Einführung in die Digitale Elektronische Klangsynthese

MITWIRKENDE			
	TITEL : Einführung in die Digitale Elektronische Klangsynthese		
AKTION	*NAME*	*DATUM*	*UNTERSCHRIFT*
VERFASST DURCH	Orm Finnendahl	2020-11-19	

VERSIONSGESCHICHTE			
NUMMER	DATUM	BESCHREIBUNG	NAME

Inhaltsverzeichnis

1 Vorwort **1**

2 Übersicht über unterschiedliche Computermusiksysteme **2**

2.1 Grafische, anwenderorientierte Programme . . . 2

2.2 Grafische, patcherbasierte Systeme . . . 3

2.3 Textbasierte Systeme . . . 5

3 Installationsanleitungen **7**

3.1 Installation von pure data . . . 7

3.2 Installation von SuperCollider . . . 9

3.3 Installation von incudine . . . 9

4 Grundlagen der digitalen Signalverarbeitung **11**

4.1 Analog-Digital- und Digital-Analog-Wandlung . . . 11

4.2 Animation der Analog-Digitalwandlung von Schallwellen . . . 12

4.2.1 **Erklärung der Animation** . . . 13

4.2.1.1 Die obere Zeile der Grafik . . . 13

4.2.1.2 Die mittlere Zeile der Grafik . . . 13

4.2.1.3 Die untere Zeile der Grafik . . . 13

4.3 Darstellung komplexer Audiosignale . . . 14

4.4 Räumliche Schallabstrahlung . . . 15

4.5 Blockverarbeitung digitaler Signale und Latenz . . . 17

4.6 Interner Takt von Computermusiksystemen . . . 17

4.7 Unitgeneratoren und der Audiograph . . . 18

4.8 Oszillatoren, Frequenz und Amplitude . . . 19

4.8.1 Erzeugung periodischer Wellenformen durch einen Phasor . . . 19

4.8.2 Oszillatoren . . . 19

4.8.2.1 Generierung einer Sinusschwingung durch Rechnen . . . 20

4.8.2.2 Generierung einer Sinusschwingung durch Table-Lookup . . . 20

4.8.3 Amplitude . . . 21

4.8.4 Frequenz . . . 22

4.9 Elementare Wellenformen . . . 25
4.9.1 Sägezahn . . . 25
4.9.2 Dreieck . . . 26
4.9.3 Rechteck . . . 26

5 Synthesemodelle **27**
5.1 Waveshaping . . . 27
5.1.1 Verzerrer . . . 31
5.1.2 Sinusförmige Transferfunktionen . . . 32
5.1.3 Chebyshev Polynome . . . 34
5.1.4 Bandbegrenzter Puls mit Hilfe der Exponentialfunktion . . . 35
5.2 Amplitudenmodulation (Ringmodulation) . . . 36
5.3 FM-Synthese . . . 37
5.4 Aufgaben . . . 37
5.4.1 Grundlagen der digitalen Signalverarbeitung . . . 37
5.4.2 Synthesemodelle . . . 38

6 Abstraktionen und Steuerung **39**
6.1 Hüllkurven . . . 39
6.2 Instrumente und Partituren . . . 41
6.3 Polyphonie . . . 41

7 Vertiefungen **42**
7.1 Vertiefungen zur Einführung in die Sprache clojure . . . 42
7.1.1 Liste mathematischer Operatoren . . . 42
7.1.2 Funktionsdefinitionen . . . 43
7.1.3 Bindungen . . . 44
7.2 Vertiefungen zur Einführung in overtone . . . 44

8 Bibliografie **45**

Abbildungsverzeichnis

2.1 Beispiel einer Bedienoberfläche von Reaktor . . . 2
2.2 Beispiel eines pure data (pd) Patches . . . 3
2.3 Modularer Analogsynthesizer und Flussdiagramm (Programmablaufplan) als Modelle für patcherbasierte Computermusiksysteme, wie Max/MSP oder pd. . . . 4
2.4 Beispiel einer SuperCollider Session . . . 5

3.1 Startbildschirm von pure data . . . 8
3.2 Audiotest Fenster von pure data . . . 9

4.1 Schema von Analog-Digital- und Digital-Analog-Wandlung . . . 11
4.2 Animation der AD-Wandlung von Schallwellen . . . 12
4.3 Die ersten 20 Sekunden von Beethovens 5. Sinfonie in Wellenformansicht . . . 15
4.4 Beispiel einer 8-kanaligen Lautsprecheranordnung für räumliche Schallabstrahlung . . . 16
4.5 Blockverarbeitung von Samples (Blockgröße: 8 Samples) . . . 17
4.6 Verbindung verschiedener Audiomodule . . . 18
4.7 Sägezahnwelle (Phasor) . . . 19
4.8 Generierung einer Siunusschwingung durch Rechnen . . . 20
4.9 Generierung einer Siunusschwingung durch Table-Lookup . . . 20
4.10 Amplitudenveränderung eines Audiosignals . . . 22
4.11 Frequenzveränderung durch Inkrementveränderung eines Phasors . . . 23
4.12 Frequenzveränderung durch ganzzahlige/nichtganzzahlige Multiplikation eines Phasors 24
4.13 Frequenzveränderung durch ganzzahlige/nichtganzzahlige Multiplikation eines Phasors und Normalisierung nach der Multiplikation . . . 25

5.1 Prinzip von Waveshaping . . . 28
5.2 Beispiel von Waveshaping (Verzerrung durch Clipping) . . . 30
5.3 Simulation eines analogen Verzerrers mit Waveshaping . . . 31
5.4 Waveshaping mit Cosinusfunktionen . . . 32
5.5 Cosinus als Transferfunktion über erweiterten Wertebereich . . . 33
5.6 in x Richtung gestauchter Cosinus als Transferfunktion über erweiterten Wertebereich . 33
5.7 Waveshaping mit Chebyshev Polynomen . . . 34
5.8 Erzeugung eines bandbegrenzten Pulses durch Waveshaping mit der Exponentialfunktion 35

5.9 Übung: Errechnung von Audiowerten 37

6.1 Abschnittsweise definierte Hüllkurve mit linearen Segmenten 39
6.2 Abschnittsweise definierte Hüllkurve mit exponentiellen Segmenten 39
6.3 ADSR Hüllkurve 40
6.4 Besipiele verschiedener ADSR Hüllkurven 41

Kapitel 1

Vorwort

Dieses Buch ist als Einführung in die digitale elektronische Klangsynthese gedacht. Es gibt mittlerweile sehr viele verschiedene computerbasierte Systeme zur Klangsynthese in Echtzeit. Alle diese Systeme haben Vor- und Nachteile. Dieses Buch verfolgt den Ansatz, die vorgestellten Klangsyntheseverfahren zunächst theoretisch zu erklären und anschließend in verschiedenen Systemen praktisch zu realisieren. Durch die direkte Vergleichbarkeit verschiedener Realisationen kann sowohl ein tieferes Verständnis erzielt werden, als auch durch Abwägung der spezifischen Vor- und Nachteile der Systeme Perspektiven eröffnet werden, welches System für welchen Anwendungsfall besser geeignet ist.

In Kapitel 2 werden verschiedene Computermusiksysteme in einer Übersicht vorgestellt.

In diesem Buch werden davon drei Systeme behandelt:

- pure data
- SuperCollider
- incudine, ein auf der Programmiersprache Common Lisp basierendes System

Kapitel 3 beschreibt die Installation dieser drei Systeme. Da die Systeme sich in fortlaufender Entwicklung befinden, kann es sein, dass dieser Teil verhältnismäßig schnell veraltet. Daher wird empfohlen, in einem solchen Fall die jeweiligen Installationsanleitungen online zu recherchieren.

Kapitel 4 behandelt Grundlagen der digitalen Signalverabeitung, die unabhängig von den verwendeten Systemen allen Systemen gemeinsam ist und die Voraussetzung für ein Verständnis der behandelten Konzepte bildet.

Kapitel 5 schließlich enthält die verschiedenen Beispiele, immer gegliedert in theoretische Beschreibung und praktische Umsetzung. Aufgaben ergänzen das Buch.

- Klangsynthese aller Art in Nicht-Echtzeit und Echtzeit.
- algorithmische Komposition.
- Entwicklung von Audioapplikationen, auch mit grafischen Benutzeroberflächen.

Wichtiger Hinweis

Dieses Buch ist ’nicht’ als umfassende Einführung in die besprochenen Systeme zu verstehen. Insbesondere die Erstellung grafischer, interaktiver Benutzeroberflächen, die oft unerlässlich für den Einsatz in Konzerten sind, oder die Anbindung externer Eingabegeräte, wie Controller oder Tastaturen werden nicht behandelt.

Dieses Buch entstand aus der Notwendigkeit einer deutschsprachigen Einführung zu Unterrichtszwecken für Kompositionsstudierende. Hier ist das Buch auch als pdf Datei abrufbar.

Aufgaben und eine Bibliografie zur weitergehenden Vertiefung der Kenntnisse schliessen das Buch ab.

Kapitel 2

Übersicht über unterschiedliche Computermusiksysteme

Grob lassen sich Computermusiksysteme in drei ”Klassen” gliedern, die sich in der Form der Bedienung unterscheiden.

Für einen Vergleich gängiger Systeme siehe auch: Comparison of audio synthesis environments

2.1 Grafische, anwenderorientierte Programme

Beispiele: fuzzy:”https://de.wikipedia.org/wiki/Reaktor_(Software)”[Reaktor] Zynaddsubfx, amsynth, Loomer Aspect, 6PM, ALSA Modular Synth

Abbildung 2.1: Beispiel einer Bedienoberfläche von Reaktor

Dies sind Programme, die zumeist fertige Module zur Klangsynthese enthalten, manchmal bereits in einer vorkonfigurierten Anordnung. Grafisch orientieren sich dies Programme zumeist an existierenden Geräten aus der Geschichte der elektronischen Musik, wie beispielsweise modularen Analogsynthesizern. Der Nutzer kann diese vorkonfigurierten Module ähnlich den Vorbildern mit grafischen Bedienelementen, wie Knöpfen, Schiebereglern, etc. mit der Maus oder angeschlossenen Hardwarecontrollern steuern. Ein Vorteil dieser Systeme ist eine vergleichsweise kurze Einarbeitungszeit. Mittlerweile lassen sich diese Systeme sehr variabel konfigurieren, wodurch sie erheblich vielseitiger werden, als Ihre analogen Vorbilder. Der Vorteil der Vorstrukturierung in Modulen, die oft von

sehr professionellen Programmierern entwickelt wurden und dementsprechend gute klangliche Eigenschaften besitzen, ist aber zugleich auch der Nachteil dieser Systeme, da dadurch ihre Flexibilität eingeschränkt wird.

2.2 Grafische, patcherbasierte Systeme

Beispiele: Max/MSP, Pure Data

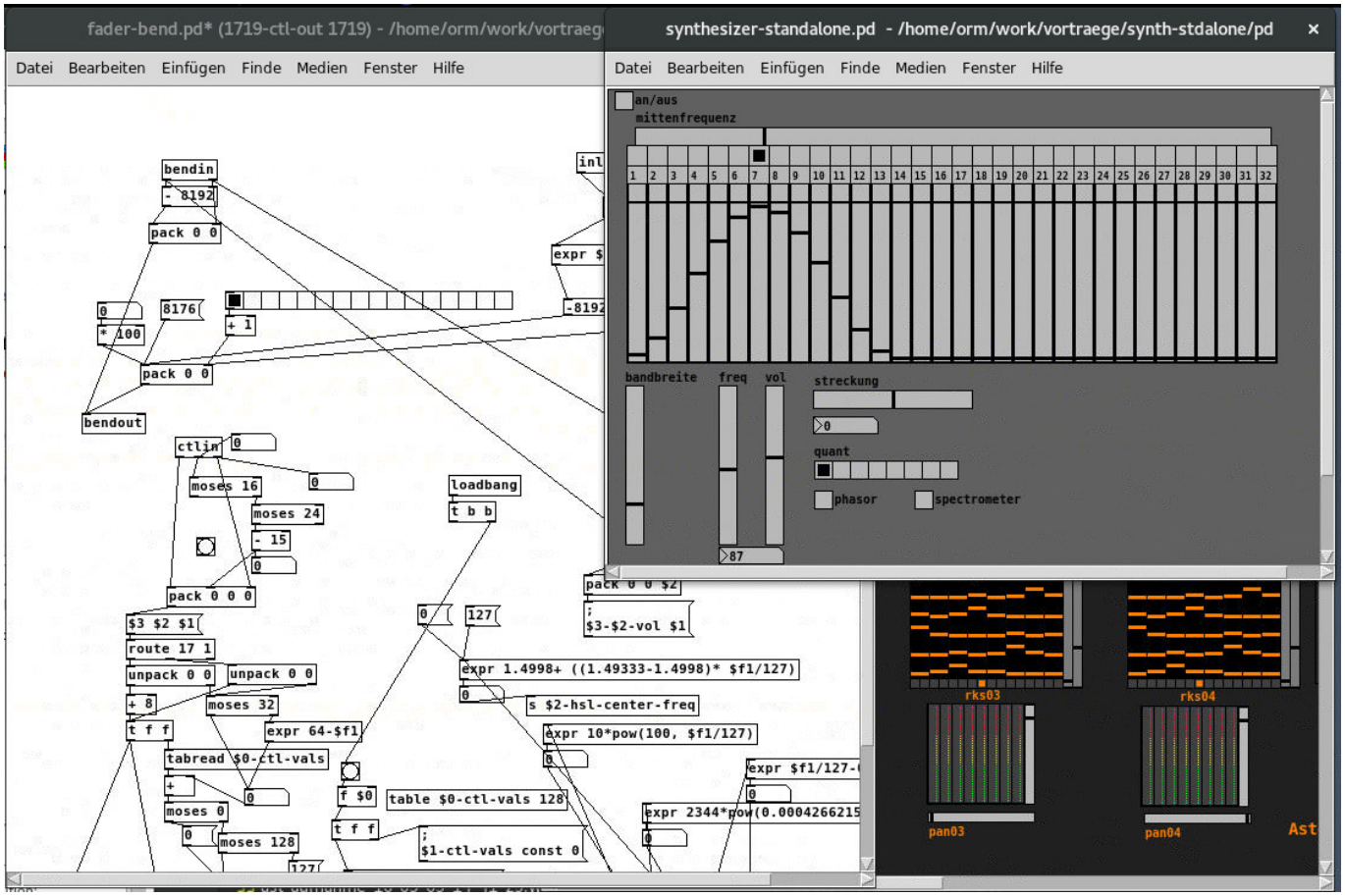
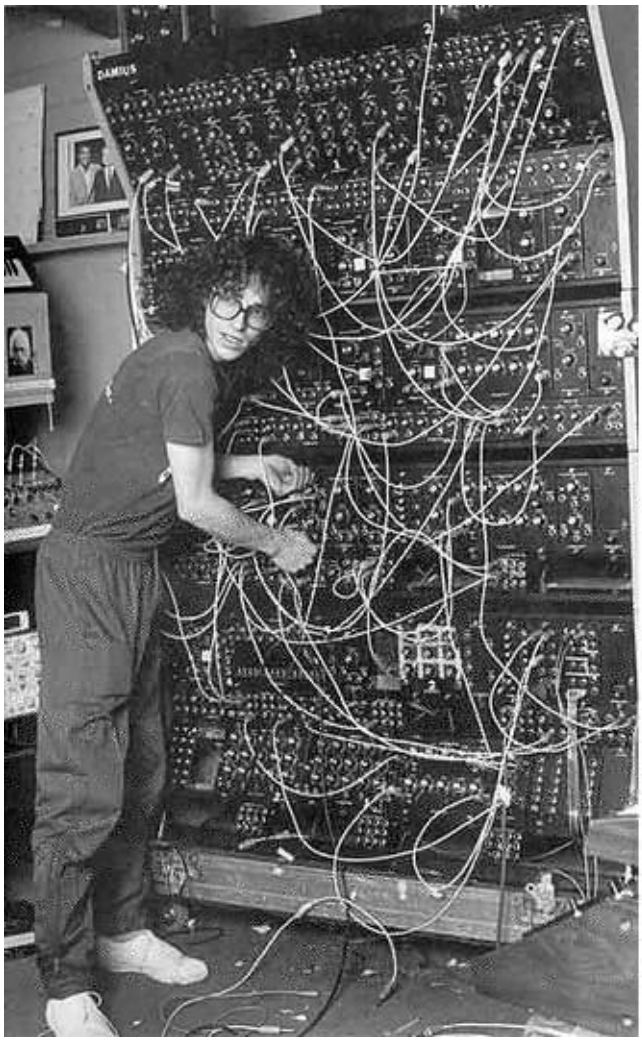


Abbildung 2.2: Beispiel eines pure data (pd) Patches

Darunter versteht man Programme, die über eine grafische Benutzeroberfläche bedient werden: Mit Hilfe eines grafischen Editors werden verschiedene verfügbare Module (sogenannte ”Objekte”) grafisch angeordnet. Ein- und Ausgänge dieser Module können mit Hilfe von sogenannten ”patchcords” verbunden werden. Eine solche Anordnung von Modulen und ihren Verbindungen nennt man einen ”Patch”. In dem System repräsentieren die Module Rechenoperationen, die sowohl elementare Operationen, als auch komplizierte Verfahren zur Audiosynthese enthalten können. Insofern ist das Erstellen eines Patches eine Form der Programmierung.



