideen oder Assoziationen eines menschlichen Mitspielers zu vergleichen. Dieser berechnet keine Tonfolgen, sondern schöpft aus einem jahrelangen Erfahrungsschatz als Musiker. Deshalb möchte ich keinen Sequenzgenerator einbauen. Eine für mich in Frage kommende Möglichkeit wäre, eine Phrasenerkennung zu entwickeln. Diese erkannten Phrasen könnten dann mit üblichen Variationsmethoden verändert werden. Da dies aber bisher nicht möglich ist, stellt sich die Frage nach der Eigenständigkeit des Patch.

### *5.4 Die musikalische Eigenständigkeit des Patch*

Von einer musikalischen Eigenständigkeit des Patch kann zur Zeit nicht gesprochen werden, denn es kann in der Improvisation zu Momenten des absoluten Stillstandes kommen. Dies tritt vor allem dann auf, wenn der Spieler kein Input-Material und damit auch keine neuen Samples und Steuerbefehle liefert. Ein Beispiel dafür ist berit03 (6:35). Die Bassistin hat aufgehört zu spielen und das Patch befindet sich in einer Art Warteschleife. Bis 7:10 findet keine Veränderung des Output statt. Die Granular-Player spielen als einzige weiter, da sie nach einmaliger Aktivierung im

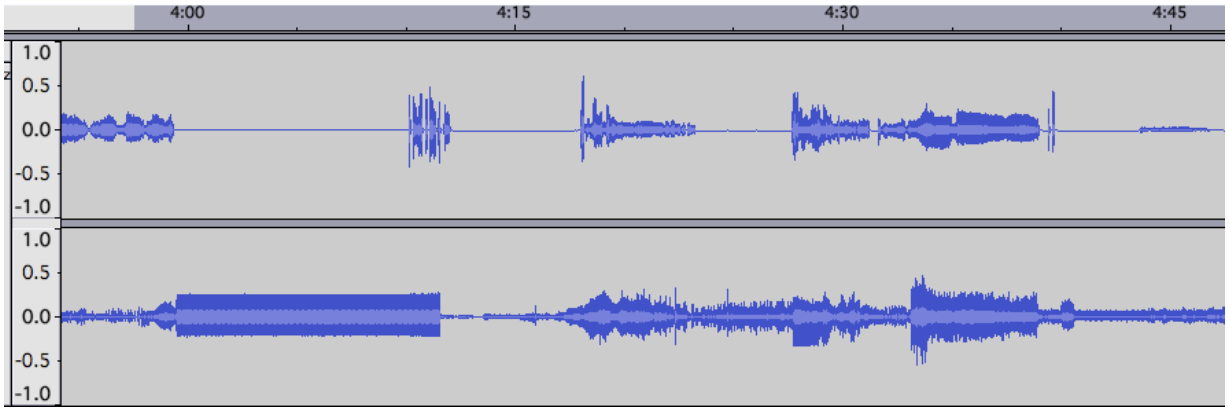
Subp.“mover01“ Bewegungen innerhalb des Buffer ausführen. Dadurch bewegt sich der Klangteppich etwas. Generell ist die Aufnahme berit03 ein Beispiel für den Fall des nicht Eintretens der gegenseitigen Beeinflussung und gemeinsamen Entwicklung des Solos. Von Seite der Bassistin wurden im Laufe aller drei Improvisationen kaum Versuche unternommen, Anregungen bzw. Aktionen vom Patch zu übernehmen oder zu beantworten. Gerade das Ende ab 7:10 bis 8:50 zeigt Spieler und Patch mit völlig konträrem Material, unbeeindruckt von einander spielend. Vielleicht ist das aber gerade dadurch ein Beispiel für die Eigenständigkeit des Patch.
Hingegen bietet Beispiel mat09 Minute 6:00 einen ganz anderen Eindruck. Obwohl kein neues Inputmaterial zur Verfügung gestellt wird, entwickeln hier zwei Player untereinander neue Strukturen und Klangfarben. Begonnen bei der Repetition eines Samples in zwei verschiedenen Tonhöhen gibt es einen Übergang zu Granularflächen (6:30). Diese bleiben auch in ständiger Veränderung bis zum Ende der Improvisation, welche durch „ahform“ bestimmt ist. Diese Entwicklung von neuem Material ohne Input ist für mich auf zwei mögliche Tatsachen zurück zu führen. Die Einfachste ist es, sie mit dem Versuchsaufbau zu begründen. Der Output aus dem Lautsprecher kann immer zu einem gewissen Teil wieder vom Mikrofon aufgenommen werden. Dadurch kommt es zu einer Art „Informationsfeedback“. Wenn der Spieler nicht spielt, hört das Patch sich selbst. Wenn der Spieler zu leise ist, oder das Patch zu laut, kann dieses Phänomen die Improvisation extrem beeinflussen! Ein imposantes Beispiel dafür findet sich in der Aufnahme berit01 bei Minute 5:40. Um solche „Feedbacks“ zu vermeiden wäre ein Versuchsaufbau mit Kopfhörern geeigneter gewesen. Darauf habe ich verzichtet um eine natürliche Spielsituation zu schaffen, mit einer im Raum ortbaren zweiten Klangquelle.
Ein Beispiel, welches die typische Entwicklung des Patch mit sehr wenig Inputmaterial zeigt, ist die Aufnahme mat06 ab Minute 7:00.

