Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Studienarbeit

Martin Sukale

Computergestützte Kunstprojekte - Neuere technologische Entwicklungen

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Martin Sukale

Computergestützte Kunstprojekte - Neuere technologische Entwicklungen

im Studiengang Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Kai v. Luck

Abgegeben am 9. April 2008

Martin Sukale

Thema der Studienarbeit
Computergestützte Kunstprojekte - Neuere technologische Entwicklungen
Stichworte
Kunst und Computer, Open Source Multimedia
Kurzzusammenfassung
Analyse von computergestützten Kunstprojekten mit dem Schwerpunkt auf Freie Software und offene Standards.

Martin Sukale

Title of the paper
Computerbased art projects - recent technological progress
Keywords
Computer And Art
Abstract
Analysing computerbased art projects with focus on Free Software and open standards.

Inhaltsverzeichnis

1 What I Want

1.1 Motivation

Durch Gespräche und Arbeiten mit Kunstschaffenden, Veranstaltern, Ausstellungsmachern und Architekten ist mir der Bedarf an einer interdisziplinären Zusammenarbeit zwischen (technischen) Informatikern und der ”kreativen Ecke” deutlich geworden. Es gibt in der Kunstszene eine große Neugier an technisch raffinierten Produkten bzw. Installationen. Auf dem Markt erhältliche Hard- und Software aus den Bereichen Consumer-Electronics, Veranstaltungstechnik, Gebäudetechnik stellen meist nicht die gewünschten Funktionen zur Verfügung. Welche Eigenschaften werden benötigt? Wo sind die Schnittstellen zwischen industriellen Anwendungen der technischen Informatik und Kunst?

1.2 Kunst und Computer

Bei den Vorüberlegungen zu dieser Arbeit faszinierte mich die Möglichkeit ein Kunstwerk mit Hilfe moderner Computertechnologie vollständig autark zu erschaffen. Als Kunstschaffender selbstbestimmt eine Idee umzusetzen ohne Vorgaben von außen. Dieser künstlerische Ansatz lässt sich virtuell in der geschlossenen Welt eines Computersystems zumindest theoretisch umsetzen:

Prinzipiell könnte ein Künstler alleine durch die Beherrschung von Programmiersprachen, Programmiertechniken und Algorithmen ein Werk entstehen lassen, welches hundertprozentig seinen Idealen entspringt: Mit dem Wissen um Klangsynthese, ausgehend von ein paar Zeilen Code (z.B. einer Sinus-Funktion) kann ein Musikinstrument programmiert und gespielt werden. Dieses ließe sich ohne ein reelles Instrument je spielen gelernt zu haben oder auf materiellen Besitz des selben angewiesen zu sein sogar soweit fortspinnen, dass ein ganzes Orchester von Fantasie-Instrumenten oder synthetisierten realen Instrumenten alleine durch die Programmierleistung des Künstlers bzw. der künstlerischen Leistung des Programmierers - welche hier zu einer Person verschmelzen - erschaffen werden könnte. Dieser Ansatz war Motivation der Erfinder des Synthesizers - man glaubte, der Synthesizer würde eines Tages Orchester und Musiker überflüssig machen - und ist teilweise schon umgesetzt worden von einigen Softwarefirmen. Als Beispiel sei hier die Reihe von virtuellen physikalisch modellierten VST-Instrumenten [1] zu nennen, die z.B. die Nachbildung einer

[1] Virtual Studio Technology der Firma Steinberg Media Technologies, Hamburg

Hammond-Orgel, eines Klaviers oder Gitarre ermöglichen. Forschungen auf diesem Gebiet finden derzeit an der HAW-Hamburg unter der Leitung von Wolfgang Fohl statt (vgl. Jagdmann (2005)).

Abbildung 1: Oberfläche der virtuellen Hammond Orgel B4 ©2000-2008 Native Instruments

Analog hierzu ließe sich ein Bild malen, wobei - ausgehend vom Wissen um Farbsynthese und der Vorstellungskraft des Künstlers - ohne die Investitionskosten für Ölfarbe und Leinwand eine virtuelle Leinwand bemalt bzw. programmiert wird und beispielsweise auch in virtuellen Galerien ausgestellt werden könnte. Forschungen auf diesem Gebiet der Weiterentwicklung klassischer Malerei mit modernen elektronischen Medien finden z.B. unter der Leitung von Daniel Hausig an der HBK-Saar statt (Malerei und Intermedia). Selbstverständlich ließen sich auch dreidimensionale Skulpturen erschaffen. Der virtuelle Bildhauer ist als 3D-CAD-Grafiker bereits Realität. Letztere Varianten müssen uns schon alltäglich vorkommen, da bereits sämtliche Aspekte des öffentlichen Lebens und somit auch Kunst in 3D-Spielewelten virtualisiert wurden. Während früher bereits die Kopie eines Bildes durch Drucktechniken oder Fotografie bzw. die Konservierung und Vervielfältigung von musikalischen Aufführungen zu Zweifeln an der Originalität und dem künstlerischen Wert in der Kunstszene führte (vgl. Benjamin (1981)), ruft heute die Erschaffung von digitalen virtuellen Kunstwerken kaum mehr Erstaunen hervor. Bereits in den 1980er Jahren (die Zeit der 8bit-Rechner, Commodore C64, Amiga500 etc.) entstand eine Spielart digitaler Kunst: Eine Szene von Programmierern (auch ”demoscene” genannt, vgl. Wikipedia (2008)) die sich mit der Komposition multimedialer Kunstwerke beschäftigte. Unter gewissen selbst auferlegten und durch die damalige Hardware gegebenen Beschränkungen, z.B. limitiert auf 4kB RAM wurden effektvolle Grafiken und Klänge kreiert und auf Veranstaltungen und im Netzwerk vorgestellt.

1.3 Do-It-Yourself

Dieses 4kB-Dogma (oder entsprechende Disziplinen) zwingt die Künstler bzw. Programmierer in der geschlossenen Welt von 4096 Bytes und vordefinierten Hardware-Restriktionen, hundertprozentig selbsttätig aus Programmcode und Algorithmen ein Kunstwerk zu kreieren. Hier sind große ingenieurstechnische und künstlerische Leistungen vollbracht worden, die zur Schaffung von etwas Neuem, eines eigenständigen Genres führten. Analog hier zu seien afroamerikanische Musiker genannt, die in den 1920er Jahren mangels teurer Musikinstrumente auf sog. Jugs (engl. Kanne, Krug), Kazoos (”professionelle Blaskämme”) oder Broomsticks (Wassereimer, Besenstiel, Stahlseil) wesentlich die Blues und Jazz-Szene prägten. Diese Do-It-Yourself-Mentalität - ob aufgezwungen oder selbst auferlegt - trägt meiner Meinung nach viel zur Entstehung von Kunst und der künstlerischen Freiheit bei. Nun kann man aber nicht - wie von einem professionellem “Hacker” oder “Coder” (vgl. Wikipedia (2008)) - von Kunststudenten erwarten, in Assembler zunächst ein Betriebssystem und einen Compiler zu schreiben, um dann darauf aufbauend einen Grafiktreiber und Routinen zur Ausgabe von farbigen Pixeln zu entwickeln. Genauso wenig wie man von Malern erwarten kann, Indigopflanzen zu züchten, Wasserbrunnen zu bohren und Leinen zu weben, um sich die künstlerische Freiheit zu bewahren. Auch wenn das Selbst-Löten von Synthesizern unter Elektro-Musik-Künstlern evtl. noch Anerkennung (und selbstverständlich Frust) einbringt, so wird hier doch etwas grundlegendes deutlich:

Im Zeitalter der Technologisierung der Umwelt fehlen dem Menschen allgemein das Wissen und die Mittel, um die Bestimmungsgewalt über diese wiederzuerlangen: Ubiquitus Computing soll dem Menschen zwar mehr Freiheit geben, nimmt ihm aber die Chance über sein Leben selbst zu bestimmen: Telefon, Fernseher, Musikspieler und globale Netze diktieren dem Anwender sein Konsum- und Kommunikationsverhalten sowie im schlimmsten Fall seine Emotionen (z.B. elektronisches Kinderspielzeug, farbige LED-Stimmungsleuchten). Die Art und Weise unsere Umwelt zu begreifen ist durch Technik geprägt und fordert auf der Seite der Kunst zu kritischer Auseinandersetzung heraus:

Es kann zur Aufgabe der Kunstschaffenden werden, dem Menschen zu einem bewussteren Umgang mit der Technik zu verhelfen. Voraussetzung hierzu ist, dass der Künstler über eben jene Technologie frei verfügen kann. Als Informatiker kann ich hier meinen Beitrag leisten indem ich mein Know-How nicht in die Entwicklungsabteilung eines Consumer-Electronic-Konzerns oder Rüstungsproduzenten einfließen lasse, sondern Konzepte für z.B. Kunststudenten entwickele um diesen besseren Zugang zur Computertechnik zu ermöglichen. Offene Systeme spielen dabei eine entscheidende Rolle.

1.4 Open Source

Während die Ansichten von Richard Stallmann einem der Vordenker der Open Source Bewegung unter wirtschaftlichen Aspekten umstritten sind, so sind sie doch in dem Kontext dieser Arbeit und auf dem Gebiet der Bildung einleuchtend:

> *Außerdem braucht die Gesellschaft Freiheit. Wenn ein Programm Eigentümer hat, verlieren die Nutzer die Freiheit, einen Teil ihres Lebens selbst kontrollieren zu können.* (Stallman (1994))

Beispielhaft sei hier die Geschichte von MIDI-Sequencer-Software umrissen (siehe 3.2.1) und die Entstehung von Techno [2]. Das westliche Musik-Notations System basiert seit Ende des 17.Jahrhunderts auf dem Takt. Wobei gegenwärtige Populärmusik fast ausschließlich auf dem Vier-Viertel-Takt (4/4) aufbaut. Diese Takte sind wiederum meist in weitere durch vier teilbare Einheiten gegliedert. Dieses führte zur Entwicklung von sog. Step-Sequencern die exakt 16 Viertel-Taktschläge bzw. vier Takte speichern konnten. Jedem Schlag wurde dabei eine Note (beispielsweise ein Basston oder ein perkussives Geräusch) zugeordnet. Der Nutzer konnte ansonsten wenig Einfluss auf dieses Programm nehmen.

