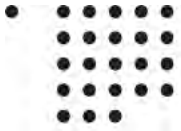


Fachhochschule Köln
Cologne University of Applied Sciences

Bachelorarbeit:

Bewegung als Musikinstrument

Entwicklung eines Prototypen auf Basis von Kinect und Pure Data

Ausgearbeitet von Adrian Rennertz

Halmstraße 17

50825 Köln

Matrikelnummer: 11068174

vorgelegt zum 20.03.2012 an der

FACHHOCHSCHULE KÖLN CAMPUS GUMMERSBACH

FAKULTÄT FÜR INFORMATIK UND

INGENIEURWISSENSCHAFTEN

STUDIENGANG - MEDIENINFORMATIK

Erstprüfer: Prof. Dr. Wolfgang Konen

Zweitprüfer: Prof. Christian Noss

Selbständigkeitserklärung

Hiermit erkläre ich, dass ich die von mir am heutigen Tage dem Prüfungsausschuss der Fakultät Medieninformatik eingereichte Bachelorarbeit mit dem Thema “Bewegung als Musikinstrument: Entwicklung eines Prototypen auf Basis von Kinect und Pure Data” vollkommen selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Köln den 20.03.2012

Adrian Rennertz

Danksagung

An dieser Stelle danke ich Herrn Wolfgang Konen für die gute Betreuung, Jens Heinen, Marcel Panne und den restlichen Künstlern von Lichtfaktor, die mir ihr Wissen und ihre Technik zur Verfügung stellten, allen Testpersonen, die mir während der Entwicklung und zur Evaluation wertvolles Feedback gaben und allen anderen, die mir geholfen haben diese Arbeit erfolgreich abzuschließen.

Kurzfassung

Es wurden Möglichkeiten der Bewegungsanalyse durch Gamecontroller untersucht und dazu passende Audioerzeuger erstellt. Dazu wurden vier Prototypen erstellt, die auf verschiedenen Analysetechniken basieren. Dazu zählt die Analyse per Webcam, der Sony Playstation Move Controller, und die Microsoft Kinect. Dann habe ich einen Ansatz auf Basis der Kinect weiterentwickelt. Die Eingabedaten der Kinect wurden über Open Sound Control zu Pure-Data übertragen. In Pure-Data habe ich ein Instrument erstellt, das durch die geschickte Kombination von Oszillatoren und Filtern einen individuellen und interessanten Klang erzeugt. Dazu wurde ein entsprechendes Kontrollkonzept umgesetzt, welches sich an dem eines Saiteninstrumentes orientiert, dieses Konzept jedoch von der Bewegung der Finger und Hände auf größere Bewegungen des ganzen Körpers überträgt. Es wurden Schnittstellen erstellt und diskutiert, mit denen das System via MIDI und Open Sound Control auch an andere Audioerzeuger angeschlossen werden kann. Während der ganzen Entwicklung wurde darauf geachtet, Open-Source-Software zu bevorzugen. Am Ende steht ein elaboriertes System, welches abgesehen von der Hardware völlig frei verfügbar und quelloffen ist.

Inhaltsverzeichnis

1 Einleitung

1.1 Worum geht es

Tanz als Bewegung zu Musik wird allgemein verstanden und als passend gesehen. Ein Tänzer kann mit seinem Arm eine Melodie nachfahren oder auch einen Rhythmus schlagen, so dass jeder, der die Bewegung sieht und gleichzeitig die Musik dazu hört beides in Verbindung miteinander bringen kann. Doch auch der umgekehrte Weg, nämlich aus Bewegung Musik zu machen ist nicht ganz neu.

Ein Dirigent hebt die Arme, um mit seinem Orchester zu kommunizieren. Jedes Instrument setzt im Grunde eine Bewegung dessen, der es spielt in einen Sound um, selbst wenn es nur die Bewegung eines Fingers ist um die Pianotaste herunterzudrücken. Um diese kleinen und doch komplizierten Bewegungen interpretieren zu können muss der Betrachter selbst wissen, wie ein Instrument gespielt wird. Wenn ein Pianist einen komplizierten Akkord spielt und dazu eine Melodie, wird nur ein anderer Pianist verstehen, welche Taste welchen Ton erzeugt hat, wenn er aber dazu im Takt hin und her wiegt, kann jeder Zuschauer die Bewegung interpretieren.

Diesen Zusammenhang auf eine andere Art zu interpretieren war bereits Gegenstand vieler interessanter Arbeiten (z.B. MIDAS [1], das MEGA Project [2]). Es wurden Sensoren verwendet wie Lasergitter (z.B. bei einer Laserharp), Kameras und viele weitere. Mit aktuellen Gamecontrollern wie der Nintendo Wii oder der Microsoft Kinect sind jedoch neue oftmals einfachere Möglichkeiten der Analyse mit geringer Latenz gegeben. Auch hier gab es schon Ansätze, die der Aufgabenstellung durchaus gerecht wurden. Es gibt einige Videos von mehr oder weniger gut funktionierenden Schlagzeug-Simulationen mit der Wiimote, oder von verschiedenen Kinect Instrumenten (siehe 2.1).

Diese Arbeit befasst sich mit einer Untersuchung des “State of the Art” in diesem Bereich und dem Erstellen eines eigenen Systemes zur Bewegungssteuerung, das diese

neuen Techniken auf eine individuelle Art nutzt, um Musik erlebbar zu machen und mit natürlichen Bewegungen Teil der Musik zu werden. Dies wurde teilweise im Rahmen eines Praktikums bei dem Kölner Lightart-Büro Lichtfaktor realisiert.

1.2 Warum Gamecontroller?

Um Bewegungen zu analysieren, gibt es verschiedene Möglichkeiten und Sensoren. Motion Capturing beispielsweise ist eine Wissenschaft für sich. In dieser Arbeit geht es jedoch um allgemein verfügbare Systeme, die viele Menschen kennen, weil sie diese bei sich im Wohnzimmer stehen haben. Die Analyse soll möglichst simpel und billig gehalten werden, damit die Hürde zur Anwendung und Nachahmung gering ist. Diese Kriterien erfüllen Gamecontroller. Der Preis ist im Verhältnis zu der Technik, die man dafür bekommt, gering. Gamecontroller sind ausserdem in der Regel robust. Die Umnutzung des Bekannten übt einen Reiz aus, Betrachter werden also eventuell selbst kreativ und bekommen Ideen, wie man einen Controller anders nutzen könnte. Die untersuchten Gamecontroller sind außerdem einfach in ihrer Benutzung. Ein weiterer Vorteil von Gamecontrollern ist die geringe Reaktionszeit. Um Videospiele zu steuern, müssen Bewegungen direkt sein, da jede Verzögerung sofort auffällt. Gleiches gilt für die Solidität. Genauigkeit ist auch wichtig, muss aber bei Gamecontrollern nicht im Bereich wissenschaftlicher Messungen liegen. Hier haben Solidität und die Geschwindigkeit Vorrang. Damit eignen sie sich perfekt für die Steuerung von Musik.

1.3 Vorgehen

Im ersten Teil wird untersucht, welche Ansätze zu dem Thema es gibt. Daraufhin wird eine Auswahl von Gamecontrollern getroffen, die näher untersucht wird. Systeme zur Audiosynthese auf dem Computer werden erforscht, wobei externe Audiosynthesesysteme wie Hardware-Synthesizer oder DSP-Chips ausser Acht gelassen wurden, weil das der geforderten Einfachheit und dem niedrigen Preis entgegen gewirkt hätte und einfach nicht zur Verfügung stand. Dann wurden Prototypen angefertigt, die aus verschiedenen Analysemöglichkeiten, der Anbindung der Analyse an den Audioerzeuger und der Programmierung/Steuerung des Audioerzeugers bestehen. Anhand dieser Prototypen konnten Systemzusammensetzungen und einzelne Komponenten auf verschiedene Kriterien hin bewertet werden.

Auf Basis dieser Untersuchung wurde die Kinect als Analyseinstrument gewählt und Pure Data zur Interpretation der Daten sowie teilweise auch zur Audiosynthese. Der zweite Teil bfasst sich mit dem Erstellen eines verfeinerten Systemes mit diesen Komponenten. Er lässt sich in drei Bereiche unterteilen: Analyse, Komposition und Evaluation. Der Analyseteil erklärt die Auswertung der Controllerdaten und die erstellten Parameter. In dem Kompositionsteil werden verschiedene Techniken der Audiosynthese und ihre Umsetzung in Pure Data näher erläutert, und untersucht, wie gut sich welche Kontrollparameter auf welche Audioparameter abbilden lassen. In der Phase der Evaluation befrage ich einzelne Testpersonen zu dem System und zu den verschiedenen Interaktionsparadigmen. Auch werden technische Aspekte untersucht, um festzustellen, wie gut das System einsetzbar ist.

Teil I

Untersuchung von Komponenten

2 Vorüberlegungen und Recherche

2.1 Bestehendes

Mit dem Thema Motioncontroller für Musik haben sich schon viele Arbeiten auseinandergesetzt.

Die Urform der Bewegungssteuerung für Musik ist sicher das Theremin. Dabei handelt es sich um ein Instrument, das um 1920 von Leon Theremin erfunden wurde. Es ermöglicht es dem Spieler, durch Beeinflussung elektromagnetischer Felder mit den Händen einen Ton zu steuern. Dabei werden jeweils nur die Abstände der beiden Hände zu einer Spule gemessen. Die eine Hand steuert die Lautstärke eines Tones, während die andere Hand die Tonhöhe bestimmt. Diese Steuerungsparameter sind wenn man es genau bedenkt eine Reduktion auf die grundlegendsten Parameter eines Tones, die nötig sind, um damit Musik zu erzeugen. Mit den neuen Möglichkeiten, die mit der 3D Erfassung durch die Kinect gegeben sind, ist es nicht verwunderlich, dass jemand versuchen würde, das Theremin digital nach zu bauen. Genau das hat Martin Kaltenbrunner mit dem "Therenect" getan[3].

In jüngster Zeit zog das "Kinect Controlled Granular Scretching" von Chris Vik[4] eine Menge Aufmerksamkeit auf sich. Hier werden mit einer Kinect einige Audiosyntheseparameter gesteuert. Die Position einer Hand steuert die Frequenz eines LFO (Low Frequency Oscillator), der dann die Frequenz eines Bandpass-Filters verändert. Dieser Effekt ist im Musik-Genre "Dubstep" als "Wobble Bass" bekannt. Weitere Positionsdaten der Hände werden für Audioparameter wie Hall und Tonhöhe genutzt. Da Dubstep gerade ein sehr populäres Genre ist, trifft Chris Vik damit recht gut den Nerv der Zeit. Doch hier werden weniger Bewegungen, als mehr bestimmte Positionen in Musik Umgewandelt.

Tim Thompson schafft mit seinem Instrument "Multi Multi Touch Touch"[5] eine Verbindung von virtuellem und analogem Raum, indem er einen Holzrahmen gefertigt hat,

der in mehrere kleine Fenster unterteilt wird. Greift man nun in die einzelnen Fenster, werden verschiedene Effekte ausgelöst und durch die Position im Fenster moduliert.

Mit der Wiimote wurde das Hacken von Gamecontrollern populär und gerade für diesen Controller gibt es millterweile viele Ansätze, Musik damit zu machen im Internet. Zum Beispiel gibt es mehrere Videos und auch Tutorials, die sich mit dem Erstellen eines virtuellen Schlagzeuges befassen [6].

Studenten der Yonsei Universität in Süd Korea haben einen Kinect-Midi-Controller programmiert, der den einzelnen Gliedmaßen und ihren Positionen MIDI-Kontrollwerte zuordnet.[7] Dieser Ansatz liefert bereits viele Eingangsdaten, die musikalisch verwendet werden können.

Auch vor der Umnutzung von Gamecontrollern gab es schon versuche, Bewegung in Musik umzuwandeln, oder Musikinstrumente ohne Berührung zu spielen. Beispielsweise Erfand Bernhard Szajner 1981 die Laserharp, ein Musikinstrument, das durch das unterbrechen von Laserstrahlen gespielt wird und damit auch gleichzeitig für seinen visuellen Eindruck sorgt[8].

David Rokeby beschäftigte sich schon seit den frühen 80ern in seinen Arbeiten mit Interaktion mit Musik durch Bewegung[9]. Besonders interessant für dieses Projekt ist seine Arbeit "Very Nervous System"[10]. Dieses System teilt ein Videobild in Abschnitte auf, und misst dann die relative Helligkeitsänderung in den Abschnitten. Dieser Messwert wird dann als Kontrollparameter (z.b. Lautstärke) für Audioerzeuger genutzt.

Es gibt einige interessante Projekte, die sich mit der Interpretation von tänzerischem Ausdruck beschäftigen[2, 1, 11, 12]. Es wird unter anderem versucht, aus Bewegungscharakteristiken auf Emotionen zu schließen und diese dann in eine musikalische Ausgabe zu leiten.

2.2 Detaillierungsgrad

In welchem Detailierungsgrad sollen die Bewegungen analysiert werden? Sollen die Finger einzeln analysiert werden, die ganze Hand oder die ganze Person und ihre Position im Raum? Wie weit sollen Eingangsdaten interpretiert werden? Es gibt die Möglichkeit, Gesten der Hände, Arme oder des ganzen Körpers zu erkennen. Wenn ein Tanz analysiert wird, könnte sogar die Dramaturgie im Klang abgebildet werden. Die Mimik eines Tänzers könnte Nuancen der Musik bestimmen. Wie viel muss das System interpretieren?

Um die Körpersprache einer Person zu analysieren und ohne Verzögerung in Audio umzuwandeln, wäre ein Aufwand nötig, der den Rahmen des Projektes sprengen würde. Daher liegt der Schwerpunkt dieser Arbeit in der Analyse einer möglichst atomaren Ebene, also auf der Umsetzung von einfachen Bewegungen, Bewegungsrichtungen oder einfachen Bildbestandteilen wie Helligkeit oder Farbwerte.

Die Entscheidung für diesen Detaillierungsgrad begründet sich neben dem Aufwand auch in der Latenz. Je mehr Analyseaufwand nötig ist, desto langsamer wird das System. Und gerade bei der Erzeugung oder Steuerung von Musik ist eine geringe Latenz wichtig, da in der menschlichen Klangwahrnehmung kleinste Verzögerungen bemerkbar sind. Ausserdem ist eine Untersuchung von möglichst kleinen und genau bestimmten Anteilen eines Bildes oder einer Bewegung einfacher nachzuvollziehen.

In der Audioebene wird der Detaillierungsgrad genauso klein gehalten wie in der Videoebene. Es werden einzelne Elemente manipuliert wie Tonhöhe, Spatialisierung, Klangcharakteristiken einzelner Töne (z.B. durch das Einstellen eines Filterparameters) oder Lautstärke.

Es werden also direkte Abbildungen von einfachen Analysen auf ebenso einfache Audiobestandteile übersetzt. Selbst wenn komplizierte Verfahren zur Verfügung stehen, werden atomare Analysen bevorzugt, damit die Systeme möglichst direkt bleiben.

2.3 Einsatz und Umgebung (Performance)

2.3.1 Einsatzbereiche

Es gibt zwei grundlegende Situationen, in denen das System funktionieren soll: Als Installation bzw. Ausstellungsstück oder als Bühnenperformance. Für beide Situationen treten verschiedene Bedingungen auf und es werden unterschiedliche Anforderungen an das System gestellt.

Als Ausstellungsstück ist der Einsatz auf Kunst-Ausstellungen oder für Firmen auf Messen denkbar. Hierbei ist wichtig, dass das Objekt bemerkt wird und dass Interakteuere es verstehen. Es muss erkennbar sein, welche Möglichkeiten der Interaktion gegeben sind und was sie bewirken. Die Wirkung kann abstrakt sein, aber sie soll direkt nachvollziehbar sein. Ein Ausstellungsstück muss einfach sein. Ein Benutzer will kein Musikinstrument lernen, sondern sofort interessante Ergebnisse erleben.

Als Bühnenperformance ist wichtig, dass der Betrachter den Zusammenhang zwischen System und Musik erkennt. Ihm muss bewiesen werden, dass die Musik nicht einfach im Hintergrund erzeugt wird, sondern dass der Performer die Musik steuert. Das ist insofern ein wichtiger Unterschied zur Ausstellung, dass der Betrachter nicht einfach durch die Interaktion selbst überzeugt werden kann. Hier spielt die Latenz eine Rolle und auch die Nachvollziehbarkeit, ohne dass der Betrachter selbst interagieren kann. Als bewegungsgesteuertes Musikinstrument kann die Interaktion etwas komplizierter zu erlernen sein und damit auch vielschichtiger.

2.3.2 Technische Aspekte

Eine wichtige Vorüberlegung ist, bei welcher Lichtsituation das System funktionieren muss. Eine rein kamerabasierte Analyse wird im Dunkeln nicht funktionieren, während andere Sensoren davon vielleicht nicht betroffen sind.

Als Beispiel für eine Performance kann man die Lichtfaktor-Live Show[13] betrachten. Das ist eine Show die Lichtfaktor bei Festivals und anderen Events aufführt. Hier werden mit Hilfe von Langzeitbelichtung und verschiedenfarbigen Lichtquellen live Licht-Bilder gemalt. Durch eine spezielle von Lichtfaktor geschriebene Software kann der Belichtungsprozess live mitverfolgt werden. Dies wird mit Musik untermalt. In diesem Zusammenhang könnte das bewegungsgesteuerte Musikinstrument Anwendung finden. Hier ist mit Dunkelheit zu rechnen und mit verschiedenen Lichtquellen, die sich bewegen. Des weiteren gibt es Blitzlichter, die für einen Moment alles ausleuchten.

Bei einer Anwendung als Ausstellungsstück, ist mit hellem Kunstlicht zu rechnen. Die Lichtsituation lässt sich beeinflussen, zum Beispiel indem man die Installation in einer Box aufstellt. Dann muss sich ein Betrachter allerdings aktiv in das Ausstellungsstück hineinbegeben. Von Messeständen wird erwartet, dass die Exponate Eyecatcher sind. Das müsste in dem Fall also ausgeglichen werden.

Es wäre wünschenswert, ein System zu entwickeln, das vom Umgebungslicht unabhängig ist. Falls das mit angemessenem Aufwand nicht möglich ist, müssen verschiedene Modi für helle und dunkle Umgebungen implementiert werden.

Ein fertiges System muss solide sein. Sowohl als Ausstellungsstück, als auch bei einer Bühnenperformance muss es nach dem Aufbau funktionieren. Es darf nicht zu schwierig in Betrieb zu nehmen sein, und der Aufbau sollte schnell gehen.

