Christian Dorfmüller

„Interaktiver Onlinekurs zur Benutzung eines Audioequalizers“

DIPLOMARBEIT

HOCHSCHULE MITTWEIDA

UNIVERSITY OF APPLIED SCIENCES

Fakultät Medien

Mittweida, 2009

Christian Dorfmüller

„Interaktiver Onlinekurs zur Benutzung eines Audioequalizers“

eingereicht als

DIPLOMARBEIT
an der

HOCHSCHULE MITTWEIDA

UNIVERSITY OF APPLIED SCIENCES

Fakultät Medien

Mittweida, 2009

Erstprüfer: Prof. Dr. Michael Hösel
Zweitprüfer: Dipl. -Ing. Undine Schmalfuß

Vorgelegte Arbeit wurde verteidigt am:

Bibliografische Beschreibung:

Dorfmüller, Christian:

Interaktiver Onlinekurs zur Benutzung eines Audioequalizers.–2009. – 64 S.

Mittweida, Hochschule Mittweida (FH), Fakultät Medien, Diplomarbeit, 2009

Referat:

Ziel dieser Diplomarbeit ist es, einen Onlinekurs zum Thema 'Anwendung eines Audioequalizers' mit interaktiven Elementen zu erstellen. Dazu soll ein Audiotool implementiert werden, welches es ermöglicht die interaktiven Kursteile zu integrieren. Die Kursinhalte sollen aufbereitet und strukturiert werden. Mit Hilfe des Audiotools sollen entsprechend den Kursinhalten interaktive Übungen gestaltet werden.

Nach der Definition der Anforderungen an das Audiotool, werden die zur Umsetzung nötigen Grundlagen bezüglich der Programmierung des Audiotool zusammengetragen und erläutert, um die tatsächliche Implementierung vorzunehmen.

Anschließend werden die für die Benutzung eines Equalizers entscheidenden theoretischen Grundlagen aufbereitet und mit den, mit Hilfe des Audiotools realisierten interaktiven Elementen zu einem Online-Kurs zusammengefügt.

Abschließend werden die erstellten Inhalte für die Verwendung im Bildungsportal Sachsen vorbereitet und in einer Zusammenfassung wird ein Fazit der Arbeit gezogen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Eines der grundlegendsten Werkzeuge zur klanglichen Bearbeitung von Audiomaterial ist der Equalizer oder Entzerrer. Die Kenntnis der Arbeitsweise und der Umgang mit diesem Gerät ist für jeden, der mit der Bearbeitung von Audiomaterial in Berührung kommt von Bedeutung. Allerdings ist die Benutzung eines Equalizers nicht gerade einfach zu erklären, denn jenseits der Theorie erfordert der Einsatz eines solchen Gerätes eine Menge Erfahrung und ein gut geschultes Gehör. In der Fachliteratur ist die Anwendung eines Equalizer oft sehr theoretisch oder für ganz konkrete Anwendungsfälle beschrieben. Eher selten findet man Hinweise, welche Vorgehensweisen allgemein nützlich sein können. Zusätzlich liegt es in der Natur der Sache, dass es nur begrenzt möglich ist etwas klangliches textlich zu beschreiben.

Zur Erläuterung der Funktion eines Equalizers ist es also wünschenswert, die Theorie in Verbindung mit akustischem Material interaktiv zu präsentieren. Es bietet sich an, die interaktiven Möglichkeiten einer Online-Umgebung zu nutzen, um praktische Vorgehensweisen zu erläutern und es zu erleichtern die Zusammenhänge auch akustisch nachzuvollziehen. Damit wäre die Verbindung zwischen Theorie und tatsächlichem akustischen Ereignis einfacher zu erfassen und nachzuvollziehen.

Die Möglichkeiten in E-Learning Inhalte interaktive Elemente zu integrieren, waren noch vor wenigen Jahren auf die Möglichkeiten der Hypertext-Markuplanguage Html begrenzt. Das heißt, anklickbare Grafiken, sogenannte Image-Maps, Links oder Formulare ihrer normalen Funktion als Navigations- und Eingabeelemente zweckzuentfremden, war im Grunde die einzige Möglichkeit in Wechselwirkung mit dem User den Content pseudo-interaktiv zu präsentieren. Die Präsentation von Audiomaterial war auf das Abspielen von Audiodateien beschränkt. Außerdem gab es keine Möglichkeit die dargestellten Inhalte in Echtzeit zu verändern oder aufwändige Berechnungen durchzuführen, was für einen erhöhten Grad an sinnvoller Interaktivität nötig ist. Die Rechenleistung, die bei diesen klassischen Anwendungen serverseitig zur Verfügung stehen muss, hätte auch nicht ausgereicht, solche Inhalte einer größeren Zahl an Nutzern zur Verfügung zu stellen.

