michael STEPPAT

AUDIO-PROGRAMMIERUNG

// KLANGSYNTHESE
// BEARBEITUNG
// SOUNDDESIGN

HANSER

Im Internet: Aktuelle Ergänzungen und weitere Informationen

Steppat

Audioprogrammierung

Medien

Herausgeber: Ulrich Schmidt

Weitere Bücher der Reihe:
Fries: Mediengestaltung
Görne: Tontechnik
Heyna/Briede/Schmidt: Datenformate im Medienbereich
Petrasch: Videofilm
Raffaseder: Audiodesign
Rehfeld: Game Design und Produktion
Schmidt: Digitale Film- und Videotechnik

Michael Steppat

Audioprogrammierung

Klangsynthese, Bearbeitung, Sounddesign

Mit 66 Listings, 33 Tabellen, 72 Bildern und 48 Übungsaufgaben

HANSER

Der Autor:
Dr. Michael Steppat, Beuth Hochschule für Technik Berlin

Der Herausgeber:
Ulrich Schmidt, HAW Hamburg

Bibliografische Information Der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Lektorat: Mirja Werner, M.A., Franziska Jacob, M.A.
Herstellung: Der Buch*macher*, Arthur Lenner, München
Satz: Dr. Michael Steppat/le-tex, Leipzig
Coverconcept: Marc Müller-Bremer, www.rebranding.de, München
Coverrealisierung: Stephan Rönigk
Datenbelichtung, Druck und Bindung: Kösel, Krugzell
Printed in Germany

ISBN: 978-3-446-43222-2
E-Book-ISBN: 978-3-446-44198-9

www.hanser-fachbuch.de

Vorwort

Die Möglichkeit, Audiosignale in diskrete Zahlenfolgen zu verwandeln, sie in beliebiger Weise am Computer zu bearbeiten und anschließend wieder in hörbare Signale zurückzuwandeln, bietet eine Reihe von interessanten Ansätzen. Klänge lassen sich mit verschiedenen Algorithmen erzeugen, Synthesizer, die früher aufwendige elektronische Schaltungen erforderlich machten, können mit Softwaresynthesizern nachgebaut werden. Auch digitale Effekte wie eine Klangfilterung oder die Erzeugung eines Echos lassen sich in Programmen erzeugen.

Wer sich tief greifender mit Audioprogrammierung beschäftigen möchte, kommt an der Nachrichtentechnik mit ihrer sehr mathematisch gehaltenen Ausdrucksweise nicht vorbei. Für Studierende der Ingenieurwissenschaften werden die dafür erforderlichen Grundlagen in den Vorlesungen zur Höheren Mathematik vermittelt. Da dieses Buch sich in erster Linie an Studierende und Interessierte der Informatik, Mediengestaltung und Musikwissenschaft richtet, können über die Möglichkeit der Programmierung oder Berechnung mit der Tabellenkalkulation hier Brücken gebaut werden, die über diesen Weg den Zugang zu der mathematischen Sprache öffnen sollen.

Grundkenntnisse in einer objektorientierten Programmiersprache wie JAVA und/oder C++ werden in diesem Buch vorausgesetzt. Die Programmbeispiele eignen sich aber auch ganz gut als ergänzende Übungen für Leser, die mit der Programmierung erst beginnen.

Alle Programmbeispiele sind bewusst einfach gehalten. Auf eine Fehlerbehandlung wurde weitgehend verzichtet, damit die eigentlichen Abläufe des Produktivcodes deutlich bleiben und die Lesbarkeit der Quelltexte nicht durch Fehlerbehandlungsroutinen eingeschränkt wird. Die dadurch entstehende Fehleranfälligkeit, die insbesondere durch Bedienfehler wie falsch übergebene Parameter auftreten kann, dient dem Ansatz der testgetriebenen Entwicklung und kann auch dazu benutzt werden, den Blick für die Auftretenswahrscheinlichkeit bestimmter typischer Fehler in der Audioprogrammierung zu schulen.

Ganz herzlich danken möchte ich Frau M.A. Mirja Werner, Frau M.A. Franziska Jacob, Frau Dipl.-Ing. Franziska Kaufmann und Herrn Dr. Martin Feuchte vom Carl Hanser Verlag Leipzig für die gute und konstruktive Zusammenarbeit. Herrn Stephan Rönigk danke ich für die Umschlaggestaltung und Herrn Arthur Lenner für die Unterstützung beim Layout. Meinem Herausgeber Herrn Prof. Dr. Ulrich Schmidt danke ich für die wertvollen Hinweise zur inhaltlichen Gestaltung des Buches und den fachlichen Austausch. Alle hier genannten haben viel zum Gelingen dieses Buches mit beigetragen. Besonders danken möchte ich

auch meiner Frau Regina und meinen Söhnen Adrian und Frederic für ihre Unterstützung während der Entstehung dieses Buches.

Trotz großer Sorgfalt lässt sich sicherlich nicht vollständig verhindern, dass sich vielleicht der eine oder andere Fehler eingeschlichen hat. Wenn Sie also Kritik, Anmerkungen oder auch Wünsche haben, senden Sie einfach eine E-Mail an steppat@beuth-hochschule.de. Wir werden versuchen, dies in den kommenden Auflagen zu berücksichtigen. Die jeweils aktuellsten Ergänzungen und weitere Informationen können Sie unter *http://projekt.beuth-hochschule.de/audioprogrammierung* finden.

Michael Steppat Im März 2014

Inhalt

1 Einleitung

1.1 Was ist Audioprogrammierung?

Seitenweise mathematische Formulierungen zu studieren oder handschriftlich Berechnungen zu erstellen, kann leicht zu einer ermüdenden Angelegenheit werden. Mit dieser Mühe hatte auch Computer-Erfinder Konrad Zuse zu kämpfen. Als Bauingenieur mit statischen Berechnungen beauftragt, die er als monoton und mühsam empfand, entwickelte er die Idee, diese automatisch durchführen zu lassen. Aus dieser entstand 1938 Zuses erster Rechner: Die Z1. Sie kann aufgrund der heute noch in ähnlicher Weise aufgebauten Architektur als Vorläufer des modernen Computers betrachtet werden.