Es werden sowohl Setups konzipiert, bei denen der Interakteur einen Gamecontroller in der Hand hält (PSMove Controller) als auch Setups bei denen der Interakteur freihändig bleiben kann.

2.4 Übersicht Eingabegeräte

Für diese Arbeit sind drei Gamecontroller interessant, und zwar die Kinect von Microsoft, die Playstation Eye Kamera (alleine) und der Playstation Move Controller (zusammen mit der PSEye Kamera)

Kinect

Die Kinect von Microsoft ist eine 3D Kamera. Sie erzeugt mit Hilfe einer Infrarotkamera und einer Infrarotbeleuchtung ein räumliches Bild mit einer Auflösung von 640 * 480 Bildpunkten, welche mit einer Framerate von 30 Bildern pro Sekunde ausgegeben werden. Jeder Bildpunkt hat einen Tiefenwert und daraus ergibt sich das Tiefenbild. Es können Tiefenwerte von ca. 60cm bis ca. 5m gemessen werden. Desweiteren gibt es die Möglichkeit des Skeleton-Tracking. Hier wird die Position des Interakteurs im Raum aus dem 3D-Bild extrahiert, und in ein Skelettmodell umgesetzt. Es lassen sich Koordinaten von Gelenken und Gliedmaßen ausgeben. Für das Abgreifen des Tiefenbildes gibt es Bausteine für viele Programmiersprachen und Entwicklungsumgebungen. Für die Skeleton-Detection gibt es zwei große und ausgereifte Frameworks: Das Microsoft Kinect SDK und OpenNI von Primesense. OpenNI ist lizenzfrei, also auch kommerziell nutzbar, das Microsoft SDK dagegen ausschließlich nicht kommerziell. Daneben gibt es auch hier einige weniger umfangreiche Frameworks und Analyseprogramme, von welchen noch OSCeleton interessant ist, da es plattformunabhängig ist, und Daten via OSC (Open Sound Control) verschickt und auf dem Macintosh Synapse, da es sehr einfach ist (und dem selben Prinzip folgt, wie OSCeleton)

Playstation Eye

Die Playstation Eye Kamera ist eine billige und sehr schnelle Webcam. Sie hat kein besonders hochwertiges Bild, kann aber dafür mit anderen analytischen Qualitäten aufwarten. Sie arbeitet mit 60 Frames / Sekunde bei einer Auflösung von 640*480 und mit 120 Frames / Sekunde bei einer Auflösung von 320*240. Ein normaler Wert bei einer Webcam liegt bei 25 Frames. Gerade der Modus mit 120 Frames / Sekunde ist für die

Anwendung als Audiocontroller interessant. Die Kamera funktioniert bei sehr geringem Umgebungslicht, was zur Solidität beiträgt.

Playstation Move

Der Playstation-Move Controller (im weiteren PSMove Controller genannt) funktioniert ähnlich wie die Wiimote Plus, was Lage- und Beschleunigungsmessung angeht. In beiden Controllern sind Beschleunigungssensoren und ein Gyroskopsensor verbaut. Der PSMove Controller kann zusätzlich mit mit einem digitalen Kompass aufwarten[14]. Die Erkennung der Position im Raum (Tracking) wird bei dem PSMove Controller mit einem leuchtenden Ball an dessen Spitze realisiert. Als Kamera bietet sich hier die Playstation Eye Kamera an. Es wird erkannt, welche Farbe im Raum gerade nicht oder wenig zu sehen ist und der Ball wird in dieser Farbe zum leuchten gebracht. Dann kann der Ball einfach im Raum erkannt werden. Durch die Größe des Balles auf dem Bild lässt sich die Distanz zur Kamera errechnen. Das unterscheidet den PSMove Controller von der Wiimote Plus, deren Tracking nur funktioniert, solange die Fernbedienung auf die Sensorenleiste gerichtet ist. Vereinfacht kann man den PSMove Controller als etwas genauere und robustere Wiimote Plus bezeichnen, weshalb die Wiimote Plus auch nicht einzeln untersucht wurde, sondern nur der PSMove Controller.

2.5 Übersicht über verwendete Programme und Protokolle

Um die Eingabedaten der Gamecontroller unabhängig von den zugehörigen Spielkonsolen nutzen zu können, sind Programme und Programmiersprachen sinnvoll, die für den Umgang mit Medien konzipiert sind, und die mit den entsprechenden Übertragungsprotokollen umgehen können. Processing und Cinder kommen aus der Ecke der Generativen Gestaltung (eine Bewegung die das Ziel hat, Programmieren für Designanwendungen möglichst einfach zu machen, so dass sich auch Jemand der statt des IT-Hintergrundes eher aus dem Design kommt, schnell damit zurechtfindet.) , während Pure Data und Max aus der Audioprogrammierung kommen.

Processing [15]

Processing ist eine Bibliothek für Java und zugleich eine eigene Programmiersprache für generative Gestaltung. Man kann es entwender in Java importieren, oder die Processing eigene API benutzen. Processing hat den Vorteil, dass es sehr einfach ist und für viele audiovisuelle Programmieraufgaben schon hoch ausgereifte Hilfsfunktionen bietet. Wer

Java kennt, wird sich in Processing sehr schnell zurechtfinden. Da Processing schon eine Weile existiert gibt es auf der Projektseite (http://www.processing.org) viele gute Tutorials, eine einfache und gut verständliche Referenz und Librarys für fast alles.

Cinder[16]

Cinder ist das Äquivalent zu Processing in C++. Für Cinder gibt es keine eigene Programmierungebung, es müssen also C++ Umgebungen wie XCode oder Eclipse genutzt werden. Es ist etwas leistungsfähiger als Processing, da viele C++ Bibliotheken näher an der Hardware und damit schneller sind. Cinder ist leider noch relativ jung und hat daher keine vollständige Referenz. Um sich in dieser Sprache zurechtzufinden, muss man sich Programmierbeispiele ansehen. Während Processing generative Gestaltung für Designer angesehen werden kann, ist Cinder generative Gestaltung für Programmierer.

MIDI

MIDI ist das aktuelle Standard-Steuerungsprotokoll in der Ton- und Lichttechnik. Stark vereinfacht hat eine MIDI-Nachricht einen Typen (Kontrollwert, Notenwert, etc.), eine Controller-Nummer zwischen 0 und 127 und einen Wert zwischen 0 und 127. Aufgrund dessen ist MIDI auf 128 Controller beschränkt und man muss acht geben, dass kein Controller doppelt belegt wird. Jeder Controller hat zusätzlich 16 Channels. MIDI ist also ein Protokoll, das sehr viel Struktur vorgibt, an die man sich dann auch halten muss. Das kann mitunter recht kompliziert sein. Da der Wert einer MIDI Nachricht nur 128 Abstufungen hat, kann man, wenn man damit z.b. die Tonhöhe steuern will, eventuell die Abstufungen hören.

Open Sound Control[17]

Open Sound Control (OSC) ist ein relativ neues, von der University of California in Berkeley entwickeltes Protokoll. Eine OSC-Nachricht besteht aus einem Pfad, für den Parameter der geändert werden soll (z.B. “ableton/wobblebass/lowpassfilter”), und dem entsprechenden Wert (z.B. “56.9”). Der Wert einer OSC Nachricht kann verschiedenen Datentypen entsprechen, z.B. Float, Integer, String oder Blob. Damit ist das Protokoll einfach und flexibel. Da jedoch seit Jahren alle externen Controller, Interfaces und Programme auf MIDI basieren, wird OSC vermutlich trotzdem noch eine Weile brauchen, um sich ganz durchzusetzen.

Pure Data / Max[18][19]

Bei Pure Data (PD) handelt es sich um eine grafische Programmiersprache die nach dem "Patching" Paradigma funktioniert. Das bedeutet, dass Objekte als Kästen dargestellt werden und mit Linien verbunden um zu kommunizieren. Man spricht dann von Patches anstatt von Programmen, und von Patching statt Programmieren, PD ist ein Ableger von Max und ähnelt dieser Sprache daher auch sehr. Die Idee hinter Pure Data ist, Audio, Video, Text und was es sonst noch an Datentypen gibt als reine Daten zu behandeln. Das bedeutet, dass die meisten Datenströme einfach in einen anderen Typ umgewandelt werden können. Pure Data ist im Gegensatz zu Max quelloffen und kostenlos. Pure Data kann OSC-Messages, MIDI-Messages, und andere Daten über das Netzwerk versenden und empfangen und eignet sich somit auch als Middleware, um zwischen Programmen zu vermitteln.

Max wird oft auch mit Max/MSP bezeichnet. Dabei ist MSP ein Framework, das Max beiliegt und zur Audioerzeugung dient. Max ist gegenüber Pure Data etwas polierter, was das Design angeht. Es wird von der Firma Cycling74 entwickelt und ist nicht quelloffen.

Synapse[20]

Synapse ist ein Programm, das die Kinect benutzt, um ein Skelett zu tracken, und diese Daten dann als OSC-Messages ins Netzwerk schickt. Synapse versendet die räumlichen Koordinaten der Gelenke, wenn ein Benutzer erkannt wurde. Damit es einen Nutzer erkennt, muss dieser sich in der Kalibrierungspose vor die Kinect stellen. Für schnelle Tests mit dem Tracking ist Synapse perfekt, da es sehr einfach und schnell funktioniert. Synapse ist quelloffen und basiert auf dem Framework OpenNI. Es läuft unter MacOs, Unix und Windows.

2.6 Übersicht Audioerzeuger

Um aus der Analyse Audio zu erzeugen wurden zwei Audioprogrammiersprachen untersucht und zwei kommerzielle Softwaresynthesizer.

Super Collider[21]

Super Collider (SC) ist eine Audio-Entwicklungsumgebung, bestehend aus dem SC-Server und der Programmiersprache SClang. Der SC-Server ist ein schlankes Kommandozeilen-Programm ohne GUI, das auf einer sehr niedrigen Ebene arbeitet und dadurch sehr performant ist. SClang ist die zugehörige Steuerungssprache für den SC-Server, wobei

dieser auch über das Netzwerk gesteuert werden kann. Die Kommunikation der beiden Komponenten funktioniert über OSC, was gut in das Projekt passt. Die grundlegenden Bausteine der Audioerzeugung in Super Collider bilden so genannte UGens (Unit Generators). Ein UGen ist ein atomarer Bestandteil eines Synthesizers, wie zum Beispiel ein Sinusoszillator oder ein Tiefpassfilter. Ein oder mehrere UGens können zu einer Synthdef (Synthesizer Definition) zusammengefasst werden. Diese Synthdef kann dann auf dem Server gespeichert und über OSC gesteuert werden. In SClang können natürlich auch ganze Lieder programmiert werden, aber für dieses Projekt ist die externe Steuerung interessant.

Pure Data / Max[18]

Pure Data und Max werden noch einmal als Audioerzeuger erwähnt, da sie Audio-Objekte beinhalten die Sounds erzeugen können. Sie bieten einfache Möglichkeiten mit Samples, Oszillatoren und Filtern umzugehen. Diese werden einfach in dem Patch erzeugt und fangen dann an, Audio zu erzeugen. Durch das Patchen von verschiedenen Objekten lassen sich mit Patcher-Sprachen Synthesizer und Sampler nach Belieben zusammensetzen. Die so geschriebenen Patches können bearbeitet werden, während das Programm läuft.

Professionelle Sampler / Synthesizer (Ableton, Kontakt, etc.)

Für die Audioerzeugung wurden auch kommerzielle Programme getestet, die die entsprechenden Schnittstellen aufweisen. Jedes professionelle Audioprogramm kann mit MIDI arbeiten und bei einigen kommt OSC langsam dazu. Diese Programme haben den Vorteil, dass sie schon vorgefertigte Effekte und Synthesizer haben, die entsprechend interessant klingen. Die grafisch dargestellten Regler, Tasten und Knöpfe lassen sich dann MIDIcontrollern oder OSC-Parametern zuweisen.

3 Prototypenentwicklung

Nachdem die Vorüberlegungen abgeschlossen sind, werden Prototypen entwickelt, bei denen sowohl konzeptuell als auch technisch verschiedene Ansätze geprüft werden. Es werden vier Systeme zu einem Grad umgesetzt, ab dem sie sich untersuchen lassen, und bewertet werden können. Entwickelt wurde unter Macintosh OS X 10.6.8

3.1 Bewertungskriterien

Die Bewertungskriterien nach denen ein Prototyp untersucht wird, teilen sich in technische und konzeptuelle Aspekte auf. “Konzeptuell” bezieht sich dabei auf die Idee, die dem Prototyp zugrunde liegt, während “technisch” sich mit den Details der Umsetzung befasst.

Konzept

Konzeptuell wird bei dem Entwurf eines Prototypen herausgefunden, wie gut die entsprechenden Aufbauten aus den Augen des Interakteurs oder Betrachters funktionieren. Diese sehen nicht die Technik im Hintergrund, sondern bewerten einen Aufbau alleine danach, wie er auf sie wirkt. Ein großer Faktor in der Wirkung auf den Betrachter ist die Verständlichkeit. Versteht ein Nutzer sofort, warum das System jetzt gerade so reagiert, wie es das tut und welche Aktion (des Interakteurs) mit welcher Reaktion (des Systems) zusammenhängen? Ein weiterer Faktor ist die Responsivität. Wie schnell reagiert das System? Bei Konzepten, die in Richtung Musikinstrument gehen, wird die Reproduzierbarkeit von Klängen untersucht. Eine schwierig zu beantwortende konzeptionelle Frage ist: Wie gut klingt der Prototyp, und wie gut passen Klang und Bewegung zusammen? Diese Fragen wurden zusammen mit den Lichtfaktor-Mitarbeitern besprochen, welche auf einige Erfahrung zum Thema Interaktive Systeme auf Messen und auch Liveshows zurückblicken können. Die konzeptuellen Bewertungen der Prototypen stellen hierzu meine eigene Meinung dar, mit den Einflüssen und Anregungen aus diesen Gesprächen.

Technik

Es wird untersucht, wie gut der Prototyp aus technischer Sicht funktioniert. Die einzelnen Komponenten werden auf ihre Benutzbarkeit untersucht und auf den Aufwand, der nötig ist, um ein System zu erstellen (z.B. wie schwierig ist es, mit einem Audioerzeuger einen interessanten Klang zu erzeugen). Desweiteren wird die Erweiterbarkeit betrachtet. Wie flexibel sind einzelne Teile des Systems zu ersetzen, oder weiter zu entwickeln? Die Abhängigkeit von Umgebungseinflüssen sollte möglichst gering sein. Der Aufbau sollte robust sein, damit ist sowohl die Software, als auch die Hardware gemeint. Auch der Preis der Komponenten soll eine Rolle spielen. Im Optimalfall benötigt man einen Gamecontroller und einen Computer und kann sich alles weitere kostenlos herunterladen. Aber es wird auch anhand von Probelizenzen untersucht welchen Unterschied kostenpflichtige Programme machen. Damit Aktionen und Reaktionen miteinander in Verbindung gebracht werden können, wie von der Konzeption gefordert, muss der Aufbau desweiteren eine geringe Latenz haben.

3.2 Prototypen

Zu jedem der entwickelten Prototypen gibt es zwei Filme auf der beiliegenden CD. Diese zeigen schnell um was es geht und sollten zusätzlich zu den Beschreibungen angesehen werden.

3.2.1 Helligkeitsanalyse

Es wird ein Kamerabild analysiert und aus der durchschnittlichen Helligkeit, oder aus der Helligkeit eines bestimmten Punktes ein Kontrollparameter bestimmt. Dieser Parameter steuert die Höhe eines Sinustones. Je heller der Punkt bzw. die Durchschnittshelligkeit, desto höher der Ton. Die Lichtsituation wird so angepasst, dass die Helligkeit zur Kamera hin zunimmt und der Hintergrund dunkel ist. Dadurch kann der Interakteur den Ton damit steuern, wie weit er ins Licht tritt oder wie weit er sich der Kamera nähert.

3.2.1.1 Konzeptbewertung

Der Zusammenhang zwischen der Bewegung und dem erzeugten Sound ist schnell zu erfassen. Wenn nur die Helligkeit eines bestimmten Punktes analysiert wird, begreift jeder Betrachter, dass sich die Tonhöhe ändert, wenn er die Hand in einen bestimmten

Bereich hält. Das Licht wird so eingestellt, dass die Helligkeit zur Kamera hin zunimmt. Dadurch entsteht der Effekt, dass der Ton höher wird, je näher die Hand oder irgendein anderer Gegenstand der Kamera kommt. Der Unterschied zwischen der Helligkeitsanalyse an einem Punkt und der Anaylse der Durchschnittshelligkeit ist nicht bemerkbar.

Dieses Setup reagiert mit geringer Latenz. Wenn man die Hand nach vorne oder hinten bewegt, ändert sich sofort die Tonhöhe. Dieser Zusammenhang lässt sich schnell begreifen. Es entsteht das Gefühl, den Ton wirklich mit der Hand hin und her zu ziehen. Auch auf ruckartige Bewegungen reagiert das Setup entsprechend.

Die Abhängigkeit von der Lichtsituation ist hier allerdings groß. Für Messen wäre es möglich, die Umgebung entsprechend anzupassen, aber der Aufwand würde sich für den Effekt nicht lohnen. Für Bühnenperformances ist der Aufbau auch zu umständlich und zu eingeschränkt. Durch die so erzeugte Lichtsituation ergibt sich andererseits ein interessantes Setup. Wenn der Interakteur das visuelle Feedback auf einem Bildschirm hat, kann er die Helligkeit oder vielleicht auch die Nähe zur Kamera mit der Tonhöhe in Verbindung bringen. Da hier nur ein einziger Parameter verändert wird, gibt es nicht viele Möglichkeiten der Interaktion. Die Spanne zwischen hoher Ton und niedriger Ton ist schnell ausgeschöpft und es entsteht der Wunsch nach mehr Möglichkeiten den Ton zu beeinflussen.

3.2.1.2 Technik-Erklärung und Bewertung

Es wurde ein Processing Programm geschrieben, welches ein Kamerabild analysiert und die Durchschnittshelligkeit, oder die Helligkeit an einem Punkt als normalisierten Wert ausgibt. Diese wird via OSC an Super Collider gesendet und Super Collider steuert mit diesem Kontrollwert eine zuvor erstellte Synthdef (siehe 2.6). In dieser Synthdef wird ein Sinus-Oszillator definiert. Diesem wird der Kontrollwert als Frequenz in einer festgelegten Spanne übergeben.