Allerdings hat sich auf Anwenderseite gerade die Rechenleistung mittlerweile so rasant entwickelt, dass eigentlich jeder durchschnittliche Arbeitsplatzrechner mit dem bloßen

browsen des Internet unterfordert ist und die vorhandenen Ressourcen meist ungenutzt bleiben. Unter diesen Gegebenheiten sind seit einiger Zeit auch Erweiterungen der Browser wie z.B. Flash oder Java für die Entwicklung ernstzunehmender Anwendungen geeignet. Diese Plattformen bieten Entwicklern mittlerweile umfangreiche Funktionen, die an die Möglichkeiten betriebssystemgebundener Desktopentwicklungsumgebungen heranreichen. Zusätzlich bieten sie den großen Vorteil, dass die Benutzung einer solchen Online-Anwendung für den User viel einfacher und für den Contentanbieter viel besser mit klassischem Webcontent zu verbinden ist. Außerdem ist eine Anwendung, die in einem Browser läuft, Betriebssystem unabhängig, das heißt der Entwicklungsaufwand verringert sich drastisch, ohne dass sich die potentielle Nutzerbasis verkleinert. Dazu kommt die Bequemlichkeit für die Nutzer, die eine solche Anwendung ohne vorherige Software-Installation direkt starten können.

Diese Möglichkeiten sollen genutzt werden, um eine Anwendung im Browser zu realisieren, die es ermöglicht den Onlinekurs mit interaktiven Elementen zu erweitern.

2 Anforderungen an das Audiotool

Das Audiotool soll als interaktives Element in ein E-Learning System eingebunden werden können und ohne Installation auf jedem Rechner im Browser lauffähig sein. Es soll zur Demonstration der Arbeitsweise eines Equalizers dienen. Dazu soll es möglich sein ein Audiosignal abzuspielen und in Echtzeit dessen Spektrum darzustellen. Weiterhin soll es für den Benutzer die Möglichkeit geben, mit einem Equalizer das abgespielte Signal klanglich in Echtzeit zu verändern. Zur Darstellung des Spektrum ist es notwendig eine Fouriertransformation zu implementieren. Der Equalizer soll über ein digitales Filter realisiert werden. Das Audiotool soll für verschiedene Fragestellungen anpassbar sein.

3 Wahl der Programmierschnittstelle

Die Möglichkeiten diese Funktionen zu realisieren bieten Java und Flash. Ein Vorteil von Flash ist seine mittlerweile große Verbreitung. Prinzipiell gibt es zwar für alle Betriebssysteme auch Java-Runtimes. Allerdings ist die einfache, im Browser integrierte Installation von Flash sehr viel benutzerfreundlicher als die Installation von Java. Außerdem ist beim am weitesten verbreiteten Betriebssystem Windows bereits eine Version von Flash vorinstalliert, so dass im unangenehmsten Fall ein Update durchgeführt werden muss.

Auf der Entwicklerseite bringt die Flashumgebung eine graphische Authoringoberfläche mit und stellt grundlegende Elemente für die Benutzeroberfläche bereit. Diese müssten in Java über externe Bibiliotheken eingebunden werden, was einen unnötigen Mehraufwand bedeuten würde.

Mit der Einführung von Actionscript 3 ist der objektorientierte Programmieransatz nun auch zufriedenstellend in Flash realisiert. Dadurch ist es möglich das Audiotool so zu gestalten, dass es für einzelne Aufgabenstellungen einfach angepasst werden kann.

4 Grundlagen zur Programmierung

4.1 Spektralanalyse

4.1.1 Fouriertransformation

Jean Baptist Fourier (1768-1830) erkannte, dass jedes kontinuierliche Signal als Summe bestimmter Sinuswellen dargestellt werden kann. Diese Entdeckung ermöglicht aus der Zeitdarstellung eines Signals seine spektrale Darstellung oder anders ausgedrückt die einzelnen im Signal vorhandenen Frequenzen zu berechnen.[1] Der mathematische Zusammenhang ist in Gleichung (1) dargestellt.

$$X(f) = \int_{-\infty}^{+\infty} x(t)\, e^{-2\pi i f t}\, dt \tag{1}$$

Wie man sieht ist die Fouriertransformation zunächst für Signale, die sich ins unendliche erstrecken, definiert.