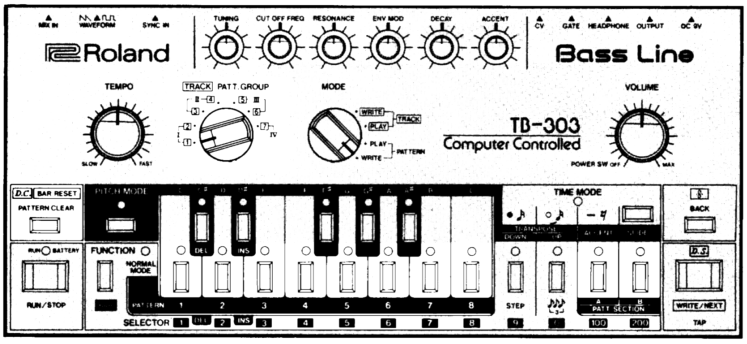

Abbildung 2: Schemazeichnung des Roland TB303 Elektrophons, Quelle: Handbuch TB303

Dieser Nachteil der Geräte (ursprünglich als Ersatz für eine Begleitband gedacht) war so offensichtlich unmusikalisch, dass diese recht schnell vom Markt auf dem Flohmarkt landeten und die Produktion eingestellt wurde. Der Nutzer hatte zu wenig Freiheiten sich musikalisch auszudrücken mit den Geräten. Dennoch wird bis heute an diesem Prinzip festgehalten. Was vor allem daran liegt, dass diese Geräte (z.B. Roland TB303, Roland TB808) nachdem Sie bereits als unbrauchbar galten, von Künstlern wiederentdeckt wurden und mit Hilfe von elektronischen Verzerrern und unsachgemäßer Bedienung zu ganz neuen Klangerzeugern

[2] Es sei angemerkt, dass ”Techno” nur als ein Oberbegriff für diverse Spielarten elektronischer Musik verstanden werden kann, somit also auch die Entstehung von Techno den Umfang dieser Arbeit sprengen würde und die hier vorgestellte Entstehungsgeschichte lediglich Beispielcharakter hat

umgewandelt wurden (siehe 1.3), so dass eine vollkommen neue und eigenständige Spielart entstand, die bis heute weiterlebt (allgemein: Techno, speziell: Acid-House).

Dieses Beispiel zeigt, dass die Vorgaben der Unterhaltungsindustrie oft nicht förderlich sind für das Musizieren oder allgemein die künstlerische Arbeit, dass erst durch den kreativen und sogar missbräuchlichen Umgang mit der Technik bedeutungsvolle Kunst entstehen kann: Erst das Öffnen der Geräte oder im metaphorischen Sinne das Aufbrechen der Struktur, die Wiedererlangung der Bedienungsfreiheit über die Technologie eröffnet in manchen Fällen den künstlerischen Nutzen.

Open (Source) Software und Open Hardware kann so also auch als Aufforderung verstanden werden: Öffne die Software und öffne die Hardware! Letzteres ist als ”Hacking” bekannt und gewinnt hier konträr zur kriminalisierenden Sichtweise eine Legitimation unter künstlerischen Gesichtspunkten.

Am praktischen Beispiel (z.B. auch aus der Erfahrung durch die Zusammenarbeit mit Daniel Hausig, HBK-Saar) und durch theoretische Überlegungen zeigt sich, dass kommerzielle Lösungen zur Komposition, Wiedergabe und Archivierung von elektronischer Kunst Open Source und Open Hardware Lösungen unterlegen sein können.

Zurückkehrend zur anfänglichen Motivation ein Kunstwerk hundertprozentig Do-It-Yourself zu entwickeln - was mit dem rasanten technischen Fortschritt zunehmend schwierig erscheint - eröffnet sich immerhin die Möglichkeit, dieses durch die freie Zusammenarbeit von Kunstschaffenden und Informatikern entstehen zu lassen, wobei die Freiheit bleibt, sämtliche Komponenten genau so zu verändern, dass sie der künstlerischen Idee entsprechen.

1.5 Interdisziplinäre Zusammenarbeit

Die interdisziplinäre Zusammenarbeit von Ingenieuren, Wissenschaftlern und Künstlern wird von mehreren Instituten gefördert. Bereits 1967 erkannten Billy Klüver und Robert Rauschenberg die Bedeutung dieser Zusammenarbeit und gründeten die Non-Profit-Organisation E.A.T. - Experiments in Art and Technology

> ...um die Kollaboration zwischen Ingenieuren und Künstlern zu fördern.
>
> In den Jahren 1967 und 1968 war das E.A.T. zunächst einmal bemüht, die Aufmerksamkeit der Öffentlichkeit auf sich zu lenken und Kontakte herzustellen. Auf diese Weise wollte man Ingenieure dafür gewinnen, mit Künstlern zusammenzuarbeiten, und zugleich Gelder für die Projekte des E.A.T. beschaffen. [..] Wir haben einmal pro Woche einen Tag der offenen Tür in den Räumen des E.A.T. eingeführt, an dem sich Künstler und Ingenieure treffen und informell austauschen konnten. Im März 1967 hatten wir einen Stand gemietet beim Jahrestreffen des Institute of Electrical and Electronics Engineers (IEEE), bei dem Künstler überzeugende Argumente für eine Beteiligung von Ingenieuren liefern konnten. [..] Unser Werben um Interessenten waren erfolgreich. Innerhalb von drei Jahren

> konnten wir zusätzlich zu den 2.000 Künstlern mehr als 2.000 Ingenieure aus dem ganzen Land zur Mitgliedschaft bewegen. Dadurch war es uns möglich, auf jede Anfrage eines Künstlers, der technische Hilfe brauchte, zu reagieren und auf diese Weise Künstler mit Ingenieuren zusammenzubringen, die mit ihnen bei Projekten zusammenarbeiten konnten, die die Künstler alleine gar nicht hätten realisieren können.

Aus diesem Ansatz hervorgegangen sind viele weitere Projekte: So gibt es u.A. an der Columbia University im Computer Music Center seit 2000 das "dorkbot" Projekt von Douglas Repetto mit dem Motto

> people doing strange things with electricity.

Als europäische Institute seien hier das IRCAM [3] im Centre Pompidou als Ausgangspunkt für viele bedeutende Projekte genannt sowie in Deutschland das ZKM [4]. Festivals wie z.B. die ”Ars Electronica” in Linz, Österreich vergeben Preise und organisieren Wettbewerbe.

Mittlerweile gibt es eine Fülle von Software und Hardware die aus diesem interdisziplinären Spektrum hervorgegangen ist.

1.6 Überblick

Es sollen Anforderungen rechnergestützter Kunstprojekte aufgezeigt, vorhandene Lösungen analysiert und ein Ausblick auf fortschrittliche Technologien gegeben werden mit dem Schwerpunkt auf offene Systeme.

2 What They Want

Interaktive multimediale Installationen stellen wie industrielle Umgebungen hohe ingenieurstechnische Anforderungen an Hard- und Software.

- Flexibilität (um Kunstschaffenden die Möglichkeit zu geben, sich so frei wie möglich auszudrücken)
- Wiederverwendbarkeit (aus wirtschaftlichen Gründen, und zur evolutionären Weiterentwicklung der künstlerischen Konzepte)
- Erweiterbarkeit (s.o.)

[3] Institut de Recherche et Coordination Acoustique/Musique, Paris
[4] Zentrum für Kunst und Medientechnologie, Karlsruhe

- Stabilität (Der Ausstellungsbetrieb kann sehr stressig werden für Mensch und Maschine)
- Wartungsfreiheit (Notfall-Service für Installationen dieser Art ist teuer und schlecht für die Reputation)
- Bedienbarkeit (Installationen müssen nicht nur von Entwicklern einfach bedienbar sein, sondern z.B. auch von Aufsichtspersonal in Kunsthallen)
- Autonomie (gerade bei temporären Installationen oder Kunst im Öffentlichen Raum kann man sich nicht auf vorhandene Infrastruktur verlassen. Ein- und Ausschalten von Arbeiten, das Booten, Initialisieren und die Fehlerbehandlung von Systemen sollte möglichst selbständig geschehen.)
- Kosteneffizienz (gerade im Hinblick auf studentische Projekte und Do-It-Yourself Initiativen)
- Energieeffizienz (elektronische Kunst verbraucht Ressourcen, Dauerbetrieb im ”Endless-Loop”-Modus wie teilweise üblich verschwendet solche)

Zusätzlich zu diesen allgemein gültigen Anforderungen müssen ästhetische Aspekte berücksichtigt werden. Vor allem hardwareseitig: z.B. Geräuschentwicklung, Format, optische Integrierbarkeit bzw. (Un)Sichtbarkeit. Softwareseitig sollte der kreative Schaffensprozess nicht durch Restriktionen (wie sie in anderen informationstechnischen Anwendungsgebieten evtl. erwünscht sind) gestört werden.

Verschiedene Formen von Ausstellungen müssen berücksichtigt werden:

Temporäre Ausstellungen: beispielsweise eine Verkaufsausstellung in einer Galerie, Kunstmessen, Festivals, Kunst im Öffentlichen Raum. Die Objekte werden nur für einen kurzen Zeitraum aufgebaut, evtl. sogar nur für wenige Tage konzipiert und hergestellt.

Langzeit-Ausstellungen und Festinstallationen: beispielsweise Kunst an Bauwerken, öffentliche oder private Sammlungen. Die Objekte werden z.B. fest installiert und müssen über langen Zeitraum funktionsfähig sein (auch mit Blick auf Wertsteigerung, beispielsweise im Vergleich zu Gemälden).