Der zweite Grund für eigenständige Output-Entwicklungen des Patch ohne Input kann sein, dass Veränderungen an den Playern oder an Lautstärken in dem jeweiligen Formteil mit Analysewerten verknüpft werden, die sich wenn der Instrumentalist nicht spielt, extrem sprunghaft verhalten. Grund dafür ist, dass das Rauschen innerhalb der Technik analysiert wird. Im Hörbeispiel mat06 Minute 4:13 bis 4:17 hört man ein willkürliches Pitchen von Samplegrains. Da im Rauschen alle Frequenzen enthalten sein können, verhält sich die Analyse der Tonhöhe in solchen Momenten sprunghaft. Um dies zu verhindern habe ich eine -30 dB Sperre in die Input/Output Matrix eingebaut. Jedoch direkt mit der Inputanalyse verbundene Steuerungsparmeter enthalten diese Sperre nicht, wie man am Beispiel des „pitchplayer" an dieser Stelle hört.
Diese Sperre kann auch hinderlich sein, denn sehr leise Anfänge wie in den Soli mat08 oder berit02 wurden dadurch ignoriert und folglich nicht beantwortet.

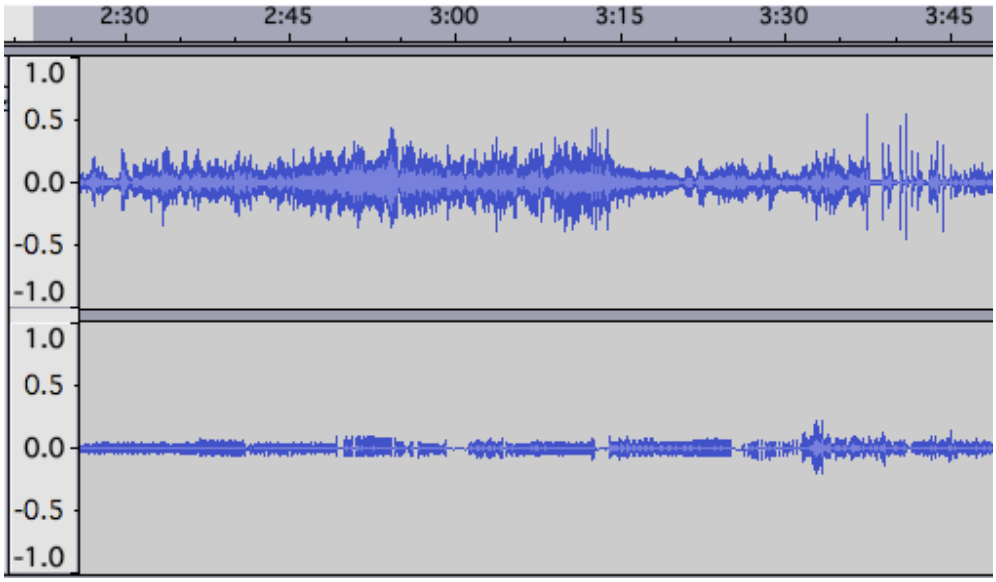
### *5.5 Erkennbare Reaktionsmuster*

Reaktionsmuster innerhalb des Patch sind zweifelsohne zu erkennen. Schon alleine dadurch, dass alle Steuerungen an den Input geknüpft sind. Obwohl sich die Verknüpfung der Parameter bei verschiedenen Formteilen stets unterscheidet, ist aber ein Audio-Input Voraussetzung für Veränderungen des Patch-Output. Zum Beispiel können Lautstärkeveränderungen der Player nur bei Audio-Input auftreten, egal an welchen Parameter (pitch, noise etc..) diese geknüpft sind. Bei Minute 4:00 im Solo mat06 spielt das Patch eine dichte, laute, granular-artige Fläche. Diese kann dynamisch bis 4:10 nicht verändert werden, da kein neuer Input kommt. Erst die neue Saxophonphrase löst eine Lautstärkenveränderung aus. Wie man an der folgenden Grafik sieht, tritt diese nicht sofort, sondern erst zwei Sekunden später ein. Dies hat nichts mit einer Latenz zu tun, sondern erst mit dem verspäteten Erfüllen einer Bedingung für die Veränderung der Lautstärke. Nachdem die Fläche endet, wird ein anderer Player wird hörbar. Die

