HOCHSCHULE ANSBACH

multimedia und kommunikation

BACHELORARBEIT

Audiovisuelle Gestaltung und Implementierung einer gestengesteuerten interaktiven medialen Installation

Betreuung
Prof. Dr. Cornelius Pöpel

Zweitgutachten
Prof. Robert Rose

Autor
Julian Vogels
Hoffmann-von-Fallersleben-Straße 5
99423 Weimar

Matrikelnummer
44634

Ort, Abgabetermin
Ansbach, 22.06.2012

Hochschule für angewandte Wissenschaften
University of Applied Sciences

I Kurzfassung und Abstract

Weiß, unschuldig, harmlos steht es in dem neutralen Raum. Nichts offenbart seine wahre Natur - seine Boshaftigkeit - bevor man das Podest betritt. Die Intrige des Maskenrads ist spürbar real.

Interaktive Medieninstallationen sind immer häufiger in Austellungen und auch im öffentlichen Raum anzutreffen. Interaktive Kunst fasziniert und ermöglicht ein Eintauchen in die Kunst, wie es in dieser Form nicht besser umsetzbar ist. In Zeiten von „immersive Apps" und interaktiven Schaufenstern ist diese Kunstform auch zu einem interessanten Berufsfeld geworden.
Die vorliegende Arbeit behandelt den gestalterischen und technischen Aspekt der interaktiven Installation „Maskenrad". Es wird einerseits die Implementierung aller wichtigen Funktionen und der zugehörige Aufbau der Elektronik und Sensorik erklärt und andererseits die konzeptuelle Arbeit und die Gestaltung der Klanglandschaften und der Lichtsituationen dargelegt.
Durch die evolutionären Herangehensweise bei der Beschreibeung der Arbeitsschritte kann der Leser die Projektarbeit Schritt für Schritt nachvollziehen und Abläufe sowie aufgetretene Probleme und deren Lösung verstehen.

White, innocent, harmless it stands in the neutral room. Nothing reveals its true nature - its evilness - before you step on the podium. Maskenrad's intrigue is drastically real.

Interactive media installations are shown increasingly often in exhibitions and in public space. Interactive art fascinates people and makes it possible to dive into the art in a manner that is not producable with other means. In times of immersive apps and interactive shop windows this form of art has become an interesting industrial field, too.
The work in hand concerns the design and technical aspects of the interactive installation „Maskenrad". On the one hand the implementation of every important function and the appendant configuration of electronics and sensors is explained and on the other hand the conceptual work and the design of the sound scapes and light situations is demonstrated.
Due to the evolutionary approach at the description of work steps, the reader can follow this project work step by step and comprehend the development as well as the occured issues.

This paper is written in german. A certified translation is available upon request.

II Vorwort

An dieser Stelle möchte der Autor seinen Dank an Dr. Prof. Cornelius Pöpel ausdrücken, der diese Arbeit mit gutem Rat und Elan betreut hat. Er trug durch konstruktive Hilfestellung zu einer äußerst positiven Motivation bei der Erstellung dieser Bachelorarbeit bei.
Ein besonderer Dank gilt Jonas Bansemer, der dem Autor viel Wissen um die Elektronik vermitteln konnte und einen Teil zum Circuit Design beigetragen hat.
Während des gesamten Projekts konnte ich auf die Unterstützung von M^{lle} Cindy Beuhlah zählen, die zum Beispiel durch die Erstellung die Ausstellungsbroschüre zum Erfolg des Projektes beitrug.
Auch Andreas Muxel und Martin Nawrath konnten mir mit ihren Anregungen und Vorschlägen bei Herausforderungen der Elektronik und der Implementierung in Max/MSP weiterhelfen.
Maßgeblich trug Sarah Lübke zur Richtigkeit und Vollständigkeit, insbesondere zur Steigerung der sprachlichen Qualität der Inhalte bei.
Der Autor dankt dem Leiter des Studiengangs Prof. Dr.-Ing. Helmut Roderus und allen involvierten Personen an der Hochschule Ansbach, die es ermöglichten, ein reichhaltiges pluridiziplinäres Wissen im Zuge des Studiums Multimedia und Kommunikation erlangen zu können.

III Inhalt

IV Abbildungs- und Tabellenverzeichnis

V Abkürzungsverzeichnis

BPM	Beats Per Minute Taktschläge pro Minute
DAW	Digital Audio Workstation Software zum Bearbeiten von Audio
HSL	Hue Saturation Luminosity Farbbeschreibungsmodell durch Farbton, Farbsättingung und Helligkeit
LFO	Low Frequency Oscillator
MSP	Max Signal Processing, Signalverarbeitungsversion des Programmes Max
One Shot	Eine kurze Audiodatei, die als ein Effekt abgespielt wird
OSC	Open Sound Control Protokoll zur Übertragung von Daten
RGB	Additiver Farbraum aus den Farben Rot, Grün und Blau
VST	Virtual Studio Technology Schnittstelle für Audio im Softwarebereich

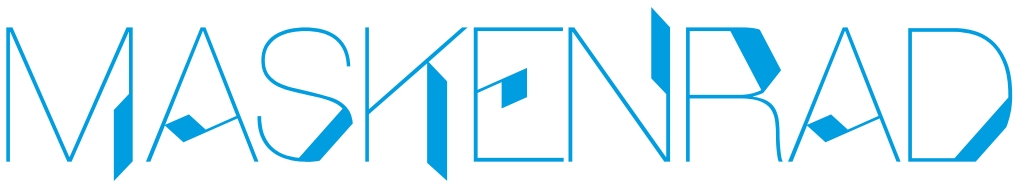
1 Einführung

Die vorliegende Arbeit behandelt das Konzept, die gestalterischen Aspekte in Grafik und Ton und die technische Umsetzung in Programmierung und Elektronik einer interaktiven Installation.
Es handelt sich um die konzeptionelle Idee, eine maschinelle Falle zu konstruieren. Ihr Ziel ist es, beim Akteur Emotionen auszulösen, die jenen Emotionen nahe kommen, die Intrigenopfer in einem möglichen Verlauf einer Intrige empfinden können. Die Umsetzung dieser Idee in einer interaktiven Installation ist Hauptbestandteil und Problemstellung dieser Arbeit. Es werden audiovisuelle Lösungen und Entscheidungen ebenso wie technische Vorgehensweisen erläutert. Einzelheiten zur Konstruktion und die Recherchearbeit vor Beginn des Designs der jetzt verwendeten Elektronik können hier leider nicht weiter dargelegt werden, da sie nicht Bestandteil der offiziellen Bachelorarbeit sind.
Da der Autor als Studienschwerpunkt die Bereiche Audio und Programmierung wählte, sind diesem Kontext entsprechend die Schwerpunkte der Arbeit zu sehen.

Die Erkenntnisse, die beim Lesen dieser Arbeit erlangt werden, dienen künftigen Medienkünstlern als Grundlage und Hilfestellung für die technische Planung und Umsetzung von interaktiven Medieninstallationen. Als eine Art Machbarkeitsstudie zeichnen sie mögliche technische Herangehensweisen aus. Auch als Inspirationsquelle können sie einen großen Beitrag für die Ausweitung von Features einer interaktiven Installation leisten.

Im Folgenden wird zuerst auf das Konzept eingegangen, das dem Leser die Idee hinter der Installation nahe zu bringen versucht. Darauf folgt ein Überblick der Konstruktion und Details zur verbauten Elektronik, um sich ein Bild von der Funktionsweise des Objektes machen zu können. Überlegungen zum Sound Design schließen sich an. Schlussendlich wird explizit auf die Implementierung u.a. in Max/MSP eingegangen.

Der Name der Installation lautet „Maskenrad" und wird im Folgenden unter dieser Bezeichnung angeführt.



2 Konzept

Oft haben sensationelle Meldungen in der Tagespresse etwas mit Lügen, Verschweigen und hinterhältigem Handeln zu tun. Menschen sind zu sehr ausgefeilten Machenschaften fähig und sie lieben es, darüber zu erfahren und so im Kreis der Wissenden zu sein. Intrigen faszinieren die Menschen seit jeher.

2.1 Problemstellung

Den Autor dieser Arbeit interessiert der menschliche Wesenszug der Intrige, den er selbst schon oft beobachten konnte und zu spüren bekam. Von 2007 bis 2008 befand er sich in Togo / Westafrika, wo er Opfer von Rassismus und sogar von Vodun-Zauber wurde, woraufhin ihn einige gläubige Togoer mieden. Nicht jeder schätzt dort die Anwesenheit eines Mitteleuropäers. Auch auf politischen Koferenzen der Jugendbewegung der SPD, den Jusos, konnte er beobachten, wie Menschen schon in jungen Jahren versuchen, Delegationen gegeneinander Auszuspielen und zu ihrem eigenen Vorteil anstatt zum Wohl des Ganzen handeln.
Diese Inspirationen formen das inhaltliche Konzept der Installation. Es geht darum, Menschen zur Selbstreflexion anzuregen und eine Senibilität für dieses Thema zu schaffen, von dem immer noch eher hinter vorgehaltener Hand geredet wird, um nicht selbst als intrigant zu gelten (MICHALIK, 2011, S.7).

Intrigen und Fallen sind ein uraltes menschliches Konzept, um Anderen zu schaden und zum eigenen Vorteil zu handeln. Theoretisch definiert ist die Intrige nach Michalik mit fünf Merkmalen: Sie muss hinterhältig (1) und geplant (2) sein. Sie bedarf eines Motives (3) und muss folgerichtig (4) durchgeführt werden. Es müssen stets mindestens drei Akteure(5) beteiligt sein (MICHALIK, 2011, S. 36ff).
In wie weit kann dieses Konzept auf eine Installation übertragen werden?
Ein Hinterhalt wird durch Vortäuschen falscher Tatsachen oder durch das Verschweigen von Tatsachen erreicht. Im Falle des Maskenrads ist es die Verkleidung in Licht und Ton, die die wahre Natur der Installation verbirgt. Planvoll zu handeln liegt wiederum bereits in der Natur einer programmierten Maschine. Ein Motiv ist allerdings nicht zu erkennen. Es nützt der Maschine nichts, eine Intrige auszuführen. Sie ist nur der loyale Vollstrecker des Willens des Programmiers, dem eigentlichen Intriganten. Seine Intention ist die bloße Selbstreflektion des Intrigenopfers. Er möchte zum denken anregen und die Ausstellungsbesucher für dieses Thema sensibilisieren und faszinieren.
Dem Intrigenopfer ist diese Installation gewidmet. Es geht nicht um Sinn oder

Unsinn einer Intrige, sondern um die Gefühle und Perzeption eines Opfers. Diese darzustellen ist das Ziel.
Grundlegende Emotionen während des Lebenszyklus der Installation sind das Unbehagen und eine Ahnung von dem, was passieren wird, das Realisieren, dass etwas nicht stimmt, der Schmerz und die Scham, die dem Opfer durch eine Intrige zugefügt wird, die Isolation, der das Opfer einer Verschwörung ausgesetzt ist, der Versuch des Verstehens einer Intrige, der bloße Hass der beiden kontrahierenden Seiten aufeinander.

2.1.1 Phasen

Diese eben genannten Emotionen eines Intrigenopfers in einem möglichen Verlauf einer Intrige lassen sich kategorisieren. Diese Kategorien werden im zeitlichen Verlauf der Installation in Phasen umgesetzt.
Die Installation ist in fünf Phasen aufgeteilt.

Tab. 1: Phasen des Lebenszyklus der Installation

Phase00	**Attraktion** *Zustand des Systemstarts. Die Maschine lockt den ahnungslosen Ausstellungsbesucher an.*
Licht	• Mehrfarbig, Farbe wechselt langsam durch das gesamte Spektrum
Bewegung	• Sanfte, geometrisch generierte Wellenbewegung der Servomotoren
Klang	• Harmonische Bb-Durmelodie, Impulsfilterung • Digitaler Synthesizer-Klang zwischen Mundharmonika und Akkordeon, zufälliger pentatonischer Akkord mit zufälligem Notenwert, der abhängig von den stochastischen Daten ausgelöst wird
Phase01	**Kalibrierung und Unbehagen** *Tritt der Besucher auf die Platte, verändert sich die Stimmung schlagartig. Jetzt muss die Kinect kalibriert werden. Er wird angewiesen, die Hände zu heben.*
Licht	• Mehrfarbig wie in Phase00, Verdunklung bis 4% Intensität

Bewegung	• Wellenbewegung stoppt, Servomotoren richten sich auf 140 Grad aus, sodass sie auf den Besucher zeigen
Klang	• Harmonische Bb-Mollmelodie, Impulsfilterung • Grollende, unterschwellige Disharmonien aus tiefen Sinusschwingungen werden hinzugemischt
Video	• Simulation eines neongrünen Scans wird eingeblendet: Eine gefilmte Person zeigt, wie man die Kamera kalibriert
Phase02	**Intrige und Konfrontation** *Die Falle schnappt in einem Crescendo störender Klänge zu. Interaktion, Sprache Licht und Klang verwirren und verstören.*
Licht	• Zufallsgesteuert in bestimmten Intervallen • Farbe grün, später rot
Bewegung	• schneller werdende konstante Wellenbewegung
Klang	• subtraktive Synthese durch Filtern von Rauschen und spektralreichen Signalen • Crescendo • Abspielen von vorgefertigten Klangsequenzen (Sprache, aufgenommene Geräusche)
Interaktion	• Gesten von Händen, Kopf und Knien führen zu einer Manipulation von Klang und Licht
Phase03	**Isolation und Verstehen** *Der Protagonist wird sich seiner Lage bewusst. Isolation und Angst machen sich breit. Er muss versuchen, die Falle zu verstehen, um aus ihr zu entkommen.*
Licht	• Zunächst Dunkelheit, Lichtintensität wird durch Gesten gesteuert • Farbe blau
Bewegung	• Bewegung der Servomotoren je nach Armstellung: • Winkel richtet sich nach Y-Position und wird relativ zur X-Position mit einem Wert zwischen eins und null multipliziert

Klang	• Softwareinstrument Razor und Massive, Tracking-Daten steuern die Klangparameter und somit das Timbre • Abspielen von „One Shots“
Interaktion	• Alternierende Aufwärtsbewegung der Hände erhöht die Lichtintensität und führt schließlich zum Start von Phase04 • Y- und Z-Position steuert Servobewegung
Phase04	**Wut und Hass** *Der Besucher wird mit seiner eigenen Reaktion konfrontiert.* *Der Zorn beider gegenüberstehender Seiten wird klar.*
Licht	• Aggressiv rhythmisch pulsierend • Farbe rot • Bei Servobewegung Aufblitzen
Bewegung	• Servos fahren rhythmisch einen zufälligen Winkel an
Klang	• Softwareinstrumente Razor als Synthesizer und Absynth als Effekt (Fade nach einer bestimmten Zeit)
Video	Drei Quellen für Videobilder: • Snapshots, die während des Lebenszyklus zu bestimmten Events gemacht wurden • Snapshots aus derselben Perspektive, die Personen zeigen, die um das Podium herum stehen, ohne Besucher • Live-Kamerabilder
Interaktion	• Schnelle Bewegung von Hand, Fuß und Kopf löst Servobewegung aus

2.1.2 Namensfindung

„Maskenrad“ ist ein Neologismus, erreicht durch die Veränderung des Wortes Maskerade. Das Maskieren oder Verkleiden ist eine der einfachsten und ältesten Formen der Intrige, die wir alle aus Märchen wie Schneewittchen kennen.

Das Kaschieren des ursprünglichen Begriffs repräsentiert die Natur der Installation. Auch entsteht das Hauptwort „Rad“, das als ein nicht perfekter Kreis nicht nur die Form des Objekts, sondern auch die hauptsächliche Bewegungsrichtung des pulsie-

renden Lichts, das sich über die 16 Lichtquellen verteilt aufgreift. Auch das Schwenken der Servomotoren ist eine Kreisbewegung.

2.2 Gestensteuerung

Die Steuerung von Algorithmen über das Tracking, also das Verfolgen von Bewegungen über das Bild einer Kamera, ist eine relativ neue und sehr interessante Entwicklung auf dem Gebiet der Mensch-Maschine-Interaktion. Technisch gesehen liefert diese Methode eine Vielzahl von Sensorendaten, die genau wie zum Beispiel die Daten eines Drehreglers in der Software auf Parameter gemappt werden können. Der gravierende Unterschied jedoch ist, dass der Benutzer keinerlei physische Interaktionsobjekte benötigt, um mit dem System interagieren zu können. Darin liegt der Reiz der Gestenerkennung für die Verwendung in dieser Installation. Die Nutzererfahrung ist tiefgehender, da er sich nach der erfolgreichen Kalibrierung der Kamera nur durch das Verlassen des Bildbereiches dem Tracking entziehen kann. Folglich kann der Besucher nichts gegen die Reaktion der Maschine tun, denn jede Bewegung führt automatisch, und nicht nur durch die bewusste Benutzung eines Drehreglers, zu einer Interaktion.