Die Audiotechnik als ein Teilgebiet der Physik kann rein verbal nur in begrenzter Weise erklärt werden. Viele Vorgänge können zwar mit Hilfe von Modellen beschrieben werden, doch dies gelingt oftmals nur durch eine Idealisierung. Daher wird hier auf die Mathematik zurückgegriffen, die dazu geführt hat, das Audiotechnik in weitem Umfang aus mathematischen Formulierungen besteht.

Gerade dieser Umstand bedeutet aber eine Chance, denn mathematische Formulierungen lassen sich mit Leichtigkeit in Computerprogramme umsetzen. Mit Hilfe der Audioprogrammierung gelingt es, Lösungen von Gleichungen hörbar zu machen oder sie in geeigneter Weise als Kurven zu visualisieren. Die Physik der Tonentstehung, z.B. bei Musikinstrumenten, kann damit nachvollzogen werden. Sie kann einerseits dazu genutzt werden, virtuelle Synthesizer zu bauen oder helfen die Physik von Musikinstrumenten zu untersuchen.

Die Bearbeitung von Klangmaterial deckt ein weiteres Feld der Audioprogrammierung ab. In der trockenen Akustik eines Tonstudios aufgenommene Instrumente erhalten ihren Klang erst nach einer Reihe von Signalbearbeitungen. Dazu zählt die Bearbeitung mit Klangfiltern, die Anpassung der Dynamik und das Hinzufügen digitaler Effekte wie Reverb, Delay, Chorus usw. Programme, die diese Signalbearbeitung durchführen, werden in der Regel als Plug-ins auf dem Rechner installiert.

Audiodaten in Rohform zu übertragen erfordert hohe Bandbreiten, daher müssen die zu übertragenen Daten vor dem Senden komprimiert werden und beim Empfänger wieder entpackt werden. Mit der Audioprogrammierung können Codecs erstellt werden, die diese Aufgaben übernehmen.

Der Begriff der Audioprogrammierung deckt somit ein weites Feld ab. Das Einspielen einer Melodie mit einem MIDI-Editor oder die Erstellung eines Songs an einer digitalen Audioworkstation (DAW) zählt hierzu ebenso, wie die maschinennahe Programmierung eines Treibers für einen Audioadapter. Die Anforderungen an die Programmiererinnen und Programmierer sind daher sehr unterschiedlich. Eine Komposition eines Musikstücks am Rechner setzt neben der Vertrautheit mit der Software vor allem Kenntnisse der Harmonie- und Kompositionslehre voraus, wohingegen bei der Programmierung eines Audiotreibers fundierte Kenntnisse der Hardware und der Rechnersystemarchitektur erwartet werden. Die Programmierung hängt zudem auch vom verwendeten Betriebssystem ab.

Neben den schon genannten Beispielen gehören zu den praktischen Anwendungsgebieten der Audioprogrammierung die Entwicklung digitaler Musikinstrumente wie z.B. Softwaresynthesizern. Da Klänge als mathematische Funktionen beschreibbar sind, können diese in Computern als diskrete Zahlenfolgen generiert werden. Das Ergebnis sind dann Audiodatenströme (Streams), die über den Audioadapter hörbar gemacht werden können oder Audiodateien. Auch die Bearbeitung von Audiodatenströmen lässt sich softwaretechnisch mit viel weniger Aufwand umsetzen als eine Hardwarelösung.

So kann zum Beispiel durch eine einfache Multiplikation einzelner Samples eines Streams eine Verstärkung bewirkt werden. Die Berechnung kann in einer Schleife erfolgen und benötigt nur wenige Codezeilen. Ein Verstärker als elektronische Schaltung ist hingegen um ein Vielfaches aufwendiger. Ebenso können in einer Schleife zwei Streams durch Addition zu einem zusammengemischt werden. Damit kann dann in wenigen Schritten ein digitales Mischpult programmiert werden, wobei der notwendige Aufwand für die Programmierung einer zugehörigen funktionsfähigen Oberfläche nicht unerheblich ist.

Die digitale Signalverarbeitung von Streams erfolgt blockweise. Jeder Block besteht aus einer bestimmten Anzahl von Samples. Daher erfolgt die Bearbeitung (Processing) fast ausschließlich in Schleifen, deren Durchläufe identisch mit der Blocklänge sind. Da viele Bearbeitungen im Frequenzbereich stattfinden, muss der Stream dazu häufig in diesen Bereich transformiert werden. Für eine effiziente Transformation eignen sich daher Blocklängen ganz gut, die ganzzahlige Zweierpotenzen darstellen wie z.B. 128, 256, 512, 1024. Kurze Blocklängen ermöglichen eine schnelle Reaktionszeit des Systems, wenn z. B. Parameter dynamisch verändert werden müssen, wie z.B. bei einer Mischpultanwendung.

Bei zu großen Blocklängen setzt bei einer Änderung eine hörbare Verzögerung ein, die ein für Audioproduktionen sehr wichtiges intuitives Arbeiten unmöglich macht. Größere Blöcke haben hingegen den Vorteil, dass sie weniger anfällig für Störungen sind, wenn es im Betriebssystem zu zeitlichen Problemen bei der Abarbeitung von Prozessen kommt.

Die Blöcke werden in Datenfeldern gespeichert. Der Datentyp hängt dabei von der Wortbreite der Samples ab. Eine Wortbreite von 8-bit wird in Datenfeldern vom Typ char gespeichert, eine Wortbreite von 16-bit im Typ short. Für die Signalverarbeitung sind diese als Ganzzahlwerte nur bedingt geeignet. Um Rundungsfehler bei der Signalverarbeitung zu vermeiden, erfolgt diese fast ausschließlich mit float-Werten, deren Wortbreite bei 32-bit liegt.

Da Datenfelder indiziert sind und jeder Index dividiert durch die Abtastrate eine diskrete Zeitinformation darstellt, kann dieser sehr gut zum Aufbau von Verzögerungsgliedern verwendet werden. Mit Verzögerungsgliedern lassen sich einfache Effekte wie z.B. ein Echo, aber auch komplexe Effekte wie das Reflexionsverhalten eines Raums als künstlicher Nachhall erzeugen. Verzögerungsglieder, Multiplizierer und Addierer sind die Grundelemente

der digitalen Audiosignalverarbeitung und dienen neben der schon genannten Realisierung von Audioeffekten auch der Realisierung digitaler Filter.