Es folgt die Vorstellung von vier typischen Projekten an der Schnittstelle von Kunst und (Computer-) Technik, welche exemplarisch die Anforderungen an Hard- und Software in diesem Bereich verdeutlichen:

2.1 Beispielszenarien

2.1.1 DNA-2-Light

Sequenzen des menschlichen Genoms (insbesondere der ”time-less”- Abschnitt, zuständig für das Zeitgefühl) sollen für unterschiedliche ”Kunst am Bau”- Projekte, als Farb-

lichtsequenzen dargestellt werden. Auf einer mit Leuchtobjekten ausgestatteten Fassade (am Biologischen Institut der Universität Saarbrücken, sowie am Spredaturm in Burgdorf/Schweiz) sollen ”stand-alone” computergenerierte Matrix-Animationen abgespielt werden. Drei Zeichenpaare (medizinisch: Codon) entsprechen ca. vier Sekunden Animation, der gesamte DNA-Strang des Menschen würde ca. 380 Jahre brauchen, um vollständig auf der Fassade gezeigt zu werden(vgl. Orgelstück ”Organ 2/ASLSP” von John Cage, `http://www.john-cage.halberstadt.de`). Theoretisch sieht das Konzept vor, die Daten aus einer öffentlich zugänglichen Datenbank der bekannten entschlüsselten menschlichen Gene (z.B. ”The GDB Human Genome Database”) auszulesen und in Echtzeit in Farbsequenzen zu wandeln.

Für dieses Projekt wurde ein Java-Programm zum Einlesen von Gen-Codes entwickelt, welche diese in Steuerinformationen für ein kommerzielles Theaterlichtprogramm konvertierte. Leider wurden von dem Hersteller keine Angaben zu dem verwendeten Format herausgegeben, so dass dieses vom Autor ”gecracked” werden musste. Die Steuerung der Leuchtobjekte an der Fassade wurde mit DMX512-Datenbus und einem Mikrocontroller-basierten kommerziellen Embedded System realisiert.

Abbildung 3: Fassade der DNA-2-Light-Installation an der Universität Saarbrücken, jedem Leuchtobjekt kann über DMX512 einer von $(2^8)^3 = 16777216$ Farbwerten zugewiesen werden. ©Daniel Hausig

2.1.2 voice.decoder.2

Auszug aus dem Katalogtext der Sammlung Saarlandmuseum Saarbrücken:

Voice.decoder.2 ist eine computergesteuerte Lichtinstallation. Eine Gruppe von 10, mit Tischbeinen versehenen und in unterschiedlichen Schräglagen über dem Boden schwebenden, gleichformatigen und selbstleuchtenden Farbflächen bilden raumgreifende Elemente, eine quasi ungeordnete Möblierung. Die Farbtischplatten aus Glas reagieren soundgesteuert auf Lautsprechergeräusche im Raum. Die Installation voice.decoder.2 thematisiert Aspekt des mobilen Telefonierens. Aus Handygesprächen, die vom Autor [Daniel Hausig] über einen längeren Zeitraum als Ausgangsmaterial gesammelt wurden, sind einzelne Sequenzen bearbeitet und in Licht- und Tonwerte umgewandelt worden. Der Besucher kann die Interaktivität von Ton und Licht an den selbstleuchtenden Tischen verfolgen, die ihre Farben chamäleonartig ändern. Telefongespräche werden in einen Kommunikationszusammenhang zwischen Farbfeld und Hörstück umgewandelt.

Lasergraviertes Glas, gefasst in Alu-Profilrahmen mit rückseitig montierten Tischbeinen, LED Leuchtmittel, Steuerelektronik , Computer, Aktivlautsprecher. 10 Einzelelemente zu: L 110 x B 80 x H 50

Abbildung 4: Aufbau der ”voice.decoder.2” Installation ©Daniel Hausig

Abbildung 5: Hardware zum Mitschneiden von Telefongesprächen für die ”voice.decoder.2” Installation

Dieses Projekt wurde mit kommerzieller Kompositionssoftware auf Windows XP komponiert und dann auf einen DMX512 / Audio-Datenrecorder aufgezeichnet und abgespielt. Der Datenrecorder wurde vom Autor unter der Verwendung eines Embedded Linux Systems mit Solid-State-Disks (Ausfallsicherheit) und USB-to-DMX512-Interface (OpenHardware) entwickelt. Entsprechende Software wurde erweitert und neuentwickelt, wobei auf Treiber-Ebene Prozeduren zur Synchronisation von Licht und Ton-Daten geschrieben wurden. Telefongespräche wurden mit einem umgebauten Bluetooth-Headset aufgezeichnet.

2.1.3 Sonderausstellung im Museum der Arbeit

Im Rahmen dieser Ausstellung sollen nicht nur Bilder und Texte gezeigt werden sondern auch typische Gebrauchsgegenstände (hier: Kissen, Autoradio, Muschel). Jeder Gegenstand soll durch eingebaute Lautsprecher eine Audioinformation abspielen, ausgelöst durch eine Besucheraktion (z.B. Drücken eines Schalters, Bewegung). Zudem soll eine "Hörbar" aufgebaut werden, eine Anhörstation für Audiomaterial. Die Ausstellung ist zunächst für längere Zeit aufgebaut, muss dann abgebaut werden und geht auf Tournee. Die Funktionen sollen dabei erhalten bleiben. Auf der technischen Seite haben wir wieder, trotz unterschiedlichem Konzept die gleichen Anforderungen:

Die Ausstellungsmacher brauchen eine Administrationssoftware (Kompositionsprogramm für Audiomaterial und entsprechende Hardware). Es werden dann Medienplayer benötigt, die in diesem Fall keine Farbverläufe sondern Audiosequenzen (im weiten Sinne: kontinuierliche Daten) abspielen müssen. Diese sind entweder zentral gesteuert und müssen über ein Bussystem verbunden werden, oder aber brauchen jedes einen eigenständigen Hardwarecontroller mit entsprechender Software.

Es gibt nahe liegende Lösungen aus dem Bereich der Unterhaltungselektronik (mp3-

Player, CD-Player, etc.), die jedoch nicht ohne zusätzlichen Aufwand (z.B. Umbau der Starttasten) eingesetzt werden können.

Die vorgeschlagene Lösung mit OpenSource Embedded Systemen auf Linux-Basis und Aufbau eines Funknetzwerks mit günstigen Standard-Komponenten, wurde zu Gunsten der konventionellen Variante (Teure Spezial-CD-Player mit Trigger-Eingängen usw.) verworfen.

2.1.4 Textile Interfaces am Department Design

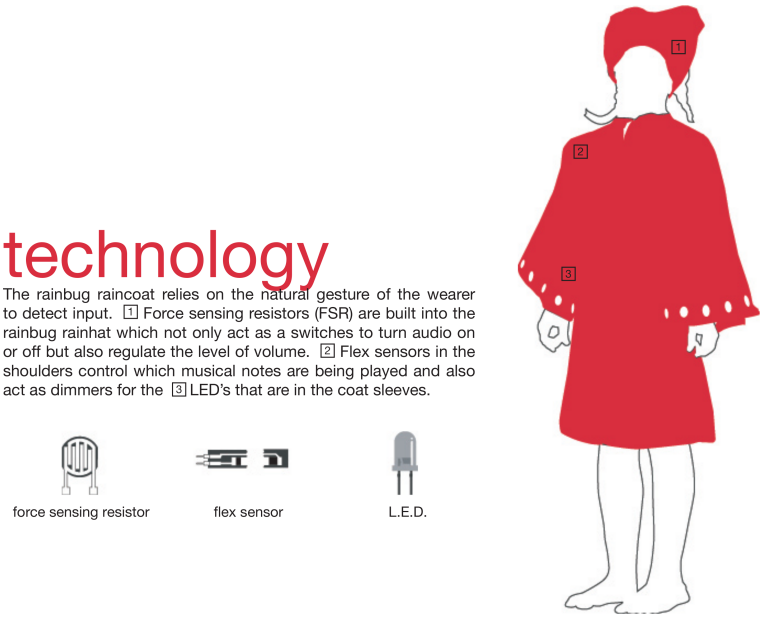

Abbildung 6: Funktionsübersicht des ”Rainbugs”, Prototyp interaktiver Textilien ©Franziska Hübler, Jeremy Tai Abbett

Im HAW Studiengang Design bietet Franziska Hübler einen Kurs zum Thema ”Textile Interfaces” an. Interaktive Kleidung, ausgestattet mit Sensoren (textilen Schaltern sog. ”soft switches”, elektronischen Reißverschlüssen, Dehnungs-Sensoren, etc.) und Aktoren (Leuchtmittel, Lautsprecher, etc.) soll entwickelt und erprobt werden. Studierende sollen einfachen Zugang zu entsprechender Hardware bekommen. Offene Mikrocontroller Systeme (siehe 3.3.1) werden eingesetzt. Schalter werden aus verschiedenen Stoffschichten zusammengesetzt, leitfähige Garne werden vernäht und mit der Steurungshardware verbunden. Es entstehen interaktive multimediale Installationen und Produkte, bei denen Besucher bzw. Kleidungsträger durch Bewegung Reaktionen auslösen. Typisches Beispiel ist der interaktive Regenmantel für Kinder. Kinder haben eine Vorliebe für Verwandlungen, Verkleiden-Spielen. Der Mantel ermöglicht seinen Trägern, durch Gesten Farbigkeit und Klang der Kleidung zu beeinflussen. (vgl. Hübler und Abbett (2002))

Für die Realisation dieser Projekte sind u.A. Hardware Hacks eingesetzt worden (MIDI-Keyboards und Mobiltelefone öffnen und Tasten an selbstgebastelte Schalter löten). Es soll nun aber verstärkt offene Hard- und Software eingesetzt werden (z.B. Arduino).