4.1.2 Diskrete Fouriertransformation

Da Computer nur mit endlichen Werten rechnen können, ist es nötig diese unendlich langen Signale in eine diskrete Form zu bringen, um mit ihnen umgehen zu können. Dazu nimmt man sich einen Teil aus einer kontinuierlichen, unendlichen Funktion und verlängert diese in beide Richtungen entlang der x-Achse, indem man den Teil, der transformiert werden soll, periodisch bis nach unendlich wiederholt. So entsteht ein periodisches, diskretes Signal. Solche Signale können mit einer endlichen Zahl an Sinuswellen dargestellt werden.

Die Darstellung des Spektrums wird also jeweils für einen bestimmten Ausschnitt aus einem längeren aperiodischen Signal realisiert, den man sich periodisch wiederholt vorstellt.

1 Theorie zur Digitalen Signalverarbeitung und Regelung, S. 1

Die Transformation eines solchen Signals ist in Gleichung (2) dargestellt und heißt Diskrete Fourier-Transformation (DFT).[2]

$$X(n) = \frac{1}{N} \sum_{v=0}^{N-1} x(v) e^{-(2\pi ivn/N)} \quad (2)$$

Wobei hier N die Länge des transformierten Stückes in Samples ist.

4.1.3 Berechnung der DFT mittels Fast-Fourier-Transformation

Zur Berechnung der DFT gibt es mehrere Möglichkeiten. Die Methode, die sich für Echtzeitberechnung auf einem Computer eignet, ist die Fast-Fourier-Transformation, wie sie in Grundzügen bereits von Karl Friedrich Gauss (1777-1855) beschrieben wurde und 1965 von J.W. Cooley und J.W. Turkey wiederentdeckt und für die Nutzung in Computersystemen formuliert wurde.[3] Dabei werden die nötigen Rechenschritte vereinfacht, indem man Teilsummen bildet. Dieses Vorgehen macht eine Echtzeitberechnung auf heutiger Computerhardware möglich.

Der Fast-Fourier-Algorithmus beginnt damit, dass ein n Samples langes Signal im Zeitbereich solange aufgespalten wird, bis n einzelne Signale übrig sind, die jeweils nur aus einem Sample bestehen. Dabei werden bei jeder Aufspaltung jeweils die Werte an geraden Stellen im Signal und die Werte an ungeraden Stellen im Signal getrennt.

2 Smith, Steven W. : The Scientist and Engineer's Guide to Digital Signal Processing, S. 145

3 Smith, Steven W. : The Scientist and Engineer's Guide to Digital Signal Processing, S. 225

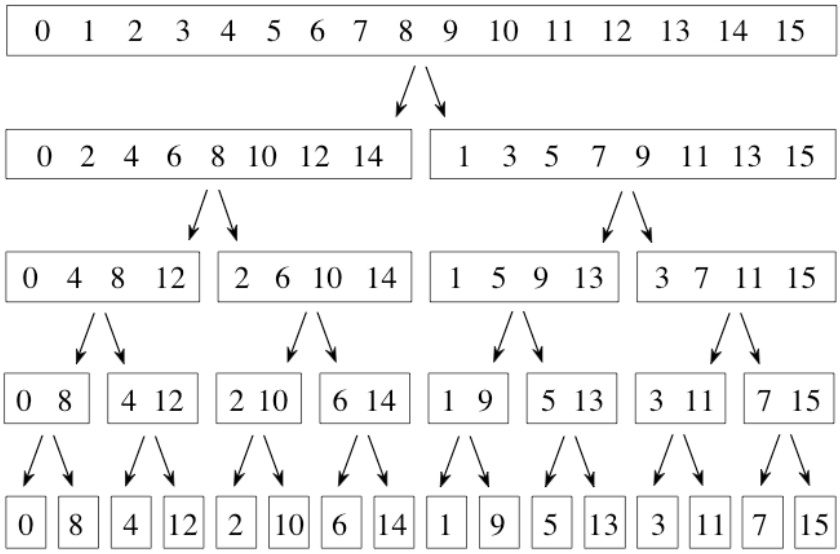

Abbildung 1: FFT-Aufspaltung

Die Anzahl der nötige Schritte beträgt also $\log_2 N$.

Betrachtet man nun den Vorgang mit bitweiser Darstellung der Samplepositionen, so erkennt man, dass eine Neuordnung der Samples stattgefunden hat, und zwar so, dass die Positionen, binär ausgedrückt, jeweils umgekehrt wurden.[4]

4 Smith, Steven W. : The Scientist and Engineer's Guide to Digital Signal Processing, S. 229

Beispiel der Bit-Umkehrung für N=16:

Normale Reihenfolge		**Sortierte Reihenfolge**	
Dezimal	**Binär**	**Dezimal**	**Binär**
0	0000	0	0000
1	0001	8	1000
2	0010	4	0100
3	0011	12	1100
4	0100	2	0010
5	0101	10	1010
6	0110	6	0100
7	0111	14	1110
8	1000	1	0001
9	1001	9	1001
10	1010	5	0101
11	1011	13	1101
12	1100	3	0011
13	1101	11	1011
14	1110	7	0111
15	1111	15	1111

Tabelle 1: FFT-Bit-Umkehrung

Als nächstes findet die Transformation in den Frequenzbereich statt. Ein einzelnes Sample in den Frequenzbereich zu transformieren ist denkbar einfach, denn das Frequenzspektrum eines Samples ist genau das Sample. Es ist also nur zu beachten, dass man sich ab jetzt nicht mehr im Zeitbereich, sondern im Frequenzbereich befindet.

