Studien- und Prüfungsabteilung
Leonhardstraße 15, A-8010 Graz
Tel: 0043(0)316/389-1310, -1313, Fax: 0043(0)316/389-1311
Mail:studienabteilung@kug.ac.at

Bitte deutlich leserlich ausfüllen!

Deckblatt einer

w i s s e n s c h a f t l i c h e n Bachelorarbeit

Vor- und Familienname	Matrikelnummer
Anna-Katharina Maly	0873097
Studienrichtung	Studienkennzahl
Elektrotechnik - Toningenieur	V 033 213

Thema der Arbeit:

Wellenklänge - Computermusikinstrument mit interaktiver Wellensimulation und Klangsynthese

Angefertigt in der Lehrveranstaltung: Musikinformatik 01
(Name der Lehrveranstaltung)

Vorgelegt am: 05.07.2012
(Datum)

Beurteilt durch: Ao. Univ. Prof. Dipl.-Ing. Winfried Ritsch
(LeiterIn der Lehrveranstaltung)

Wellenklänge -

Computermusikinstrument mit interaktiver

Wellensimulation und Klangsynthese

Bachelorarbeit von

Anna Maly

Datum: 05.07.2012

Institut: Institut für elektronische Musik und Akustik

Angefertigt in der Lehrveranstaltung: Musikinformatik 01 / SS 2011

Betreuer: Ao. Univ. Prof. Dipl.-Ing. Winfried Ritsch

ANNA – KATHARINA MALY
(Name in Blockbuchstaben)

0873097
(Matrikelnummer)

Erklärung

Hiermit bestätige ich, dass mir der *Leitfaden für schriftliche Arbeiten an der KUG* bekannt ist und ich diese Richtlinien eingehalten habe.

Graz, den …….

……
Unterschrift der Verfasserin / des Verfassers

Kurzbeschreibung

Es soll ein Instrument entwickelt werden, das durch Anregung von Oberflächenwellen Klänge und Klangmuster erzeugt. Die Idee war es Wellen zu erzeugen, die entstehen, wenn man einen Stein ins Wasser wirft, die also Kreise ziehen, und dem entsprechende musikalische Muster zu entwerfen. Diese Klangerzeugung soll vom elektronischen Musiker noch bearbeitet und durch unterschiedliche Klangsynthesen ersetzt werden können. Diese Arbeit befasst sich mit der Erarbeitung und der Entstehung dieses Instruments, der Verwendung der grafischen Programmiersprache Pure Data [Kapitel 3] zur Implementierung des physikalischen Modells, sowie auch mit dem visuellen 3D-Feedback unter Verwendung von Open-GL Erweiterung GEM [Kapitel 2.4.4].

Abstract

I want to develop an instrument, that creates sounds and sound patterns by the excitation of surface waves. The idea was to generate waves that occur when somebody throws a stone into water which creates circles and then to design corresponding musical patterns.The electronic musician can edit the sound generation and replace the different sound syntheses through others.
This work deals with the development of this instrument, working with the graphical programming language Pure Data [chapter 3] to implement the physical model, as well as with the 3D visual feedback, which is realized by the open-GL extension GEM [chapter 2.4.4] .

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Idee

Das Wellengesteuerte Klanginstrument soll aus drei Teilen bestehen.

- Wellensteuerung
- Instrument (z.B: Metallophon)
- Hardware-Steuerung

Die **Wellensteuerung** besteht aus einem virtuellen Wasserbecken, in dem ein Netz von Messpunkten verankert ist. Jeder Messpunkt entspricht einem Parameter des Klanginstruments. Die virtuelle Wasseroberfläche muss durch eine äußere Einwirkung in Bewegung gesetzt werden können, was intuitiv zum Beispiel einem virtuellen Steinwurf ins Wasserbecken entsprechen könnte.

Das **Instrument** soll ein Phantasieinstrument sein, dessen Klang länger anhalten kann. Der Klang soll mit Wasserwellen assoziiert werden.

Mit der **Hardwaresteuerung** soll man den Steinwurf ins Wasser simulieren können.

1.2 Problemstellung

1.2.1 Wellensteuerung

Es gilt herauszufinden, wie eine Wasseroberfläche bzw. Flüssigkeitenoberläche simuliert oder gemessen werden kann, dass das Ergebnis auch unverkennbar mit einer Flüssigkeitenoberfläche assoziiert wird.

1.2.2 Instrument

Der Klang des Instruments spielt eine entscheidende Rolle. Es wird von Anfang an davon ausgegangen, den Klang mit Pure Data synthetisch zu erzeugen. Es muss ein Mittelweg gefunden werden zwischen einfacher Bedienung und nicht zu primitiv in der Klangqualität. Und dabei soll der Klang in direktem, assoziativen Zusammenhang mit den Wellen der Wellensteuerung stehen.

1.2.3 Hardware-Steuerung

Es soll eine Hardware-Steuerung gefunden werden, mit der das Spielen des Instruments einfach und intuitiv erlernt werden kann. Um den Rahmen der Arbeit nicht zu sprengen, soll die Entwicklung nicht zu aufwändig bzw. die Anschaffung nicht zu teuer werden.

1.3 Aufbau der Arbeit

Die Arbeit teilt sich auf in drei große Teile. Erster Teil ist die Wellensteuerung, in dem eine Lösung für die Simulation einer Wasseroberfläche bzw. dem Messen einer echten Wasseroberfläche gesucht wird.
Es folgt ein Abschnitt mit der Klangerzeugung in Pure Data.
Und der letzte Teil widmet sich der Hardwaresteuerung.
Am Schluss der Arbeit folgt eine Zusammenfassung und das Quellenverzeichnis.
Zeitlich entstand die Arbeit nicht in dieser Reihenfolge. Ich begann mit einer breiten

Recherche, in der viele Möglichkeiten in Betracht gezogen wurden. Die Recherche sollte in verschiedene Richtungen gehen. Erst danach sollte eine Entscheidung getroffen werden, wie das Instrument entstehen soll.

1.4 Ziel der Arbeit

Die Arbeit soll ein funktionierendes, wellengesteuertes Instrument hervorbringen, dessen Klang ohne Umwege optisch mit der Wellenausbreitung assoziiert wird. Der Weg dahin soll dokumentiert und nicht eingeschlagene Wege begründet werden. Dabei ist es mir wichtig meinen praktischen Umgang mit der Programmiersprache Pure Data zu verbessern, und nach Möglichkeit auch mit Physical Modelling zu arbeiten. Eine Hardwarelösung soll aufgezeigt, und auch realisiert werden.

2 Die Wellensteuerung

Meine Vorstellung der Wasseroberfläche sieht ungefähr aus wie in Abbildung 2.1.

Abbildung 2.1: Kapillarwellen, Bildquelle: [4]

Eine schöne Simulation davon kann man auf folgender Homepage finden: [12] Um das zu erreichen werde ich zuerst grundsätzliche Überlegungen zu Wasseroberflächen aufstellen. Erst dann überlege ich mir Lösungen, die für die Steuerung in Frage kommen.

2.1 Physikalische Überlegungen zu Wasseroberflächen

Die ungestörte Oberfläche einer Flüssigkeit ist eben und horizontal. Eine Störung der Oberfläche erzeugt eine Verschiebung aller Moleküle, die sich direkt unterhalb der Oberfläche befinden. Jedes Volumenelement der Flüssigkeit beschreibt dann einen geschlossenen Weg. Die Amplitude der horizontalen und der vertikalen Verschiebungen eines Volumenelements der Flüssigkeit verändern sich unter anderem mit der Tiefe. Direkt

an der Oberfläche beschreiben die Volumenelemente Kreise, in tieferen Schichten eher elliptische Bahnen. Am Grund können Moleküle natürlich nicht mehr vertikal verschoben werden, sie werden nur noch horizontal bewegt. An der Oberfläche einer Flüssigkeit wirken zusätzlich zum Luftdruck noch weitere Kräfte. Eine Kraft wird durch die Oberflächenspannung hervorgerufen. In einem Wellental bewirkt die Oberflächenspannung eine aufwärts gerichtete Kraft und in einem Wellenberg eine abwärts gerichtete Kraft auf ein Volumenelement. Dies ist vergleichbar mit dem Verhalten einer elastischen Feder – die Oberflächenspannung steht dabei für die Federkonstante D. Eine andere Kraft ist die Gewichtskraft der Flüssigkeit, die sich oberhalb der Lage der ungestörten Flüssigkeit befindet. [13]

2.1.1 Die Ausbreitungsgeschwindigkeit (Phasengeschwindigkeit)

$$v = \sqrt{\left(\frac{g\lambda}{2\pi} + \frac{2\pi\sigma}{\rho\lambda}\right) \cdot \tanh\left(\frac{2\pi h}{\lambda}\right)} \tag{2.1}$$

g... Fallbeschleunigung

λ... Wellenlänge

σ... Oberflächenspannung

ρ... Dichte

h... Wassertiefe

Man unterscheidet grundsätzlich zwischen drei Wellenarten.

