Bachelorarbeit

Philipp Kühn

Reaktor - Erstellung einer DJ Software für Microsoft Surface

Philipp Kühn

Reaktor - Erstellung einer DJ Software für Microsoft Surface

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Angewandte Informatik

am Department Informatik

der Fakultät Technik und Informatik

der Hochschule für Angewandte Wissenschaften Hamburg

Philipp Kühn

Thema der Bachelorarbeit

Reaktor - Erstellung einer DJ Software für Microsoft Surface

Stichworte

Reaktor, DJ, Music, Surface, Multitouch, Traktor, Reactable, mixiTUI

Kurzzusammenfassung

Im Rahmen dieser Arbeit sollen Anforderungen an DJ Software auf Multitouch Eingabesystemen untersucht und darauf basierend eine DJ Software auf dem Microsoft Surface umgesetzt werden. Es wird zunächst bestehende DJ Software analysiert, aus den gewonnenen Erkenntnissen ein Konzept erstellt und darauf aufbauend die Realisierung beschrieben.

Philipp Kühn

Title of the paper

Reaktor – Creation of a DJ software for Microsoft Surface

Keywords

Reaktor, DJ, Music, Surface, Multitouch, Traktor, Reactable, mixiTUI

Abstract

In this paper, the requirements of DJ software on multi touch systems will be analysed and based on this work a DJ software for the Microsoft Surface will be created. At first existing DJ software will be analysed and from the findings of this analysis, a concept will be created and realized.

Inhalt

1. Einleitung

Im Folgenden sollen die Ziele, und der Anlass zu dieser Arbeit erläutert werden.

1.1 Motivation

Seit es Discos und Clubs gibt, in denen Musik gespielt wird, gibt es DJs. Und seit je her wird versucht, mit immer neuen Kniffen die Musik gekonnt zu mixen. So wurde früher nur mit Plattenspielern und Vinyl Schallplatten aufgelegt. Da Musikstücke nicht immer dieselben „Beats Per Minute“ (BPM) haben, war es eine der Hauptaufgaben eines DJs, die zwei Platten in Geschwindigkeit und Beat anzugleichen. Dies geschieht mit Fadern für das Tempo und dem Verlangsamen oder Beschleunigen des Plattentellers mit der Hand und muss live, während des Auflegens passieren. Mit dem Aufkommen von DJ Software hat sich hier vieles geändert. Es können im Vorfeld die BPM eines Stücks bestimmt werden und ein Beatgrid gesetzt werden. Damit fiel eine der Hauptaufgaben eines DJs während des Auflegens weg, nämlich das Angleichen des Tempos und der Beats von zwei oder mehreren Stücken, da dies jetzt von der Software übernommen werden kann. Somit kann sich mehr auf Effekte und andere Dinge während des Auflegens konzentriert werden. Da vielen DJs allerdings eine gute Haptik wichtig war, welche von Computern mit einer DJ Software einfach nicht gegeben war, entstanden MIDI-Controller und Time-Code-Systeme. Hierdurch war die bekannte Haptik wieder gegeben. Mit dem Aufkommen von Touch und Multitouch Systemen kamen wieder neue Möglichkeiten zur Bedienung von DJ Software auf, welche den Anlass gaben, hierfür eine Möglichkeit der Interaktion zu schaffen, welche die Anforderungen eines DJs erfüllt und darüber hinaus eine gute Haptik bietet.

1.2 Zielsetzung

Ziel dieser Arbeit ist es, eine DJ Software für das Microsoft Surface zu entwickeln, welche die Anforderungen eines DJs erfüllt und darüber hinaus eine gute Haptik und einfache Bedienbarkeit bietet. Dafür wird zunächst untersucht, welche Anforderungen an eine gute DJ Software von einem DJ gestellt werden. Anschließend wird sowohl konventionelle, als auch Multitouch DJ Software untersucht, um weitere Anforderungen und bisherige Umsetzungen herauszufinden. Hierbei werden die angebotenen Funktionen, die Bedienbarkeit und die Haptik untersucht.

2. Analyse

In den folgenden Abschnitten sollen die Zielplattform und bestehende Systeme betrachtet werden. Diese Analyse bildet die Grundlage für das weitere Vorgehen und die Umsetzung einer eigenen Software.

2.1 Abgrenzung

In dieser Arbeit soll es um DJ Software auf Multitouch Tischen gehen. Dies bedeutet, dass es eine tischähnliche Oberfläche mit eingelassenem Display geben muss. Darüber hinaus muss der Tisch in der Lage sein, eingaben von Fingern und/oder Objekten, welche auf dem Tisch platziert werden, entgegenzunehmen.

Dadurch fallen Konzepte, welche ohne einen Multitouch Tisch auskommen, wie z.B. D'Groove [Beamish et. al. 2004], welches auf eine rotierende Schallplatte als Eingabemedium setzt, heraus. Solche Konzepte spielen ihre Stärken am besten in Verbindung mit vorhandener, konventioneller DJ Software aus, da sie als Controller für eben solche genutzt werden können und hier interessante neue Eingabemöglichkeiten bieten.

Abbildung 1 D'Groove [Beamish et. al. 2004]

Wichtig ist auch, dass es sich bei der Software um DJ Software handelt, welche von einem professionellen DJ in einer Disco eingesetzt werden könnte. Hierdurch fallen Konzepte, welche eher auf experimenteller Basis mit Musik oder Sound im Allgemeinen interagieren, wie z.B. OtoMushi [Andre 2010], welches auf ein Mikrofon als Eingabemedium setzt und dann zufällig aufgenommene Sounds miteinander kombiniert, auch heraus. Solche Software ist eher für Ausstellungen, oder aber für weitergehende Forschung geeignet, eignet sich aber nicht als DJ Software, da sie dem DJ nicht die volle Kontrolle über die abgespielte Musik gibt. Somit könnte der DJ sein Können hier nicht ausspielen, da die Software zu viel von alleine und zufällig macht.

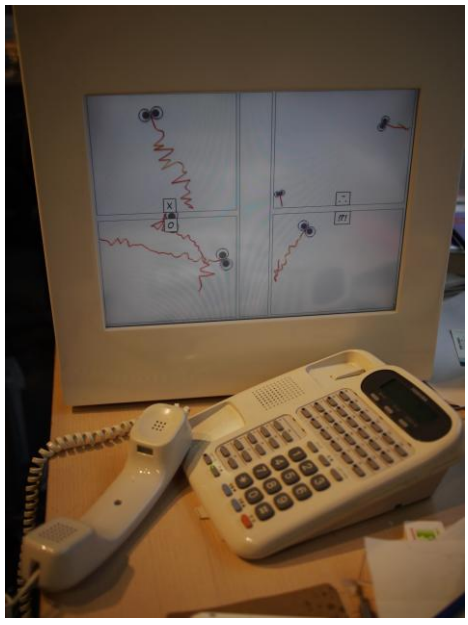

Abbildung 2 OtoMushi [Andre 2010]

2.2 Microsoft Surface

Der Microsoft Surface [Surface 2011] ist ein Computer in Tischform, welcher mit einem Display zur Objekterkennung ausgerüstet ist. Das Aussehen erinnert hierbei an einen Couchtisch. Der Computer ist dabei mit einem normalem Desktop PC zu vergleichen. Als Betriebssystem kommt Microsoft Windows Vista (32 Bit) zum Einsatz, wobei es inoffiziell auch möglich ist, Windows 7 zu benutzen. Das Display besteht aus einer Plexiglasscheibe mit milchiger Oberfläche, welche von unten mit einem Beamer angestrahlt wird. Das Display misst 30“ und bietet eine native Auflösung von 1024x768 Pixeln. Zur Objekterkennung werden im inneren 5 Infrarotkameras verwendet. Somit können neben mehreren Fingern auch komplexe Objekte erkannt werden. Als Benutzeroberfläche wird ein Natural User Interface verwendet, welches nicht mit Maus und Tastatur, sondern mit Fingern und Objekten bedient wird. Auf der CES 2011 stellte Microsoft die Version 2 des Surface vor, auf welche hier aber nicht näher eingegangen wird.

Mit dem Surface SDK, welches für die .NET Plattform von Microsoft zur Verfügung gestellt wird, ist eine einfache Abfrage der auf dem Display erkannten Objekte (Finger, Byte/Identity Tag, Blob) möglich. Es werden sowohl die Position, als auch die Rotation erkannt. Bei Byte und Identity Tags wird zudem noch der Wert erkannt. Der Wert eines Byte Tags hat hierbei eine Länge von einem Byte (4Bit, 0-254), der Wert eines Identity Tags gliedert sich in 2 64Bit Integer Zahlen. Gestenerkennung ist mit dem Surface SDK nicht möglich und muss selber implementiert werden. Mit dem SDK werden Templates für XNA und WPF mitgeliefert, sodass diese sich als Frameworks anbieten.

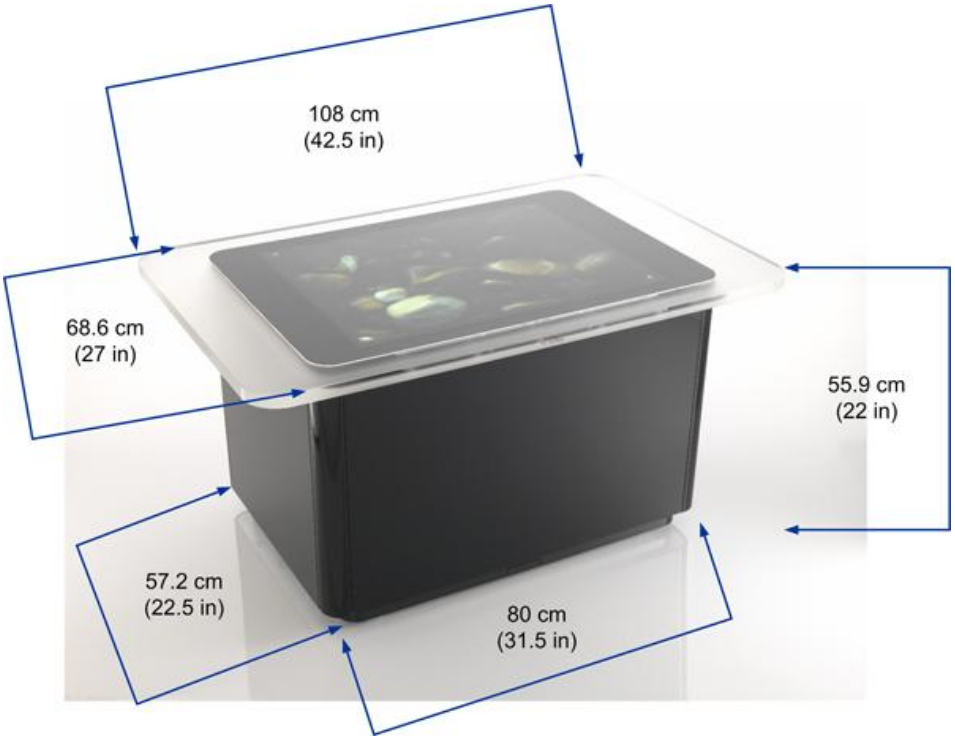

Abbildung 3 Microsoft Surface [Abb Surface]

2.3 Anforderungen an DJ Software

Im Folgenden sollen für einen DJ wichtige Anforderungen an eine DJ Software aufgezeigt und diese kurz erläutert werden. Diese Anforderungen wurden zusammen mit dem seit über 4 Jahren praktizierenden Resident DJ [wertzui] des größten Fantreffens des Animexx e.V., einem eingetragenen und gemeinnützigen Verein der Freunde und Förderer der japanischen Sub- und Populärkultur in Deutschland, der Chisaii, erarbeitet. Anhand dieser Anforderungen werden dann in Kap. 2.4 bestehende DJ Programme analysiert.