Um beispielsweise das Spektrum eines N=16 langen Signals zu erhalten, müssen die Spektren der einzelnen Samples entsprechend rückwärts der vorangegangenen Aufspaltung kombiniert werden. Das heißt, dass aus den 16 Einzelspektren Schritt für Schritt erst 8, dann 4 und dann 2 Teilspektren errechnet werden müssen, um schließlich das endgültige Spektrum errechnen zu können.

Um diese Rückwärtssynthese auszuführen schaut man sich zunächst an, was im Zeitbereich passiert, wenn man zwei beispielsweise 4 Samples lange Signale zu einem 8 Samples langen Signal addiert.

Das erste Signal wird mit Nullen gefüllt, sodass abcd zu a0b0c0d0 wird. Das zweite Signal wird ebenfalls mit Nullen gefüllt und zusätzlich um 1 Sample verschoben. Das heißt, efgh wird zu 0e0g0f0h. Addiert man beide Signale, ergibt sich das 8 Samples lange Signal abcdefgh.

Im Frequenzbereich entspricht das einer Verdopplung des ersten Spektrum ABCD zu ABCDABCD. Das zweite Signal muss verdoppelt und um 1 Sample verschoben werden, was durch Multiplikation mit einem Sinusoid geschieht.[5]

Das 8-stellige Frequenzspektrum entsteht also aus den zwei 4-stelligen Spektren indem diese verdoppelt werden, das zweite Spektrum zusätzlich mit einem Sinusoid multipliziert, und anschliessend beide Spektren addiert werden.[6]

Das folgende Flussdiagramm stellt den Ablauf des Algorithmus dar. Die tatsächliche Konvertierung eines Samples in den Frequenzbereich taucht darin nicht auf, da für diesen Schritt, wie weiter oben beschrieben, gar keine Berechnung nötig ist.

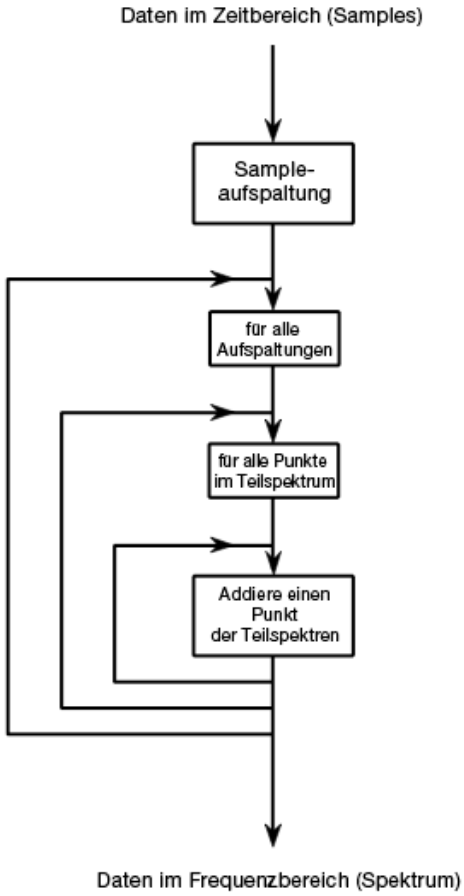

Abbildung 2: Ablauf der FFT-Berechnung

5 Smith, Steven W. : The Scientist and Engineer's Guide to Digital Signal Processing, S. 230

6 Smith, Steven W. : The Scientist and Engineer's Guide to Digital Signal Processing, S. 232

4.2 Digitale Filter

4.2.1 FIR-Filter

Es gibt zwei grundsätzliche Arten von Digitalen Filtern. Infinite-Impulse-Response-Filter (IIR) und Finite-Impulse-Response-Filter (FIR).