- Wenn die Tiefe der Flüssigkeit sehr klein im Vergleich zur Wellenlänge ist, und die Oberflächenspannung vernachlässigbar ist, dann ergeben sich **Wellenwannen-Wellen**.

- Wenn die Tiefe der Flüssigkeit sehr groß im Vergleich zur Wellenlänge ist, und die Wellenlänge groß genug ist, dass der zweite Summand der Gleichung 2.1 vernachlässigt werden kann, dann ergeben sich **Schwerewellen**.

- Wenn die Tiefe der Flüssigkeit sehr groß im Vergleich zur Wellenlänge ist, und die Wellenlänge sehr klein, dann ergeben sich **Kapillarwellen**, welche den Kreisen

entsprechen, die von Beginn an erwünscht waren.

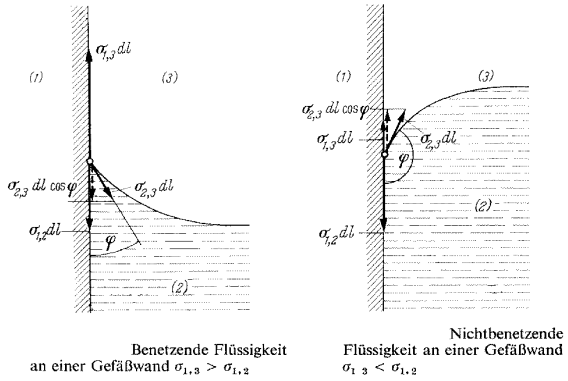
Wenn die Tiefe h der Flüssigkeit viel größer als die Wellenlänge λ ist, dann ist $\frac{2\pi h}{\lambda}$ sehr groß gegen 1, d.h. $\tanh\left(\frac{2\pi h}{\lambda}\right) \approx 1$. Ist die Wellenlänge sehr klein, sodass der zweite Summand dominiert, so ergibt sich die vereinfachte Phasengeschwindigkeit für Kapillarwellen.

$$v = \sqrt{\left(\frac{2\pi\sigma}{\rho\lambda}\right)} \tag{2.2}$$

Je größer also die Wellenlänge, desto langsamer breitet sie sich aus. Die Abhängigkeit der Phasengeschwindigkeit von der Wellenlänge wird als Dispersion bezeichnet. Die Dispersion ist für nicht-sinusförmige Wellen wie die Oberflächenwellen von Flüssigkeiten von Bedeutung, denn eine nicht-sinusförmige Welle besteht aus der Überlagerung von Sinuswellen mit verschiedenen Wellenlängen. **Die Eigenschaften der Kapillarwelle hängen am stärksten von der Oberflächenspannung der Flüssigkeit ab.** [13]

2.1.2 Die Oberflächenspannung

Die Oberflächenspannung von Wasser bei $20°C$ beträgt $\sigma = 0,07275N/m$. Jetzt ist noch interessant zu untersuchen, wie sich Wasser an einer Gefäßwand verhält. Moleküle an der Wand erfahren anziehende Kräfte von den Flüssigkeitsmolekülen und von den Molekülen in der festen Wand. Die Flüssigkeit bildet einen Randwinkel φ mit der Gefäßwand, den man ableiten kann, wenn man annimmt, dass auch an der Oberfläche fester Stoffe eine Oberflächenspannung herrscht. [5]



Benetzende Flüssigkeit an einer Gefäßwand $\sigma_{1,3} > \sigma_{1,2}$

Nichtbenetzende Flüssigkeit an einer Gefäßwand $\sigma_{1,3} < \sigma_{1,2}$

Abbildung 2.2: Flüssigkeit an Gefäßwand, Bildquelle: [5]

Das Kapillargesetz besagt:

$$\cos\varphi = \frac{\sigma_{13} - \sigma_{12}}{\sigma_{23}} \tag{2.3}$$

2.2 Mögliche Steuerungslösungen

Möglich wäre:

- Feder-Masse-System
- Physikalische Wasserwellenberechnung, siehe [14]
- Smoothed Particle Hydrodynamics, siehe [15]
- Echtes Wasser abtasten zum Beispiel mit Kinect oder CCD Sensor bzw. SCCD Sensor
- Trommelmembran anstatt Wasser [16]

2.2.1 Feder-Masse System

Der große Vorteil an der Oberflächensimulation durch ein Feder-Masse System ist der, dass es für Pure Data ein Tool gibt, das kurz PMPD genannt wird, das genau dafür da ist Feder-Masse Systeme einfach herzustellen und mit PD zu steuern. PMPD [17] heißt Physical Modelling in Pure Data. Damit kann man relativ einfach ein solches System simulieren und das in Echtzeit ohne allzu große Rechenleistung zu benötigen.

2.2.2 Physikalische Wasserwellenberechnung

Die physikalische Wellenberechnung wäre für das Echtzeitsystem zu aufwändig. Zudem werden keine exakten Werte für das Spielen des Instruments benötigt.

2.2.3 Smoothed Particle Hydrodynamics

Smoothed Particle Hydrodynamics arbeitet damit, Wassermoleküle und deren Interaktion mit den Wassermolekülen in der Nähe einzeln zu betrachten. Diese Möglichkeit ist auch viel zu aufwändig für unsere Problemstellung. Zudem ist eine Echtzeitimplementation nicht möglich.

2.2.4 Echtes Wasser

Ein echtes Wasserbecken abzutasten wäre sicher intuitiv, aber um eine Ebene exakt abzutasten wären mindestens drei Sensoren und einige Korrelationsberechnungen notwendig. Außerdem wäre man sehr unflexibel bezüglich den physikalischen Parametern.

2.2.5 Trommelmembran

Man könnte auch die Berechnungen einer Trommelmembran verwenden, die es schon zu Genüge gibt. Jedoch wären die Wellen ganz anders als die von Wasser.

2.2.6 Schlussfolgerung

Ich entscheide mich für das Feder-Masse-System, da es relativ einfach ist, dieses mit Pure Data unter Verwendung von PMPD zu realisieren und das beste Ergebnis verspricht.

2.3 Überlegungen zum Feder-Masse System

Da Kapillarwellen hauptsächlich von der Oberflächenspannung abhängen, können sie durch ein Feder-Masse System nachgebildet werden. Meine erste Intuition war, den Rand als Fixpunkt zu verwenden und innerhalb regelmäßig Massen zu verteilen, die mit Federn verbunden sind. Diese Methode hat, bei Betrachtung der Oberflächenspannung, wenig mit Kapillarwellen zu tun.
Die Oberflächenspannung ist jene Kraft, die die Moleküle über der Nullebene nach unten zieht, und die Moleküle unter der Nullebene nach oben zieht. Es muss also ein weiteres

Masse-Feder-Netz darüber gelegt werden, wo jede Masse nur mit der darüber liegenden verbunden wird. So kommt es auch weniger zu Transversalwellen und es müssten sich eher Kapillarwellen bilden.

2.3.1 Der Rand

Nehmen wir an, wir haben ein mit Wasser gefülltes Becken, das tief genug ist, damit Kapillarwellen entstehen können und keine Wannenwellen, und, dass die Wellenlänge klein genug ist, damit keine Schwerewellen entstehen. Dann werfen wir einen kleinen Stein hinein und es entstehen Wellen. Treffen diese Wellen auf den Rand, so kann man sich vorstellen, sie gehen einfach weiter, nur anstatt über den Rand hinaus, werden diese Wellen reflektiert. Siehe Simulation: [12]
Um das umzusetzen, wurden die äußersten Massen fixiert.