- Echtzeit: Das Programm muss in der Lage sein, alle Tätigkeiten, die der DJ vornimmt, in (nahezu) Echtzeit wiederzugeben. Es dürfen keine spürbaren Latenzen (Lag) zwischen Aktion in der Software und hörbarer Reaktion existieren. Da auf herkömmlichen Mischpulten, welche in Hardware fest verdrahtet sind keine Verzögerungen auftreten, würde dies viele DJs von der Nutzung des Programms abhalten.
- Robustheit: Unter keinen Umständen darf die Musik ausfallen. Dies wäre für einen DJ der größtmöglich anzunehmende Patzer. Sollte einmal die GUI einfrieren (was natürlich auch möglichst vermieden werden sollte), ist es oberste Priorität, dass die Musik weiterspielt. Auch muss bei größerer Last des Computers stets gewährleistet sein, dass die Musik ohne Knacken oder Ausfälle weiterspielt.
- Visuelles Feedback: Bei einem herkömmlichen DJ Setup mit Platten-/CD-Spielern und Mixer sieht der DJ sofort, welche Knöpfe gedrückt, welche Potentiometer gedreht und, welche Fader aufgezogen sind. Da bei einer reinen Software dies wegfällt, muss dem DJ auf andere Weise gezeigt werden, welche Einstellungen gerade aktiv sind. Oft geschieht dies durch abbilden der herkömmlichen Elemente eines Mixers auf dem Bildschirm, wobei auch andere Möglichkeiten denkbar sind.
- Mixen: Das Kernelement der Arbeit eines DJs ist das ineinander Mixen mehrerer Lieder. Eine Software, welche nicht die Möglichkeit bietet mehr, als ein Lied gleichzeitig abzuspielen, wäre für einen DJ also nutzlos. Auch muss die Lautstärke der einzelnen Musikstücke individuell regelbar sein. Hierbei sind zwei Lieder gleichzeitig Minimum, wobei viele Programme auch vier Lieder gleichzeitig ermöglichen.
- Tempo und Beat Synchronisation: Damit es sich gut anhört, wenn Lieder gemixt werden, ist es wichtig, dass sie mit dem gleichen Tempo und Beatsynchron laufen. Die Beatsynchronisation ist dabei eine der schwierigsten Aufgaben, die an DJ Software gestellt werden. Hierfür ist ein guter Algorithmus zur Erkennung von Beats nötig.
- Effekte: Um der Musik eine persönliche Note zu verpassen und sie für die Hörer interessanter zu machen, können Effekte eingesetzt werden. Diese verändern die Musik nach bestimmten Regeln. Bekannte Effekte sind der Flanger Effekt, oder der Filter des Xone:92er Mischpultes von Allen & Heath
- Loops: Seit der Ära der „Digital DJs“ gibt es Loops. Diese dienen dazu einen bestimmten Teil der Musik sich immer wiederholen zu lassen. Dieses erleichtert z.B. das Überblenden zwischen zwei Liedern, da man so Anfang und Ende von Liedern beliebig verlängern kann. Looplängen werden normalerweise in Beats gezählt. Übliche Längen reichen von 1/32, bis 32 Beats, wobei als Schritte 2er Potenzen üblich sind.

- Haptik: Klassische DJs sind es gewöhnt mit Plattenspielern und Mischpult zu arbeiten. Der Vorteil hierbei, im Gegensatz zu einer reinen Software Lösung ist, dass man sehr einfach und intuitiv auch mehrere Dinge gleichzeitig tun kann, wohingegen man mit der Maus immer auf eine Sache limitiert ist. Oft wird DJ Software daher mit Controllern eingesetzt, welche wie ein Mischpult aussehen und zu bedienen sind, in Wirklichkeit aber nur Signale an die Software senden. Es wurden sogar sog. Time-Code-Systeme entwickelt, um es zu ermöglichen, normale Plattenspieler in Kombination mit DJ Software zu nutzen. Dies lässt den Schluss zu, dass die Haptik besonders wichtig bei der Bedienung von DJ Software ist.
- Musikverwaltung: Es sollten einfach neue Musikstücke in die Sammlung integriert werden können. Die Sammlung muss verwaltbar und durchsuchbar sein. Nötige Analysen sollten beim Hinzufügen automatisch erfolgen, um den Aufwand für den User so gering wie möglich zu halten.

2.4 Bestehende DJ Software

Die in 2.3 erarbeiteten Anforderungen sollen nun auf DJ Software angewandt werden, welche bereits veröffentlicht wurde.

2.4.1 Konventionelle DJ Software

Als erstes soll Software betrachtet werden, welche auf einem normalen Computer läuft und auch häufig in Discos eingesetzt wird.

2.4.1.1 Native Instruments - Traktor Pro 2

Stellvertretend für konventionelle DJ Software soll hier Traktor Pro 2 [NI 2011] stehen, welche der Nachfolger einer der am weitest verbreiteten DJ Software auf dem Markt ist [Golden 2011].

- Echtzeit: Mithilfe von ASIO Treibern und entsprechenden Soundkarten können Latenzen von unter 10ms umgesetzt werden.
- Robustheit: Selbst, wenn die GUI nicht mehr reagiert, wird die laufende Musik nicht unterbrochen. Sollte die GUI wieder reagieren, kann normal weitergearbeitet werden. Ansonsten ist ein Neustart des Programms notwendig, welcher zwangsläufig eine Unterbrechung der Musik nach sich zieht.
- Visuelles Feedback: Jedes Deck hat eine mitlaufende Waveform Ansicht des gerade laufenden Tracks. Alle virtuellen Fader, Potentiometer und Knöpfe können auf Wunsch eingeblendet werden.
- Mixen: Man hat die Möglichkeit, bis zu 4 Musikstücke gleichzeitig abzuspielen, wobei man jedes Deck auch in ein Sample Deck umwandeln kann, welches die Möglichkeit bietet, bis zu 4 kurze Samples abzuspielen.
- Tempo und Beat Synchronisation: Werden neue Musikstücke zur Bibliothek hinzugefügt, können diese automatisch analysiert werden. Es werden hierbei die BPM und auch das Beatgrid automatisch erkannt. Diese Funktion funktioniert bei elektronischer Musik bereits sehr gut, sodass kaum mehr Einstellungen von Hand gemacht werden müssen. Bei Musik ohne klar erkennbare Beats, funktioniert dies hingegen nicht sehr zuverlässig, sodass oft von Hand nachjustiert werden muss. Per Knopfdruck kann nun die Musik synchronisiert werden, sodass ein manuelles pitchen und angleichen entfällt.
- Effekte: Es sind über 30 Effekte integriert, welche auf bis zu 4 Effekteinheiten verteilt werden können. Jede Effekteinheit kann entweder 3 Effekte aufnehmen, welche dann mit einem Regler gesteuert werden können, oder 1 Effekt aufnehmen, welche dann mit 3 Reglern und 3 Knöpfen gesteuert werden kann.
- Loops: Loops der Größen 1/32 bis 32 Beats können genutzt werden.
- Haptik: Um die Haptik zu gewährleisten, werden Controller auf MIDI Basis unterstützt, wobei die Funktionen des Controllers nach eigenen Wünschen frei festgelegt werden kann. Es werden darüber hinaus die hauseigenen Controller unterstützt, welche nicht auf Midi Basis, sondern als herkömmliches HID operieren und somit eine höhere Präzision, als Midi erreichen. Es wird außerdem das hauseigene Time-Code-System unterstützt.
- Musikverwaltung: Es gibt einen integrierten Browser für die Musiksammlung. Neue Musikstücke können per Drag & Drop oder über ein Kontextmenü hinzugefügt werden. Die Sammlung ist durchsuchbar und kann in Playlisten organisiert werden. ID3 Tags werden automatisch ausgelesen.

Der klare Vorteil liegt also in der Portabilität, da die Software mit allen Musikstücken auf einem Laptop transportiert werden kann, in den vielen Funktionen und der Unterstützung für Controller. Auch ist hervorzuheben, dass es sehr einfach ist, neue Musikstücke hinzuzufügen und vor dem Auflegen zu bearbeiten (Beatgrid, Cues und Loops setzen). Die Nachteile hingegen sind die Haptik ohne Controller, welche quasi nicht vorhanden ist und die Erweiterbarkeit der Software nach eigenen Wünschen.

Abbildung 4 Native Instruments - Traktor Pro 2 (Screenshot)

2.4.2 DJ Software auf Multi Touch Eingabesystemen

Nachdem in 2.4.1 konventionelle DJ Software betrachtet wurde, soll nun DJ Software auf Multi Touch Eingabesystemen untersucht werden, wie auch das Microsoft Surface eines ist.

2.4.2.1 Reactable Systems – Reactable

Eines der bekanntesten Systeme ist der Reactable [Jordà 2010].