Tab. 2: Gestensteuerung

<table>
<tr><th></th><th>X</th><th>Y</th><th>Z</th></tr>
<tr><td rowspan="2">Phase02</td><td>Grundlegendes Signal der Synthese ändert sich je nach Position der Hände von Rauschen zu Pulswelle</td><td>Lichtintensität und Amplitude des Signals sind abhängig von der Position der Hände</td><td></td></tr>
<tr><td colspan="3">Schnelle Bewegungen lösen Klangereignisse aus („One Shots“)</td></tr>
<tr><td>Phase03</td><td></td><td>Abhängig von der Z-Position wirkt die Handhöhe unterschiedlich stark auf die Winkel der Servomotoren
Hände alternierend nach oben: Lichtintensität steigt, Klangauslösung</td><td>Servomotoren richten sich wellenförmig entsprechend der Position der Hände aus</td></tr>
<tr><td>Phase04</td><td colspan="3">Schnelle Bewegungen ändern den Videoinput, lösen eine neue Note im Synthesizer aus und stellen die Winkel der Servomotoren zufallsbedingt neu ein</td></tr>
</table>

In der obenstehenden Tabelle wird eine Übersicht über die verwendeten Sensorendaten gegeben. Bei Tests mit verschiedenen Personen stellte sich heraus, dass sich vor allem die Y-Achse als Steuerparameter eignet, da sie kognitiv bereits mit oben – viel und unten – wenig besetzt ist.

2.3 Vergleichbare Arbeiten

Kunst ist immer ein Produkt unserer Imagination und unserer Umwelt. So steht auch Maskenrad im Kontext dessen, was der Autor dieser Arbeit wahrgenommen hat. Nur durch das Wissen um vergleichbare Arbeiten kann eine Idee entstehen und die Originalität der Arbeit sichergestellt werden.

2.3.1 Technische Verwandtheit

Dominique Wollniok : „+50° 36‘ 16.27‘‘, +11° 34‘ 33.45‘‘ Drachenschwanz“

Ein im Rahmen des Projekts „Acoustic Turn“ unter der Leitung von Prof. Nathalie Singer und Mario Wiese realisiertes Projekt der Fakultät Medien an der Bauhausuniversität Weimar zeigt, wie man Kameratracking und Klanginstallation verbinden kann. Genau wie bei der vorliegenden Arbeit wird eine Microsoft Kinect für das Erfassen einer Person im Raum verwendet. Ziel der Installation ist das Erleben von Klangatmosphären („Sound Scapes“) anhand einer Naturfotografie. Dabei ändert sich das Klangbild je nach Position des Beobachters. Stellt man sich vor den Bildbereich, in dem ein Fluss zu sehen ist, so wird man folgerichtig das Geräusch von fließendem Wasser vernehmen.
Technische Parallelen dieses Projekts zur Installation Maskenrad sind das Tracking mittels der Kamera Microsoft Kinect und die Klangsteuerung und Implementierung in Max/MSP. Im Unterschied zu Maskenrad wurde allerdings EyeCon als Software für das Auswerten der Kinect-Daten und das Übertragen mittels Open Sound Control gewählt. Dieses Programm schloss der Autor aus, da es seit April 2010 keine Neuentwicklungen gibt, das Programm plattformabhängig nur Windows-PCs bedient und noch dazu kostenpflichtig ist (WEISS 2012).
Interessant ist, dass die Installation gleichermaßen nur für eine Person konzipiert ist und das Programm eine eindeutige Position benötigt. Stellt sich eine zweite Person dazu, so funktioniert die Installation nicht mehr. Bei Maskenrad ist es ähnlich, doch wird die Anzahl der Personen hier bereits durch die Konstruktion auf eine festgelegt, da auch nur eine Person Platz auf dem Podest findet.

Visual System: „Organic Culture"

Das Kollektiv Visual System aus Belgien ist bekannt für den intensiven Gebrauch von Licht in allen Farben. Viele Installationen, so auch „Organic Culture", zeichnen sich durch Fabwechsel und schnelle Veränderungen von Lichtsituationen aus. Dabei kann man bei einigen Arbeiten die Herkunft und Motivation des Kollektivs erkennen. „Little Ghost" und auch „Blue Rider" erinnern beispielsweise stark an Optiken von bekannten Retro-Computerspielen aus den achtziger und neunziger Jahren. Auch Maskenrad wechselt schnell zwischen verschiedenen Lichtsituationen, möchte jedoch weniger an ein Computerspiel erinnern. Daher wird bewusst auf reine Neonfarben wie Magenta, Giftgrün oder Grellgelb verzichtet. Außer in der Phase der Attraktion soll nicht der Eindruck entstehen, Maskenrad sei ein bunt leuchtendes visuelles Objekt, das zur Belustigung der Zuschauer gebaut wurde.

Yes!Yes!No!: „Night Lights"

Die Installation „Night Lights" des neuseeländischen Kolektivs Yes!Yes!No! beinhaltet großformatige Projektoren, die Bildsequenzen an die Fassade eines bekannten Gebäudes in Auckland strahlen, die interaktiv durch die Bewegung von Passanten vor einer weißen Wand generiert werden.
Im Gegensatz zur verwendeten Technik der vorliegenden Arbeit kommt hier reines Kameratracking zum Einsatz. Der Vorteil dieser Technik besteht darin, dass die Akteure nicht auf eine Kalibrierung der Kamera warten müssen und die Interaktion mit mehreren Personen problemlos möglich ist. Ein Nachteil ist die eingeschränkte Interaktivität, die mit diesem Verfahren möglich ist. Algorithmen können nur auf die Silhouette der interagierenden Personen zugreifen, nicht aber auf genaue 3D-Positionsdaten der Extremitäten.

Dieter Vandoren: Integration.03

Der niederländische Medienkünstler beschäftigt sich seit 2008 in seiner Serie „Integration" mit der Steuerung von Klang und Licht durch die Bewegung des menschlichen Körpers. Gestik spielt dabei genauso eine Rolle wie Ausdruck und Tanz. Vandoren greift für die Klangerzeugung durch Granularsynthese auf die Arbeiten von Iannis Xenakis zurück. Auch Maskenrad nutzt die Erkenntnisse über stochastische Klanggenerierung des griechisch-französischen Komponisten Xenakis (siehe „6.4 IanniX" auf Seite 40).

2.3.2 Konzeptuelle Verwandtheit

United Visual Artists: „Rien a Cacher Rien a Craindre"

Die interaktive Installation „Rien a Cacher Rien a Craindre" (dt. sinngemäß: „Nichts zu verstecken nichts zu befürchten") des britischen Kollektivs United Visual Artists hatte es im März 2011 zum Ziel, die Besucher der Gaîté lyrique in Paris zu „verleiten" und „verstören". Ein Spiel mit Dunkelheit, unerwartetem Abspielen von Aufnahmen der Gesichter von Ausstellungsteilnehmern und der positionsbestimmten Bestrahlung von Personen mit Scheinwerfern in einer aufreibenden Klangatmosphäre schafft Parallelen zur Installation Maskenrad.

Don Ritter: „Intersection"

Die mit über 600000 Besuchern außerordentlich erfolgreiche Klanginstallation „Intersection", die von Don Ritter seit 1993 ausgestellt wird, findet komplett im Dunkeln statt. Um in den Ausstellungsbereich zu kommen, mussten die Ausstellungsbesucher den 16 mal 13 Meter messenden Raum der Installation blind durchschreiten. Die Besucher hören währenddessen Geräusche von vorbeifahrenden Autos. Bleibt der Besucher stehen, so wird der Klang von quietschenden Reifen eines anhaltenden Autos abgespielt, in das bei längerem Stehenbleiben weitere Autos von hinten hineinfahren. Die Personen werden von von Infrarot-Sensoren erfasst.
Ähnlich wie bei der vorliegenden Installation Maskenrad wird dem Besucher keine Handlungsalternative gelassen. Man muss den Raum durchschreiten, in dem man – wenn auch nur klanglich – beinahe überfahren wird. Maskenrad lässt dem Besucher nicht die Wahl, das Podium zu verlassen, denn sonst stünde er im Dunkeln. Auch die Überraschung der Besucher über das Erfassen ihrer Position durch das System ist bei beiden Installationen gegeben. Interessant ist auch die Weiterentwicklung der technischen Hilfsmittel. Wurde 1993 noch auf Infrarot-Sensoren zurückgegriffen, würde man diese Installation heute wohl mit einem Infrarot-Kameratracking-System ausstatten.

Marco Donnarumma: „Xth Sense"

Genannt „biophysikalisches Musikinstrument", steuert die neue Entwicklung Xth Sense einen Klang nicht über das Tracking einer Kamera, sondern direkt über die bei Muskelbewegungen emittierten Tieffrequenzschwingungen. Die Herangehensweise unterscheidet sich grundlegend von der bei Maskenrad gewählten, da es keine auswertbaren Positionsdaten gibt. Aber auch Xth Sense wertet Gestik und Bewegung des Körpers zur Steuerung von Klangparametern aus.

John Maeda: „Simplicity"

Diese Referenz ist nicht nur eine Inspiration, sondern beeinflusste die gesamte Arbeit des Autors. John Maedas Simplicity steht für das Konzept der Einfachheit. Und das bedeutet nicht weniger Arbeit. Im Gegenteil ist es oft schwierig, zu einem einfachen Ergebnis zu kommen. Das Max/MSP Patch wurde einige Male umstrukturiert, um es effizienter und damit einfacher zu machen. Es wurde absichtlich auf jeglichen Texthinweis auf dem Bildschirm verzichtet, um nicht durch Komplexität vom Inhalt abzulenken. Auch die Bedienung durch Gesten ist sofort verständlich. Im Umkehrschluss soll in Phase02 (siehe „2.1.1 Phasen" auf Seite 3) konzeptuell Verwirrung durch Komplexität ausgelöst werden.

Morton Subotnik: „Sidewinder"

Das experimentelle Musikstück Sidewinder verwendet eine der in Phase02 eingesetzten ähnlichen Methode zum Stereopanning. Dabei werden Geräusche nicht gleichmäßig über die Zeit nach links und rechts gepannt, sondern „organisch" in zufälligen Verläufen von einem modulierten LFO.

ANBB – mimikry: „Berghain"

Bei der Entscheidung, ob und wie Sprache eingesetzt wird, half dem Autor im Besonderen das Werk „Berghain" von Carsten Nicolai alias Alva Noto und Blixa Bargeld, auch bekannt als Gründer und Sänger der Band „Einstürzende Neubauten". Letzterer spricht in „Berghain" mit klarer, fast monotoner Stimme und schafft es doch, eine tiefgehende Nähe zum Hörer zu erreichen.

Noisia – Split the Atom: „Alpha Centauri"

Die Dubstep-Gruppe Noisia sticht durch besonders drückende Klänge und harte Bässe hervor. Diesen Sound erreichen die Niederländer durch gut kalkulierte Verzerrung und Modulation der Bässe, aber vor allem auch durch das Einsetzen von hölzernen und metallischen Klängen. Diese Naturklänge schaffen eine schaurige subtile Nähe zum Hörer.

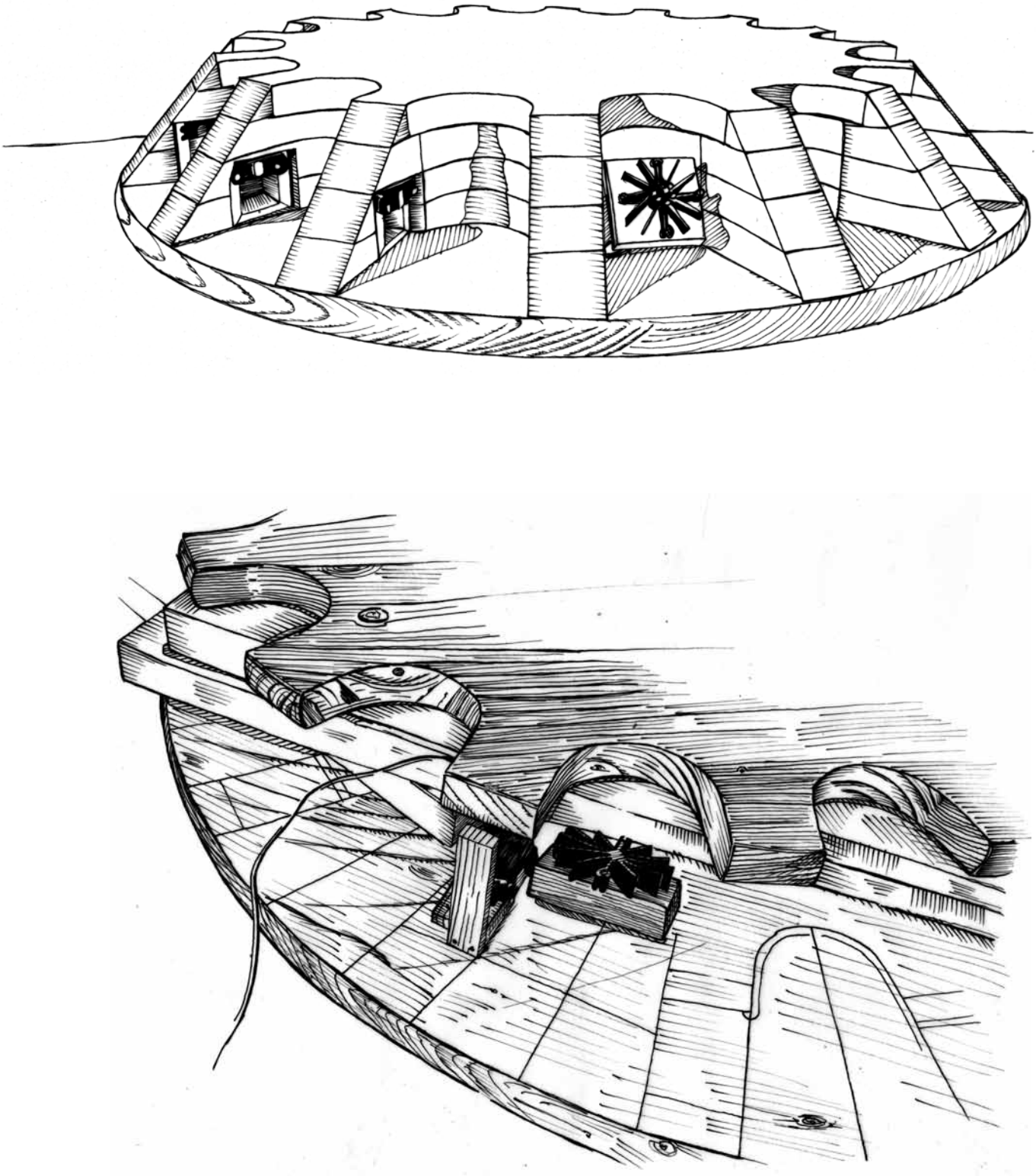


Abb. 5: Konstruktionszeichnungen

3 Konstruktion

Abb. 1: Fotografie des Maskenrads

Das Herzstück der Installation ist ein 15 cm hohes, kegelförmiges Podest mit 16 halbkreisförmigen Einbuchtungen. Die runde hölzerne Bodenplatte hat einen Durchmesser von einem Meter und eine Dicke von drei Zentimetern. Auf dieser Platte sind in Form geschnittene Styrodurplatten verklebt, auf denen eine abnehmbare Holzscheibe von 80 cm Durchmesser ruht. Die Standfläche von 60 cm Durchmesser besteht aus einer transparenten Plexiglasscheibe, die einseitig mit Folie bespannt ist (Abb. 1).

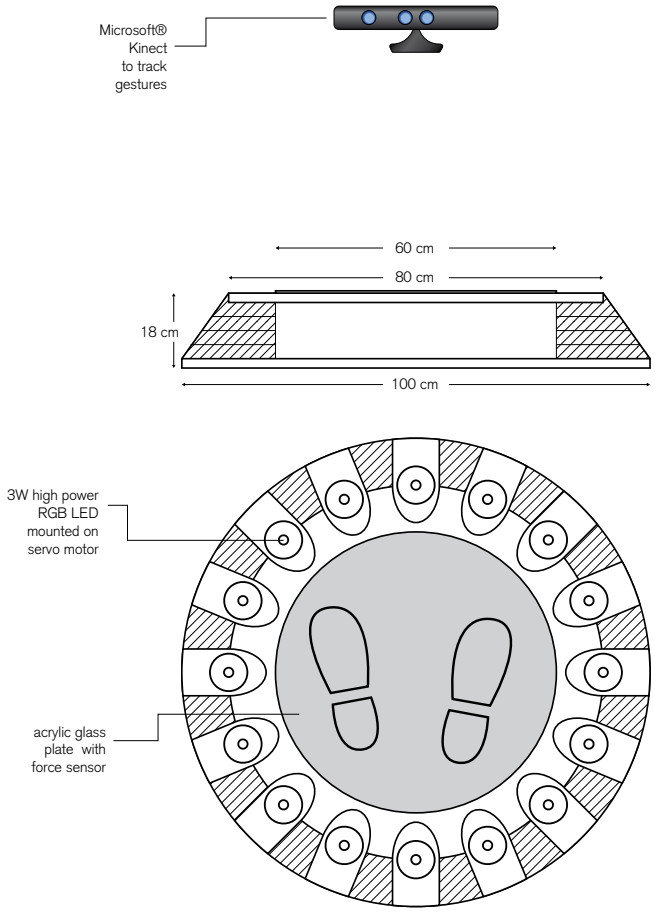


Abb. 2: Konstruktionszeichnung

In den Stegen der Ausbuchtungen sind die Servomotoren eingelassen, die die auf Holzplättchen montierten RGB LEDs schwenken. Das Objekt ist mit weißem PVC verkleidet.
Die Form des Objektes ist bewusst gewählt. Ein kreisrundes Podium verkörpert einerseits die Isolation in einem Kreis, der durch seine Rundheit keinen Ausgang oder Fluchtpunkt erkennen lässt. Andererseits steht man erhöht, wie auf einem Präsentierteller dem Geschehen ausgesetzt. Diese Eindrücke finden sich im Lebenszyklus der Installation wieder und bilden zwei Kerneigenschaften einer Intrige.