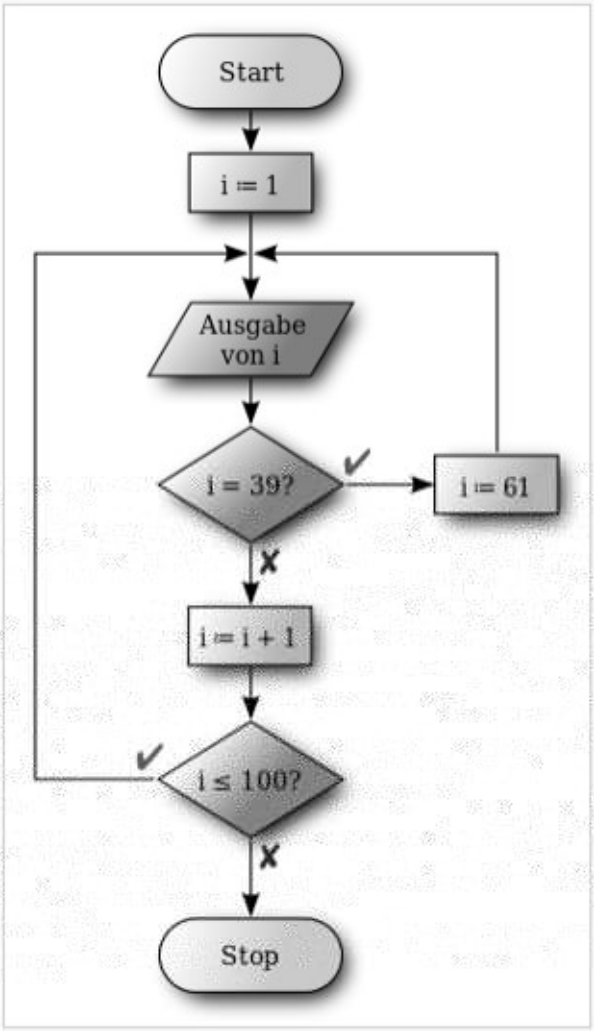


Abbildung 2.3: Modularer Analogsynthesizer und Flussdiagramm (Programmablaufplan) als Modelle für patcherbasierte Computermusiksysteme, wie Max/MSP oder pd.

Ein Patch verbindet dabei das bereits aus der vorherigen Gruppe von Computermusiksystemen bekannte Konzept von virtuellen, miteinander verbundenen Komponenten und Modulen mit einer Art Flussdiagramm der Programmverarbeitung des jeweiligen Patches (siehe Abbildung 2.3). Der Vorteil dieser Systeme ist ihre große Flexibilität und Offenheit.

Aufgrund der grafischen Bedienung kann eine interaktive, grafische Steuerung direkt in den Patch integriert werden (einige Objekte sind grafische, interaktive Komponenten, wie Regler, Schalter, Anzeigen, Menüs, etc.). Einen weiteren Vorteil bietet ein -gegenüber den nachfolgend dargestellten textbasierten Systemen- vereinfachter Einstieg für Programmieranfänger durch die Anschaulichkeit grafischer Module.

Nachteil dieser Systeme ist die schlechte Lesbarkeit komplexerer Patches und damit eine im Vergleich zu generellen Programmiersprachen schwierige Wartung. Auch das Erstellen eines Patches kann sich durch die Notwendigkeit, Datenflusszusammenhänge mit grafischen Linien darzustellen, mitunter recht umständlich gestalten. Bei zunehmender Komplexität eines Patches kann sich zudem die vermeintlich anschauliche grafische Visualisierung als trügerisch erweisen, da die zugrundeliegenden Prozesse zu komplex sind, um von der Grafik angemessen dargestellt zu werden. Darüber hinaus ist in der Regel ein zumindest grundlegendes Verständnis digitaler Signalverarbeitung erforderlich, um zu zufriedenstellenden Ergebnissen zu kommen.

2.3 Textbasierte Systeme

Beispiele: Csound, SuperCollider, incudine, cl-collider, Faust, ChucK, CLM, Common Music

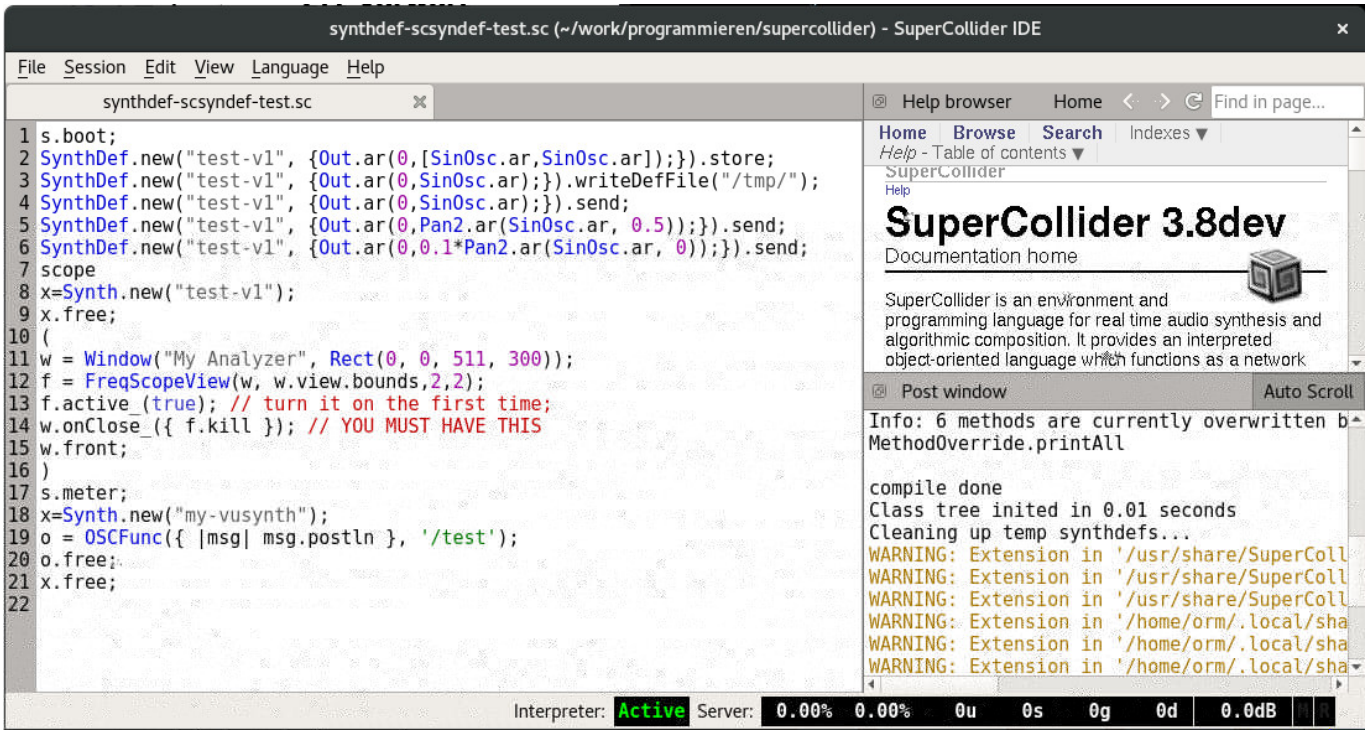


Abbildung 2.4: Beispiel einer SuperCollider Session

Vom Arbeitsprinzip her handelt es sich bei diesen Systemen um die ältesten Systeme der computerbasierten Klangsynthese (Csound, dessen 1. Version 1985 entstand, hat seine Wurzeln im MUSIC System von 1957 und dessen Nachfolgern). Jedoch sind einige Systeme auch relativ neu und die oben genannten Systeme werden noch immer aktiv weiterentwickelt.

Alle Systeme gleichen sich im grundlegenden Arbeitsablauf:

1. Zunächst wird eine Textdatei erstellt, die eine Beschreibung der zu erzeugenden Klänge und Abläufe in einer speziellen Syntax enthält.
2. Dieser Text wird dann über einen speziellen Befehl dem jeweiligen System zur Auswertung übergeben.

   Das Resultat ist dann entweder ein Klang, der direkt über die Ausgänge des Computers ausgegeben wird, oder auch eine Klangdatei (soundfile), die dann mit einem separaten Abspielprogramm hörbar gemacht werden kann.

Es handelt sich bei diesen Systemen aufgrund der speziellen Syntax immer auch um Programmiersprachen. Im Fachjargon spricht man in einem solchen Fall von einer Domänenspezifischen Sprache.

Einige der existierenden Systeme sind völlig eigenständig (Csound, Supercollider, Faust, Chuck), andere Systeme (CLM, incudine, cl-collider) erweitern bereits existierende, sogenannte Universelle Programmiersprachen und sind daher in ihrer Syntax an ihre jeweilige Basissprache angelehnt. Der Vorteil dieser Systeme ist eine oft hervorragende Anschlußfähigkeit an bereits existierende Module aus anderen, nicht klangsynthesespezifischen Bereichen (z.B. Mathematik, Grafikverarbeitung, Webprogrammierung, Datenbanken, Notation).

Ein weiterer Vorteil ist eine bei ausgereiften Systmenen deutlich bessere Strukturierbarkeit, als dies mit grafisch orientierten Systemen möglich ist: Nicht ohne Grund sind Programmiersprachen, mit denen sehr umfangreiche Programme erstellt werden, textbasiert. Zusammenhänge lassen sich in den allermeisten Fällen textuell konziser darstellen, insbesondere dann, wenn die Projekte umfangreicher und komplexer werden. Zudem haben textbasierte Programmiersprache mittlerweile eine lange Tradition und es sind viele Programmierkonzepte entwickelt worden, die sich insbesondere bei der Verwaltung von komplexen, oder auf mehrere Programmierer verteilte Aufgaben bemerkbar machen.

Ein gewichtiger Nachteil ist die Einstiegshürde, insbesondere für Programmieranfänger. Die Benutzeroberfläche wirkt abstrakt und wenig intuitiv und man benötigt nicht selten Grundlagenwissen in Programmierung, um mit solchen Systemen effizient arbeiten zu können. Insofern sind diese Systeme sicher nicht für jeden Komponisten mit Interesse an elektronischer Klangverarbeitung geeignet.

Die erheblich leichtere Wartung von Projekten in diesen Systemen, die auch umfangreiche Projektdateien über Jahre hinweg verständlich, lesbar und damit gut wiederverwendbar oder modifizierbar hält, könnte allerdings ein guter Grund dafür sein, die anfänglichen Hürden zu überwinden.

Kapitel 3

Installationsanleitungen

Die Installation der drei behandelten Systeme (pure data, SuperCollider und incudine) sind unterschiedlich schwer: Für Pure Data und SuperCollider gibt es im Internet fertige Installationspakete, ein sofort lauffähiges System erzeugen. Die Installation von Incudine ist etwas aufwändiger und insbesondere bei Windows nur für erfahrenere Nutzer zu empfehlen.

3.1 Installation von pure data

Das Programms kann von dieser Downloadseite heruntergeladen und durch Doppelklick installiert werden. Dabei sollte man darauf achten, die richtige Version für das verwendete Betriebssystem herunterzuladen.

Nach Doppelklick des Programms sollte sich das in Abbildung 3.1 dargestellte Fenster öffnen.

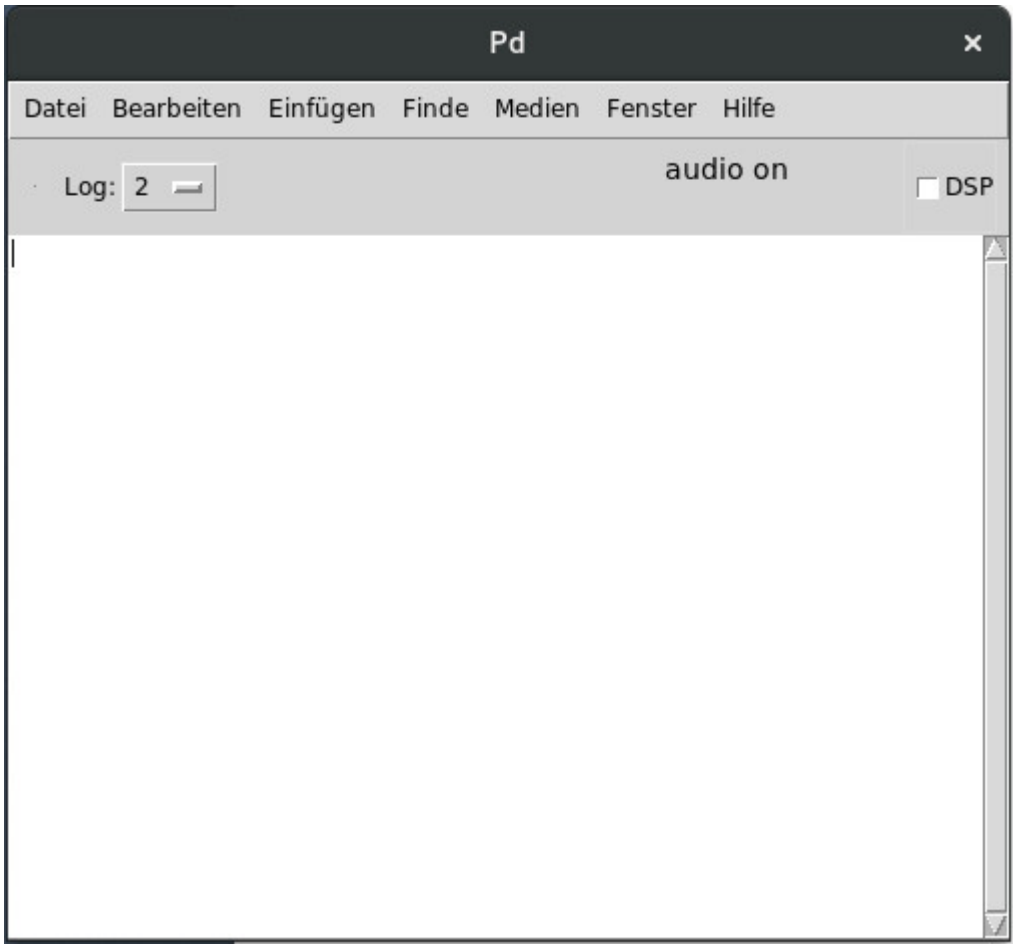


Abbildung 3.1: Startbildschirm von pure data

Um die Klangausgabe zu testen, öffnet man über das Menü ”Medien→Teste Audio und Midi...” das Audiotestfenster. In diesem Fenster sollte ein Klick auf die Schaltfläche neben der ”80” unter ”TEST TONES” in der linken oberen Mitte des Fensters einen Sinuston von 440 Hz auf den Audioausgängen des Computers ausgeben (siehe den rot umkreisten Bereich von Abbildung 3.2). Bei Schließen des Fensters sollte der Ton automatisch ausgeschaltet werden.

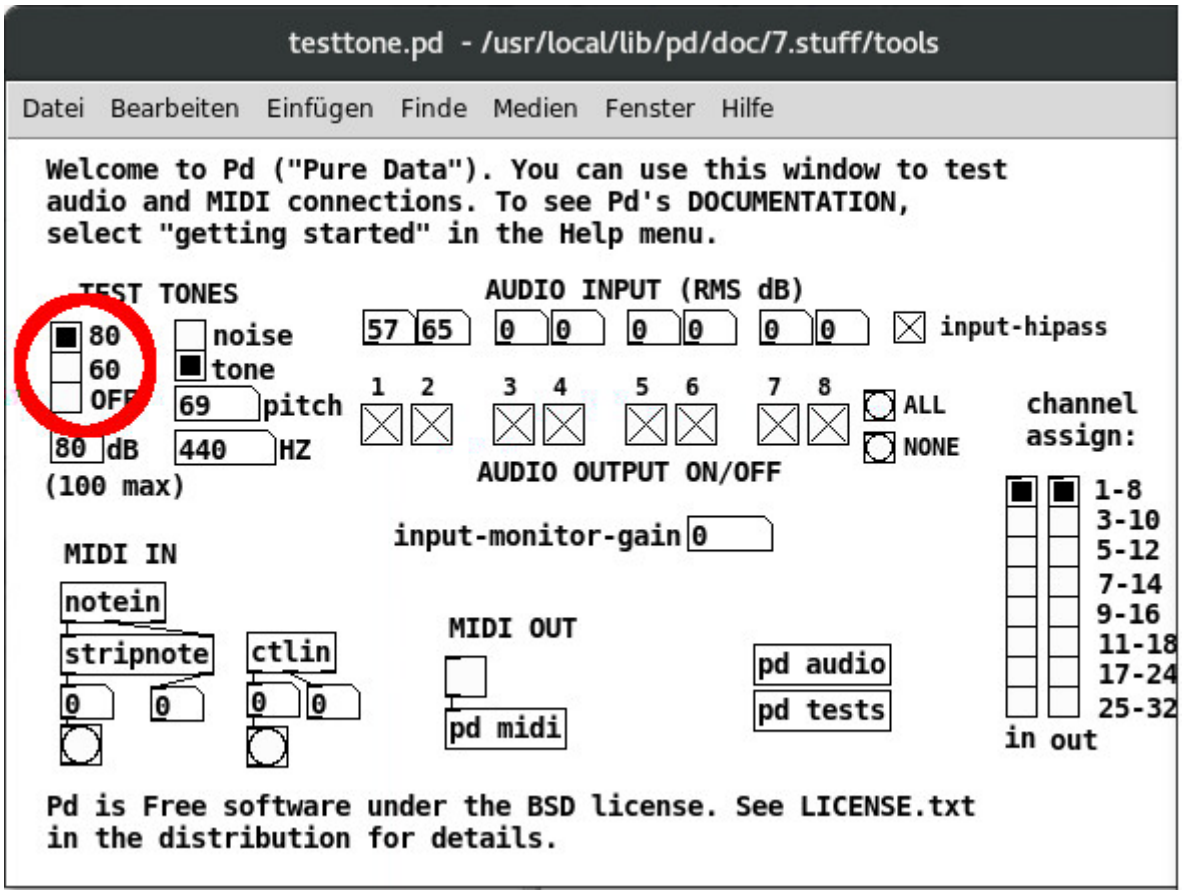


Abbildung 3.2: Audiotest Fenster von pure data

3.2 Installation von SuperCollider

Ausführbare Super Collider Programme können von dieser Seite heruntergeladen werden. Nach Start des Programms sollte sich ein Fenster ähnlich der Abbildung 2.4 öffnen.

3.3 Installation von incudine

Folgende Komponenten müssen zunächst auf dem Computer installiert und konfiguriert werden:

- Der Texteditor Emacs
- Der sbcl Compiler für die Programmiersprache Common Lisp
- quicklisp
- jack

Emacs und sbcl haben unterschiedliche Versionen für verschiedene Betriebssysteme und müssen zuerst installiert werden. Anschließend muss der Texteditor Emacs so konfiguriert werden, dass eine Verbindung zu sbcl hergestellt werden kann, um die Programmiersprache aus Emacs heraus nutzen zu können. Zuletzt wird quicklisp entsprechend der Anleitung auf der quicklisp Webseite eingerichtet.

Nachdem quicklisp eingerichtet und gestestet ist, werden mit dessen Hilfe folgende lisp Pakete installiert:

- quicklisp-slime-helper
- incudine

Die Installation von quicklisp-slime-helper wird auf der quicklisp Seite beschrieben.