Abbildung 7: Aufbau eines ”Mixed Media Gebildes” am HAW Department Design: Durch eingewebte Sensoren, können Besucher mit dem textilen Dschungel interagieren ©Judith Stryczek

2.2 Fazit

Die exemplarisch vorgestellten Projekte zeigen, welche Anforderungen für kommende Konzepte zu erwarten sind. Die Umsetzung hätte teilweise wesentlich eleganter gelingen können unter Verwendung von offener Hard- und Software.

3 What We Have

3.1 Software

An den Schnittstellen von Kunst und Computertechnologie haben sich bereits zahlreiche, hauptsächlich ”freie” Softwareprojekte angesiedelt. Im Folgenden sollen die - meiner Meinung nach - wichtigsten vorgestellt werden.

3.1.1 Processing

Processing ist eine java-basierte, objektorientierte, quelloffene Programmiersprache und Entwicklungsumgebung zur Erstellung von interaktiven, dynamischen Grafiken. Processing

wurde am MIT[5] für Studenten, Grafiker und Kunstschaffende entwickelt als sog. ”Sketchbook” zum Erlernen von Programmiertechniken und zum Erstellen von Prototypen in einem visuellen Kontext. Software im Allgemeinen wird dabei von den Autoren als eigenständiges Medium mit einzigartigen Qualitäten verstanden, gleichbedeutend mit älteren Medien wie Film, Fotografie und Malerei. Analog zur Malerei wird bei Processing auch von Skizzen gesprochen: Programmskizze die schnelles und einfaches Ausprobieren einer künstlerischen Idee ermöglichen. Über eine Exportfunktion können diese als Linux, Windows oder MacOS Anwendungen ausgegeben werden. Der offene Charakter von Processing ist bewusst gewählt: gerade im Bereich der bildenden Künste dominieren auf dem Sektor der elektronischen Bildverarbeitung wenige große Softwarefirmen (z.B. Adobe), die mit ihrer Firmenstrategie auch Einfluss auf den Schaffensprozess von Künstlern ausüben (z.B. über Preispolitik und Formatvorgaben). Processing möchte ein offenes Werkzeug sein, dass Kunstschaffenden die Fähigkeit gibt Technologie einfach zu erlernen, zu bedienen und nach eigenen Vorstellungen zu erweitern. Processing wird von einer großen ”Web-Community” unterstützt und hat durch den offenen Charakter weite Verbreitung gefunden. Processing läuft auf allen gängigen Plattformen und es gibt Processing-Derivate für Mikrocontroller-basierte Hardware (siehe 3.3.2) und mobile Geräte. (vgl. Reas und Fry (2007)

3.1.2 Datenstromorientierte Programmiersprachen

Bei der Komposition von interaktiven, multimedialen Anwendungen sind datenstromorientierte Programmiersprachen weit verbreitet. Diese erlauben meist über ein grafisches Interface Objekte zur Generation, Manipulation, Analyse sowie Ein- und Ausgabe von kontinuierlichen Mediendaten zu komplexen Netzen (Patches) zu verschalten. Dabei werden eine Auswahl an Quellen, Knoten und Senken mit Graphen verbunden. Diese können z.B. Hardwareschnittstellen oder Signalgeneratoren sein. Der Schwerpunkt liegt bei der Echtzeitverarbeitung von Signalen. Die visuelle Programmierung motiviert Anwender durch die intuitivere Bedien- und Erlernbarkeit, durch den anschaulichen Realitätsbezug und Abschwächung der syntaktischen Strukturen (vgl. Schiffer (1996)). Intensive Forschungsarbeit auf diesem Gebiet kommt von Miller Puckette der zunächst am IRCAM und später am CRCA[6] die Sprachen MAX/MSP und PureData entwickelte zur Erstellung von elektronischer Musik (vgl. Puckette (2007)).

MAX/MSP, PureData, jMAX MAX/MSP ist die erste visuelle datenstromorientierte Programmiersprache und Entwicklungsumgebung von Autor Miller Puckette. Seit Beginn der Arbeit an MAX (bzw. damals ”Patcher”) im Jahre 1986 ist eine ganze Familie von verwandten, teilweise untereinander kompatiblen Programmierumgebungen entstanden. MAX/MSP

[5] Massachusetts Institute of Technology, Cambridge, USA

[6] The Center for Research in Computing and the Arts, University of California, San Diego, USA

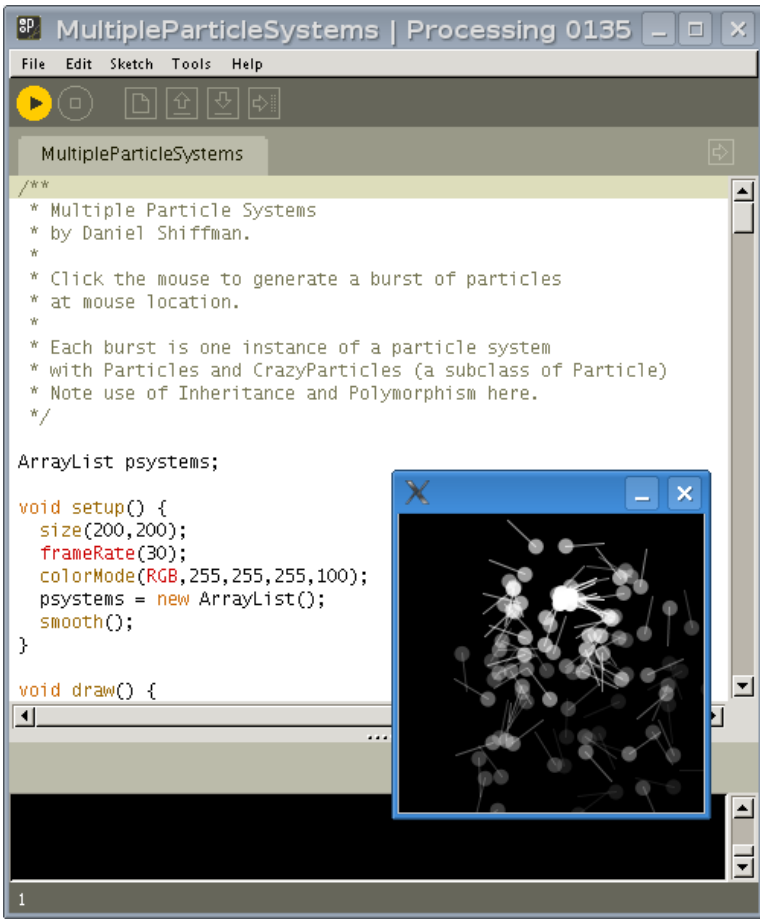

Abbildung 8: Processing Entwicklungsumgebung mit Vorschaufenster und Beispielanwendung zu Partikelschwärmen

und Zusätze zur Videoverarbeitung sind mittlerweile aus dem universitären Bereich heraus zu kommerziellen Produkten der Firma Cycling74 geworden und laufen derzeit auf Mac OS X und Windows XP. Aufbauend auf dem gleichen Konzept entstand die Open Source Variante PureData (welche das Derivat DesireData hervorgebracht hat). PureData wird betreut von Hauptentwickler Miller Puckette am CRCA. PureData ist mit einer großen Bibliothek von sog. ”Externals”[7] im Quellcode und als ”binary” für die Betriebssysteme Linux,Windows und MacOS frei erhältlich. Alternativ zu PureData wird jMAX vom IRCAM entwickelt basierend auf Java. Diese Sprachen stehen in der Tradition modularer Analogsynthesizer, bei denen die einzelnen Funktionsmodule wie z.B. Oszillatoren und Filter mit sog. ”Patchkabeln” verbunden wurden.

[7] PureData Externals sind modulare Erweiterungen, herausgegeben von der ”PureData Community”, welche die Standard-Objekt-Bibliothek um zahlreiche Funktionen erweitern (z.B. Video, Physical Modelling, etc.)

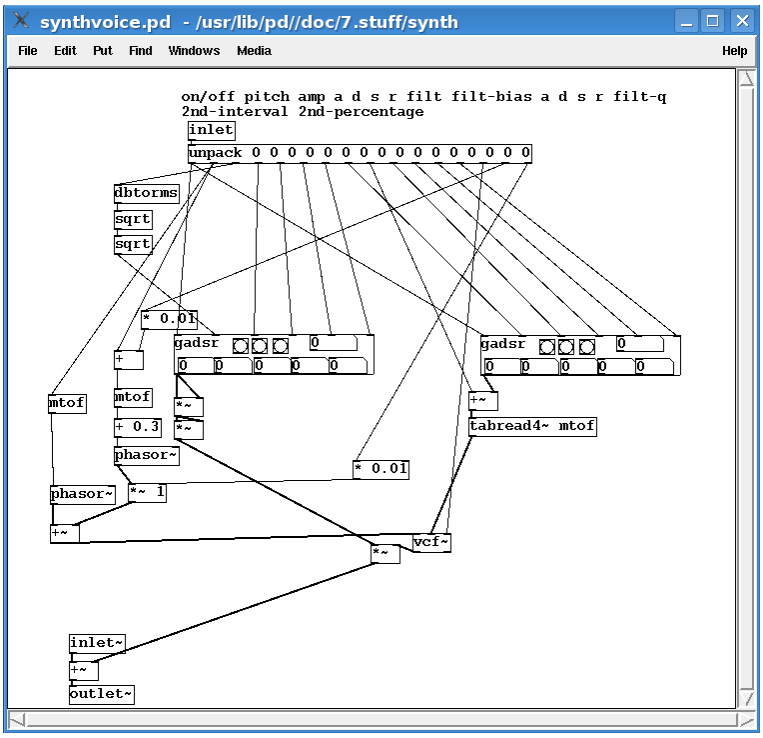

Abbildung 9: Typischer Aufbau eines PureData Patch, hier: Klangsynthese

vvvv, EyesWeb Die Softwareprojekte vvvv und EyesWeb funktionieren nach den gleichen Prinzipien wie die MAX/MSP-Familie nur liegt hier der Schwerpunkt auf kontinuierlichen Videodaten. Objekte zur Filterung und Analyse von Videodaten werden bereitgestellt (z.B. Motion-Tracking).