3.1 Material

Die Verwendung vieler verschiedener Materialien (Holz, Styrodur, PVC) machte die Konstruktion in Verbindung mit dem beschränken Platz für die elektronischen Bauteile zu einer Herausforderung. Alle Teile mussten sehr präzise gesägt bzw. geschnitten werden. Für diesen Zweck wurde zum Beispiel eine selbstgebaute Styroporschneidemachine verwendet, die mittels eines Konstantandrahtes die Styrodur-

platten auch rund ausschneiden konnte. Ein weißer, matter Acryllack sorgt dafür, dass die Holzteile besser mit der PVC-Verkleidung harmonieren und bei Anstrahlung das Licht gut aufnehmen. Nach einigen Versuchen stellte sich heraus, dass Holz, Plastikverschalung und Styrodurplatten am besten mit Epoxidharz zu verkleben sind, da dieses eine hohe Festigkeit und Beständigkeit aufweist und das Styrodur nicht chemisch angreift.

4 Elektronik

Die Installation Maskenrad verwendet eine Vielzahl elektronischer Bauteile, um die Lichtsituation und Ausrichtung der LEDs zu kontrollieren und um zu messen, ob sich jemand auf dem Podest befindet. Jeder Servomotor und jede LED kann für ein Höchstmaß an Flexibilität separat über eine serielle Schnittstelle angesteuert werden.

4.1 Bauteile

Maskenrad nutzt 16 High Power Vollong RGB LEDs (SUPERBRIGHTLEDS.COM 2012) auf Kühlkörpern, die an Hitec HS-322 HD Deluxe Servomotoren montiert sind. Die Bauteile werden mit einem 200W Netzteil bei 5V betrieben und von zwei Arduino Microcontrollern gesteuert, die mit dem Computer verbunden sind. Um die notwendige Anzahl von 48 separaten Steuersignalen für die jeweils drei Anschlüsse der 16 RGB LEDs bereitzustellen, musste ein Arduino um drei Texas Instrument TLC5940 integrierte Schaltungen erweitert werden. Diese LED Driver versorgen einen Transistorstromkreis mit dem nötigen PWM-Signal, um die Intensität der farbigen LEDs zu ändern und somit z.B. eine Farbänderung hervozurufen.

Als Eingabeschnittstellen dienen eine gehackte Microsoft® Kinect® als optischer 3D-Tiefensensor und ein Kraft-Sensor (FSR).

4.1.1 Aufgetretene Probleme

Die 3 Watt High Power RGB LEDs können nicht direkt mit einem Arduino Board betrieben werden, da sie mehr Strom ziehen, als das Arduino Board über eine USB-Verbindung leisten kann.
Der Schaltkreis besteht deshalb aus einem Steuer- und einem Arbeitsstromkreis, welche durch Transistoren voneinander getrennt sind. Die 48 parallelen Stromkreise, die den Strom jeweils für die rote, blaue und grüne LED der 16 RGB LEDs liefern,

werden geschlossen, sobald das PWM-Signal den jeweiligen Transistor schaltet.

Da selbst der Microcontroller Arduino Mega 2560 nur 12 veritable PWM Ausgänge hat, musste eine Lösung gefunden werden, um 48 unabhängige PWM-Signale senden zu können. Es gibt hierfür vorgefertigte Lösungen wie z.B. das Arduino PWM Shield von Practical Maker, was die Zahl der PWM Ausgänge auf 32 erhöht hätte. Allerdings scheinen die Shields vergriffen, bestellte Shields kamen jedenfalls auch nach Nachfragen nie beim Autor an und die Produktseite im Internet existiert nicht mehr. Die Lösung war also, direkt die auf dem Shield verbauten TLC5940 PWM Driver bei Texas Instruments zu bestellen und zu lernen, wie man sie mittels eines Arduinos ansteuerte. Einen großen Dienst erweist hierbei die TLC5940 Arduino Library von A.C. Leone (LEONE 2010). Das Bauteil benötigt vier PWM Ausgänge des Arduino, um daraus 16 PWM Signale zu machen. Die Driver werden dann mittels Daisy Chain verkettet, sodass man eine nahezu beliebig hohe Anzahl PWM-Signale realisieren kann.

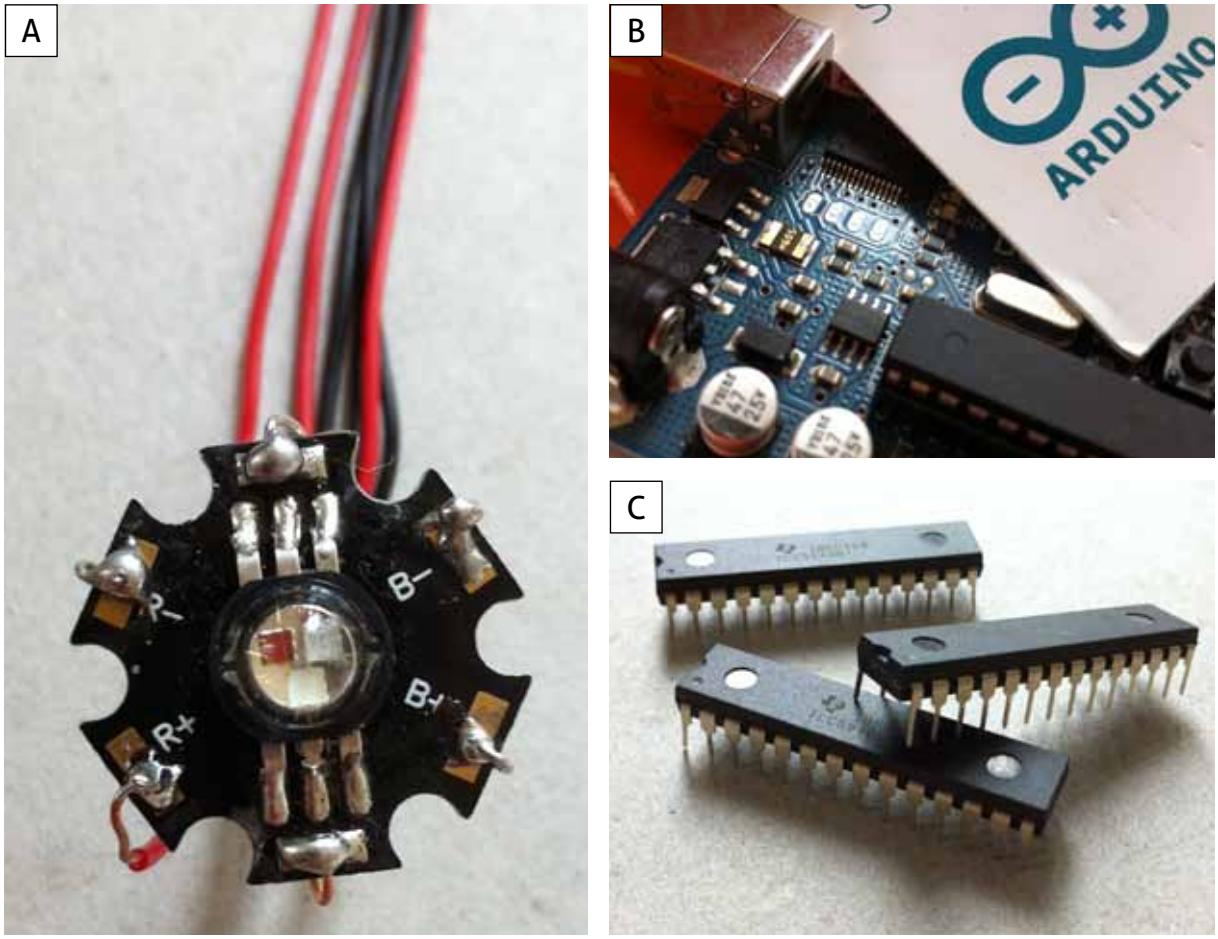


Abb. 3: Auswahl verwendeter elektronischer Bauteile
A Vollong 3W High Power RGB LED, **B** Arduino Microcontroller,
C Texas Instruments TLC5940 PWM Driver

Die TLC5940 PWM Driver sind Konstantstromsenken. Dieses "Detail" zu übersehen hat dem Autor als Elektronik-Neuling Tage seines Lebens gekostet. Konstantstromsenke bedeutet, dass der Strom nicht weg vom, sondern hin zum PWM Driver fließen muss. So konnte die Schaltung erst funktionieren, als die verbauten NPN Tranistoren durch PNP Transistoren ausgetauscht wurden.

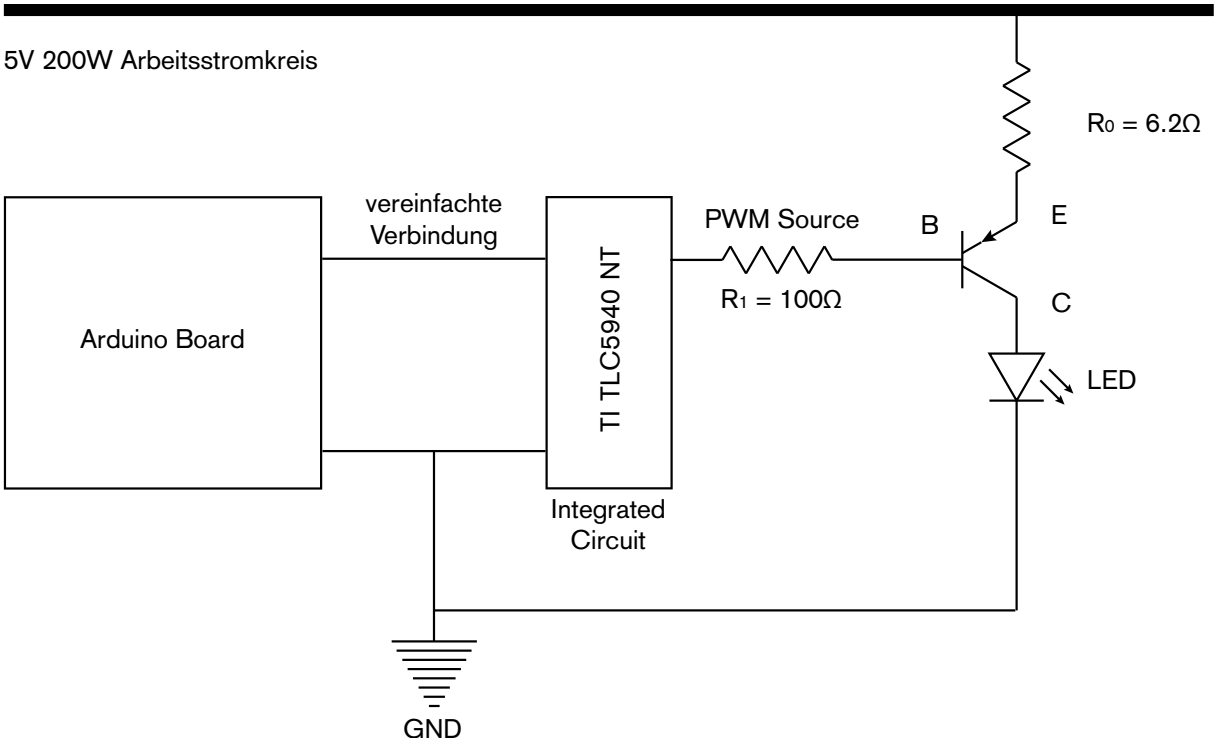


Abb. 4: vereinfachter Schaltplan

In Abb. 4 ist ein vereinfachter Schaltplan zu sehen. Der Arbeitsstromkreis ist nicht geschlossen, da der PNP-Transistor sperrt. Liegt eine Spannung an der Basis des Transistors an, so kann der Strom durch ihn hindurchfließen und die LED zum Leuchten bringen.

Der Vorwiderstand R_0 lässt sich berechnen, indem man die Diodenspannung (2,5V) von der Versorgungsspannung (5V) abzieht und das Ergebnis durch den Diodenstrom (400mA) teilt. Das Endergebnis ist 6,25 Ohm. Der nächste verfügbare, niedrigere Widerstand war 6,2 Ohm.

Auch musste die Dimensionierung des Netzteils bedacht werden. Nach Berechnung der ungefähren, insgesamten Leistung der Komponenten mit der Formel Leistung P = Spannung U × Stromstärke I stellte sich heraus, dass mindestens 200W Leistung benötigt werden.

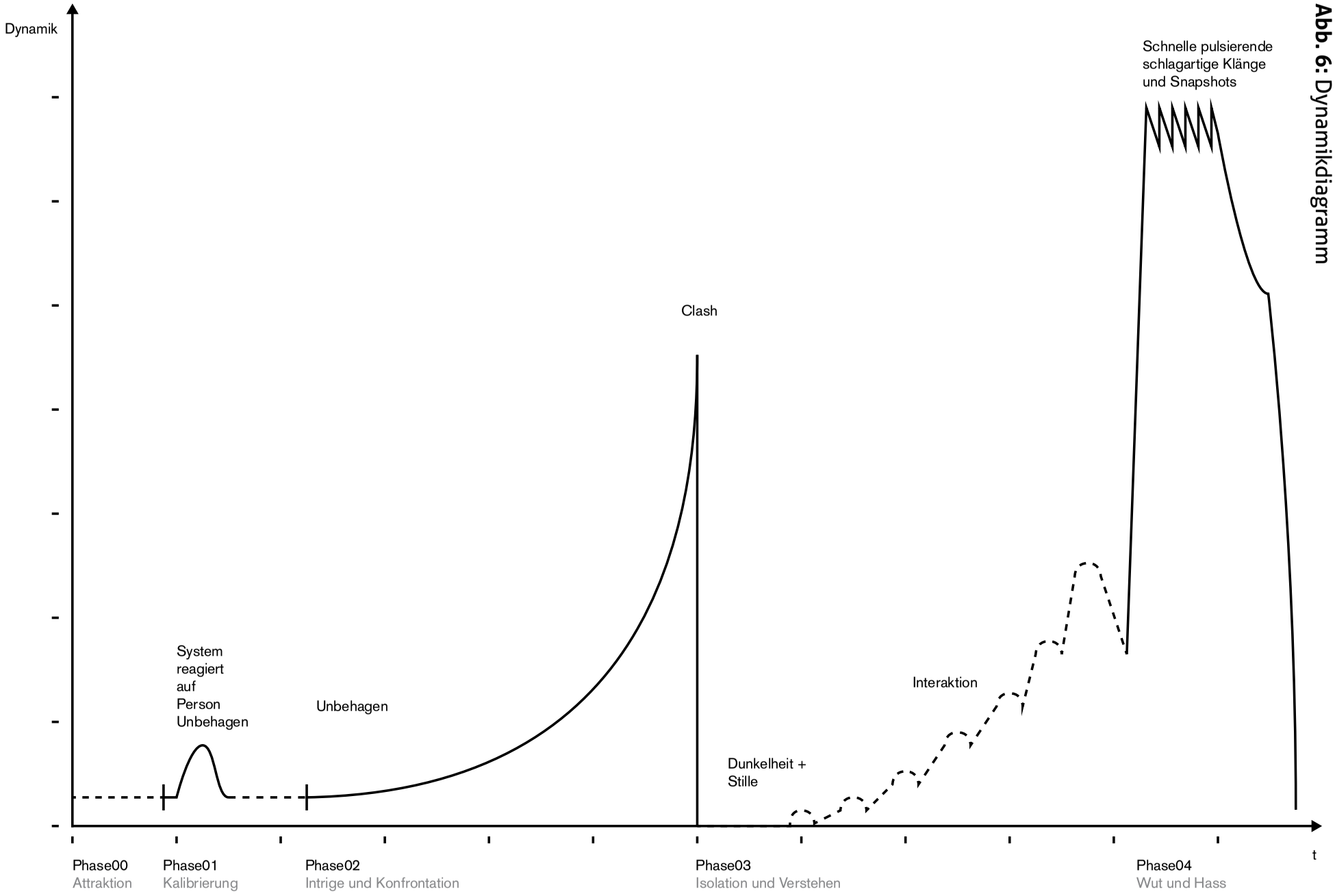


Abb. 6: Dynamikdiagramm

5 Sound Design

Das Einsetzen verschiedenster Mittel verhilft Maskenrad zu einer einzigartigen Klangatmosphäre.
Die Arbeitsschritte bei der Klangerstellung lassen sich systematisch nach einem Schema beschreiben. An erster Stelle steht das Konzept, das eine grobe Vorstellung über die gewünschte inhaltliche Funktion des Klangs vorgibt. Mithilfe von Mind Maps wird der Klang semantisch klassifiziert und daraufhin die subjektive Vorstellung in objektive oder zumindest intersubjektiv wahrnehmbare physikalische Eigenschaften übertragen.
Das geschieht entweder durch das Einstellen von Parameter von Softwareinstrumenten, das Aufnehmen und Verändern von Klang oder durch Synthese von Signalen direkt in Max/MSP.
Neue Herangehesweisen und interessante Klangstrukturen entstehen oft auch improvisativ beim Experimentieren mit Synthesealgorithmen.
Das Sound Design hängt vor allem stark von der Phase innerhalb des Lebenszyklus der Installation ab, in der der Klang vorkommen soll. Da die Phasen die Gefühlszustände exemplifizieren, die im Laufe des Durchlebens einer Intrige empfunden werden können, ist schon einmal ein gewisser Rahmen geschaffen.
Dazu kommt die entsprechende Lichtsituation und der Lichtwinkel. Manchmal ist die Konfiguration der LEDs konzeptuell bedingt, woraufhin sich eher der Klang an der Konfiguration orientiert. Es gibt aber auch Situationen, in denen sich eher das Licht und die Servostellung am Klang orientiert.
In Abb. 6 ist der dynamische Verlauf der Installation zu sehen. Es ist hierbei nicht der reine Amplitudenverlauf gemeint, sondern es soll ein Überblick gegeben werden, wie immersiv und in welcher Intensität die Installation zum jeweiligen Zeitpunkt auf den Besucher wirkt.