Dieses Setup brachte vor allem technische Erkennntnisse. Eine davon ist die, dass die Audiosynthese mit Super Collider in der Praxis zwar einsetzbar ist, jedoch sehr aufwändig, da auf niedrigem Level programmiert wird. Um zu einem Ergebnis zu kommen, das auditiv interessant ist, muss ein gewisses Know-How vorhanden sein, oder bereits erstellte Synthesizer-Definitionen, mit denen man arbeiten kann. Diese Ressourcen (vor allem die benötigte Zeit, um sich hier einzuarbeiten) stehen hier nicht zur Verfügung. Super

Collider ist codebasiert. Es gibt zwar einzelne GUI-Objekte, das Programm ist aber nicht darauf ausgelegt, dass man damit Benutzeroberflächen bereitstellt.

Die Verwendung von OSC macht das System flexibel. Denn die simple Videoanalyse kann auch durch komplexere Verfahren ersetzt werden, oder durch ein ganz anderes Programm, solange es dieselben OSC Nachrichten sendet. Gleiches gilt für den Audioerzeuger.

Der Aufbau ist einfach und das System robust. Solange der Interakteur die Hardware nicht zerstört, sind Fehler unwahrscheinlich. Sollte sich jedoch die Lichtsituation ändern, wird das System darauf reagieren. Es muss nichts ausser der Kamerahelligkeit kalibriert werden. Als zusätzliche Hardware wurde nur eine externe Kamera benutzt. Der Preis des Aufbaus ist also gleich dem Preis einer entsprechenden Webcam wie beispielsweise der Playstation Eye. (Da das Webcam Management unter Mac OSX die Schwäche hat, dass die automatische Helligkeitsregelung nicht auszuschalten ist, wurde für den Prototyp eine DV-Kamera benutzt. Unter anderen Betriebssystemen lässt sich aber eine PS-Eye gut benutzen und kann alle ihre Stärken ausspielen.)

3.2.2 Gitterbasiertes Sample Playback

Bei diesem Prototyp wird der Interaktionsraum in ein Gitter aufgeteilt, so dass sich viereckige Bildausschnitte bilden. Diesen wird dann eine Tiefe zugeordnet, so dass sich daraus Voxel bilden. Den Bereichen wird ein Ton zugeordnet. Per Tiefenbildanalyse der Kinect wird nun erkannt, ob sich etwas in einem der Voxel befindet. Wenn sich etwas in dem Bereich befindet, zum Beispiel eine Hand, dann wird der Ton abgespielt. Befindet sich nichts an der Stelle im Raum, so fadet der Sound langsam aus. So kann der Interakteur an verschiedenen Stellen im Raum verschiedene Töne triggern. Es ist hierbei egal, mit welchem Körperteil, oder sogar Objekt er dies tut. Das Gitter hat zwar eine bestimmte Tiefe, in der es ausgelöst wird, ist aber an sich zweidimensional. Es wäre möglich auch in Richtung der Tiefe mehrere Voxel anzuordnen. Dabei würden dann aber die vorderen Voxel die Hinteren verdecken, wenn sie getriggert werden.

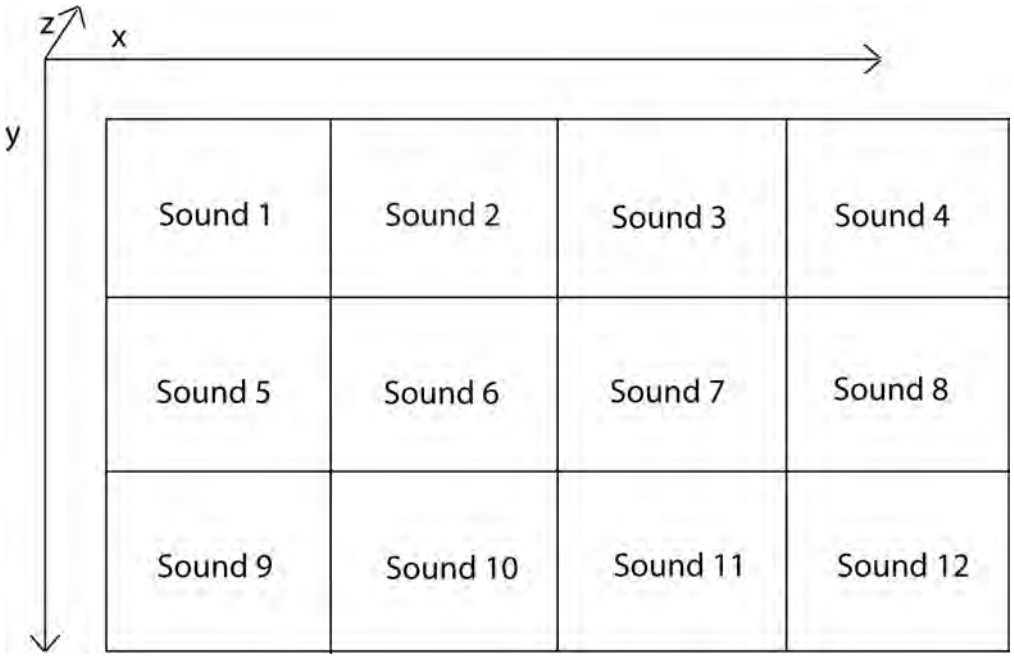


Abb. 3.1: Gitter

3.2.2.1 Konzeptbewertung

Dieser Aufbau bietet schon einige Vorteile gegenüber den vorherigen Prototypen. Er funktioniert als Blickfang, da er Töne erzeugt, wenn man daran vorbeiläuft. Die Töne, die auf den Gitterpunkten liegen sind auf 2 Kanäle spatialisiert (stereo) , das bedeutet ein Ton der zum Beispiel räumlich links liegt, klingt auch links. Dieser Zusammenhang wird schnell begriffen. Das Konzept wurde mit einem Gitter getestet welches 4 Voxel breit und 3 hoch war. In den unteren beiden Reihen befindet sich ein sehr flächiger Sound, während in der oberen Reihe ein Windspiel-Klang ist. Dass der obere Bereich höhere Klänge triggert, wird als zusammenhängend Wahrgenommen. Ein Interakteur versteht hier nicht, dass da ein Gitter im Raum hängt, aber er versteht, dass der Klang nach links geht, wenn er sich nach links bewegt, und dass hochtonige Sounds erscheinen, wenn er über seinem Kopf winkt. Der Ansatz, den Raum in Triggerpunkte aufzuteilen bietet viele interessante Möglichkeiten.

Was bei diesem Ansatz jedoch nicht untersucht wird, ist die Bewegung an sich. Ob sich ein Interakteur schnell oder langsam in einen Triggerpunkt hineinbewegt macht keinen

Unterschied. Durch die Spatialisierung kann eine Bewegung im Audioraum dadurch nachvollzogen werden, dass sich der Ton mitbewegt. Als Eyecatcher und als Ausstellungsstück ist dieser Aufbau funktional, da die Zeitspanne, in der ein Besucher mit dem System interagiert, nicht groß ist und die Interaktion sehr schnell begriffen wird. Die vertikale Achse kann noch mit Samples mit unterschiedlichen Tonhöhen bestückt werden, um den Eindruck zu verstärken. Auch unterschiedlich starke Bandpassfilter wären eine Möglichkeit. Doch um mit dem System wirklich Musik zu machen, ist der Aufbau zu einseitig. Sounds sind gut reproduzierbar, aber die Möglichkeiten der Interaktion sind auch schnell ausgeschöpft. Dafür fehlt noch so etwas wie eine Abbildung der Bewegungsgeschwindigkeit auf die Intensität des Sounds.

3.2.2.2 Technik-Erklärung und Bewertung

Das Konzept ist vollständig in Processing implementiert. Um das Tiefenbild der Kinect benutzen zu können, wurde die Processing Library Openkinect importiert. Das Tiefenbild wird dann von einem angelegten Kinect Objekt als Array ausgegeben, mit einem Tiefenwert für jeden Bildpunkt. Nun wird ein Tiefenbereich T definiert und die Bildbereiche, in denen sich die Sounds befinden sollen. Für den Prototypen reicht eine vereinfachte Analyse der Bildbereiche. Daher wurden hier nur die Gitterschnittpunkte untersucht, nicht die vollständigen Voxel. Der Tiefenwert dieser Bildpunkte wird nun durchgehend betrachtet. Wenn sich der Tiefenwert in dem Tiefenbereich T befindet, gilt der Trigger als an.

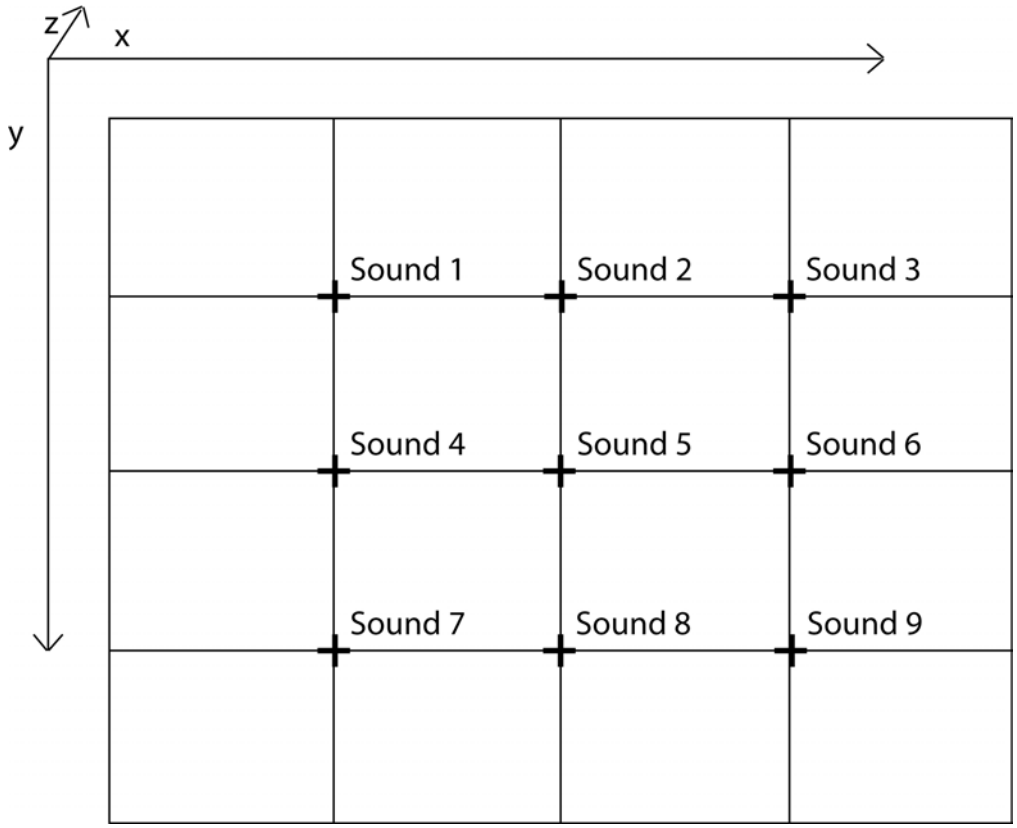


Abb. 3.2: Gitter Schnittpunkte

Jeder Triggerpunkt hat ein Sample, das als Loop ständig abgespielt wird. Die Lautstärke der Loops liegt initial bei 0 (Stille). Ist der Trigger an, hebt sich die Lautstärke auf einen Maximalwert. Ist der Trigger aus, senkt sich die Lautstärke kontinuierlich, bis sie auf 0 steht. Das Abspielen der Samples ist mit einer einfachen Standard-Library (Minim) direkt in Processing realisiert.

Bei der Untersuchung des Prototyps fällt auf, dass die Verwendung mehrerer Samples die Latenz erhöht. Die Tiefenbildanalyse weist alleine kaum spürbare Verzögerungen auf. Solange nicht mehr als 15 Samples dazukommen, ist auch der Sound kaum spürbar verzögert. Doch wenn mehr Samples benutzt werden, fängt das ganze System an zu ruckeln. Da die Werte für die Triggerpunkte nur aus dem Tiefenbild-Array abgerufen werden, was so gut wie keine Rechenleistung beansprucht, muss Minim für die Verzögerung verantwortlich sein. Ein Vergleich mit expliziten Audioumgebungen wie Pure Data zeigt,

dass diese mit einer weit höheren Anzahl Samples locker klarkommt. Zum Zeitpunkt der Entwicklung gibt es kein besseres Audioframework für Processing als Minim, was bedeutet, dass der Sound ausgelagert werden muss, wenn man diesen Aufbau weiter verfolgen möchte.

Da zur Analyse die Infrarotkamera der Kinect verwendet wird, benötigt das System kein Licht. Es kann also in völliger Dunkelheit oder auch in hellen Umgebungen benutzt werden. Die einzige Lichtsituation, die der Kinect Probleme macht ist grelles Tageslicht (wegen der Infrarotanteile).

Die Verwendung der Rohdaten des Tiefenbildes bringt hier noch ein Problem mit sich: Wenn die Kinect den Tiefenwert eines Pixels nicht interpretieren kann, oder die Tiefe ausserhalb der Reichweite liegt, wird dem Pixel der Wert 0 zugeordnet. Die verwendete API (Openkinect) hat jedoch Probleme mit nicht interpretierbaren Pixeln. Wenn viele Pixel nicht interpretierbar sind, zum Beispiel weil die Kamera verdeckt wird, kann es zu heftigen Rucklern und sogar zu Programmabstürzen kommen. Um diesen Aufbau einzusetzen, müsste die Stabilität deutlich erhöht werden.

Die Flexibilität dieses Aufbaus ist gering. Da alles in Processing implementiert wurde, können Änderungen nur durch Umschreiben des Codes erfolgen. Die Flexibilität könnte erhöht werden, indem die Audioerzeugung auch hier ausgelagert wird und in Processing nur die Gitteranalyse stattfindet. Die Werte für die Gitterpunkte (an oder aus) könnten dann über OSC an ein anderes Programm gesendet werden, welches diese dann interpretiert, bzw. in Audio umsetzt. Auch hier wurden nur kostenlose Programme verwendet und damit wäre die einzige Ausgabe die Anschaffung einer Kinect, sofern diese nicht vorhanden ist.

3.2.3 Skeleton Tracking

Die Kinect Kamera macht in ihrer ursprünglichen Form (an einer Xbox) eine Skeleton-Detection. Das bedeutet, die Position des Skelettes des Nutzers wird durch die Tiefenbildanalyse errechnet. Aus dieser Analyse lassen sich die einzelnen Gelenke tracken (siehe Abb. 3.3).

Es werden die Positionen der Hände im Raum ermittelt und auf die 3 Koordinaten x, y und z gelegt. Dadurch entstehen für jede Hand 3 Variablen, welche dann Kontrollwerte steuern können. Diese Kontrollwerte wurden beispielsweise auf Tonhöhe, Panning und

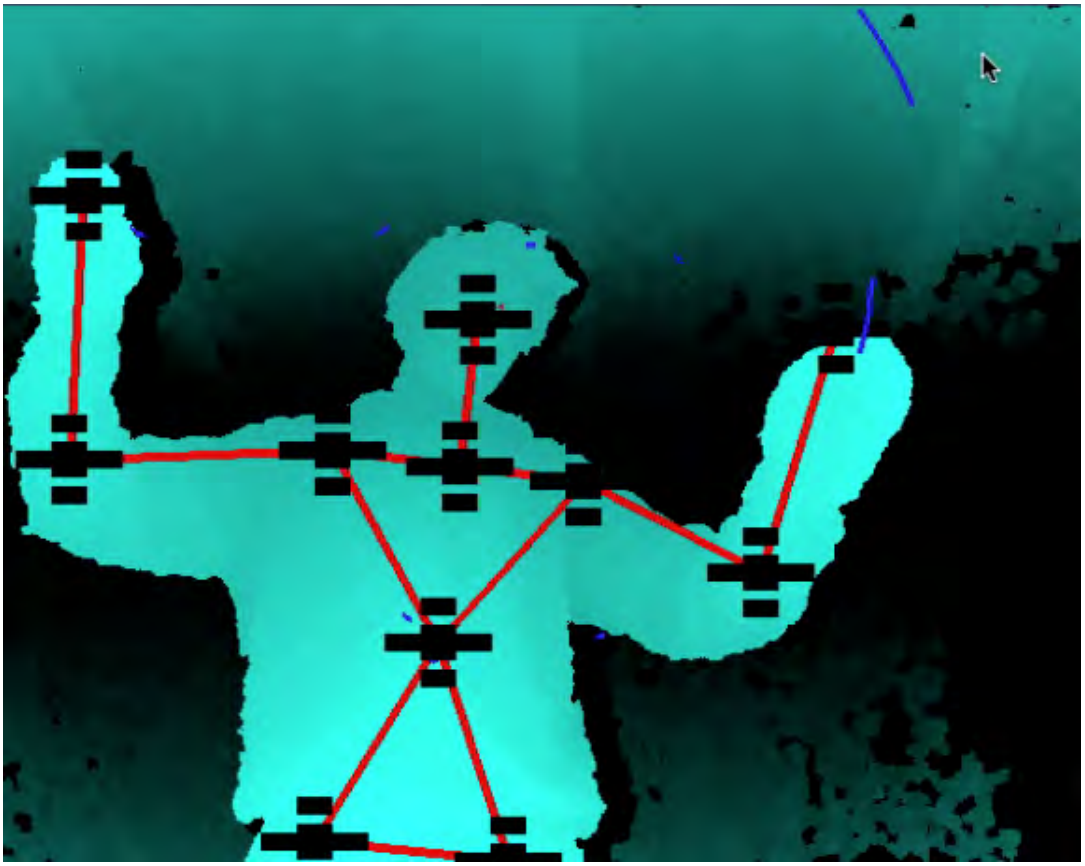


Abb. 3.3: Skeleton-Detection mit Synapse

Lautstärke eines Sounds gelegt. Das Setup ist zunächst einmal der Nachbau von Chris Viks "Kinect Controlled Granular Scretching" [4] jedoch mit eigenen Sounds. Ziel war es, ein Gefühl für die Kontrolle von Audioparametern in dieser Form zu bekommen, und dann zu sehen ob sich das System weiterentwickeln lässt.

In der Projektarbeit ergab sich die Chance, mit Heiner Kruse zusammenzuarbeiten, dem Chef des Musik Labels “Basswerk”. Dieser konnte mir die Funktionsweise von Ableton Live und Kontakt (Sampler-Software) grob erklären und mir ein Sample in Kontakt ein Sample so vorbereiten, dass es sich durch MIDI-Nachrichten steuern ließ.