Incudine lässt sich von dieser Seite herunterladen (siehe ”Download”; dafür ist eine Installation von Git erforderlich).

Um incudine bei quicklisp bekannt zu machen, muss der oberste Ordner in das Verzeichnis ”<home>/quicklis projects” verschoben werden.

Bevor incudine gestartet werden kann, muss jack installiert sein. Informationen der Installation für die verschiedenen Betriebssysteme finden sich auf der Webseite des Projektes.

Anschließend sollte nach Start von Common Lisp in Emacs die Eingabe von

```
(ql:quickload ”incudine”)
```

in der REPL incudine starten und verfügbar machen (für eine Einführung siehe die Tutorials von der incudine Webseite).

Kapitel 4

Grundlagen der digitalen Signalverarbeitung

4.1 Analog-Digital- und Digital-Analog-Wandlung

Digitale Signalverarbeitung (englisch ’digital signal processing’, abgekürzt DSP) bezeichnet die Verarbeitung von Signalen unterschiedlichster Art (Audiosignale, Videosignale, Funksignale, etc.), die in Form von ’diskreten Zahlenfolgen’ (sogenannten ’Samples’) vorliegen. Dies bildet einen Unterschied zur ’analogen Signalverarbeitung’, die ’kontinuierliche Signale’ anstelle der diskreten Zahlenfolgen behandelt.

Die digitale Signalverarbeitung hat seit den 1980er Jahren im Zusammenhang mit der allgemeinen Verbreitung der Computertechnologie stark an Bedeutung zugenommen und die bis dahin vorherrschende analoge Signalübertragung und -verarbeitung in vielen Bereichen der Übertragungstechnik verdrängt.

Wenn die zu verarbeitenden Signale natürlichen Ursprungs sind, werden sie technisch zumeist in mehreren Schritten in digitale Signale übersetzt. Dieser Übersetzungsvorgang nennt sich ’Analog-Digital-Wandlung’ (englisch ’analog digital conversion’, abgekürzt ADC). Der obere Teil von Abbildung 4.1 zeigt den Vorgang dieser Übersetzung am Beispiel der Umwandlung von Schallwellen:

1. Das Schallsignal im Raum wird zunächst mit Hilfe eines ’Mikrofons’ erfasst und in ein analoges (elektrisches) Signal umgewandelt.
2. Der ’Analog-Digital-Wandler’ setzt dieses Signal dann in diskrete Zahlenfolgen um.

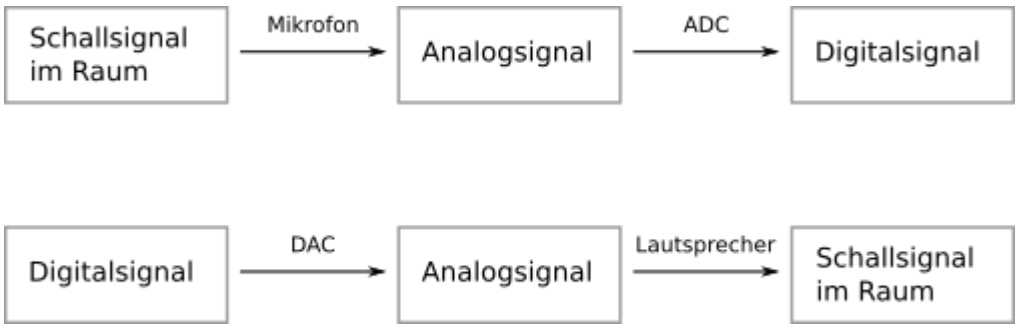


Abbildung 4.1: Schema von Analog-Digital- und Digital-Analog-Wandlung

Im unteren Teil der Abbildung 4.1 ist der umgekehrte Fall dargestellt:

1. Der ’Digital-Analog-Wandler’ setzt diskrete Zahlenfolgen in ein analoges (elektrisches) Signal um.
2. Das elektrische Signal wird nach einer Verstärkung mit Hilfe eines ’Lautsprechers’ in ein Schallsignal umgewandelt und in einen Raum abgestrahlt.

Die AD bzw. DA-Wandlung kann dabei an verschiedenen Stellen erfolgen:

1. In einem Audiointerface im Computer

   In diesem Fall wird das Mikrofon\slash{}der Lausprecher über einen Analoganschluss (zumeist Miniklinke) direkt mit dem Computer verbunden.

2. In einem externen Audiointerface

   Das externe Audiointerface ist mit dem Computer über eine Digitalschnittstelle (bspwse. USB) verbunden. Das Mikrofon/der Lausprecher sind mit dem Interface über einen Analoganschluss (XLR/Klinke) verbunden.

3. In einem Mischpult

   In diesem Fall sind Lautsprecher\slash{}Mikrofone mit dem Mischpult verbunden und das Mischpult verfügt über digitale Ein\slash{}Ausgänge (ADAT, MADI, AES\slash{}EBU), die mit einem Computerinterface verbunden sind.

4.2 Animation der Analog-Digitalwandlung von Schallwellen

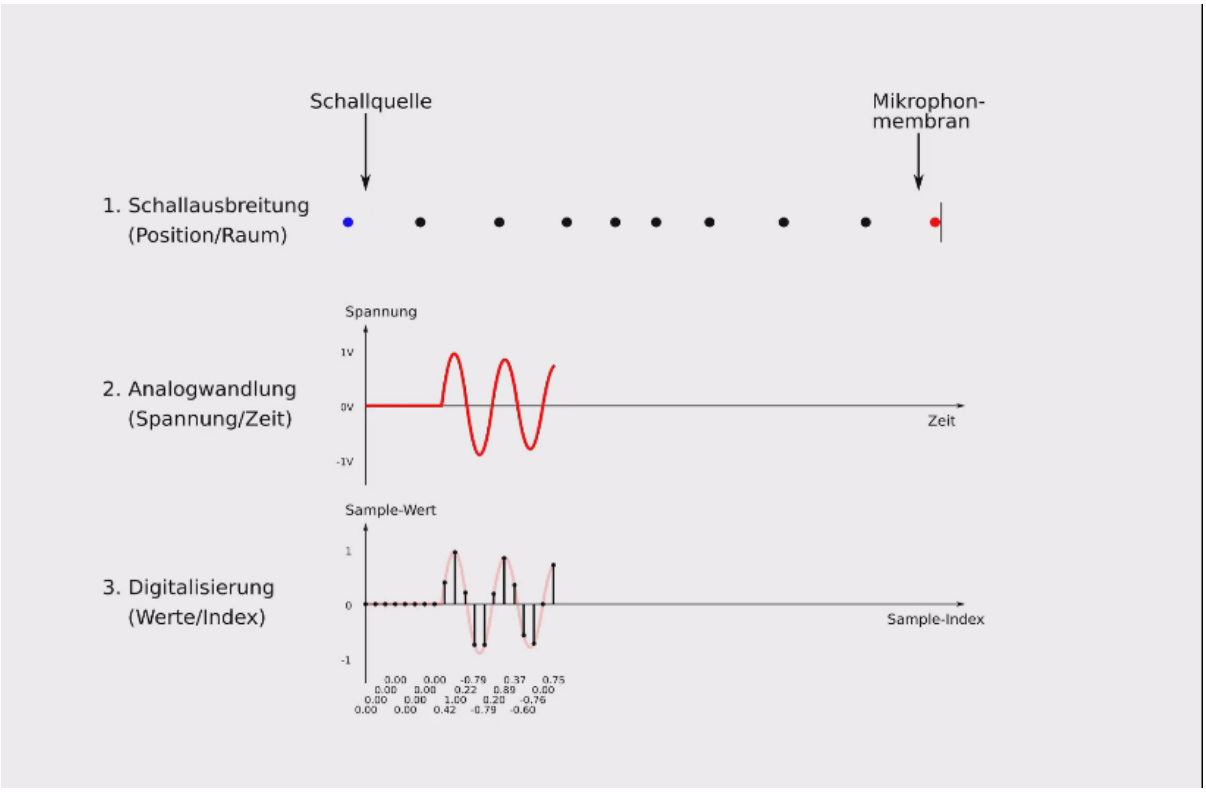


Abbildung 4.2: Animation der AD-Wandlung von Schallwellen

(Link zum Video)

Die Animation verdeutlicht die verschiedenen Schritte der Wandlung von Schallwellen in digitale Signale.

Die umgekehrte Umwandlung digitaler Signale in Schallwellen ist gleich, jedoch muss die Reihenfolge der Zeilen umgekehrt werden und die Mikrophonmembran durch eine Lautsprechermembran ersetzt werden.

4.2.1 Erklärung der Animation

4.2.1.1 Die obere Zeile der Grafik

Eine Schallwelle breitet sich in Form einer 'Longitudinalwelle' von Luftpartikeln im Raum aus. Der Abstand der Punkte kennzeichnet dabei den 'Luftdruck': Eng beieinander stehende Punkte entsprechen einem hohen Luftdruck, weit auseinanderliegende Punkte einem niedrigen Luftdruck. Insofern ist eine Schallwelle als 'Propagation einer Druckschwankung im Raum' zu verstehen.[1] Diese Druckschwankung führt zu einer Bewegung der Membran in einem Mikrophon, die wiederum (beipielsweise mit Hilfe des Prinzips der elektromagnetischen Induktion) in elektrischen Strom umgewandelt wird.[2]

4.2.1.2 Die mittlere Zeile der Grafik

zeigt das vom Mikrofon erzeugte elektrische Signal (rote Linie):

- Die Bewegung des roten Luftpartikels an der Mikrofonmembran in der ersten Zeile wird dabei (um 90 Grad gegen den Uhrzeigersinn gedreht) als 'zeitlicher Verlauf' von links nach rechts dargestellt.
- Diese Darstellung entspricht dem 'Spannungsverlauf' des elektrischen Stroms in der Zeit: Eine Auslenkung des roten Luftpartikels der ersten Zeile nach rechts entspricht einer 'positiven' Spannung, eine Auslenkung nach links einer 'negativen' Spannung. Ist der Luftpartikel in der mittleren Ruhelage, so ist die Spannung = 0 Volt.
- Im analogen Audiobereich sind die Geräte in der Regel so eingestellt, dass der maximal verarbeitbare Wert des Spannungspegels bei +/- 1V ist. Treten höhere Pegel auf, so führt dies zu 'Verzerrungen' des Signals.

4.2.1.3 Die untere Zeile der Grafik

schließlich zeigt die 'Digitalisierung' des Spannungsverlaufs:

- Die Spannungswerte werden dabei in regelmäßigen Zeitintervallen gemessen und der gemessene Wert (die Länge der senkrechten schwarzen Linien mit Punkt am Ende) als Zahl weitergeleitet.
- Die Taktgeschwindigkeit dieser Messung wird 'Abtast-' oder 'Samplerate' genannt. Für qualitativ hochwertige Audiosignale sind Sampleraten von 44100, 48000, 88200 oder 96000 Werten/Sekunde üblich, es werden allerdings für manche Zwecke auch noch höhere Samplingraten eingesetzt.
- Die Zahlenwerte werden als geordnete Zahlenfolge im Computer zur Weiterbearbeitung gespeichert. Hierbei ist der 'Index' der Zahlen ein Maß für die verstreichende 'Zeit' des analogen Signals aus der mittleren Zeile. Das Zeitintervall zwischen zwei aufeinanderfolgenden Zahlen errechnet man, indem man den 'Kehrwert' der Samplerate bildet. Bei einer Samplerate von 44100 Samplen/Sekunde ergibt sich also eine Zeitdifferenz von 1/44100 s = 0.022675737 ms zwischen aufeinanderfolgenden Zahlen.

[1] Die Zeichnung ist vereinfacht, da sich Schallwellen ohne Hindernis nicht nur in eine Richtung, sondern 'kugelförmig' in alle Richtungen ausbreiten.

[2] siehe hierzu auch: https://de.wikipedia.org/wiki/Dynamisches_Mikrofon

- Die Samplerate bestimmt, welche maximale Frequenz durch die Abtastung noch darstellbar ist. Es gilt, dass die maximal darstellbare Frequenz bei der Hälfte der Abtastrate erreicht ist. Diese Frequenz nennt sich 'Nyquist-Frequenz'. Bei einer Abtastrate von 44100 Werten/Sekunde ergibt sich also eine obere Grenzfrequenz von 22050 Hz. Wenn höhere Frequenzen abgetastet werden, ergibt sich aufgrund eines 'Aliasing' genannten Phänomens eine niedrigere Frequenz, die an der oberen Grenzfrequenz (Samplerate\slash{}2) 'gespiegelt' (und im Vorzeichen umgekehrt) auftaucht.

  Beispiel

  Wird ein Signal von 25000 Hz mit einer Samplerate von 48000 Hz abgetastet, so ergibt sich im Ergebnis eine Frequenz von -23000 Hz (Abtastrate/2=24000 Hz; 25000Hz = 24000+1000 Hz; gespiegelt ist das 24000-1000 Hz = 23000 Hz, mit negativem Vorzeichen also -23000 Hz [3]).

- Ein anderer wichtiger Wert bei der Digitalisierung von Signalen ist die Genauigkeit der Zahlendarstellung. Die Genauigkeit wird in 'Bit' (die Anzahl von 0-en und 1-en in binärer Zahlendarstellung). Üblich sind 16, 24 oder 32 bit. Das entspricht 2^16, 2^24, oder 2^32 verschiedenen Werten. Je höher diese Zahl ist, desto geringer ist das digitale Rauschen, das sich insbesondere bei sehr leisen Signalpegeln auf die Klangqualität auswirken kann.

Abbildung 4.2 verdeutlicht die verschiedenen Schritte der Wandlung von Schallwellen in digitale Signale.

4.3 Darstellung komplexer Audiosignale

Wie aus Abbildung 4.2 ersichtlich, entspricht die zeitliche Darstellung des Spannungsverlaufs dem Verlauf des 'Luftdrucks' an einer Position des Raums. Wird dieser Luftdruckverlauf über eine Lautsprechermembran in den Raum übertragen, so lässt sich mit einer einzigen Linie der Luftdruckverlauf auch komplexer Audiosignale, wie beispielsweise dem Klang eines großen Orchesters darstellen und bei Lautsprecherabstrahlung in einen Raum für das menschliche Ohr erkennbar wiedergeben. Wird ein einziges Signal für die Schallabstrahlung verwendet, so spricht man von einem 'Monosignal'. fuzzy:fig:beethoven-5[fig:beethoven-5] zeigt beispielsweise den Luftdruckverlauf einer mit einem Mikrofon angefertigten Orchesteraufnahme des Beginns der 5. Sinfonie von Ludwig von Beethoven in verkleinerter Darstellung. Wird diese Grafik stark vergrößert, stellt man fest, dass es sich im Detail um eine einzige Linie handelt, die die Luftdruckschwankungen am Ort des für die Aufnahme verwendeten Mikrofons beschreibt.

[3] Das negative Vorzeichen führt bei einem Audiosignal dazu, dass die Samplewerte an der x-Achse gespiegelt werden; bei Audiosignalen spricht man dabei von 'negativer' bzw. 'gedrehter' Phase, da dies bei Analogsignalen dem Vertauschen der beiden elektrischen Leitungen entspricht, die ein Analogsignal transportieren.

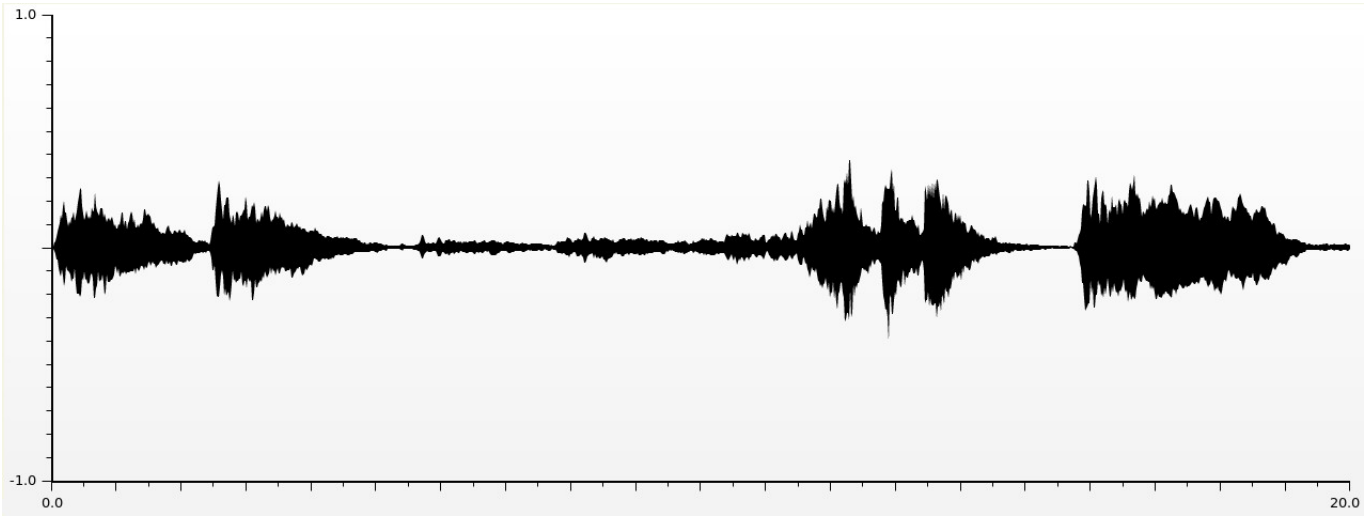


Abbildung 4.3: Die ersten 20 Sekunden von Beethovens 5. Sinfonie in Wellenformansicht

Die ’digitale elektronische Klangsynthese’ beschäftigt sich mit der Erzeugung solcher Signale. Anders ausgedrückt geht es dabei also um die Generierung von Zahlenfolgen mit Hilfe eines Computers, die es ermöglicht, vielseitige akustische Verläufe und Phänomene zu gestalten.

Die in fuzzy:Übersicht über unterschiedliche Computermusiksysteme[Kapitel 2] besprochenen Systeme lassen sich als Hilfsmittel auffassen, eine solche Zahlenfolge zu generieren und für den Anwender in übersichtlicher Form kontrollier- und damit gestaltbar zu machen.

4.4 Räumliche Schallabstrahlung

Sollen klangliche Phänomene räumlich in Erscheiung treten, so ist dies mit Hilfe mehrerer der im vorigen Abschnitt beschriebenen Monosignale möglich, die über verschiedene im Raum verteilte Lautsprecher wiedergegeben werden. (Abbildung 4.4).