2.4 Realisierung

Für die Simulation der Wasseroberfläche erstelle ich ein Feder-Masse-Gitter in PMPD. [18]

2.4.1 Physical Modelling mit Pure Data

In PMPD arbeitet man mit Massen, Verbindungen und Interaktoren in 1D, 2D ud 3D. Massen haben ein Gewicht, eine Position und sie bewegen sich, wenn eine Kraft auf sie wirkt. Verbindungen sind die Interaktionen zwischen zwei Massen. Die Verbindungen erhalten die Positionen der anschließenden Massen. Sie berechnen die Kräfte der zwei Massen und weisen jeder Masse eine Kraft zu. Interaktoren sind eine Art Verbindung, aber beeinflussen mehr als nur zwei Massen. Die resultierenden Bewegungen sind sehr natürlich.
Wenn man Massen mit einer Bang-Nachricht ausliest, geben sie ihre Position aus. Wenn man Verbindungen ausliest, geben sie die auf die Masse wirkende Kraft aus.
In PMPD müssen keine Kräfte berechnet werden. Wir brauchen nur Eingangsparameter.

Die Simulation beschränkt sich auf die Flüssigkeitsoberfläche. PMPD verwendet keine Einheiten, sondern arbeitet mit Verhältnissen. Arbeitet man also mit Newton, so sollte man die Masse in Kilogramm angeben.
Parameter und Aktionen werden in PD mittels Nachrichten (Messages) an das Objekt geschickt. Massen und Verbindungen (Links) sind Objekte.

Massen

Eingangsparameter:

- setM: Gewicht der Masse
- setX: Position der Masse
- setXmin .., setXmax ..: Minimum und Maximum der Position

Aktionen:

- bang: Gib die Position aus!
- Zahl-Object: Füge der Masse Kraft zu!
- reset: In Ausgangsposition!
- resetF: Setze die Kräfte zurück!

Zusätzlich bei mass3D:

- setT: Threshold. Wenn sich die Masse bei einem Maximum- oder Minimum-Punkt befindet, so bewegt sie sich möglicherweise nur noch, wenn die einwirkende Kraft größer als der Threshold ist.
- setD: Dämpfung der Geschwindigkeit der Masse.
- force3D: Füge der Masse Kraft in Form eines Vektors zu!
- off: Stoppe die Bewegung der Masse!

- on: Setze die Bewegung der Masse fort!

Ausgabe:

- Position
- Kraft
- Geschwindigkeit

Verbindungen (Link)

Eingangsparameter:

- setL: Länge
- setK: Steifigkeit
- setD: Dämpfung der Verbindungsdeformierung
- setD2: Dämpfung der Massengeschwindigkeit
- Zahl-Object: Position der ersten Masse
- Zahl-Number: Position der zweiten Masse

Aktionen:

- bang: Gib die Kräfte aus!
- reset: Resette die Position der Massen und setze die Geschwindigkeit der Verbindung auf Null
- resetF: Resette die Dämpfung und die Trägheitskraft der Verbindung
- resetL: Setze die Länge der Bindung auf die derzeitige Länge
- setLmin: Setze die Mindestlänge
- setLmax: Setze die Maximallänge

Je größer die Masse, desto langsamer bewegt sie sich. Stellt man eine Dämpfung ein, so wird die Bewegung irgendwann zu Null werden. Je geringer die Steifigkeit, desto größer die Elastizität.

2.4.2 Dynamisches Patchen

Um die Simulation so genau wie möglich zu machen, müssen relativ viele Massen verwendet werden. Damit die Verbindungen zwischen Massen und Federn nicht von Hand gezogen werden müssen, verwende ich dynamisches Patchen.

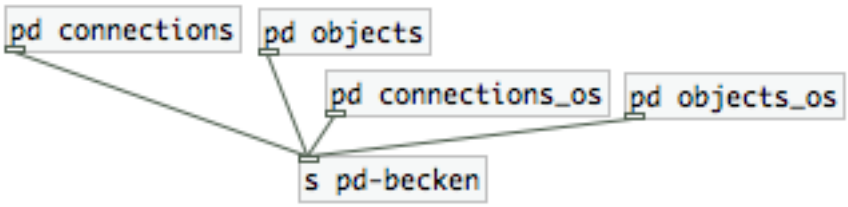


Abbildung 2.3: Dynamisches Patchen

In obiger Abbildung sehen wir vier Unterprogramme, deren Ausgänge in das Send-Objekt führen. Irgendwo anders gibt es ein Unterprogramm *becken*, und das Send-Objekt s *pd-becken* sorgt dafür, dass die Messages aus den Unterprogrammen im Unterprogramm *becken* Objekte und Verbindungen generieren. Eine solche Message hat folgendes Format:

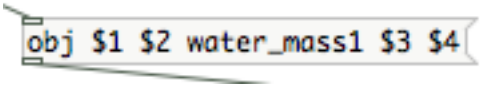


Abbildung 2.4: Dynamisches Patchen - Message

Die ersten zwei Parameter der Message beinhalten die Position des Objekts innerhalb des Unterprogramms *becken* und die zweiten zwei Parameter sind selbst definierte Parameter für die Funktion *water mass1*.

2.4.3 GEM Simulation

Die ersten Simulationsbilder sind in Abbildung 2.5 bis 2.7 dargestellt. Die Größe bzw. die Massenanzahl und die physikalischen Werte können vom Benutzer verändert werden.

Die Anregung einer bestimmten Masse ist natürlich noch nicht sehr benutzerfreundlich.

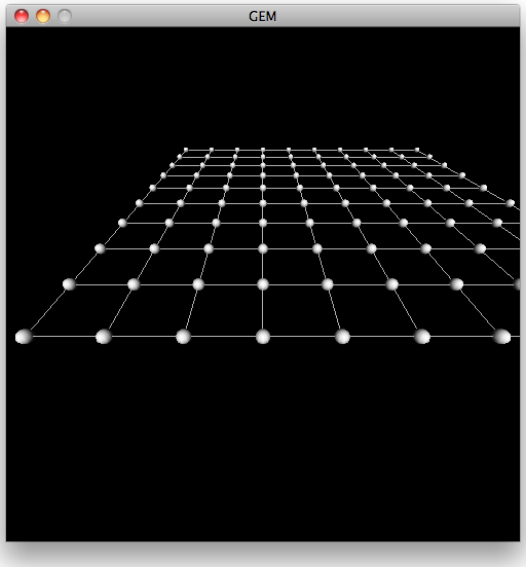


Abbildung 2.5: Simulation 1

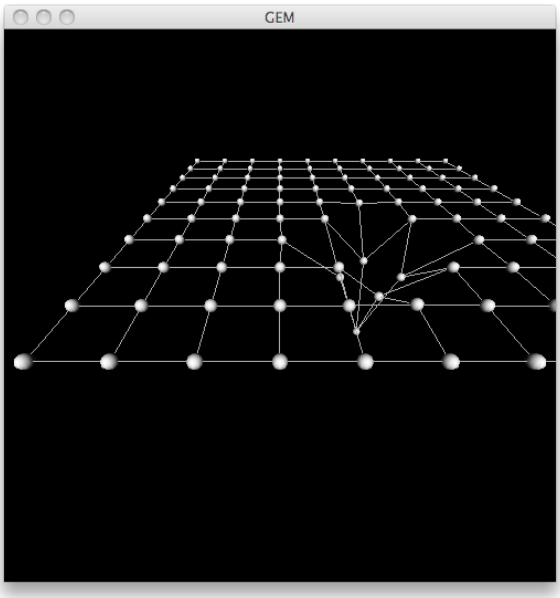


Abbildung 2.6: Simulation 2

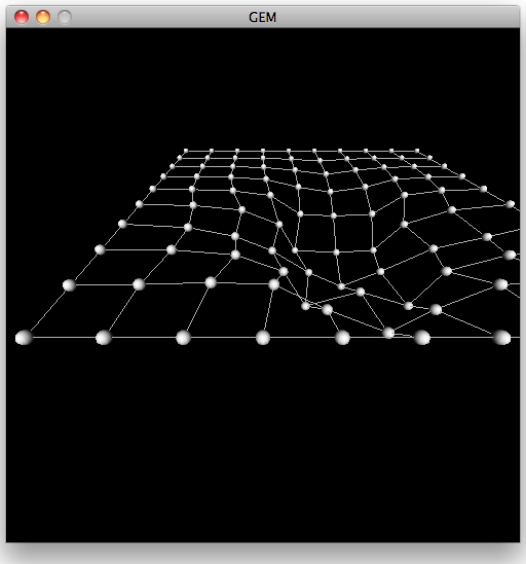


Abbildung 2.7: Simulation 3

Wie bereits im Punkt Überlegungen zum Feder-Masse System erwähnt, werde ich als nächstes versuchen, Oberflächenspannung miteinzubeziehen, indem ich ein starres Massenetz darunter lege, das nur in Z-Richtung mit dem ursprünglichen Netz zusammen hängt.
Dann müssen die richtigen Parameter eingesetzt werden. Und schon haben wir Kapillarwellen.