FIR-Filter sind mathematische Vorschriften zur Veränderung des Frequenzspektrums eines Signals und können mit analogen Bauteilen nicht realisiert werden.[7]

Im Grunde besteht ein FIR-Filter aus einer Verzögerungskette, deren Ausgang die Summe der einzelnen Kettenglieder ist, welche vor dem Summieren mit den entsprechenden Filterkoeffizienten multipliziert wurden. Mit der Länge der Verzögerungskette kann die Filterantwort eine steilere Charakteristik erhalten. FIR-Filter, bei denen die Länge der Verzögerungskette ungerade ist, weisen einen konstanten Phasengang auf.[8] Die Berechnung der Filterkoeffizienten ist recht kompliziert und kann nicht direkt von den gewünschten Eigenschaften des Filters wie z.B. der Ansatzfrequenz abgeleitet werden. Aufgrund des rückkoppelungslosen Designs sind FIR-Filter stabil, das heißt sie können nicht selbständig anfangen zu schwingen. Die benötigte Rechenleistung ist höher als bei IIR-Filtern.[9]

4.2.2 IIR-Filter

Diese Filter können als mathematische Repräsentationen analoger Filter verstanden werden.[10] Der Kern eines solchen Filters ist die Rückkoppelungsschleife des Filterausgangs auf den Eingang. Grundlegende Eigenschaften eines IIR-Filters sind ein glatter Amplitudengang und Phasenverschiebungen, entsprechend den analogen Filtern. Die Filterkoeffizienten können aus den vom User spezifizierten Daten für Ansatzfrequenz, Bandbreite und Verstärkung berechnet werden.

Durch Kombination mehrerer IIR-Filter kann eine steilere Filterkurve erreicht werden. Je nach Zahl der kombinierten Filter spricht man von einem Filter erster, zweiter, usw.

---

7 Rocchesso, Davide: Introduction to Sound Processing, S. 24

8 Smith, Steven W. : The Scientist and Engineer's Guide to Digital Signal Processing, S. 270

9 Hayes, Monson H. : Schaum's Outline of Theory and Problems of Digital Signal Processing, S. 359

10 Hayes, Monson H. : Schaum's Outline of Theory and Problems of Digital Signal Processing, S. 366

Art. Pro kombiniertem Filter kann die Amplitudenantwort um 6dB pro Oktave steigen.

Durch das Rückkopplungsdesign kann es zu Instabilitäten und Eigenschwingungen des Filters kommen. Besonders wenn extreme Filterantworten erreicht werden sollen, kann das zu unerwünschten Effekten führen. Generell brauchen IIR-Filter weniger Rechenleistung als FIR-Filter.[11]

4.2.3 Auswahl der Filterimplementation

Das Hauptkriterium zur Wahl des Filters ist die Rechenleistung. In einer Hochsprache wie Actionscript 3 ist die effizientere Filtervariante vorzuziehen, da man programmiertechnisch nur begrenzten Einfluss auf die Performance nehmen kann. Zum Beispiel ist es nicht möglich, die in aktuellen Prozessoren integrierten Floating-Point-Units anzusprechen, da AS3 dafür keine Schnittstelle bereitstellt.

4.2.4 Das Biquad-Filter als Peaking-EQ

Ein für die Anwendung als Equalizer oft genutzter analoger Filter ist das Biquad-Filter.

Für den speziellen Fall eines Peaking-EQ gilt die analoge Transferfunktion, auch Frequenzgang genannt, nach Gleichung (3).[12]

$$H(s) = \frac{(s^2 + s\frac{A}{Q} + 1)}{(s^2 + \frac{s}{(AQ)} + 1)} \tag{3}$$

Das ist, wie bei analogem Filterentwurf üblich, die Transferfunktion im Bildbereich der Laplace Transformation.

11 Smith, Steven W. : The Scientist and Engineer's Guide to Digital Signal Processing, S. 276

12 Udo Zölzer, Digital Audio Signal Processing, Second Edition, S. 125

Dieses Analoge Filter kann mit Hilfe der Bilinearen Transformation in ein digitales Filter übertragen werden.[13]

Man substituiert:

$$s = \frac{1}{\tan(\frac{w_0}{2})} \frac{1-\frac{1}{z}}{1+\frac{1}{z}} \tag{4}$$

und erhält eine Gleichung in der z-Ebene mit der Form

$$H(z) = \frac{b_0 + b_1(\frac{1}{z}) + b_2(\frac{1}{z^2})}{a_0 + a_1(\frac{1}{z}) + a_2(\frac{1}{z^2})} \tag{5}$$

was sozusagen die allgemeine Form eines digitalen Biquad-Filters zweiter Ordnung ist.[14]