3.2.3.1 Konzeptbewertung

Die Idee hinter diesem Konzept ist es, die Bewegungen der Hände zu verfolgen, da diese die ausdrucksstärksten Bewegungen machen. Der Interakteur soll mit der Musik kommunizieren können, wie ein Dirigent mit seinem Orchester. Das wird getestet, indem eine Hand getrackt wird. Die vertikale Achse steuert die Höhe eines Tones, die Entfernung zur Kamera steuert die Lautstärke und die Horizontale Achse die Spatialisierung des Tones.

Wenn man sich die Achsen bewusst macht, spielt sich das Setup so ähnlich wie ein Theremin (Ein Musikinstrument das durch das Verändern des Abstands der Hände zu einer Spule, und die daraus Resultierende Änderung im Magnetfeld, gesteuert wird.). Es fällt anfangs schwer, aber mit etwas Übung ist es möglich den Ton zu steuern. Richtig präzise wird das Ganze nicht, da es schwierig ist, eine Position im Raum genau zu treffen, aber die erzeugten Klänge sind zu einem gewissen Grad reproduzierbar. Der Ton, der gesteuert werden kann, ist aus dem Sample eines angeschlagenen Glöckchens, welches dann geloopt wird. Das unterscheidet sich nur leicht von einem Sinuston, klingt aber viel natürlicher.

Dadurch, dass die Bewegungen auf einen MIDI-Controller gemappt werden, kann man jedes erdenkliche elektronische Instrument steuern. In der Variante von Chris Vik[4] liegt eine Notenfolge vor, in der man sich bewegen kann. Dieser Ansatz kommt also einem Musikinstrument am nächsten.

Für Musiker, die mit den entsprechenden Systemen arbeiten, bietet dieser Aufbau ein Instrument mit vielen Möglichkeiten. Ein Nachteil dieser Art von Interaktion ist, dass es keine Tasten gibt. Das heißt, es muss entweder ein zweiter Musiker dabeisein, der

den Ton startet und stoppt, oder der Ton muss so konzipiert sein, dass die Lautstärke regulierbar ist, um ihn auszuschalten.

3.2.3.2 Technik Erklärung und Bewertung

Zur Erkennung des Skelets wird Synapse (siehe 2.5) verwendet. Synapse ist ein vorgefertigtes Programm, das die Kinect erkennt, sobald sie angeschlossen ist. Das Skeletontracking startet automatisch. Synapse sendet OSC Daten, so lange es selbst durch eine OSC Nachricht immer wieder mitgeteilt bekommt, welche Daten versendet werden sollen. Für die Steuerung von Synapse habe ich einen Pure Data Patch geschrieben, der Synapse im Abstand von 1500ms die OSC-Nachrichten /righthand_trackjointpos 3 und /left-hand_jointtrackpos 3 sendet. (siehe Abb. 3.4). Die von Synapse via UPD gesendeten OSC Nachrichten werden von dem Pure Data Patch mit dem Objekt "udpreceive" (in der Mitte) empfangen, im Wertebereich angepasst und auf MIDI-Controller gelegt. Die vertikalen Balken wurden zur Kontrolle der ausgehenden MIDI-Daten erstellt. Die grauen Felder in dem Patch sind Werte, die verändert werden können. Mit den Checkboxen wird das Versenden der MIDI Daten an- oder ausgeschaltet. Dies wurde implementiert, um die sogenannte MIDI-Learn-Funktion von Audioerzeuger nutzen zu können. Diese Funktion braucht die Eingangsdaten der Controller einzeln und kann diese dann automatisch auf die entsprechenden Regler legen. In dem Patch ist noch die Möglichkeit gegeben, die Variablen "high_treshold" und "low_treshold" einzustellen. Diese legen den Tiefenbereich fest, der auf den entsprechenden Controller gemappt werden soll. Da es sich nicht um eine metrische Angabe handelt, müssen die Werte ausprobiert werden. Da wir wissen, dass die Kinect ihre Tiefenwerte als Integer zwischen 0 und 2047 ausgibt, müssen die beiden Variablen auch in diesem Bereich sein. Damit entstehen für jede Hand 3 Controller. Die werden dann in den systeminternen MIDI-Bus eingespeist. Von hier kann jedes Programm die MIDI-Controller verwenden. Für den Prototyp wurde der Kontakt 5 Player verwendet und wie unter "Konzept" beschrieben gesteuert. Die Benutzeroberfläche in Kontakt ist an analoge Synthesizersysteme angelegt und hat verschiedene Knöpfe, Regler und Tasten. Um einen MIDI-Controller zuzuweisen, muss dieser nur per Drag and Drop auf den entsprechenden Regler gezogen werden.

Synapse benötigt zum Starten eine Kalibrierungspose, in der der Anwender die Arme nach oben anwinkeln muss. Dadurch wäre dieser Aufbau für Messen und Ausstellungen ungeeignet. Es gibt allerdings die Möglichkeit durch Verwendung anderer Treiber und

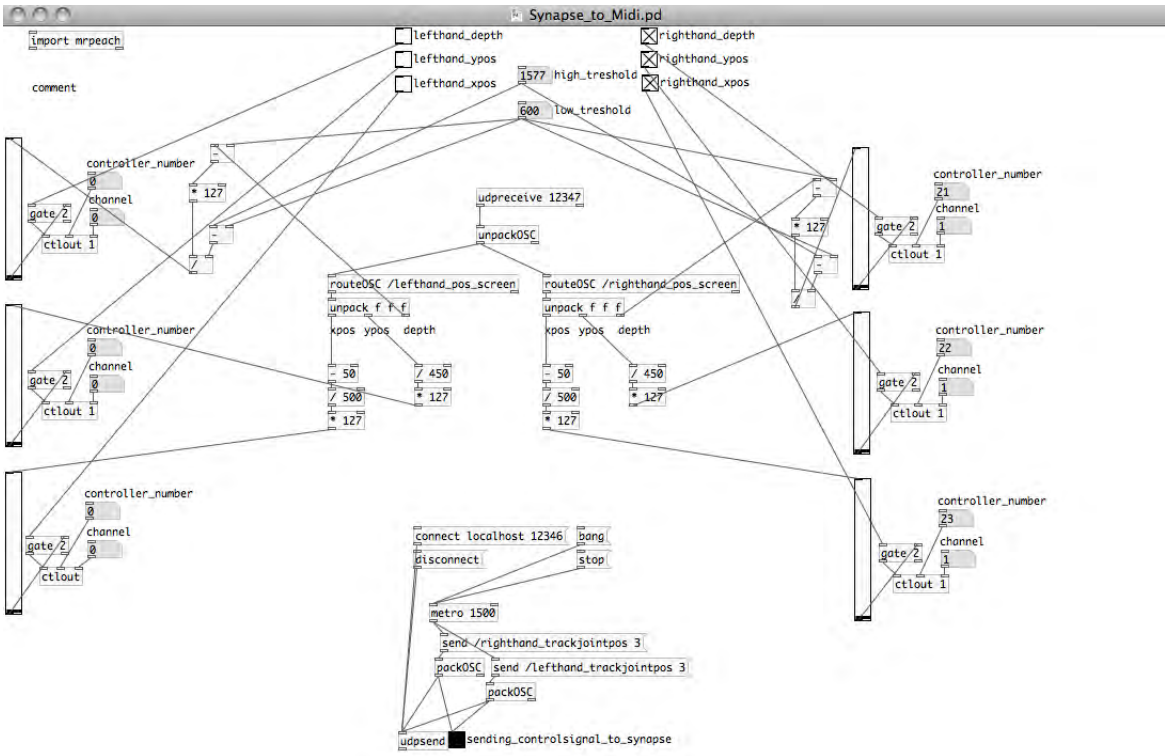


Abb. 3.4: Umrechnung von OSC zu MIDI

Librarys die Kalibrierungspose zu umgehen. Um nur die Position der Hände zu verfolgen gibt es z.B. im offiziellen Kinect SDK von Microsoft einfache Beispiele.

Die Latenz ist bei diesem Aufbau unerwartet gering. Sie liegt in einem kaum wahrnehmbaren Bereich. Die Änderung der Tonhöhe erfolgt augenblicklich. Durch Verwendung der Kinect ist auch dieses Setup unabhängig von der Lichtsituation. Synapse ist robuster als die OpenKinect Library für Processing. Es ruckelt nicht, wenn man zu nah an der Kamera ist und stürzt auch nicht ab. Dafür kann es hier vorkommen, dass die Skeleton-Detection Aussetzer hat. Vor allem mit Drehungen und verdeckten Gliedmaßen kommt die Software nicht klar. Den vereinzelten Fehlwerten ist aber mit Smoothing-Verfahren beizukommen.

Der Preis wird bei diesem Aufbau vor allem durch die Synthesizer Software in die Höhe getrieben. Im Projekt wurde mit kostenlosen Probeversionen gearbeitet, oder an Arbeitsplätzen mit entsprechenden Lizenzen (bei Basswerk). Es stellte sich heraus, dass mit Synthesizersoftware schneller hochwertige Ergebnisse zu erstellen sind, als mit den untersuchten Open Source Lösungen (die Einarbeitungszeit inbegriffen).

Der Prototyp zeigt, dass dieser Ansatz das Potential hat, in den beiden Disziplinen Bühnenperformance und Ausstellungsstück zu glänzen. Das System ist unabhängig von der Lichtsituation, robust und flexibel im Austausch seiner Komponenten. Es knüpft damit an das erste Konzept an und erweitert dieses.

3.2.4 Playstation Move

Der Aufbau mit PS-Move Controller funktioniert ähnlich, wie der Skeleton Tracking Prototyp. Es werden hierbei aber nicht die Hände getrackt, sondern ein Controller. Dieser hat gegenüber der Kinect den Vorteil, dass er Knöpfe hat, die man bedienen kann. Das Tracking funktioniert bei dem PS-Move Controller per Kamera. An der Spitze des Controllers ist ein Ball angebracht, in etwa in der Größe eines Tischtennisballes. In diesem sind LEDs verbaut, so dass er in unterschiedlichen Farben leuchten kann. Die Kamera kann nun die Farbe des Balls aus dem Bild herausfiltern und bekommt die Koordinaten Oben-Unten und Rechts-Links. Aus der Größe des Balles kann dann die Distanz zur Kamera errechnet werden. Es können mehrere Controller gleichzeitig verwendet werden. Die Bälle leuchten in dem Fall in unterschiedlichen Farben.

Auch hier werden die Daten erst als OSC-Nachrichten versendet, um dann in MIDI umgerechnet zu werden. Gesteuert wird auf der Audioebene der Kontakt-Player, wie im vorhergehenden Prototyp.

3.2.4.1 Konzeptbewertung

Das Tracking von einzelnen Punkten hat sich schon im vorigen Prototyp als funktional herausgestellt. Durch den Controller als Objekt, das man in der Hand hält wird die Entsprechung des Soundes klarer. Man kann den Controller auch weglegen, jemand anderem geben oder die Hände wechseln. Dafür ist die Faszination nicht da, die ein Interface bietet, das über Handbewegungen in der Luft kontrolliert wird. Weiterführend könnte zusätzlich die Ausrichtung des Controllers erfasst werden (wurde nicht realisiert). Der PS-Move Controller bietet dafür die technischen Voraussetzungen. Um ein Instrument daraus zu bauen, bietet dieser Controller die meisten Möglichkeiten. Im Ausstellungsbereich hat der Controller jedoch den Nachteil, dass er erst in die Hand genommen werden muss, bevor etwas passiert.

Im Prototyp wurde nur mit einem Controller gearbeitet. Aber wenn man das Konzept weiterdenkt, könnte der leuchtende Ball in einer bestimmten Farbe einem bestimmten Sound zugehörig sein. Es können hier einige Controller gleichzeitig angewendet werden. Diese können dann von verschiedenen Personen gleichzeitig benutzt werden.

3.2.4.2 Technik Erklärung und Bewertung

Für die Analyseebene dieses Prototypen habe ich ein Programm in Cinder (siehe2.5) implementiert. Das Framework ist im Umgang mit Kamerabildern etwas performanter als Processing. Da für den Playstation Move Controller noch keine Cinder API für Mac OSX existierte, musste eine bestehende API in Cinder integriert werden. Diese wurde dann zusammen mit Herr Heinen geschrieben. Mit dieser API kann die Farbe des Balls an der Spitze kontrolliert werden und die Daten der Sensoren können ausgelesen werden. Im PS-Move-Controller ist ein Gyroskop-Sensor, drei Beschleunigungssensoren und ein digitaler Kompass. Die Daten werden als Rohdaten ausgegeben. ausserdem können die Knöpfe des Controllers und ein abgestufter Trigger für den Zeigefinger auf der Unterseite genutzt werden.

Die Positionsbestimmung funktioniert durch einen Algorithmus, der der Reihe nach alle Pixel des Bildes darauf überprüft, ob sie dem Farbbereich entsprechen, indem der Ball an dem Controller leuchtet. Es werden am Anfang die Variablen “X” “Y” und “Pixelanzahl” festgelegt. Findet er einen Pixel in dem entsprechenden Farbbereich, wird “Pixelanzahl” erhöht und die Koordinaten des Pixels werden zu “X” und ”Y” hinzugefügt. Am Ende

eines Bilddurchlaufs werden “X” und “Y” jeweils durch “Pixelanzahl” geteilt und man erhält die Durchschnittskoordinaten. “Pixelanzahl” gibt die Distanz zur Kamera an. Was bei dieser Berechnung noch nicht berücksichtigt ist, ist Verdeckung. Wenn der Ball zum Teil verdeckt ist, wird er als weiter weg erkannt werden, als er in Wirklichkeit ist, da die Anzahl an Pixeln dann weniger wird.

Im Prototyp wurden zunächst nur die Position und die Knöpfe erfasst. Diese Daten werden als OSC-Nachrichten versendet. Für die Verarbeitung der Daten wurde OSCulatur verwendet, welches dann MIDI-Daten weiterschickt. Die MIDI-Daten werden dann von Kontakt empfangen und genau so behandelt wie die des Kinect Prototypen. OSCulator ist ein fertiges und kostenpflichtiges Programm, das OSC-Nachrichten verarbeiten kann. Es wurde eine kostenlose Version getestet, die im Funktionsumfang nicht eingeschränkt war, sondern nur in der Nutzungszeit. OSCulator bietet eine einfache Möglichkeit, OSC-Daten zu MIDI-Daten umzuwandeln, jedoch kann nichts programmiert werden, was ein großer Nachteil gegenüber Pure Data als Übersetzer ist.

Wichtig ist hierbei, dass die Farbe des Controllers so gewählt wird, dass es nichts im Raum gibt, das die gleiche Farbe hat. Diese Einschränkung muss man bei einer Verwendung als Ausstellungsstück bedenken. Da der Ball leuchtet und es an Messebesuchern wenige Gegenstände gibt, die farbig leuchten, ist trotzdem mit wenigen Fehleingaben zu rechnen.

Da das Tracking von dem leuchtenden Ball an der Spitze der Controller abhängig ist, hat der Controller ebenso wie die Kinect Probleme mit zu hellem Licht. für die Ausstellungssituation hat ein Controller den Nachteil, dass er mitgenommen oder beschädigt werden kann. Er ist also, was die Hardware angeht, anfälliger als die Kinect. Die Latenz war in dem untersuchten Prototyp gleichzusetzen mit der Latenz der Kamera. Die Interpretation des Bildes ist sehr simpel und geht daher schnell.

Der Preis dieses Setups wird wie beim Vorgänger hauptsächlich durch die verwendete Synthesizersoftware in die Höhe getrieben.

4 Ergebnisse der Untersuchung

4.1 Aufbau

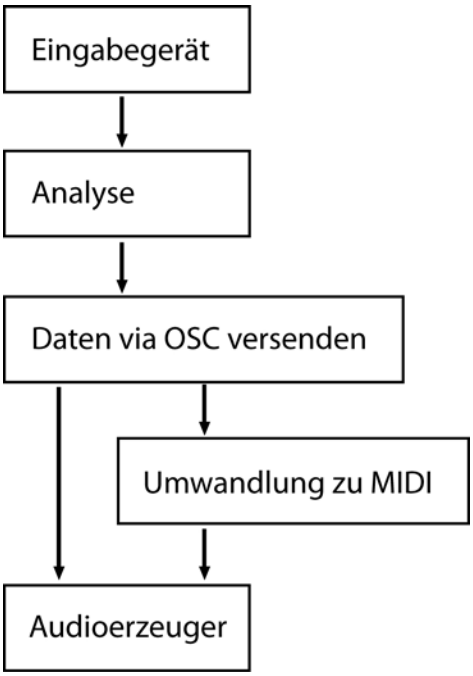


Abb. 4.1: Systemarchitektur

Eingabe, Analyse und Audioerzeuger voneinander zu trennen (siehe Abbildung 4.1) hat sich als flexibelster und sinnvollster Aufbau erwiesen. Durch die gemeinsame Schnittstelle OSC sind neue Kontrollwerte einfach hinzuzufügen. Pure Data und OSCulator bieten einfache Möglichkeiten, OSC Nachrichten zu empfangen und in MIDI umzuwandeln. Wenn Programme OSC direkt unterstützen können sie sogar direkt genutzt werden. In dieser Architektur können einzelne Teile ersetzt, oder von einander unabhängig verändert werden. In der Übertragungsebene muss lediglich festgelegt werden, ob es sich in dem übertragenen Wert um einen Trigger handelt oder um einen Kontrollwert. In OSC wurde ein Kontrollwert als eine Gleitkommazahl zwischen 1 und 0 festgelegt, ein Trigger als entweder 1 oder 0. In der Umwandlung zu MIDI muss der Wert auf 1 oder 0 als Kontrollwert, oder eine Ganzzahl zwischen 0 und 127 umgerechnet werden, da das MIDI-Protokoll dies vorschreibt.

Ein weiterer großer Vorteil des Aufteilens des Systemes in die beiden Hauptteile Analyse und Audioerzeuger ist, dass die beiden Teile auf jeweils unterschiedlichen Rechnern laufen

könnten. Da es für die Kinect auf Windows weitaus bessere Frameworks gibt (Microsoft SDK und OpenNI) könnte zum Beispiel die Analyse auf einem Windows Rechner laufen und die Steuerungsdaten könnten dann via OSC über das Netzwerk auf einen Mac geschickt werden, wo die Audioerzeugung einfacher ist.