5.1 Phase00

Attraktion

Semantische Begriffe sind in dieser Szene „Neugierde", „Unschuld" und „Unbekümmertheit", da noch nichts vom Charakter des Makenrads offenbart ist und Ausstellungsbesucher angelockt werden sollen.
Unschuld assoziiert der Autor mit hellen, freundlichen, luftigen und glitzernden Klängen, aber auch mit Gewohntem und Kindlichem. Physikalisch wird diese Wir-

kung durch die Konzentration auf hohe Frequenzen, einen langsamen Attack für hintergründige Pad-Klänge und durch Harmonien erreicht.
Die sich im Vordergrund befindlichen kurzen Anschläge von Tönen der Bb-Dr-Tonleiter sind dem Klang eines Glockenspiels ähnlich. So werden Assoziationen an dieses Instrument hervorgerufen, das oft von Kindern gespielt wird und eine Unschuld ausstrahlt. Dazu mischen sich Dur-Dreiklänge der Bb-Dur Pentatonik. Der Klang ist hier so gestaltet, dass er gleichermaßen an eine Mundharmonika und an ein Akkordeon erinnert. Dadurch, dass diese Instrumente oft in der Folklore eingesetzt werden, erinnern sie in diesem digitalen Kontext an etwas gewohntes, harmloses. Synthetisiert wird der Klang im Softwareinstrument Massive subtraktiv mit drei Oszillatoren. Eine Mischung aus Rechteck- und Sägezahnwellen sowie die Waveform „Math II" kommen zum Einsatz, die von einem LFO moduliert werden. Das sorgt für den mundharmonikaähnlichen Klang.
Das Signal wird schließlich mit einem Tiefpass und dem Scream-Effekt gefiltert, mit einem Chorus und mit Hall belegt. Durch einen sehr langsamen Attack und einem schnellen Release baut sich der Klang bei jeder neuen Note langsam auf, was an das Aufziehen eines Akkordeons erinnert.
Um eine Neugier beim Betrachter auszulösen, soll die Sequenz wenig durchschaubar sein, aber dennoch nicht völlig zufällig und damit unruhig. Um eine sich in einem langen Intervall wiederholende, scheinbar zufällige aber doch miteinander in Verhältnis gesetzte Notenfolge zu realisieren, wurde der Notenabstand durch ein geometrisches Modell festgelegt. Dabei hilft die Software IanniX (siehe „6.4 IanniX" auf Seite 40). So können rhythmisch interessante und schwer zu durchschauende Sequenzen hergestellt werden. Durch diese undurchschaubare Bewegung von Klang und wechselnden Lichtsituationen interessiert sich der Beobachter für die Installation und entwickelt eine Neugier für sie.
Auch die Dreiklänge sind an das geometrische Modell gebunden, jedoch wird nicht bei jeder Auslösung eine neue Note gespielt, sondern gewichtete zufällige Notenlängen werden generiert und diese Zeit gewartet, bevor ein Befehl von IanniX einen neuen Dreiklang auslösen kann. Dadurch gewinnt man beim Anhören der Sequenz den Eindruck, die Musik wiege leicht hin und her. Wird aber lange keine neue Note ausgelöst, so steigert sich die Amplitude und der Scream-Effekt des Klanges kontinuierlich. Dieses „Aufschreien" versteht der Autor im Kontext der Erwartungshaltung der Maschine als ein Herbeirufen von Besuchern und als ein auf sich aufmerksam machen.

5.2 Phase01

Kalibrierung und Unbehagen

Stellt sich der Beobachter auf das Podium und wird Darsteller und Protagonist des Makenrads, verändert sich der Klang augenblicklich. Der Besucher soll das Gefühl bekommen, dass etwas mit der Maschine nicht in Ordnung sei. Sie soll den Eindruck erwecken, ein Eigenleben mit einem eigenen Willen und Charakter zu haben. So wird die Verstellung aus Phase00 aufgelöst.
Dies geschieht zunächst durch einen unmittelbar einsetzenden Beckenschlag, der von einem tiefen Subbassschlag begleitet wird. Dieser Klang wurde in einer DAW vorgefertigt, es wurde hierbei auf ein Sample der Kontakt 4 Bibliothek und eine frei verwendbare Audiodatei von Herbert Boland (BOLAND 2012) zurückgegriffen.
Sofort setzt auch eine Veränderung der Klänge aus Phase00 ein. Zuvor hell und harmonisch in Dur, verändert sich jetzt das Timbre sofort ins harmonische Moll der Bb-Moll Pentatonik. Der Klang bleibt auf der selben Basis wie zuvor, aber nun wirkt die Hüllkurve nicht nur auf die Amplitude, sondern auch auf den Pitch des ersten Oszillators. Der Klang wird also zunehmend verstimmt. Dies führt außerdem zum Phänomen der „Beats", das auftritt, wenn Schwingungen mit nah aneinander liegenden Frequenzen zusammengemischt werden. So ist der Klang nicht mehr rund und linear, sondern baut sich in zunehmenden Wellenbewegungen mit dem Steigen der Tonhöhe des ersten Oszillatoren auf. Die Wavetable-Position wird nun hart auf Sägezahnwelle gestellt und der Hallraum vergrößert. Außerdem treten die gewichteten Dreiklänge nun in zwei Oktavlagen auf. Das Timbre der impulsgefilterten Glockenspiel-Klänge bleibt wie gehabt, die Tonlage verschiebt sich allerdings um eine Oktave nach oben und es wird nun die Bb-Moll-Tonleiter gespielt.
Um das unbehagliche Gefühl zu verstärken, dass beim Verstimmen eines Tones erzeugt werden kann, wird ein weiterer Klangsynthesealgorithmus verwendet. Es mischen sich eine Anzahl tieffrequenter Sinusschwingungen zum Signal. In klassischen Konzerten und beim Anhören von Filmmusik faszinieren den Autor seit jeher tief und gleichzeitig einsetzende Hörner, Cellos und Kontrabässe, welche durch ihre Subbassanteile ein unbehagliches Empfinden auslösen können – im Film oft eingesetzt, um beim Betrachter eine Vorahnung eines Ereignisses zu erzeugen. Um ein ähnliches Empfinden auszulösen, werden die sich stetig in der Frequenz verändernden Schwingungen im Subbassbereich eingesetzt, um ein nicht klar wahrnehmbares Grollen zu erzeugen.

5.3 Phase02

Intrige und Konfrontation

Das Erreichen von Phase02 bedeutet, dass der Protagonist das Risiko eingegangen ist, trotz der Stimmung in Phase01 die Kalibrierung durchzuführen. Die Maschine hält den „naiven" Protagonisten folglich erfolgreich gefangen und trackt bzw. überwacht seine Bewegungen. Die Falle ist zugeschnappt und beginnt zu wirken. Der Protagonist wird während dieser Phase realisieren, dass das Maskenrad eine Falle ist.
Oft ist eine Intrige mit Erniedrigung, Scham und Peinlichkeit verbunden. Deshalb werden in dieser Phase nicht nur synthetische Klänge, sondern auch menschliche Laute und Sprache eingesetzt, um diese Emotionen stärker herauszuarbeiten.
Die Verwendung von Sprache ist deshalb wichtig, weil Intrigen vor allem über Absprachen funktionieren und sie ein zentraler Bestandteil für die hinterhältige Planung ist.
Zunächst wird ein veränderter, in der ehemaligen Waffenkammer des Gebäudes der Kreativ-Etage Weimar aufgenommener Klang einer zufallenden Tür abgespielt. Da Maskenrad in diesem Raum ausgestellt werden wird, schafft das die Illusion – oder führt zumindest zum Gedanken – man würde in den fensterlosen Raum eingesperrt werden.
Daraufhin setzt ein Rhythmus ein, der vor allem hölzerne und metallische Klangelemente enthält. Diese Naturklänge schaffen nach Meinung des Autors eine drückendere und tiefgehendere Atmosphäre als der Einsatz von synthetischen Klängen.
Fast zeitgleich beginnt das Abspielen der Sprachaufnahmen. Die dem Rhythmus entsprechende, zusammengestellte Sprachsequenz besteht aus den folgenden Sätzen und Worten: „Gerüchte – von langer Hand geplant – als dein guter Freund rate ich dir: Tritt zurück! – Fairness (Lachen) – jeder ist sich selbst der Nächste … Sich selbst der Nächste – Dolchstoß – im Namen Gottes". Die Art der Sprechweise ist nüchtern, sehr trocken und stellenweise arrogant. Sie ist vom Stück „Berghain" inspiriert, in dem Blixa Bargeld spricht (siehe „2.3 Vergleichbare Arbeiten" auf Seite 7). Durch die Nüchternheit und Arroganz wird eine macht ausübende Distanz zum Besucher geschaffen, der sich auf dem Podest stehend exponiert fühlt. Die Bedeutung des gesprochenen Textes greift verschiedene Felder der Gesellschaft auf, in denen Intrigen stattfinden. „Gerüchte" gibt es im Freundeskreis, Zurücktreten müssen vor allem Politiker, Fairness wird beim Sport nicht immer berücksichtigt. Sich selbst der Nächste sein ist ein Äquivalent zum in der Berufswelt weit verbreiteten Ellenbogen-Dogma. Mit dem Wort „Dolchstoß" wird an eine große Intrige der

deutschen Geschichte erinnert, während „im Namen Gottes" die Gesamtheit der hinterhältigen Verbrechen wie Hexenverbrennungen und der Verbreitung von Unwahrheiten für das eigene finanzielle Interesse der römisch-katholischen Kirche in ihrer Vergangenheit ausdrücken möchte.
Am Ende des gesprochenen Textes beginnt eine Frau aus sexueller Lust zu stöhnen. Aus meinem eigenen Freundeskreis ist mir bekannt, zu welchen ungeheuren Intrigen und Betrügereien der Mensch fähig ist, wenn es um Sexualität und Partnerschaft geht. Ein simpler Trieb triumphiert über das logische Denken - ja schlimmer noch - das logische Denken wird für das Planen und Ausführen einer Intrige gebraucht, die allein die Befriedigung deses Triebes zum Zweck hat. In der Liebe finden sich viele Intrigenopfer (VON MATT, 2009, S. 161).
Oft werden Reize als Lockmittel eingesetzt, wie beispielsweise der schöne Gesang der Sirenen in Homers Odysseus. Das Sprichwort „Liebe macht blind" lässt sich in diesem Zusammenhang auch auf die reine Sexualität beziehen.
Daher wird auch auf diese Spielart der Intrige eingegangen. Die Stöhnlaute wirken auf den Akteur befremdlich, wenn nicht sogar Scham erregend. Aufgelöst wird dieses Element durch Gelächter und einen höhnischen Kommentar in französischer Sprache („...la tête que t'avais! On aurait cru que je t'ai baisé, quoi!" [freie Übersetzung: „...wie du aussahst! Man hätte glauben können, ich hätte Sex mit dir gehabt"]). Die Fremdsprachigkeit ist in diesem Fall gewollt: Das Nicht-Verstehen ist ein zentrales Element dieser Phase.
So sind auch die übrigen Klänge in Phase02 gestaltet: Undurchschaubar und nicht zu verstehen. Eine Mischung aus Gestensteuerung und zufälliger Klangerzeugung lässt dem Besucher schon bei der Interaktion Rätsel offen. Zwar merkt er, dass eine Interaktion statttfindet, jedoch ist nicht klar, wie er den Klang tatsächlich verändert. Vor allem kann das Geschehen durch seine Interaktion nicht gestoppt werden.
Es handelt sich um eine Mischung von rauschartigen Klängen mit maschinell und künstlich-digital klingenden Rechteck- und Dreieckswellen. Diese Oszillatoren wurden deshalb ausgewählt, da Rauschen und ein moduliertes Rechteck- oder Dreieicksignal als besonders störend wahrgenommen werden. Letzteres klingt wie das wiederholte Warnsignal von Maschinen oder das Hupen von Autos. Diese Klänge verändern sich stetig über die Zeit. Mal eher im Hintergrund, können sie sich schlagartig in den Vordergrund drängen. Dies geschieht zufällig durch die Veränderung von Filterkoeffizienten und dem daraus resultierenden Verstärken und Vermindern unterschiedlicher Frequenzbereiche.
Auch wandern die Signale im Stereoraum, um dem Besucher den Eindruck zu geben, der Klang sei ständig in Bewegung und nicht zu kontrollieren. Zu einem späteren Zeitpunkt wird die Gestensteurung des Stereopannings durch die X-Position des Kopfes des Besuchers aktiviert. Die Einstellung ist jedoch seitenverkehrt: Blickt man

nach links, so wandert der Ton zum rechten Lautsprecher und umgekehrt. Damit wird ein „Entwischen" und „Verstecken" des Klangs simuliert, der die Intrige verkörpert.
Um dem Protagonisten aufzuzeigen, dass Bewegungen eine Wirkung auf das System haben, wird bei schnellen Bewegungen ein Filter-Swipe über das Signal veranlasst. Es werden kurz nacheinander alle Frequenzen des Signals von tief nach hoch betont. Diese Funktionalität steht metaphorisch für das Streifen des Klangs, der sich nicht bändigen lässt.
Gegen Ende der Phase wird der Klang notenweise in der Tonhöhe erhöht, währenddessen der Schrei eines Mannes ertönt und die restlichen Signale überlagert. Der Klang wurde in einer DAW vorgefertigt. Zu dem aus einer freien Quelle stammenden Schrei (FERMONT 2012) wird schnell ein Rauschen zugemischt. Das Rauschsignal wirkt durch seine Präsenz und seinen schlagartigen Release wie eine Ohrfeige. Der Schrei verhallt und es herrscht Stille. Damit ist der Clash der Phase02 abgeschlossen.

5.4 Phase03

Isolation und Verstehen

Nach dem Finale der Phase02 wird der gesamte Raum dunkel und still. Laut Konzept soll dieses Verhalten der Maschine im Protagonisten das Gefühl der Machtlosigkeit, Angst und Isolation erzeugen, das ein Opfer einer Intrige erlebt. Auch aus psychologischen Gründen sollte hier weder Ton noch Licht vorhanden sein, sodass der natürliche Eindruck, der Lebenszyklus der Installation sei vorbei, nicht zum Verlassen des Podiums durch den Besucher führt. In einem absolut dunklen Raum auf einem Podium stehend, wird kaum jemand den Entschluss fassen, den Raum zu verlassen. Nach einigen Momenten absoluter Ruhe ist das Rauschen von Wind und eine gleich bleibender tiefer, astral anmutender Ton zu hören. Zeitgleich beginnt ein Algorithmus zu wirken, der die Aufwärtsbewegungen der Arme dazu nutzt, um das Licht leicht zu erhellen. Der Protagonist muss selbst herausfinden, dass er durch alternierende Aufwärtsbewegungen der Arme „Licht ins Dunkel" bringen kann. Um das Verständnis dieser Handlung zu unterstützen und um diese konzeptionelle Idee hörbar zu machen, wird bei jeder Aufwärtsbewegung wird ein Klang ausgelöst, der mit der Zahl der Aufwärtsbewegungen klarer und heller wird. Es wird gleichzeitig der manipulierte Klang einer Triangel aus der Kontakt 4 Bibliothek ausgegeben und auch ein höherer Ton im Synthesizer Razor gespielt, der mit Parameterveränderungen des Softwareinstruments einhergeht. Dem Triangelklang wurde durch das Voranstellen

des rückwärts abgespielten Klang eine Unnatürlichkeit hinzugefügt. So klingt es mehr nach astraler Magie, da ein solcher Ton in der Natur nicht vorkommt.
Ist die volle Helligkeit erreicht, ertönt ein weiteres vorgefertigtes Audiostück, um den Erfolg zu bestätigen. Ein mit Kontakt 4 realisierter gesampleter Frauenchor singt eine Bb-Dur Kadenz. Die Stimmen werden kurz darauf durch einen Razor-Vocoder und durch Frequenzshifting verzerrt und in der Tonhöhe verändert. Außerdem wird ein digitaler Sample-and-Hold-Klang zugemischt, der mit einem Standard-Pro-Tools-Plugin mit dem Namen „Sci-Fi" produziert wird. Daraufhin beginnt Phase04.