Abbildung zeigt die Dynamikhüllkurven der Stelle dabei befindet sich oben der Audio-Input und unten der Patch-Output.




Da das Eintreten dieser Veränderungen noch zusätzlich an eine Mehrzahl von Bedingungen geknüpft sein kann, können Reaktionen für längere Zeit ausbleiben. Ein Beispiel dafür findet sich in der Improvisation berit02. Von Minute 2:30 bis 3:20 hört man zwar verschiedene Player unterschiedliche Aktionen ausführen, jedoch bleiben ihre Dynamikverhältnisse immer gleich. Besonders deutlich wird dies in der folgenden Abbildung:

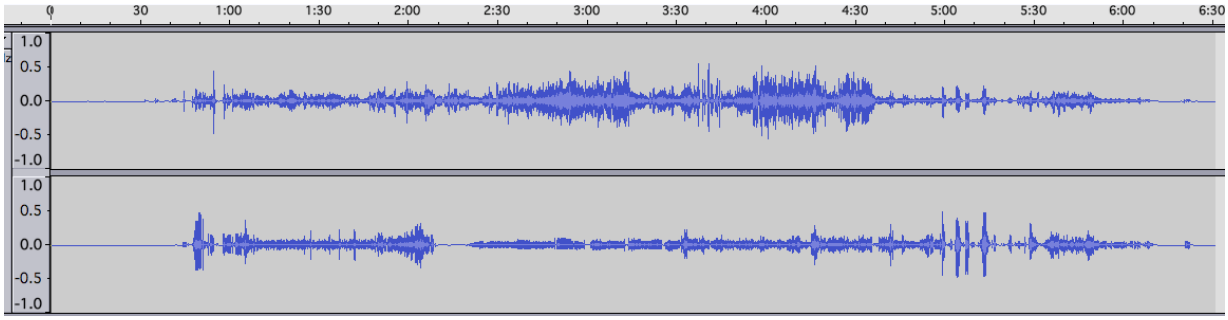



Die Grafik zeigt oben den Audio-Input und unten der Patch-Output. In dem eingehenden Audio-Input wird die Bedingung zur Neuberechnung der Lautstärken nicht erfüllt. Diese beiden Phänomene sind in allen Beispielen zu

beobachten. Mit einfachen Worten: Nur wenn Input kommt, kann etwas Neues im Output des Patch passieren. Diese Tatsache unterscheidet das Patch ganz deutlich von einem individuellen Spieler, der aus Intuition oder Inspiration spontane Ideen in die Improvisation einbringt. Dennoch kann man es an dieser Stelle auch als eine ganz eigene Charakteristik dieses „Spielers" betrachten.
Matthias Schubert hat im Gespräch einen Vergleich mit einem Spiegelkabinett aufgestellt, welcher in Bezug auf diese Beobachtungen gut nachzuvollziehen ist. Auch wenn das eigene Material, geschnitten, gestaucht oder gedehnt erklingt, so wird es nur verändert wenn ich meine Position verändere.

### *5.6 Entwicklung der Improvisation*

Eine Entwicklung innerhalb der Improvisation kann auf jeden Fall stattfinden. Am Beispiel berit02 ist dies gut zu erkennen. Eine grobe Formanalyse lässt mich diese Improvisation (mit der Gesamtlänge von 6:30 min.) in 4 große Abschnitte unterteilen. Diese unterscheiden sich durch das gespielte Material der Bassistin und dem daraus resultierenden Patch-Output.