4.2 Open Source oder kommerzielle Lösungen

Was die Übertragung angeht, hat sich Open Sound Control als einfachster Standard erwiesen. Durch den einfachen Aufbau von OSC-Nachrichten (siehe 2.5) und die Übertragungsmethode über das Netzwerk ist OSC MIDI weit überlegen. Um MIDI zu nutzen muss ein MIDI-Bus vorhanden sein, um es an einen anderen Computer zu senden braucht man sogar ein MIDI Interface. OSC Geräte können einfach per Netzwerk miteinander verbunden werden. Da nur Kontrolldaten übertragen werden, ist die Datenmenge sehr gering und es tritt keine wahrnehmbare Latenz auf. Den einzigen Vorteil, den MIDI gegenüber OSC noch hat, ist seine Verbreitung. Mehr und mehr große Firmen implementieren eine OSC-Unterstützung in ihren Programmen.

Es fällt auf, dass die Open Source Lösungen alles können, was man sich vorstellen kann, wenn man sich eingehend mit ihnen beschäftigt. Die Offenheit bietet die Möglichkeit, das, was man vermisst, einfach selbst zu programmieren. Supercollider zum Beispiel bringt keinerlei GUI mit, alles ist rein Text basiert. Aber der Supercollider-Server (siehe 2.6) wird mit OSC Nachrichten gesteuert. Das bedeutet, dass jeder, der die Zeit und Muße dazu hat, sich eine GUI und einen eigenen Synthesizer aus den gegebenen Bausteinen zusammensetzen kann.

Wie fast immer wenn man Open Source mit kommerziellen Lösungen vergleicht, ist Open Source das Equivalent zu einzelnen Ziegelsteinen, während kommerzielle Lösungen ganze Wände oder sogar Fertighäuser anbieten. Einzig der Vergleich von Pure Data und Max / MSP weicht von dieser Darstellung ab. Beide Programme basieren auf dem selben Prinzip, ja sogar auf dem selben Ursprungsprogramm. Es ist mit beiden Programmen möglich, die gleichen Dinge zu realisieren, und die Programme entwickeln sich erst jetzt langsam auseinander. Durch die direkte Anbindung in Ableton Live mit Max for Live hat Max ein Alleinstellungsmerkmal gegenüber Pure Data. Max Patches können direkt in Ableton integriert werden.

Bei allen untersuchten Open-Source Programmen braucht man um den Umgang zu

erlernen länger als in kommerziellen. Dies fällt vor allem beim Vergleich von Pure Data und Max auf. Für Max gibt es eine größere Community und der Hersteller (Cycling74) hat auf seiner Homepage Tutorials.

Die großen kommerziellen Programme, die untersucht wurden (Ableton Live und Kontakt) haben beide den Vorteil, dass man sehr schnell einen fertigen Synthesizer starten kann. Man ist jedoch auf die Funktionalitäten und Ansatzpunkte angewiesen, die einem zu Verfügung gestellt werden und muss sich in diesem Rahmen bewegen. Dazu ist zu sagen, dass selbst diese Möglichkeiten Raum lassen für eine unendliche Klangvielfalt, aber die gesamte Vorgehensweise ist eine andere.

4.3 Abwägung Eingabegeräte

Als Eingabegeräte stehen Video, Kinect und Playstation-Move zur Auswahl.

Das Videobild kann sich in dieser Form nicht durchsetzen. Es gibt bessere Videoanalysealgorithmen, die hier nicht untersucht wurden, doch allein durch das Fehlen der Tiefeninformation hat die Videoanalyse einen Nachteil gegenüber den anderen Eingabegeräten, der sich kaum ausgleichen lässt. In der Geschwindigkeit konnte die Videoanalyse zwar mit den anderen Systemen mithalten, aber die Analyse ist zu flach, bietet zu wenige Parameter. Es gäbe Möglichkeiten, mehr Parameter zu bekommen, z.B. Motionvektoren zu erzeugen und den Optical Flow eines Bildes zu untersuchen, Object Tracking, oder Gestenerkennung. All diese Methoden erfordern jedoch einen gewissen Aufwand, um gute Ergebnisse zu liefern, und noch mehr Aufwand, um flüssig, bzw. mit geringer Latenz zu funktionieren.

Bleiben noch die Kinect und der PS-Move Controller. Diese Geräte haben beide ihre Vor und Nachteile. Der große Vorteil der Kinect ist die Wirkung auf den Betrachter. Irgendetwas einfach mit den Händen in der Luft zu steuern löst (heutzutage noch) Faszination bei Interakteuern oder Betrachtern aus. Dazu muss ein Nutzer nicht einmal den technischen Hintergrund verstehen. Der große Nachteil ist die Abwesenheit von Knöpfen. Diesem Nachteil könnte natürlich Abhilfe geschaffen werden, indem man Fußpedale benutzt, oder Knöpfe selbst auf eine andere Art integriert, aber damit wäre die Wirkung der Interaktion ohne Berührung weg. Auch funktioniert die Skeletondetection nicht immer zuverlässig. Die Freihändigkeit hat den Vorteil, dass bei den Bewegungen nicht auf ein zusätzliches Gerät acht gegeben werden muss. Bewegungen können direkt

und natürlich ausgeführt werden.

Der PS-Move Controller hat das gleiche Interaktionsprinzip, nämlich räumliches Tracking. Er bietet gegenüber der Kinect den Vorteil, dass der Controller zusätzlich Knöpfe hat. Die Positionserkennung hat gegenüber der Kinect Vor- und Nachteile. Der PS-Move Controller braucht keine Skelett-Erkennung, das heißt sobald der Controller im Bild ist, gibt es Positionsdaten. Dafür muss das System immer auf eine Farbe eingestellt werden, die im Raum noch nicht vorhanden ist. Beide Systeme haben Probleme mit Verdeckung. Dies wäre mit Hilfe der anderen Sensoren bei dem PS-Move Controller lösbar, würde aber noch einiges an Realisierungsaufwand bedeuten.

Ein Objekt in der Hand zu halten, verändert die Art, wie ein Interakteur sich bewegt. Daher ist, um das anfängliche Ziel zu erfüllen, Bewegung in Musik umzuwandeln, die Bewegungsanalyse mit der Kinect-Kamera der natürlichste Weg. Zur interpretation der Kinect-Daten bietet sich Synapse2.5 an. Es bietet in seiner Grundausstattung alles was benötigt wird und zwar die Koordinaten aller Gelenke als OSC Nachrichten. Es ist zusätzlich quelloffen und damit erweiterbar.

4.4 Abwägung Audioerzeuger

Die getesteten Audioerzeuger haben sehr unterschiedliche Herangehensweisen und bringen alle ihre Vor- und Nachteile mit sich.

Super Collider ist als Audioerzeuger durchaus interessant, da es die Freiheit mit sich bringt, einen Synthesizer so zu programmieren, wie man es möchte, um diesen danach über OSC so zu steuern, wie man es möchte. Die Möglichkeiten sind theoretisch unbegrenzt. Aber um diese großen Ideen umzusetzen, muss man auf einem ziemlich niedrigen Level programmieren, und braucht viel Aufwand, um überhaupt zu irgendwelchen Ergebnissen zu kommen. Um eine einfache Synthesizer Definition zu schreiben, die nur eine Sinuswelle generiert muss man sich schon tief in die Programmiersprache SCLang einarbeiten. Die Frustration war hier hoch und führte zu einem Wechsel der Audiosyntheseumgebung.

Die Patcher-Sprachen (Pure Data und Max/MSP) haben gegenüber Supercollider den Vorteil, dass es eine grafische Benutzeroberfläche gibt. Das ist schnell zu erlernen. Das Patching-Paradigma bietet eine grafische Repräsentation der erzeugten Audioobjekte und ihrer Verknüpfungen untereinander. Um damit interessante Audioobjekte zu erzeugen,

muss man sich immer noch in die Theorien der Audiosynthese an sich einarbeiten, aber die Programmierung ist in den Patcher-Sprachen doch wesentlich einfacher.

Am schnellsten kommt man mit vorgefertigten Softwaresynthesizern zu einem Ergebnis. Ableton und Kontakt haben fertige Syntheseobjekte oder Samplerobjekte, in denen nur noch ein paar Parameter eingestellt oder Sounddateien geladen werden müssen und schon kann man sie benutzen. Wenn man schon Erfahrung mit Hardwaresynthesizern hat, fällt der Einstieg in diese Programme noch leichter, da sie meist mit grafischen Repräsentationen der Knöpfe und Regler arbeiten. Man legt die gewünschten MIDI Controller einfach auf die gewünschten Regler, und schon ist eine Tonerzeugung gegeben. Die Möglichkeiten, diese zu beeinflussen, sind natürlich immer schon vorgegeben und der Preis solcher Software ist auch nicht zu unterschätzen.

Im Rahmen des Projektes haben sich die Patcher-Sprachen als am besten nutzbar herausgestellt. Sie sind nicht so komplex wie die Low-Level Programmiersprache Supercollider und bieten trotzdem mehr Freiheiten als vorgefertigte Softwaresynthesizer. Max/MSP ist zugänglicher als Pure Data, da eine Firma dahinter steht. Dadurch ist bei der Entwicklung und auch bei den Tutorials eine klare Linie erkennbar, was bei Pure Data nicht der Fall ist, da die Tutorials von Einzelpersonen produziert und im Netz verstreut sind. Max hat viele Objekte, die das Interfacedesign erleichtern und ist auf eine gute Benutzbarkeit ausgerichtet. Pure Data bietet dafür große Freiheiten, auch eigene Erweiterungen zu programmieren, aber ist im allgemeinen schwieriger zu benutzen. Trotzdem fällt die Entscheidung in diesem Projekt auf Pure Data, zumindest als Middleware um die Daten von Synapse zu interpretieren. Für den Umgang mit Datenströmen, (so wie die OSC-Nachrichten, die von Synapse gesendet werden) liegt das Patcher Paradigma einfach nahe. Auch im Bereich Audiosynthese kann mit Pure Data einiges realisiert werden.

5 Zwischenfazit

Am Ende der Voruntersuchung steht ein Konzept zur Entwicklung und Integration von Motioncontrollern. Die letzten Prototypen (Kinect und Playstation Move) haben einiges Potential, um als bewegungsgesteuertes Musikinstrument genutzt zu werden. Das gitterbasierte Konzept steht etwas aussen vor, hat aber auch seine Anwendungsgebiete gefunden. Es ist für Messen gut geeignet und kann zusammen mit einer anderen Installation von Lichtfaktor, dem “Golden Fluid” ausgestellt werden, um dieses mit einer Audioebene zu bereichern.

Es wurde ein grundlegender Aufbau entwickelt, in den sich Bausteine wie Eingabegeräte oder Audioausgabegeräte einfügen lassen. Es wurden verschiedene Eingabegeräte und Audioerzeuger getestet. Die Eingabegeräte wurden auf ihre Eignung zur Umwandlung von Bewegungs- in Steuerdaten untersucht. Die Arbeitsweise der Audioerzeuger wurde unter die Lupe genommen und abgewogen mit welchem Audioerzeuger man ohne Vorkenntnisse in der Audioerzeugung am schnellsten in die Materie einsteigen kann. Die verschiedenen Entwicklungsparadigmen der Audioerzeuger wurden untersucht. Es wurde untersucht, wie Programme aus dem Bereich der Audioerzeugung miteinander kommunizieren können. Die beiden großen Übertragungsprotokolle OSC und MIDI wurden gegeneinander abgewägt.

Während des Projektes wurde auch deutlich, wie schwierig es ist, interessante Klänge generativ zu erzeugen. Ich ging ohne Vorwissen im Bereich der Audiosynthese an das Thema heran, und merkte, dass man einiges an Grundwissen braucht, um Musik mit dem Computer zu machen. Wenn man sich dann eingearbeitet hat, bieten Programme wie Pure Data spannende Möglichkeiten. Der Kreativität sind kaum Grenzen gesetzt.

Teil II

Verfeinertes System mit Kinect und Pure Data

6 Einleitung

Als nächster Schritt steht die Realisierung eines Systemes, das aus einzelnen verfeinerten Teilen der erstellten Prototypen besteht. So soll ein wirklich spielbares Instrument konstruiert werden. Mit der Auswahl der Kinect kann nun die Skeleton-Detection robust gemacht werden und es kann eine interessante Audioebene dazu entwickelt werden. Der Grundstein für die Entwicklung eines Musikinstrumentes, das Bewegungen interpretiert, ist gelegt. Aus dem entstandenen Grundprinzip für den Aufbau können verschiedene Audiocontroller entwickelt werden und in eine Umgebung gebracht in der sie zusammen angewendet werden. Pure Data kann dabei sowohl als Middleware, als auch als Audioerzeuger funktionieren.

7 Analyse

7.1 Eingangsdaten

Die grundlegenden Eingangsdaten des Systems werden von Synapse als kartesische Koordinaten geliefert (siehe 3.2.3). Die horizontale Achse ist dabei die X Achse, die vertikale die Y Achse und die Tiefenachse wird im weiteren mit D benannt.

Die in Synapse verfolgten Gelenkpunkte sind: linke Hand, linker Ellenbogen, Kopf, rechter Ellenbogen, rechte Hand, Torso, linkes Knie, linker Fuß, rechtes Knie, rechter Fuß. Synapse unterstützt drei Varianten des Trackings für die einzelnen Gelenkpunkte.

Die erste Variante gibt Screen-Koordinaten aus, also X und Y Werte, die relativ zum Bildausschnitt erzeugt werden. Der Wertebereich ist die Auflösung, in der das Bild gerendert wird (640*480) . Das bedeutet, dass eine Bewegung in der X,Y Ebene als stärker wahrgenommen wird, je näher sich das betreffende Gelenk an der Kamera befindet. Dieser Effekt muss nicht unbedingt eine Schwäche sein, man kann ihn eventuell sogar nutzen. Auf jeden Fall aber muss er im Design berücksichtigt werden. Die D Werte werden absolut in Millimetern ausgegeben.

In der zweiten Variante werden Welt-Koordinaten ausgegeben , also absolute Koordinaten, die von der perspektivischen Verzerrung berreinigt sind. Damit wird eine Bewegung über eine bestimmte Distanz überall gleichstark ausgegeben, egal wie weit sich der Interakteur von der Kamera weg befindet. Die Werte werden alle in Millimetern ausgegeben, wobei sich der Nullpunkt in der Kinect Kamera befindet und die D Achse mittig aus der Kamera herauskommt. Das bedeutet, dass X und Y wenn sich ein Gelenk in der Bildmitte befindet 0 sind, und positiv oder negativ werden, je nachdem in welche Richtung man es bewegt.

Variante drei gibt die Koordinaten relativ zum Torso aus. Die Koordinaten werden genau wie bei den Welt-Koordinaten in Millimetern ausgegeben. Hierbei bildet der Torso den Nullpunkt des Systemes.

7.2 Abgeleitete Werte

Aus den Eingangsdaten werden nun nutzbare Werte abgeleitet, damit diese später Audiovariablen steuern können.

Die Koordinaten der Gelenke sollen als Variable zur Verfügung stehen. Dafür bieten sich zunächst relative Koordinaten (relativ zum Torso) an, um möglichst wenig Kalibrierungsaufwand zu haben. Der Interaktionsraum ist dann allerdings nicht so groß, da er sich zusammen mit dem Interakteur verschiebt. Wenn sich ein Interakteur bespielsweise seitwärts neigt, um mit der Hand weiter zu einer Seite zu kommen, bewegt sich der ganze Interaktionsraum mit. Benutzt man allerdings Welt- oder Bildschirmkoordinaten, muss immer ein Grenzwert für die Tiefe (D) festgelegt werden, in dem sich eine Variable bewegt. Um dieses Problem zu lösen habe ich eine automatische Grenzwertverschiebung programmiert, die sich an der absoluten Position des Torso orientiert und ansonsten Bildschirmkoordinaten benutzt. Die Bildschirmkoordinaten bieten den Vorteil, dass man die Bildschirmausgabe von Synapse als visuelles Feedback nutzen kann, um zu sehen wo man sich befindet. Das hat auch den Effekt, dass sich der Interaktionsraum verändert, sobald man den Abstand zur Kinect verändert. .Dieser Effekt wurde im vorigen Abschnitt bereits angesprochen. Ein Ansatz zum umgehen dieses Problemes wäre, eine eigene Bildschirmausgabe zu programmieren und dann mit Weltkoordinaten zu arbeiten. Das hätte jedoch den Rahmen dieser Arbeit gesprengt. Ausserdem hat der Effekt des sich verändernden Interaktionsraumes auch den Vorteil, dass die visuelle Repräsentation immer erhalten bleibt.

Da die absoluten Koordinaten von Synapse praktischerweise in Millimetern ausgegeben werden, können diese zur Berechnung von Geschwindigkeiten für die Gliedmaße genutzt werden. Die Koordinaten kommen als Datenstrom an, daher muss lediglich der vorherige Wert von dem aktuellen abgezogen werden, um den Bewegungsvektor auf einer Achse zu bekommen und damit die Geschwindigkeit in einer Richtung. Um für ein Gelenk eine absolute Geschwindigkeit zu errechnen wurde der Betrag der drei Bewegungsvektoren gebildet.

$$|\vec{v}| = \sqrt{v_x^2 + v_y^2 + v_d^2}$$

Ein weiterer interessanter Wert ist der Abstand zwischen zwei Gelenken. Dieser Wert wird so berechnet wie die Geschwindigkeit, nur dass statt den zwei Werten des selben Gelenkes zu unterschiedlichen Zeitpunkten die Ortsvektoren von zwei Gelenken genommen werden.

$$|\vec{d}| = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (d_1 - d_2)^2}$$

Auf diese Art kann also jeder Abstand zwischen zwei Gelenken berechnet werden. Es wird zunächst nur der Abstand zwischen den Händen ausgegeben, da dieser für die Interaktion interessant ist. Auch der absolute Abstand zur Kamera wird als Variable ausgegeben. Dafür wird einfach der D-Wert des Torsos benutzt.

Pro Gelenk werden also folgende Daten ausgegeben: x, y, d, speed, speed_x, speed_y, speed_d. Zusätzlich wird die Variable distance_hands und die Variable distance_abs (Abstand des Torsos zur Kamera) bereitgestellt.