Im konkreten Fall Peaking-EQ ergeben sich die einzelnen Filterkoeffizienten zu

$$b_0 = 1 + \alpha\ A \tag{6}$$

$$b_1 = -2\cos(w_0) \tag{7}$$

$$b_2 = 1 - \alpha\ A \tag{8}$$

$$a_0 = 1 + \frac{\alpha}{A} \tag{9}$$

$$a_1 = -2\cos(w_0) \tag{10}$$

$$a_2 = 1 - \frac{\alpha}{A} \tag{11}$$

13 Udo Zölzer, Digital Audio Signal Processing, Second Edition, S. 128

14 Udo Zölzer, Digital Audio Signal Processing, Second Edition, S. 128

wobei

$$\alpha = \frac{\sin(w_0)}{2Q} \tag{12}$$

$$w_0 = 2\pi \frac{f_0}{f_s} \tag{13}$$

$$A = \frac{10^{gaindB}}{40} \tag{14}$$

Über die Gleichungen (12), (13) und (14) hat man also Zugriff auf die vom Benutzer einzugebenden Filterparameter Q, f_s und A. Damit ist es möglich, Q-Faktor, Ansatzfrequenz und Verstärkung des Peaking-EQ variabel zu gestalten.

Die zur Transferfunktion (5) zugehörige Differentialgleichung lautet:

$$y(n) = \frac{b_0}{a_0} x(n) + \frac{b_1}{a_0} x(n-1) + \frac{b_2}{a_0} x(n-2) - \frac{a_1}{a_0} y(n-1) - \frac{a_2}{a_0} y(n-2) \tag{15}$$

Damit ist es möglich die gefilterten Samples konkret zu berechnen.

4.3 Die verwendete Flashtechnologie

4.3.1 Flash als Plattform für interaktive Inhalte

Flash als Plattform für interaktive Internetinhalte war in seiner Anfangsphase vor allem zur Erstellung von graphisch aufwändigen, interaktiven und dabei einigermaßen bandbreiteschonenden Inhalten interessant. Die Möglichkeiten zur Darstellung und zur Integration von interaktiven Elementen gingen und gehen viel weiter als die des Web-Grundgerüsts, der Hypertext-Markuplanguage (Html).

Auf Entwicklerseite steht die Flash-Anwendung zur Verfügung, die sowohl das erstellen graphischer Elemente als auch das Erstellen von Animationen erlaubt.

Die in der graphischen Umgebung erzeugten Elemente stehen zusätzlich als Objekte in der angegliederten Skriptsprache Actionskript zur Verfügung. Seit der Version 3.0 ist Actionscript durchgängig objektorientiert und die mitgelieferten Funktions-bibiliotheken sind mittlerweile so umfangreich, dass es möglich ist auf Flashbasis Anwendungen zu entwickeln, die vor der Einführung von AS 3.0 nur mit herkömmlichen Desktopentwicklungswerkzeugen realisierbar waren.[15]

4.3.2 Verbreitung

Mittlerweile ist die Verbreitung des Flash-Browserplugins, welches auf Anwenderseite benötigt wird, um Flash-Inhalte anzuzeigen, so weit fortgeschritten, dass man davon ausgehen kann, dass für die Flashplattform erstellte Inhalte auf fast allen internet-fähigen Personalcomputern unabhängig vom verwendeten Betriebssystem abrufbar sind.[16]

15 Moock, Colin: Essential ActionScript 3.0, S. XV

16 http://www.adobe.com/products/player_census/flashplayer/version_penetration.html

	Flashplayer 8	Flashplayer 9	Flashplayer 10
USA, Kanada	99,7%	99,6%	94,5%
England, Deutschland, Frankreich	99,3%	99,3%	91,7%
Neuseeland, Australien	99,2%	99,2%	92,6%
Japan	99,5%	99,0%	92,8%
Gesamt	99,7%	99,6%	93,5%

Tabelle 2: Verbreitung des Flashplayer

4.3.3 Der Aufbau einer Event-basierten, objektorientierten AS3-Anwendung

Das Eventmodell wurde mit der Einführung von AS3 erheblich verbessert, so dass es möglich ist, in Flash komplett eventgesteuerte Anwendungen unabhängig von der Animationstimeline zu erzeugen. Die Flashbühne an sich dient in diesem Fall nur noch zur Gestaltung der Benutzeroberfläche.

Ein Event wird durch bestimmte Programmereignisse wie z.B. Benutzereingaben oder das Ende eines Dateiladevorgangs ausgelöst. Die Anwendung weist einem Event eine Eventhandlerfunktion zu, die beim Auftreten des zugehörigen Events aufgerufen wird.[17]

Auch der objektorientierte Programmieransatz zieht sich seit der Einführung von AS3 durch die gesamte Flashumgebung und der Programmcode lässt sich sauber in einer Klassenstruktur organisieren.