Den ersten Abschnitt stellt der markante leise Anfangsteil dar, auf den keine Reaktion vom Patch folgt. Zu Beginn des zweiten Abschnitts (ab Minute 0:45) findet seitens der Bassistin ein dynamischer Wechsel statt. Mit dem Bogen erzeugt sie laute, perkussive  Geräusche. Diese werden in die Slots des „attacplayers" geladen und als Groove arrangiert ausgegeben. Ab 1:30 kommt der „pitchplayer" zum Einsatz, der den Obertönen des gestrichen Instrumentes folgt. Beendet wird dieser Teil wieder durch die Rhythmen. Ein

dritter Abschnitt beginnt bei Minute 2:10. Die tiefen, lang gestrichen Töne der Bassistin werden vom „cdhangplayer" und „hackplayer" beantwortet. Die Dynamik des Patch ist dabei sehr statisch. Innerhalb dieses Teils finden kaum Veränderungen statt, der Bassklang variiert dynamisch zwischen forte bis fortissimo, das Patch in mezzopiano bis mezzoforte. Der Output wird durch den hinzukommenden „loopplayer" dichter. Ab der fünften Minute beginnt die Bassistin mit dem Bogen Flageoletttöne zu spielen. Das Patch greift diese wieder mit dem „attacplayer" auf. Es entstehen sich überlagernde Strukturen mit dem Input. Die gespielten Flächen bekommen dadurch eine vom Patch erzeugte rhythmische Struktur.

Das Beispiel mat03 hingegen hinterlässt einen ganz anderen Eindruck. Ein zufälliges Hineinhören in zwei oder drei verschiedene Stellen der 5:00 Min. langen Improvisation lässt einen keine unmittelbare Entwicklung oder einen signifikanten Unterschied ausmachen. Das von dem Tenorsaxophonisten gespielte Material ist, wie im Kapitel „Kommunikation" beschrieben, sehr streng thematisch. In dieser Improvisation setzt es sich nur aus Spaltklängen und Flageoletttönen zusammen. Spaltklänge haben keine eindeutig erkennbare Tonhöhe für den analyzer~ und können von meinem Patch nicht gesondert verarbeitet werden. So hört man dann auch fast nur den „kurzzeitplayer", der nicht auf analysiertes Material zurückgreift, sondern einfach in der Zeitachse der letzten 3 Sekunden springt. Ab und zu ist der „cdhangplayer" zu hören, der sein Material aus der Dynamikanalyse entnimmt. Der „loopfadeplayer" hält sich an den Flageoletttönen auf, die offensichtlich als Übergang zwischen Ton und Geräusch deklariert wurden. Außerdem entstand bei dieser Aufnahme ein Grundmetrum von ungefähr 210 Bpm, welches immer wieder auftaucht. (z.B.0:15, 1:00, 2:55, ab 4:15 bis zum Ende).
Diese beiden Beispiele sind vom Input sowie vom Output derart unterschiedlich, dass man zum Einen von einer außerordentlichen Flexibilität