- Echtzeit: Da der Reactable ein integriertes System ist, ist eine schnelle Signalverarbeitung garantiert. Eine genaue Angabe zur Latenz war nicht herauszufinden.
- Robustheit: Hierzu wird keine Angabe gemacht.
- Visuelles Feedback: Sind Klötze nahe genug beieinander, verbinden diese sich miteinander und das Audiosignal wird durch verschiedene Klötze geroutet, bevor es zum Output in der Mitte des Tisches gelangt. Diese Verbindung wird grafisch durch Linien, Waveform o.ä. dargestellt.
- Mixen: Es können so viele Soundgeneratoren gleichzeitig genutzt werden, wie auf den Tisch passen und Klötze vorhanden sind.
- Tempo und Beat Synchronisation: Die Musik wird immer auf ein globales Tempo synchronisiert. Die Hierfür notwendige Analyse der Samples erfolgt automatisch.
- Effekte: Es gibt 4 Filter/Effekte, welche jeweils 3 Untertypen besitzen. Somit gibt es insgesamt 16.
- Loops: Da keine langen Musikstücke, sondern nur kurze Samples abgespielt werden, werden diese Samples am Ende automatisch geloopt. Ein verkürzen des Loops auf einen Wert, der geringer, als die Länge des Samples ist, ist hierbei nicht möglich.
- Haptik: Die Interaktion erfolgt mit Klötzen, welche auf einen runden Tisch gelegt werden. Klötze können entweder Sound generieren, ihn verändern, oder globale Einstellungen ändern. Der Reactable bietet dadurch eine sehr gute Haptik auf einem Multitouch System.
- Musikverwaltung: Neue Loopsamples müssen im 16bit wav Format vorliegen. Diese können einfach in einen Ordner kopiert werden und sind dadurch benutzbar.

Da der Reactable nicht als DJ System, sondern als Instrument gedacht ist, ist es umständlicher, neue Musikstücke einzubinden, als z.B. in Traktor Pro 2. Auch sind die Klötze, welche Musik wiedergeben eher dazu gedacht, kurze Samples abzuspielen, als längere komplette Musikstücke. So gibt es auch viele Sound Generatoren, welche einem Synthesizer ähneln und z.B. ein Sinus, oder Sägezahn Signal ausgeben. Auch gibt es Klötze, welche ein Instrument simulieren und somit als Sequenzer agieren.

Abbildung 5 Reactable Systems - Reactable [Abb Reactable]

2.4.2.2 Vectorform - Surface DJ

Surface DJ [Havir, Engalan 2009] von Vectorform ist die bisher einzige DJ Software auf dem Microsoft Surface. Ziel dieser Software ist auf einfache Weise Samples zusammenzuziehen und somit Musik zu machen.

- Echtzeit: Surface DJ benutzt die eingebaute Soundkarte mit den WDM Treibern. WDM Treiber haben gegenüber ASIO Treibern für gewöhnlich eine höhere Latenz, welche sich im Bereich von 10 bis 30 ms bewegt.
- Robustheit: Hierüber liegen keine Informationen vor.
- Visuelles Feedback: Samples werden als virtuelle Plättchen auf dem Display dargestellt und können mit dem Finger in die Mitte zu einem dargestellten Lautsprecher gezogen werden. Je näher sie an der Mitte des Lautsprechers sind, desto lauter werden sie wiedergegeben.
- Mixen: Alle auf dem Display dargestellten Samples können gleichzeitig abgespielt werden.
- Tempo und Beat Synchronisation: Die Samples werden automatisch auf ein festes Tempo synchronisiert.
- Effekte: Effekte sind nicht vorhanden.
- Loops: Die Samples selber werden geloopt. Eine Variation der Länge ist nicht möglich.
- Haptik: Die Samples werden mit einem Finger per Drag & Drop auf den dargestellten Lautsprecher gezogen. Eine Bedienung darüber hinaus ist nicht möglich.
- Musikverwaltung: Es gibt keine Möglichkeit, neue Musik hinzuzufügen.

Für einen spielerischen Umgang mit Musik ist Surface DJ durchaus geeignet, hat aber deutlich zu wenige Funktionen, um als vollwertige DJ Software angesehen werden zu können. Da Surface DJ nur mit den Fingern bedient wird, fehlt hier auch die Haptik.

Abbildung 6 Vectorform - Surface DJ [Abb Vectorform]

2.4.2.3 mixiTUI

mixiTUI [Pedersen, Hornbæk 2009] versteht sich selbst als tangible Sequencer. Vom Aussehen ähnelt es dem Reactable sehr, da es auf die gleiche API (reacTIVision) setzt.

- Echtzeit: Wie der Reactable ist auch mixiTUI ein integriertes Systems, wodurch eine schnelle Signalverarbeitung garantiert ist.
- Robustheit: Hierzu wird keine Angabe gemacht.
- Visuelles Feedback: Werden Scheiben, welche Musik erzeugen auf den Tisch gelegt, verbinden sie sich mit der dem DJ gegenüberliegenden Seite des Tisches. Dies wird durch eine Waveform Visualisierung der Musik von der Scheibe zu der gegenüberliegenden Seite angezeigt. Die Verbindungen sind hierbei immer gerade und parallel zu den kurzen Seiten des Tisches. Wird die Session geändert, so ändert sich die Hintergrundfarbe des Bildschirms passend zu der Session.
- Mixen: Es können so viele Musikstücke abgespielt werden, wie Scheiben auf den Tisch passen. Als Besonderheit bietet mixiTUI Sessions an. So kann jede Scheibe für jede Session eine andere Belegung haben. Sessions können mit einer bestimmten Scheibe auf der rechten Seite des Bildschirms gewählt werden, indem diese auf ein andersfarbiges Quadrat gelegt wird.
- Tempo und Beat Synchronisation: Als Musikeingabe wird auf kurze Loopsamples gesetzt. Diese werden mit einem globalen Tempo abgespielt. Für jedes Loopsample kann hierbei gewählt werden, ob es mit dem Auflegen der Scheibe sofort, oder zeitverzögert, z.B. nach dem nächsten Takt starten soll. Dasselbe gilt auch beim Entfernen einer Scheibe.
- Effekte: Werden Scheiben, welche einen Effekt repräsentieren auf eine Linie zwischen Musikstück und Tischseite gelegt, werden diese als Effekt zwischengeschaltet. Es können hierbei beliebig viele Effekte hintereinander geschaltet werden.
- Loops: Die Musikstücke sind Loopsamples. Eine Variation der Länge ist nicht möglich.
- Haptik: Die Interaktion erfolgt mittels Quadratischen Scheiben, welche auf einen rechteckigen Tisch gelegt werden. Der DJ steht hierbei an einer Längsseite des Tisches.
- Musikverwaltung: Hierzu wird keine Angabe gemacht.

mixiTUI geht schon deutlich mehr in die Richtung eines DJ Programms, ist aber auf das Abspielen kurzer Samples ausgelegt und nicht für lange Musikstücke gedacht.

Abbildung 7 mixiTUI [Abb mixiTUI]

2.5 Zusammenfassung

Eine DJ Software sollte möglichst viele der in 2.1 Genannten Anforderungen umsetzen. Wichtig ist hierbei eine gute Bedienbarkeit durch den User: Es muss eine haptische Komponente geben und Musikstücke müssen einfach in die bestehende Sammlung integriert werden können und sollten nicht nur kurze Samples, sondern auch vollwertige Lieder, sein. Auch sehr wichtig ist, dass die Musik keine Ausfälle hat und leicht ineinander gemixt werden kann. Viele unterschiedliche Effekte sind hingegen nicht ganz so wichtig, so dass eine Auswahl an beliebten Effekten hier ausreichend ist. Auch ist Echtzeit nicht ganz so wichtig, da es zwar schwieriger ist, mit hohen Latenzen zu mixen, aber nicht unmöglich.

	Traktor Pro 2	Reactable	Surface DJ	mixiTUI
Echtzeit	++	+	-	+
Robustheit	++	/	/	/
Visuelles Feedback	+	++	+	++
Mixen	++	++	o	++
Tempo und Beat Synchronisation	++	++	++	++
Effekte	++	+	--	+
Loops	++	o	o	o
Haptik	-- (++ mit Controller)	++	--	++
Musikverwaltung	++	-	--	/

Tabelle 1 Zusammenfassung

Legende: ++ sehr gut, + gut, o befriedigend, -ausreichend, --mangelhaft, / keine Angabe

3. Konzept

Aus den in Kap. 2 gewonnenen Erkenntnissen soll in den folgenden Abschnitten ein Konzept erarbeitet werden, welches möglichst viele Anforderungen umsetzt.

3.1 Interaktion

Bei der Software soll viel Wert auf eine intuitive und leicht erlernbare Bedienung gelegt werden, damit der DJ sich nicht auf die Bedienung der Software, sondern auf die Musik konzentrieren kann. Des Weiteren ist die Haptik sehr wichtig, sodass eine reine Fingereingabe ausscheidet. Die Bedienung soll ähnlich der des Reactable sein, da diese sich als sehr gut für Multitouch DJ Software herausgestellt hat (vgl. 2.4.2.1), wobei für eine noch simplere Bedienung gesorgt wird, indem keine zusätzlichen Fingergesten möglich sind. Objekte können nur verschoben und gedreht werden, darüber hinaus ist keine Interaktion möglich. Dadurch soll gewährleistet werden, dass Neueinsteiger sich schnell mit dem System zurechtfinden und keine weiteren Gesten erlernen müssen. Dies schließt zwar komplizierte Vorgänge durch Gesten aus, kann aber durch mehrere Steine kompensiert werden. Als Beispiel sei hier die Soundausgabe mit der des Reactable verglichen. Beim Reactable gibt es nur einen globalen Soundoutput in der Mitte des Tisches. Soll etwas nicht hörbar sein, kann man eine „Cut Geste“ durch die Verbindung machen und somit den Sound stummschalten. Bei diesem System hingegen kann für jeden Sound separat die Lautstärke geregelt werden, oder der Würfel, welcher den Lautsprecher symbolisiert einfach vom Tisch genommen werden. Hierdurch ist sogar eine weitreichendere Bedienung (individuelle Lautstärken) ohne den Einsatz zusätzlicher Gesten möglich. Gleichzeitig ist die Drehbewegung der Drehbewegung eines Potentiometers auf einem DJ Controller nachempfunden und somit für einen DJ eine bekannte Geste. Das Auflegen der Würfel auf den Tisch erinnert dabei an einen Tastendruck oder das Auflegen der Platte auf den Plattenteller. Das Verschieben der Würfel kann hierbei als neuverkabeln einzelner Komponenten angesehen werden. Somit sind alle möglichen Interaktionen dem DJ bereits bekannt oder leicht abgewandelt, wodurch eine kurze Eingewöhnungszeit garantiert ist.