OpenMusic - A Visual Programming Language for Computer-Aided Composition Auch OpenMusic (entwickelt am IRCAM) ist eine quelloffene visuelle Programmiersprache, die es erlaubt Objekte aus einer Bibliothek am Bildschirm zu verschalten. Hier liegt der Schwerpunkt auf der algorithmischen Komposition von musikalischen Werken. Die Ausgabe erfolgt im klassischen Notensatz. Die Entwickler sprechen von einer ”Icon”-orientierten Programmiersprache, die größtenteils nach dem einfachen ”Drag-And-Drop”-Prinzip funktioniert. Zu Grunde liegt die Programmiersprache Common Lisp und ein entsprechender OpenSource Compiler. Bereits früher gab es Projekte, die für die rechnergestützte Komposition Lisp als Grundlage verwendeten. Hier sei beispielsweise ”Common Music” erwähnt, welches 1989 am CCRMA[8] ins Leben gerufen wurde, um Musikern einen Zugang zu den vielen kreativen Möglichkeiten der aufkommenden günstigen Computerhardware zu geben.

[8]Center for Computer Research in Music and Acoustics, Stanford University, USA

SuperCollider, ChucK Exemplarisch für den Bereich der textbasierten Sprachen zur Komposition von elektronischer Kunst seien hier SuperCollider und ChucK vorgestellt. Beide veröffentlicht unter der GPL[9] und lauffähig unter den gängigen Betriebssystemen Linux, Mac OS, Windows. Beide wurden entwickelt für Klangsynthese in Echtzeit, algorithmische Komposition und Live Perfomance (bzw. ”Live Coding”) von Klangkunst sowie für wissenschaftliche Analyse von Klang. Wie bei den visuellen, datenstromorientierten Programmiersprachen können auch hier aus einfachen Grundmodulen komplexe Netzwerke erstellt werden. SuperCollider funktioniert dabei seit der aktuellen Version 3 nach dem Client-Server Prinzip, wobei Nachrichten über das OSC Protokoll transportiert werden. Andere Anwendungen mit OSC Schnittstellen können so zur Steuerung des Servers eingesetzt werden. (siehe 3.2.3). Die jüngere Sprache ChucK wurde 2003 auf der ICMC[10] vorgestellt und wird derzeit aktiv an der Princeton Universität entwickelt (vgl. Wang und Cook (2003)). Sie ermöglicht sog. ”On-The-Fly”-Programmieren, das Programmieren in Echtzeit. Funktionen werden dabei von einer virtuellen Maschine nach strengen Zeitkriterien ausgeführt (”strongly timed”). Für die Synchronisation von Funktionen wurde ein eigener Scheduler entwickelt um sample-genaue gleichzeitige Verarbeitung zu ermöglichen. Eine eigene Syntax wurde eingeführt, die auf dem ChucK-Operator ”=>” basiert. Dieser erlaubt die Verknüpfung (z.B. $noise => env => biQuad => dac$;), Verschachtelung und Verzweigung von Signalen. ChucK unterstützt zur Interaktion mit der Umwelt u.A. MIDI, OSC und HID[11]. Als visuelle Erweiterung und Entwicklungsumgebung zu ChucK ist das Projekt ”Audicle” entstanden. Es visualisiert Datenströme und Timing-Informationen der ChucK-VM und ist gleichzeitig (Echtzeit-) Editor für ChucK-Programme (vgl. Wang und Cook (2004)).

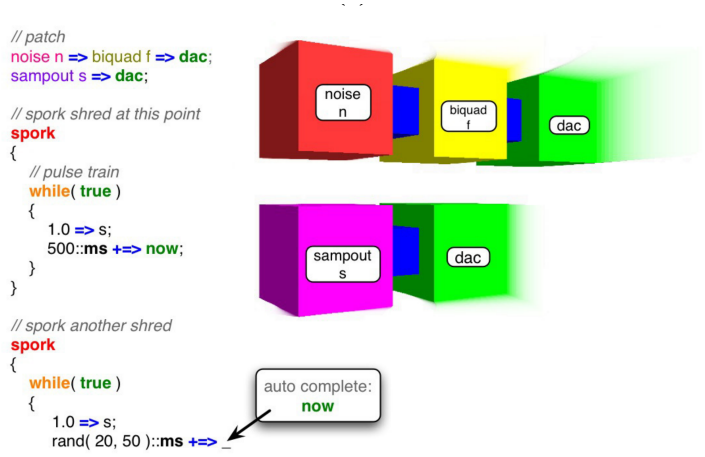

Abbildung 10: ChucK On-The-Fly-Editor, Teil der visuellen Entwicklungsumgebung Audicle

[9] GNU General Public License, verbreitete Lizenz für freie Software
[10] International Computer Music Conference
[11] Human Interface Device, USB-Geräteklasse zur Mensch-Maschine-Interaktion

3.2 Kommunikation

Erfolgreiche Interaktion setzt gemeinsame Sprache, bekannte Kommunikationsprotokolle voraus (vgl. (Igoe, 2007, Making Things Talk)). Auf diesem Gebiet haben sich über die letzten Dekaden nach Ablösung von Analog-Steuerungen zwei digitale Standards etabliert: DMX512 (siehe 3.2) für das Vernetzen von Scheinwerfern und Theaterlichtsteuerungen. MIDI (siehe 3.2.1) für das Vernetzen von elektronischen Klangerzeugern und Sequencern. Beide erscheinen im Vergleich mit aktuellen technischen Möglichkeiten als veraltet sind aber so verbreitet, dass weiterhin viele Hersteller diese Schnittstellen in Ihre Produkte integrieren. Trotz der zahlreichen Nachteile haben diese beiden Protokolle wesentliche Vorteile: DMX512 und MIDI sind offene, einfach zu implementierende Protokolle und werden überall verstanden.

3.2.1 DMX512

Das DMX512 Protokoll (Digital Multiplex, maximal 512 Kanäle) wurde 1986 vom United States Institute for Theatre Technology (USITT) empfohlen zur Steuerung von Theaterlicht Anlagen.

- basiert auf EIA-485
- symmetrische Leitungsführung
- maximal 32 Geräte, parallel verschaltet
- 512 Adressen
- Payload: 8bit
- Baudrate: 250kbps
- Point-to-Multipoint, bzw. Broadcast
- unidirektionale Kommunikation, Master->Slave
- manuelle Adressierung
- standardisiertes Protokoll, u.a. DIN 56930

Ein typischer DMX-Datenbus startet am Lichtsteuerpult in der Regie und führt von dort zu mehreren sog. Dimmerpacks neben oder hinter der Bühne, welche die Lichtintensität von konventionellen Scheinwerfern regeln (vgl. Stickel (2004)). Während in den Anfängen der Lichtregie noch von Hand betätigte Blenden (”Shutter”) verwendet wurden, um die Helligkeit der Leuchtmittel zu beeinflussen und später analog (0-10V) geregelte Lastelektronik zum Einsatz kam, lassen sich mit DMX512-Netzwerken komplexere Aufgaben erfüllen. Nicht nur

Intensität sondern auch Farbe, Bewegungen (z.B. Pan/Tilt von sog. Moving Heads) und Blendenformen können adressiert und geregelt werden. DMX512-fähige Geräte belegen daher meistens nicht nur einen Kanal, sondern beginnend bei einer Startadresse mehrere (z.B. drei für RGB-Farbsynthese). Zunehmend komplexere Installationen erfordern eine wachsende Anzahl an DMX-Adressen. Erweiternd sind dafür sog. DMX-Universen definiert (maximal 16), welche jeweils den kompletten Adressraum (512 Kanäle) abbilden. Der klassische DMX-Master ist das Lichtpult, bestehend aus einer Vielzahl von Fadern, die jeweils einen Kanal im Wertebereich von 0-255 Regeln. Im Zuge des technischen Fortschritts wurden diese Steuerungen mit Speicher für einzelne Szenen (eines Bühnenstücks oder einer sog. ”Show”) und Abläufe (”Cues”, ”Fades”) oder Schaltungen für Algorithmen (”Chases”, Lauflichter) und Sensorik (Sound-to-Light) ausgerüstet. Moderne Lichtsteuerungen basieren auf leistungsfähigen Rechnersystemen und sind ausgestattet mit umfangreichen Bedienungselementen und Monitoren. Ebenso sind DMX-Schnittstellen für handelsübliche PCs verfügbar, die mit entsprechender Lichtsteuer-Software die Funktionen von klassischen Lichtpulten virtualisieren und um zahlreiche Möglichkeiten (Simulation, Integration von Multimedia-Daten wie Audio, MIDI und Video) erweitern. So betrachtet erscheint das DMX512-Protokoll überholt: Rechnersysteme verfügen bereits standardmässig über verschiedene Schnittstellen (USB, Bluetooth, IEEE 802.3, IEE 802.11, IEEE 1394 usw.) welche größere Bandbreite, komplexere Strukturen und größere Adressräume ermöglichen. DMX512 hat diesen gegenüber eindeutige Nachteile:

- keine automatische Adressierung, vgl. TCP/IP,DHCP
- kleiner Adressraum (im Gegensatz zu MAC/IP)
- unidirektional, Geräte können kein Feedback über ihren Zustand geben
- keine Sicherungsschicht, vgl. OSI-Referenzmodel
- geringe Datenbreite (8bit)
- kabelgebundende Leitungsführung