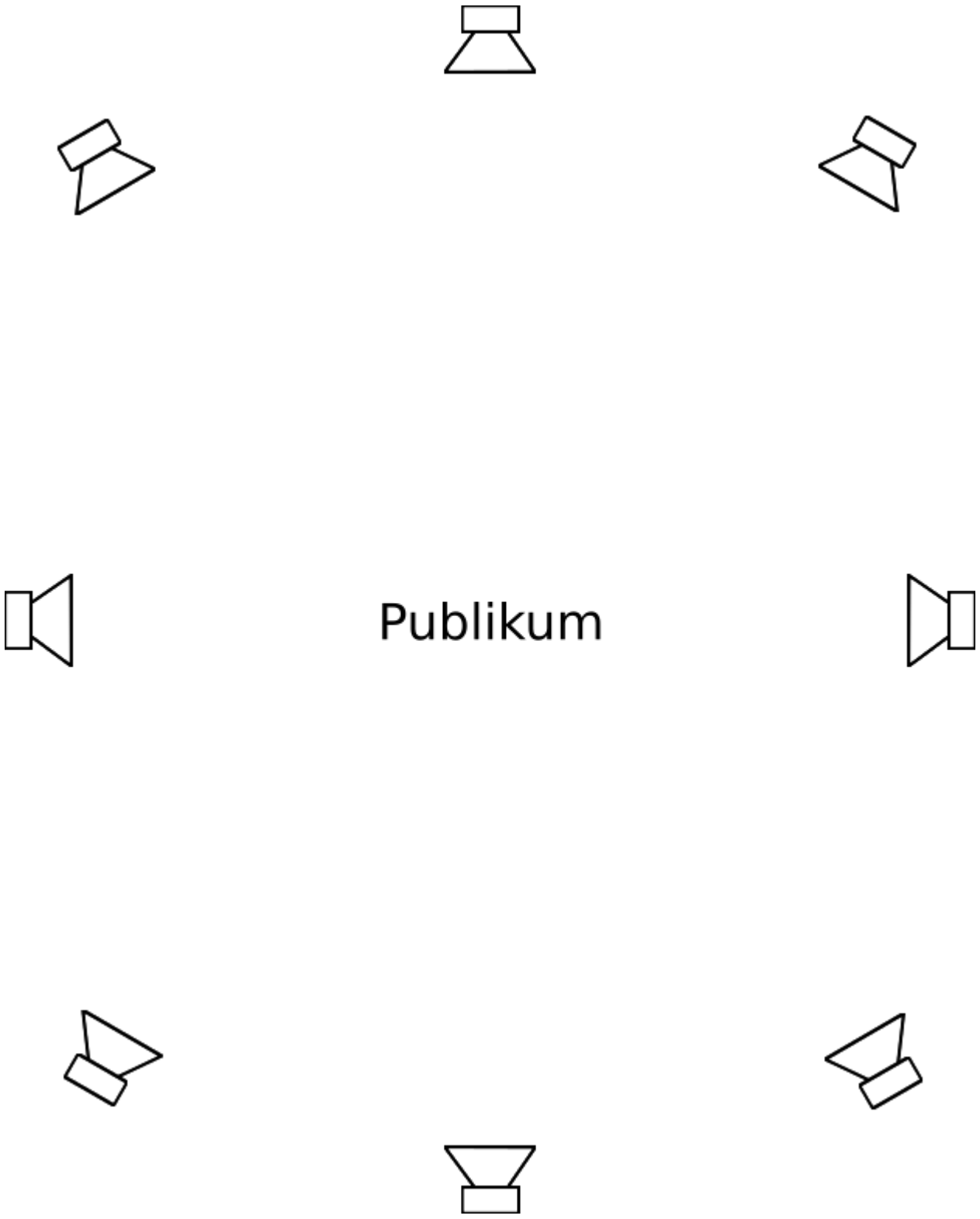


Abbildung 4.4: Beispiel einer 8-kanaligen Lautsprecheranordnung für räumliche Schallabstrahlung

4.5 Blockverarbeitung digitaler Signale und Latenz

Aus fuzzy:Die untere Zeile der Grafik[Abschnitt 4.2.3] geht hervor, dass bei der AD/DA Wandlung Zahlenfolgen mit einer kontinuierlichen, konstanten Samplerate verarbeitet werden. Diese Auflösung in getaktete Folgen 'einzelner Zahlen' geschieht jedoch erst im Wandler direkt. Die Kommunikation zwischen dem Wandler und dem Betriebssystem des Computers oder den Audioprogrammen geschieht aus Effizienzgründen in 'Blöcken', in denen mehrere Zahlen der Zahlenfolge zusammengefasst werden. Die Größe dieser Blöcke sind zumeist Zweierpotenzen. Typische Werte sind 128, 256, 512 oder 1024 Samples (siehe Abbildung 4.5).

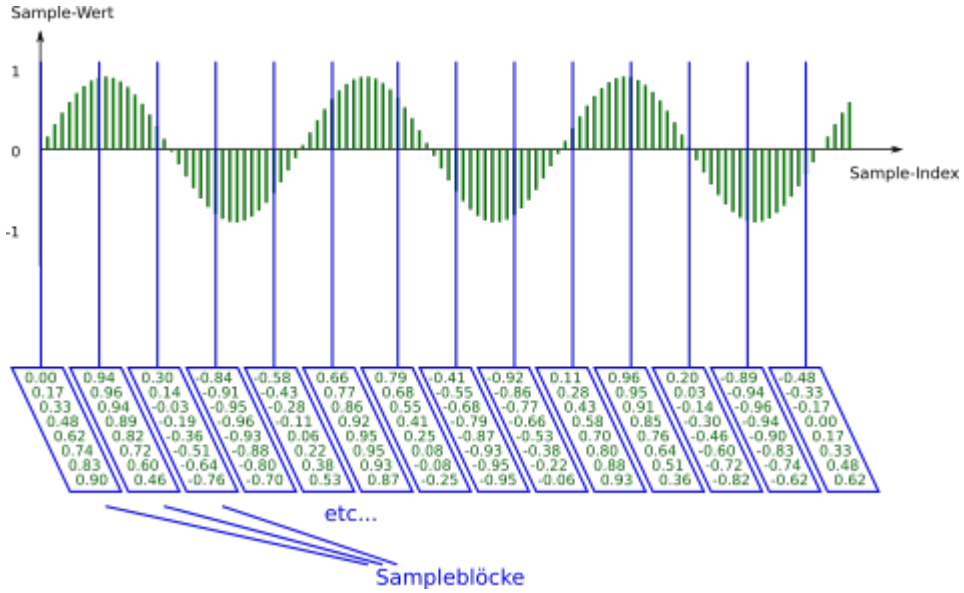


Abbildung 4.5: Blockverarbeitung von Samples (Blockgröße: 8 Samples)

Dieser Umstand führt zu einer wichtigen Konsequenz bei der 'Echtzeitverarbeitung' von Audiosignalen: Da diese Blöcke zunächst gefüllt werden müssen, bevor sie an den Wandler bzw. Computer weitergeleitet werden, ergibt sich daraus eine Zeitverzögerung zwischen dem Signal und seiner Weiterleitung, die dem Quotienten aus Blockgröße und Samplerate entspricht. Diese Zeitverzögerung wird 'Latenz' (englisch 'latency') genannt.

Wie aus Abbildung 4.5 hervorgeht, bedeutet dies zugleich, dass auch die Blöcke -wie die einzelnen Samples- mit einer konstanten Rate zwischen DA/AD Wandler und Computer übertragen werden. Diese Rate ergibt sich aus dem Quotienten von 'Samplerate\slash{}Blockgröße'. Bei einer Samplerate von 48000 Hertz und einer Blockgröße von 256 Zahlen (Samples) ergibt sich also eine Taktrate von 48000 Hz/256 = 187.5 Hz, mit der diese Blöcke zwischen Wandler und Computer übertragen werden.

4.6 Interner Takt von Computermusiksystemen

Das im fuzzy:Blockverarbeitung digitaler Signale und Latenz[letzten Abschnitt] vorgestellte Prinzip der Blockverarbeitung wird auch von Computermusiksystemen angewendet. PureData beispielsweise verwendet standardmäßig eine Blockgröße von 64 Samples. Bei vielen Systemen ist die Größe dieser Blöcke sogar einstellbar.

Wenn die Audioverarbeitung von Computermusiksystemen angeschaltet ist, werden automatisch sämtliche erforderlichen Berechnungen in dem Takt durchgeführt, der sich aus Blockgröße und Samplerate ergibt.

Dabei ist es wichtig zu verstehen, dass bei eingeschalteter Audioverarbeitung vom System 'immer' Samples in der eingestellten Samplerate produziert werden müssen, auch wenn kein Klang am Ausgang erwünscht ist. In einem solchen Fall wird eine Folge von Nullen produziert und diese an den Ausgang (und damit den DA-Wandler) weitergegeben.

4.7 Unitgeneratoren und der Audiograph

Jedes Computermusiksystem sollte in der Lage sein, komplexe Signale aus einfacheren Bausteinen zusammenzubauen. Solche Bausteine können elementare Signalgeneratoren, wie beispielsweise Oszillatoren sein, aber auch Filter oder mathematische Operationen, mit denen Signale verändet werden können. Signalgeneratoren und Filter heissen bei vielen dieser Systeme 'Unitgeneratoren' (englisch 'unit generators'). Unitgeneratoren fassen in der Regel bereits mehrere elementare Operationen zusammen. Da sie aber vom System bereitgestellt werden, sind sie in ihrem inneren Aufbau festgelegt und vom Nutzer nicht modifizierbar und damit die kleinsten elementaren Einheiten, die das System zur Verfügung stellt.

Werden mehrere dieser Module miteinander verbunden, so muss sichergestellt werden, dass die Werte in der richtigen Reihenfolge berechnet werden. In Abbildung 4.6 beispielsweise macht es keinen Sinn, die Werte für den Filter zu berechnen, 'bevor' die Ausgangswerte des Oszillators berechnet wurden.

Abbildung 4.6: Verbindung verschiedener Audiomodule

Wenn für einen komplexeren Klang mehrere dieser Unitgeneratoren miteinander verbunden werden, so müssen dabei 'zwei' Dinge vom System eindeutig festgelegt werden:

1. Das System muss darüber informiert werden, in welcher 'Form' die einzelnen Module miteinander verbunden werden, d.h. welche Ausgänge mit welchen Eingängen der verwendeten Module verbunden sind.
2. Das System muss darüber informiert werden, in welcher 'Reihenfolge' diese Module Ihre Werte errechnen müssen, damit das Signal auch entsprechend der beabsichtigten Transformationskette der Signale richtig errechnet wird.

Im Normalfall ergibt sich die Reihenfolge, in der Module ihre Werte errechnen, direkt aus der Form, in der die Module verbunden sind, da es auf der Hand liegt, dass es wenig Sinn macht, die Werte eines nachgeschalteten Moduls zu berechnen, bevor das davorgeschaltete Modul seine Werte berechnet hat. In pd wird die Reihenfolge der Rechnung von pd daher automatisch festgelegt, wenn Module miteinander verbunden werden.

Komplizierter ist dies, wenn zur Laufzeit eines Systems Module dynamisch in den Signalverarbeitungspfad ein- und ausgehängt werden, wie dies beispielsweise charakteristisch für SuperCollider ist. Um hier zu gewährleisten, dass alle Module ihre Werte rechtzeitig im richtigen Moment zur Verfügung stellen, ist es daher in SuperCollider möglich, die Verarbeitungsreihenfolge der Module unabhängig von ihrer Verbindgungsform explizit (über sogenannte 'Groups') festzulegen.

4.8 Oszillatoren, Frequenz und Amplitude

4.8.1 Erzeugung periodischer Wellenformen durch einen Phasor

Ein grundlegendes Verfahren der digitalen Klangsynthese ist die Erzeugung 'periodischer' Prozesse, die kontinuierliche Signale produzieren. Hierzu wird in aller Regel ein Zähler verwendet, der für jedes Sample um einen bestimmten Wert (< 1) inkrementiert wird und, sobald sein Wert > 1 ist, um die Zahl 1 verringert wird. Dadurch entsteht eine Wellenform, die in Abbildung 4.7 dargestellt ist. Solch eine Wellenform wird aufgrund ihres Aussehens als 'Sägezahnwelle' bezeichnet.

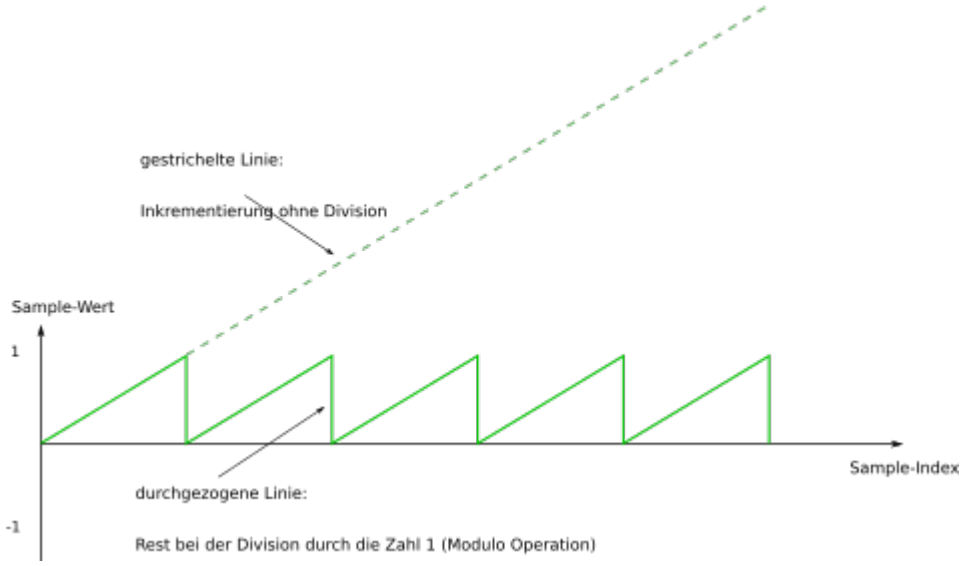


Abbildung 4.7: Sägezahnwelle (Phasor)

Die Verringerung des Zählerwertes um 1 ist mathematisch gleichbedeutend mit der 'Modulo' Operation, die den 'Rest einer Division' angibt, wenn man als Divisor die Zahl 1 verwendet:

1.43 modulo 1 = 0.43

5.7613 modulo 1 = 0.7613

etc...

aber auch:

-3.2 modulo 1 = 0.8 (!)

In manchen Computermusiksystemen, wie pd, wird dieses Verfahren als 'wrapping' bezeichnet, der zugehörige UnitGenerator in pd nennt sich dementsprechend 'wrap\~'.

Da bei dieser grundlegende Wellenform in einer Periode die Zahlen von 0 bis 1 erzeugt werden, entsprechen diese y-Werte der Funktion genau der 'Phase' dieser Wellenform. Aus diesem Grund wird ein UnitGenerator, der diese Form erzeugt, ein 'Phasor' genannt. Das Inkrement zwischen benachbarten Samplewerten ist dabei proportional zur 'Frequenz' dieser Wellenform.

4.8.2 Oszillatoren

Mit Hilfe eines solchen Phasors kann nun relativ einfach eine andere periodische Wellenform, wie beispielsweise ein Sinus generiert werden:

4.8.2.1 Generierung einer Sinusschwingung durch Rechnen

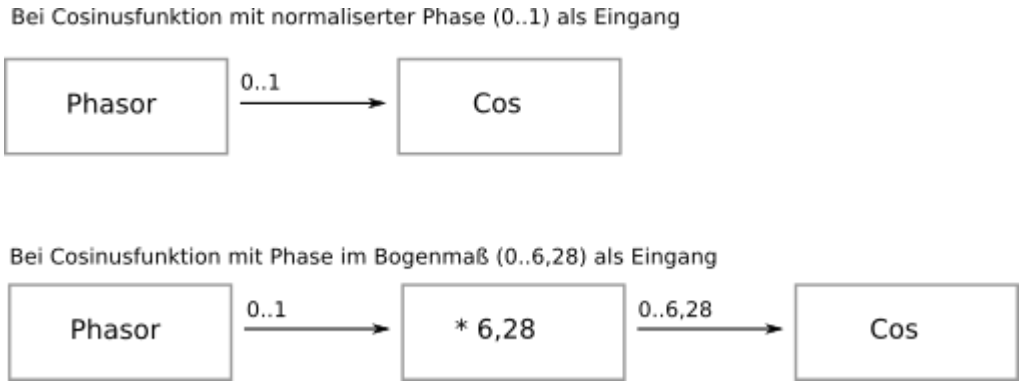


Abbildung 4.8: Generierung einer Siunusschwingung durch Rechnen

<<[[fig:cos-osc01.fig>> zeigt, wie eine Sinusschwingung durch Rechnen erzeugt wird: Der Ausgang des Phasors bildet den Eingang (die x-Werte) einer Cosinus Funktion. Der Ausgang der Cosinus Funktion ist dann bereits die gewünschte Schwingung. Dabei unterscheiden sich vetrschiedene Computermusiksysteme darin, wie der Cosinus berechnet wird: Bei pd ist der Eingang des Cosinus die Phase (Werte von 0..1), die eine volle Periode bedeuten. Bei anderen Systemen wird als Eingangswert der Phasenwinkel im Bogenmaß (0..2*PI) erwartet. In diesem Fall, muss der Ausgang des Phasors zunächst bit 2*PI=6.28 multipliziert werden, bevor die Cosinus Funktion berechnet wird.