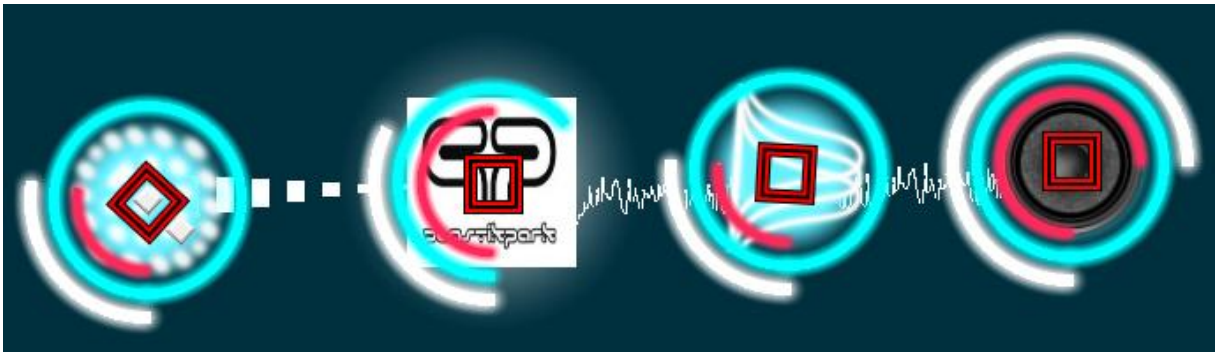

Abbildung 8 Interaktion zwischen Aktoren (Screenshot)

3.1.1 Interaktion durch Drehen

Jedes Objekt hat genau einen Wert, welcher auf zwei Arten verändert werden kann. Bei einem Musikstück ist dies z.B. das Tempo, bei einem Lautsprecher die Lautstärke. Es wurde sich für diese Werte entschieden, da das Drehen des Würfels, welcher ein Musikstück repräsentiert mit dem Drehen des Plattentellers verglichen werden kann. Bei einem Lautsprecher spiegelt das Drehen des Würfels das Drehen an einem Lautstärkeregler wieder. Da jedes Objekt nur einen Wert verändern kann, werden andere Eigenschaften auf mehrere Objekte ausgelagert. Um z.B. die Frequenzbereiche eines Musikstückes einzustellen, kann man Equalizer Objekte zwischenschalten, welche dann wiederum jeweils einen Frequenzbereich (Höhen, Mitten, Tiefen) verändern können. Der Wert eines Objektes setzt sich zusammen aus:

1. Internem Wert: Dieser wird durch Drehen des Objektes verändert. Er wird durch einen roten Kreisbogen visualisiert. Der Wertebereich liegt im Intervall]0,1[. Der Startwert kann unterschiedlich sein. Ein Musikstück hat als Startwert 0,5, was einem Tempo von 100% entspricht. Ein Lautsprecher hat einen Startwert von 0, was einer Lautstärke von 0% entspricht.
2. Externer Wert: Dieser kann durch andere Objekte verändert werden. Er wird durch einen blauen Kreisbogen visualisiert. Der Wertebereich liegt im Intervall]0,1[. Der Startwert ist immer 1, was keiner Beeinflussung durch ein externes Objekt entspricht.

Der Wert berechnet sich nun durch $Wert = interner\,Wert \times externer\,Wert$ und wird durch einen weißen Kreisbogen symbolisiert.

3.1.2 Interaktion durch Bewegen

Werden diese Objekte nun nahe genug aneinander gelegt, beginnen sie miteinander zu interagieren. Hierbei gibt es zwei Basisarten der Interaktion:

1. Über In-/Output: Hierbei wird der Sound durch verschiedene Objekte geroutet, wobei diese den Sound beliebig verändern können. Den Anfang bildet hierbei immer ein Input (Abb.5, 2. v. li.), welcher z.B. ein Musikstück lädt und es über seinen Output verfügbar macht. Am Ende steht immer ein Output (Abb.5, 4. v. li.), welcher den Sound über seinen Input annimmt und ausgibt. Dies geschieht normalerweise über Lautsprecher, wobei auch ein Stream an einen anderen Computer oder eine Dateiausgabe denkbar wären. Zwischen Input und Output können nun Effekte (Abb.5, 3. v. li.) geschaltet werden. Diese nehmen den Sound über ihren Input auf, verarbeiten ihn und machen ihn über ihren Output wieder verfügbar, sodass daran ein weiterer Effekt oder ein Output angeschlossen werden kann.
2. Über Manipulation eines anderen Objekts: Objekte, die keinen Sound verändern, können andere Objekte direkt verändern(Abb.5, 1. v. li.). Dadurch ist es z.B. möglich mehrere Musikstücke zu synchronisieren, oder Werte an andere Objekte zu senden (vgl. 3.1.1 Punkt 2), welche dann aufgrund dieser Werte etwas tun.

3.2 Plug-In System

Ein Plug-In System beschreibt ein System, welches auf einfache Art und Weise durch neue Komponenten, die Plug-Ins, erweitert werden kann. Wichtig hierbei ist, dass das Grundsystem hierfür nicht verändert werden muss. Lediglich die Plug-Ins müssen ggf. Konfiguriert werden.

Damit es auch anderen Entwicklern möglich ist, neue Funktionen hinzuzufügen, soll ein „Pure Plug-In System“ [Birsan 2005] erstellt werden. Ein solches System ist, im Gegensatz zu einem traditionellen Plug-In System, so aufgebaut, dass es ohne Plug-Ins keine Funktionalität, außer das Verwalten von Plug-Ins, zur Verfügung stellt und sehr leichtgewichtig ist. Es soll die Möglichkeit bieten, auf einfache Art und Weise das bestehende Programm zu erweitern, indem z.B. neue Effekte hinzugefügt werden können. Wichtig hierbei ist, dem Entwickler alle Freiheiten zu lassen, die er möchte, gleichzeitig aber Standardimplementationen bereitzustellen. So kann der Entwickler sich entscheiden, ob er nur ein wenig neue Funktionalität hinzufügen, oder etwas ganz Eigenständiges programmieren möchte. Diese Plug-Ins sollen automatisch geladen werden, sodass kein nachträgliches Modifizieren des Quellcodes des eigentlichen Programms nötig ist, um ein solches Plug-In einzubinden. In einer separaten XML Datei werden dabei alle Tags mit assoziiertem Plug-In und zu übergebenden Parametern gespeichert. Hierdurch weiß das Programm, welches Plug-In zu laden ist, wenn ein bestimmter Tag auf das Microsoft Surface gelegt wird und welche Parameter an das Plug-In übergeben werden müssen. Hierbei lässt sich z.B. steuern, welches Musikstück von einem Input Plug-In geladen werden soll.

Beispiel:

```
<Tag Series="1" Value="2" Type="File" File="Musikdatei.mp3" />
```

Diese Zeile sagt aus, dass das Plug-In „File” mit dem Tag 1-2 assoziiert ist. Es wird also geladen, wenn ein Tag mit entsprechender Series und Value auf dem Microsoft Surface platziert wird. Zusätzlich wird das Argument „File“ (nicht zu verwechseln mit dem Type) mit dem Wert „Musikdatei.mp3“ übergeben.

3.2 Architektur

In den folgenden Abschnitten soll die Architektur beschrieben werden, welche eine gute Umsetzung der Anforderungen und der bereits beschriebenen Konzepte ermöglicht.

3.2.1 Komponenten

Um den Gesamtaufbau des Systems darzustellen, werden in den folgenden zwei Abschnitten zuerst die fachliche und anschließend die technische Architektur dargestellt.

3.2.1.1 Fachlich

Da die fachliche Architektur des Reactable bereits eine sehr gute Visualisierung und Bedienung hat, wurde die fachliche Architektur an selbige angelehnt. Allerdings leicht verändert, um der geänderten Interaktion (vgl. 3.1) Rechnung zu tragen.

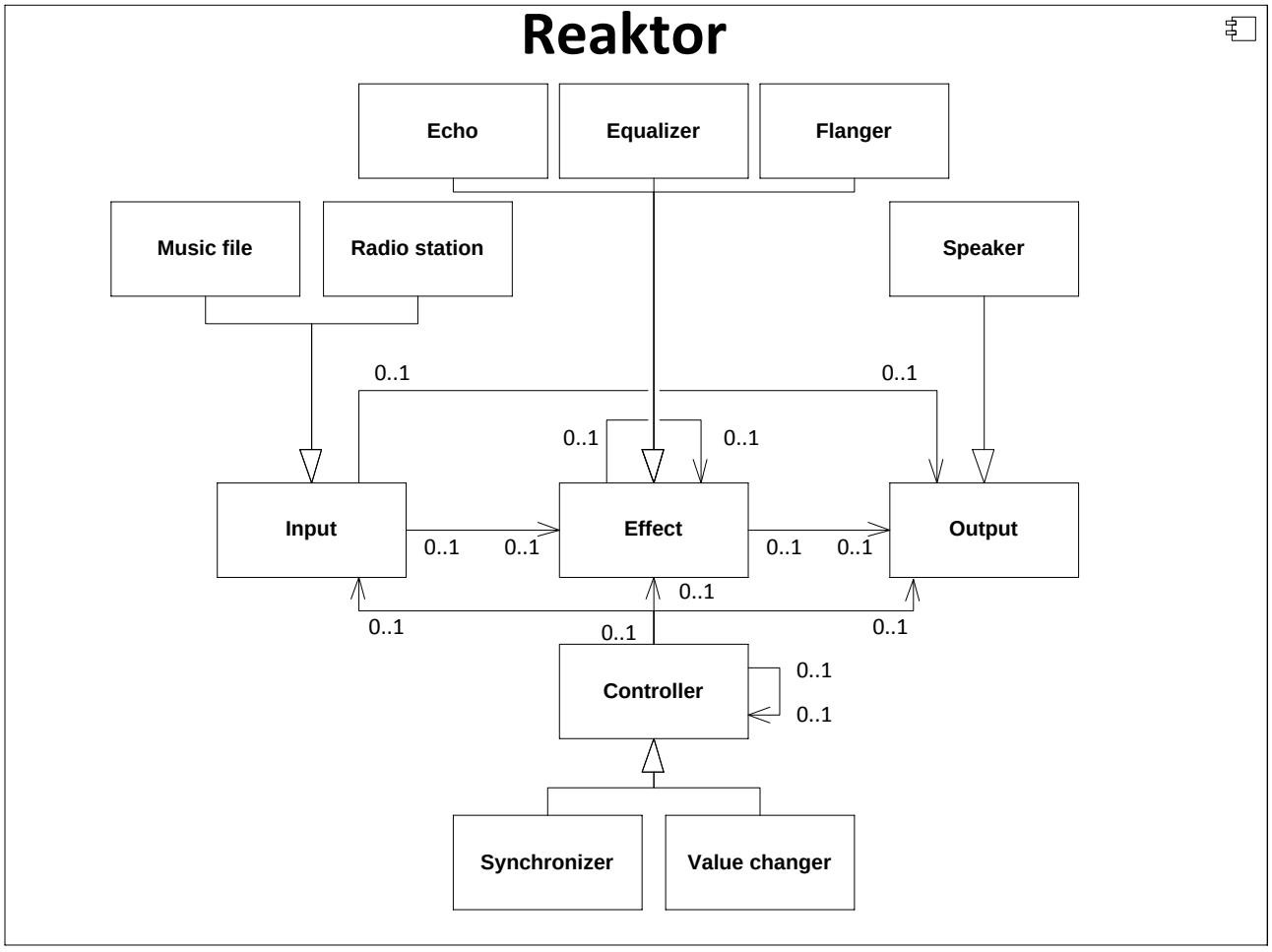

Abbildung 9 Fachliche Architektur (UML Diagramm)

Das Programm besteht aus 4 unterschiedlichen Arten von Klassen.