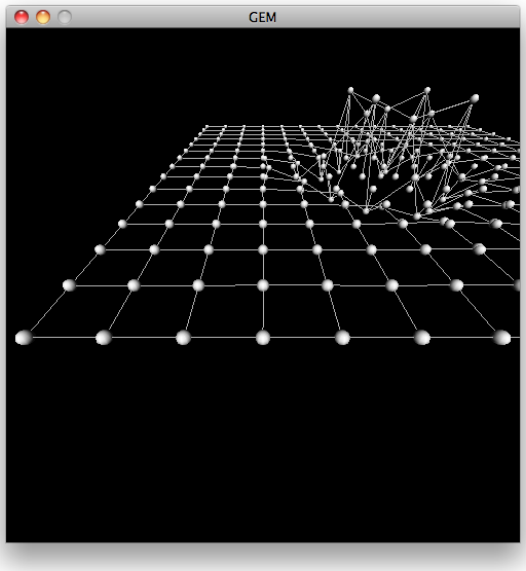


Abbildung 2.8: Simulation mit Oberflächenspannung 1

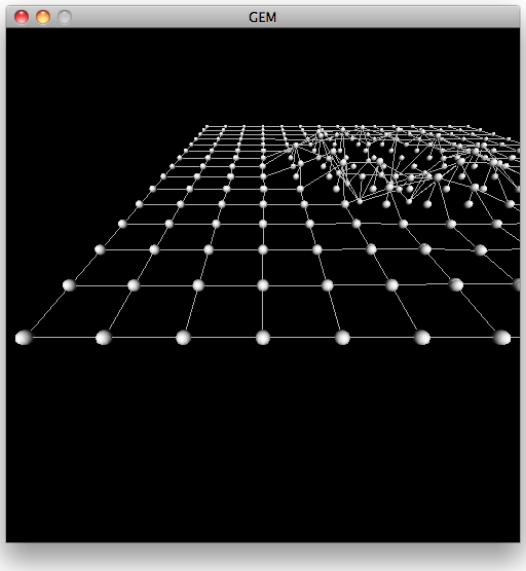


Abbildung 2.9: Simulation mit Oberflächenspannung 2

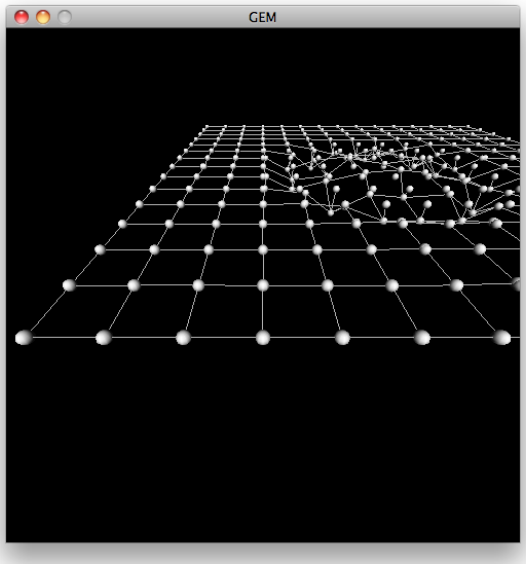


Abbildung 2.10: Simulation mit Oberflächenspannung 3

2.4.4 Visualisierung

Für die Visualisierung wird GEM verwendet. GEM (Graphics Environment for Multimedia) ist eine Bibliothek für Pure Data um OpenGL- und Pixelgrafiken in Echtzeit darzustellen und zu steuern. [19]

3 Klangerzeugung in Pure Data

Noch kann man keinen Ton hören, es gibt lediglich Eingangsparameter für ein Instrument.
Die Klangerzeugung soll mit Pure Data programmiert werden. Pure Data (Abkürzung: Pd) ist eine datenstromorientierte Programmiersprache und Entwicklungsumgebung, die visuelle Programmierung benutzt. Sie wird vor allem zur Erstellung von interaktiver Multimedia-Software eingesetzt, etwa für Software-Synthesizer in der elektronischen Musik. [20]
Wie die Werte der Steuerung aber genau interpretiert beziehungsweise wie der Klang entstehen soll, soll in diesem Kapitel besprochen werden.

3.1 Klangvorstellung

Der Klang soll auf jeden Fall auch rein intuitiv und nicht erst über Erklärungen mit dem Bild zusammen passen. Es gibt mehrere Möglichkeiten den Klang zu gestalten. Hier ein paar Möglichkeiten, die für mich in Frage kommen:

- Ein großer schwingender, glockenartiger Klang.
- Eine Art Windspiel, das wellenartig klingt indem viele kurze Klänge zu hören sind.
- Eine Art Windspiel, das wellenartig klingt indem wenige längere Klänge zu hören sind.

Die Windspiel- Metallophon- Idee gefällt mir ganz gut. Der schlussendliche Klang wird durch die Physical Modelling Lösung variabel sein.

3.2 Lösungsweg

Ich wollte von Beginn an ein Physical Modeling Instrument entwerfen. Dazu wären einige in Frage gekommen:

- Karplus Strong
- Feder-Masse-Dämpfer-System
- Differentialgleichung
- Modal Synthesis
- Waveguide Synthesis
- Scanned Synthesis

Am Besten fand ich die Waveguide Synthesis, da diese wohl die spannendsten Klänge herausbringen würde. Da es schon fertige Systeme gibt, wollte ich ein solches in meinen PD Code implementieren. Dazu habe ich einige Wege ausprobiert. Die Zuordnung der Elemente zu den Spielparametern erwies sich als zu kompliziert. Schlussendlich habe ich diesen Weg dann aufgegeben.
Schlussendlich habe ich die optische Ähnlichkeit der Wasserwellen mit der Besselfunktion ausgenutzt und eine Quasi-FM-Methode angewendet. Auf diese Methode, die Waveguide Synthesis und die Scanned Synthesis werde ich genauer eingehen.

3.3 Scanned Synthesis (patentiert)

Diese Methode klang sehr spannend und ich wollte sie auf jeden Fall ausprobieren. Es war von Beginn an aber sehr ungewiss, was da für ein Klang herauskommen würde. Die Informationen zu Scanned Synthesis habe ich aus dem Paper [21].
Scanned Synthesis ist eine Technik, mit der man durch motorisches Bewegen von Wavetables Klangfarben in Echtzeit verändern kann. Das Besondere an dieser Methode ist,

dass der Benutzer direkten Eingriff auf die Klangfarbe hat. Scanned Synthesis funktioniert aber nur für Steuerungssignale unter 15 Hz. Bei dieser Methode braucht man irgendein periodisch scannbares Material, also zum Beispiel eine Schnur, dessen Auslenkung bestimmt wird, ein bewegliches Blech oder eben unsere Wasseroberfläche. Die gescannten Werte werden in eine Tabelle geschrieben, mit der dann eine Wavetable Synthesis gesteuert wird.
Wavetable Synthesis funktioniert normalerweise so, dass Werte in eine Tabelle geschrieben werden, die dann periodisch ausgelesen und an einen Schallwandler weitergegeben werden. Bei der Scanned Synthesis Methode ändert sich diese Tabelle aber periodisch, und sie beschreibt nicht den Zeit- sondern den Frequenzbereich. Diese Periode steuert die Tonhöhe der Synthese. Das heißt, Tonhöhe und Klangfarbe werden also komplett separat gesteuert.

3.3.1 Psychophysikalische Grundlagen

In den psychophysikalischen Grundlagen geht es um die menschliche auditive Wahrnehmung und dessen motorischen Fähigkeiten. In den 1960er Jahren hat Jean-Claude Risset gezeigt, dass interessante Klangfarben sich mit der Zeit ändernde Spektren besitzen. Wie bereits erwähnt soll diese Änderung aus klanglich ästhetischen Gründen nicht mehr als 15 Hz betragen. Sehr interessant ist, dass diese Frequenz auch genau die ist, mit der der Mensch maximal seinen Körper bewegen kann.

3.3.2 Haptische Frequenzen

Um der Steuerfrequenz einen passenden Namen zu geben, werden die Spektralände-rungsfrequenzen in Scanned Synthesis *Haptische Frequenzen* genannt.