7.3 Mapping

Die Bisherigen Koordinaten bestehen aus nicht normalisierten Werten, die sich alle in unterschiedlichen Wertebereichen bewegen. In Pure Data werden Datenströme als Linien dargestellt und diese Linien können während der Laufzeit verändert werden. Damit ist es möglich, während der Laufzeit Zuweisungen von Variablen zu ändern. Damit das funktioniert, müssen alle benutzten Variablen zunächst den gleichen Wertebereich haben. Für die Anwendung im Audiobereich gibt es zwei Wertebereiche die wichtig sind. Für MIDI-Anwendungen ist das der Bereich von 0-127; Für andere Audioanwendungen, wie z.b. die Lautstärke ist es der Bereich von 0-1. Da viele Objekte in Pure-Data direkt Eingangsvariablen im MIDI Werteberiech haben entschied ich mich dafür, alle Werte zunächst in einen Wertebereich von 0-127 zu legen (Die einzige Ausnahme bildet distance_abs, welche als absoluter Wert in Millimetern bleibt). Um das zu gewährleisten musste ein solides Mapping-Objekt in Pure Data geschrieben werden, dem ein Eingangs- und ein Ausgangs-Wertebereich angegeben wird, und das die Werte auf die Grenzwerte legt, wenn sie aus dem Wertebereich austreten. Seltsamerweise ist ein solches Objekt nicht in der normalen Distribution von Pure Data enthalten, obwohl auch andere Programmierer ein solches Objekt oft brauchen könnten.

7.4 Space Trigger

Ein zentrales Element in Pure Data ist die “Bang” Message (im weiteren Bang genannt). Ein Bang wird benutzt um Ereignisse irgendeiner Form auszulösen. Alles was man als Ablauf in Pure Data programmieren kann, muss durch einen Bang gestartet werden. Das gestartete Event kann ein Ton sein, eine Berechnung, das Speichern eines Wertes, das Senden eines Wertes an ein anderes Programm, oder irgendetwas anderes. Somit liegt es nahe, ein Objekt zu erstellen, das solche Bangs erzeugen kann. Um Events mit einem präzisen Timing auslösen zu können erstellte ich das Space Trigger Objekt. Das ist ein Bereich im Koordinatensystem der Kinect, der bei Berührung einen Bang feuert, eine Art virtueller Knopf. Dieser Knopf hat drei Koordinaten, eine Größe und einen Cooldown, der in Millisekunden angegeben wird. Der Cooldown ist wichtig, damit der Knopf nicht einen Bang nach dem anderen abfeuert, wenn sich ein Gelenk an dem Triggerpunkt in Ruhe befindet. Wenn sich das Gelenk aus dem Trigger-Bereich herausbewegt fängt ein Timer mit der Cooldown-Zeit an herunter zu zählen. Die Koordinaten werden einmal pro Frame übermittelt. Bei einer Framerate von 30fps (siehe 2.4) ergibt sich, dass die Cooldown-Zeit mit mindestens 33ms (1000ms / 30frames) angegeben sein muss, damit der Trigger nicht von einem ruhenden Gelenk wieder und wieder ausgelöst wird.

8 Komposition

Bisher wurden Interaktionsparadigmen erstellt und auch Variablen. Damit es eine Audioentsprechung gibt, wird in Pure Data eine Audioebene erstellt, die eine Anzahl von Eingangsvariablen besitzt. Um eine Audioebene mit entsprechenden Variablen zu erstellen, musste ich mich tiefer in das Thema Audiosynthese einarbeiten. Ich wähle einige Methoden aus und erkläre die dafür wichtigen Grundlagen der Audiosynthese. Dann werden die Variablen der Interaktionsebene auf die Audioebene gelegt und es wird untersucht, welche Entsprechungen als passend empfunden werden.

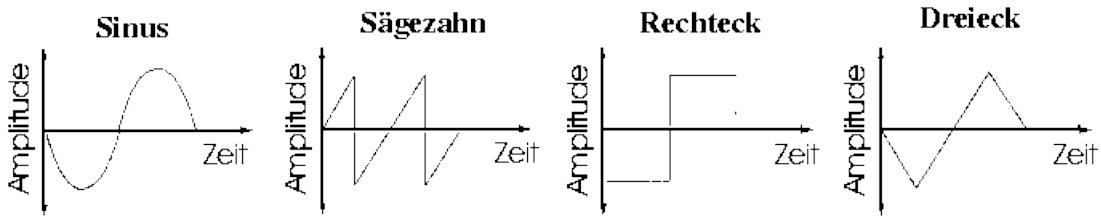
8.1 Audioerzeugung

In der Audioerzeugung gibt es die beiden Möglichkeiten, einen Sound entweder vollständig zu generieren, oder ein Sample oder Eingangssignal zu benutzen, das dann bearbeitet wird. Ich versuche, den Klang möglichst vollständig generativ zu erzeugen, da es darum geht, Klänge zu erzeugen, Bewegung in Musik umzuwandeln und nicht nur bereits erzeugte Klänge zu steuern.

Um einen vielseitigen Synthesizer zu programmieren, sind weitreichenre Kenntnisse der verschiedenen Audiosyntheseformen nötig. Sowohl das Einarbeiten als auch die Erklärung hätte den Rahmen dieser Arbeit gesprengt. Deshalb werden hier nur die wichtigsten Grundlagen erörtert.

Die hier angesprochenen und weitere Techniken der Audiosynthese werden in Curtis Roads - The Computer Music Tutorial[22] noch näher erläutert. Hier soll eine möglichst einfache Erklärung der verwendeten Techniken gegeben werden.

Abb. 8.1: Wellenformen



8.1.1 Synthesizer

Um Sounds völlig generativ zu erzeugen, wird in der Regel mit Oszillatoren gearbeitet, die ein Signal in einer bestimmten Wellenform erzeugen. Die Form des Signals kann einer Sinuswelle, einem Rechteck, einem Dreieck oder einer selbst ertsellten Form entsprechen (siehe Abb.8.1). Auch zufällig generierte Signale (Rauschen) werden oft verwendet. Diese Wellenformen können in Frequenz und Amplitude verändert werden und mehrere Signale können miteinander kombiniert werden. Hierzu stehen verschiedene mathematische Operatoren zur Verfügung. Signale können beispielsweise addiert, subtrahiert, multipliziert und dividiert werden. Durch Frequenzmodulation ist es möglich, dass das Ausgangssignal eines Oszillators auf die Frequenz eines anderen gelegt wird. Additive Synthese beschreibt das Addieren vieler Oszillatoren, subtraktive Synthese geht von einem Ausgangssignal aus und erreicht verschiedene Töne durch Subtraktion. So gibt es unzählige Möglichkeiten, durch geschickte Kombination von Signalen Schwebungen, Vibratos, und andere interessante Effekte entstehen zu lassen.

8.1.2 Filter

Signale können durch Filter geleitet werden, um ihnen interessante Charakteristiken hinzuzufügen. Der am meisten benutzte Filter ist wohl der Reverb (Hall). Hallfilter können auf verschiedene Arten realisiert werden, eine davon ist das Feedback Delay Network. Dabei werden mehrere verzögerte Spuren des eigentlichen Signals aufgezeichnet und in absteigender Lautstärke abgespielt. Der Output dieser Delay-Spuren wird dann wieder aufgezeichnet und erzeugt ein Feedback. Dieser Hallfilter ist interessant, weil man

ihn mittels “falschen” Parametern zweckentfremden, und so spannende Effekte erzeugen kann.

Ein weiterer interessanter Filter für diese Arbeit ist der Bandpass-Filter. Dieser Filter lässt nur einen bestimmten Frequenzbereich passieren, und dämpft den Rest ab. Die Klangcharakteristik ist vom “Wahwah”-Sound bei Gitarren bekannt. Der Filter ist deshalb interessant, weil der Grad der Abdämfung eines Soundes sich gut auf Bewegungen oder Positionen übersetzen lassen könnte. Man könnte den Klang der verschiedenen Filter-Einstellungen als offen und geschlossen beschreiben.

Verzerrer funktionieren nach dem Grundprinzip, ein Signal zu übersteuern, so dass Clipping-Effekte entstehen. Dieser Effekt ist von der E-Gitarre bekannt, kann allerdings auch auf jedes andere Signal angewendet werden.

8.2 Tanz-Analyse oder Instrument?

Während dem Testen von verschiedenen Abbildungen von Kontrollparametern auf musikalische Elemente stellten sich zwei grundlegende Richtungen der Analyse heraus. Eine Tanz-Analyse würde bedeuten, einen Tänzer vor das System zu stellen, und diesen einfach tanzen zu lassen. Der Tänzer müsste dabei nicht unbedingt wissen, wie das Mapping funktioniert. Dabei müssen andere Parameter erfasst werden als für das erstellen eines Bewegungscontrollers. Um Tanz zu analysieren, können allgemeine Parameter benutzt werden, wie die Bewegungsintensität aller Gliedmaße zusammen, oder die Position des Tänzers im Raum.

In der Diplomarbeit von Sofia Dahl[12] werden Emotionen in einer Tabelle bestimmten Bewegungsparametern zugeordnet (siehe Tab. 8.2).Diese Parameter lassen sich aus den Kinect-Daten leicht berechnen, Der gemittelte Abstand zwischen den äußeren Extremitäten (Hände und Füße) geben an, ob die Bewegung allgemein groß oder klein ist. Die gemittelten Geschwindigkeiten ergeben einen Wert für die allgemeine Geschwindigkeit. Anhand der Änderungsrate dieser Geschwindigkeit, könnte eine Aussage über die Gleichförmigkeit der Bewegung getroffen werden. Allerdings wäre das eine Analyse über einen bestimmten Zeitraum, also nicht direkt. Das Ziel dieser Arbeit ist es, ein direktes Interface zu erstellen (siehe 2.2). Diese Art der Tanzanalyse ist ein interessanter Bereich, in dem sich durch die Kinect neue Möglichkeiten eröffnen. Das bietet Stoff für eine eigene

Abb. 8.2: Emotionstabelle aus Dahl (2005)

	amount *sound level*	speed *tempo*	fluency *articulation*	regularity *tempo var.*
Happiness	large *high*	fast *fast*	jerky *staccato*	 *small*
Sadness	 *low*	slow *slow*	smooth *legato*	 *final ritard*
Anger	 *high*	 *fast*	jerky *staccato*	 *small*
Fear	small *low*	 *fast*	 *staccato*	irregular *large*

Arbeit und wird daher hier nicht weiter verfolgt.

Als Instrument legt das System die Rahmenbedingungen fest. Der Interakteur ist hier eher als Musiker zu verstehen, das bedeutet er muss sich mit der Funktion des Systems vertraut machen. Hierbei liegt das Augenmerk darauf, besonders präzise Töne oder andere Events zu treffen zu können. Töne und Effekte müssen nachvollziehbar und reproduzierbar sein. Der Interakteur muss sich hier also in der Struktur des Systems einfinden, bzw. benutzt es zur gezielten Kontrolle der Musik. Die Interaktion muss gelernt werden (oder zumindest kurz erklärt).

8.3 Instrument

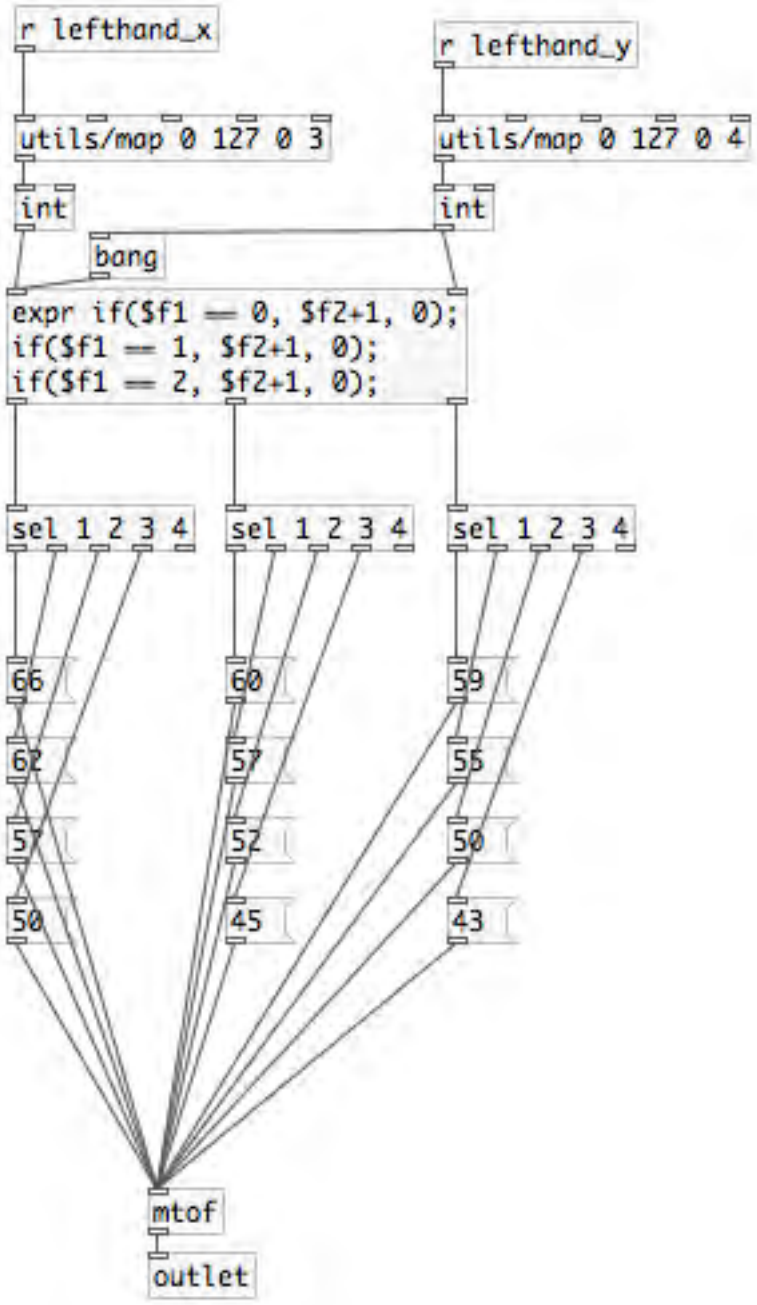
Das Sceleton-Tracking hat sich bei dem Test als Prototyp (siehe 3.2.3) als nachvollziehbares Eingabeparadigma erwiesen. Doch die Benutzung von absoluten Koordinaten als Steuerungswerte bringt einige Probleme mit sich. Es ist schwierig, sich genaue Positionen im Raum zu merken. Die Tonhöhe mit der vertikalen Achse zu verknüpfen, ist für den Anwender gut nachzuvollziehen. Jedoch ist es schwierig in einem nicht gerasterten Frequenzspektrum Töne genau zu treffen. Beim Theremin wird dies dadurch gelöst, dass der Interakteur die Finger der Hand einzeln ausstreckt, oder beugt. Diese Möglichkeit gibt es

bei der Kinect nicht, da das Tracking dafür nicht genau genug ist. Dieses Problem wird durch eine Rasterung gelöst. Hier kommt das Konzept des Gitter-Prototypen zum tragen (siehe 3.2.2) Es wird ein ähnliches Gitter aufgespannt, in dem sich harmonisch zueinander passende Töne befinden. Dazu benutzte ich eine Gitarre, auf der ich eine passende Folge von Arpeggios heraussuchte (D, Am, G). Aus den Akkorden wurden jeweils vier Töne ausgesucht und übereinander gelegt. Die einzeln gespielten Gitarrentöne wurden über den Line-Eingang in den Computer gespielt und mit einem Pure-Data Objekt konnte dann einfach die Grundfrequenz gemessen werden. Diese Grundfrequenz wurde in Midi-Notenwerte (als Zahl zwischen 0 und 127) umgewandelt. Die so entstandenen Harmonien wurden dann Seitlich nebeneinander angeordnet. So liegen Töne nebeneinander, die als Abfolge zu einander passen (siehe Abb. 8.3). Dann wurden die X und Y Werte der linken Hand auf das Gitter gelegt, so dass immer genau ein Ton als ausgelöst gilt. Zuletzt werden die MIDI-Töne wieder in Frequenzen umgerechnet. Der Umweg von Frequenzen über MIDI zu Frequenzen wird beschritten, um die Möglichkeit zu haben, einfach andere Töne in das Gitter einzutragen. MIDI-Töne sind immer ganzzahlig und daher einfacher zu handhaben als Frequenzen. Auf der beiliegenden CD befinden sich unter "/Finales Instrument/Filme" einige Filme, die das fertige Instrument in Aktion zeigen. Darunter auch der Versuch, eine kleines Lied zu spielen.

8.3.1 Tonerzeuger

Um einen interessanten Klang erzeugen zu können programmierte ich ein Instrument, das als Eingangsvariable eine Frequenz annimmt und daraus einen Ton erzeugt. Dazu werden zwei Sinus-Oszillatoren und ein Sägezahnoszillator mit dem Objekt "min" kombiniert und dann durch einen Feedback-Delay-Network (FDN) Filter gespeist (siehe Abb. 8.4). Das min Objekt leitet immer den kleinsten Wert der Eintreffenden Signale weiter. Wenn nun die Oszillatoren um einen bestimmten Faktor zueinander verschoben werden, entstehen Obertöne und wenn der Faktor nicht ganzzahlig ist Schwebungen. Es fiel bei der Erstellung des Tonerzeugers auf, dass kleine Ungenauigkeiten dazu beitragen, einen Ton natürlicher wirken zu lassen. Bevor der Klang durch das Feedback-Delay-Network geht, wird noch ein Bandpass-Filter angewendet. Damit stehen folgende Audioparameter zur Verfügung: Grundfrequenz, Gesamtlautstärke, Verschiebung des zweiten Sinusoszillators, Lautstärken des zweiten Oszillators, Nachhallzeit des Feedback-Delay-Network. Es gibt noch zusätzliche Einstellungen, wie die Charakteristiken des FDN, die sich allerdings

Abb. 8.3: Harmonisches Gitter



Die X- und Y-Werte der linken Hand werden von ihrem ursprünglichen Wertebereich 0-127 auf die Breite (0-3) bzw. Höhe (0-4) übertragen. Dann wählt das [expr]-Objekt aus, welche Spalte angesprochen wird und [sel]-Objekt (select) wählt die Reihe aus. Die Zahlen in den Message-Boxen sind MIDI-Notenwerte. Sie werden an das [mtof]-Objekt gesendet, wenn sie einen Bang empfangen. Das [mtof]-Objekt wandelt diese dann in Frequenzwerte um.

nicht gut zur Laufzeit einstellen lassen.