Eine Input Klasse ist dafür zuständig, Musikstücke zu laden und abzuspielen. Sie kann entweder eine Datei zum Abspielen öffnen, oder sich mit einem Internet Radio verbinden.

Eine Effect Klasse kann auf das Musikstück unterschiedliche Effekte anwenden. Dies kann z.B. ein Equalizer, ein Flanger oder ein Echo Effekt sein.

Eine Output Klasse symbolisiert den Lautsprecher und ist dafür zuständig, das abgespielte Musikstück hörbar zu machen und die Lautstärke anzupassen.

Die Controller Klasse ist die einzige Klasse, die nicht direkt mit dem Musikstück interagiert. Sie kann lediglich die anderen Klassen verändern, z.B. die Intensität eines Filters kontinuierlich ändern. Auch kann es eine Implementation geben, welche Musikstücke (Input) Synchronisiert.

Die Musik kann hierbei durch die Klassen geroutet werden. Sie startet in einem Input, kann durch beliebig viele Effects gehen und kommt schließlich bei einem Output an. Hierbei kann der Weg nicht verzweigt werden. Ein Controller hingegen kann beliebig viele andere Klassen beeinflussen.
Nehmen wir
$I = \{i | i\ ist\ Input\},\ E = \{e | e\ ist\ Effect\},\ O = \{o | o\ ist\ Output\}\ und\ C = \{c | c\ ist\ Controller\}$, so kann man die möglichen Verbindungen als Relationen beschreiben:
Für den Musikfluss: $R_1 \subseteq (I \cup E) \times (E \cup O)\ mit\ R_1\{(x,y) | x \neq y \wedge \nexists\ (x,z) \in R_1 : y \neq z\}$
(Es gibt keinen Musikfluss, wo Start und Ziel gleich sind und es gibt keine Verzweigungen)
und für die Controller: $R_2 \subseteq C \times (E \cup I \cup O \cup C)\ mit\ R_2\{(x,y) | x \neq y\}$
(Kein Controller ist mit sich selbst verbunden)
Wichtig hierbei ist, dass je nach Realisation noch weitere Einschränkungen bestehen können. So kann ein Synchronizer nur mit Musikstücken (Input) verbunden werden, nicht aber mit anderen Klassen.

3.2.1.2 Technisch

Der gewählte Architekturstil ist ein Aktorensystem (vgl. 3.2.3), damit eine einfache Erweiterbarkeit gewährleistet ist, wobei einige Teile durch die verwendeten Frameworks (Surface SDK, XNA, BASS.NET) vorgegeben sind.

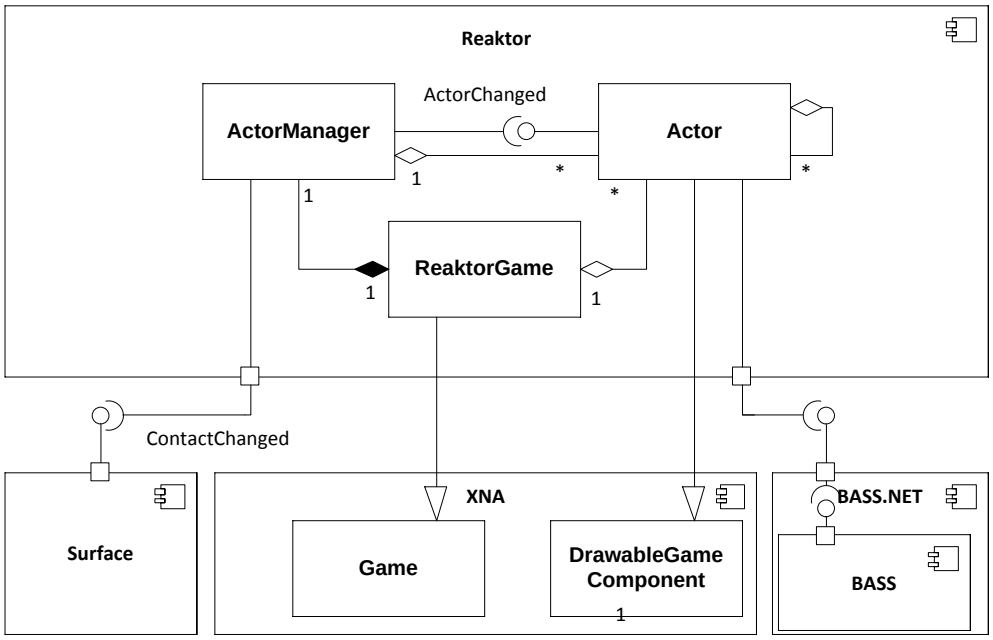

Abbildung 10 Technische Architektur (UML Diagramm)

Als Einstiegspunkt dient die ReaktorGame Klasse. Diese ist für die Initialisierung zuständig und hält alle Aktoren in der GameComponent Auflistung, sodass die Update und Draw Methoden der Aktoren regelmäßig aufgerufen werden, damit diese z.B. sich selbst auf dem Bildschirm zeichnen können. Dies ist nötig, da das XNA Framework diese Architektur vorsieht.

Die ActorManager Klasse ist dafür zuständig, Eingaben vom Surface SDK entgegen zu nehmen und an die betroffenen Aktoren weiterzuleiten.

Die Aktoren können mit den Informationen der ActorManager Klasse nun autark handeln. Sie bestimmen selbst, was sie mit den Informationen tun. Die komplette fachliche Architektur befindet sich in dieser Klasse und davon abgeleiteten Klassen. Ein Aktor, der ein Musikstück symbolisiert, wird bei einer Drehung das Tempo anpassen, wohingegen ein Aktor, der einen Lautsprecher symbolisiert, bei derselben Bewegung die Lautstärke ändern wird. Aktoren kennen darüber hinaus ihre Nachbarn in einem Umfeld von 200px und können auch hier selbst entscheiden, mit welchen sie sich verbinden.

Für die Audiomanipulation und –wiedergabe ist das BASS.NET Framework zuständig. Alle Aufrufe, welche eine Veränderung des Audiosignals zur Folge haben, werden an diese Komponente weitergereicht. Da BASS.NET nur ein Wrapper für die in C geschriebene BASS Bibliothek ist, werden auch hier alle Aufrufe weitergereicht. Daraus ergibt sich auch, dass die fachliche Architektur, welche vorsieht, Musik durch Actorklassen zu routen technisch nicht umgesetzt wird, weil die BASS Bibliothek dies nicht zulässt, da alle Musikverarbeitung intern geschieht.

3.2.3 Aktorensystem

Um den Würfeln, welche auf das Microsoft Surface gelegt werden können auch ein Verhalten zu geben, wurde ein Aktorensystem realisiert.

Die Aktoren sind in vier Kategorien aufgeteilt. Ein Aktor kann entweder Input, Effect, Output, oder Controller sein. Jeder Aktor hat eine Position und Ausrichtung, welche die Position des zugehörigen Würfels auf dem Microsoft Surface wiederspiegelt. Des Weiteren hat jeder Aktor einen internen und einen externen Wert. Der interne Wert wird durch die Drehung des Würfels bestimmt, der externe kann durch andere Aktoren beeinflusst werden. Außerdem hat jeder Aktor einen Input und einen Output, mit welchen sich andere Aktoren verbinden können.

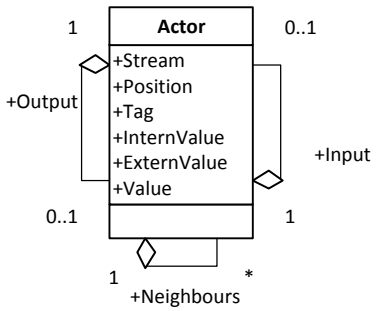

Abbildung 11 Aktor (UML DIagramm)

Es gibt eine zentrale Komponente (ActorManager), welche alle Eingaben auf dem Microsoft Surface entgegennimmt. Ist der Aktor, welcher mit dem Tag assoziiert ist noch nicht geladen, so passiert dies. Anschließend werden die Position und Richtung an den jeweiligen Aktor weitergeleitet. Außerdem wird allen in der Umgebung befindlichen Aktoren mitgeteilt, dass sich ein Aktor bewegt hat. Für jede darüber hinausgehende Verarbeitung sind nun die Aktoren selbst zuständig.

Jeder Aktor hat nun zu prüfen, ob sich etwas an sich selbst, oder seinen Nachbarn geändert hat und kann entsprechend darauf reagieren.