4.8.2.2 Generierung einer Sinusschwingung durch Table-Lookup

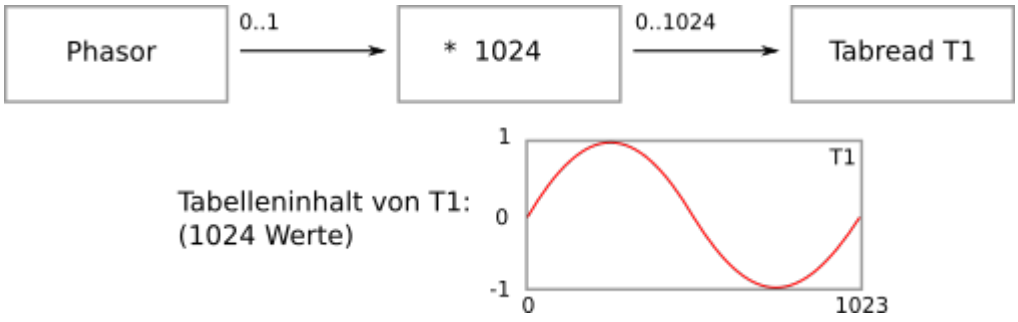


Abbildung 4.9: Generierung einer Siunusschwingung durch Table-Lookup

[?] zeigt eine andere Möglichkeit zur Erzeugung einer Sinusschwingung durch sogenanntes 'Table-Lookup'. In diesem Fall wird eine Tabelle (englisch 'table', in pd auch 'array' genannt) verwendet, die die Ergebnisse der Cosinusfunktion einer einzigen gesamten Periode enthält. Um eine möglichst feine Auflösung der Funktion zu erhalten, empfiehlt es sich, eine möglichst große Tabelle (beispielsweise mit 1024 Werten) zu verwenden.

Ein Oszillator wird nun dadurch erzeugt, dass man den Ausgang des Phasors mit der Größe der Tabelle multipliziert und dann den Tabellenwert ausliest, der diesem errechneten Index entspricht.

Da der errechnete Wert in den meisten Fällen nicht ganzzahlig ist und daher nicht genau auf einem Tabellenwert liegt, gibt es verschiedene Möglichkeiten, wie man vorgehen kann:

1. Rundung

Es wird der nächstliegende Tabellenwert ausgelesen (in pd wird dieses Verfahren von ”tabread\~” verwendet).

2. Lineare Interpolation

   Es wird eine Linie durch die benachbarten Tabellenwerte gezogen, zwischen denen der errechnete Indexwert liegt und der y-Wert auf dieser Linie verwendet, der an der Position des errechneten Indexwertes liegt.

3. Polynomiale Interpolation

   Es werden mehrere (meist 4) benachbarte Werte aus der Tabelle um den errechneten Indexwert verwendet und mit Hilfe eines speziellen Algorithmus wird eine möglichst ausgeglichene Kurve durch diese Werte gelegt. Der Kurvenwert an der Stelle des errechneten Wertes wird dann verwendet (in pd wird dies mit ”tabread4\~” realisiert).

Qualitativ ist das erste Verfahren am schlechtesten und das letzte Verfahren am besten, allerdings ist das erste Verfahren auch das schnellste Verfahren, während das letzte Verfahren rechenintensiver ist. Man kann die Nachteile des ersten Verfahrens allerdings auch durch ausreichend groß gewählte Tabellen kompensieren.

Das Verfahren durch Table-Lookup hat den Vorteil, dass man damit ’beliebige’ periodische Wellenformen erzeugen kann. Nachteil ist unter Umständen eine geringere Flexibilität bei dynamischer Modulation des Klangs, da es schwer ist, die Wellenform zu verändern, während der Oszillator Töne produziert.

4.8.3 Amplitude

Die Veränderung der Amplitude (= Lautstärke) eines Oszillators, oder ganz allgemein von Audiosignalen lässt sich sehr einfach durch Multiplikation des Ausgangssignals erreichen (Abbildung 4.10).

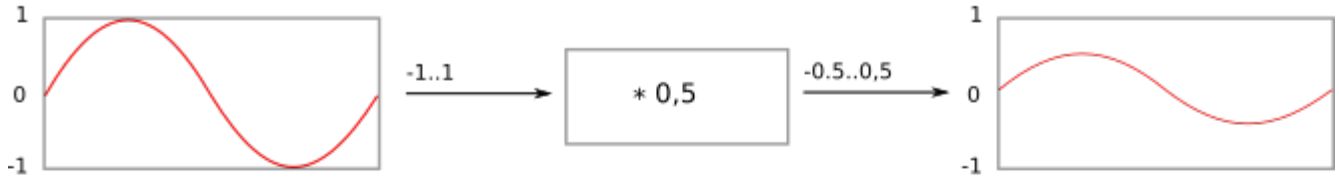


Abbildung 4.10: Amplitudenveränderung eines Audiosignals

4.8.4 Frequenz

Die Frequenz eines Oszillators kann auf verschiedene Weisen verändert werden:

1. Die Samplerate wird verändert

2. Das Inkrement des Sample-Indexes wird verändert.

   Im Falle eines Oszillatoraufbaus aus Phasor und Cosinus Funktion bzw. Table-Lookup bedeutet dies, dass das Phaseninkrement zwischen benachbarten Samplewerten im Phasor UnitGenerator verändert wird (Abbildung 4.11).

   Alternativ könnte auch der Ausgang des Phasors skaliert werden. Dies funktioniert jedoch nur bei ganzzahligen Faktoren, da ansonsten das Zurückspringen des Phasors auf 0 nicht synchron mit der Phase des Ausgangssignals nach der Multiplikation ist und dadurch Phasensprünge im Ausgangssignal in der Frequenz des Phasors entstehen (Abbildung 4.12 und Abbildung 4.13).

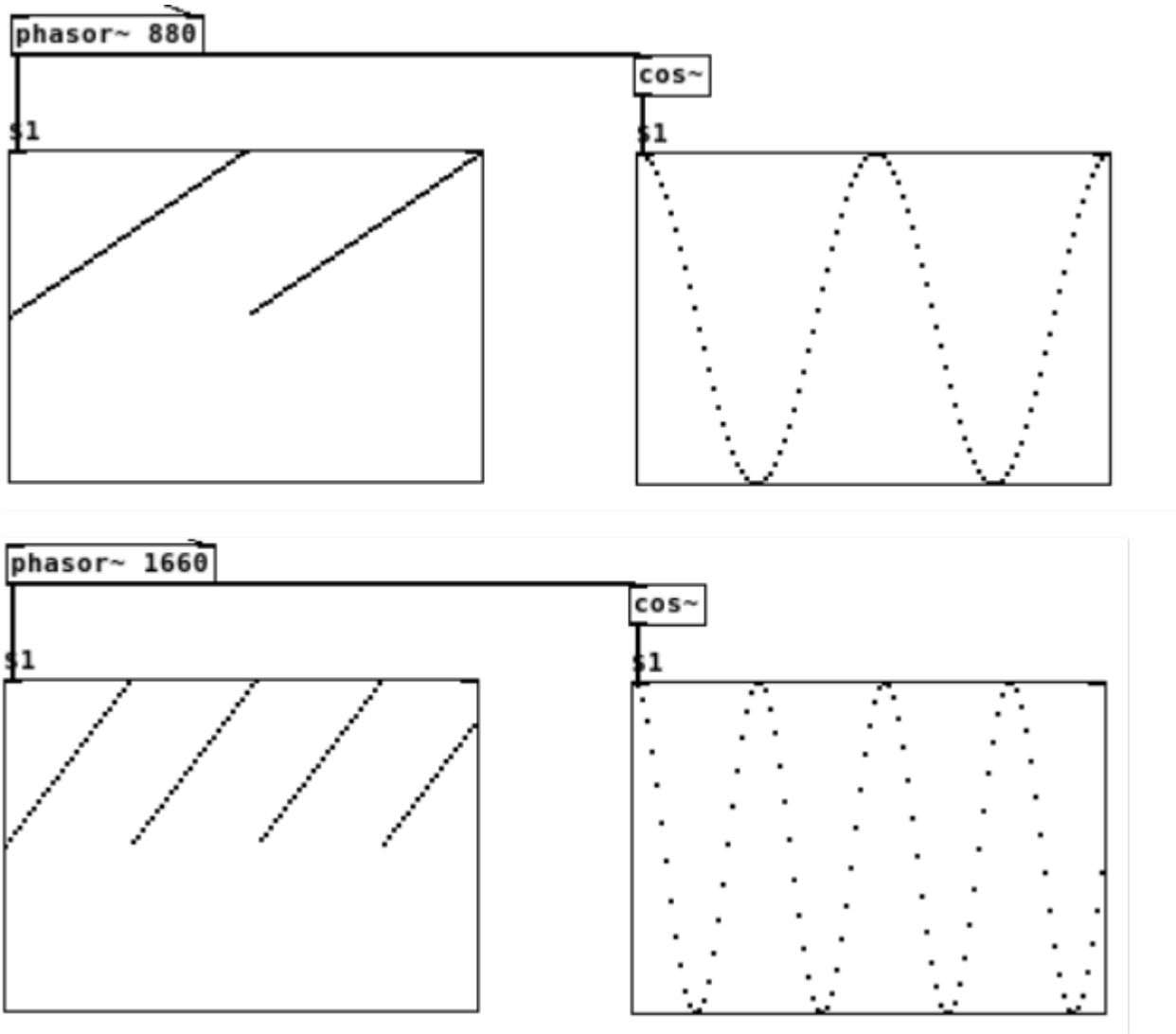


Abbildung 4.11: Frequenzveränderung durch Inkrementveränderung eines Phasors

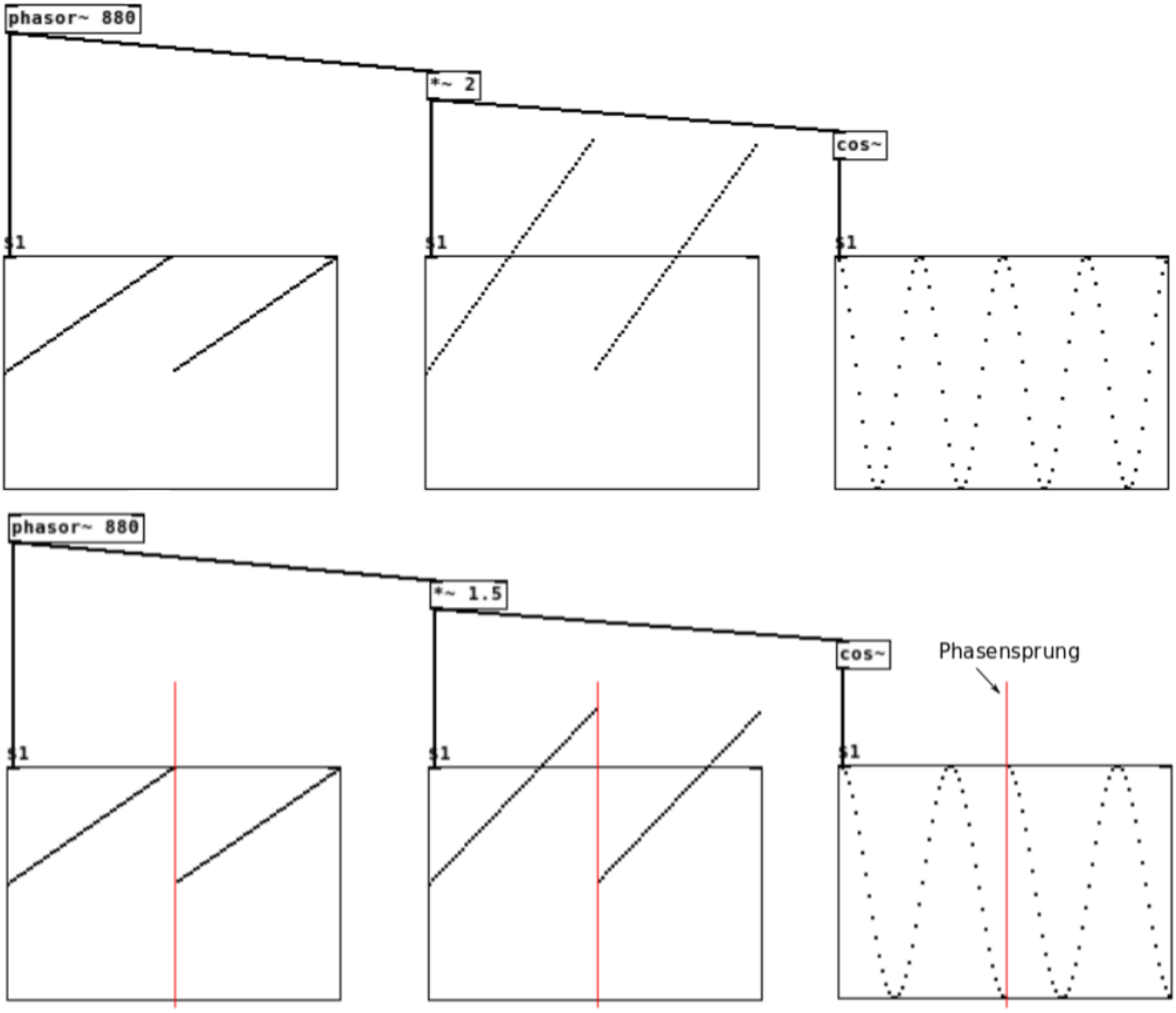


Abbildung 4.12: Frequenzveränderung durch ganzzahlige/nichtganzzahlige Multiplikation eines Phasors

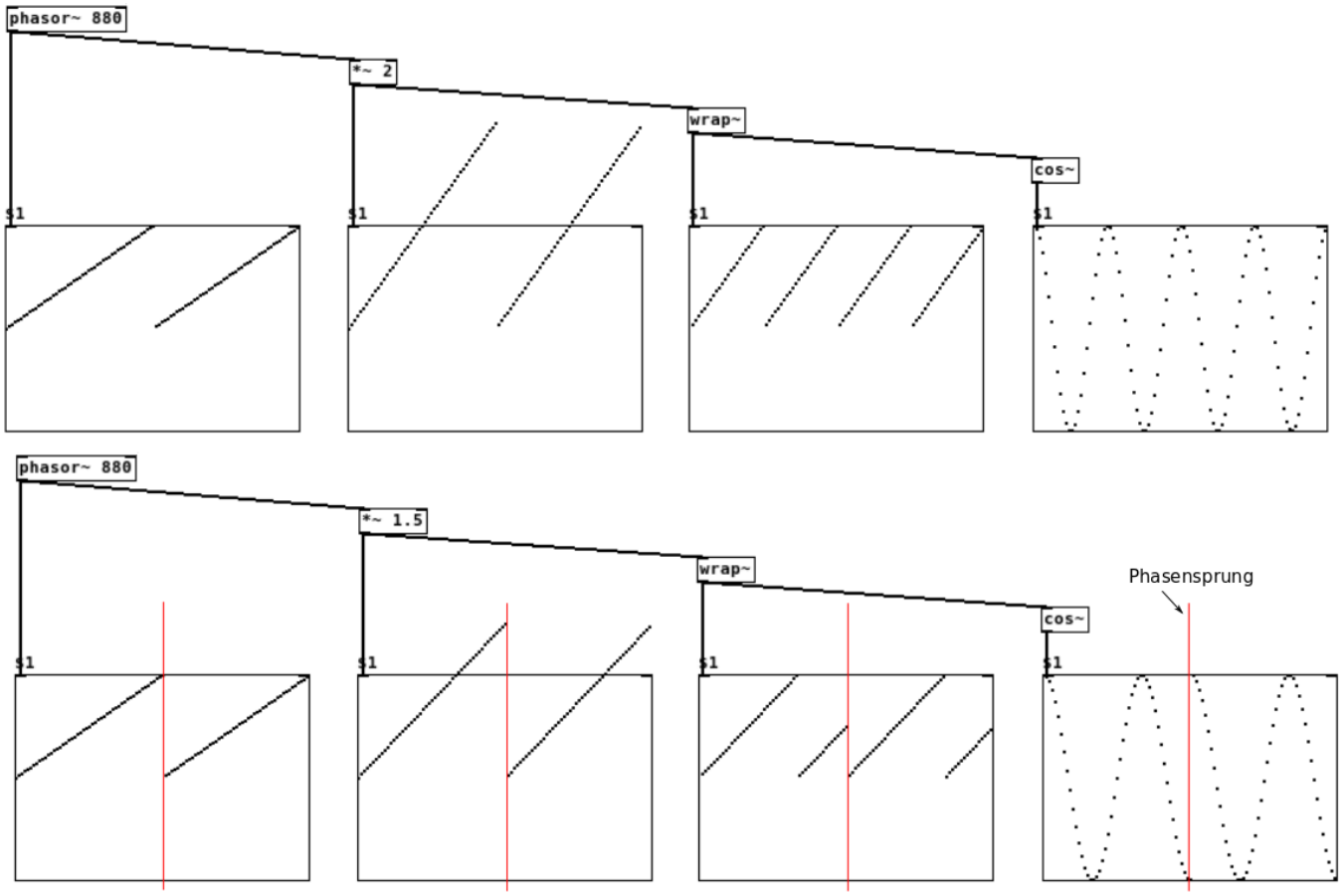


Abbildung 4.13: Frequenzveränderung durch ganzzahlige/nichtganzzahlige Multiplikation eines Phasors und Normalisierung nach der Multiplikation

Da in den allermeisten Computermusiksystemen die Samplerate konstant ist, wird heute praktisch ausschließlich das zweite Verfahren angewendet.

4.9 Elementare Wellenformen

In Analogsynthesizern wurden als Oszillatoren verschiedene partialtonreiche Wellenformen verwendet, die sich technisch leicht erzeugen lassen und deren Spektrum sich dann durch nachgeschaltete Filter verändern liess. In der digitalen Signalverarbeitung spielen diese Grundformen als direkt verwendete Audiosignale aufgrund anderer Syntheseverfahren im digitalen Bereich eine untergeordnete Rolle. Daher existieren sie nicht in allen Computermusiksystemen als UnitGeneratoren. Dennoch sind sie unter Umständen, insbesondere bei niederfrequenten Steuerungen sinnvoll einsetzbar. An dieser Stelle wird daher der Vollständigkeit halber kurz erläutert, wie man diese Formen mit Hilfe elementarer Rechenoperationen aus einem Phasor erzeugen kann.

4.9.1 Sägezahn

Der Sägezahn (englisch ’sawtooth’) ist mit dem Phasorsignal vergleichbar. Wenn man es als Audiosignal mit Amplitude 1 verwenden möchte, ist jedoch der Wertebereich (0..1) ungünstig. Es empfiehlt sich die aus Abb. hervorgehende Skalierung und Verschiebung.

4.9.2 Dreieck

Die Erzeugung eines Dreieckssignals geht aus Abb. hervor. Im Normalfall sind beide Flanken des Dreiecks gleich lang, jedoch gab es auch Analogsynthesizer, bei denen die Flankensteilheit bis zum Sägezahn verändert werden konnte, daher wurde in dem Beispiel eine Modulationsmöglichkeit für die Flankenbreite implementiert.