Das windartige Rauschen bleibt während der gesamten Phase konstant. Es ist mit dem Synthesizer Massive gestaltet worden. Rauschen wird über einen Acid-Filter gefiltert, dessen Cutoff-Frequenz von drei sich kaskadierend modulierenden LFOs manipuliert wird. Daraufhin wird das Signal teilweise von einem Sine Shaper verändert, sodass es durch seine Verzerrung mehr wie ein eisiger Wind wirkt.

Der astrale, veränderliche Klang ist ein Produkt des Synthesizers Razor. Das Preset „Shining" wurde mit kleineren Feinjustierungen übernommen, da es sich sehr gut in die Phase einfügt. Die Parameter „Safe Bass Slope" und die Amplitude des zweiten Oszillators werden im Laufe der Phase verändert.

5.5 Phase04

Wut und Hass

Phase04 ist kurz und hart. Ihre Deutung ist in zwei Richtungen offen: Sie verkörpert entweder einen Racheakt oder die bloße Wut des Opfers einer Intrige, oder aber das Scheitern bei der Aufklärung einer Intrige, weil der Gegenpart schon einen Schritt voraus ist. Im Grunde genommen ist sie einfach nur eine Verkörperung von Wut und Hass, denn das ist das, was eine Intrige hauptsächlich hervorruft.
Es werden Snapshots, die während des Lebenszyklus vom Akteur gemacht wurden, gemischt mit Photos getriggert (siehe „6.2.7 Video control" auf Seite 33). Diese Funktion muss vom Klang getragen werden.
Um ein finales Inferno aus Klang und Licht zu schaffen, muss der Klang erschrecken und attackieren. Das Musikgenre Dubstep ist vor allem um eine „wobble bass" genannte Technik herum entstanden und für seine Kräftigkeit und Aggressivität beliebt. Die modulierten Bässe schwanken basierend auf der Frequenz des LFOs hin und her. Die Rate kann dann dem Takt angepasst werden, wodurch der Bass ein rhythmisches Element wird. Diese Technik wird auch in Phase04 eingesetzt. Zum

Takt wobbelt der Bass, was als sehr präsent empfunden wird. Zu jedem neu geladenen Snapshot wird eine neue Note gespielt und die Parameter des Wobble-Bass neu eingestellt. Außerdem wird jedes mal ein gesampleter Crashbecken- und Bassdrumschlag ausgelöst.
Ab Takt 19 verändert sich der Klang. Der Rhythmus scheint sich aufzulösen. Schließlich entfernt sich der Klang immer weiter vom Beobachter, wie in einer Traumsequenz, bevor die Maschine ihre Tätigkeit einstellt und der Lebenszyklus beendet ist. So wirkt die Installation am Ende wie ein böser Traum. Erreicht wird dieser Eindruck durch den Einsatz von Absynth 5 als Effektgerät. Das Preset „Sequential Airbrush" verzerrt den Klang rhythmisch. Ändert man gewisse Effektparameter, so wird der Klang verhallt, mehr und mehr verschwommen und scheint zu entschwinden. Es bleibt nur eine kurze Hallfahne, bevor Stille eintritt.

5.6 Vorzeitiges Abbrechen

Wird das Podium vorzeitig verlassen oder verliert die Synapse-Software den getrackten User aus dem Fokus, setzt sich ein Algorithmus in Gang, der den Protagonisten anweist, die Kalibrierung erneut auszuführen. Wenn dem nicht Folge geleistet wird, geht die Installation in Phase00 über, der Ausgangssituation.
Ziel des Sound Designs ist es, die Person zurück auf die Plattform zu holen, um den Lebenszyklus zu beenden. Es soll überraschen, dass die Installation auf das Verlassen reagiert.
Da es im scheinbaren Interesse der Maschine liegt, den Lebenszyklus zu beenden, ist die Reaktion der Maschine auf das Verlassen erbost. So als würde der Maschine dadurch Schaden zugeführt werden, beginnt eine Klangkomposition, die elektrische Schläge und Rufe einer robotischen Stimme enthält, die „Stop!" und „Komm zurück!" lauten. Die durch Veränderung einer aufgenommen Sprachpassage mittels eines Vocoders erhaltene Roboterstimme ist zuerst stark und kräftig, später nur noch hochtonig und leiser. Durch das Verwenden eines Vocoders, der stetig die Frequenz des Signals erhöht, wird der Effekt erzielt, die Maschine bäume sich noch einmal auf, nachdem das Geräusch des Herunterfahrens eines Computers schon das vorzeitige Ende der Installation vorausgesagt hatte. Der zugrunde liegende Klang ist ein frei verfügbarer Klang der Internet-Plattform „freesound.org" (CLUB SOUND 2012). Dieser Klang passt bereits so perfekt auf die Szenerie, dass nur noch das Hinzufügen weiterer Elemente nötig war, um zusammen mit der Lichtstimmung und Bewegung der Servomotoren einen gut inszenierten Systemabsturz zu imitieren.

6 Software-Entwicklung

Die Software des Projekts basiert auf der grafischen Entwicklungsumgebung Max 6 von Cycling'74 (CYCLING'74 2012). Die Entscheidung, nicht die OpenSource-Variante PureData, sondern Max 6 zu verwenden, ist vor allem der guten Unterstützung der Microsoft Kinect und des im Vergleich besseren Workflows zuzuschreiben. Zum Einsatz kommen außerdem Synapse for Kinect von Ryan Challinor, das Kinect Daten via Open Sound Control zur Verfügung stellt, und der grafische Sequencer IanniX, der auf Iannis Xenakis Arbeit aufbaut und bei der Servosteuerung und Klangerzeugung eingesetzt wird. Als VST-Plugins werden eine Reihe Programme von Native Instruments geladen: Razor, Massive, Absynth 5 und Maschine.

Um effizient zu einer funktionierenden Software zu kommen, wurde ein Zeitplan mit Milestones (dt.: „Meilensteine") verwendet, die bis dahin zu erreichen seien. Qualitätssicherung wurde ständig betrieben und das Patch ist vollständig kommentiert. Da Max/MSP auf mehreren Plattformen lauffähig ist, ist auch die Portabilität der Software gewährleistet. Eingeschränkt wird diese aber von den externen Programmen, die zeitgleich zum eigentlichen Programm laufen müssen (IanniX, Native Instruments, Synapse). Es kann sogar als Stand-Alone-Anwengung exportiert werden.

Bis zuletzt gab es aber erhebliche Schwierigkeiten mit der Performance des Programms. Die Rechenleistung des bisher eingesetzten MacBook Pro Late 2008 reicht nicht aus, um bei gleichzeitiger Verwendung von Max/MSP, Kameratracking, IanniX, Softwareinstrumenten und serieller Übertragung von Daten eine ausreichend geringe Latenz sicherzustellen. Auch bei performanceoptimierenden Einstellungen in Max/MSP (bspw. hoher Signalvektor) kommt es zu einem abgestocktem Verhalten und Befehle werden nicht mehr gleichmäßig an die serielle Schnittstelle gesendet. Bisher konnten noch keine Tests an leistungsstärkeren Rechnern gemacht werden, da viele, zum Teil kostenpflichtige Dependenzen wie Softwareinstrumente von Native Instruments auf dem Testrechner installiert werden müssten.

6.1 Max 6

Max ist eine grafische Entwicklungsumgebung für Signalverarbeitung von Ton und Bild. Das Programm ist durch externe Bibliotheken erweiterbar und als Lingua Franca im Bereich der Multimediainstallationen etabliert.

6.2 Aufbau des Max Patches

Das Max Patch führt alle Komponenten zusammen. Auf Basis der Sensordaten des Arduino und der Informationen der Kinect über die Körperstellung werden hier Entscheidungen getroffen und Prozesse angestoßen. Tritt man auf die Platte, ändert sich der Wert des Kraftsensors und Phase01 der Installation wird ausgelöst.
Durch die Integration von OSC-Daten in den Prozess ließen sich die Programme Max 6, Synapse Kinect und Iannix problemlos verbinden. Um die TLC5940 PWM Driver nutzen zu können, war es allerdings nicht möglich, auf eine Standardlösung zurückzugreifen. Das Firmata-Protokoll, das für gewöhnlich für die Kommunikation zwischen Max und Arduino eingesetzt wird, ist nicht für die Nutzung dieses Chips ausgelegt. Deshalb musste ein spezieller Arduino Code geschrieben werden.

6.2.1 Command center

Das Command center ist die zentrale Organisationseinheit des Patches. Hier werden Befehle geroutet und weitergeleitet, was für das Debuggen und Testen vorteilhaft ist. Es befindet sich in diesem Subpatch auch das globale Transport-Objekt, sozusagen das „Metronom" der Installation. Es werden hier die verschiedenen Phasen ausgelöst.
Außerdem sind hier die Algorithmen, die ausgeführt werden, wenn ein Besucher die Installation vorzeitig verlässt. Im Subpatch „cancel_process" sitzt die Logik, die die möglichen auftretenden Fälle unterscheidet. Hier muss genauso daran gedacht werden, dass ein Verlassen der Installation in Phase01 die Startphase aktivieren soll, aber auch daran, dass ein Zurückkommen zur Installation nach Verlassen in einer der anderen Phasen die Installation an der Stelle wiederaufnimmt, an der sie verlassen wurde. Für letzteres muss zum Beispiel der Transport angehalten werden und das vorher zwischengespeicherte Audiorouting des sich in der ersten Ebene des Patches befindlichen Matrix-Objektes wieder aufgerufen werden.

6.2.2 Synapse Kinect controls and routing

Für die Integration von Synapse werden die Max/MSP-Externals von der Synapse Webseite ebenso wie ein von Luke Woodbury veröffentlichtes Max Patch verwendet (SYNAPSE 2011, WOODBURY 2011).
Das verwendete Patch ist nicht viel mehr als eine grafische Benutzeroberflache für das Senden und Empfangen von OSC-Daten an Synapse. So können Einstellungen vorgenommen werden oder Positionsdaten und sogenannte „Joint-Hits" empfangen werden. „Joint-Hits" werden gesendet, wenn eine Körperbewegung (z.B. eine

Handbewegung) über eine gewisse Distanz mit einer gewissen Geschwindigkeit ausgeführt wird und können beispielsweise „/righthand up" beinhalten. Durch ein Routing wird diese Nachricht in ein „bang" umgewandelt – ein Symbol, das in Max/MSP für das Auslösen von Algorithmen verwendet wird.
Es sind drei Modi möglich, wie Positionsdaten von Synapse interpretiert werden sollen. Ein Modus setzt die Positionsdaten der Körperteile in Beziehung mit der Position des Torsos. Dieser Modus eignet sich am Besten für Maskenrad, da der Benutzer auf dem Podest steht und sich ohnehin nicht im Raum fortbewegt.
Jeder Algorithmus zur Erkennung und Verarbeitung von aufwändigeren Gestiken ist in ein Subpatch ausgegliedert, wohingegen einfach Anwendungen von Werten direkt auf z.B. die Klangsynthese wirken. Für Phase02 wurden die Gestik-Algorithmen „two_hands_down" und „watch_your_back" entwickelt. Für Phase03 sind es „hands_up_alternate" und „hands_pos_servo". Phase04 nutzt nur die Joint-Hits für die Interaktion.

Über die Nachricht „/tracking_skeleton" kann festgestellt werden, ob zum gegenwärtigen Zeitpunkt ein Benutzer von Synapse erfasst ist. Befindet sich die Installation in Phase01, wird dadurch Phase02 ausgelöst.

6.2.3 Erläuterung der Gestik-Algorithmen

two_hands_down

Dieses Subpatch routet die Joint-Hits der Hände so, dass eine Aktion ausgelöst wird, wenn beide Hände eine Bewegung gleichzeitig entweder nach unten oder vorne, oder die rechte Hand nach rechts und die linke Hand nach links ausführt. Da die Bang-Symbole normalerweise allerdings nicht exakt gleichzeitig eintreffen wird im Subpatch „same_time" bei Empfangen eines Bang-Symbols eine Zeit von 300ms auf das zweite Bang-Symbol der jeweils anderen Hand gewartet, bevor eine Bestätigung der Ausführung an die „mission_control" im Command center gesendet wird. Es wird dann ein Snapshot-Befehl an die Video controls gesendet. Die Gestik entspricht einem Wegstoßen und Verteidigen.

watch_your_back

Das Subpatch „watch_your_back" ist nicht in die Synapse Kinect controls and routing Sektion integriert, sondern befindet sich direkt im subpatch „p02_sp" in der Sound Processing Sektion. Das liegt daran, dass die Gestiken sehr direkt auf die Klangerzeugung wirken und keine Vorbehandlung durch Steueralgorithmen benötigen. Sie werden nur auf die gewünschte Parametergröße gemappt. Die X-Position

der Hände wird dazu verwendet, das grundlegende Signal der Synthese zu verändern. Die linke Hand fadet zwischen der Verwendung des rand~-Objektes und des vs.rand0~-Objektes hin und her, während die rechte Hand dazu verwendet werden kann, zwischen einem rect~-Rechtecksignal und einer tri~-Dreieckswelle stufenlos zu wählen.
Die X-Position des Kopfes ist für den Mix aus beiden Signalen verantwortlich und wird später, nachdem an „become_weird" eine Eins gesendet wurde, auch für das Panning des Stereosignals verwendet (jedoch seitenverkehrt).

hands_up_alternate

Werden linke und rechte Hand abwechselnd nach oben bewegt, wird eine Variable inkrementiert. Sobald die rechte Hand zum ersten Mal eine Aufwärtsbewegungs ausgeführt hat, gibt ein switch-Objekt das Bang-Symbol nur noch alternierend weiter. Jedes durchgelassene Symbol erhöht die Lichtstärke um 10%. In einer früheren Version von Maskenrad ging die Lichtstärke nach Ablaufen der 1,5-fachen Zeit zwischen zwei Aufwärtsbewegungen wieder um 10% zurück. Um auf 100% Lichtstärke zu kommen, musste der Protagonist also nicht nur alternierende Aufwärtsbewegungen ausführen, sondern immer schneller werdend alternieren. Diese Funktionalität ließ sich in der Praxis nicht etablieren, da die Testpersonen den Algorithmus nicht erkannten und die Installation abbrachen. In der aktuellen Implementierung müssen zwar alternierend die Hände gehoben werden, der Zeitfaktor spielt jedoch keine Rolle. Bei 60% Lichtstärke wird ein Snapshot-Befehl an die Video controls gesendet. Bei 100% Lichtstärke wird eine Bestätigung an die „mission_control" im Command center geschickt und Phase04 wird ausgelöst.

hands_pos_servo

Die Z- und Y-Positionen beider Hände werden in diesem Subpatch vom Algorithmus verarbeitet, sodass die Y-Position den Winkel der Servomotoren bestimmt und die Z-Position die Anwendung des Winkels auf bestimmte Servomotoren ausführt, indem der Winkelwert entsprechend mit einer Zahl zwischen Null und Eins multipliziert wird. Diese Funktionalität wird mit deinem zl.rot-Objekt realisiert, das den auf einen Wert von 1 bis 8 skalierten Z-Achsen-Positionswert als Parameter verwendet, um eine Liste aus Faktoren um die jeweilige Anzahl an Schritten zu rotieren. Um trotz der harten acht Schritte eine weiche Bewegung der Servomotoren sicherzustellen, existiert das Subpatch „list_smoother", das mit einer Latenz von 500 Millisekunden über einen Verlauf den neuen Winkel einstellt.
Die Daten des „list_smoother" werden dann auch der LED Control Sektion zur Verfügung gestellt, sodass die richtigen LEDs angesteuert werden.