Als Beispiel sei hier eine LED-Matrix angeführt: Durch die rasche Entwicklung auf dem Gebiet der lichtemittierenden Halbleiter, ist die kostengünstige Massenproduktion von leuchtkräftigen LED in einem breiten Farbspektrum möglich. Installationen großer RGB-LED-Pixel-Arrays sind daher wirtschaftlich geworden. Will man diese mit konventioneller DMX-Lichttechnik steuern, stößt man bereits bei einer Matrix von 13x13 Pixeln an die Grenzen eines DMX-Universums $13 * 13 * 3 = 507$

Diesen Nachteilen wird mit zwei wesentlichen Lösungsansätzen begegnet: Erweiterung des DMX512-Standards unter Beibehaltung der Grundlegenden Spezifikationen und Einführung neuer Übertragungsprotokolle basierend auf gängigen Standards. Hier sei die Entwicklung von proprietären Erweiterungen zu DMX512 zu nennen (z.B. Zusammenlegung

von Kanälen zur Steigerung der Datenbreite: LSB = Grobeinstellung, MSB = Feineinstellung). Auch von Seiten der USITT gibt es Bestrebungen zur offiziellen Erweiterung des Protokolls: DMX512/2000 und RDM[12] (vgl. (Howell, 2007a, What is RDM?)). RDM bietet u.a. bidirektionale Kommunikation und besseres Adressmanagement. Nahe liegend ist der Ansatz DMX512-Daten oder allgemein Steuersignale einer Installation über Ethernet bzw. IP-Netze zu übertragen. Mit zunehmender Ver-(IP)-netzung der Lebensräume (z.B. von Veranstaltungsräumen oder öffentlichen Einrichtungen wie Museen) und Miniaturisierung und Massenproduktion von Ethernet-Schnittstellen-Bausteinen und Controllern wird das Einbetten von IP Stacks in Endgeräte zum Standard. Der intelligente Scheinwerfer besitzt eine Ethernetschnittstelle sowie ein Mikrocontroller-System mit entsprechendem IP-Protokoll-Stack, adressiert sich selber, bietet Dienste an (Licht an, Licht aus), sendet Status-Meldungen und Position ("Mir wird zu warm hier oben") und reagiert in Echtzeit auf Befehle. In Bezug auf oben genanntes Beispiel werden die Pixel einer LED-Matrix zu Knoten eines riesigen Lichtnetzwerks, zu ”smart dust” oder besser: ”smart dots”. Doch gibt es hier noch keine Normierung. Eine Vielzahl von Implementierungen von DMX-Over-IP sind auf dem Markt, meistens proprietäre Lösungen großer Veranstaltungstechnik-Firmen (siehe 3.2.2). Auch in der OpenSource-Gemeinde gibt es einige Projekte, die sich mit DMX512 beschäftigen. So haben beispielsweise Studenten der HAW-Hamburg (vgl. Stickel (2004)) ”dmx4linux” veröffentlicht, eine Treiber-API für DMX512 Kommunikation auf Linux-Systemen.

3.2.2 MIDI

Das Pendant zu DMX512 in der Lichttechnik ist MIDI im Bereich Tonstudiotechnik: seit Einführung im Jahre 1983 hat sich MIDI als Protokoll zur Kommunikation zwischen Geräten im Audio-Bereich als Quasi-Standard etabliert. (vgl. Enders (1997)). Typischerweise werden über MIDI seriell (mit 31250 bps) Daten von einem Controller (Keyboard mit Klaviertastatur) an einen Tongenerator (Synthesizer, Sampler) übertragen. Es können je nach Implementation auf Controller und Empfänger eine Reihe von Informationen übertragen werden u.A.:

- MIDI-Kanal (1-16)
- Notennummer (entspricht gedrückter Taste auf einem Klavier, Tonhöhe)
- Anschlagstärke und “Aftertouch”
- Pitch Bend (Tonhöhenverschiebung)
- MIDI Clock und Timecode (zur Synchronisation mehrerer Geräte)
- diverse Steuerdaten (Control Change, Program Change, System Exclusives, etc.)

[12] Remote Device Management

Auch MIDI muss unter heutigen technischen Möglichkeiten (wie DMX512) als veraltet angesehen werden: MIDI ist auf 16 Kanäle beschränkt und MIDI-Daten auf Werte zwischen 0-127. MIDI hat aber wiederum den entscheidenden Vorteil der weiten Verbreitung und Akzeptanz. Aktuelle Geräte werden weitgehend noch mit MIDI Schnittstellen angeboten da bis jetzt kein neuer Standard durchsetzungsfähig war. So auch auf dem Gebiet der interaktiven Kunst, gerade bei der Erfassung von Sensoren und Umsetzung von Sensordaten in multimediale Ereignisse. Analog-to-MIDI Converter ermöglichen versatile Mensch-Maschine-Kommunikation.

Im Zuge der Weiterentwicklung von Multimedia-Übertragungsprotokollen gibt es bei MIDI ebenso wie bei DMX512 die Bestrebung MIDI-Netzwerke durch IP-Netze zu ersetzen. Ein entsprechender Entwurf wurde z.B. von der IETF herausgegeben (vgl. (Lazzaro, 2006, RTP Payload Format for MIDI)). Neuere Controller kommunizieren zudem zunehmend via USB mit Softwaresystemen.

3.2.3 Artnet

Ein erfolgreiches Beispiel DMX512- oder allgemein Lichtsteuerungs-Daten über IP-Netze zu übertragen ist ”Artnet”. Artnet wurde als offener Kommunikations-Standard von der Firma Artistic Licence herausgegeben (vgl. Howell (2007b)) unter folgenden Gesichtspunkten:

- Ethernet bietet eine viel höhere Datenbreite als DMX512
- Ethernetprodukte wie z.B. Hubs sind weiter verbreitet als Bühnentechnik-Komponenten und können daher kostengünstiger angeboten werden.
- Ethernet verwendet kostengünstige Kabel, die standardmäßig installiert und getestet werden können
- Evtl. bestehende Infrastruktur kann genutzt werden
- Lösungen zur Überwindung von physikalischen Barrieren sind bereits erhältlich (WiFi, Telefonnetze, etc.)
- Stern-Topologie: verlässlicher als Ring oder ”Daisy-Chain”-Schaltung (bei DMX512)
- Bidirektional

Artnet findet bereits relativ große Verbreitung bei Herstellern von Veranstaltungstechnik, hat aber auch einige Nachteile. Hauptsächlich die Synchronisation von Geräten: es gibt kein Zeitmaß innerhalb der Artnet-Universen (z.B. im Gegensatz zu DMX512, OSC, MIDI und konkurrierenden proprietären Lösungen). Werden Daten über mehrere Knoten transportiert, kann es zu Zeitverzögerungen kommen. Zudem können viele Broadcast-Nachrichten bei Artnet zu Netzüberlastung führen. Dennoch ist Artnet als offener Standard interessant für den

Kontext dieser Arbeit. Simon Newton entwickelte in seiner Abschlussarbeit an der CSSE[13] zum Thema ”DMX over IP” eine quelloffene Artnet Bibliothek (libartnet) für Linux und portierte diese auf eine ”gehackte” embedded Plattform. Diese wurde in Projekten erfolgreich eingesetzt.

3.2.4 OSC - OpenSound Control

OpenSound Control ist ein Kommunikationsprotokoll für Computer, Klangerzeuger und andere Multimediageräte, optimiert für den Einsatz in modernen IP-Netzwerken. Der Fokus liegt auf Echtzeitverarbeitung und Steuerung von Multimedia-Daten. OSC ist bereits weit verbreitet und wird in neueren Eingabegeräten für musikalische Aufführungen (z.B. Lemur, monome), beim Musizieren über verteilte Systeme in lokalen und globalen Netzwerken sowie zur Interprozesskommunikation innerhalb von Anwendungen (siehe 3.1.2) eingesetzt. OSC wurde 1997 am CNMAT von Matthew Wright und Adrian Freed vorgestellt als ein potentieller Nachfolger des MIDI-Standards. OSC ist quelloffen und wurde bereits von vielen Softwareprojekten implementiert und auf zahlreiche Architekturen portiert. Es existieren Plug-Ins und Bibliotheken u.A. für C/C++, Java, (liblo, WOscLib), für MAX/MSP, PureData, Reaktor[14], SuperCollider, Adobe Flash (flosc), Processing etc. (vgl. Wright (2003))

Eigenschaften von OSC sind:

- Client/Server Modell
- Nachrichten-Orientiert
- hierarchische Adressierung (Baumstruktur)
- ”Adress Pattern Matching” mit regulären Ausdrücken (z.B. mehrere Empfänger)
- symbolische und hochauflösende numerische Datenpakete
- hochauflösende ”Time Tags”
- atomare Ereignisse (durch sog. Nachrichten-”Bundles”)
- dynamische und flexible Infrastruktur
- Abfragesystem zur Entdeckung von Diensten erreichbarer Server

OSC stellt bereits eine gute Lösung für verschiedene Anforderungen dar, fraglich ist aber, in wie weit OSC mit umfassenderen Ansätzen (wie z.B. Jini, UPnP) Schritt halten kann.

[13] School of Computer Science and Software Engineering, University Of Western Australia

[14] erfolgreiche, kommerzielle Software zur virtuellen Klangsynthese

Abbildung 11: Lemur Multitouch Controller der Firma JazzMutant, monome Interface von Crabtree und Cain, basierend auf dem OSC Kommunikationsprotokoll

3.3 Hardware

Wie einleitend erläutert ist es evtl. gerade der nicht sinngemäße Umgang mit Hardware, das Öffnen von Geräten und vor allem die Zweckentfremdung von Technologie (z.B. Nutzbarmachung verschlossener Funktionalität oder Ausnutzung geringer Anschaffungskosten) die zu kreativen Entwicklungen führt. Sicher ist es für Kunstschaffende und Studenten vorteilhaft wenn Systeme eingesetzt werden können, die von Grund auf quelloffen sind (siehe 3.3.1) aber nicht immer sind diese die interessantesten oder günstigsten Lösungen.