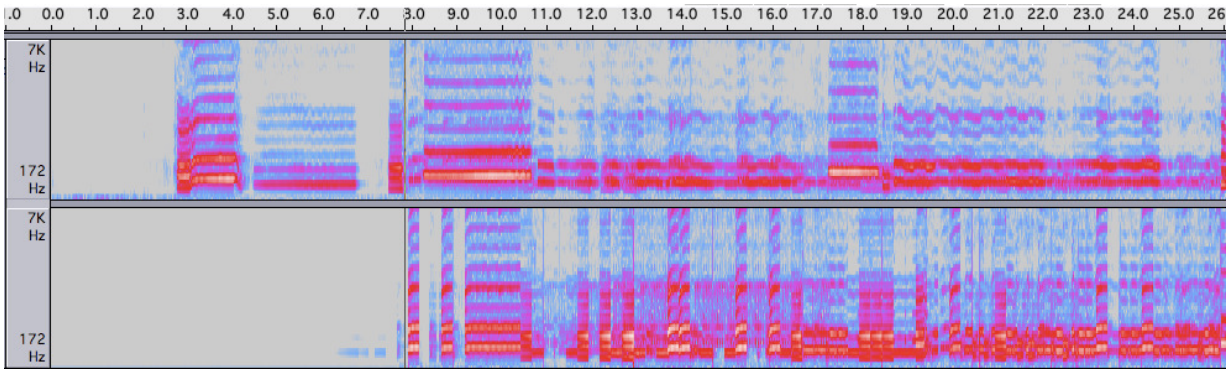
des Patch in unterschiedlichsten Spielsituation sprechen kann, aber auch den Mut haben muss, zu fragen, ob nicht doch alles der Spieler entscheidet. Haben die Formteile vielleicht gar keine veränderliche Wirkung, wenn der Instrumentalist keine Entwicklung zulässt? Diesen Eindruck kann man durchaus gewinnen und das Patch somit lediglich als eine Erweiterung für eine freie Soloimprovisation betrachten. Für die Kategorisierung als Duett müsste vom Player durchaus mehr Dominanz und Eigenständigkeit gezeigt werden. Formteile verändern zwar die Gewichtung der Player, verknüpfen neue Steuerparameter und entfernen die Analysedaten des letzten Teils, aber wenn das neue Material ähnlich oder gleich beschaffen ist, sind die Auswirkungen oft nur gering. So sind einige Player manchmal weiterhin nicht zu steuern oder womöglich sogar die Lautstärkenkontrolle blockiert.
In dem Interview beschrieb Schubert das ausgegebene Audiomaterial als „eine Konfrontation mit sich selbst" (seinem eigenen Spielen). Obwohl es ihn auch überrascht hat, in welcher Weise das Patch auf sein Spielen eingegangen ist. Gerade das nicht menschliche, die zu komplexen Reaktionen und drastischen Verarbeitungen des Materials, beschrieb er als inspirierend.
Auf die Frage, welchen Eindruck das generierte Audiomaterial gemacht hat, stellte die Bassistin Jung einen Vergleich mit der Kombination aus digitalem Effekt, der auf ihr Audio-Signal gelegt wurde, und einem Computerprogramm was Vorgaben abarbeitet an. Das Gefühl, eine musikalische Richtung angeben zu können, war für sie während der Improvisation nicht vorhanden. Des weiteren äußerte sie, das Audiomaterial habe sie teilweise beim Spielen gestört. Allerdings muss an dieser Stelle erwähnt werden, dass sie mit sehr konkreten Erwartungen an das Patch herangegangen ist. Versuche mehrdimensionale Schichten aufzubauen seien in Ihren Improvisationen gescheitert, da sich das Patch ihrem Material zu schnell annahm, so ihre Aussage.

Die Aufnahme almut03 zeigt bei Minute 1:30 jedoch, wie ein Formteilwechsel des Patch eine Richtungsänderung der Improvisation vorgeben kann. So kann man deutlich hören, wie zuvor der „attacplayer" dominiert, und auch die Sängerin sich rhythmischen Motiven zuwendet. An der Stelle 1:40 wird der „attacplayer aber vom „loopfadeplayer" abgelöst und es kommt unmittelbar danach der „pitchplayer" hinzu. Dies führt zu solch einer Veränderung des Output, dass ich hier einen neuen Formteil wahrnehme, obwohl die Sängerin diesen nicht explizit vorzugeben scheint.

### *5.7 Verwandtschaft des Patch-Output mit dem Inputmaterial*

Mein Ansatz zur Erstellung des Programms lautete: „Das Audiomaterial sollte während der Improvisation festgehalten und markante Elemente sollten in veränderter Form zurückgegeben werden."
Doch was ist als markantes Element zu verstehen? Und wie werden markante Elemente von beliebigen unterschieden. Die Umsetzung dieser Überlegung findet in der Form statt, dass ständig Durchschnitte von allen Analysewerten gebildet werden. Was vom Durchschnitt abweicht, wird als markant deklariert. Eine Höranalyse wird zeigen, ob dies in den Beispielen wieder zu finden ist.
Das Originalmaterial ist in dem Outputmaterial leicht wieder zu erkennen. Zur Anschauung hilft eine Spektrumanalyse. Da das Material von meinen Playern nur geschnitten und gepitcht wird, sind die Sampleabschnitte durch ihr Obertonspektrum gut wieder zu erkennen. Das Patch benutzt keinerlei Filter. Verfälschen kann die Ansicht lediglich die Kombination der sich überlagernden Player und extreme Tonhöhenveränderungen. Schnitte sind deutlich zu erkennen, wie gleich zu Beginn der Aufnahme almut03. Hier wurden kleine Sampleabschnitte mit immer gleichem Abstand gespielt. Ein deutlicher Bezug vom Input Sekunde 3.5 zu dem Output Sekunde 8 ist zu sehen. Jedoch in einer rhythmisch völlig anderen Gestalt. Das Material ist

herausgeschnitten und wird wiederholend eingesetzt. Die Grafik zeigt oben das Input-Signal und unten den Patch-Output.