Audioprogrammierung kann visuell oder textbasiert durchgeführt werden. Die visuelle Programmierung bietet die Möglichkeit, den Signalfluss als Graphen darzustellen, dessen Knoten einzelne Funktionseinheiten darstellen. Die Verbindungskanten dienen als „virtuelle" Kabel.

Durch die Möglichkeit, die Graphen ineinander zu verschachteln, indem komplette Graphen als Knoten in einen übergeordneten Graph eingefügt werden, ist so eine gute Modularisierung und ein intuitives Programmieren möglich. Visuelle Programmierung hat den Vorteil, bestimmte Funktionseinheiten, wie z.B. einen Tongenerator oder Effekt als Blackbox zu betrachten. In sogenannten Filtergraphen oder patches lässt sich in einer grafischen Umgebung der Signalfluss zeichnen.

Programme, die diese Möglichkeit bieten, sind GraphEdit (ein Werkzeug von Microsoft), Reaktor der Firma Native Systems und Max MSP von cycling 74. Zu Letzterem gibt es noch die Open-Source-Alternative pure Data, die von Max-Entwickler Miller Puckette weiterentwickelt wurde.

In der textbasierten Programmierung werden objektorientierte Programmiersprachen wie Java, C++ oder Javascript bevorzugt eingesetzt. Bei der Softwarearchitektur ist als Paradigma die datenstromorientierte Programmierung von Vorteil. Datenströme werden in Form von Parametern an eine Methode übergeben, welche die Bearbeitung dann durchführt. In Audioeditoren erfolgt die Bearbeitung von Effekten häufig mit Plug-ins. Diese Module sind eigenständige Anwendungen, die vom Audioeditor über eine Schnittstelle aufgerufen werden können. Die Signalbearbeitung findet dann in einer dafür vorgesehenen Methode statt, welche die Audiodaten in einem Puffer übergeben bekommt und die bearbeiteten Daten zurückgibt. Alle weiteren Methoden in einem Plug-in sind Zugriffsmethoden, mit welchen die Parameter des Effekts eingestellt werden können.

Der Aufbau von Audiodateien lässt sich durch die Programmierung einer digitalen Synthese von Sinustönen und Klängen veranschaulichen. Das Grundverständnis für die Signalverarbeitung wird durch einfache Bearbeitungen wie Verstärkung, Normalisierung und Mischen von Daten deutlich.

Audiodatenmengen lassen sich mit der Datenkompression reduzieren, die sich grob in verlustlose und verlustbehaftete Verfahren aufteilen lässt. Das mittlerweile weit verbreitete MP3-Format ist ein solches verlustbehaftetes Verfahren. Es werden nur die Signalanteile codiert, die auch wahrgenommen werden. Die Grundlage dafür liefern die physiologischen Eigenschaften des Gehörs. Zur Encodierung und Decodierung werden Codecs verwendet, welche auf unterschiedliche Weise programmiert werden können.

Audiodaten lassen sich als Zeitfunktion oder Spektrum visualisieren. Die Visualisierung der Zeitfunktion dient einer schnellen Erkennung der Aussteuerung und als Hilfe beim Schneiden. Anfang und Ende eines Audioereignisses können dort schnell gefunden werden. Im Spektrum lassen sich die charakteristischen Frequenzanteile einzelner Instrumente bestimmen. So kann einer Klangverfärbung durch Überbetonung oder Fehlen einzelner Frequenzanteile vorgebeugt werden. Mit Hilfe der Fouriertransformation wird eine Zeitfunktion in den Frequenzbereich transformiert. Da es sich bei der Fouriertransformation um eine Integraltransformation handelt, ist die analytische Berechnung sehr aufwendig.

Die viel einfachere numerische Berechnung kann mit Hilfe von Tabellenkalkulationen oder Programmen durchgeführt werden.

Mathematische Funktionen eignen sich sehr gut für die Berechnung von synthetischen Klängen und sind die Grundlage von Softwaresynthesizern. Durch Filterung kann das erzeugte Spektrum klanglich bearbeitet werden und mit ebenfalls mathematischen Hüllkurvenfunktionen kann der für Musikinstrumente sehr charakteristische Ein- und Ausschwingvorgang simuliert werden.

Neben der Speicherung der Audiodaten kann Musik auch in MIDI-Dateien gespeichert werden, welche dann die Daten der zu spielenden Noten mit Tonhöhe, Anschlagstärke und zeitlicher Länge für die beteiligten Instrumente enthält. Kompositionen lassen sich mit MIDI-Editoren oder Notensatzprogrammen erstellen. Dies hat den Vorteil, dass die Instrumente nach Fertigstellung des Werkes beliebig ausgetauscht werden könnnen. Zudem sind MIDI-Daten im Vergleich zu Audiodaten sehr platzsparend. Softwaresynthesizer werden über MIDI-Daten gesteuert.

Ein weiterer Aspekt der Klangsynthese ist die physikalische Modellierung einfacher Klangkörper wie Saiten, Stäbe und Platten. Diese erfolgen über eine numerische Berechnung der Bewegungsgleichungen der Klangkörper. Als Eingabeparameter dienen dabei reale physikalische Größen wie Materialeigenschaften und geometrische Abmessungen.

Audioprogrammierung findet auch unter dem Aspekt des Sounddesigns bei Film und audiovisuellen Darbietungen wie z.B. für Tanz oder Medieninstallationen statt. Programmiert wird zum einen bei der Komposition der Filmmusik, zum anderen bei der Erstellung der Sounds mit Hilfe von selbst programmierten Softwaresynthesizern oder der physikalischen Modellierung.

Mit dem in HTML 5 verfügbaren Audioplayer gewinnt die Audioprogrammierung mif Javascript auch für webbasierte Anwendungen zunehmend an Bedeutung. Die Mozilla Audio Data API bietet neben einer guten Dokumentation auch auf der Startseite zwei schöne Beispiele für die Audioprogrammierung mit Javascript: *https://wiki.mozilla.org/Audio_Data_API*. Im ersten Beispiel wird die Visualisierung mit einem Spektrum-Analyser vorgestellt, das zweite ist ein einfacher Tongenerator. Weiterführende Links führen zu einem reichhaltigen Angebot an Beispielen.