Beispielhaft sei hier die Suche eines Aktors nach einem neuen Input aufgeführt, welche ein Aktor immer dann ausführt, nachdem sich seine Nachbarn (entweder durch eigene, oder durch Bewegung eines anderen Aktors) verändert haben, womit dies eine der Hauptaufgaben eines Aktors ist:

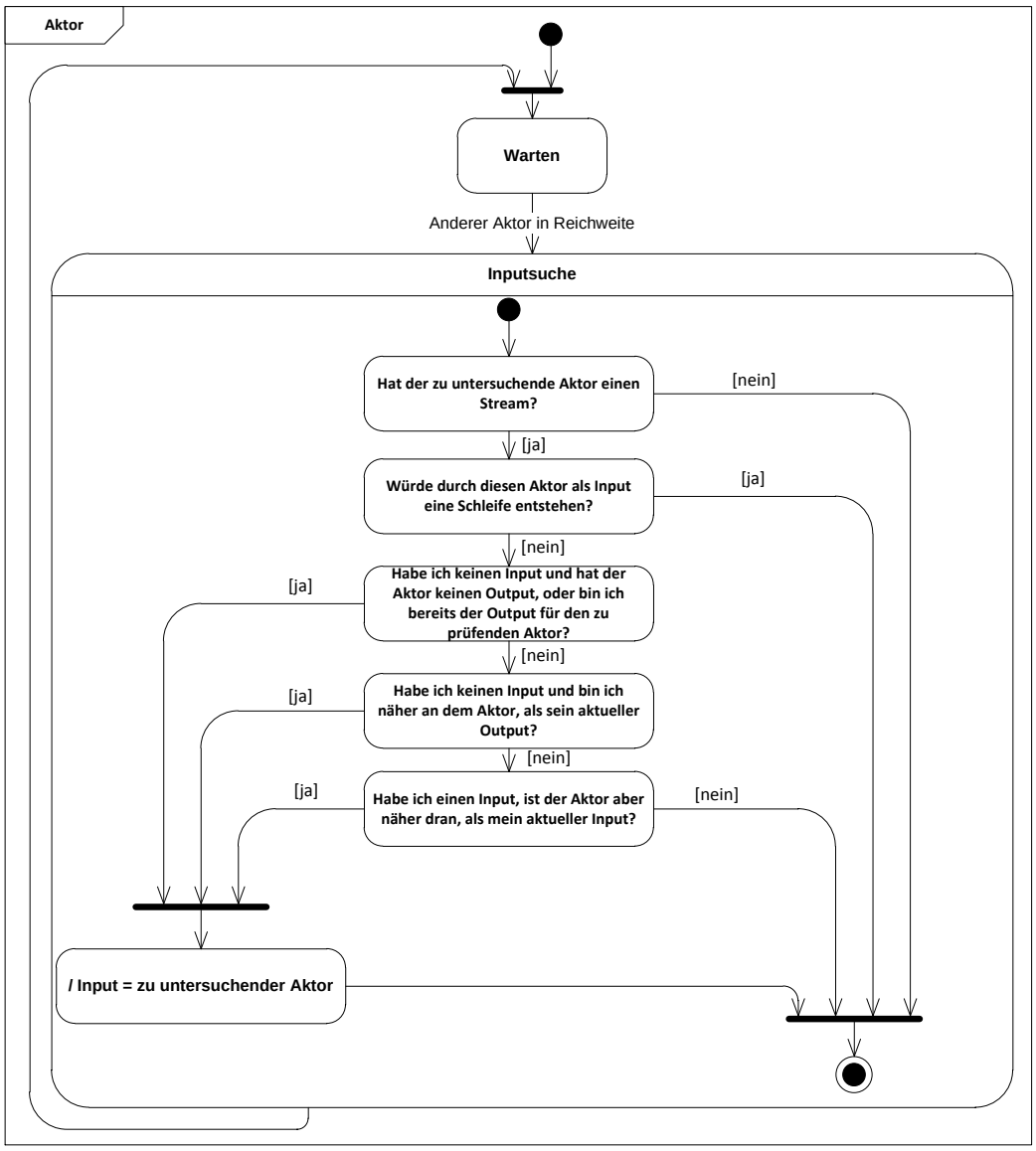

Abbildung 12 Inputsuche (UML Diagramm)

Jedes Plug-In kann hierzu noch weitere Verfeinerungen hinzufügen, oder auch jegliche Beschränkung aufheben. So kann ein Aktor, welcher ein Input Plug-In ist keinen Input haben, da er nur einen Output bereitstellt, seine Quelle aber z.B. eine Datei ist. Umgekehrt kann ein Output Plug-In keinen Output haben, da es die Musik, welche es über seinen Input entgegennimmt bereits z.B. über einen Lautsprecher ausgibt.

Dieses Aktorensystem gibt Plug-In Entwicklern die Möglichkeit, möglichst frei und unkompliziert neue Funktionen hinzuzufügen, da sie entweder mit wenig Aufwand einfache Plug-Ins schreiben können, indem sie schon vorhandene Basisimplementationen nutzen, oder auch Aktoren von Grund auf neu erstellen können.

4. Realisierung

Die folgenden Abschnitte beschreiben die im Rahmen der Arbeit entstandene Umsetzung der Software. Alle im Konzept beschriebenen Komponenten wurden auch realisiert. Bei der folgenden Beschreibung der Realisierung soll der Fokus nicht auf einzelnen Implementierungsdetails, sondern auf der Umsetzung des Gesamtkonzepts, liegen.

4.1 Verwendete Werkzeuge und Frameworks

Folgende Werkzeuge und Frameworks wurden verwendet:

- Programmiersprache: C# (4.0)
- IDE: Microsoft Visual Studio (2010 SP1)
- Microsoft Surface SDK (1.0 SP1) zum Ansprechen des Microsoft Surface Systems.
- Microsoft XNA (4.0) für die grafische Ausgabe
- BASS.NET (2.4.7.1) für die Audiomanipulation und –wiedergabe

4.2 Interaktion

Aufgrund der Haptik wurde sich dafür entschieden, auf Würfel als Bedienung zu setzen. Diese Würfel werden auf der Unterseite mit einem Identity Tag beklebt, sodass das Microsoft Surface sie erkennen kann. Auf die Oberseite kann ein Bild geklebt werden, welches eine visuelle Repräsentation bietet. Dies kann z.B. das Cover des Liedes sein, welches der Würfel repräsentieren soll, damit der Benutzer weiß, wofür der Würfel da ist. Die Würfel haben eine Kantenlänge von 1,125 Zoll, welches der Größe entspricht, die ein Identity Tag haben muss, um vom Microsoft Surface erkannt zu werden. Eine mögliche Interaktion mit Fingern wurde, wie bereits im Konzept beschrieben, nicht implementiert, sodass eine Interaktion nur durch Positionierung und Rotation der Würfel möglich ist. Dies soll eine möglichst einfache und intuitive Bedienung gewährleisten.

Dabei wurden die in 3.1.1 beschriebenen Konzepte der Reaktion von Aktoren auf Drehung und Bewegung dieser Würfel umgesetzt.

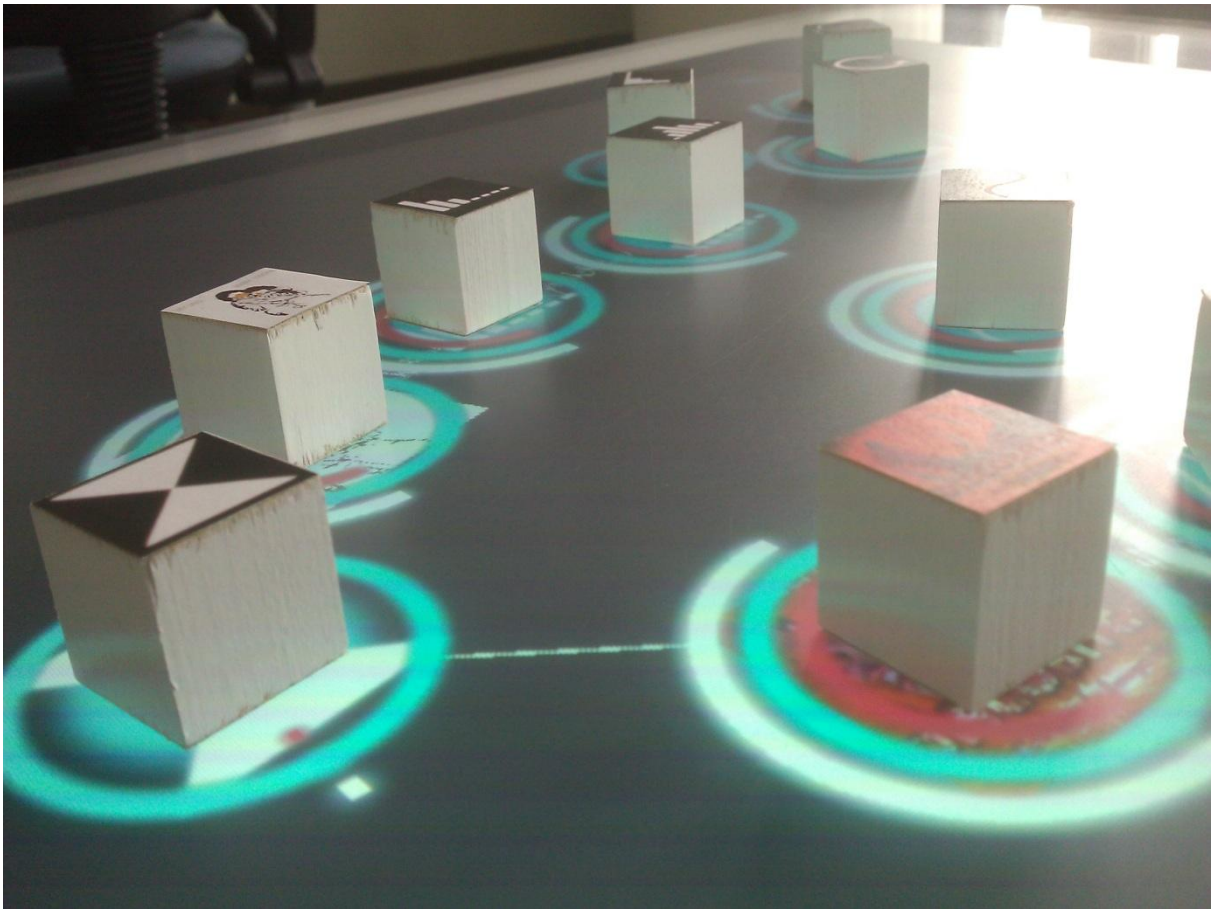

Abbildung 13 Würfel auf dem Microsoft Surface (Foto)

4.3 Konfigurationsoberfläche

Da einige Dinge, wie z.B. Konfigurationseinstellungen nur sehr schwer als mit Würfeln bedienbares Programm auf dem Microsoft Surface umgesetzt werden können, wurde zusätzlich ein Konfigurationsprogramm entwickelt, welches mit Maus und Tastatur bedient wird. Dieses Konfigurationsprogramm bietet die Möglichkeit, Plug-In Einstellungen anzupassen und auch die Entsprechenden Tags und Bilder für die Würfel zu drucken. Die Konfigurationsoberfläche lädt hierfür die in 3.1.2 beschriebene XML Datei und zeigt alle darin enthaltenen Informationen grafisch an. Mögliche Konfigurationseinstellungen werden automatisch per Reflection aus den Konstruktoren der Plug-Ins ausgelesen. Um außerdem die passenden Bilder anzuzeigen, ist es nötig, das eigentliche Programm zu laden, ohne dass eine Grafikausgabe erfolgt. Da diese Möglichkeit in keiner Dokumentation erwähnt wird, sei sie hier kurz erläutert:

1. Erzeugen einer neuen Instanz der Game Klasse

```
var game = new MyGameClass();
```

2. Den GraphicsDeviceManager Initialisieren

```
game.Graphics.ApplyChanges();
```

Neben dem Anpassen der Konfiguration und dem Drucken von Plug-Ins ist es auch möglich, BPM und Beatgrid einer Musiksammlung von Native Instruments Traktor zu importieren. Hierzu muss einfach der Speicherort der Collection.nml Datei angegeben werden (Normalerweise „Eigene Dokumente\Native Instruments\Traktor <Version>\Collection.nml“). Da diese Datei im XML Format gespeichert ist, kann sie automatisch ausgelesen werden. Anhand der Dateinamen wird dann verglichen, ob eine Übereinstimmung vorliegt und gegebenenfalls die Werte importiert.