6.2.4 LED control

Es befinden sich sowohl die Befehls-Nachrichten wie auch die LED-Objekte innerhalb des Subpatches „LEDs".
Innerhalb der 16 Subptches „rgb_led" findet die Logik statt, die die LEDs in verschiedenen Modi steuert. So müssen nur noch entsprechende Nachrichten an das Subpatch gesendet werden. Das spart Rechenleistung, denn die Subpatcher sind Objektinstanzen, die denselben Algorithmus mit verschiedenen Variablen für jede LED verwenden.
Angeordnet nach den fünf Phasen finden sich Objekte, die vom Command center ausgelöst werden, um eine LED-Funktion entsprechend der aktuellen Phase zu aktivieren. Das Subpatch „led_color_ramp" sorgt in Phase00 dafür, dass die Farben der LEDs sich langsam über das gesamte Farbspektrum verändern. Um diese Funktionalität zu realisieren, werden nacheinander sich wiederholdende Verläufe für jede einzelne LED aktiviert, sodass der Effekt entsteht, die Farbe würde von LED zu LED wandern. Phase01 behält diese Funktioalität bei, sendet aber den Befehl an die LEDs, die Intensität über die Zeit von einer Sekunde bis auf 4% der Helligkeit zu vermindern. Phase02 sendet an das „rgb_leds"-Subpatch eine Nachricht, die es dazu veranlasst, die LEDs, ausgehend von einer Farbe, sich immer weiter von dieser Farbe zufallsgesteuert zu entfernen. Dies geschieht mithilfe mehrerer drunk-Objekte.
Auch ist ein Bestätigungsaufleuchten nach der Kalibrierung sowie ein Wechseln der Farbe nach rot bei Takt 21 der Phase implementiert. Ein weiterer Algorithmus verkunkelt die LEDs schlagartig bei jedem ungeraden Taktanfang, um es so erscheinen zu lassen, als ob der zeitgleich zu hörende Bassschlag der Grund dafür sei. Das Licht wird über ein line-Objekt innerhalb eines Taktes wieder auf seine volle Leuchtkraft gebracht.
Phase03 setzt die Daten aus den Subpatches „hands_pos_servo" und „hands_up_alternate" um. Die Z-Position der Hände werden analog zur Verteilung auf die Servomotoren auch auf die LEDs verteilt – mit dem Unterschied, dass sie vorher von mit dem aktuellen Wert aus „hands_up_alternate" skaliert werden. Das führt dazu, dass die LEDs in Abhängigkeit vom Fortschritt der Interaktion heller werden.
Wird Phase04 aktiv, pulsiert das Licht rot. Das Signal, das die Pulsierung auslöst, kommt aus dem Subpatch „dirty_signal" in der Video Control Sektion, da diese Sägezahnwelle auch zur Steuerung von Videoparametern verwendet wird. Es wird skaliert und gegen Ende der Phase schwächer, wenn der Absynth-Effekt hinzugemischt wird. Außerdem wird bei jeder von einem Zufallsalgorithmus in der Servo Control Sektion veranlassten Neustellung eines Servowinkels ein Lichtblitz der LED auf dem entsprechenden Servomotor ausgelöst. Dies geschieht durch das Senden einer

Nachricht an das line-Objekt in der „Plus Section" innerhalb eines jeden „rgb_leds"-Subpatch. In der aktuellen Implenetierung wird so der aktuellen Intensität ca. 20% hinzugefügt und innerhalb 200 Millisekunden wieder auf das ursprüngliche Niveau gebracht.
Innerhalb der „rgb_leds"-Subpatch befindet sich ein swatch-Objekt, dass sich gut für Umrechnungen zwischen den Farbsystemen HSL und RGB eignet. Um zu gewährleisten, dass in einem Abbruchprozess die LEDs nicht gleichzeitig verschiedene Befehle empfangen, wurde für diesen Fall ein switch-Objekt eingebaut, dass bei Schalten nur die Befehle aus dem Subpatch „cancel_process" weitergibt.

Abb. 7: Maskenrad in Aktion

6.2.5 Servo control

Im Subpatch „Servos" wird der Status der jeweiligen Phase dazu verwendet, ein switch-Objekt zu schalten. So können jeweils nur die Algorithmen wirken, die für die Phase vorgesehen sind. Produzierte Daten für die Servostellungen werden dann an die Arduino Connect Sektion weitergegeben. Phase00 leitet die IanniX-Daten

weiter, während mit dem Start von Phase01 die letztmalig empfangenen IanniX-Daten im subpatch „alignment" sanft auf einen 120°-Winkel angeglichen werden, sodass sie den Besucher anstrahlen. Wird Phase02 ausgelöst, beginnt der Algorithmus des Subpatches „slight_waves" zu wirken. Entsprechend der Datenrate für die serielle Kommunikation mit den Servos wird hier alle 30 Millisekunden eine Sinusschwingung abgetastet. Durch eine Phasenverschiebung von $^1/_{16}$ zwischen benachbarten Servomotoren entsteht eine Wellenbewegung. Die Servosteuerung in Phase03 ist direkt in die Gestensteuerung im Subpatch „hands_pos_servo" integriert. Wird Phase04 aktiv, so beginnt ein metro-Objekt, kontinuierlich in einem bestimmten Intervall „bang"-Befehle zu senden. Diese Befehle veranlassen einen zufälligen Servomotor dazu, seinen Winkel in Zwanzigerschritten auf einen Zufallswert zwischen 40 und 140 zu setzen. Das Intervall des metro-Objekts ist abhängig vom zufällig generierten Notenwert aus dem Subpatch „p04_sp".
Die Nachrichten werden insgesamt nur dann an die Arduino Connect Sektion weitergeleitet, wenn der Abbruchprozess nicht gestartet wurde. In zweiterem Fall wird einen zweites switch-Objekt geschaltet, so dass prioritär der Algorithmus des Subpatches „angry_servos" zum Tragen kommt. Innerhalb dieses Patches sorgen 16 Instanzen des Externals „vs.between" von Virtual Sound dafür, dass die Servomotoren zufällige Winkelwerte in einem bestimmten Bereich empfangen (CIPRIANI & GIRI 2010). Das führt zu einem Zittern der Servomotoren.

6.2.6 IanniX

Die OSC-Daten der IanniX-Software werden hier mit dem udpreceive-Objekt empfangen. IanniX sendet die Positionen der Cursor und detektiert die Kollision eines Cursors mit einem Trigger-Point (siehe „6.4 IanniX" auf Seite 40). Bei einer Torus-Anordnung des IanniX-Scores ergibt sich bei verschiedenen Geschwindigkeiten der Cursor eine Wellenbewegung, die immer unregelmäßiger wird, nach einem bestimmten Intervall aber wieder zur ursprünglichen Wellenform zurückfindet. Diese Bewegung wird in Phase00 eingesetzt, um die Servomotoren zu steuern. Das Kollisions-Event dient als Auslöser für die einfachen perkussiven Klänge, die in Phase00 und Phase01 durch Impulsfilterung synthetisiert werden.
Realisiert wird diese Funktionalität durch das Empfangen der OSC-Daten und dem darauf folgenden Routen der Cursor- und Trigger-Daten an das entsprechende Teil im Progamm. Über ein udpsend-Objekt können Start-, Stop- Zurückspul- und Geschwindigkeitsbefehle an das Programm IanniX gesendet werden.

6.2.7 Video control

Ziel dieser Einheit ist das Aufnehmen von Snapshots, das Manipulieren des Bildmaterials mit Filtern und Effekten und das Abspielen der Bildfolge in Phase04. Der Infrarot-Videostream der Kinect-Kamera, der durch das External jit.synapse sichtbar gemacht weden kann, wird kontinuierlich in eine Jitter Matrix geschrieben. Sendet man dieser Matrix eine entsprechende Nachricht, so speichert sie einen Snapshot im jxf-Format ab. Dieses unkomprimierte Format kann schneller gelesen werden als zum Beispiel JPG und verbessert damit die gesamte Performance.
Über ein sich wiederholendes counter-Objekt werden die Nachrichten inklusive laufender Nummer generiert, sodass sie nur bis zur Nummer sechzehn laufen können. Wird aus anderen Teilen des Programms ein „bang" an den Snapshot-Algorithmus gesendet, wird eine Nummer hochgezählt und eine Datei gespeichert. Das „take_photo"-Subpatch funktioniert genauso, nur dass ein Snapshot manuell ausgelöst wird und der Dateiname ein anderer ist.
Wird in Phase04 ein „bang" an einen Algorithmus in der Video control Sektion gesendet, so wird durch ein weiteres counter-Objekt eine Zahl bis sechzehn hochgezählt. Ein random-Objekt sorgt dafür, dass entweder ein der Zahl entsprechender, vorher automatisch geschossener Snapshot, ein aktuelles Livebild aus dem Kinectstream oder ein manuell geschossener Snapshot in eine Jitter Matrix geladen wird.

Diese Bilddaten werden daraufhin an eine Reihe VIZZIE-CLIPPINGS weitergeleitet. Mit diesen vorgefertigten Max-Objekten lassen sich grundlegende Videofilter, -fades und -effekte leicht realisieren. Nachdem das Bildsignal mit dem Effekt SKRIBBLR versehen wurde, der die Konturen hart nachzeichnet, wird es zum Originalsignal gemischt. Schließlich erreicht das Signal das Subpatch „dirty_signal", das aus einem Jitter Tutorial von Andrew Bensos von der Webseite „http://cycling74.com/author/abenson/" stammt und in großen Teilen verändert wurde. Aufgabe dieses Patches ist es, dem Bildsignal Störeffekte hinzuzufügen. Es wird eine Bildverzerrung wir bei defekten Röhrenfernsehern simuliert, die auch mit Bildverschiebungen in der Y-Achse einhergeht. Das Bildsignal wird außerdem stark gesättigt und ein Blooming-Effekt wird angewandt, der von einer Sägezahn-Schwingung moduliert wird.
Diese Schwingung wirkt ebenso auf die Intensität der LEDs, sodass der Effekt in Bild und Licht gleichermaßen vorkommt. Um schließlich das Graustufen-Bild rot einzufärben, wird es einfach das zweite Inlet eines jit.pack-Objektes gespeist, was bei einem RGBA-Signal der Ebene eins entspricht, der roten Farbebene. Das bearbeitete Signal wird über ein weiteres VIZZIE-CLIPPING mit dem Namen VIEWR im Fullscreen-Modus ausgegeben.

Die Video Control Sektion beinhaltet auch einen Algorithmus zum Abspielen des Quicktime-Films, der in Phase01 in Endlosschleife gezeigt wird. Auch Fades sowie das vertauschen der Farbebenen in Phase02 von grün nach rot ist implementiert. Die Farbebenen werden getauscht, um zu signalisieren, dass die Kalibrierung abgeschlossen ist. Zusammen mit dem akustischen Signal aus dem Subpatch „p02_sp" wird damit die zweite Phase eingeleitet.
Ursprünglich für die Aufnahme des Kinect-Videostreams für die spätere Bearbeitung in Adobe After Effects implementiert, lässt sich der Algorithmus zum Aufnehmen von Videosignalen auch gut für Dokumentationszwecke oder für spätere Projekte mit den aufgezeichneten Streams nutzen.
Die Bearbeitung des Quicktime-Movies in After Effects hatte zum Ziel, die Kalibrierungssituation zu verdeutlichen. Durch das Maskieren von Bildanteilen und durch die lumineszierende grüne Farbe entsteht der Eindruck, ein Scan würde durchgeführt. Gleichzeitig zeigt die aufgenommene Person, dass für die Kalibrierung die Arme gehoben werden müssen. Das Schimmern wurde durch die Kombination eines konturenbetonenden Effekts mit einer Überlagerungsfunktion erreicht.

6.2.8 Arduino connect

Im „serial"-Subpatcher findet die Kommunikation zwischen Max/MSP und den beiden Arduinos statt. Dazu dienen zwei serial-Objekte mit den ports a und b. Die Baudraten sind unterschiedlich und wurden empirisch optimiert.
Aus Performancegründen und um die Arduinos mit den seriellen Nachrichten nicht zu überfordern, wird die Ausgabe von einem speedlim-Objekt gedrosselt.
An eine Nachricht für die LEDs wird mit „prepend 255" der Nachrichtenkopf 255 vorangestellt, damit der Arduino-Code korrekt ausgeführt wird. Der Arduino-Code für die Verarbeitung von Nachrichten an die Servomotoren wird über die „messenger"-Bibliothek abgewickelt. Für das korrekte Funktionieren des Call-and-Response-Mechanismus muss nach jeder Nachricht der ASCII-Character „A" mit dem numerischen Wert 13 gesendet werden. Dieser wird als separate Nachricht durch ein trigger-Objekt gleich hinterhergeschickt.
Sendet man ein „bang" an ein serial-Objekt, so wird ausgegeben, was das Arduino Board durch den Befehl „Serial.write()" an die serielle Schnittstelle sendet. Durch ein route-Objekt werden die Call-and-Response-Steuerdaten herausgefiltert. Was übrig bleibt sind die Sensordaten, die das Arduino Board kontinuierlich sendet. Dieser Wert wird an das Command center weitergeleitet. Es ist möglich, den Threshold des Kraftsensors für durch einen Klick auf einen Wert, der 50 kleiner ist als der bisher höchste gemessene Wert (mit dem peak-Objekt) zu setzen.

6.2.9 Sound Processing

In der Sound Processing Section finden sich die Subpatches für die Klangsynthese der jeweiligen Phase, denen die Softwareintrumente in eigenen Patches vorgelagert sind. Die VST-Plugins befinden sich nicht innerhalb der Klangsynthese-Patches, damit sie für verschiedene Phasen eingesetzt werden können. Unterhalb der Synthese-Patches befindet sich dann ein matrix~-Objekt, das nur den Klang eines bestimmter Klangsynthese-Patches an den Audioausgang bzw. das dac~-Objekt weitergibt („dac" steht für Digital-Analog-Konverter).

Für ein besseres Verständnis empfiehlt es sich, begleitend zu diesem Kapitel die in Max/MSP geöffneten Patches zu betrachten.

Phase00 – „p00_sp"

Die von IanniX empfangenen Trigger-Hits beinhalten die Nummer des getroffenen Triggers. Diese Nummer wird dazu verwendet, innerhalb des Subpatches „bbdur" eine Note aus der Bb-Dur Tonleiter auszuwählen und den MIDI-Wert dieser Note weiterzusenden. Verbunden mit einer „target"-Nachricht wird mit diesem Wert eine Instanz des poly~-Objektes mit dem Namen „clickvoice" angesprochen, die mithilfe eines einfachen reson~-Bandpassfilters ein click~-Impuls filtert und so einen hellen kurzen Klang erzeugt.
Bei Aktivierung der Phase wird ein metro-Objekt gestartet, das kontinuierlich „bang"-Nachrichten sendet. Einerseits werden diese Nachrichten an ein itable-Objekt gesendet, dass auf „bangs" mit dem zufälligen Ausgeben eines gespeicherten Wertes reagiert. Diese Funktion kann dazu genutzt werden, wahrscheinlichkeitsabhängige Wertegenerierung zu implementieren und sorgt dafür, dass das metro-Objekt in unterschiedlichen Intervallen „bangs" ausgibt.
Andererseits wird das „bang" des metro-Objekts auch an ein onebang-Objekt weitergeleitet, welches auf ein anderes „bang" wartet, das von einem IanniX-Trigger ausgelöst wird. Erst wenn dieses angekommen ist, schickt das Objekt ein „bang" weiter. Das hat zur Folge, dass der folgende Algorithmus erst nach Ablauf des Notenwerts und nach Kollision eines IanniX-Cursors mit einem -Trigger ausgeführt wird. Es wird nun ein gewichteter zufälliger, aber harmonischer Akkord der Bb-Dur-Pentatonik an das Softwareinstrument Massive gesendet, welches den Klang produziert.
Trotz des irreführenden Namens „p00_sp" findet in diesem Patch nicht nur die Klangsynthese für Phase00 statt, sondern auch der Übergang und die veränderte Klangsituation in Phase01.
Wird Phase01 durch betreten des Podests eingeleitet, ändern sich einige Parameter

durch automatisches Auswählen eines Presets in einem preset-Objekt. Das beinhaltet das Verändern einiger Klangeigenschaften im Softwareinstrument Massive, das Ändern der Oktavlage der Impulsfilterung und des Oktavraumes der Dreiklänge.
Es wird außerdem ein Prozess angestoßen, der zu dem Signal des poly~-Objektes „clickvoice" das Signal eines weiteren, sechsstimmigen poly~Objektes mit dem Namen „padvoice" mischt. Dieses Objekt erzeugt sechs auf den Bewegungen der IanniX-Cursor basierende Sinusschwingung. Die Frequenz der Sinusschwingungen wird exponentiell über die Position der Cursor berechnet, sodass sie nicht unter 40 aber auch nicht über 184 Hz kommen.

Phase01 – p01_sp"

In diesem Subpatch wird nur eine vorgefertigte, in einen Buffer geladene Audiodatei mittels eines groove~-Objekt abgespielt, sobald Phase01 aktiviert wird.

Phase02 – „p02_sp"

Das Subpatch für die Klangerzeugung in Phase02 ist bezeichnenderweise auch das größte und unübersichtlichste Subpatch der Sound Processing Sektion.
Zu Beginn wird das Audiosignal der ersten beiden Phasen ausgefadet und schließlich werden die vorangehenden Phasen deaktiviert, das Routing eingestellt und IanniX gestoppt. Zeitgleich wird im subpatch „door_shot" über ein groove~-Objekt der zuvor modifizierte Klang einer zufallenden Tür abgespielt. Sobald das Abspielen fertig ist, wird der Bypass des Softwareinstruments Maschine bei einem ungeraden Takt aufgehoben, sodass sie mit einem Bassschlag beginnt, Samples in einem Rhythmus auszugeben. Maschine ist eine Kombination eines Samples mit einem Sequenzer und weit darüber hinaus.
Ein Takt später wird eine Audiodatei gestartet, die Sprache enthält und sich bis zum Ende der Phase durchzieht. Außerdem ist es ab diesem Zeitpunkt möglich, durch drei in einem Zeitraum von 300 ms aufeinander folgende, schnelle Bewegungen von Kopf, Händen oder Füßen im Subpatch „laughter" das Abspielen eines zufälligen „One Shots" auszulösen, der eine Aufnahme von Lachen beinhaltet. Solange diese Audiodatei abspielt, ist kein erneutes Auslösen möglich.
Bereits drei Takte nach Beginn der Phase wird ein Fade eingeleitet, der zu den übrigen Audiosignalen das Stereosignal des Subpatches „watch_your_back" hinzumischt, des Herzstücks der Klangsynthese in Phase02.
Innerhalb des Subpatches „watch_your_back" wird Klang durch subtraktive Synthese erstellt, jedoch auf eine experimentelle Art und Weise. Einerseits wirken Positionsdaten auf die Verwendung von Oszillatoren und andere Parameter (siehe „6.2.2 Synapse Kinect controls and routing" auf Seite 27), andererseits sind auch viele zufällig generierte Werte im Spiel.