Eine Flashanwendung instanziert bei ihrem Start eine Hauptklasse. Dabei wird der Konstruktor dieser Klasse aufgerufen. Dieser dient dazu, die Initialisierung des Programms vorzunehmen. In einer eventgesteuerten Anwendung können hier die Eventhandler deklariert und somit der weitere Ablauf des Programms bestimmt werden.[18]

Das Anwendungsfenster, welches man in der Authoring-Umgebung auch mit graphischen Werkzeugen bearbeiten kann, steht ebenfalls als Objekt zur Verfügung und kann über seine Eigenschaften und Methoden gesteuert werden.

17 Moock, Colin: Essential ActionScript 3.0, S. 202

18 Moock, Colin: Essential ActionScript 3.0, S. 8

4.3.4 Flash-Komponenten zur Realisierung der graphischen Benutzeroberfläche

Flash bringt in einer Komponenten-Bibiliothek standartisierte Elemente zum Erstellen einer graphischen Benutzteroberfläche mit.[19]

Die Komponenten können im Anwendungsfenster angeordnet werden und stehen dem Programm dann als Objekte zur Verfügung. Die Komponenten verfügen über alle nötigen Funktionen zum Auslesen von eingegebenen Daten oder zur Veränderung ihrer Eigenschaften und Zustände. In den Programmablauf werden diese Komponenten eingebunden, indem Eventhandler für durch die Komponenten ausgelöste Events deklariert werden.

4.3.5 Entwicklung der Audiofunktionen in Flash

Die Audiofunktionen waren in Flash lange Zeit sehr begrenzt. Das heißt, es gab zwar Funktionen zum Abspielen von Audiomaterial, die für viele Anwendungs-möglichkeiten, wie z.B. interaktive Web-Audioplayer ausreichend waren, jedoch konnten diese Funktionen nur mit konkreten, bereits vor der Laufzeit der Applikation feststehenden, Audiodateien umgehen.

Der Flash-Community gingen diese Funktionen nicht weit genug. Vor allem fehlte die Möglichkeit anstelle von Audiodateien konkrete Samples an die Flash-Sound-Schnittstelle zu übergeben. Mit einer solchen Funktion wäre es möglich Audio-anwendungen im Browser zu realisieren, die in ihrer Funktionalität bisherigen Desktopanwendungen das Wasser reichen.

Man muss dazu erwähnen, dass das bisherige Fehlen einer solchen Funktion wohl vor allem dadurch begründet war, dass die auf Anwenderseite zur Verfügung stehende Rechenleistung selbst bei der vorhandenen Möglichkeit in Echtzeit berechnete Samples durch Flash abspielen zu lassen, die realisierbaren Audioanwendungen sehr eingeschränkt hätte.

19 De Donatis, Antonio: Advanced ActionScript Components - Mastering the Flash Component Architecture, S. 65

Die von der Community gewünschten Verbesserungen wurden als Dynamic-Audio bezeichnet und einzelnen Mails verschiedener Entwickler an Adobe folgte eine Eintragung der Wünsche in den Adobe-Bug-Tracker, in dem Flashentwickler normalerweise auftretende Fehler des Flashplayers melden. Nachdem mehrere bekannte Webseiten, die sich mit dem Thema beschäftigten, dazu aufgerufen hatten sich der 'Adobe-make-some-noise' getauften Kampagne anzuschliessen und Adobe den 'Fehler' im Flashplayer zu melden, reagierte Adobe und stellte zunächst ohne öffentliche Ankündigung die Funktionen in einer Revision des Flashplayer 9 bereit.[20] Die einzige Dokumentation der Funktionen postete ein bei Adobe mit dem Thema beschäftigter Programmierer in seinem Blog.[21] Seit dem Release des Flashplayer 10 Ende 2008 dokumentiert Adobe die Funktionen auch offiziell.[22]

4.3.6 Die dynamischen Soundfunktionen im Flashplayer 10

Zur Implementation von Dynamic-Audio gehört zunächst ein neues Ereignis des bereits bekannten Flash-Soundobjekts. Dieses Ereignis 'sampleData' wird vom Flashplayer regelmäßig ausgelöst und wartet darauf, dass die Anwendung neue Samples für das Soundobjekt bereitstellt, welches über den Flashplayer abgespielt wird.

Die Übergabe geschieht durch Füllen des vordefinierten Array 'sampleData.data', welches eine Eigenschaft des 'sampleData'–Events selbst ist.