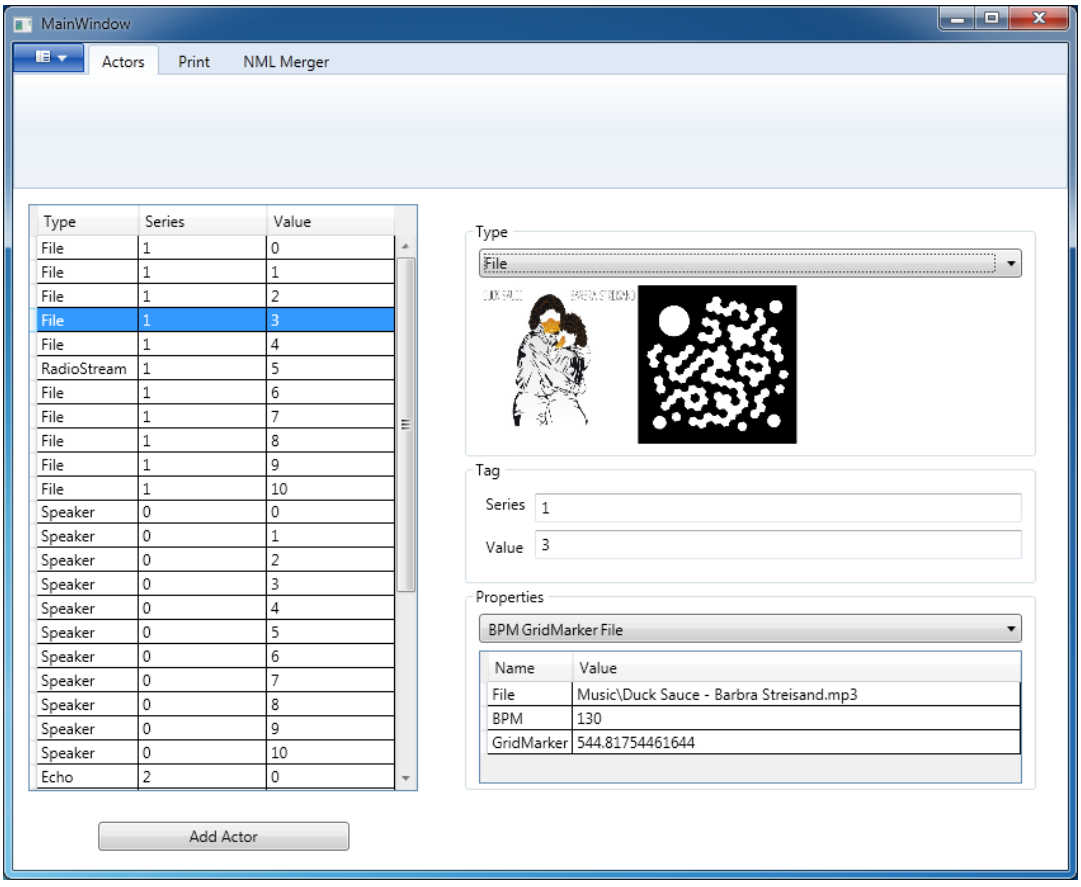

Abbildung 14 Konfigurationsoberfläche (Screenshot)

4.4 Plug-In System

Um ein einfaches Plug-In System zu realisieren, werden Basisklassen bereits mitgeliefert. Diese enthalten Standardimplementierungen von Aktoren und den daraus abgeleiteten möglichen Aktorarten Input, Effect, Output und Controller. Plug-In Entwickler können sich also entscheiden, ob sie einfache Plug-Ins auf die Basisklassen aufsetzen, oder komplexe Plug-Ins von Grund auf neu entwickeln wollen.

Folgende Interfaces und Klassen (hierarchisch geordnet) werden mitgeliefert und können durch Entwickler genutzt werden:

Interfaces

- IActor
  - IInput
  - IOutput
  - IEffect
  - IController

Klassen

- Actor
  - Input
  - Effect
  - Controller
    - ValueController

Um ein fertiges Plug-In vollständig in das System zu integrieren, reicht es die kompilierte .dll Datei in das „Actors“ Unterverzeichnis zu kopieren. Eine weitere Einordnung in die Unterverzeichnisse „Controller“, „Effect“, „Input“, „Output“ dient rein zur Übersicht und ist nicht zwingend notwendig. Hat der Entwickler keine eigene Routine geschrieben, um eine bildliche Repräsentation des Plug-Ins auf dem Microsoft Surface anzuzeigen, kann er außerdem ein Bild mit demselben Namen, wie seine Klasse, in das Verzeichnis kopieren, welches automatisch geladen und angezeigt wird, sofern er die Basisimplementation eines Aktors als Ausgangspunkt für sein Plug-In genutzt hat.

Das Plug-In erscheint daraufhin in der Konfigurationsumgebung und kann einem Tag zugewiesen und mögliche Parameter konfiguriert werden. Die Parameter werden hierbei automatisch aus den erstellten Konstruktoren der Klassen in dem Plug-In extrahiert. Gespeichert werden die hier vorgenommenen Anpassungen in der in 3.1.2 beschriebenen Tags.xml.

Ist das Plug-In nun mit einem Tag assoziiert und wird dieser auf den Tisch gelegt, wird das Plug-In mittels Reflection geladen und es werden die ihm zugewiesenen Parameter übergeben.

4.5 Audiomanipulation und Ausgabe

Für die Audiomanipulation und Ausgabe wird auf die BASS Engine gesetzt, für welche ein .NET Wrapper (BASS.NET) erhältlich ist, wodurch eine einfache Einbindung gewährleistet ist. Die BASS Engine ist in C geschrieben und bietet reichhaltige Funktionen zur Audiomanipulation und Wiedergabe. Da die BASS Engine nicht objektorientiert arbeitet, sondern eine flache Fassade bietet, geschieht die Audiomanipulation hier anders, als es dem Benutzer auf dem Bildschirm suggeriert wird. Jeder Audiostream wird in der Engine durch einen Integer Wert repräsentiert und es können diverse Methoden mit diesem Wert aufgerufen werden, die diesen Audiostream beeinflussen. Um dieses Konzept auf das objektorientierte Konzept des Aktorensystems zu übertragen, wurde jedem Aktor das Integer Feld „Stream“ hinzugefügt, welches den Wert des Streams in der BASS Engine wiederspiegelt. Dieses Feld wird entweder gesetzt, z.B. von einem Input, welcher eine Musikdatei lädt, oder Rekursiv berechnet (Jeder Aktor hat den Stream seines Vorgängers). Jedem Aktor stehen nun alle von der BASS Engine angebotenen Funktionen zur Verfügung und er kann sie mit dem gegebenen Audiostream benutzen.

4.6 Test

Es wurden zwei Tests durchgeführt. Der eine Test erfolgte mit einer Person, welche kein DJ ist, der andere erfolgte mit einem DJ.

Den Testern wurde eine kurze Einführung gegeben, dass Sie die Steine auf das Display zu legen haben und diese sich verbinden, sobald sie nahe genug aneinander sind. Auch wurde ihnen gesagt, dass man die Steine drehen kann. Die Bedeutung der unterschiedlichen Würfel (Musikstück, Lautsprecher, etc.) wurde nicht erklärt und sollte von allein herausgefunden werden.

Die Testperson, welche kein DJ ist, brauchte etwa 10 Minuten, um sich komplett mit dem Programm zurechtzufinden. Am längsten hat es hierbei gedauert, die einzelnen Funktionen der Würfel und der Drehungen herauszufinden. Allerdings wurden Funktionen, welche Musikstücke synchronisieren und das Mixen von Musikstücken nicht sehr ausgiebig genutzt. Es wurden eher einzelne Musikstücke hintereinander abgespielt und ausprobiert, was die unterschiedlichen Effekte bewirken.

Die Testperson, welche ein DJ ist, fand sich noch schneller mit dem Programm zurecht, als die erste Testperson. Hier wurde allerdings mehr versucht, mehrere Musikstücke ineinander zu mixen und der Umgang mit dem Programm erschien routinierter und nicht so experimentierend, wie bei der ersten Testperson. Bemängelt wurden einige fehlende Funktionen, wie das Vor- und Zurückspulen in Musikstücken oder das Scratchen. Auch wurden mehr Effekte gewünscht. Kritisiert wurden auch die teilweise hohen Latenzen und die fehlende Möglichkeit, Kopfhörer zum Vorhören zu benutzen.

Der Anspruch einer einfach zu erlernenden Bedienung konnte durch die Tests bestätigt werden. Die noch fehlenden Funktionen können durch das Plug-In System später noch nachgeliefert werden. Die Latenzen sind hingegen nicht einfach behebbar, da das Surface System einige Zeit braucht, um Eingaben auf dem Display zu verarbeiten. Eine geringere Letenz der Audiowiedergabe könnte noch dadurch erreicht werden, indem man eine externe, ASIO fähige Soundkarte installiert. Hierdurch könnte auch die fehlende Vorhörfunktion integriert werden, wenn die Soundkarte mindestens 2 Stereoausgänge bietet.

4.7 Umsetzung der Anforderungen

Im Folgenden soll die Umsetzung der in 2.3 aufgestellten Anforderungen untersucht werden.