8.3.2 Kontrollhand und Triggerhand

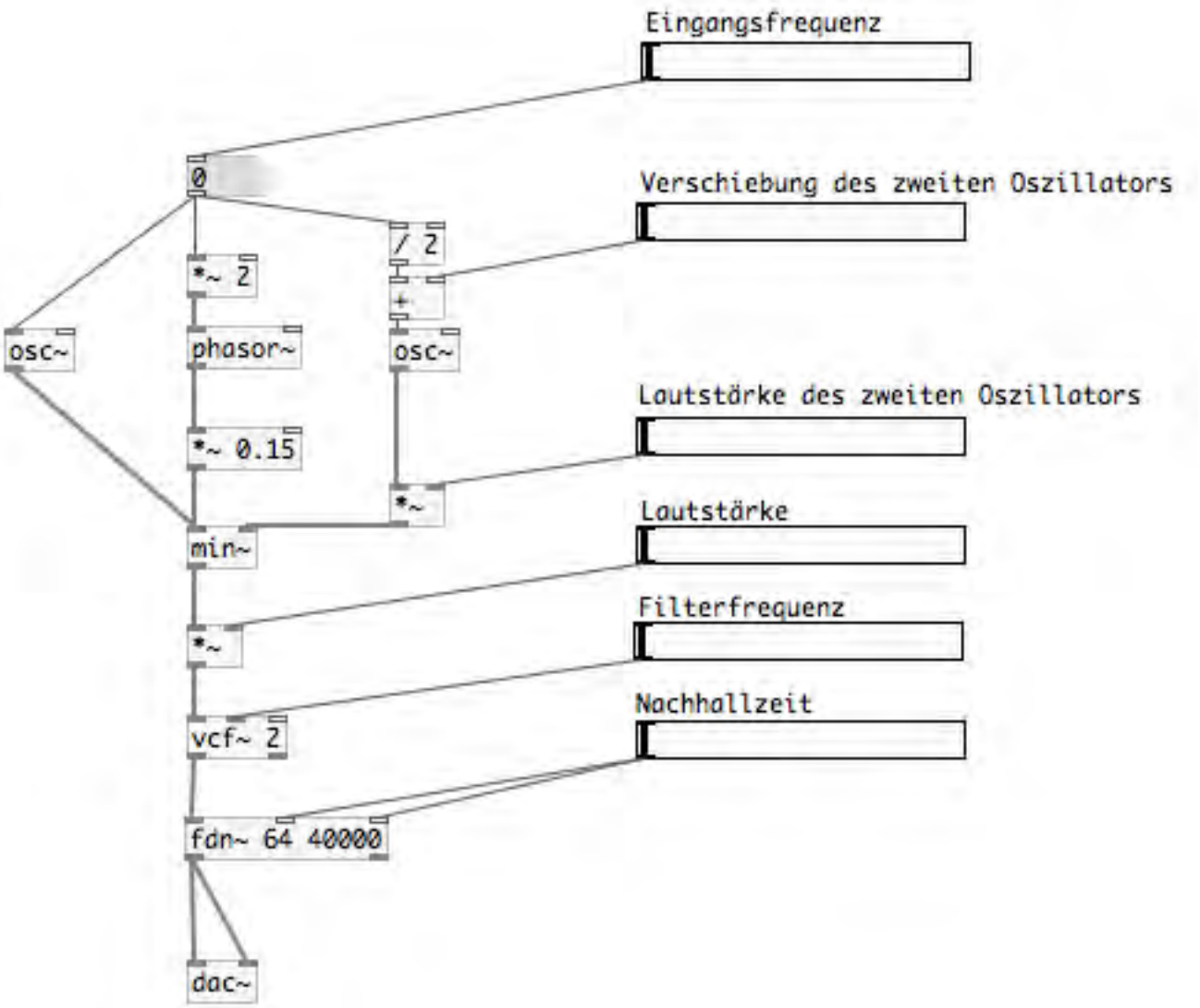
Im nächsten Schritt werden das Gitter und der Tonerzeuger miteinander kombiniert. Dazu muss lediglich das Frequenz-Outlet des Gitters an den Frequenzwert des Tonerzeugers angeschlossen werden. Um jetzt noch den Anschlag zu modellieren, wird die Geschwindigkeit der rechten Hand auf die Gesamtlautstärke gelegt. Die Idee für dieses Setup entstand aus der Analogie zu Saiten-Instrumenten. die linke Hand ist bei klassischen Saiteninstrumenten die Griffhand, die rechte ist für die Erzeugung des Tones zuständig. Sie hält beispielsweise den Bogen bei einer Geige, oder zupft die Saiten bei einer Gitarre. Da das Ziel dieser Arbeit jedoch nicht ist, ein klassisches Instrument zu simulieren, sondern eine neue Form der Interaktion mit Musik zu erschaffen, wurde dieses Konzept auf den gesamten Körper übertragen. Die linke Hand wird zur Kontrollhand und die Rechte zur Triggerhand. Die Kontrollhand greift einen Ton im Raum, während die Triggerhand dann den Klang erzeugt. Damit ist das Instrument Monophon. Weitere Charakteristika der Triggerhand können Klangparameter steuern. Es hat sich als interessant erwiesen, den Tiefen-Wert der Triggerhand auf die Frequenz des Bandpassfilters zu legen, so dass der Ton offener klingt, wenn die Hand näher an der Kamera ist. Die Nachhallzeit des Feedback-Delay-Networks wurde auf die absolute Distanz des Interakteurs zur Kinect gelegt, so dass der Ton mehr Hall bekommt, je weiter man von der Kamera weg ist.

8.3.3 Interaktionsebene

Bis zu diesem Punkt konnten Töne immer nur mit weichem Anschlag erzeugt werden, da die Lautstärke auf die Bewegungsgeschwindigkeit umgesetzt wurde. Da Handbewegungen immer eine Beschleunigung haben und nicht direkt auf einer hohen Geschwindigkeit starten, ist es sehr anstrengend, zu versuchen einen Ton hart zu beginnen oder zu beenden, da das bedeutet, dass man die Hände schnell beschleunigen und bremsen müsste. Um hier eine andere Möglichkeit zu geben, definiere ich eine Interaktionsebene. Auf der beiliegenden CD befindet sich unter "/Finales Instrument/Filme/interaktionsebene.m4v" ein Film, der die Funktion der Interaktionsebene verdeutlicht.

Im Interaktionsraum wird eine Ebene definiert, die den Raum teilt. Wenn der Interakteur in die Ebene eintaucht, wird der Ton getriggert, so wie bei einer Wasseroberfläche. Durch

Abb. 8.4: Der Tonerzeuger



Eine vereinfachte Form des Tonerzeugers. Zur besseren Verständlichkeit wurden an alle Variablen Regler gesetzt, mit denen die Parameter beeinflusst werden können. Der Patch ist auf der beiliegenden CD unter /Finales Instrument/tonerzeuger_simpel.pd zu finden.

das Abgreifen der Geschwindigkeit beim Eintritt in die Ebene wird eine physikalische Analogie zugrunde gelegt. Bei gleichbleibender Masse eines auftreffenden Objektes wird der Impuls nur durch die Geschwindigkeit bestimmt. Dieser Zusammenhang ist als Bewegung intuitiv umsetzbar. Wenn einem Interakteur das Bild von der Wasseroberfläche vermittelt wird, versteht er seine Interaktionsmöglichkeiten.

Die Interaktionsebene kann ohne großen mathematischen Aufwand drei verschiedene Grundausrichtungen haben: auf der X-Y Ebene, Auf der X-D Ebene, oder auf der Y-D Ebene. Dazu muss nur in der entsprechenden übrig gebliebenen Koordinatenachse ein Grenzwert gesetzt werden. Sobald dieser unterschritten wird, wird das Signal ausgelöst. Mit dem Setzen des Grenzwertes bekommt die Triggerebene einen genauen Ort. Die Aktivierungsrichtung sollte immer von der Körpermitte weggehen. Wenn die Interaktionsebene beispielsweise nach den X und Y Achsen ausgerichtet ist, und damit vor dem Nutzer, dann muss die Aktivierungsrichtung nach vorne gehen (die Triggerebene hinter den Nutzer zu legen würde keinen Sinn ergeben). Liegt die Ebene auf den Y und D Achsen, dann geht die Richtung zur rechten Seite. Die Ebene auf die linke Seite zu legen kommt nicht in Frage, da der Nutzer sonst zur Aktivierung die Körpermitte mit seiner Hand kreuzen müsste, was eine ziemlich unnatürliche Bewegung ist. Nur auf den X und D Achsen gibt es zwei Möglichkeiten, da die Ebene überhalb, oder (ein Stück) unterhalb der Schulter liegen könnte. Für mich hat sich die Anordnung vor dem Körper am besten angefühlt, doch verschiedene Nutzer bewerteten dies unterschiedlich (siehe 9.1 Test mit Nutzern).

Wenn die Ebene für die Trigger-Hand bei dem Instrument genutzt wird, entsteht beim Einbeziehen der Position auf der Ebene das Problem , dass es dann nicht mehr möglich ist durch kreisende Bewegungen einen gleichförmigen Ton zu erzeugen. Doch die Ebene an sich gibt dem Instrument neue interessante Möglichkeiten. So ist es mit der Ebene möglich, Töne mit einem kurzen Anschlag zu triggern, indem man die Hand mit hoher Geschwindigkeit in die Ebene eintaucht. Genau so abrupt lassen sich Töne beenden, indem man aus der Ebene austritt. Zusätzlich zu der Geschwindigkeit beim Eintauchen kann nach wie vor auch die Position der Triggerhand im Raum Klangparameter steuern. Durch die Triggerebene verliert aber ein Koordinatenparameter einiges von seiner Variabilität, da der gesamte Bereich, der ausserhalb des Grenzwertes liegt, nicht nutzbar ist. Daher funktioniert es beispielsweise nicht mehr so gut, die Tiefe der Triggerhand auf den Band-Pass-Filter zu legen, wenn die Interaktionsebene vor dem Nutzer nach den X und Y

Achsen ausgerichtet ist.

8.3.4 Sound Trigger

Die Space Trigger können dazu genutzt werden, um Töne im Raum zu positionieren. Wenn beim Triggern die Bewegungsgeschwindigkeit des Gliedmaßes ausgegeben wird, das den Ton auslöst, kann der Interakteur die Intensität des Tones bestimmen. Dies wurde mit dem Ablegen von Schlagzeugkomponenten auf entsprechende Trigger im Raum durch Ableton Live getestet. Natürlich soll hier kein Schlagzeug nachgebildet werden. Aber um die Grundidee zu testen und das Gefühl bei Benutzung der Trigger, sind Schlagzeugsounds eine gute Möglichkeit, da sie den meisten Nutzern bekannt sind, und daher die Erwartung klar ist. Wenn die Trommel getroffen wird, soll ein Sound entstehen, wenn der Schlag stärker geführt wird soll der Ton lauter sein. Die Geschwindigkeit der Hand bestimmt beim Eintreten in die Trigger-Zone die Stärke der getriggerten Schlagzeugteile. Dadurch entsteht eine gut nutzbare Anschlagsdynamik.

Um die Trigger zu positionieren, können Koordinaten der entsprechenden Gelenke benutzt werden. Das bedeutet, der Nutzer hält seine Hand an die Stelle, an die ein Trigger gesetzt werden soll, und klickt dann (mit der freien Hand) auf den entsprechenden Knopf. Damit ist es auch ohne visuelle Repräsentation gut möglich, sich die Trigger an die Stelle zu legen, an der man sie haben möchte. Die Sound-Trigger können aufgrund der Systemarchitektur sehr einfach mit dem Rest des Instrumentes verbunden werden, indem sie einfach in einem zweiten Patch-Fenster laufen.

9 Evaluation

9.1 Test mit Nutzern

Zum Abschluss des Projektes, wurden Tests mit 4 Nutzern durchgeführt, die das System erprobten und denen ich einige Fragen stellte. Auch innerhalb der Entwicklung habe ich das System immer wieder verschiedenen Menschen gezeigt, und mir deren Meinungen dazu aufgeschrieben. Dieser Test kann in seiner Ausführlichkeit und auch in seiner Aussage keine wissenschaftliche Nutzerbefragung sein, aber er reicht völlig, um eine der Arbeit angemessene grobe Einschätzung liefern zu können. Die Nutzer sind verschiedene Menschen aus meinem Umfeld mit unterschiedlichen Hintergründen. Alle befragten Nutzer sind zwischen 25 und 30 Jahre alt. Die Namen wurden geändert, die Hintergrundinformationen nicht. Janosch ist Toningenieur, sehr medienaffin, sehr bewandert auf dem Feld der Audioproduktion, spielt mehrere Instrumente. Sam ist Interfacedesigner, ebenfalls technik- und medienaffin, spielt E-Gitarre. Janine ist Sozialpädagogin, weder medien- noch technikaffin, spielt kein Instrument. Gerd ist Student der Sozialpädagogik, weder medien- noch technikaffin, spielt ein wenig Bass.

Grundsätzlich wurde der Test mit den Nutzern nach dem Prinzip des offenen Interviews durchgeführt. Es wurde ein grober Ablauf mit einigen Fragen strukturiert und die Fragen dann auch gestellt, aber auch versucht, offen zu bleiben für Anregungen und Meinungen, die die Nutzer frei aussprachen. Die Ergebnisse werden hier in Textform nach den einzelnen Fragen oder Themengebieten geordnet zusammengefasst.

Wie leicht fällt es dir, die Interaktion zu erlernen?

Das System wurde zunächst so eingestellt, dass es ohne Interaktionsebene funktioniert. Dann wurden alle Nutzer erst einmal ohne Erklärung vor das System gestellt, um herauszufinden, ob sie das System intuitiv begreifen können. Das hat bei keinem der

Nutzer funktioniert. Alle Nutzer haben gemerkt, dass Töne entstehen, und konnten auch verstehen, dass die Töne aus der Bewegung der Triggerhand entstehen. Allerdings begriff keiner den Zusammenhang zwischen rechter und linker Hand. Erst nach dem Erklären des Konzeptes Triggerhand und Kontrollhand konnten die Nutzer etwas mit dem System anfangen. Die Lerngeschwindigkeit war bei allen Nutzern ähnlich schnell. Nach ein paar Minuten verstanden sie die Grundlagen des Instrumentes. Nach einer Demonstration konnten die Nutzer schon selbst kreativ werden.

Wie findest du das Konzept Interaktionsebene?

Die Interaktionsebene wurde eingeführt und Nutzer wurden gefragt, wie ihnen das Konzept gefällt und wo sie diese am liebsten angeordnet hätten. Dann wurden verschiedene Einstellungen probiert. Alle Nutzer ausser Sam fanden es angenehm, wenn die Interaktionsebene von ihnen liegt, also parallel zu den X und Y Achsen. Sam fand es angehmer, wenn die Interaktionsebene nach unten (X-D Ebene) ausgerichtet ist, da so die Analogie zu einem Tasteninstrument gegeben ist. Alle Nutzer empfanden die Ebene an sich als sinnvolle Erweiterung, die dem System mehr Möglichkeiten gibt.

Wie beurteilst du das System bezüglich der Direktheit / Latenz?

Bei dieser Frage gingen die Antworten stark auseinander. Janosch bewertete das System als zu stark verzögernd, um damit in einer Livesituation Musik machen zu können, beziehungsweise sieht hier starke Einschränkungen, bezüglich der getriggerten Sounds. Sam fand das Instrument sehr direkt. Die anderen Nutzer empfanden das System als einigermaßen direkt. Daraus ist zu schließen, dass das System schon recht direkt ist, aber nicht professionellen Studio-Ansprüchen entspricht, wenn man damit Signale mit präzisem Timing erzeugen will. Dazu später mehr (siehe 9.3). Auch könnte die unterschiedliche Art zu spielen zu einer unterschiedlichen Einschätzung geführt haben. Wenn man einen Ton sanft erzeugen will, fällt die Latenz nicht sehr auf. Je mehr Möglichkeiten jedoch gegeben werden, einen harten Anschlag auszulösen, desto stärker machen sich selbst kleinste Verzögerungen bemerkbar.

Wie gut steuerbar ist für dich das Instrument?

Bei dieser Frage oder diesem Thema ging es darum, wie gut die Kontrollpunkte zu treffen sind, wie gut sich Töne oder Tonfolgen reproduzieren lassen und wie präzise das Instrument ist. Auch hier taten sich die Probanden unterschiedlich schwer. Man muss dazu sagen, dass das Instrument in unterschiedlichen Umgebungen aufgebaut wurde, da ich z.b. Janosch in seinem Studio besucht habe, und Sam bei sich zu Hause. Alle Nutzer taten sich anfangs schwer, konnten aber nach etwas Übung Melodien spielen und Effekte reproduzieren. Da das Instrument die Möglichkeit hat, den absoluten Abstand des Torsos auf die Nachhallzeit zu legen, konnten die Nutzer sich zur Kamera hin, oder von ihr weg bewegen. Dabei tauchten bei Janine Probleme auf, da sie sich schnell bewegte und dabei oft aus dem Bildbereich austrat. Die meisten Nutzer brauchten, wenn sie sich zusätzlich zu den Handbewegungen vor und zurück bewegten, das Ausgabebild von Synapse als Orientierung. Alles andere klappte auch ohne visuelle Repräsentation. Das zeigt, dass das Wahrnehmungsfeld der Kamera ein großes Problem ist für Nutzer, die sich nicht mit den Spezifikationen auskennen. Die Kinect-Kamera benötigt einen Mindestabstand von 60cm um ein Tiefenbild erzeugen zu können (siehe 2.4) und das Wahrnehmungsfeld wird zur Kamera hin kleiner. Das Austreten aus dem Kamerafeld machte gerade für die beiden nicht technikaffinen Nutzer einen Hauptteil der Fehlerquellen aus. Um diesem Problem zu begegnen, müsste man das System einfach an einem Ort aufbauen, an dem der Interaktionraum genau eingegrenzt wird (zum Beispiel mit Markierungen auf dem Boden.) und an dem genug Platz ist, um den Inteaktionsraum weit genug von der Kamera anfangen zu lassen. Wenn nur kleine Überschreitungen stattfinden wird das System davon nicht allzustark beeinflusst.

Wie gut gefällt dir der Sound?

Hier wurden die verschiedenen Filter-Settings eingestellt und gefragt, wie gut den Nutzern der Sound gefällt. Alle Nutzer empfanden den Sound als interessant und durch verschiedene Settings auch als abwechslungsreich. Sam hätte gerne noch eine Möglichkeit gehabt, die Settings selbst zu ändern, während er spielt. Das Mapping des Bandpassfilters auf die X-Achse kam auch nur bei ihm gut an, die anderen fanden das eher störend. Allgemein ist zu sagen, dass der Grundsound als interessant und angenehm empfunden wurde, wobei er nicht dafür reicht, als einzelnes Instrument lange gespielt zu werden, es

fehlt immer noch etwas, wie ein Rythmus, oder weitere Instrumente oder Tonspuren.