Von Vorteil ist es, sich bei der Audioprogrammierung nicht auf eine bestimmte Programmiersprache festzulegen, sondern mit verschiedenen Sprachen zu experimentieren. Da bei der Audioprogrammierung (von der Programmierung der Benutzeroberfläche mal abgesehen) die Algorithmen zur Signalbearbeitung im Vordergrund stehen, lassen sich diese oftmals mit wenig Aufwand von der einen in die andere Programmiersprache portieren.

Audioprogrammierung lässt sich in drei Kategorien entlang der Signalverarbeitungskette einteilen. Die erste Kategorie umfasst die Erzeugung von Signalen, die zweite die Verarbeitung und die dritte die Ausgabe bzw. Verteilung. Die im Text schon angesprochenen Unterkategorien sind in der folgenden Tabelle 1.1 aufgelistet.

TABELLE 1.1 Kategorien der Audioprogrammierung

Erzeugung	Verarbeitung	Ausgabe
▪ Softwaresynthesizer ▪ Audiotreiber (Aufnahme) ▪ Komposition von MIDI-Songs ▪ physikalische Modellierung ▪ Stream empfangende Websockets	▪ Dynamikbearbeitung ▪ Klangfilter ▪ Audioeffekte ▪ Schnittbearbeitung ▪ Fast Fourier Transformation ▪ schnelle Faltung	▪ Audio-Codecs ▪ Audiotreiber (Wiedergabe) ▪ Stream sendende Websockets ▪ Visualisierung

1.2 An wen richtet sich dieses Buch?

Das Buch richtet sich in den Grundlagen an alle, die mit Hilfe der Programmierung ein tief greifenderes Verständnis der Audiotechnik erlangen wollen. Die Programmierung kann zu einem besseren Verständnis mathematischer Formulierungen beitragen, da mit ihr die Rechengänge besser nachvollzogen werden können. Ebenso lassen sich durch Simulation physikalische Sachverhalte untersuchen, wie die Akustik von Musikinstrumenten. Für die Übertragung und Speicherung von Audiodaten gibt es eine Vielzahl verschiedener Datenformate. Ein Verständnis für den Aufbau dieser Formate lässt sich durch die Programmierung herstellen.

Aufbauend auf diesen Grundlagen werden in der Weiterführung Möglichkeiten aufgezeigt, Softwaresynthesizer, Effektprogramme und andere Anwendungen wie z.B. auch Audioeditoren zu programmieren. Mit der Modellierung realer Musikinstrumentenklänge können die akustischen Eigenschaften und physikalischen Vorgänge von Musikinstrumenten anschaulich gemacht werden und die Erkenntnisse im Sounddesign eingesetzt werden. Somit richtet sich dieses Buch an Studierende und Lehrende der Informatik, Elektrotechnik, Akustik, Musik, Musikwissenschaft und ebenso an angehende Filmmusikkomponisten und Sounddesigner.

1.3 Wie kann man mit diesem Buch arbeiten?

Da die einzelnen Kapitel weitgehend thematisch in sich geschlossen sind, kann grundsätzlich jedes Kapitel als Einstieg verwendet werden. Die Schwerpunkte liegen in den vorderen Kapiteln (2–4) stärker bei den Grundlagen und der Theorie. In den hinteren Kapiteln (5–10) steht die Anwendungsorientierung im Vordergrund. Daher eignet sich das Buch auch gut dazu, die Kapitel der Reihe nach durchzuarbeiten.

Die grundlegenden Programmiertechniken zum jeweiligen Thema werden in den Kapiteln als Quelltexte vorgestellt und können unter der URL *http://projekt.beuth-hochschule.de/audioprogrammierung* heruntergeladen werden. Im Anhang findet sich eine Übersicht der verwendeten Beispiele mit Erläuterungen. Jedes Kapitel schließt mit Übungsaufgaben ab, die es ermöglichen, die Inhalte weiter zu vertiefen. Es finden sich dort auch Anregungen für eigene Entwicklungen.

Für alle in Java geschriebenen Beispiele ist für deren Ausführung das Java-Entwicklungswerkzeug (JDK) und die Java-Laufzeitumgebung (JRE) ausreichend. Eine komfortable Bearbeitung ermöglicht die integrierte Entwicklungsumgebung (IDE) Eclipse mit welcher die Beispiele auch erstellt wurden. Eclipse kann unter der URL: *http://www.eclipse.org/downloads/* heruntergeladen werden.

Die Programmierbeispiele in C++ wurden mit der IDE Microsoft Visual Studio erstellt. Eine kostenlose Express-Edition dieser IDE kann unter der URL: *http://msdn.microsoft.com/de-DE/vstudio* bezogen werden.

Im kommenden Kapitel werden die Grundlagen der Audiosignalerzeugung und deren Speicherung vorgestellt.

2 Audiosignale

Fragen, die dieses Kapitel beantwortet:

- Was ist Schall?
- Was versteht man unter einem Signal?
- Wie wird ein akustisches Signal in ein elektrisches Signal umgewandelt?
- Wie ist ein Signal zusammengesetzt?
- Was versteht man unter einem Ton?
- Warum können analoge Signale nicht in Computern gespeichert werden?
- Wie wird ein analoges Signal in eine digitale Zahlenfolge umgewandelt?
- Was versteht man unter Abtastung und wie entstehen Abtastfehler?
- Was ist Quantisierung?
- Wie ist eine Audiodatei im WAV-Format aufgebaut? Welche Formate gibt es?
- Wie lässt sich eine Datei mit einem Ton erzeugen?
- Wie kann die Amplitude von Audiodaten bearbeitet werden?
- Wie werden Audiodatenströme gemischt?
- Was versteht man unter Normalisierung und wie wird diese durchgeführt?
- Was unterscheidet eine AIFF-Datei von einer WAV-Datei?