Alle vier Takte wird der Notenwert Bb in einer zufälligen Oktavlage ausgewählt und die Frequenz der Oszillatoren neu gesetzt. Es handelt sich hierbei um die spektralreichen Signale rand~, vs.rand0~ (ein External der Virtual Sound Library, CIPRIANI & GIRI 2012), rect~ und tri~. Letztere zwei werden von einer Sinusschwingung mit zufälliger rhythmusgebundener Frequenz über ihren Duty Cycle moduliert.
Die beiden Signaltypen werden jeweils gemischt und schließlich entsprechend der Kopfposition untereinander gemischt. Das Ausgangssignal wird nun von zwei biquad~-Filtern, die sich in einem cascade~-Objekt befinden, gefiltert und zum ursprünglichen Ausgangssignal gemischt, nachdem wiederum ein anderer Algorithmus das Signal mittels tapin~- und tapout~-Objekten bereits in der Tonhöhe veränderte und zum gefilterten Signal mischte.
Die Filterung geschieht nicht nach herkömmlichen Gesichtspunkten. Ziel ist es, unvorhergesehene und störende Klänge zu erzeugen. Deshalb ist die Cutoff-Frequenz zwar die des ursprünglichen Signals, aber der Gain-Wert kann zufallsbedingt knapp unter Null fallen, was den Filter dazu veranlasst, bestimmte Frequenzen sehr stark zu betonen. Der Gain-Wert befindet sich in einer stetigen rhythmusgebundenen Veränderung, was durch die Kombination zweier random-Objekte mit einem line-Objekt realisiert wurde.

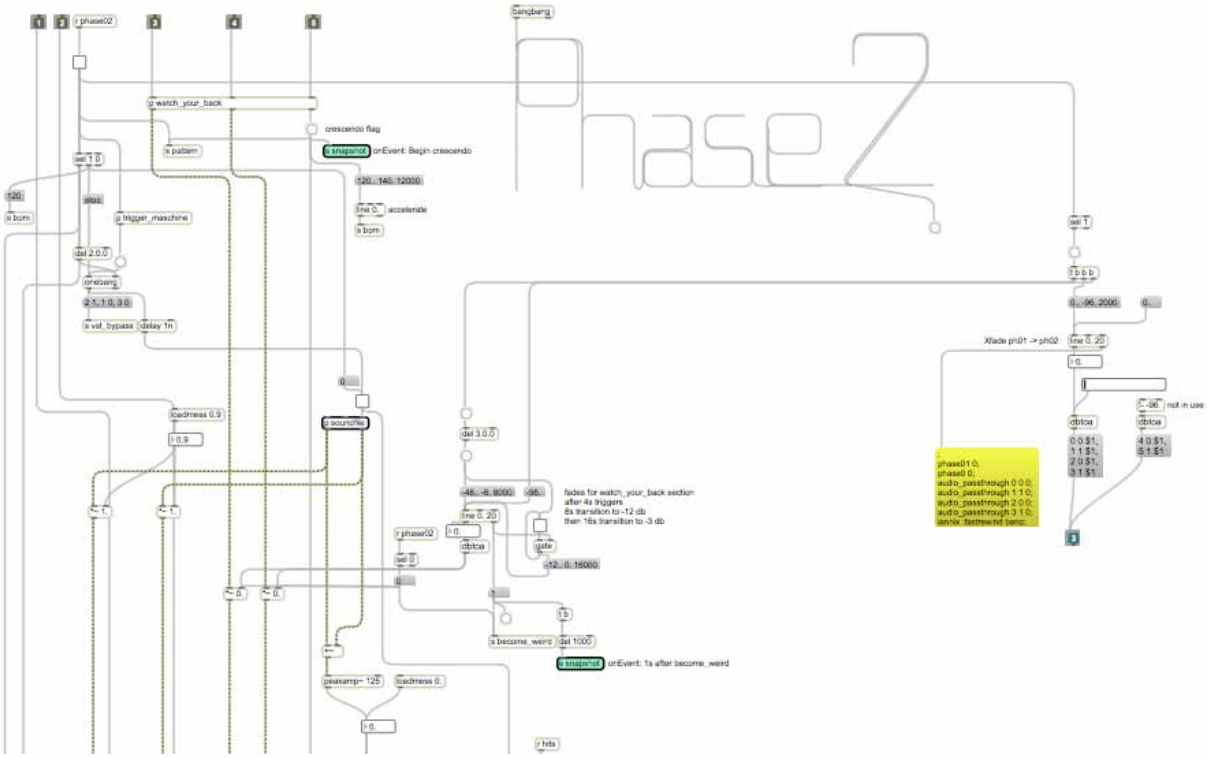


Abb. 8: Ausschnitt aus „p02_sp"

Zu dem bereits genannten Filter kommt ein zweiter Filter, der zum Zweck hat, bei Joint-Hits durch schnelle Bewegungen der Person ein Feedback zu geben. Dadurch

wird klarer, dass eine Interaktion stattfindet. Ein Joint-Hit regt ein line-Objekt an, das einen schnellen Filter-Swipe von 20 Hz bis 20 kHz über das Signal in 400 ms ausführt.
Die Amplitude des gefilterten und in der Tonhöhe veränderten Signals wird jetzt je nach algomethrischem Mittel der Handstellungen beider Hände mit einer Zahl zwischen 0,7 und 1 multipliziert. Anschließend wird das Signal von einem clip~-Objekt entsprechen der Kniehaltung des Besuchers verzerrt, was anhand einiger grafischer Objekte anschaulich gemacht wird.
Dann wird das Signal durch einen Hall-Algorithmus geschleust, dessen Ausgabe zum Ausgangssignal gemischt wird. Der Hall-Algorithmus stammt aus den Max/MSP Beispieldateien und wurde unverändert übernommen.
Schließlich wird das Signal von einem modulierten LFO unregelmäßig nach links oder rechts gepannt und aus dem Subpatch „watch_your_back" ausgegeben.
Es geschehen mehrere Parameterveränderungen über die Zeit, die von dem Objekt „become_weird" und bei Erreichen des Takts 38 angestoßen werden. „become_weird" wird ausgelöst, nachdem das „watch_your_back"-Signal hinzugemischt wurde und ist selbst für das Zumischen des gefilterten Signals zuständig. Nach Takt 38 wird ein Crescendo durch viele verschiedene Parameter ausgelöst. Das Pattern des Softwareinstruments Maschine wird gewechselt, sodass mehr Elemente im Takt vorhanden sind, das Tempo wird von 120 auf 140 BPM gesteigert, die Frequenz der Signale und die Cutoff-Frequenz des Filters wird kontinuierlich erhöht, das gefilterte Signal wird kontinuierlich mehr nach oben gepitcht und außerdem wird im Algorithmus, der den Gain-Wert des Filters kontrolliert, langsame Rhythmen ausgeschlossen.
Kurz vor Ende des zwölfsekündigen Finale wird eine Audiodatei eines Schreis abgespielt und durch einen Crossfade in den Vordergrund vor alle anderen Signale gerückt. Das Ende dieser Audiodatei löst die nächste Phase aus.
An mehreren Stellen innerhalb dieses Patches wird ein Befehl für einen Snapshot an die Video Control Sektion gesendet.

Phase03 – „p03_sp"

Im Command Center wird bei Auslösen der Phase03 erst nach 4,5 Sekunden das Routing für das Subpatch „p03_sp" an das matrix~-Objekt gesendet. Dann beginnt die Sequenz mit dem Senden von Midi-Events und dem Einfaden der Synthesizer Massive und Razor.
Es gibt insgesamt drei Eingabeparameter: „phase03_sp" empfängt die hochzählende Variable aus dem Subpatch „two_hands_alternate", „phase03_tri" ein „bang" bei Auslösen des Inkrements ebenda und „phase03_pitch" die Y-Position der Hand, die gerade aktiv ist.

Über das Objekt change mit dem Attribut „+" oder „-" kann erfasst werden, ob eine kontinuierliche Aufwärtsbewegung stattfindet. Wird die Zahl größer, so zählt ein counter-Objekt hoch und verändert damit einen Parameter des Razor-Synthesizers. Sobald eine niedrigere Zahl als die vorangehende erfasst wird, setzt ein Befehl den counter wieder auf Null.
Bei jedem Inkrementieren der Variable aus „hands_up_alternate" wird ein entsprechend höherer Ton gespielt und ein Parameter des Softwareinstrumentes verändert, der das Timbre betrifft. Außerdem wird über den Eingabeparameter „phase03_tri" ein „One Shot" eines bearbeiteten Triangelklangs abgespielt, der die Veränderung verdeutlicht und bestätigt, dass ein Inkrementieren stattgefunden hat. Wird der Wert der Variable gleich Eins, so startet ein Ausfaden der Synthesizer und ein Abspielen eines Bestätigungsklangs in einer Audiodatei.
Für Testzwecke lässt sich das Inkrementieren auch durch Druck auf die Taste „y" auslösen.

Phase04 – „p04_sp"

Das Subpatch „p04_sp" hat Ähnlichkeiten zu „p03_sp", da auch hier eine Instanz des Softwareinstrument Razors gesteuert wird. Dass nicht dieselbe Instanz verwendet wird, liegt daran, dass Razor keine Parameter-Presets speichern kann, sondern nur das gesamte Reaktor-Ensemble. Dieses Ensemble lässt sich zwar auch von Max/MSP ändern, benötigt aber ca. 2 Sekunden Ladezeit. So ist es das kleinere Übel, mehr Daten beim Starten des Programms zu laden.
Bei Aktivierung der Phase und nach einem Delay von sechs Sekunden wird das transport-Objekt mit einer hohen Taktrate von 240 BPM gestartet. Diese Rate wird innerhalb der ersten 24 Sekunden kontinuierlich in einem Kurvenverlauf auf 120 BPM gesenkt. Der Kurvenverlauf bewirkt, dass die Taktrate nur in den ersten paar Sekunden sehr hoch ist.
Die Verzögerung ist nötig, da erst der Bestätigungsklang von Phase03 wahrnehmbar sein muss. Nachdem das del-Objekt einen „bang" sendet, wird einen Ton im Synthesizer ausgelöst. Ab jetzt beginnt ein Algorithmus zu wirken, dessen Hauptelemente zwei verschachtelte metro-Objekte sind. Das erste metro-Objekt sorgt dafür, dass alle zwei Takte ein Parameterwert im Synthesizer neu gesetzt wird und gibt über ein itable-Objekt einen gewichteten zufälligen Notenwert an das zweite metro-Objekt weiter. Ist dieser Notenwert außerdem höher als 4n, so hat er auch Einfluss auf Koeffizienten im Subpatch „dirty_signal" in der Video Control Sektion.
Das zweite metro-Objekt gibt aus einem weiteren itable-Objekt eine gewichtete Zufallszahl zwischen Null und Fünf oder die Zahl Zwölf aus. Diese Zahl wird zufällig mit Eins oder minus Eins multipliziert, zum Midi-Grundwert 34 (Bb1) addiert und an

den Synthesizer gesendet. Dieser Algorithmus kann auch durch Joint-Hits ausgelöst werden.
Außerdem wird durch diese Aktion in der Video Controls Sektion ein Snapshot in die Jitter Matrix geladen und im Subpatch „phase04_sequencer" der Takt von neuem begonnen. In der aktuellen Implementierung kann der Sequencer nicht vollständig genutzt werden, da weder mit den Plugins Maschine, Battery 3 oder mit dem aktuell verwendeten Kontakt 4, noch durch das manuelle Auslösen von Samples mit dem detune-Objekt eine ausreichende Genauigkeit erreicht werden kann. Dazu ist die Latenz der gesendeten Befehle bei der Verwendung des Programms mit dem Testcomputer zu hoch. Es werden in der aktuellen Implementierung nur die Anfangsschläge des Rhythmus gespielt.
Eine Besonderheit in diesem Patch ist die Verwendung einer separaten matrix~ für das Routing zwischen Rhythmus, Synthesizer und Effekt. Bei Eintritt in Takt 19 wird ein Crossfade durchgeführt, der das Signal des Synthesizers über vier Takte mehr und mehr über das Effektgerät Absynth 5 ausgibt. Mit einer zweitaktigen Verzögerung werden über sechs Takte die Effektparameter verändert. Ist dieser Prozess abgeschlossen, ist der Lebenszyklus von Maskenrad beendet und das System beginnt einen Wartestatus, bis der Besucher von der Platte tritt und damit einen System-Reset auslöst, der die Phase00 aktiviert.

6.3 Synapse

Synapse ist ein Open Source Programm, das die Position verschiedener Körperteile wie Hände, Torso oder Kopf im dreidimensionalen Raum in Abhängigkeit zum erfassten Raum oder zur Grundposition des Körpers erkennt und als OSC-Daten zur Verfügung stellt. Diese Daten können dann für die Auslösung von Prozessen oder zur Steuerung von Audioparametern oder MIDI verwendet werden.
Für die Kalibrierung der Kamera muss der Benutzer eine Weile warten und sich bewegen. Am Besten funktioniert die Erkennung der Person, wenn die die Arme dabei hebt.

6.4 IanniX

Der Open-Source-Grafiksequencer Iannix wird in der vorliegenden Arbeit verwendet, um Servomotoren in sich verändernden Wellen pulsieren zu lassen und um Klangerzeugung auszulösen. Basierend auf der Arbeit von Iannis Xenakis sendet IanniX die generierten OSC-Daten an Echtzeit-Entwicklungsumgebungen wie Max/MSP (IANNIX 2012).

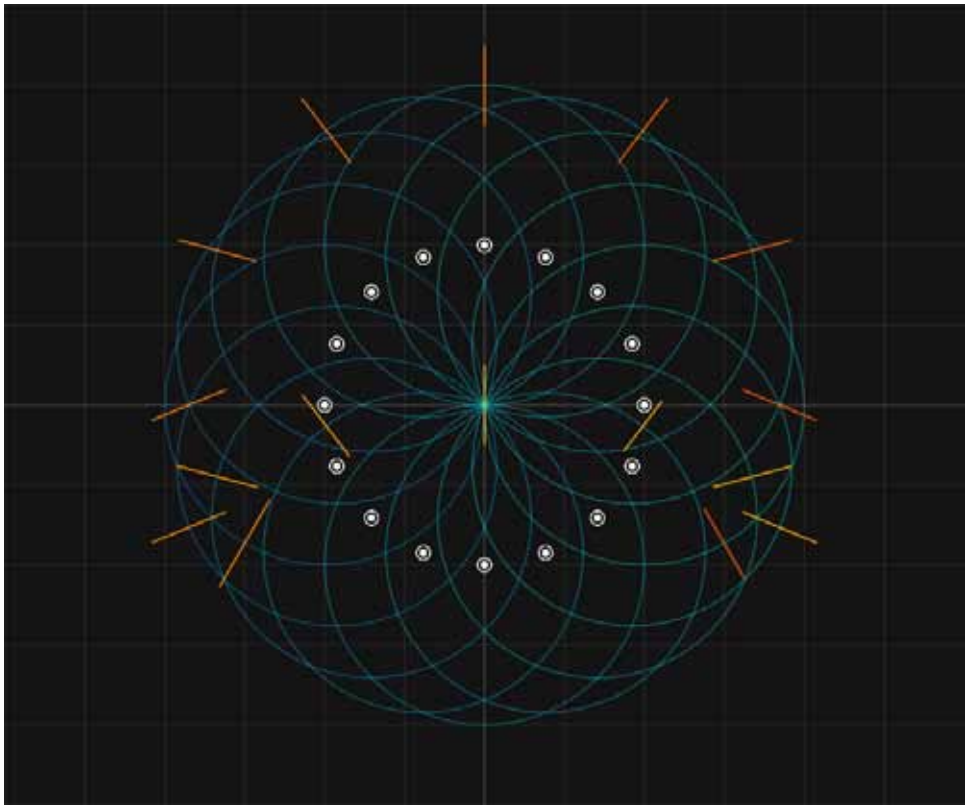


Abb. 9: Abbildung eines IanniX-Scores

6.5 Arduino

Arduino bezeichnet gleichermaßen eine Serie von Open-Source-Microcontrollern sowie die zugehörige Entwicklungsumgebung, die auf Processing basiert und eine auf Wiring basierende Programmiersprache voraussetzt (Fry *et al.* 2012, Barragán *et al.* 2000, Banzi *et al.* 2012).
Es werden zwei Arduino Microcontroller simultan eingesetzt. Der Grund hierfür ist die Ansteuerung der Texas Instruments TLC5940 Chips. Zu Beginn war es geplant, nur ein Board zu verwenden, was aber daran scheiterte, dass die tlc5940 Library die Taktfrequenz auf 50Hz drosselt, sobald man ein TLC5940 für die Verwendung mit Servomotoren konfiguriert (Leone 2010, Texas Instruments 2012). Da dieses Setup nicht für die Verwendung mit LEDs gewünscht war, entschied sich der Autor für die Alternative mit zwei Microcontrollern – einen für die LEDs und einen für die Servomotoren. Im späteren Verlauf wurde der TLC5940, der die Servomotoren ansteuerte, wegrationalisiert, und die Servomotoren direkt an ein

Abb. 10: Arduino Logo

Arduino Mega 2560 angeschlossen. So wurde die Stabilität des Programms verbessert, da die tlc5940 erst seit kurzem auch für Servomotoren verwendet werden kann und in eigenen Testreihen unbrauchbare Ergebnisse lieferte.
An einem analogen Pin des Arduino Mega 2560 wird ein Sensorwert gelesen und über die serielle Schnittstelle an Max 6 gesendet.