- Echtzeit: Da das Microsoft Surface standardmäßig keine Soundkarte mit ASIO Treiber installiert hat, können keine sehr geringen Latenzen garantiert werden.
- Robustheit: Die BASS Engine spielt die Musik auch weiter ab, wenn das Hauptprogramm einfriert. Wird das Hauptprogramm beendet, so wird auch die BASS Engine und somit die Musikausgabe beendet.
- Visuelles Feedback: Durch die Würfel und die visuelle Repräsentation derer und der Musik ist ein gutes visuelles Feedback gegeben.
- Mixen: Es können so viele Musikstücke gleichzeitig abgespielt werden, wie Identity Tags von dem Microsoft Surface System gleichzeitig erkannt werden. Dies entspricht mehr als 10 Musikstücken gleichzeitig und ist ausreichend.
- Tempo und Beat Synchronisation: Es wurde ein Sync Controller realisiert, welche alle Musikstücke in seiner Umgebung in Tempo und Beat synchronisiert.
- Effekte: Es wurden ein 3 Band Equalizer, ein Flanger und ein Echo Effekt realisiert. Aufgrund des Plug-In Systems ist es jedem Entwickler möglich, hier weitere Effekte zu programmieren.
- Loops: Ein Loopstein bietet die Möglichkeit, Loops der Größen 1/8 – 32 Beats zu benutzen. Kleinere Werte führten zu einem Absturz der BASS.NET Engine, sofern diese kurz vorher oder nachher nach der Waveform des Musikstücks gefragt wurde, welche als visuelle Repräsentation der Verbindung zwischen Aktoren benutzt wird.
- Haptik: Durch die Verwendung von Würfeln ist eine sehr gute Haptik gegeben. Allerdings ist man durch die Verwendung von Würfeln auch beschränkt. So ist es z.B. nicht möglich mit diesen zu scratchen.
- Musikverwaltung: Die Musikverwaltung ist im Gegensatz zu konventioneller DJ Software, wie Traktor Pro 2 komplizierter, allerdings einfacher, als die des Reactable. Neue Musikstücke müssen in der Konfigurationsoberfläche mit dem Identity Tag des passenden Würfels assoziiert werden. Die Musikstücke müssen hierfür weder ein spezielles Format, haben, noch anderweitig bearbeitet werden.

Es wurden nicht alle Anforderungen komplett umgesetzt. Durch das Plug-In System kann allerdings vieles noch im Nachhinein umgesetzt werden, ohne dass Änderungen am Hauptprogramm nötig werden.

5. Zusammenfassung und Ausblick

Die Arbeit hat gezeigt, dass es möglich ist, DJ Software auf Multitouch Eingabesystemen zu realisieren. Die meisten in 4.7 genannten negativen Punkte, lassen sich beseitigen, wobei es schwer ist, die Haptik eines Controllers zu erreichen, da Fader und Plattenteller nur schwer darzustellen sind, wenn diese nicht nur einfach virtuell auf dem Bildschirm vorhanden sind.

Ein weiteres Problem ist die Größe von Musiksammlungen. Diese beträgt oftmals viele 1000 Musikstücke. Nimmt man nun für jedes Musikstück einen Würfel, wie in dieser Arbeit gezeigt, kommt man auf genauso viele Würfel. Diese sind allerdings nicht so leicht durchsuchbar, wie eine Musikbibliothek auf dem PC, weshalb es schwer ist, ein spezielles Musikstück zu finden. Eine Möglichkeit, dies zu umgehen, wäre das dynamische Mapping von Würfel zu Musikstück. Hierbei muss man entweder den Kompromiss eingehen, dass die Würfel kein Bild des jeweiligen Musikstücks zeigen, oder man müsste kleine Bildschirme integrieren, welche dann dynamisch das Bild des jeweiligen Musikstücks anzeigen, wobei sich hier wiederum die Frage der Realisierbarkeit stellt.

Die beste Bedienung bietet zurzeit noch konventionelle DJ Software mit einem oder mehreren Controllern. Hierdurch kann der DJ sein Setup individuell konfigurieren. Auch ist die Haptik von Controllern bisher von Tangible User Interfaces noch nicht erreicht. Ein mögliches Einsatzgebiet wäre hierbei, konventionelle DJ Software mit einem Tangible User Interface als Controller zu kombinieren und so die Vorteile beider Welten zu vereinen.

Zusammenfassend lässt sich sagen, dass DJ Software auf Multitouch Eingabegeräten durchaus eine Zukunft hat, konventionelle DJ Software aber voraussichtlich nicht ablösen wird.

Literaturverzeichnis

[Abb mixiTUI] Abbildung mixiTUI – URL http://www.mixitui.com/speciale.pdf S. 46 – Abruf 20.06.2011

[Abb Reactable] Abbildung Reactable – URL http://www.reactable.com/files/reactable_pictures_xavier_sivecas.zip – Abruf 20.06.2011

[Abb Surface] Abbildung Microsoft Surface – URL http://technet.microsoft.com/en-us/library/ee692114(Surface.10).aspx – Abruf 20.06.2011

[Abb Vectorform] Abbildung Vectorform – Surface DJ – URL http://apps.vectorform.com/wp-content/files_flutter/1280875644sdj_MusicPlaying.png – Abruf 20.06.2011

[Andre 2010] Alexis Andre: OtoMushi: Touching Sound. In SIGGRAPH Asia 2010, December 15 – 18, 2010, Seoul, South Korea

[Beamish et. al. 2004] Timothy Beamish, Karon Maclean, Sidney Fels: Manipulating Music: Multimodal Interaction for DJs. In CHI 2004, 24-29 April, Vienna, Austria.

[Birsan 2005] Dorian Birsan: On Plug-ins and Extensible Architectures. In ACM Queue March 2005, März 2005.

[Golden 2011] Ean Golden: Dj TT Survey Results and S4 Winner Announced. – URL http://www.djtechtools.com/2011/01/27/dj-tt-survey-results-and-s4-winner-announced – Abruf: 11.05.2011

[Havir, Engalan 2009] Eric Havir, Joe Engalan: Can anyone be a DJ? – URL http://blogs.msdn.com/b/surface/archive/2009/07/03/can-you-be-a-dj.aspx – Abruf: 15.04.2011

[Jordà 2010] Sergi Jordà: The Reactable: Tangible and Tabletop Music Performance. In CHI 2010: Media Showcase Session 1 April 10–15, 2010, Atlanta, GA, USA

[NI 2011] Native Instruments: Traktor – URL http://www.native-instruments.com/traktor –Abruf: 17.04.2011

[Pedersen, Hornbæk 2009] Esben Warming Pedersen, Kasper Hornbæk: mixiTUI: A Tangible Sequencer for Electronic Live Performances. In Proceedings of the Third International Conference on Tangible and Embedded Interaction, Feb 16-18 2009, Cambridge, UK

[Reactable 2011] Reactable Systems: Reactable – URL http://www.reactable.com – Abruf: 15.04.2011

[Surface 2011] Microsoft Technet: Microsoft Surface 1.0 SP1 Administration Guide. – URL http://technet.microsoft.com/en-us/library/ee692060(Surface.10).aspx – Abruf 17.05.2011

[wertzui] Steckbrief wertzui – URL http://wertzui.animexx.jp – Abruf 20.06.2011

Abbildungsverzeichnis

Tabellenverzeichniss

Glossar

ASIO: Audio Stream Input/Output bezeichnet ein von Steinberg entwickeltes Audiotransferprotokoll. Es bietet sehr geringe Latenzzeiten.

Beat: Taktschlag in einem Musikstück

Beatgrid: Ein regelmäßiges Raster, welches über ein Musikstück gelegt wird und eine Trennung durch Beats wiederspiegelt. Kann durch einen Beat und die BPM berechnet werden.

BPM: Beats per minute, Schläge pro Minute, gibt Geschwindigkeit der Musik an.

Cue: Eine Markierung in einem Musikstück, zu welcher normalerweise per Tastendruck gesprungen werden kann.

Equalizer: Ein Equalizer ist ein Gerät oder eine Softwarekomponente, welche die Lautstärke einzelner Frequenzbereiche verändern kann.

HID: Human Interface Device bezeichnet eine Klasse von USB Geräten, mit welchen der Benutzer direkt interagieren kann. DJ Controller werden oft entweder als HID oder als MIDI Controller angesprochen.

Identity Tag: Eine Art zweidimensionaler Barcode, welcher eine 128 Bit Zahl repräsentiert. Von Microsoft zum Einsatz mit dem Microsoft Surface entwickelt. Programmtechnisch gliedert sich der Tag in wie 64 Bit Zahlen, der Series und der Value. Das Microsoft Surface System kann diese Tags erkennen, wenn sie auf dem Tisch liegen und dem Programmierer Auskunft über Position, Ausrichtung, Series und Value geben.

Lag: Siehe Latenz.

Latenz: Als Latenz wird die Verzögerung zwischen Eingabe des Benutzers und der Reaktion der Software auf diese Eingabe bezeichnet. Oft wird auch nur von Lag gesprochen, wenn diese Verzögerung merkbar ist.

Loop: Ein sich wiederholendes Stück Musik. Typische Längen reichen von 1/32 bis 32 Beats.

MIDI: Musical Instrument Digital Interface bezeichnet ein Protokoll, welches oft von Elektronischen Instrumenten und auch von DJ Controllern zur Datenübertragung genutzt wird.

Pitch: Als Pitch wird traditionell die Tonhöhe bezeichnet. Möchte man bei einer Schallplatte die Tonhöhe ändern, so kann man diese schneller oder langsamer laufen lassen. Daher wird heutzutage mit Pitch auch oft das Tempo eines Musikstückes bezeichnet.

Reflection: Reflection bezeichnet die Fähigkeit eines Programms seine eigene Struktur zu kennen, zu analysieren und zu modifizieren. Auf diese Weise können dynamisch neue Teile eines Programms geladen werden.

Sample: Ein sehr kurzes (meistens 1-32 Beat langes) Musikstück, welches sich immer wiederholt.

Sequenzer: Ein Sequenzer bezeichnet ein Gerät oder eine Software zur Erstellung von Musiksequenzen.

Tag: Siehe Identity Tag.

Tangible User Interface: Bezeichnet ein Interface, bei dem der Benutzer etwas anfassen kann und nicht nur einen Bildschrim berührt.

Time-Code-System: Bei diesen Systemen wird ein Plattenspieler zur Kontrolle einer DJ Software eingesetzt. Dabei werden spezielle Platten benutzt, welche keine Musik, sondern ein Time-Code-Signal haben. Dieses wird nun über einen Eingang einer Soundkarte in den PC gespeist. Die Software wandelt dieses Signal nun in Befehle um, sodass die Geschwindigkeit der Platte und die Position der Nadel auf der Platte erkannt werden kann. Es dient dazu, Musikstücke auf dem Computer mit der gewohnten Haptik des Plattentellers zu verbinden.

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung (APSO-TI-BM vom 16.11.2006) nach §16(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 01.08.2011 _____

Ort, Datum Philipp Kühn