2.1 Akustische Signale

Mit **Schall** werden alle Ereignisse bezeichnet, die durch das Gehör wahrgenommen werden. Physikalisch gesehen ist er eine Abfolge von Luftdruck- und Dichteschwankungen, welche sich in Form von Wellen von einer Schallquelle her ausbreiten [Kutt04]. Diese Form der Wellenausbreitung lässt sich zum Beispiel am Ufer eines möglichst ruhigen Gewässers veranschaulichen, indem man einen Stein in das Wasser wirft. Die Wellen breiten sich dann von der Einwurfstelle ringförmig aus [Vorl08]. Ursache für eine Schallentstehung

können mechanische Schwingungen sein, wie z.B. das angeschlagene Fell einer Trommel, eine angezupfte Saite oder sich plötzlich verändernde Luftvolumen wie z.B. das Knallen eines Sektkorkens oder eines Feuerwerkskörpers. Auch der auf einen Blitz folgende Donner entsteht durch eine Volumenveränderung. Die beim Blitz entstehende Hitze dehnt die Luft schlagartig aus und erzeugt eine Druckwelle. Analog zu den mit der Schallausbreitung verbundenen Lufdruckschwankungen und der Dichte ändert sich bei allen Schallereignissen in der Welle auch die Lage der Luftmoleküle. Sie bewegen sich mit einer Wechselgeschwindigkeit um ihre Ausgangslage. Diese Größe wird Schallschnelle genannt und steht im Zusammenhang mit dem Schalldruck.

Wird bei einem Schallereignis eine Tonhöhe wahrgenommen, spricht man umgangssprachlich von einem **Ton**. Dieser kann gesungen oder auf einem Musikinstrument gespielt werden. Jede Tonhöhe hat eine der Oktave und Tonleiter entsprechende Bezeichnung. Eine C-Dur-Tonleiter besteht z.B. aus den Tonhöhen: c, d, e, f, g, a, h, welche den weißen Tasten auf dem Klavier entsprechen. Wird keine Tonhöhe wahrgenommen, so spricht man von einem Geräusch. Beide sind im technischen Sinn Klänge, welche sich aus einzelnen Tönen zusammensetzen. Ein **Klang** entsteht durch die Überlagerung einzelner Töne.

Der Begriff des Tones ist im technischen Sinne anders zu verstehen. Als **Ton** bezeichnet man dort eine periodisch ablaufende sinusförmige Schwingung. Diese lässt sich z.B. mit einer Stimmgabel erzeugen. Sie besteht aus einer positiven Flanke (Auslenkung > 0) mit einem Maximum, welcher nach einem Nulldurchgang (Auslenkung = 0) in eine negative Flanke (Auslenkung < 0) mit einem Minimum übergeht und lässt sich mit den drei Parametern Amplitude, Frequenz und Phasenlage beschreiben: Die **Amplitude** gibt den Betrag der maximalen Auslenkung in positiver und in negativer Richtung an [Webe85]. Je größer die Amplitude, desto lauter wird der Sinuston wahrgenommen. In Bild 2.1 ist eine Sinusschwingung mit den Amplitudenwerten 0,25; 0,5 und 1,0 dargestellt.

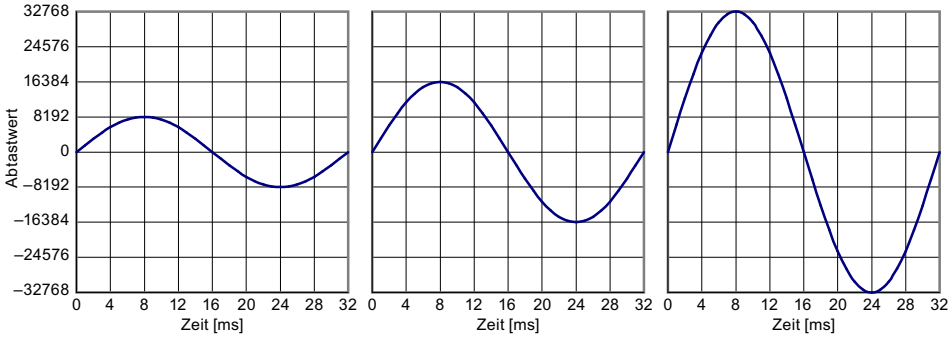

BILD 2.1 Sinusschwingung mit verschiedenen Amplituden

Bei Sinustönen hat die Frequenz einen Einfluss auf die wahrgenommene Tonhöhe. Mit der **Frequenz** wird die Anzahl der Periodendurchläufe pro Sekunde angegeben. Je höher die Frequenz, desto höher auch der wahrgenommene Ton. Das menschliche Gehör nimmt im Idealfall im sogenannten Hörbereich Frequenzen von 20 bis 20000 Hz wahr. Als **Infraschall** bezeichnete Töne unter 20 Hz werden als pulsierende Luftdruckschwankungen wahrgenommen, welche ab 16 bis 20 Hz in eine kontinuierlich wahrgenommene Tonhöhe übergehen. Diese Grenze wird als untere **Hörgrenze** bezeichnet. Die obere Hörgrenze, welche bei Kleinkindern noch bis zu 20000 Hz beträgt, nimmt mit zunehmendem Lebensalter ab

und kann individuell sehr verschieden sein [Meye94]. Bei jungen Erwachsenen geht man von einer oberen Hörgrenze von 16000 Hz aus, welche sich pro Lebensjahrzehnt durchschnittlich um ca. 1000 bis 2000 Hz nach unten verschiebt. Das Bild 2.2 zeigt drei Sinusschwingungen mit den Frequenzen 31,25; 62,5 und 73,75 Hz. Der abgebildete Zeitraum beträgt immer 32 ms.