6.5.1 Code

Der Arduino Code für LED- und Servomotorenansteuerung ist relativ unterschiedlich. Die serielle Kommunikation mit den LEDs erfolgt über eine festgelegte Nachrichtengröße. Das erste Byte ist der „message header", über den die Nachricht erkannt wird und dann vom Microcontroller verarbeitet wird. Die Idee dazu stammt von Scott Fitzgerald (FITZGERALD 2012).
Entsprechend des Ratschlags von Andreas Muxel (Persönliche Kommunikation, 21.03.2012) ist die serielle Kommunikation mit dem Microcontroller, der die Servomotoren ansteuert, mit der freien Bibliothek Messenger realisiert (FREDERICKS 2008). Diese Bibliothek stellt mithilfe eines nach der Nachricht gesendeten Wagenrücklauf-Steuerzeichens (im speziellen Fall der Wert 13 im ASCII-Zeichensatz) die Vollständigkeit einer Nachricht fest. So kann eine serielle Nachricht unbestimmter Länge gesendet werden, die im Microcontroller zwischengespeichert und erst bei Empfangen des Wagenrücklaufs ausgeführt wird.
In beiden Verfahren wird jedoch die Nachricht in ein Array geschrieben und in der loop()-Sektion des Programmes auf die Hardware angewendet, indem die write-Funktion des Servo-Objektes oder die update-Funktion des TLC-Objektes aufgerufen wird.

7 Zusammenfassung

Im Gespräch mit Personen, die die Installation Maskenrad ausprobiert und erlebt haben, stellte sich heraus, dass sie tatsächlich von der Reaktion der Maschine überrascht waren und sich gerne länger mit der Installation beschäftigt hätten „um die Intrige besser spüren zu können" – so ein dem Autor unbekannter Student der Hochschule Ansbach.
Daraus lässt sich ableiten, dass das Konzept der Intrige auf Maschinen übertragbar ist und einige Besucher der Installation zum Denken angeregt werden.

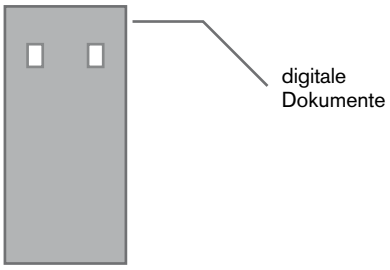
Allerdings ist das Projekt Maskenrad noch lange kein perfektes Produkt, wenn auch schon kein Prototyp mehr.

7.1 Weiterführende Entwicklungen

Die Installation Maskenrad ist in einem Beta-Stadium. Das bedeutet, dass alle Funktionen bereits stabil funktionieren. Allerdings gibt es noch denkbare Verbesserungen und Erweiterungen, die bis zum jetzigen Zeitpunkt noch nicht realisiert werden konnten.
Maskenrad wird in der aktuellen Version über ein MacBook Pro Late 2008 gesteuert, das mit gleichzeitiger Kameratracking, Klangsynthese, Videobearbeitung und serieller Kommunikation schon nahe an seine Grenzen kommt. Es kann vorkommen, dass Serielle Befehle mit einer Latenz von bis zu 400 ms gesendet werden. Für einen praktischen Einsatz auf einer Ausstellung muss also mehr Rechenleistung zur Verfügung gestellt werden.
Die Software Synapse for Kinect ist bisher erst in der Version 1.1 verfügbar (SYNAPSE 2011). Es gibt noch nennenswerte Bugs: Synapse kann bei Performanceschwäche die Reaktion verweigern und dadurch sogar Max/MSP zum Absturz bringen. Es ist leider keine ausreichende Fehlerbehandlung implementiert und die Entwicklung scheint seit Mitte 2011 nicht voranzugehen. Mit der voranschreitenden Technologie wird hoffentlich in Zukunft die Erkennungszeit für die Kalibrierung des Kameratrackings reduziert werden. Das würde die Technik mehr in den Hintergrund rücken und den Fokus mehr auf Klang, Licht und Bewegung lenken.
Eine weitere mögliche Weiterentwicklung ist das erweitern des Stereoklangs auf einen Surround-Klang. Das Sound Design könnte so noch mehr auf die Bewegungen des Protagonisten abgestimmt werden, da so die Z-Achse der Gestiken auch klanglich direkt umzusetzen wäre. Realisieren ließ sich diese Idee aus zeitlichen und finanziellen Gründen noch nicht.

8 Anlagen

8.1 Digitale Dokumente



Dieser USB-Stick enthält das Max/MSP Patch sowie alle nötigen Externals, um das Perogramm zu starten. Auch eine Kopie von IanniX, Synapse und der Arduino Code ist vorhanden. Nicht enthalten sind die kostenpflichtige Software Max/MSP und die Softwareinstrumente von Native Instruments.
Die bereitgestellte Software ist für das Betriebssystem Mac OS X 10.6 Snow Leopard oder neuer bestimmt.

Bitte gehen Sie wie folgt vor, um das Programm lauffähig zu machen:

1. Kopieren Sie den Ordner maskenrad_externals in den Ordner Applications/Max6/patches/externals. Vermeiden Sie Duplikate.
2. Installieren Sie IanniX
3. Kopieren Sie die Datei maskenrad.nxscore in das Verzeichnis User/Documents/IanniX
4. Installieren Sie Synapse
5. Verbinden Sie Ihren Computer mit einer Microsoft Kinect Kamera
6. Öffnen Sie Synapse
7. Öffnen Sie IanniX
8. Wählen Sie den msakenrad Score aus der Liste rechts oben aus
9. Öffnen Sie die Datei maskenrad.maxpat

Da Maskenrad eine Installation ist und ohne die entsprechende Hardware nicht funktioniert, sollen die vorhandenen Daten in einem akademischen Kontext betrachtet werden. Sie können Inspiration und Ideen geben oder Techniken aufzeigen.

8.2 Anlage A: Arduino Code für die Steuerung der TLCs

```
//Include TLC5940 Library by Alex Leone
#include <Tlc5940.h>
// defines a message header, that is there to identify the incoming message
#define MSG_HEADER       255
// this message length consists of 3x16 data bits for the TLCs + the message header
#define MSG_LEN          49

// this array serves as storage structure for the serial data
unsigned int PWMData[]=
{ 0,0,0, 0,0,0, 0,0,0, 0,0,0,
  0,0,0, 0,0,0, 0,0,0, 0,0,0,
  0,0,0, 0,0,0, 0,0,0, 0,0,0,
  0,0,0, 0,0,0, 0,0,0, 0,0,0 };

void setup()
{
  // setup the Serial connection at a baud rate of 57800 (empiric value)
  Serial.begin(57600);
  Tlc.init(0); // initialize the TLCs
}

/*
*   readSerial() Function
*   looks up serial messages, writes the data into the storage array and calls the TLC
function.
*/
void readSerial()
{
  boolean serialMessage = false;
  // check for serial message header and serial availability
  if(Serial.available() && (Serial.read()==MSG_HEADER)) {
    // if message header is received write whole message into the array
    while(Serial.available()<MSG_LEN);
    for(int i=0;i<MSG_LEN-1;i++){
      PWMData[i]=Serial.read();
      serialMessage=true;
    }
  }
  // calls the TLC function if there is some serial message
  if (serialMessage==true){
    DoTLC();
  }
  else{
    Serial.flush();
  }
}

/*
*   DoTLC() Function
*   sets the TLC data it gets from the array
*/
void DoTLC(){
  for(int i =0;i<MSG_LEN-1;i++){
    int LEDval=PWMData[i]*16;
    Tlc.set(i, LEDval);
  }
}

void loop()
{
  readSerial(); // call function to handle serial messages
  Tlc.update(); // send the prepared data to the TLCs
}
```

8.3 Anlage B: Arduino Code für die Servosteuerung

```
//*********************************************************
// LIBRARIES
#include <Messenger.h>
#include <Servo.h>

//*********************************************************
// GLOBALS

#define numberServos 16  // total number of controlled leds
unsigned int valuesServos[numberServos]; // values for servos

Servo servo01;  // create servo object to control a servo
Servo servo02;
Servo servo03;
Servo servo04;
Servo servo05;
Servo servo06;
Servo servo07;
Servo servo08;
Servo servo09;
Servo servo10;
Servo servo11;
Servo servo12;
Servo servo13;
Servo servo14;
Servo servo15;
Servo servo16;

int forceSensor = 0;

Messenger message = Messenger();

//*********************************************************
//*********************************************************
// SETUP

void setup() {

  servo01.attach(22);  // attaches the servo on pin 22 to the servo object
  servo02.attach(23);
  servo03.attach(24);
  servo04.attach(25);

  servo05.attach(26);
  servo06.attach(27);
  servo07.attach(28);
  servo08.attach(29);

  servo09.attach(30);
  servo10.attach(31);
  servo11.attach(32);
  servo12.attach(33);

  servo13.attach(34);
  servo14.attach(35);
  servo15.attach(36);
  servo16.attach(37);

  // init serial connection
  Serial.begin(115200);

  // set all servos to zero
```

```
for(int i=0; i<numberServos; i++){
    valuesServos[i] = 0;
  }

}

//*********************************************************
//*********************************************************
// LOOP

void loop() {
  // check serial port
  checkSerialPort();

  // drive servos
   for(int i=0; i<numberServos; i++){
   servo01.write(valuesServos[0]);
   servo02.write(valuesServos[1]);
   servo03.write(valuesServos[2]);
   servo04.write(valuesServos[3]);
   servo05.write(valuesServos[4]);
   servo06.write(valuesServos[5]);
   servo07.write(valuesServos[6]);
   servo08.write(valuesServos[7]);
   servo09.write(valuesServos[8]);
   servo10.write(valuesServos[9]);
   servo11.write(valuesServos[10]);
   servo12.write(valuesServos[11]);
   servo13.write(valuesServos[12]);
   servo14.write(valuesServos[13]);
   servo15.write(valuesServos[14]);
   servo16.write(valuesServos[15]);
   }

   forceSensor = analogRead(A0)/4;
   Serial.write(forceSensor);
}

//*********************************************************
// SERIAL

void checkSerialPort(){
  //delay(30);
  while (Serial.available() ){
    if ( message.process(Serial.read()) ){
      // get values for servos
      for(int i=0; i<numberServos; i++){
        valuesServos[i] = message.readInt();
      }
    }
  }
}
```

9 Literaturverzeichnis

Banzi, M., Cuartielles, D., Igoe, T., Martino, G., Mellis, D. et. al. (2012). Arduino. Abgerufen am 15.05.2012 von http://www.ardiuino.cc/

Barragán, H., Hagman, *B., Brevig, A. e***t al.** (2012). Wiring. Abgerufen am 15.05.2012 von http://wiring.org.co/

Beck, B. (2011). Intrigenopfer: *Vom Aufstieg und Fall großer* Männer. Wiesbaden: marixverlag GmbH.

Boland, H. (2012). Fr*eesound.org* – HerbertBoland. Abgerufen am 21.06.2012 von http://www.freesound.org/people/HerbertBoland/

Cipriani, A. & Giri, *M. (2010). Electronic Music and Sound Design: Theory and* Practice with Max/MSP volume 1. Rom: ConTempoNet s.a.s.

club sound. (2012). Freesound.or*g – club sound. Abgerufen am 21.06.2012 von http://www.freesound.org/people/*club%20sound/

Cycling'74. (2012). W*hat is Max?. Abgerufen am 20.06.2012 von http://cycling74.com/whatismax/*

Donnarumma, M. (2*011). Xth Sense | Res, a matter. Abgerufen am 18.06.*2012 von http://res.marcodonnarumma.com/projects/xth-sense/#biophysical-music

Fermont, C. (2012). Freesound.org *- cdrk. Abgerufen am 21.06.2012 von http://ww*w.freesound.org/people/cdrk/

Fitzgerald, S. (20*12). Arduino + LED Workshop. Abgerufen am 15.05.2012 von http://www.ennuigo.com/sharing/Arduino-LEDs-and-DMX-Workshop-2/*

Fredericks, T. O. (2008). Messenger library for Arduino. Abgerufen am 15.05.2012 von http://www.arduino.cc/playground/Code/Messenger

Fry, B., Reas, C. et al. (2012). Processing. Abgerufen am 15.05.2012 von http://www.processing.org/

Godøy, R. I. & *Leman, M. (2010). Musical Gestures: Sound, Movement, And Meaning. Ab*ingdon: Routledge.

IanniX. (2012). Ia*nniX: A graphical real-time open-source sequencer for digital art. Abgeru*fen am 14.05.2012 von http://www.iannix.org/

Leone, A. (2010). tlc5940arduino: An Arduino Library for the TI TLC5940 16-Cha*nnel PW*M Chip. Abgerufen am 15.05.2012 von http://code.google.com/p/tlc5940arduino/

Matt, P. von (2009). Die Intrige: Theorie und Praxis *der H*interlist (2. Aufl.). München: Deutscher Taschenbuch Verlag GmbH & Co. KG.

Michalik, R. (2011). Intrige: Mac*htspiele* - wie sie funktionieren - wie man sie durchschaut - was man dagegen tun kann. Berlin: Econ.

Nicolai, C. und Emmerich, C*. (2010). ANBB - Alva Noto und Blixa Bargeld - mimikry: Bergha*in. Released auf Raster-Noton.

Owsinski, B. (2007). M*ischen wie die Profis:* Das Handbuch für Toningenieure (2. Aufl.). München: GC Carstensen Verlag.

Ritter, D. (1993). Intersec*tion. Abgerufen am 14.05.201*2 von http://www.aesthetic-machinery.com/intersection.html

superbrightleds.com (2012). V*ollong 3W RGB High* Power LED. Abgerufen am 20.06.2012 von http://www.superbrightleds.com/moreinfo/component-leds/vollong-3w-rgb-high-power-led/899/2214/

SYNAPSE. (2011). SYNAP*SE for Kinect.* Abgerufen am 14.05.2012 von http://synapsekinect.tumblr.com

Texas Instruments. (2012). Lighting and Display Solutions – LED Driver – TLC5940 – TI.com. Abgerufen am 15.05.2012 von http://www.ti.com/product/tlc5940

United Visual Arti*sts. (2011).* Rien a Cacher Rien a Craindre. Paris. Abgerufen am 14.05.2012 von http://www.uva.co.uk/work/rien-a-cacher-rien-a-craindre-3

Vandoren, D. (2011). *Integration.03. Abgerufen am 14.05.2012 von htt*p://www.dietervandoren.net/index.php?/project/integration03/

Viers, R. (2008). The Sound Effects Bible: How To Create And Record Hollywood Style Sound Effects. Studio City: Michael Wiese Productions.

Visual System. (2009). *Organic Culture. Nantes.* Abgerufen am 18.06.2012 von http://www.visualsystem.org/ORGANIC-CULTURE

Weiss, F. (2010). *EyeCon Support and Download Page. Abge*rufen am 18.06.2012 von http://eyecon.palindrome.de/

Wollniok, D. (2011*). +50° 36‘ 16.27“, +11° 34‘ 33*.45“ Drachenschwanz. Weimar. Abgerufen am 18.06.2012 von http://www.uni-weimar.de/cms/medien/experimentelles-radio/studentische-arbeiten/drachenschwanz.html

Woodbury, L. (2011). Post in Cycling ‚74 Forums. Abgerufen am 15.05.2012 von http://cycling74.com/forums/topic.php?id=35683#position-7

YesYesNo. (2010). Night Lights. Auckland. Abgerufen am 18.06.2012 von http://yesyesno.com/night-lights

10 Erklärung

Ich versichere, dass ich die Arbeit selbständig angefertigt, nicht anderweitig für Prüfungszwecke vorgelegt, alle benützten Quellen und Hilfsmittel angegeben sowie wörtliche und sinngemäße Zitate gekennzeichnet habe.

Ort, Datum

Julian Vogels