3.3.3 Der Versuch

Bill Verplank und Max Mathews von der CCRMA Stanford University, die das Paper verfasst haben, auf das ich mich beziehe, haben die Scanned Synthesis an einem Versuch

angewandt. Sie sind zum Schluss gekommen, dass es nicht ausreichend ist, dass sich die Werte der Tabelle irgendwie mit 15 Hz ändern. Eine gewisse Anzahl an Werten müssen von ihren benachbarten Werten abhängen. Eine gute Lösung die richtigen Werte zu scannen ist das eindimensionale String-Modell der finiten Elemente Abschätzung, was soviel bedeutet wie ein Feder-Masse-System aufzustellen. Die resultierenden Differentialgleichungen können dann vom Computer gelöst werden. Im Versuch hat sich herausgestellt, dass die interessantesten Klänge entstehen, wenn die physikalischen Werte der Massen und Federn entlang des gescannten Pfades nicht einheitlich sind.
Außerdem wurde festgestellt, dass wenn die Werte der Tabelle plötzlich für einige Sekunden still stehen, dass das Spektrum also gleich bleibt, der Klang sofort langweilig wird. Das hat die Aussage von Jean-Claude Risset eindeutig bestätigt.

3.3.4 Ergebnis

Die fundamentalen Elemente der Scanned Synthesis sind also

- Ein dynamisches System, das auch langsame Vibrationen zulässt.
- Eine Steuerung, durch die ein Künstler direkt auf das System eingreifen kann und es mit haptischen Frequenzen verändern kann.
- Eine periodische Abtastung der Werte muss gegeben sein mit Frequenzen im hörbaren Bereich (20 Hz bis 15 kHz). Es muss aber nicht der gesamte hörbare Bereich abgedeckt werden.

Die Versuche haben gezeigt, dass durch Scanned Syntheses sehr interessante Klänge entstehen, die echt und live klingen, und doch kein schon bestehendes Instrument nachahmen.
Es ist durchaus denkbar Wellenpunkte abzutasten um sie dann als Wavetables zu verwenden, auch wenn das nicht die ursprüngliche Idee war. Sind die Abtastpunkte einmal vorhanden, sollte es auch kein großer Aufwand mehr sein, daraus Wavetables zu generieren.

3.3.5 Eine Abwandlung von Scanned Synthesis

Die Idee ist es, die Tabelle der z-Positionen der Massen als Amplitudenspektrum zu Interpretieren. Es soll im Zeitbereich ein Signal wie zum Beispiel weißes Rauschen oder ein Sägezahn mit FFT in den Frequenzbereich gebracht werden, da mit dem z-Positionsspektrum multipliziert und dann wieder in den Zeitbereich rücktransformiert werden, um anschließend ausgegeben zu werden. Das Ganze entspricht also einer Filterung eines Rauschens, wobei sich das Filter ständig ändert. Richtige Scanned Synthesis würde das Spektrum direkt in den Zeitbereich transformieren, das ist aber in meinem Fall schwierig, da ich nur ein Amplitudenspektrum habe, die Phase also fehlt.

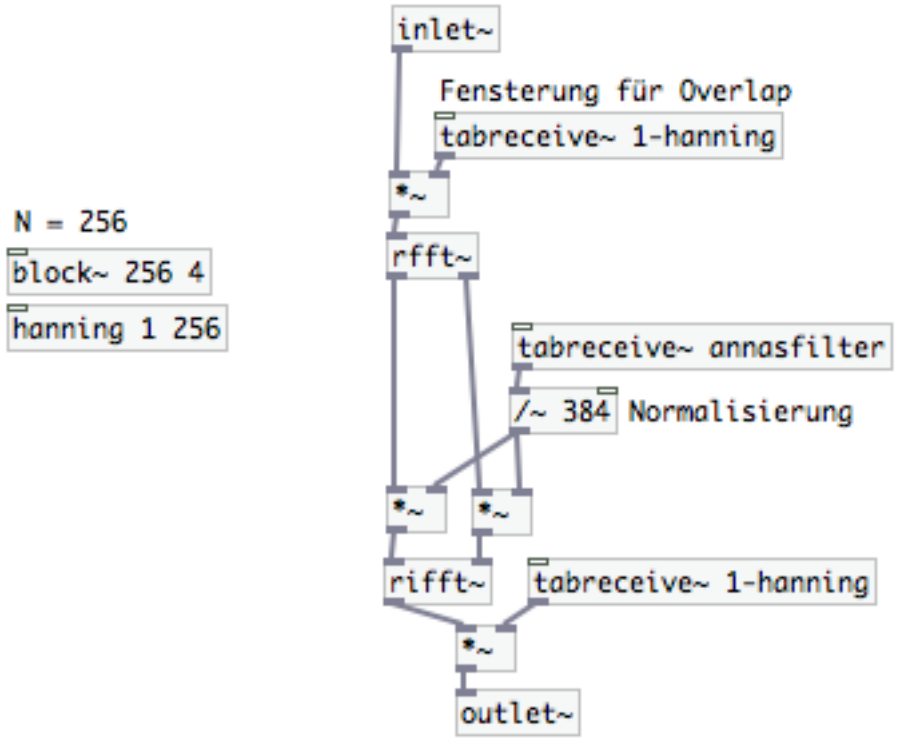


Abbildung 3.1: Scanned Synthesis in PD

Der Code dazu ist relativ einfach. FFT in PD muss immer in einem separaten Fenster geschehen. FFT funktioniert nur in Verbindung mit der Funktion *block*, die die Fensterlänge und den Overlapfaktor bestimmt. Das Eingangssignal kommt rein, wird mit einem Hanning-Window gefenstert und in den Frequenzbereich übertragen. Dort wird es mit

der Scanned Synthesis Tabelle, hier heißt sie *annasfilter* gefiltert, nachdem normalisiert wird, was man immer machen muss, wenn man mit der FFT Funktion von PD arbeitet. Anschließend wird nur noch rücktransformiert und wieder gefenstert. Als Eingang habe ich ein weißes Rauschen und auch einen Sägezahn ausprobiert und beides sehr spannend gefunden.

3.4 Digital Waveguide Synthesis [1] [2]

3.4.1 Grundlagen Waveguide Synthesis

Ein verlustloser digitaler Wellenleiter ist definiert als eine bidirektionale Delay Line.

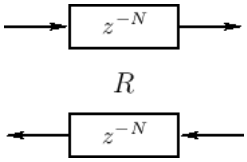


Abbildung 3.2: Bidirektionale Delay Line, Bildquelle: [6]

Seit 1747 ist bekannt, dass die Vibration einer idealen Saite als die Summe von zwei entgegengesetzten Wellen beschrieben werden kann. Um Verlust und Streuung zu berücksichtigen werde ich auch Filterung verwenden. Der Widerstand R wird benötigt um den Wellenleiter mit anderen physikalischen Simulationen zu verbinden.
Physikalische Ausgänge werden folgendermaßen berechnet:

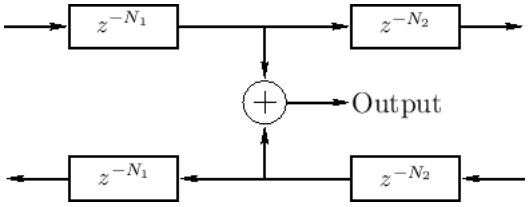


Abbildung 3.3: Physikalischer Ausgang, Bildquelle: [7]

1D Wanderwellen sind eindeutig bestimmt durch zwei linear unabhängige physikalische Variablen entlang des Wellenleiters. Diese Variablen können Vibration, Geschwindigkeit, oder Druck sein.
Physikalische Eingänge werden folgendermaßen behandelt:

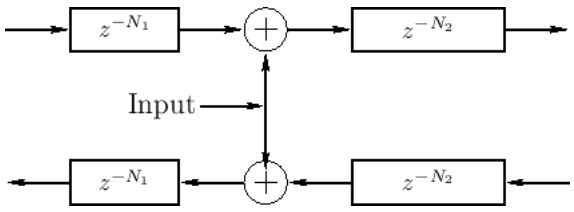


Abbildung 3.4: Physikalischer Eingang, Bildquelle: [8]

Ein physikalisches Eingangssignal verhält sich wie eine Störung auf das 1D Ausbreitungsmedium. Wird eine Kraft auf das Ausbreitungsmedium ausgeübt, so wirkt sie gleichermaßen auf die hin- und rücklaufende Welle. Die Störung wird zum aktuellen Zustand dazuaddiert.
In Wirklichkeit stimmt das mit der Superposition nicht ganz. Normalerweise gibt es eine Interaktion zwischen dem schon vorhandenen Vorgang und dem neuen.