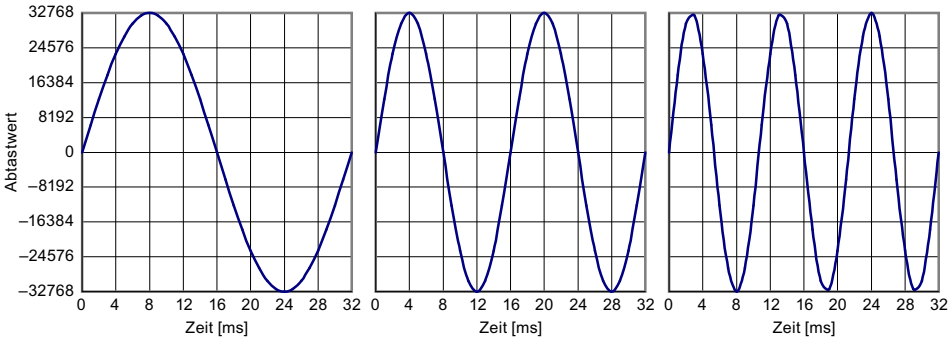

BILD 2.2 Sinusschwingung mit unterschiedlichen Frequenzen

Mit der **Phase** als drittem Parameter wird die Anfangsposition der Auslenkung zu einem bestimmten Zeitpunkt bezeichnet. Da man sich die Sinusfunktion als Auslenkung eines rotierenden Zeigers vorstellen kann, dessen Position mit einem Winkel relativ zu einem Bezugspunkt angegeben werden kann, wird auch die Phase in einem Winkel angegeben. Hierbei geht man von einem linksdrehenden Zeiger aus, dessen Anfangsposition 0° der Position 3 Uhr entspricht. Einer Vierteldrehung nach links entspricht 90°, einer Umdrehung 360° [Veit74].

In der Literatur wird für die Winkelangabe fast ausschließlich das Bogenmaß verwendet. Dies ist das Verhältnis eines Kreisbogens zu dessen Radius. Dabei entsprechen 2π einer ganzen Umdrehung. Eine Umrechnung vom Gradmaß in das Bogenmaß b erfolgt dadurch, indem man das gegebene Gradmaß β mit π multipliziert und dann durch 180 teilt: $b = \beta * \pi/180$. Um die Konstante 2π nicht immer mit angeben zu müssen, wird bei der Frequenzangabe in der Literatur fast ausschließlich die Kreisfrequenz ω verwendet. Diese ist die mit dem Faktor 2π multiplizierte Frequenz [Bart11].

Mit diesen drei Parametern sind die Eigenschaften einer Sinusschwingung hinreichend beschrieben. In Bild 2.3 werden die unterschiedlichen Phasenlagen veranschaulicht.

Ein **akustisches Signal** ist eine zeitveränderliche messbare physikalische Größe. Die Schwingungen der Stimmgabel drücken die umgebenden Luftmoleküle periodisch zusammen und ziehen sie auseinander. Dies führt zu einer Schwankung des Luftdrucks, welcher in der Einheit Pascal gemessen wird.

Zeichnet man diese Luftdruckschwankungen auf, erhält man eine Darstellung des Schalldrucks in Abhängigkeit von der Zeit. Diese Funktion wird **Zeitfunktion** genannt und ist die in Audioeditoren gebräuchlichste Darstellung zur Bearbeitung von Audioinhalten. In ihr lässt sich auf einen Blick erkennen, an welchen Passagen Schallereignisse stattfinden, welche Aussteuerung (Amplitude) sie haben und wo sich Pausen befinden. Das Auffinden der Pausen bzw. der Nulldurchgänge ist für die Schnittbearbeitung von Bedeutung, da nur in den Nulldurchgängen geschnitten werden darf, um Klickgeräusche zu vermeiden, die

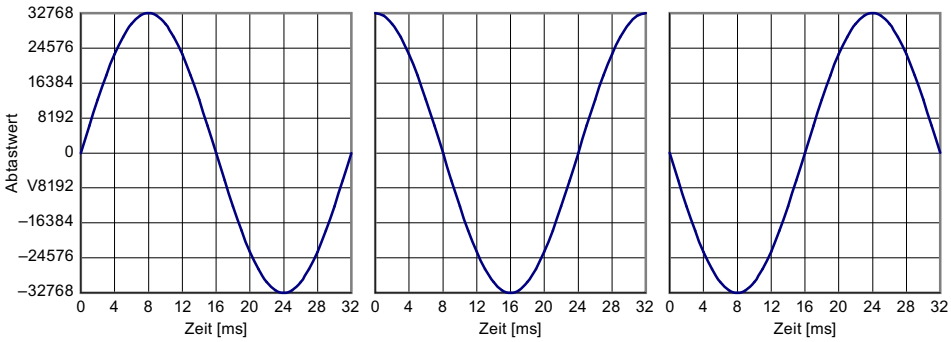

BILD 2.3 Sinusschwingungen mit den Phasenlagen 0, 90, 180° (im Bogenmaß: 0, $\frac{\pi}{2}$, π)

durch plötzliche Signalsprünge hervorgerufen werden. Die Zeitfunktion kann auch mit dem Oszilloskop sichtbar gemacht werden [Brau97].

2.2 Schallwandlung

Um akustische Signale aufzunehmen und mit Audiosoftware bearbeiten zu können, müssen diese in mehreren Schritten gewandelt werden. Als Luftschall vorliegende Schallereignisse werden aufgenommen, in elektrische Signale gewandelt und daraus Messdaten erzeugt, welche auf dem Rechner als Zahlenfolgen gespeichert und weiterverarbeitet werden können.

Der erste Schritt ist die Wandlung des akustischen Signals in ein elektrisches Signal über ein Mikrofon, welches eine zu dem Schalldruck proportionale elektrische Spannung erzeugt. Diese wird im Audioadapter oder in einem vorgeschalteten Mischpult verstärkt und zu einem **Analog-Digital-Wandler** (AD-Wandler) geleitet, welcher das in kontinuierlicher Form vorliegende Signal abtastet und in zeitdiskrete Werte wandelt. Die Abtastwerte werden digital entweder als 8, 16, 24, 32 oder 64-bit Werte ausgegeben und können nun in einem digitalen Signalverarbeitungssystem weiterverarbeitet werden [Dick08].