Zu Beginn des Solos mat03 spielt Matthias Schubert einen Ton, der durch viel Luft und Zungeneinsatz sehr geräuschhaft wird. Diesen entwickelt er zu einem Spaltklang. Ob beabsichtigt oder durch Zufall gewinnt der Obere der beiden Klänge (0:11) soviel Gewicht, das kurzzeitig ein Flageolettton ausbricht. Diesen Ton würde ich als sehr markant bezeichnen. 13 Sekunden später wird diese Stelle vom Patch aufgegriffen und verarbeitet. Somit wird ein markantes Element zurück in die Improvisation gebracht. Genauso verhält es sich mit der Stelle 3:10 (ein Flageolettglissando, und zwei Töne im Tritonusabstand) die bis 3:50 zentrales Element im Output des Patch bleiben. Dadurch das der Saxophonist diese Elemente auch immer wieder aufgreift, bekommt diese Kombination die Wirkung eines Motivs, welches sowohl vom Spieler, als auch vom Patch, bis zum Ende der Improvisation verarbeitet wird.

Auch wenn mein Ansatz, Abweichungen von der Norm als markant zu deklarieren, als durchaus fragwürdig zu bewerten ist, zeigen diese Beispiele dass es funktionieren kann.

Bei der akustischen Auswertung der Aufnahmen, spielt die Umsetzung der Daten in Klangereignisse durch die Player eine wichtige Rolle. Teilweise kann ich im Nachhinein nicht mehr unterscheiden, welcher Player an welcher Stelle

gespielt hat. Dennoch gibt es Stellen, in denen Player nur all zu deutlich erkannt werden können. Dies kann als störend empfunden werden.
Da rhythmische Veränderung des Materials sehr auffällig wirken, erkennt man an dieser Charakteristik einige Player selbst bei geringer Lautstärke.
Der rhythmisch auffälligste Player ist der „cdhackplayer". Hier einige ausgewählte Stellen:
- almut03 bei Minute 2:15, 3:22 bis 3:35
- almut02 bei Minute 0:40
- mat08 bei Minute 2:22, 3:03, 3:30

Merkmal dieses Players ist, dass er das laute Material im Buffer willkürlich loopt und bei Eintreten eines bestimmten Events die Looplänge halbiert. Das passiert so lange, bis sie unter 20 ms liegt. Dann wird ein neuer Ausschnitt ausgewählt. Die musikalischen Auswirkungen, die diese daraus entstandene ständige Beschleunigung mit sich führt, war mir während der Entwicklungszeit nicht in den Sinn gekommen. Diesen Player werde ich entweder völlig neu strukturieren oder durch einen Anderen ersetzen.

Die weiteren Player besitzen zwar auch einen markanten Sound, dennoch sind sie flexibler in ihrer Anpassung an die Improvisationssituation.
Trotz ihrer Beschränkungen im Umgang mit dem Samplematerial, führt ihre Bearbeitung nicht so schnell zu einer Hörsättigung wie es beim „cdhangplayer" der Fall ist.
Der „loopfadeplayer" kann eine enorme Dichte erzeugen, die aber durch die Lautstärkeveränderungen innerhalb des Players selbst und durch die Bewegung der Grains im Buffer stets lebendig wirkt. Der „attacplayer" ist so konzipiert, das er immer wieder neue Samples ausssucht und eher kurze perkussive Parts spielt. Der „kurzzeitplayer" stellt ein variables Delay mit Pitchfunktion da, aber schon die drei veränderlichen Faktoren Delayzeit, Pitch-Up und Pitch-Down bieten in Kombination einige Facetten.