3.3.1 Hardware Hacking

Kommerzielle Unterhaltungselektronik ist oft in Massenproduktion gefertigt (und daher sehr günstig) nur leider nicht für den Einsatz in z.B. interaktiven multimedialen Installationen vorbereitet, obwohl die Hardware entsprechende Eigenschaften hätte. Hier erlangt die Portierung des quelloffenen Linux-Kernels auf zahlreiche Prozessorsysteme besondere Bedeutung. Standard x86 Kernel und Versionen für kleinere Architekturen (arm, mips, etc.) laufen zumindest theoretisch auf Spielekonsolen, Medienplayern, Netzwerk-Routern, Navigationsgeräte usw. Problematisch bei der Portierung auf diese Geräte ist meistens der herstellerseitige Schutz der entsprechenden Speicherbereiche für Betriebssystem und Bootloader. Dennoch gibt es für zahlreiche Geräte sog. ”Hardware Hacks”, die es möglich machen, z.B. eigene Firmware in den Speicher zu schreiben. Diese Geräte werden so interessant für den kreativeren Einsatz im Ausstellungsbetrieb, wenn es z.B. darum geht, günstige und kleine Controller-Einheiten zu bekommen oder Prototypen zu entwickeln. Im Folgenden möchte ich einige dieser Projekte vorstellen, da sie vor allem für (Kunst-)Studenten und allgemein im Ausbildungsbetrieb von Interesse sein könnten: hier stehen nach wie vor nicht immer genügend finanzielle Mittel zur Verfügung, um für studentische Projekte entsprechende (offene) Hardware anzuschaffen.

XBOX-Linux XBOX-Linux ist der wohl bekannteste Hardware-Hack. Es geht um die Portierung des Linux-Betriebssystems auf die erste Version der XBOX-Spielekonsole von Microsoft. Die Hardware solcher Spielekonsolen ist vom Hersteller subventioniert um den Absatzmarkt für Spiele und Softwarelizenzen zu vergrößern. So kommt der Anwender zu einem vergleichsweise günstigen Preis in den Besitz von aufwändiger Hardware. Die XBOX verfügt über eine Intel Celeron 733 MHz CPU, eine nVidia Grafikkarte, 64 MB Hauptspeicher, eine 8 oder 10 GB Festplatte , ein DVD Laufwerk, Audioausgänge und eine Ethernet Schnittstelle. Die Eingabegeräte sind USB kompatibel bis auf die speziellen Steckverbinder. Diese Konfiguration macht die Konsole attraktiv für multimediale, interaktive Installationen. Hier vorgestellte Software wurde erfolgreich installiert und exemplarisch mit einer interaktiven Anwendungen erprobt.

Abbildung 12: ”Proof Of Concept” zur Lauffähigkeit von quelloffenen interaktiven multimedialen Anwendungen auf der Microsoft XBOX Spielekonsole. Das Gesicht auf dem Bildschirm verfolgt mit seinen Augen den Zuschauer. Das Gerät startet die Anwendung auf Knopfdruck. Motion-Tracking via USB-Webcam mit PureData, ”Gem” und ”PDP” Videoerweiterungen, video4linux-Treiber und Debian-GNU/Linux

Für die Umsetzung von ”XBOX-Linux” lobte Michael Robertson (Gründer zahlreicher erfolgreicher Computerfirmen) 100.000 USD aus. 2003 gelang es erstmals Mitgliedern des

XBOX-Linux Projektes einen Bootloader und eine spezielle Linux-Distribution auf der Konsole zu installieren, ohne dabei (und das war eine Vorraussetzung des Wettbewerbes) das Gehäuse zu öffnen. Mittlerweile gibt es detaillierte Anleitungen wie dieser Hack umgesetzt werden kann und einige ausgereifte vollwertige Linux-Distributionen speziell für die XBOX. Am 28. Februar 2007 wurde eine Sicherheitslücke veröffentlicht im Kernel der zweiten XBOX Generation (XBOX360, 3-Core PowerPC Architektur), die es möglich macht auch hier Linux zum Laufen zu bringen (entsprechende Distributionen werden vom Free60.org Projekt herausgegeben).

Ipod-Linux Analog zu XBOX Linux und der Portierung des Linux-Kernels auf zahlreiche x86-basierte Geräte (wie z.B. Spielekonsolen) entstanden Projekte, die sich motiviert sahen Linux auf Embedded Systems wie z.B. Medienplayern, Navigationsgeräte, Telefone und PDAs etc. anzupassen. Der μClinux-Kernel und die μClibc-Bibliothek für Mikrocontroller Systeme ohne MMU[15] ermöglichen diese Vorhaben, wenn es gelingt den Programmspeicher dieser Geräte zu beschreiben. Weit fortgeschritten ist das Projekt IPOD-Linux, die Portierung von Linux auf zahlreiche von Apple herausgegebene Hardware zum Abspielen von Audio- und Videodateien. Viele Modelle können bereits fast vollständig von der speziellen IPOD-Linux-Kernel-Version angesprochen werden. Es lassen sich so beispielsweise offene Datenformate für Musikdateien abspielen und die Hardwareschnittstellen des Gerätes ansprechen. Ältere "IPODs" werden so umgenutzt attraktiv für Kunstprojekte mit geringem Budget. Embedded Systems zur Steuerung von interaktiven Installationen sind denkbar (siehe Beispielszenario "Sonderausstellung im Museum der Arbeit" 2.1.2). Forschungen zu diesem Thema wurden z.B. von Martin Kaltenbrunner und Günter Geiger von der Music Technology Group der Universitat Pompeu Fabra, Barcelona angestellt. 2005 wurde eine Version von PureData für den IPOD (pdPod) vorgestellt (siehe 3.1.2) aufbauend auf "PDa": PureData für Embedded Systems (vgl. Geiger (2003)).

OpenWRT Beim Thema Hardware Hacking und Open Hardware muss auch eines der bekanntesten Projekte aus diesem Bereich, die Portierung von Linux auf WLAN-Router genannt werden. Durch eine öffentliche juristische Auseinandersetzung konnte ein Hersteller von Netzwerkhardware dazu veranlasst werden, den Quellcode für die auf seinen Geräten eingesetzte (ursprünglich freie) Software offenzulegen, so dass Modifikationen durch den Anwender möglich wurden. Diese Möglichkeit rief das Projekt "OpenWRT" und seine zahlreichen Abkömmlinge ins Leben, die eine eigene Linux Variante für Linksys WRT54G WLAN-Router und mittlerweile viele andere solcher Geräte entwickeln. OpenWRT ist ausgereift und bietet z.B. ein spezielles voll beschreibbares Dateisystem sowie einen Paketmanager an. Freie Software aus dem Umfeld multimedialer und interaktiver Anwendungen kann so auf

[15] Memory Management Unit

Abbildung 13: IPOD Linux, Quelle: `http://ipodlinux.org`

der oftmals sehr günstigen und leistungsfähigen Hardware (z.B. ARM-CPU, Flash-Speicher, Netzwerk-, USB- und serielle Schnittstellen) betrieben werden.

3.3.2 Open Hardware

Arduino, Wiring Arduino und Wiring sind zwei verwandte Open-Source Projekte, die jeweils ein Mikrocontroller-Board sowie eine zugehörige auf Processing(siehe 3.1 aufbauende Entwicklungsumgebung bereit stellen. Ziel ist es Studenten und Künstlern bzw. allgemein Nicht-Informatikern die Möglichkeit zu geben, relativ schnell und unkompliziert interaktive Hardware Projekte zu realisieren. Die Experimentier-Boards verfügen über einen Atmel-RISC-Prozessor und haben eine USB oder (bei kleineren Boards) zumindest eine serielle Schnittstelle sowie diverse Ein- und Ausgänge.

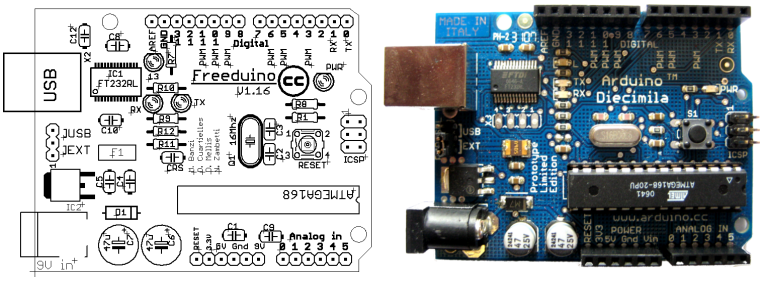

Abbildung 14: Arduino ”Diecimila” Hardware mit USB-to-serial Konverter und Steckplätzen für Experimente, nebenstehend Schema der quelloffenen Variante ”Freeduino” für die sämtliche Baupläne (Schaltpläne und Platinenlayouts) frei verfügbar sind.

Die MCU ist bereits vorprogrammiert mit einem speziellen Bootloader (kompatibel mit dem Atmel-Standard) und kann so ohne zusätzliche Programmer-Hardware direkt über USB vom PC aus beschrieben werden. Entsprechende Funktionalität ist bereits in die IDE integriert. Für die Entwicklung von Programmen ist eine eigene, C-ähnliche Sprache implementiert, die mit wenigen Befehlen fast sämtliche Funktionen der Hardware ansprechen kann.