3.4.2 Gitter

Da ich aber keine Saite und kein Holzblasinstrument simulieren will, sondern ein Instrument mit zweidimensionaler Wellenausbreitung brauche ich ein 2D Gitter. [22]

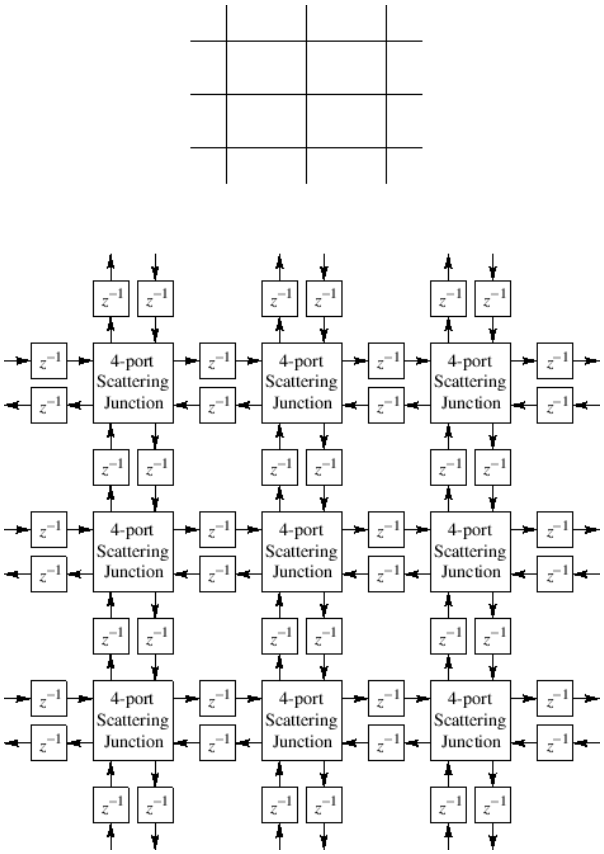


Abbildung 3.5: 2D Gitter [9]

Für jeden Knotenpunkt gilt dieselbe einfache Geschwindigkeitsgleichung:

$$v = \frac{in_1 + in_2 + in_3 + in_4}{2}$$

$$out_k = v - in_k$$

Durch das Gitter entsteht allerdings ein Ausbreitungsfehler. Die Gittergeometrie beeinflusst die Ausbreitungseigenschaften sehr stark. Die Triangular Waveguide Gitterstruktur (triangular waveguide mesh) erweist sich als beste 2D Gitterstruktur [10] . Es zeigt die geringsten Geschwindigkeitsveränderungen als Funktion der Ausbreitungsrichtung auf, also kommt der Isotropie am nächsten. Um die Richtungsabhängigkeit noch weiter zu verbessern wäre es auch möglich die Interpolated Waveguide Mesh [23] zu verwenden, diese erfordert aber einen deutlich höheren Rechenaufwand.

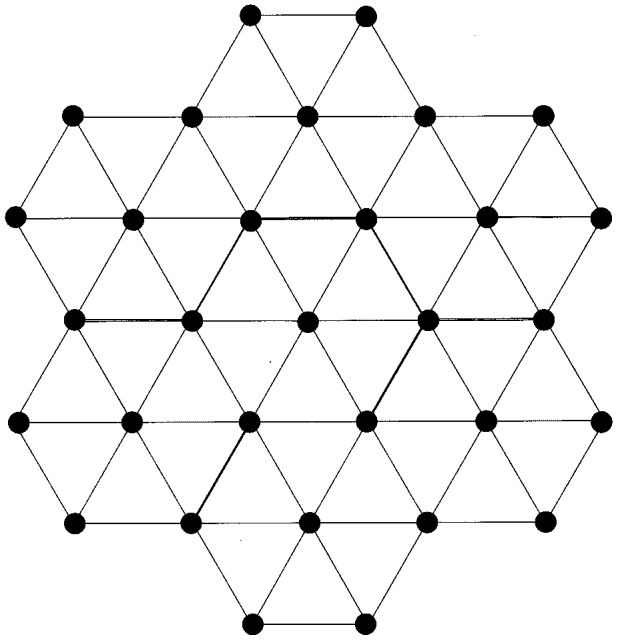


Abbildung 3.6: Triangular Waveguide Mesh, Bildquelle:[10]

Weitere Informationen bezüglich der Waveguide Methode: [24] [25]

3.4.3 Realisierung der Waveguide Synthesis

Da es schon sehr gut funktionierende Waveguide Synthesis Toolkits gibt, die nur noch eingebunden werden müssen, und die Programmierung in PD im Rahmen dieser Arbeit zu aufwändig wäre, habe ich versucht das Synthese ToolKit einzubinden. [11]

3.4.4 Das Synthese ToolKit in C++ (STK)

Perry R. Cook und Gary P. Scavone haben zusammen das Synthesis Toolkit in C++ geschrieben. Das Toolkit ist ein Open Source Programm, das mit Audio Signalverarbeitung und algorithmischen Synthese-Klassen arbeitet und in Echtzeit betrieben wird.

STK in PD

Für das Einbetten von STK in PD gibt es ein eigenes Objekt. Zum Kompilieren dieses wird Flext benötigt. Flext ist eine Reihe von Macros, die STK in PD einbinden.

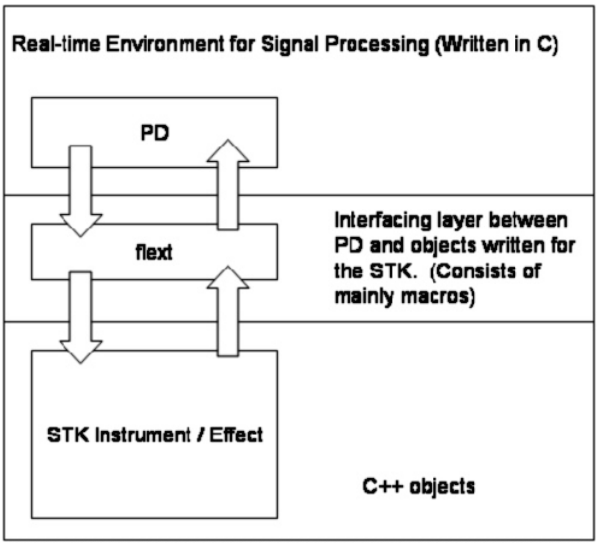


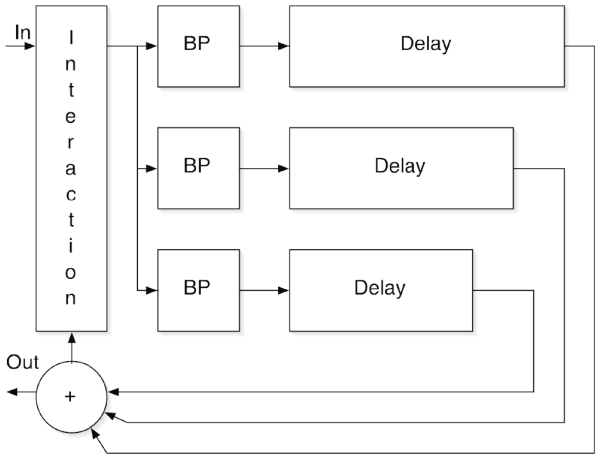
Abbildung 3.7: Architektur der STK Einbettung, Bildquelle: [11]

Fehler

Es hat sich herausgestellt, dass das Einbinden von STK in PD mit der Stanford-Anleitung nicht mehr funktioniert, da sich die Versionen von STK und Flext geändert haben und die alten nicht mehr erhältlich sind. [26]
Auch die Einbindung von STK über Faust brachte keinen Erfolg. [27]

3.5 Banded Waveguides [3]

Das Spektrum eines vibrierenden Systems wird in Frequenzbänder aufgeteilt. Jedes Band enthält vorzugsweise einen Resonanzbereich. Für jedes Band wird die wandernde Welle und die Resonanzfrequenz modelliert. Hier die elementare Struktur:



Jetzt müssen nur noch die einzelnen Komponenten definiert werden.