Der „pitchplayer", der ein Samplegrain aus seinem ursprünglichen Zusammenhang reißt und parallel zum Audio-Input pitcht, entwickelt durch verschiedenes Klangmaterial stets neue Klangfarben.
Bei diesen Playern gibt es eher noch größere Probleme bei der Klangqualität. Diese sind darauf zurück zu führen, dass Steuerbefehle, die über den Audio-Input kommen, nicht zu Ende ausgeführt werden, sobald Neue eintreffen. Dieser abrupte Abbruch einer Playfunktion wird bisher nicht über eine Dynamikhüllkurve abgefedert und dadurch entstehen akustisch wahrnehmbare Clicks. Hier muss bei einer Weiterentwicklung des Patch ein übergeordnetes System entwickelt werden, dass entweder die Steuerbefehle im Hinblick darauf kontrolliert, oder die Clicks rechtzeitig ausblendet.

### *5.8 Das Verhältnis des menschlichen Spielers zur Maschine*

Die Frage, ob das generierte Audiomaterial wirkt als ob es von einem menschlichen „Elektronik-Spieler" gespielt werden würde, wurde von allen drei Befragten verneint. Bei der Umsetzung des Patch habe ich auch nicht vorgesehen, menschliche Züge der Musikkomposition oder Improvisation nachzubilden. So werden vom Patch selbst keine neuen Ideen entwickelt oder Hörerfahrungen gesammelt. Da der Computer kein Lebewesen ist, hat die Umgebung und soziale Struktur, in der er sich befindet, auch keinen Einfluss auf seine Rechenoperationen und folglich nicht auf das Improvisationsverhalten.
Eine zusätzliche Unflexibilität stellt die bei Programmstart berechnete Form dar, denn diese wird unabhängig vom Input durchlaufen. Dadurch entsteht oftmals ein abrupt wirkendes Ende, da der Spieler nicht über den Ablauf der Form informiert wird.
Was macht trotzdem den Reiz aus mit diesem Programm zu spielen?
Alle drei Befragten haben angegeben, durch das Audiomaterial des Patch auf neue Ideen gekommen zu sein. Rhythmus, Klangästhetik, Dynamik, Tonauswahl und Struktur waren dabei genannte Aspekte. Sogar die Frage

danach ob sie damit auftreten würden, wurde nicht explizit verneint.(„vielleicht" – war 3x die Antwort) Nach einer genaueren Auseinandersetzung mit dem Patch und einigen Proben würden sie es in Erwägung ziehen, damit vor Publikum zu spielen.

### *5.9 Ausblick*

Die im Gespräch mit den Spielern geäußerten Vorstellungen und Wünsche, für eine Weiterentwicklung des Patch, waren sehr verschieden. Angefangen bei konkreten Verbesserungen an der metrischen Ebene, über die Möglichkeit des Patch auch neues Material zu entwerfen, bis hin zu dominanteren Verhaltensmustern beim Spielen waren als Ideen geäußert worden. Aber auch Erweiterungsvorschläge wie die Integration eines Mehrspielermodus wurden genannt. Mich selbst interessiert jetzt, rückblickend auf die Spielversuche, eine Gleichberechtigung auf Ebene der Kommunikation zwischen Spieler und Patch von Anfang an herzustellen. Dazu soll das Patch aber keine Kompositionsalgorithmen ausführen um Material zu entwerfen, sondern aus Daten vergangener Spiel-Sessions schöpfen können. So würde jede Kopie des Patch durch seine Mitspieler und die vergangenen Spiel-Sessions nach einiger Zeit eine individuelle Programmversion mit ganz persönlichem Output darstellen.
Des weiteren möchte ich die Abhängigkeit des Patch etwas vom Audio-Input lösen, sodass auch ohne ständigen Audio-Input neuer Output entwickelt werden kann. Dies könnte durch Hinzufügen einer übergeordneten Programmstruktur realisiert werden, die dann aktiv wird wenn kein Audio-Input ankommt.
Da der Spieler und Zuhörer die Player des Patch akustisch wahrnimmt, ist eine Weiterentwicklung dieser sowohl im Bereich der Klangqualität, als auch in ihrer Spielweise notwendig. Um diese zu verbessern wird es notwendig