Beispielsweise serielle Kommunikation:

```
void setup(){
        Serial.begin(9600);
}
void loop(){
        Serial.print("Hello");
}
```

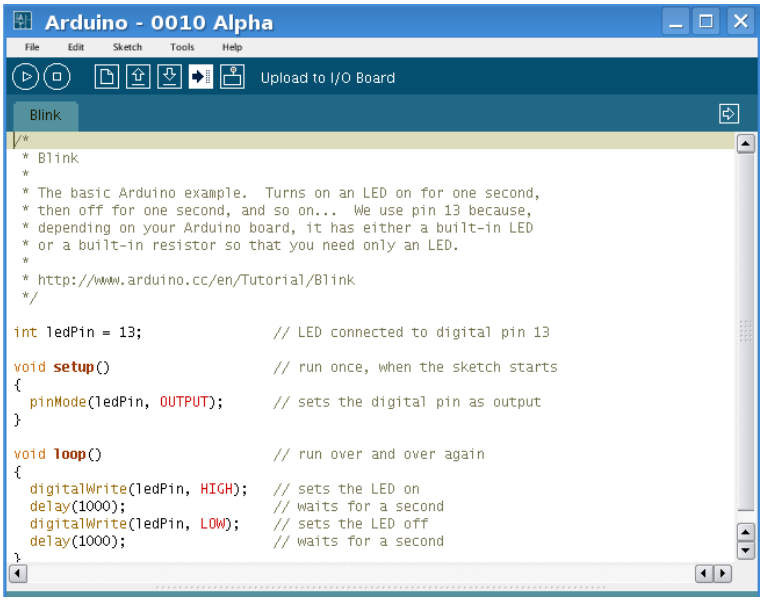

Abbildung 15: Arduino Entwicklungsumgebung aufbauend auf Processing, ein Klick auf ”Upload to I/O Board” startet das sog. ”Sketch” auf der Hardware

4 What To Do

Um zukünftig Kunstschaffende bei der Entwicklung von rechnerbasierten Kunstwerken zu unterstützen, müssen vorhandene Lösungen besser implementiert, besser dokumentiert und weiterentwickelt werden. Die hier vorgestellten Kommunikations-Protokolle sollten auf offenen Plattformen implementiert werden bzw. vorhandene Bibliotheken portiert und getestet werden. Für die vorgestellten Beispiel-Projekte sollten Open Hardware und Open Software Lösungen ausgewählt werden und gegebenenfalls ausgebaut und verbessert werden (siehe Abbildung "Typischer Use Case"). Dabei sollte verstärkt auf gute Dokumentation, Anwenderfreundlichkeit und Transparenz geachtet werden, um interdisziplinäre Zusammenarbeit zu fördern. Wenn möglich, sollten bereits vorhandene Lösungen einbezogen werden und nicht noch weitere konkurrierende Lösungen neuentwickelt werden.

Es wurde gezeigt, dass u.A. durch das Engagement der hier genannten Institute eine Vielzahl versatiler Werkzeuge und Plattformen zur Verfügung stehen zur Umsetzung interaktiver multimedialer Konzepte. Zukünftig wird es möglich sein, weiter von der Hardware zu abstrahieren und Software wie z.B. PureData und Processing plattformübergreifend auf verteilten Systemen einzusetzen. Hardware Hacking wird wohl als eigenständige Disziplin oder gar Kunstform fortbestehen, doch die zunehmende Öffnung von Standards auf Seiten der Industrie (z.B. Google "Android", Sun Java) und die weitere Etablierung von freier Software (z.B. Embedded Linux) erlaubt Kunstschaffenden und Studenten einfachere Umsetzung ihrer Ideen. Der Ball wird zurückgespielt: es sind kreative Konzepte gefragt, die die zur Verfügung stehenden Möglichkeiten in origineller Weise ausnutzen. Gleichzeitig führen die Standardisierung und Verbreitung von Anwendungen aus dem Umfeld von Kunst und Computertechnik zur Adaption in den Massenmedien. Originelle Konzepte werden kopiert und zu Werbegags degradiert. Im Zuge dessen ist wiederum eine Abkehrbewegung der Disziplinen Kunst und Computertechnik zu verzeichnen: weg von der Unterhaltungsindustrie, hin zu klassischen Medien (z.B. Renaissance von analogen Tonträgern, klassische Malerei)

Ständig vor dem Monitor sitzen und Tasten drücken, da verkümmert man!

(Konzeptkünstler Hans Haacke in der Wochenzeitung "Die Zeit" Ausgabe 47/2006)

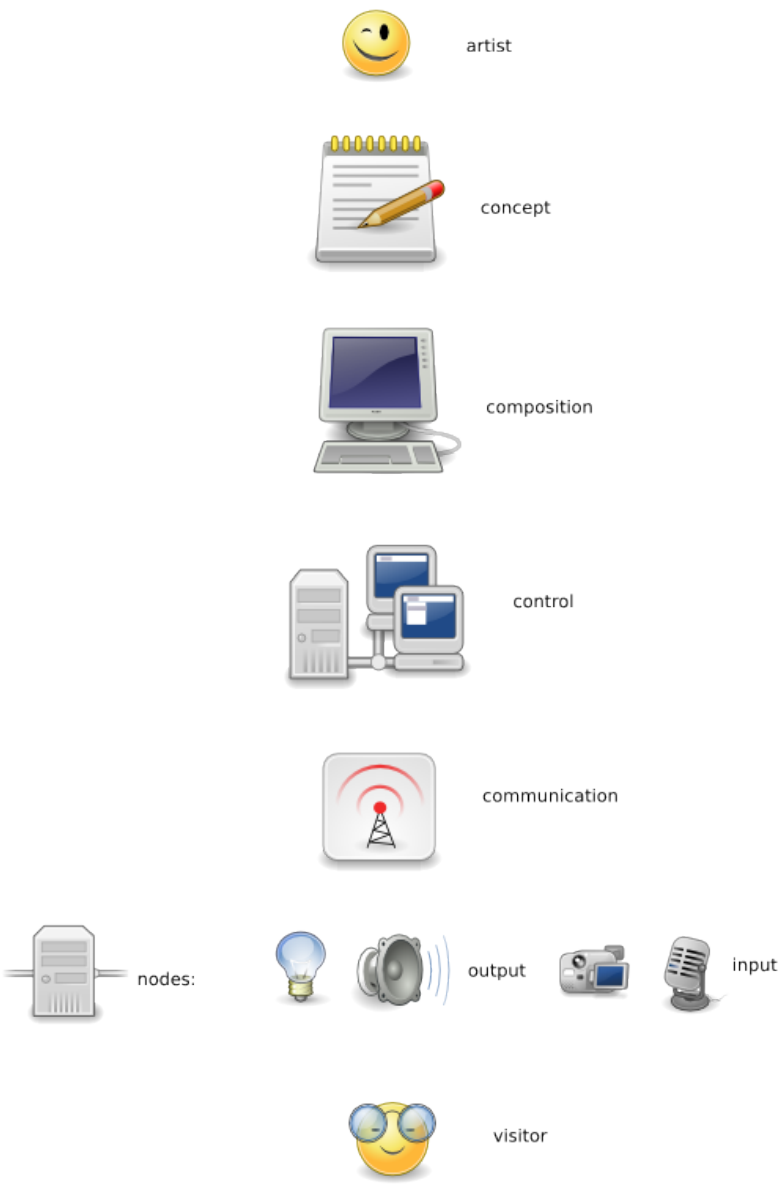

Abbildung 16: Typischer Use Case interaktiver, multimedialer Installationen

Abbildungsverzeichnis

Literatur

[Benjamin 1981] BENJAMIN, Walter: *Das Kunstwerk im Zeitalter seiner technischen Reproduzierbarkeit*. Suhrkamp Verlag, 1981

[Enders 1997] ENDERS, Roland: *Das Homerecording Handbuch*. GC Gunther Carstensen Verlag, München, 1997

[Geiger 2003] GEIGER, Günter: PDa: Real Time Signal Processing and Sound Generation on Handheld Devices / Music Technology Group, Pompeu Fabra University, Barcelona. 2003. – Forschungsbericht

[Howell 2007a] HOWELL, Wayne: Application Note 5 What is RDM? (2007)

[Howell 2007b] HOWELL, Wayne: Artistic Licence Application Note 13 What is Art-Net? (2007)

[Hübler und Abbett 2002] HÜBLER ; ABBETT: designing desire, wearables / Interaction Design Institute Ivrea, Italien. 2002. – Forschungsbericht

[Igoe 2007] IGOE, Tom: *Making Things Talk: Practical Methods for Connecting Physical Objects*. O'Reilly Media, 2007

[Jagdmann 2005] JAGDMANN, Dirk: *Analyse und Resynthese des Gitarrenklangs*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, 2005

[Lazzaro 2006] LAZZARO, Wawrzynek: RTP Payload Format for MIDI / Internet Engineering Task Force. 2006. – Forschungsbericht

[Puckette 2007] PUCKETTE, Miller: *The Theory And Technique Of Electronic Music*. World Scientific, 2007

[Reas und Fry 2007] REAS, Casey ; FRY, Ben: *Processing: a programming handbook for visual designers and artists*. The MIT Press, Cambridge, Massachusetts, 2007

[Schiffer 1996] SCHIFFER, Stefan: Visuelle Programmierung - Potential und Grenzen. In: *Beherrschung von Informationssystemen; Heinrich C. Mayr* (1996)

[Stallman 1994] STALLMAN, Richard: Warum Software keine Eigentümer haben sollte. In: *GNU Homepage* (1994)

[Stickel 2004] STICKEL: *Konstruktion eines mobilen Systems zur Steuerung von Bühnenbeleuchtungsanlagen*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, 2004

[Wang und Cook 2003] WANG, Ge ; COOK, Perry: ChucK: A Concurrent, On-the-fly, Audio Programming Language / Computer Science Department, Princeton University. 2003. – Forschungsbericht

[Wang und Cook 2004] WANG, Ge ; COOK, Perry: The Audicle: A Context-Sensitive, On-the-fly Audio Programming Environ/mentality / Computer Science Department, Princeton University. 2004. – Forschungsbericht

[Wikipedia 2008] WIKIPEDIA: *Demoszene — Wikipedia, Die freie Enzyklopädie*. 2008. – [Online; Stand 18. März 2008]

[Wright 2003] WRIGHT, Matthew: OpenSound Control: State of the Art 2003. 2003. – Forschungsbericht