3.5.1 Das Delay

Die einfachste Komponentenbestimmung ist wohl das Delay. Wir befinden uns mit der Struktur aber im Frequenzbereich, also keine einfache Zeitverzögerung, sondern eine Multiplikation mit z^{-t}. In Echtzeit wird das gelöst durch zirkuläre Buffer (Dodge and Jerse 1985). Non-Integer Brüche können durch Fractional Delay Techniken gelöst werden (Laakso, Välikmäki, Karjalainen, and Laine 1996).

$d = \frac{f_s}{f_m}$ ... Delay Line Length

f_s ... Samplefrequenz

f_m ... Modenfrequenz des jeweiligen Bandes

3.5.2 Der Bandpass

Wegen der Einfachheit und der Effizienz wird ein Filter zweiter Ordnung emfohlen. Die wichtigsten Parameter sind die Center Frequenz und der Gain bei dieser Frequenz. Die Bandbreite ist nicht so heikel.

Gain

Der Gain ist der Dämpfungsfaktor, der sich auf den multiplikativen Verlust pro Oszillationsperiode bezieht.

Schluss

Während dem Erarbeiten einer Lösung für Banded Waveguides in PD ist eine neue Idee aufgekommen, die versprach spannender und einfacher zu werden.

3.6 Frequenz Modulation FM

Wenn man sich die Oberflächenwellenausbreitung ansieht, kann man eine starke Ähnlichkeit zur Besselfunktion feststellen. Das hat mich auf die Idee gebracht, eine FM Modulation anzuwenden, ohne wirklich zu modulieren, also rein aus Summen. Um das zu verstehen, möchte ich kurz in die Theorie der FM Modulation eingehen.

3.6.1 FM Grundlagen

Stellen wir uns einen Sinuston vor, dessen Frequenz ebenfalls ein Sinus ist, so erhalten wir einen Sinuston, dessen Frequenz sinusartig steigt und fällt, also dessen Tonhöhe sich sinusartig verändert. Ist dieser Sinus langsam, so hören wir ein Vibrato.

3.6.2 FM Mathematik

$s(t) = A \cdot sin(\omega t)$
$\omega = \omega_c + \Delta\omega \cdot cos(\omega_m t)$

$\Delta\omega$ . . .Frequenzabweichung
ω_c . . .Trägerfrequenz

$s(t) = A \cdot sin((\omega_c + \Delta\omega \cdot cos(\omega_m t))t)$
$I = \Delta\omega/\omega_m$

ω_m . . .Modulationsfrequenz
I . . .Modulationsindex

$s(t) = A \cdot sin(\omega_c t + Isin(\omega_m t))$

Lösung dieser Funktion im Frequenzbereich:
$s(t) = \sum_{k=-\infty}^{\infty} J_k(I) sin(\omega_c + k\omega_m)$

Also ist die Lösung eine Summe von Sinussignalen, deren Amplituden von der Besselfunktion abhängen.

3.6.3 Realisierung

Amplitude

Wir ersetzen die Besselfunktion also durch Wassermassenamplituden. Um Rechenleistung zu sparen, und weil es im Klang auch keine signifikanten Änderungen mehr machen würde, werden die Anschlagsmasse und vier rechts liegende, nebeneinanderliegenden Massen verwendet, also pro Klang fünf Sinusoszillatoren.

Frequenz

Die Trägerfrequenz hängt vom Ort der Kraftzufuhr ab. Die Position bestimmt also die Tonhöhe. In x-Richtung entspricht jede weitere Masse einem Halbtonschritt, und in y-Richtung dem nächsten harmonischen Teilton. Die Modulationsfrequenz ist über das Verhältnis von Modulationsfrequenz zu Trägerfrequenz einstellbar. Über einen Toggle-Knopf kann eingestellt werden, dass sich die Modulationsfrequenz zur Trägerfrequenz so verhält wie die z-Position der angeschlagenen Masse zur z-Position der Masse daneben. Es klingt aber auch sehr schön, wenn man anstatt der Modulationsfrequenz das Vielfache der Trägerfrequenz verwendet.

$$s(t) = \sum_{k=-\infty}^{\infty} J_k(I) sin(k\omega_c) \tag{3.1}$$

4 Die Hardware-Steuerung

Um das Wellengesteuerte Hyperinstrument zu spielen, sind serielle Schritte erforderlich.

- Setzen der Messpunkte mit jeweiliger Zuordnung von Tonhöhe.
- Ort wählen, wo der virtuelle Stein in die Flüssigkeit fallen soll.

Da die Schritte seriell und sehr einfach sind, ist es fraglich, ob überhaupt eine Hardwaresteuerung nötig ist. Dennoch führe ich einige Möglichkeiten auf:

- Touchpad

  Es sind schon relativ günstige Evaluation-Boards mit Touchpads erhältlich. (z.B. [31]) Es wäre denkbar, das Touchpad als Draufsicht auf das Flüssigkeitsbecken zu verwenden. Hier könnten die Messpunkte einfach platziert werden und der Ausganspunkt der Wellen bestimmt werden.

- Echtes Wasserbecken

  Man könnte auch ein echtes Wasserbecken verwenden. Die Messpunkte könnte man entweder hardwaremäßig durch Wasserhöhensensoren realisieren oder auch virtuell. Jetzt könnte man einen Stein ins Wasser werfen und die Wasseroberfläche abtasten. Entweder mit Kinect von XBox oder mit ccd Sensor bzw. sccd Sensor, wie sie in Photokameras verwendet werden.

- Knopfmatrix

  Diese Lösung würde ähnlich funktionieren wie das Touchpad, nur dass sie eine viel geringere Auflösung für die Platzierung bereitstellt. Man könnte entweder schon vorhandene Boards verwenden oder auch eines selber bauen. Am Besten ist wohl

das Novation Launchpad (155 Euro) geeignet. Es funktioniert über MIDI, es kann gut über PD gesteuert werden, die Tasten können in vielen Farben leuchten, was sehr gut wäre um die Messpunkte darzustellen. [32]

4.1 Endgültige Hardware

Schlussendlich habe ich mich für ein Touchpad entschieden, auf dem PD und GEM funktionieren. Dafür mussten einfach die Anwenderfenster an der richtigen Position stehen und die richtige Größe haben. Standardmäßige Touchpads haben die Auflösung 1024 x 768, mit dem Format 4:3. Wird die Touchpad Version auf einem Computer ausgeführt, zum Beispiel MacBook Pro 13,3 oder 15,4 Zoll, so ist das Bedienfenster ebenso in einer schönen Größe.

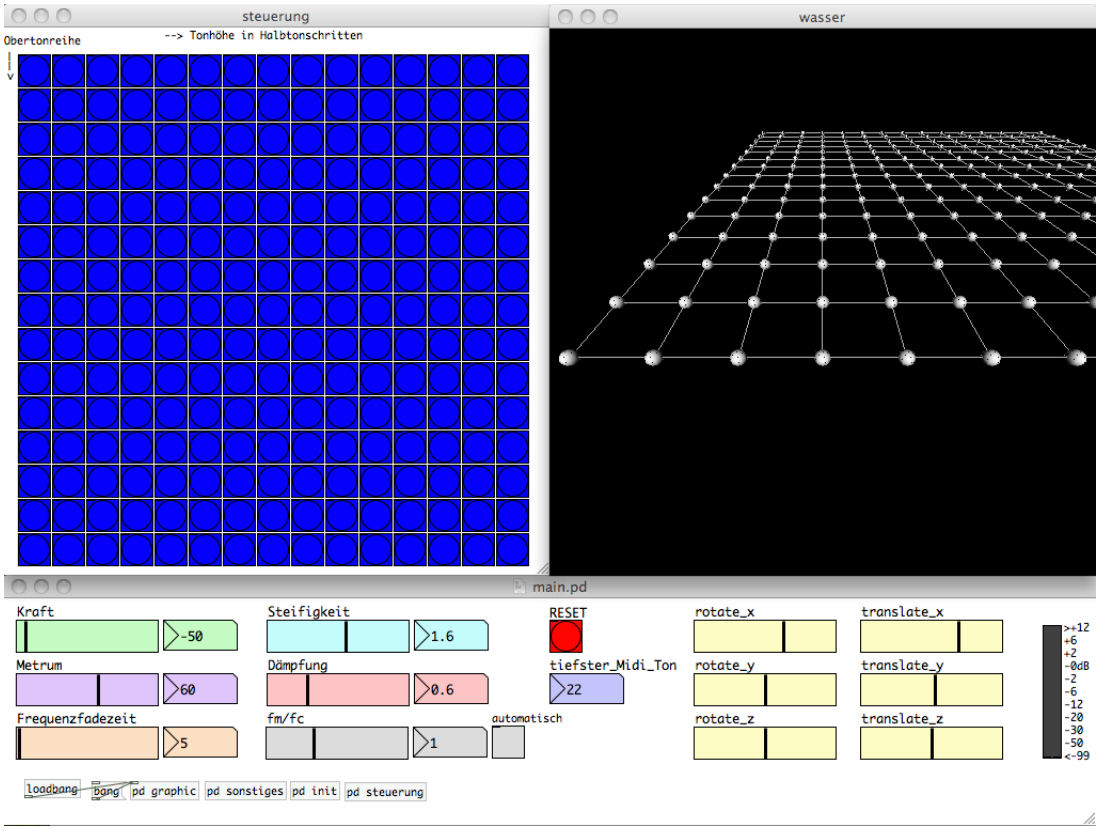


Abbildung 4.1: GUI des Instruments

Zudem funktioniert das Instrument mit Single Touch auch über die Maussteuerung, Multitouch ist aber von Vorteil, da sonst nur ein Klang auf einmal angespielt werden kann.