Der Abtastvorgang erfolgt in regelmäßigen zeitlichen Abständen in einer festgelegten **Abtastrate**. Gebräuchlich sind auf modernen Systemen die Abtastraten 44100, 48000, 96000 und 192000 Hz. Neben jeder theoretisch beliebigen Abtastrate waren früher auch die Formate: 8000, 11025, 16000, 22050 und 32000 Hz gebräuchlich. Da diese von der Bandbreite nicht den gesamten Hörbereich abdecken, geht dies auf Kosten der Klangqualität. Die Verwendung dieser Formate beschränkte sich daher in der Regel auf Sprachaufnahmen und diente hauptsächlich dazu, Speicherplatz auf der Festplatte einzusparen. Mit zunehmender Speicherkapazität der Festplatten haben diese Formate an Bedeutung verloren.

Die Abtastung entspricht der zeitlichen Momentaufnahme einer Schwingung. Veranschaulichen lässt sich dies mit einem kleinen Gedankenexperiment, in dem man die Auslenkung eines einfachen Schwingers betrachtet, welcher aus einem kleinen Gewicht besteht, das an einer Feder hängt. Zieht man an dem Gewicht und lässt es anschließend los, so pendelt das Gewicht in einer bestimmten Frequenz um seine Ausgangslage herum [Boru84].

Befestigt man an dem Gewicht einen kleinen Stift, der auf ein Lineal zeigt, kann man die Auslenkung messen. Die einzelnen Werte lassen sich erfassen, wenn die Bewegung mit einer Videokamera aufgezeichnet wird. Jedes Einzelbild liefert einen Messwert zu einem bestimmten Zeitpunkt. Durch die festgelegte Bildwechselfrequenz findet die Messung in konstanten Zeitabständen statt.

Nach dem gleichen Prinzip erfolgt die Abtastung der elektrischen Spannungswerte im AD-Wandler. In regelmäßigen Abständen wird die Spannung gemessen in einen Zahlenwert mit einer festgelegten Bitbreite übertragen und gespeichert. Bei der Abtastung ist zu beachten, dass die Abtastfrequenz immer höher als das Zweifache der höchsten zu übertragenden Frequenz ist. Bei Nichtbeachtung treten sogenannte Alias-Signale in Form von Spiegelfrequenzen auf. Die **Spiegelfrequenz** ist die Differenz von Abtastfrequenz und Signalfrequenz. Wird zum Beispiel ein 1000-Hz-Signal abgetastet, so muss die Abtastfrequenz größer als 2000 Hz sein.

Eine zu niedrige Abtastung mit nur 1500 Hz würde dazu führen, dass eine Spiegelfrequenz von 500 Hz entsteht. Um sicherzustellen, dass das Signal keine Anteile enthält, deren Frequenzen größer/gleich der halben Abtastfrequenz sind, wird das Signal vor der Wandlung mit einem Tiefpassfilter gefiltert, welches diese Anteile abschneidet. Für den Hörbereich des menschlichen Gehörs bedeutet dies, dass die Abtastrate größer als 40000 Hz sein muss, um alle hörbaren Frequenzen zu übertragen. Das Tiefpassfilter wird als Antialiasing-Filter bezeichnet [Broe99].

Da Tiefpassfilter nicht exakt ab einer bestimmten Frequenz alle höherliegenden Frequenzen abschneiden, sondern über einen Übergangsbereich verfügen, gilt für praktische Anwendungen, dass die Abtastfrequenz größer als das 2,2-fache der maximal zu übertragenden Frequenz sein muss. Daher beträgt die Abtastrate für Audio-CDs z.B. 44100 Hz.

2.3 Quantisierung

Im AD-Wandler wird die Spannung des Abtastwerts mit einer Vergleichsspannung über einen Komparator verglichen. Dieser liefert einen H-Pegel, wenn beide Spannungen übereinstimmen. Die Vergleichsspannung wird stufenweise erzeugt. Bei einer 8-bit-Wandlung besteht sie aus 256 Stufen, bei einer 16-bit-Wandlung aus 65536 Stufen. Die Verwendung dieser stufenweise erzeugten Menge an Werten wird Quantisierung genannt. In der Audiotechnik sind Quantisierungen mit 8, 16 und 24-bit üblich. Da das Signal durch die Quantisierung nicht mehr wertekontinuierlich ist, entstehen bei der Wiedergabe Signalverzerrungen, welche als Quantisierungsfehler bezeichnet werden. Diese werden als Rauschen hörbar und daher auch als **Quantisierungsrauschen** bezeichnet. Je niedriger die Quantisierung, desto größer wird die Abweichung vom Originalsignal, die zu einem Anstieg des Rauschanteils führt. Das Quantisierungsrauschen wird mit dem Signal-Rauschabstand SNR (signal to noise ratio) in Dezibel angegeben. Er ist von der Anzahl N der Bits abhängig [Broe99].

$$SNR = N \cdot 6,02dB + 1,76dB \tag{2.1}$$

Diese Formel kann als Faustformel vereinfacht werden: Mit jedem weiteren Bit nimmt der Signal-Rauschabstand um 6 dB zu und beträgt somit bei 8-bit-Aufnahmen ca. 50 dB, bei

16-bit-Aufnahmen ca. 98 dB und bei 24-bit-Aufnahmen ca. 146 dB. Welchen Einfluss die Reduktion einer Aufnahme um 1-bit hat, kann man sich anhand des folgenden Beispiels verdeutlichen: Angenommen eine Abtastfolge besteht aus den Ganzzahlwerten: 0, 4, 7, 8, 9, 11. Teilt man diese durch 2, was einer Reduktion um 1-bit entspricht, erhält man die Folge: 0, 2, 3, 4, 4, 5. Bei der anschließenden Multiplikation mit 2, was einer Erhöhung um 1 bit entspricht, lässt sich die ursprüngliche Folge nicht mehr gänzlich rekonstruieren. Alle ungeradzahligen Werte sind um 1 reduziert. Als Folge erhält man: 0, 4, 6, 8, 8, 10. Zwischen der ursprünglichen Abtastfolge und der rekonstruierten Folge besteht die Differenz: 0, 0, 1, 0, 1, 1. Diese Differenz entspricht dem durch die Reduktion um 1-bit entstandenen Quantisierungsrauschen.