4.9.3 Rechteck

Die Erzeugung eines Rechteckssignals geht aus Abb. hervor. Auch hier ist eine Modulationsmöglichkeit der Pulsbreite des Rechtecks implementiert, die auch bei vielen Analogsynthesizern möglich war.

Kapitel 5

Synthesemodelle

5.1 Waveshaping

Waveshaping ist ein Verfahren, bei dem ein Ausgangssignal mit Hilfe einer ’Transferfunktion’ verändert wird.

Abbildung 5.1: Prinzip von Waveshaping

Die in Abbildung 5.2 verwendete Transferfunktion lässt das Eingangssignal unverändert, wenn die Werte in einem Bereich von [-0,8..0,8] liegen. Erst wenn Werte ←0,8 oder >0,8 vorkommen, wird der Ausgangswert auf den Wert -0,8 bzw. 0,8 gesetzt. Mit anderen Worten ist die Form des Ausgangssignals abhängig von der Amplitude des Eingangssignals: Hätte die Sinusfunktion am Eingang eine Amplitude <0.8, so wäre das Ausgangssignal identisch mit dem Eingangssignal. erst bei Amplituden >0.8 wird das Eingangssignal verändert.

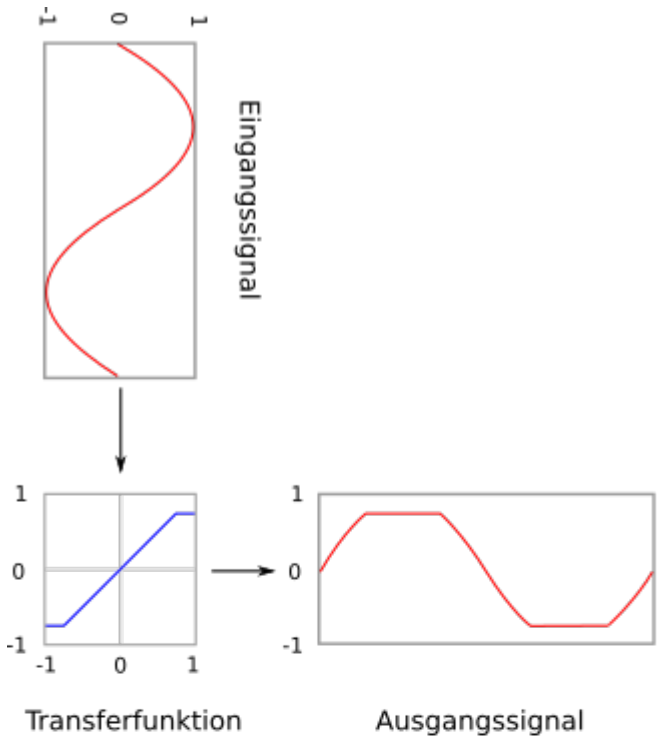


Abbildung 5.2: Beispiel von Waveshaping (Verzerrung durch Clipping)

Wenn für die Transformation eines Eingangssignals in ein Ausgangssignal gilt, dass die Transformation eines skalierten Eingangssignals identisch mit der Skalierung der Transformation des unskalierten Eingangssignals mit dem gleichen Faktor ist, so spricht man von einer 'linearen Transformation'. Mathematisch muss also für alle Zahlen a und die Transformation T folgende Identität gelten:

a * T(x) = T(a * x)

Im Falle von Waveshaping trifft dies nur für Transferfunktionen der Form T=a*x zu. Diese Transferfunktion ist allerdings nicht interessant, da sie lediglich das Eingangssignal mit dem Faktor a skaliert. Im Normalfall ist Waveshaping daher eine 'nichtlineare Transformation'. Diese Eigenschaft macht das Waveshaping mathematisch oft sehr anspruchsvoll und es ist nicht einfach, die Konsequenzen bestimmter Transferfunktionen genau einzuschätzen. Andererseits ist nichtlineares Verhalten in der Akustik von Musikinstrumenten nicht die Ausnahme, sondern die Regel und macht einen großen Teil ihrer klanglichen Faszination aus.

Es ist wichtig, dass sich Waveshaping nicht nur auf Sinusfunktionen, sondern auf jegliches Eingangssignal anwenden lässt. Die Eigenschaften bezüglich der Partialtöne der dargestellten Sinusfunktionen wirken sich bei komplexeren Eingangssignalen auf 'alle' Frequenzanteile des Eingangssignals aus, so dass sich harmonische Eingangssignale unabhängig von der Frequenz immer auch harmonisch verhalten und sich lediglich die Amplituden der Partialtöne verändern bzw. neue Partialtöne hinzutreten. Im Zusammenhang mit der Amplitudenabhängigkeit der Transformation lässt sich dadurch mit sehr wenig Aufwand ein sehr variabler Klang erzielen, der sich auch hervorragend im Bereich der Live-Elektronik einsetzen lässt.

Nachfolgend sind verschiedene typische Waveshaping Verfahren dargestellt, die sich recht gut einschätzen lassen und daher sehr beliebt sind.

5.1.1 Verzerrer

Abbildung 5.3 zeigt den Einsatz von Waveshaping zur Simulation eines Verzerrers (englisch 'distortion'), wie er beispielsweise als charakteristische Klangtransformation für E-Gitarren eingesetzt wird. Ursprünglich wurde dieser Effekt durch das Übersteuern analoger Eingangsverstärkerstufen erreicht. Das hier dargestellte Waveshaping Verfahren ist eine digitale Simulation des Verhaltens solcher analogen Verstärker.

Die in der ersten Zeile der Abildung dargestellte Form der Verzerrung liesse sich in pd beispielsweise ganz einfach durch den Unitgenerator [clip\~ -0.5 0.5] erreichen.

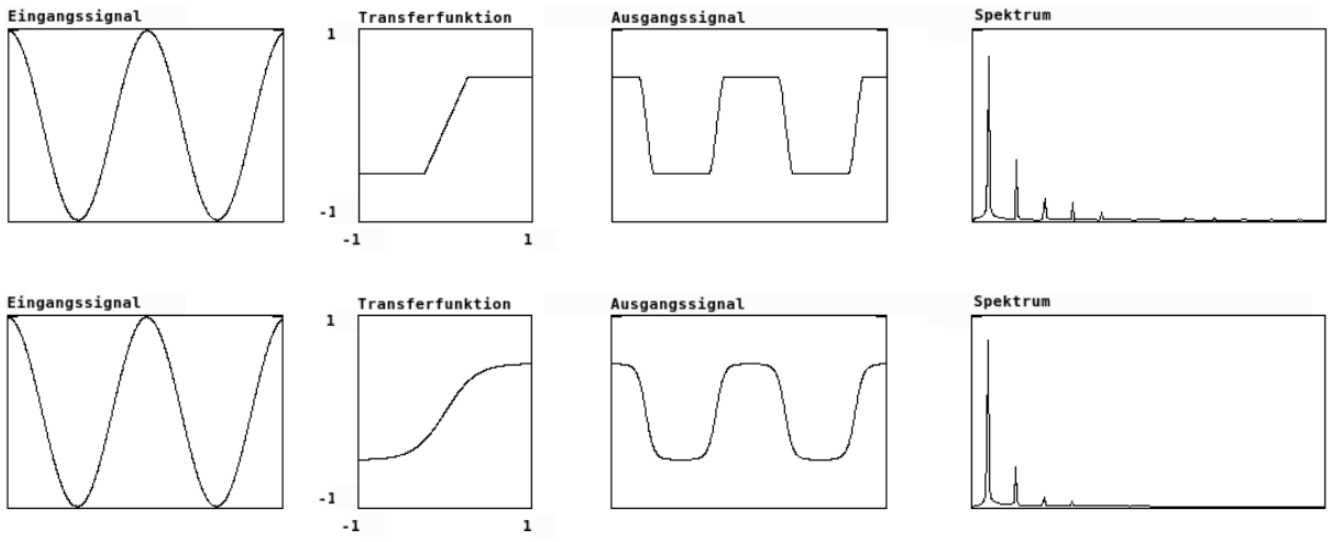


Abbildung 5.3: Simulation eines analogen Verzerrers mit Waveshaping

Durch die scharfe Kante der Transferfunktion an der Stelle, an der das Signal begrenzt wird, entsteht im Ausgangssignal auch eine scharfe Kante, wenn das Eingangssignal eine entsprechende Mindestamplitude hat. Wie in einem späteren Kapitel im Zusammenhang mit der Fourier-Transformation noch näher erläutert wird, führen scharfe Kanten in einem Audiosignal zu sehr vielen Obertönen. Im digitalen Bereich ist dies zumeist aufgrund des dadurch kaum vermeidbaren 'Aliasing' bei hohen Eingangsfrequenzen unerwünscht.

Auch in der analogen Verstärkertechnik haben Transistoren, die zur Verstärkung verwendet werden, bei Übersteuerung ein ähnliches Clippingverhalten (Die Transferfunktion nennt man bei Transistoren die 'Kennlinie'). Die historisch älteren Röhrenverstärker verwenden Röhren anstatt Transistoren, die eine sehr viel weichere Kennlinie aufweisen. Das wird aufgrund der weniger promineten hohen Partialtöne zumeist als klanglich 'wärmer' und angenehmer empfunden.

Um diesen Effekt zu simulieren, wird daher auch im digitalen Bereich für Verzerrereffekte eine weichere Kennlinie, wie man sie beispielsweise mit der 'Tangens Hyperbolicus' Funktion erreicht, verwendet. Die untere Zeile von Abbildung 5.3 zeigt die daraus resultierende Veränderung des Ausgangssignals gegenüber dem ersten Beispiel. Es ist deutlich zu sehen, dass das Spektrum dieses Signals im Vergleich zum Spektrum des oberen Signals weniger ausgeprägte Obertöne der Grundschwingung enthält. Man spricht im ersten Beispiel auch von 'hard clipping', im zweiten Beispiel von 'soft clipping'.

5.1.2 Sinusförmige Transferfunktionen

Auch Cosinusfunktionen eignen sich als Transferfunktionen für Sinusschwingungen. Die Frequenz der Cosinusschwingung hat dabei einen Einfluss auf die maximale Höhe zusätzlich entstehender Partialtöne. In der ersten Zeile der Abbildung 5.4 kann man der Darstellung des Spektrums rechts entnehmen, dass eine Cosinustransferfunktion mit 'einer' Periode im x-Wertebereich von [-1..1] zusätzlich den 2. Partialton erzeugt, während bei der Modulation in der zweiten Zeile mit einer Cosinustransferfunktion doppelter Frequenz der 3. Partialton prominent vertreten ist.

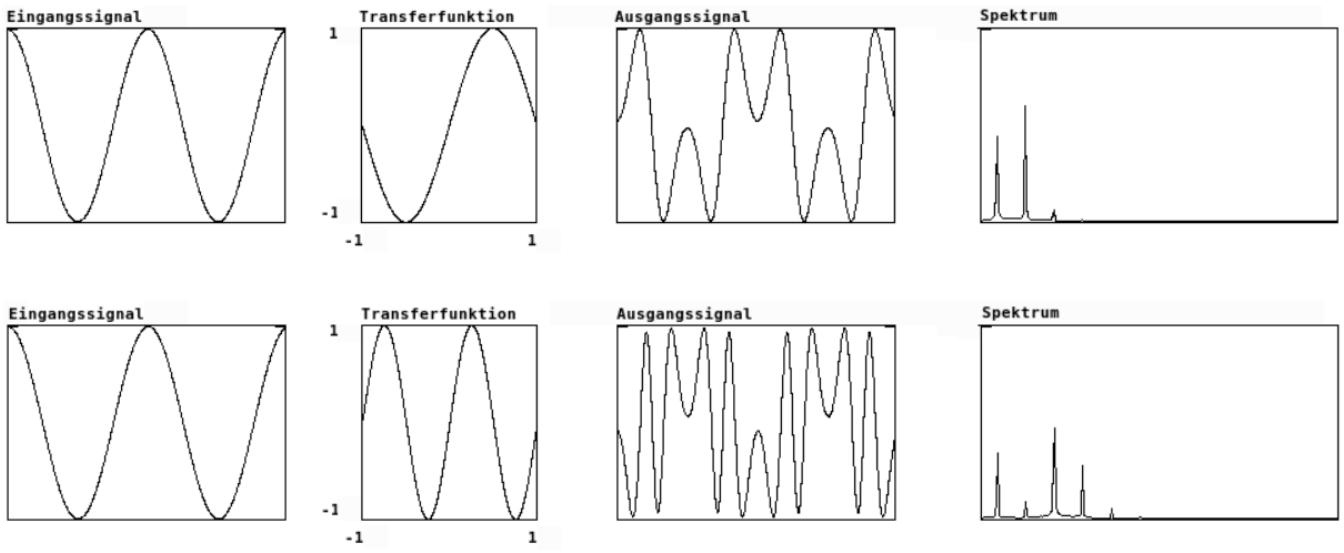


Abbildung 5.4: Waveshaping mit Cosinusfunktionen

Wenn der x-Wertebereich erweitert wird, kann man sich diese Eigenschaft zu Nutze machen, um reiche Transformationen und Erweiterungen des Partialtonspektrums zu erzielen. Abbildung 5.5 zeigt eine Transferfunktion mit erweitertem Wertebereich von -8 bis 8.

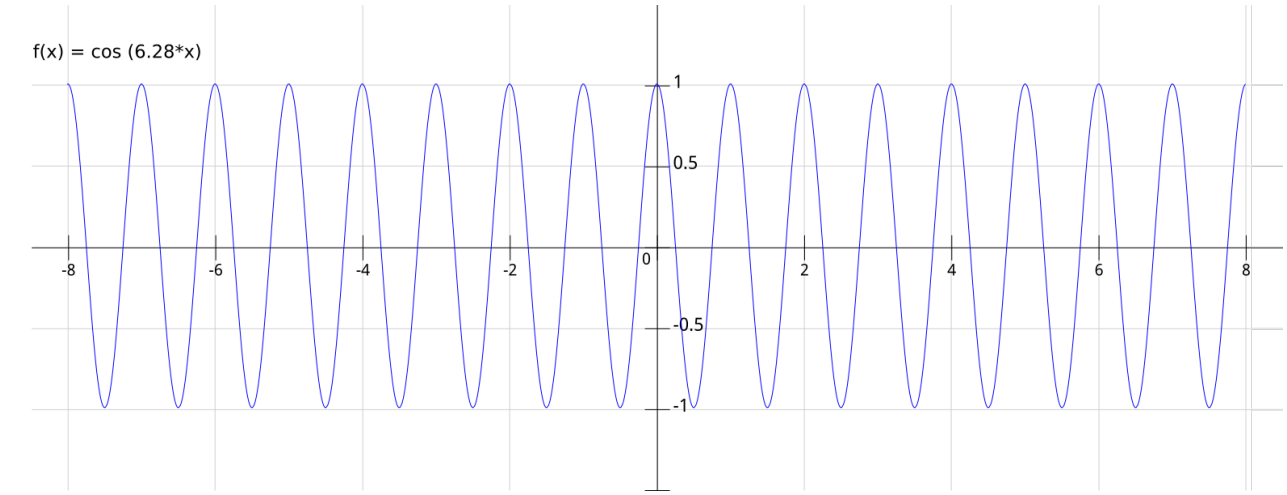


Abbildung 5.5: Cosinus als Transferfunktion über erweiterten Wertebereich

Dadurch ist es möglich, die Sinusschwingung am Eingang der Transferfunktion bis zur maximalen Amplitude von 8 verstärken und durch die Variation der Verstärkung vielfältige spektrale Veränderungen herbeizuführen. Es ist dabei sinnvoll, die Verstärkung durch Division mit dem Verstärkungsfaktor nach der Transformation wieder auszugleichen, um Übersteuerungen zu vermeiden und lediglich klangliche Veränderungen ohne große Lautstärkeunterschiede herbeizuführen. Es ist dabei allerdings zu beachten, dass große Verstäkungsfaktoren bei hohen Grundfrequenzen zu einem deutlichen Aliasing Effekt führen. In Abbildung 5.6 schließlich wird die Cosinustransferfunktion in positive und negative x-Richtung gestaucht. Hier entstehen bei Variation des Verstärkungsfaktors deutlich wahrnehmbare und reizvolle Interferenzen der Spektralveränderung.