5 Zusammenfassung

Das Ziel der Arbeit wurde vollständig erfüllt und es ist ein interessantes Klanginstrument entstanden, das auch für Livevorführungen verwendet werden kann.

Die Steuerung besteht aus einem mit PMPD erzeugten Feder-Masse-System, das eine Flüssigkeitenoberfläche darstellt. Jede Masse des Systems kann separat angeregt werden und die physikalischen Eigenschaften Dämpfung und Steifigkeit der Federn sowie die Kraft auf die Massen können, auch live, eingestellt werden. Die Ansicht erfolgt mit GEM, und ist verstellbar. Auch die Rechengeschwindigkeit des Systems ist flexibel.

Die Klangprogrammierung entspricht einer Abwandlung der FM-Synthese. Die FM-Synthese funktioniert nach der Formel $s(t) = \sum_{k=-\infty}^{\infty} J_k(I) sin(\omega_c + k\omega_m)$. Die Abwandlung beinhaltet zwei große Änderungen. Erstens wurden die Werte der Besselfunktion $J_k(I)$ durch Z-Positionswerte der Massen ersetzt, da Wasserwellen an die Besselfunktion erinnern, und zweitens werden nur fünf Teiltöne aufsummiert anstatt unendlich viele. Also:

$$s(t) = \sum_{k=0}^{4} f_{Wasser}(k) sin(\omega_c + k\omega_m) \tag{5.1}$$

Auch hier sind einige Parameter einstellbar. Zum einen der tiefste Ton des Instruments und zum anderen das Verhältnis der Modulationsfrequenz zur Trägerfrequenz.

Das Instrument ist mit einem Touchpad spielbar.

Literaturverzeichnis

[1] Julius O. Smith III. Physical Audio Signal Processing. Online-Quelle, 2012. https://ccrma.stanford.edu/~jos/pasp/.

[2] Julius O. Smith III. Digital Waveguide Synthesis, Selected Tutorials, Papers, Programming Examples, (Some) Sound Samples, and Related Links. Online-Quelle, 2012. https://ccrma.stanford.edu/~jos/wg.html.

[3] Georg Essl, Stefania Serafin, Perry R. Cook, and Julius O. Smith. Theory of banded waveguides, 2004. http://web.eecs.umich.edu/~gessl/georg_papers/CMJ04-theory.pdf.

[4] Kapillarwellen Abbildung
http://www.museum-rorschach.ch/images/Blatt_Wasser_Wellen_Nachhaltigkeit2.jpg.

[5] Prof. Dr. Gregor Herten, Physikalisches Institut, Universität Freiburg, *Experimentalphysik 1 und 2*:
http://hep.uni-freiburg.de/Lehre/ex2ss08/ex12.pdf.

[6] Julius O. Smith III. Bidirektionale Delay Line. Online-Quelle, 2012. ccrma.stanford.edu/%7Ejos/swgt/Digital_Waveguide.html.

[7] Julius O. Smith III. Physical Output Waveguide Synthesis. Online-Quelle, 2012. https://ccrma.stanford.edu/%7Ejos/pasp/Physical_Outputs.html.

[8] Julius O. Smith III. Physical Input Waveguide Synthesis. Online-Quelle, 2012. https://ccrma.stanford.edu/%7Ejos/pasp/Physical_Inputs.html.

[9] Julius O. Smith III. 2D Waveguide Mesh. Online-Quelle, 2012. `https://ccrma.stanford.edu/~jos/pasp/Rectilinear_2D_Mesh.html`.

[10] Federico Fontana and Davide Rocchesso. Signal-theoretic characterization of waveguide mesh geometries for models of two-dimensional wave propagation in elastic media. *IEEE Transactions on Speech and Audio Processing*, 9(2):152–161, 2001.

[11] Edgar J. Berdahl, Nelson Lee and Julius O. Smith III. Embedding Synthesis ToolKit (STK) Instruments in Pure Data (PD) Externals. Technical report, Center for Computer Research in Music and Acoustics (CCRMA), and the Department of Electrical Engineering Stanford University Stanford, CA, 2007.

[12] Simulation einer Wasseroberfläche: `http://commons.wikimedia.org/wiki/File:Shallow_water_waves.gif`.

[13] Dieter Heidorn. Oberflächenwellen. `http://www.dieter-heidorn.de/Physik/SS/K03_Wellen/Wasserwellen/Wasserwellen.html`.

[14] F. Büsching and N. Speranski. *Beiträge Aus Dem Küsteningenieurwesen:*. Veröffentlichungen des Fachbereichs Architektur und Bauingenieurwesen der FH Bielefeld, Abt. Minden. FB Architektur und Bauingenieurwesen, FH Bielefeld, 1996.

[15] Simulation von Wasser mit Smoothed Particle Hydrodynamics: `http://www.t35.physik.tu-muenchen.de/lectures/sph.html`.

[16] Simulation einer Trommelmembran: `http://www.allmystery.de/dateien/gw28633,1292626461,200px-Drum_vibration_mode22.gif`.

[17] C. Henry. Physical modeling for pure data (PMPD) and real time interaction with an audio synthesis. In *Proc. Sound and Music Computing*, Paris, France, October 2004.

[18] Daniel Groh. Echtzeit-Simulation eines wassergefüllten Ballons mit Feder-Masse-Dämpfer-Systemen. Master's thesis, Johann Wolfgang Goethe Universität Frankfurt am Main, 2006.

[19] GEM (Graphics Environment for Multimedia) `http://gem.iem.at/documentation/manual/manual/gem-introduction`.

[20] Pure Data Erklärung: `http://de.wikipedia.org/wiki/Pure_Data`.

[21] M. Mathews B. Verplank. Scanned Synthesis. In *Proceedings of the 2000 International Computer Music Conference, ICMA*.

[22] Scott A. Van Duyne, Julius O. Smith III, Scott A. Van, Duyne Julius, and O. Smith Iii. Physical modeling with the 2-d digital waveguide mesh, 1993. `https://ccrma.stanford.edu/%7Ejos/pdf/mesh.pdf`.

[23] Julius O. Smith III. Dispersion. Online-Quelle, 2012. `https://ccrma.stanford.edu/~jos/pasp/Dispersion.html`.

[24] Marc Aird and Joel Laird. Extending digital waveguides to include material modelling. In *IN PROCEEDINGS OF THE COST G-6 CONFERENCE ON DIGITAL AUDIO EFFECTS (DAFX-01*, pages 6–8, 2001.

[25] Marc-Laurent Aird. *Musical Instrument Modelling Using Digital Waveguides*. PhD thesis, University of Bath, 2002.

[26] Rickard Andersson. PD Forum. Online-Quelle, 2005. `https://ccrma.stanford.edu/~jos/pdf/mesh.pdf`.

[27] laternist. FaustSTK Tookit in Pure Data. Online-Quelle, 2012. `https://laternist.posterous.com/faststk-tookit-in-pure-data`.

[28] Download Faust. Online-Quelle. `http://faust.grame.fr/`.

[29] Download Pure. Online-Quelle. `http://code.google.com/p/pure-lang/`.

[30] Download LLVM. Online-Quelle. `http://llvm.org/releases/download.html`.

[31] Touchpad von Microchip. Online-Quelle. `http://www.microchip.com/meb`.

[32] Launchpad von Novation. Online-Quelle. `http://www.novationmusic.com/products/midi_controllers/launchpad`.