9.2 Weitere Anmerkungen und Fazit der Nutzerbefragung

Janine war das Instrument zu statisch. Sie hatte sich gerne viel bewegt, und hätte die Bewegungen dann auch im Sound wiedergefunden. Da sie die einzige der Testpersonen ist, die etwas aus sich herausgegangen ist, ist diese Information sehr wertvoll. Denn das bedeutet, dass der Controller für tatsächliche Tänzer, die viele große Bewegungen machen, noch einmal anders gestaltet werden müsste. Das Thema wurde schon in Kapitel 8.2 ansatzweise besprochen, doch hier zeigt sich noch deutlicher, dass ich das System eher vom Standpunkt eines Musikers entworfen habe, als von Standpunkt eines Tänzers aus. Auch das hat natürlich seine Vorteile. Doch es wäre definitiv reizvoll, ein solches System noch einmal in Kooperation mit einem/einer TänzerIn zu entwerfen.

Janosch hatte als Tontechniker die kritischsten Worte für das Instrument, aber sah dennoch Anwendungsgebiete, und könnte sich vorstellen, ein solches Setup in einer Live-Situation einzusetzen. Aus dem Gespräch ergaben sich viele technische Eingrenzungen für den Anwendungsbereich, aber auch neue Ideen. Er schlug die Verwendung für Ambient-Sounds vor und hatte die Idee, die Kinect zu benutzen um den Sound eines analogen Instrumentes in Echtzeit zu beeinflussen.

Allgemein fanden alle Tester das Instrument spannend und es hat ihnen Spaß gemacht, es zu benutzen. Doch es bleibt ein sehr spezielles Gerät, welches für einzelne Stücke in personalisierter Form verwendet werden könnte, wenn die Töne und Harmonien entsprechend für einen Song angepasst werden, welches aber wohl in seiner jetzigen Form keinen Einzug in die Musikerwelt finden wird. Um interessante unterschiedliche Effekte zu erzeugen muss man teilweise während der Benutzung an vielen Stellschrauben innerhalb des Systems drehen, was die Anwendung nur für Nutzer mit dem entsprechenden Hintergrundwissen spannend macht und da dürfte der Anwenderkreis überschaubar sein.

9.3 Latenz und Präzision

Die Framerate der Kinect beträgt 30fps. Damit ist die Dauer eines Frames ~33,3ms. Für die Auslösung eines Events beträgt die Ungenauigkeit also mindestens 33,3ms. Dazu

kommt dann noch die Verzögerung, die durch die Berechnung des Skeletts bei der Sceletondetection entsteht und die Latenz der Audioberechnung. In Audioprogrammen lässt sich die Latenz einstellen und hängt von der Leistung des Systems ab. Auf dem Testsystem konnte die Latenz in Pure-data auf 14ms heruntergeschraubt werden. Wurde die Latenz hier niedriger eingestellt entstand Knistern und Knacken. Geprüft werden konnte diese Latenz, indem ein Setup aufgebaut wurde, in dem ein Space Trigger Objekt (siehe8.3.4) an eine bestimmte Stelle im Raum gesetzt wurde. Dann wurde diesem Objekt die rechte Hand als Trigger zugeordnet. Die linke Hand wurde nun genau an die Stelle im Raum gehalten, an der sich das Objekt befindet, und mit der rechten darauf geklatscht. Das ganze wurde mit einem externen unabhängigen Aufnahmesystem mit 44100hz Samplingrate aufgenommen. In der Tonspur konnten dann die Ausschläge gesehen, und der Abstand zwischen ihnen gemessen werden. Es wurden 10 Wiederholungen gemacht, um den ungefähren Bereich bestimmen zu können, in dem sich die Latenz bewegt. (siehe Tab.9.1) Interessant ist, dass die Abweichung von Wert 6 zu Wert 8 größer ist, als die angenommenen 33,3ms und zwar 36ms. Das muss daher kommen, dass die Geschwindigkeit der Berechnung der Sceleton-Detection wohl leichten Schwankungen unterliegt. Die Latenzeinstellungen von Audioprogrammen sind in der Regel verlässlich.

Einige Studiomusiker können angeblich Latenzzeiten ab 5ms wahrnehmen. Ich hatte bei dem Nutzer-Test in einem Tonstudio, in dem die genaue Latenz mit allen Beeinflussungen der Hardware bekannt war, die Möglichkeit zu testen, ab wann ich die Latenz wahrnehme, die entsteht, wenn man auf einem Midi-Keyboard spielt. Ab 14ms war die Latenz für mich spürbar. Doch da man bei dem Motioncontroller keine haptische Entsprechung hat, wird die Latenz hier nicht ganz so schnell als störend empfunden.

Es zeigt sich bei der Messung, dass der Nutzer “Janosch”9.1 mit seiner Einschätzung recht behalten hat. Die Latenz ist für professionelle Studio-Anwendung zu hoch, und beschränkt das Instrument definitiv auf das Triggern von Sounds, bei denen ein präziser Anschlag nicht wichtig ist. Gerade für Perkussive Klänge ist die Latenz zu hoch und die Präzision nicht groß genug.

9.4 Ausblick

Mit ein paar Anpassungen könnte das System auch als Ausstellungsstück genutzt werden. Man müsste dazu lediglich beide Hände und vielleicht auch die Füße wie die Triggerhand

Tab. 9.1: Latenz

Versuch	1	2	3	4	5	6	7	8	9	10
Latenz	55	27	53	26	45	20	21	56	43	37
Mittelwert	38,3									

benutzen und die Tonfolgen entsprechend automatisieren, oder Effekte dahinter legen statt melodischen Bestandteilen. Die Nutzung des absoluten Abstandes zur Kamera kann erhalten bleiben. Dazu wäre es zusätzlich wichtig, Die Sceleton-Detection zu automatisieren, also die Kalibrierungspose wegzulassen, damit der Nutzer direkt getrackt wird, wenn er den Bildbereich betritt. Dies wurde in meinem Projekt nicht näher untersucht, da ich mich entschied, mich primär mit der Benutzung als Instrument auseinanderzusetzen. Dafür ist die direkte Interaktion nicht so wichtig, wie für ein Ausstellungsstück. Da Synapse auf dem OpenNI-Framework basiert, und es in diesem möglich ist, die Kalibrierung zu umgehen, dürfte das nicht weiter schwierig sein. Eine weitere wichtige Überlegung für die Verwendung als Ausstellungsstück ist der Umgang mit mehreren Nutzern. Auch als Instrument könnte der Aufbau noch um Interaktionsmöglichkeiten erweitert werden, die es zum Beispiel ermöglichen, den Ton umzuschalten. Es gibt keine Begrenzungen für das entwickeln neuer Audioerzeuger und das Mapping der Bewegungsparameter auf deren Audioparameter. Es gibt also in verschiedene Richtungen, in die man das Projekt weiterentwickeln könnte noch viele Möglichkeiten.

10 Fazit

Es konnte ein System entwickelt werden, welches als eigenständiges Musikinstrument funktioniert. Es basiert vollständig auf quelloffenen Frameworks, weist aber auch Schnittstellen zu kommerziellen Systemen auf. Beispielsweise ist es mit den Triggerpunkten möglich über MIDI jedes System zu steuern, das MIDI-Noten verarbeiten kann (und das kann fast jedes System in der Audioentwicklung). Bei der Entwicklung mit Pure Data lag das Augenmerk darauf, möglichst übersichtlich und einfach zu programmieren, so dass jeder die entstandenen Patches schnell begreifen und verändern kann. Die Entwicklung hat gezeigt, dass es allein mit Open-Source Software möglich ist, einen durchaus hochwertigen Audiocontroller zu bauen, und auch dazu passende Tonerzeuger. Es gab schon viele Ansätze, musikalische Parameter mit Bewegung zu steuern, oder Samples abzuspielen. Doch das erzeugte System ist das erste, das es schafft, Töne wirklich aus der Bewegung zu erzeugen. Durch das Triggerhand-Prinzip entsteht ein ähnliches Spielgefühl wie bei analogen Instrumenten. Es haben sich auch Schwächen gezeigt, vor allem das Problem der Latenz, welches aufgrund der technischen Begrenzung von 30 Frames pro Sekunde auf Videoseite und 44100 Samples auf Audioseite einfach nicht vollständig lösbar ist. Doch im Rahmen der angestrebten Bedingungen, so zum Beispiel, dass das System möglichst billig sein sollte und nach Möglichkeit auf Open-Source Software basieren sollte, wurde das bestmögliche Ergebnis erzielt und die in der Voruntersuchung gestellten Anforderungen wurden erfüllt.

Die Arbeit mit Pure Data war anfangs ungewohnt, war jedoch nach dem Testen verschiedener Audioumgebungen der beste Einstieg in die Erzeugung elektronischer Klänge. Vor der Arbeit hatte ich keinerlei Vorkenntnisse über die generative Erzeugung von Audiosignalen. Pure Data ist mit seiner grafischen Darstellung der Signalwege jedem zu empfehlen, der sich für das Thema interessiert. Allgemein ist elektronische Musik ein so weites Feld, dass es mir schwer fiel, einen Einstieg zu bekommen, da bei der Recherche immer weitere Themen und Möglichkeiten auftauchten. Doch durch diese Vielfalt stehen

auch unglaublich viele Möglichkeite zur Verfügung, mit nichts als einem Rechner Musik zu komponieren. Die Qualität der erzeugten Töne und Effekte lässt sich nicht von teuren komerziellen Programmen unterscheiden, einzig die Benutzeroberflächen sind bei den Open-Source Varianten schwieriger zu handhaben.

Das erstellte Instrument kann in einer Live-Situation Anwendung finden, zusammen mit einer Band, oder im Kontext mit anderen elektronischen Audioerzeugern. Aufgrund der Latenz kann es nicht für sehr schnelle Tonwechsel genutzt werden, aber für Bassläufe oder Ambient-Sounds und Effekte funktioniert es gut. Ein System dieser Art kann auch in der Bewegungstherapie genutzt werden, um Menschen spielerisch den Zugang zu Musik ermöglichen. Auch jetzt schon werden in Kliniken zu Beweegungsförderung Kinect Systeme eingesetzt, zusammen mit der Xbox und den dazugehörigen Spielen. Das erstellte System wäre für Blinde eine Möglichkeit Bewegungen in der Audioebene zu erfahren, statt in der visuellen.

Auf die Frage: Ist es möglich, mit billigen und leicht zugänglichen Mitteln Bewegung in Musik umzuwandeln, kann ein klares “Ja” gegeben werden. Und nicht nur das, es ist sogar jedem, der etwas Zeit investiert, möglich, diese Umwandlung nach seinen Ideen und Vorstellungen zu gestalten.

Teil III

Anhang:

11 Inhalt der CD

11.1 Finales Instrument

In diesem Ordner sind alle von mir erstellten Pure-Data Patches, die nötig sind, um das Instrument zu starten. Das Kernelement bildet Synapse_Controller_v1.pd. hier wird nur grob die Funktion erklärt. Weitere Komentare und Erklärungen sind direkt in den Patches zu finden.

11.1.1 Synapse_Controller_v1.pd

Dies ist die Schnittstelle zwischen Synapse und Pure-Data. Dieser Patch stellt für alle weiteren Pure Data Anwendungen die Kontrolldaten zur Verfügung. Um damit zu arbeiten, muss Synapse gestartet werden (siehe Synapse). Wenn Synapse ein Skelett erkannt hat, kann man im Synapse_Controller Patch einen Haken bei synapse_power setzen. Ab da werden die abgeleiteten Kontrolldaten gesendet. Um den Patch näher zu betrachten einfach auf [pd guts] klicken. Dann öffnen sich die Sub-Patches. Die Funktionsweise wird dort in Kommentaren erklärt.

11.1.2 instrument.pd

Dann kann instrument.pd gestartet werden. Wenn der Haken bei on/off gesetzt ist, und der Volume Regler etwas hoch gestellt wurde, sollte das Instrument funktionieren. Mit trigger_treshold lässt sich die Position der Interaktionsebene einstellen. Um den Patch näher zu betrachten, Linksklick irgendwohin und dann auf öffnen. Durch das anlegen eines Interfaces werden Patches in Pure-Data schnell unübersichtlich, aber die Grundfunktion ist in Kapitel 8.3 erklärt.

11.1.3 Filme

Hier sind vier Filme gespeichert. Der Film mit dem Titel “Aufbau und Spiel.m4v” zeigt Schritt für Schritt, wie das System in Betrieb genommen und eingestellt wird, und wie damit Töne erzeugt werden. In dem Film mit dem Titel “differentsounds.m4v” werden verschiedenen klanglichen Möglichkeiten vorgestellt. “Interaktionsebene.m4v” zeigt, wie die Interaktionsebene funktioniert, indem einmal eine Bewegung innerhalb des Interaktionsraumes gezeigt wird, bei der ein Ton entsteht und einmal eine Bewegung ausserhalb, bei der kein Ton entsteht.”kleiner Song.m4v” soll die Möglichkeiten kurz demonstrieren, das System tatsächlich musikalisch sinnvoll einzusetzen.

11.2 Synapse

Bei Mac OS X ist keine Installation nötig, man kann Synapse.app einfach starten. Bei Windows ist es etwas komplizierter. Die Anleitung dazu findet man auf http://synapsekinect.tumblr.com/ Beim Start ist wichtig, dass die Kinect Kamera richtig angeschlossen ist, ansonsten wird Synapse mit einem Fehler beendet. Wenn Synapse läuft, muss die Kalibrierungspose eingenommen werden, bis Synapse das Skelett erkannt hat. Die Pose besteht daraus, beide Arme gerade zur Seite vom Körper weg zu strecken, und dann die Unterarme rechtwinklig nach oben. Dass die Pose erkannt wurde sieht man daran, dass das Skelett dann in rot eingezeichnet wird.

11.3 Prototypen

Hier sind die Prototypen untergebracht. Unter Prototypen/Filme ist von jedem Prototypen jeweils ein Film von aussen und der dazugehörige Screenmovie.

11.3.1 Der Helligkeitsprototyp

ist unter /Processing zu finden. Damit er funktioniert, muss Supercollider gestartet werden, die Synthdefs unter /Supercollider/SC Synthdefs.rtf in den SC-Server geladen werden und dann das Processing Programm gestartet werden.

11.3.2 Der Gitterbasierte Prototyp

ist unter /Processing damit dieses funktioniert, mussldiglich die Kinect Kamera angeschlossen sein (wurde nur unter Mac OS X getestet).

11.3.3 Synapse + PD + Kontakt

Für diesen Prototypen muss Kontakt installiert sein (oder der Kontakt-Player). Zunächst Synapse und den Patch Prototypen/Synapse_PD_Kontakt/Synapse_to_midi.pd starten. die MIDI-Channels und Treshold im Patch einstellen. Die in Kontakt genutzten MIDI-Channels sind 21, 22 und 23. Danach bellmover.nki mit Kontakt-Player starten. Im Kontakt Player muss eine Note gedrückt werden, damit es funktioniert.

11.3.4 PSMove + Osculator + Kontakt

Das Programm befindet sich unter Prototypen/PSMove/cindertest. Der Ordner sollte sich in XCode öffnen lassen. Dazu muss die Cinder-Library installiert sein. Zuerst das Programm starten, dann /OSCulator/MoveToMidi.oscd mit OSCulator öffnen, bellmover.nki mit Kontakt-Player starten.

Literaturverzeichnis

[1] Y. Iwadate, M. Inoue, R. S. N. Hikawa, M. Makino, and Y. Kanemoto, "Mic interactive dance system - an emotional interaction system," *Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies*, pp. 1–4, nov 2000.

[2] http://www.megaproject.org/, zuletzt geprüft 7.12.2011, Antonio Camurri 2000.

[3] http://kenmooredesign.blogspot.com/2011/01/kinect-theremin-sneak-peek.html zuletzt geprüft 6.3.2012, Ken Moore.

[4] http://www.kinecthacks.com/kinect-controlled-granular-scratching/, zuletzt geprüft 7.12.2011, Chris Vik 2011.

[5] http://www.kinecthacks.com/kinect-music-multimultitouchtouch/ zuletzt geprüft 6.3.2012, Tim Thompson.

[6] http://www.otherbirds.com/15-11.html zuletzt geprüft 6.3.2012, Sam Tarakajian.

[7] M.-J. Yoo, J.-W. Beak, and I.-K. Lee, eds., *Creating Musicl Expression using Kinect*, International Conference on New Interfaces for Musical Expression, Yonsei University, South Korea, 2011.

[8] Bernhard Szajner, Patent: FR2502823 (A1) CONTROLE DE SYNTHETISEUR MUSICAL PAR LASER Datum der Erstellung : 1981-03-27.

[9] D. Rokeby, "The construction of experience: Interface as content," *Digital Illusion: Entertaining the Future with High Technology*, 1998.

[10] T. Winkler, ed., *Creating Interactive Dance with the Very Nervous System*, Connecticut College Symposium on Arts and Technology, Brown University, 1997.

[11] A. Friberg, ed., *A fuzzy analyzer of emotional expression in music performance and body motion*, Music and Music Science, KTH - School of Computer Science and Communication, Stockholm, 2004.

[12] S. Dahl, “On the beat: Human movement and timing in the production and perception of music,” Master’s thesis, KTH School of Computer Science and Communication, 2005.

[13] http://www.lichtfaktor-live.com zuletzt geprüft 15.12.2011, LichtfaktorGmbH.

[14] http://www.ifixit.com/Teardown/PlayStation-Move-Teardown/3594/ zuletzt geprüft 12.12.2011.

[15] http://www.processing.org, zuletzt geprüft 7.12.2011, Ben Fry and Casey Reas 2001.

[16] http://libcinder.org, zuletzt geprüft 7.12.2011, The Barbarian Group.

[17] http://archive.cnmat.berkeley.edu/OpenSoundControl/, zuletzt geprüft 7.12.2011, Matt Wright 2004.

[18] http://puredata.info/ zuletzt geprüft 7.12.2011.

[19] http://www.cycling74.com/products/max/, zuletzt geprüft 13.12.2011, Cycling ’74.

[20] http://synapsekinect.tumblr.com/, zuletzt geprüft 7.12.2011, Chris Kalani 2011.

[21] http://supercollider.sourceforge.net/, zuletzt geprüft 7.12.2011.

[22] C. Roads, *The Computer Music Tutorial*. MIT Press, 1996.