Der Quantisierungsfehler lässt sich auch veranschaulichen, wenn man sich vorstellt, dass z.B. der Wertebereich von 1–12 auf einer vierstufigen Skala quantisiert werden soll, was einer 2-bit-Quantisierung entspricht. Dabei werden die Werte 1, 2, 3 der ersten Stufe, die Werte 4, 5, 6 der zweiten Stufe, die Werte 7, 8, 9 der dritten und die Werte 10, 11, 12 der vierten zugeordnet werden. Hierdurch entsteht eine Mehrdeutigkeit, die nicht mehr exakt rekonstruiert werden kann. Eine Quantisierungsstufe stellt also immer einen Wertebereich dar und keinen absolut zugeordneten Wert.

2.4 Speicherung von Audioinhalten

Die Speicherung digitaler Audioinhalte erfolgt in Dateien mit unterschiedlichen Formaten. In der Datei werden neben den Audiodaten, welche in komprimierter oder unkomprimierter Form vorliegen, auch mindestens die Formatdaten gespeichert. Auch Copyrightinformationen, Markierer, Kommentare und Gerätedaten lassen sich in diesen Dateien speichern. Die einzelnen Datentypen werden in Blöcken angeordnet. Derartige Dateien werden als **Containerdatei** bezeichnet. Auch andere Multimediadateien wie Text-, Bild- und Videodateien werden in diesem Format abgespeichert.

Um eine Audiodatei abspielen zu können, müssen dem Programm und auch der Soundkarte verschiedene Formatdaten übergeben werden, welche in Tabelle 2.1 aufgelistet sind.

Für die Speicherung unkomprimierter Audiodaten ist das **Audio Interchange File Format** (mit der Endung: *.aiff) und das **Windows Wave Format** (Endung: *.wav) sehr stark verbreitet. Das Audio Interchange File Format (AIFF) basiert auf dem 1985 von Firma Electronic Arts entwickelten Interchange File Format (IFF). Dieses Format zeichnet sich dadurch aus, das darin alle Datentypen in einer dreigliedrigen Struktur angelegt sind. Diese besteht erstens aus einem Bezeichner, der in Form eines Codes angibt, um welche Art von Datentyp es sich handelt.

TABELLE 2.1 Formatdaten und eine Auswahl möglicher Audioformate

Formatdaten	Mögliche Werte
Format	PCM, ADPCM, MP3, OGG, WMA
Abtastrate	8, 11,025, 16, 22,050, 32, 44,1, 48, 96, 192 kHz
Bitanzahl je Sample	8, 16, 24, 32, 64 bit
Anzahl der Kanäle	Mono, Stereo, Surround, Multichannel

Als zweites Element wird die Länge der gespeicherten oder zu übertragenden Daten angegeben, danach folgen die eigentlichen Daten. Diese Struktur wird **Type-Length-Value** (TLV) genannt und wird auch bei Übertragungen in Netzwerken als Protokoll genutzt. Die Anordnung der TLV-Blöcke in Form sogenannter Chunks kann reihenweise oder verschachtelt (d.h. die Blöcke enthalten selbst untergeordnete TLV-Blöcke als Subchunks) erfolgen. Die Bezeichner haben eine Länge von 4 Bytes und werden daher auch als **Four Character Code** (Abk.: FourCC) bezeichnet. Das AIFF-Format wurde von der Firma Apple als Audiodateiformat für den Macintosh auf der Grundlage des IFF-Formates weiterentwickelt. Es besteht aus einem FORM-Chunk, welcher als Subchunk einen Common Chunk mit der Kennung „COMM" für die Formatdaten und eine Sound Data Chunk mit der Kennung: „SSND" für die Audiodaten enthält [Bour96].

Mit der Entwicklung von Windows 3.1 wurde im Jahr 1991 das IFF-Format ebenfalls von den Firmen Microsoft und IBM übernommen und darauf aufbauend das **Resource Interchange File Format** (RIFF) entwickelt. Dieses Format findet Anwendung für Videodateien, Audiodateien und Midi-Dateien. Eine Windows-Wave-Datei enthält mindestens einen RIFF-Chunk und die beiden Subchunks „fmt " und „data". Der fmt-Chunk enthält als viertes Byte ein Leerzeichen und enthält die Formatdaten, der data-Chunk die Audiodaten [Demb96]. Eine Übersicht des Aufbaus zeigt Bild 2.4.

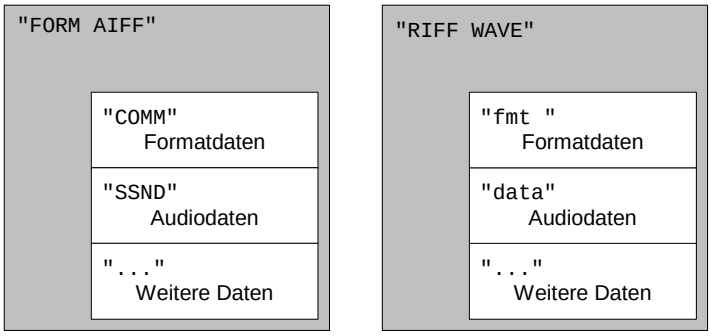

BILD 2.4 Aufbau von AIFF- und Windows-Wave-Dateien

Neben den unterschiedlichen Bezeichnungen der Chunks unterscheidet sich auch die Struktur der in den Chunks enthaltenen Daten. Ein ganz entscheidender Unterschied ist auch die Reihenfolge, in welcher Datentypen mit einer Länge von mehr als einem Byte gespeichert werden. Bei den im Macintosh verwendeteten Motorola-Prozessoren der Serie 68000 werden die Bytes mit den höherwertigen (most significant) Bits zuerst gespeichert und mit zunehmender Speicheradresse die Bytes der niederwertigeren (least significant) Bits.

Eine angenommene Hexadezimalzahl mit dem Wert 0x1A2B3C4D wird in der Datei oder im Speicher auch in der Reihenfolge: 1A 2B 3C 4D gespeichert. Dies erleichtert das Auslesen der Werte mit einem Hexeditor. Diese Byte-Reihenfolge wird **Big-Endian** genannt, das direkt übersetzt „Groß-Ender" bedeutet. Da die AIFF-Dateien zunächst überwiegend auf Macintosh-Rechnern verwendet wurden, werden die Datentypen short, int und long in dieser Byte-Reihenfolge gespeichert.