sein die Daten der Inputanalyse besser zu kategorisieren, dabei wäre die Entwicklung einer neuen Strategie (z.B. Phrasenerkennung) denkbar.
Die Aufhebung einiger von mir gesetzten Beschränkungen könnte die Flexibilität des Patch erheblich erweitern. Als Schwerpunkt denke ich da an die Weiterentwicklung verschiedenster Auswahloperationen (nicht nur if..then, und random), welche ohnehin notwendig werden um einen sinnvollen Einsatz von erkannten längeren Phrasen zu gewährleisten.
Dabei gilt es auch die Player entsprechend umzugestalten. Ob nun die Begrenzung auf Samplematerial erhalten beleibt und dafür zu Beginn der Improvisation auf eine Bibliothek vergangener Spiel-Sessions zugegriffen wird oder auch Möglichkeiten der Klangsynthese genutzt werden wird sich zeigen.
Ein Punkt, der eine freie Improvisation stark von den Vorgaben einer Komposition trennt, ist auch die Freiheit der formalen Gestaltung. Bisher ist mein Patch in diesem Punkt absolut nicht anpassungsfähig, da es an willkürlichen Stellen neue Formteile beginnt. Formteile könnten später rückwirkend definiert werden. Wenn beispielsweise der Spieler sein Material ändert, sollte das Patch in der Lage sein an dieser Stelle einen möglichen Formteilwechsel zu registrieren. Optimal wäre sogar die Wahl der Player anzupassen anstatt auszulosen.
Ein Gedanke der mich von Anfang an begleitet hat und bisher nicht zu Umsetzung kam, wäre die Möglichkeit der Selbstreflexion des Patch. Es sollte dann z.B. erkennen können das ein bestimmter Player zu dominant spielt und ihn selbst abschalten.

# Literaturverzeichnis


Neukom, Martin: *Signale, Systeme und Klangsynthese*, hrsg. von Dominik Sackmann, Bern 2003

Noglik, Bert: *Klangspuren – Wege improvisierter Musik*, Berlin 1990

Rowe, Robert: *Machine Musicianship*, Massachusetts 2001

Rowe, Robert: *Interactive Music Systems*, Massachusetts 1993

Supper, Martin: *Elektroakustische Musik & Computermusik*, Darmstadt 1997

Wandler, Heiko: *Elektronische Klangerzeugung und Musikreproduktion*, hrsg. von Siegfried Schmalzriedt, Frankfurt a. M. 2005

Wilson, Peter Niklas: *Anthony Braxton – Sein Leben, Seine Musik, Seine Schallplatten*, Waakirchen 1993

# Audio-CD Titelinformationen

Alle auf dieser CD enthaltenen Titel sind in Stereo. Auf der linken Spur befindet sich das Originalsignal des Spielers und auf der rechten Spur der Audio-Output des Patch (0.58c).

| Nr. | verwendeter Titel | |
|---|---|---|
| 1. | almut01 | 3:22 |
| 2. | almut02 (beendet durch Programmabsturz) | 1:59 |
| 3. | almut03 | 8:03 |
| 4. | mat01 | 3:49 |
| 5. | mat03 | 4:55 |
| 6. | mat05 (beendet durch Programmabsturz) | 0:39 |
| 7. | mat06 | 9:14 |
| 8. | mat08 | 5:20 |
| 9. | mat09 | 7:43 |
| 10. | berit01 | 8:57 |
| 11. | berit02 | 6:31 |
| 12. | berit03 | 8:52 |

Titel 1-3 Almut Kühne – Stimme
Titel 4-9 Matthias Schubert – Tenorsaxophon
Titel 10-12 Berit Jung – Kontrabass

Aufnahmedatum: 11.10.2006 (Titel 1-9), 12.10.2006 (Titel 10-12)
Aufnahmeort: Berlin, in den Übungsräumlichkeiten der jeweiligen Spieler

# Inhalt der CD-ROM

| **Ordner** | **Datei** | **Funktion** |
|---|---|---|
| Diplomarbeit | Diplarb_hofmann.pdf | Arbeit als PDF Datei |
| | Diplarb_hofmann.doc | Arbeit als Word-Datei |
| Patch | Player58c_lay.mxf | das komplette Patch als Max collective, inclusive aller Objekte außer FTM. |
| | Player58c_lay | Max-file – einzeln |
| Patch->meinemodule | *.mxb | die verwendeten Objekte - einzeln |
| Externals & Objects | Analyzer~ for Intel Mac (beta).sit | verwendetes External zur Analyse des Audio-Input |
| | Ftm.1.7.7b-Max46macintel.dmg | Library zur Tabellenverwaltung in Max/MSP |
| | Ftm.longer2005.pdf | Beschreibung der FTM Library |
| .. | Readme.rtf | Hinweis zum Ausführen des Patch |

**Selbstständigkeitserklärung**

Ich versichere, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe.

Alex Hofmann

Hannover, den 31.10.2006