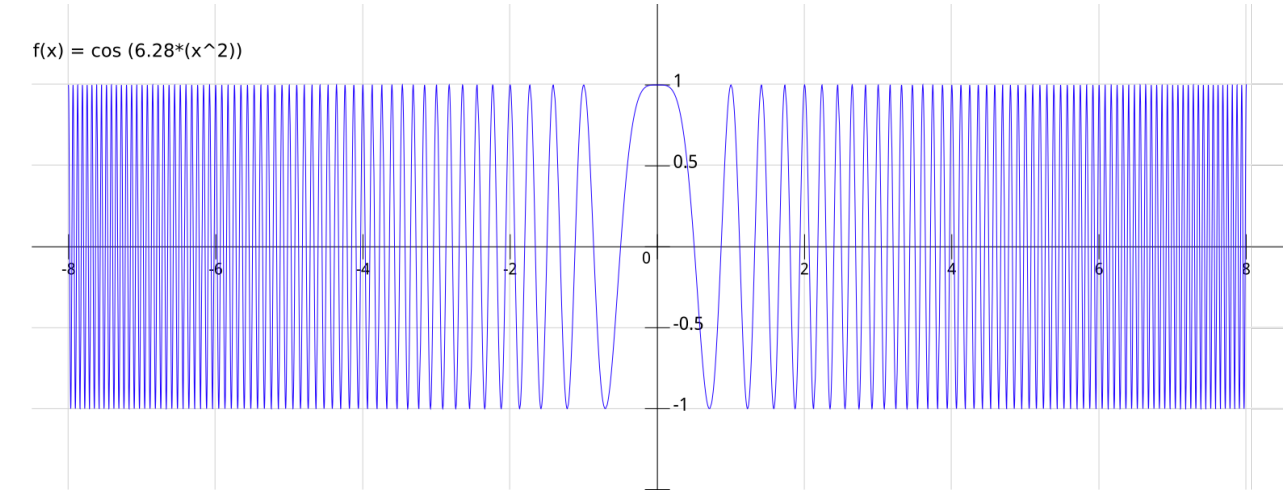


Abbildung 5.6: in x Richtung gestauchter Cosinus als Transferfunktion über erweiterten Wertebereich

5.1.3 Chebyshev Polynome

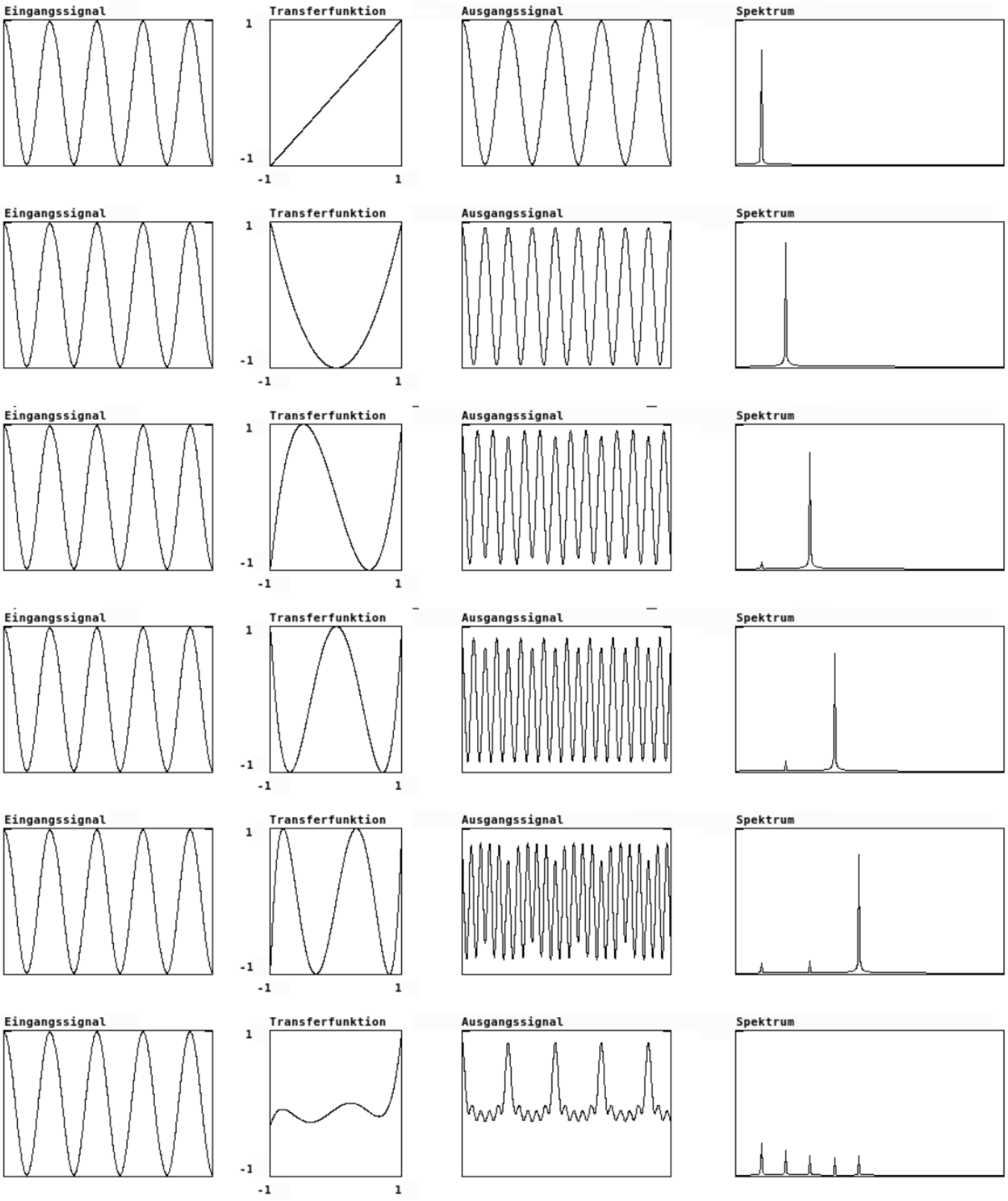


Abbildung 5.7: Waveshaping mit Chebyshev Polynomen

Chebyshev Polynome erster Art sind zum Koordinatenursprung punktsymmetrische bzw. zur y-Aches symmetrische Polynome, die vom Aussehen im Wertebereich [-1..1] trigonometrischen Funktionen ähneln. Dabei ist die Frequenz dieser ”Sinusschwingungen” proportional zur Ordnung der Polynome (die 3. Ordnung entspricht dabei einer Sinusschwingung mit einer Periode im Wertebereich [-1..1]. Verwendet man diese Polynome als Transferfunktion, so wird bei hoher Eingangsamplitude der Partialton oberhalb des Eingangscosinussignals hervorgehoben, der der Ordnung des Chebyshov Polynoms entspricht. In fuzzy:fig:waveshaping01[fuzzy:fig:waveshaping01] sind in den ersten fünf Zeilen die Chebyshev Polynome 1.-5. Ordnung als Transferfunktion verwendet worden (die Funktionen sind in der 2. Spalte (Transferfunktion) deutlich zu erkennen). Ein Vergleich der einzelnen Spektraldarstellungen zeigt deutlich, wie der Partialton, der dem Index der Chebyshev Funktion entspricht, hervortritt.

Die unterste Zeile zeigt das Ergebnis des Waveshaping mit einer Überlagerung der ersten fünf Chebyshev Polynome mit den relativen Amplituden 1, 0.685, 0.566, 0.456 und 0.535 als Transferfunktion.

Für eine andere künstlerische Verwendung von Waveshaping im Zusammenhang mit Chebyshev Polynomen siehe hier: http://cmc.music.columbia.edu/musicandcomputers/chapter4/04_06.php

5.1.4 Bandbegrenzter Puls mit Hilfe der Exponentialfunktion

Miller Puckette, der Autor von Pure Data hat in seinem Buch zur digitalen Klangsynthese hier eine andere Anwendung von Waveshaping zur Generierung bandbegrenzter Pulsfolgen dargestellt. Dafür verwendet man eine Exponentialfunktion, die an der y-Achse gespiegelt wird, im x-Wertebereich [0..1].

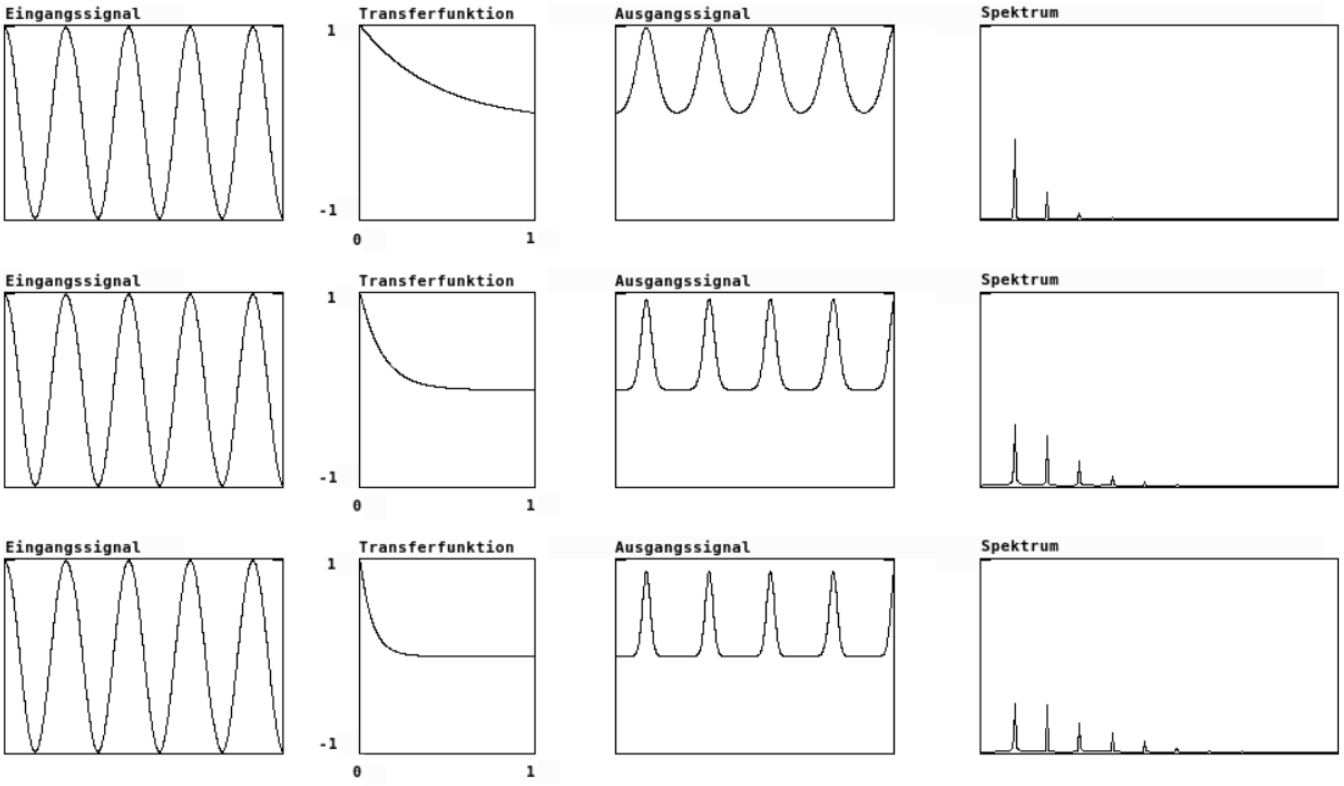


Abbildung 5.8: Erzeugung eines bandbegrenzten Pulses durch Waveshaping mit der Exponentialfunktion

Wie aus Abbildung 5.8 hervorgeht, kann durch Skalierung der Funktion in x-Richtung der Anfangswinkel der Funktion flacher oder steiler gestellt werden. Bei Verwendung dieser Funktion als Transferfunktion und Addition von 1 zum Cosinuseingangssignal erhält man am Ausgang der Transferfunktion

eine Sinusförmige Schwingung, deren Breite abhängig von der x-Skalierung (und damit dem Verstärkungsfaktor des Signals am Eingang der Transferfunktion) variiert werden kann. Es entsteht eine Pulsfolge mit -vom Verstärkungsfaktor abhängigen- variablen Anzahl von Obertönen. Im Unterschied zu den anderen dargestellten Waveshaping Verfahren ist dabei die Form der Modulation sehr gleichmäßig und gleicht von der Wirkung dem Verändern der Grenzfrequenz eines Tiefpassfilters.

Die gleichen Effekt kann man allerdings noch besser kontrollierbar mit einem anderen, direkten Verfahren erreichen, das Miller Puckette diesem Kapitel seines Lehrbuchs erklärt.

Vertiefung: Für eine Einführung in die Erzeugung bandbegrenzter Rechteck- und Sägezahnsignale, siehe: http://csoundjournal.com/issue11/distortionSynthesis.html

5.2 Amplitudenmodulation (Ringmodulation)

Unter Amplitudenmodulation versteht man die Veränderung der Amplitude/Lautstärke in der Zeit. Insofern ist Lautstärkungveränderung mit einem Regler bereits eine Form der Amplitudenmodulation. Wenn die modulierende Funktion eine periodische Schwingung (beispielsweise eine Dreiecks- oder Sinusschwingung) mit niedriger Frequenz ist, so erhält man den tremoloähnlichen Effekt einer Lautstärkeschwankung. Wird die Frequenz der periodischen Schwingung erhöht, entsteht eine starke Klangfarbenveränderung: Bei gleichbleibender Tonhöhe des nichtmodulierten Signals (dem 'Trägersignal') und kontinuierlicher Erhöhung der Frequenz des modulierenden Signals (dem 'Modulationssignal'), werden mit der Frequenzveränderung des Modulationssignals synchrone Auf- und Abwärtsglissandi wahrnehmbar.

Erklären lässt sich diese Phänomen mit folgender trigonometrischen Identität:

cos(x)*cos(y)= 1/2(cos(x-y) + cos(x+y))

Diese Gleichung bedeutet, dass das Produkt zweier Cosinusschwingungen unterscheidlicher Frequenz (x und y) identisch ist mit der Addition einer Cosinusschwingung der Summe beider Frequenzen mit einer Cosinusschwinugne der Differenz beider Frequenzen mit jerweils halber Amplitude.

Bemerkenswert hierbei sind zwei Dinge:

1. Die Frequenzanteile des Trägersignals (und des Modulationssignals) verschwinden durch die Multiplikation
2. Dies bezieht sich bei komplexeren (also Signalen mit Partialtönen) auf sämtliche Partialtöne.

Da die Verschiebung des Signales um eine konstante Frequenz erfolgt, führt dies bei komplexen Signalen zu einer Verzerrung des Partialtonspektrums, da die Frequenzverschiebung bei niedrigen Frequenzen des Trägersignals größere Intervalle ergibt, als bei höheren Frequenzen. Auf diese Weise lassen sich mit Hilfe dieser Modulation sehr einfach inharmonische Partialtonspektren erzielen.

Diese Form der Modulation wird 'Ringmodulation' genannt. Sie ist technisch sehr einfach zu erzeugen, da Träger und Modulationssignal lediglich multipliziert werden, haben aber einen sehr großen Effekt. Eine Komposition, die diese Modzulationsart zentral verwendet, ist die Komposition Mantra für zwei Klaviere und Live-Elektronik von Karlheinz Stockhausen aus dem Jahr 1970.

Der Vollständigkeit halber sollte an dieser Stelle erwähnt werden, dass die Ringmodulation eine 'symmetrische' Transformation ist, da bei einer Multiplikation die Reihenfolge der Faktoren keine Rolle spielt. Insofern sind beide an der Ringmodulation beteiligten Signale (Trägersignal und Modulationssignal) vertauschbar, ohne dass sich das klangliche Ergebnis verändert.

5.3 FM-Synthese

5.4 Aufgaben

5.4.1 Grundlagen der digitalen Signalverarbeitung

- Wie groß ist der Zeitabstand einzelner Samples bei einer Samplerate von 48000 Samples/Sekunde (mit Rechenweg)?
- Wie groß ist die Abtastrate aufeinanderfolgender Sampleblöcke bei einer Blockgröße von 1024 Samples und einer Samplerate von 48000 Hz?
- Welcher Zeitabstand zwischen diesen Sampleblöcken ergibt sich daraus?
- Welche maximale Frequenz ist digital noch darstellbar bei einer Samplerate von 48000 Hz?
- Wie nennt man diese Frequenz?
- Ein Signal mit einer Frequenz von 30000 Hz wird mit einer Abtastrate von 44100 Hz abgetastet. Welche Frequenz ergibt sich nach der Abtastung aufgrund des 'Aliasing' Effektes?
- In einem Computermusiksystem sind ein Phasengenerator mit einer Cosinusfunktion und einer nachgeschalteten Multiplikation -wie in Abbildung 5.9 dargestellt- verbunden. Der Phasor hat dabei eine Frequenz von 10 Samples/Periode, die Blockgröße des Systems ist 8 Samples. Bitte errechnen Sie sämtliche Werte der drei Blöcke in der zweiten Zeile der Abildung für die ersten 32 Samples.

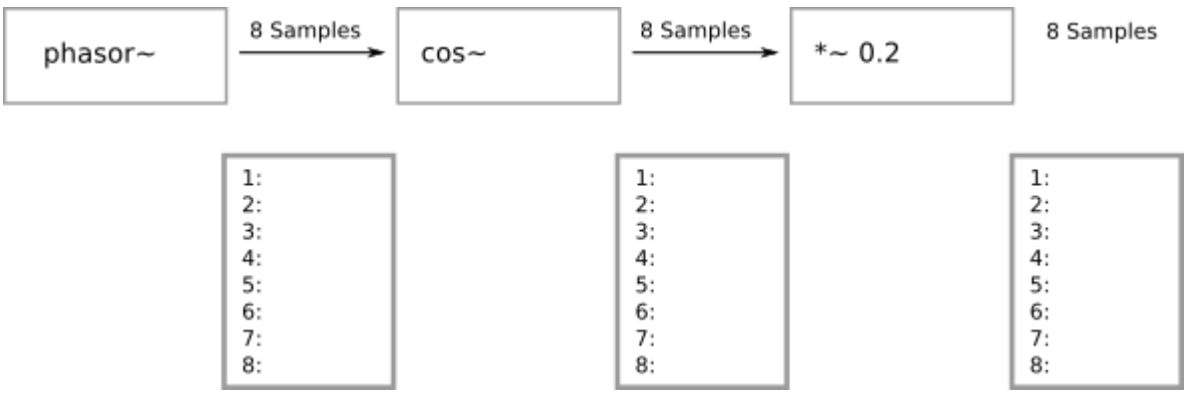


Abbildung 5.9: Übung: Errechnung von Audiowerten

- Bitte bauen Sie einen PD-Patch eines Sinusoszillators von 880 Hertz mit einem nicht knacksenden Lautstärkeregler (Wertebereich 0..1) auf drei verschiedene Weisen:
  a. Mit Hilfe eines Oszillators
  b. Mit Hilfe eines Phasors und einer sin/cos Funktion
  c. Mit Hilfe von Tablelookup (Tablegröße 512 Werte)
  Veranschaulichen Sie den Prozess durch grafische Arrays für jeden Verarbeitungsschritt.
- Bitte beschreiben Sie zwei Verfahren, mit Hilfe eines Phasors eine Wellenform zu generieren, die aus der Grundschwingung (Phasorfrequenz) und dem zweiten Partialton mit den relativen Amplituden 1 und 0.7 besteht.

  Wählen Sie als erstes Verfahren die Generierung durch Rechnen und als zweites Verfahren die Generierung durch Table-Lookup.

  Zeichnen Sie für die Lösung die Blockdiagramme, die die einzelnen Unitgeneratoren und ihre Verbindungen enthalten. Im Falle des Verwendens von Tabellen zeichnen Sie die Tabellen und ihren Inhalt.

5.4.2 Synthesemodelle

- Bitte entwerfen Sie ein Blockdiagramm mit Unitgeneratoren für die beiden am Ende von Abschnitt 5.1.2 erläuterte Waveshaping mit erweitertem Wertebereich. Verwenden Sie für die Realisation 'keinen' Table-Lokkup, sondern ermitteln Sie die Ausgangswerte der Transferfunktion durch Errechnung mit mathematischen Operatoren.
- Realisieren Sie dieses Blockdiagramm mit einem Computermusiksystem Ihrer Wahl.
- Die dargestellten Transferfunktionen haben beim x-Wert 0 des Koordinatensystems den y-Wert 1. Dies führt dazu, dass bei sehr geringer Amplitude des Eingangssignals eine hoher Gleichspannungsanteil am Ausgang der Transferfunktion ausgegeben wird. Wenn man anstelle der Cosinusfunktion eine Sinusfunktion verwendet, wird dieses Problem vermieden. Wie müsste das oben entworfene Blockdiagramm verändert werden, damit die Transferfunktion sinusförmig durch den Ursprung (0,0) geht?

Kapitel 6

Abstraktionen und Steuerung

6.1 Hüllkurven

Ein Hüllkurvengenerator (englisch ’envelope generator’) dient der Modellierung des Lautstärkeverlaufs eines Klangs. Es handelt sich also um einen Generator von Zahlenwerten im Wertebereich [0..1], dessen Zeitverlauf flexibel eingestellt werden kann. In vielen Computermusiksystemen gibt es bereits Unitgeneratoren, die solche Hüllkurven erzeugen. Man entscheidet dabei zwischen folgenden Typen:

1. Abschnittsweise definierte Hüllkurvengeneratoren. Diese Generatoren können im Normalfall beliebig viele Segmente mit variabler Dauer und variablen Zielwerten enthalten. Dabei können die Punkte der Segmente durch ’lineare’ Abbildung 6.1 oder ’exponentielle’ Abbildung 6.2 Werteverläufe verbunden werden. Die Dauernwerte zwischen benachbarten Segmenten können dabei als absolute Zeitwerte definiert werden, oder proportional bezogen auf eine Gesamtdauer angegeben werden. Es ist wichtig, zu beachten, dass die y-Werte der Segmente von exponentiellen Hüllkurven positiv sein müssen, und ’nicht’ auf den Wert 0 enthalten dürfen.

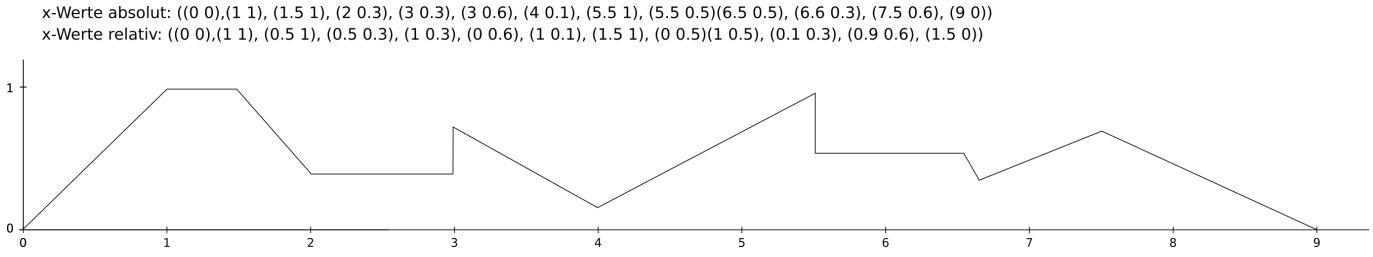


Abbildung 6.1: Abschnittsweise definierte Hüllkurve mit linearen Segmenten

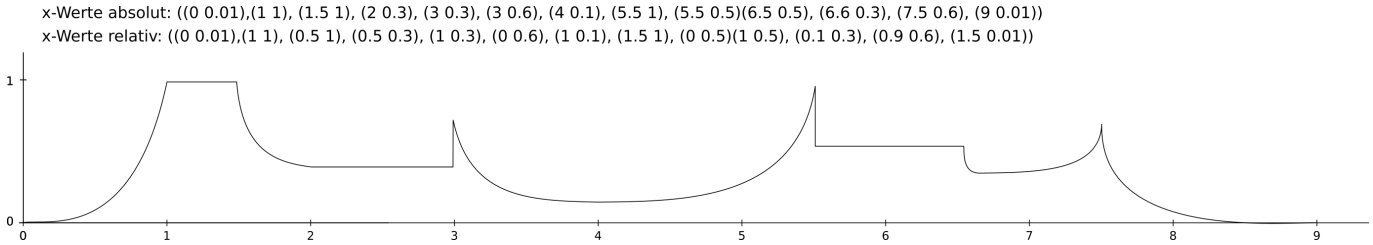


Abbildung 6.2: Abschnittsweise definierte Hüllkurve mit exponentiellen Segmenten

2. Hüllkurvengeneratoren mit einer vordefinierten Anzahl und Funktion von Segmenten. Hierzu zählen ’ADSR Hüllkurvengeneratoren’, die zumeist Bestandteil traditioneller Analogsynthesizer sind. Die Buchstaben ADSR stehen dabei für *A*ttack, *D*ecay, *S*ustain und *R*elease.

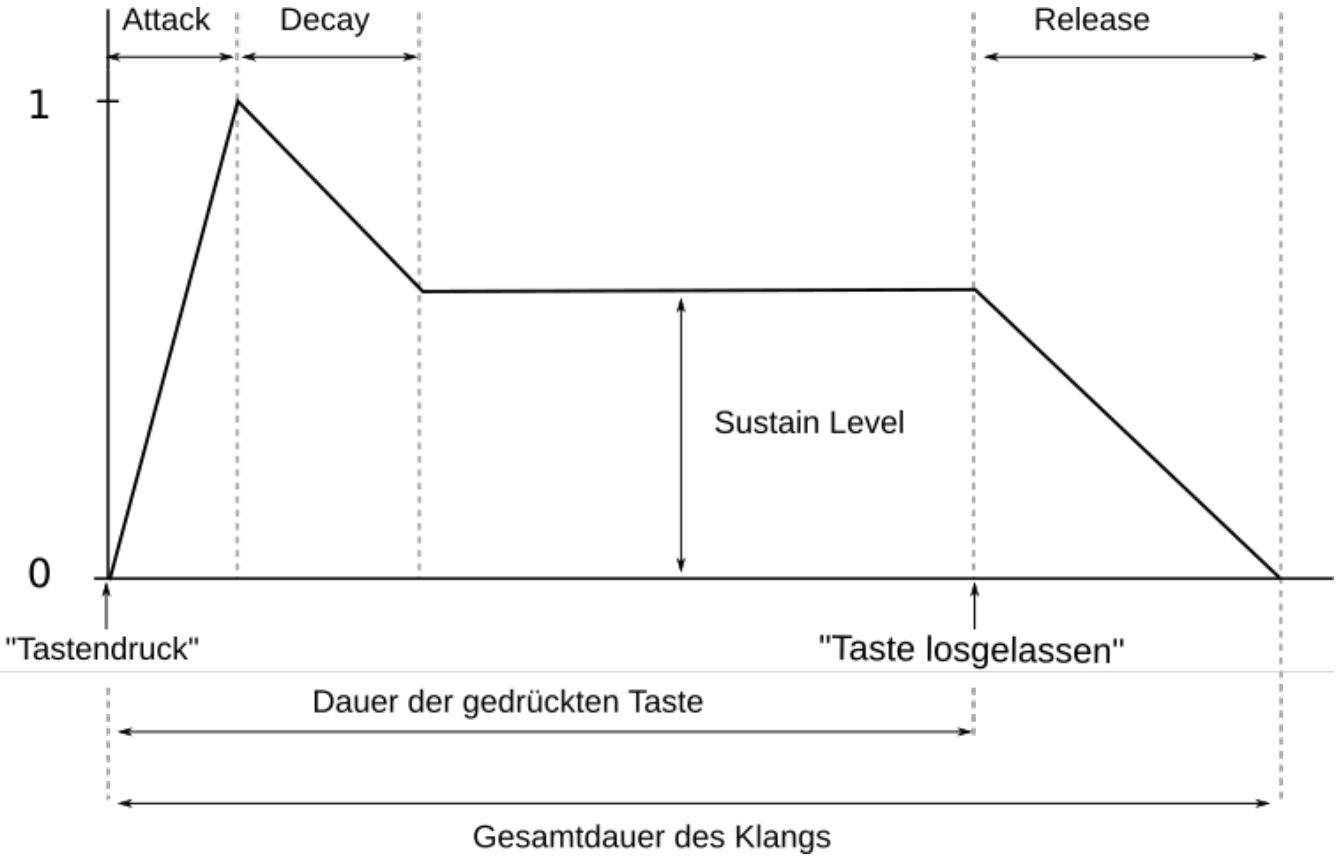


Abbildung 6.3: ADSR Hüllkurve

Abbildung 6.3 zeigt den Verlauf eines solchen Hüllkurvengenerators. Mit dieser Form der Hüllkurve soll das Ein- und Ausschwingverhalten von Instrumenten, und damit deren Artikulation simuliert werden.

- Die Attack Zeit gibt dabei die Dauer an, mit der der Hüllkurvengenerator von 0 auf den Wert 1 geht.
- Die Decay Zeit gibt die Dauer an, mit der der Hüllkurvengenerator direkt im Anschluß von 1 auf den ’Sustain Level’ geht.
- Der Sustain Level ist ein Wert zwischen 0 und 1, der die Lautstärke des Klangs nach dem durch den Attack und Decay simulierten Einschwingvorgang angibt.
- Die Release Zeit gibt die Dauer an, mit der der Hüllkurvengenerator am Ende des Klangs vom Sustain Level auf den Wert 0 geht. Damit wird das Ausschwingverhalten eines Klanges simuliert.

  Wie aus Abbildung 6.4 hervorgeht, lassen sich mit diesen Werten verschiedene traditionelle Artikulationen von Musikinstrumenten realisieren.

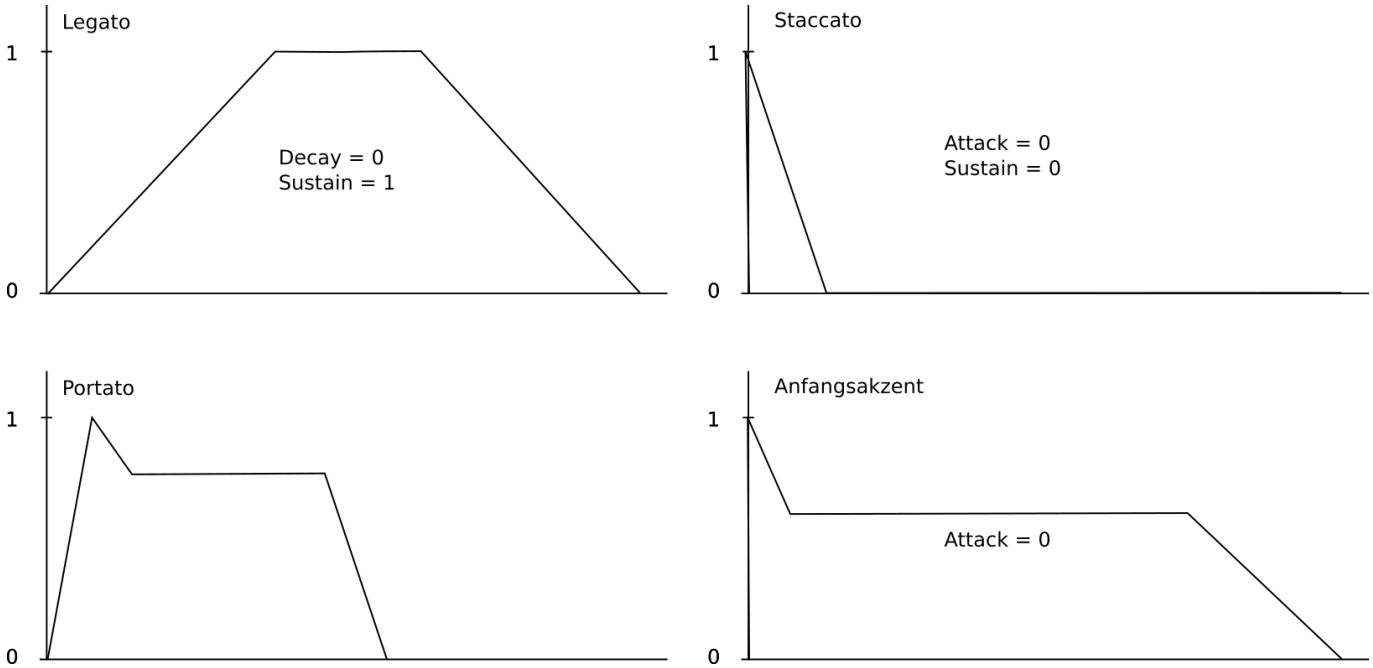


Abbildung 6.4: Besipiele verschiedener ADSR Hüllkurven

Bei Analogsynthesizern sind die Werte für Attack, Decay und Release immer absolute Zeitwerte, bei Computermusiksystemen existieren sowohl absolute, als auch proportionale Formen[1].

Wenn Instrumente live auf Tastaturen gespielt werden sollen, macht es Sinn, die Gesamtdauer des Klanges vom Spiel des Instrumentalisten abhängig zu machen. ADSR Hüllkurvengeneratoren funktionieren dabei so, dass Sie beim Auslösen des Klanges (beispielsweise durch das Drücken einer Taste auf dem Keyboard) zunächst die Einschwingphase (Attack und Decay) ausführen und dann den Sustain Level beibehalten, bis die Taste losgelassen wird. Durch das Loslassen der Taste wird die Release Phase ausgelöst und der Ton klingt, bis zum Ende dieser Phase weiter. Dadurch ist die Gesamtdauer des Klangs um die Dauer des Release Wertes länger, als die Dauer, in der die Taste gedrückt wurde (siehe Abbildung 6.3 unten).

Manche Computermusiksysteme, wie SuperCollider, simulieren dieses Verhalten mit Hilfe einer speziellen Variable, die 'Gate' genannt wird. Zu Beginn des Klangs wird dieser Wert auf 1 gesetzt. Der Klang wird dann solange gehalten, bis dieser Wert auf 0 gesetzt wird. Es ist dann Aufgabe des Anwenders, das Zeitverhalten des Klangs und die Hüllkurven für diese beiden Gatewerte zu definieren.

6.2 Instrumente und Partituren

6.3 Polyphonie

[1] Unter proportionaler Form versteht man Dauerangaben, die sich auf die Gesamtlänge eines Klangs beziehen. Dabei wird die Gesamtlänge auf den Wertebereich 0..1 normalisiert. Eine Attackzeit von 0.2 bedeutet also, dass der Attack in 1/5 der Gesamtdauer des Klanges erfolgt

Kapitel 7

Vertiefungen

7.1 Vertiefungen zur Einführung in die Sprache clojure

7.1.1 Liste mathematischer Operatoren

Liste mathematischer Operatoren und Ihre Anwendung:

```
;;; mathematische Grundoperationen: * + - /

(/ (* 4 (+ 2 3)) (- 7 5))
=> 10

;;; Inkrement und Dekrement:

(inc 2) ;=> 3
(dec 3) ;=> 2

;;; trigonometrische Funktionen: sin, cos

Math/PI ;=> 3.141592653589793

(Math/sin (* Math/PI 1/2)) ;=> 1.0

(Math/cos 0) ;=> 1.0

;;; Absolutwert, Rundung

(Math/abs -2) ;=> 2

(Math/round 2.4) ;=> 2

(Math/round 2.5) ;=> 3

;;; Erklärung von quot und mod:
;;;
;;; 11 / 4 ist ”(2 * 4) + 3”.
;;;
;;; Die 2 ist in diesem Fall der ”Ganzzahlanteil”, die 3 ist der
;;; ”Rest”.

;;; Ganzzahlanteil einer Division:

(quot 11 4) ;=> 2
```

```
;;; Rest bei einer Division:

(mod 11 4) ;=> 3

;;; höhere mathematische Funktionen: pow, exp, log

(Math/pow 2 3) ;=> 8.0

(Math/exp 1) ;=> 2.718281828459045

(Math/log 2.718281828459045) ;=> 1.0

;;; Definition des Zweierlogarithmus:

(defn log2 [x]
  (/ (Math/log x)
     (Math/log 2))) ;=> #'user/log2

(log2 8) ;=> 3.0
```

7.1.2 Funktionsdefinitionen

[?]Der Abschnitt fuzzy:Bindungen[fuzzy:Bindungen] stellt den speziellen Ausdruck `defn` als Bindung eines Symbols an eine Funktionsdefinition vor. Funktionen müssen jedoch keinen Namen haben, insbesondere dann, wenn Funktionen nur temporär oder lokal benötigt werden. Solche Funktionen werden 'anonyme Funktionen' genannt. In clojure wird eine anonyme Funktion mit dem Operator `fn` definiert:

```
;;; Definition einer anonymen Funktion:

(fn [x] (* x x))
=> #<user$eval7872$fn__7873 user$eval7872$fn__7873@503b27cc>
```

Da clojure bei der Auswertung nicht quotierter Listen als erstes Element einer Liste einen Funktionsoperator erwartet, könnte die Funktion `quadrat` aus dem Abschnitt fuzzy:Bindungen[fuzzy:Bindungen] auch folgendermaßen mit einer anonymen Funktion ausgeführt werden:

```
((fn [x] (* x x)) 4)
=> 16
```

Der spezielle Ausdruck `defn` ist im Grunde nur eine Abkürzungsnotation für den folgenden, äquivalenten Ausdruck:

```
;;; Funktionsdefinition mit "def"

(def quadrat (fn [x] (* x x)))
=> #'user/quadrat

(quadrat 4)
=> 16
```

Da bei der praktischen Arbeit mit clojure sehr häufig mit anonymen Funktionen gearbeitet wird, gibt es eine weitere Abkürzungsnotation für anonyme Funktionen. Wird einer Liste das #-Zeichen vorangestellt, wird die Liste als Definition einer anonymen Funktion interpretiert. In dieser Definition gibt es keine explizite Definition von Argumenten in eckigen Klammern. Vielmehr können Argumente der Funktion mit dem Zeichen `%` angesprochen werden. Die Quadratfunktion würde daher auch folgendermaßen aufgerufen werden können:

```
;;; Alternative Aufrufform einer anonymen Funktion:

(#(* % %) 4)
=> 16
```

Gibt es mehr, als ein Argument, werden die Argumente mit %1, %2, ... angesprochen:

```
(#(list %1 %2) 4 5)
=> (4 5)
```

7.1.3 Bindungen

7.2 Vertiefungen zur Einführung in overtone

Kapitel 8

Bibliografie

Lautstärke (mouse.x!)