Dabei kann das Array eine Länge zwischen 512 und 8192 Samples haben. Die Länge des Arrays wirkt sich direkt auf die Latenz der Soundwiedergabe aus. Die Latenz ist die Zeit, die von der Übergabe das Samples bis zu dem Zeitpunkt an dem diese Samples akustisch zu hören sind, vergeht. Je kleiner das Array, desto kleiner ist die Latenz. Allerdings steigt mit kleinerem Array die Anzahl der nötigen Datenübergaben und damit die benötigte Rechenleistung.

Das Format der Sampledaten ist festeingestellt auf eine Samplerate von 44100Hz und 2 Kanäle für Stereowiedergabe. Ein Sample ist ein normalisierter 32bit Floating-Point Wert, das heißt der kleinste Wert ist 0 und der größte 1.[23]

---

20 http://www.make-some-noise.info/

21 http://www.kaourantin.net/2008/05/adobe-is-making-some-noise-part-2.html

22 http://help.adobe.com/en_US/Flash/10.0_Welcome/index.html

23 http://www.kaourantin.net/2008/05/adobe-is-making-some-noise-part-2.html

4.3.7 Die Opensource-Library 'org.audiofx.mp3'

Das Einladen von Benutzerdateien in eine Flashanwendung realisiert die in Flash bereits integrierte Funktion 'FileReference.load'. Diese Funktion stellt allerdings die geladene Datei als Byte-Strom dar. Im Zusammenhang mit dem Laden von heutzutage gebräuchlichen mp3-Audio-Dateien ist das ein Problem, denn die Bytes einer mp3-Datei sind keine abspielbaren Samples, sondern codierte Daten.

Diesen Bytestrom als Sounddaten zu nutzen macht folglich ohne vorherige Behandlung keinen Sinn.

Abhilfe schafft hier die quelloffene 'org.aufiofx.mp3'–Funktionslibrary, die über eine Loaderfunktion 'MP3FileReferenceLoader' die passenden Möglichkeiten bietet.

Dabei benutzt die Library interne Flashfunktionen um den mp3-Bytestrom so 'umzu-organisieren', dass er als normales Flashsoundobjekt benutzt werden kann und erspart das mp3-Dekodieren von Hand.[24]

24 http://www.flexiblefactory.co.uk/flexible/?p=46

5 Die Implementierung des Audiotool

5.1 Die Klassen zur Filterberechnung und Spektralanalyse

5.1.1 Die Klasse EQ.as

Die Klasse 'EQ.as' implementiert das digitale Filter zur Berechnung der Peaking-EQ Funktion für einen Audiokanal.

Die Funktionalität wird dabei über die Methode 'EQ.compute' aufgerufen. Die Methode erwartet bei ihrem Aufruf als Übergabewert ein einzelnes Sample. Dieses Sample durchläuft dann den Filteralgorithmus und der berechnete Wert wird ebenfalls als Sample zurückgegeben.

Dabei stellt die Klasse die Variablen 'f0', 'dbGain' und 'q' öffentlich, also für den Zugriff von aussen zur Verfügung. Über den Zugriff auf diese Variablen ist es möglich auf die Filterparameter Einfluss zu nehmen. Die Werte werden vom Hauptprogramm aus verändert, wenn der Benutzer in der Graphischen Oberfläche an den Equalizer-Bedienelementen andere Werte einstellt.

Dadurch, dass die Filterkoeefizienten bei jedem Durchlauf der Filterroutine neu berechnet werden, können Änderungen im Benutzerinterface sofort in die Berechnung einbezogen werden.

5.1.2 Die Klasse FFTransformer.as

Die Funktionalität der Klasse 'FFTransformer.as' steht, genau wie bei der Klasse 'EQ.as' einer Methode 'compute' bereit.

Die Funktion erwartet beim Aufruf als Parameter ein Array mit den Sampledaten und die Länge dieses Samplearrays in der Variablen 'fftsize'.

Zunächst ist es notwendig die Daten in die zur Berechnung notwendige Form zu bringen. Da die folgende Berechnungsroutine komplexe Zahlen erwartet, muss das

Array 'fftbuffer', auf dem später die Berechnung stattfindet, abwechselnd mit dem Realteil, welcher die Nutzdaten, also die konkreten Samples, enthält und dem komplexen Teil, der bei Audiodaten immer Null ist, gefüllt werden.[25]

Als nächstes wird die tatsächliche FFT Berechnung durchgeführt, welche allerdings zunächst keine anschaulichen Daten liefert. Erst nach der Umwandlung in die polare Darstellung kann also das 'mag_buffer'-Array gefüllt werden, welches schließlich die gewünschten Amplituden der einzelnen Frequenzen im Spektrum enthält. Dieses Array wird dann zurückgegeben.

5.1.3 Die Klasse Buffer_plotter.as

Die Klasse 'Buffer_plotter.as' gibt über die Methode 'draw_bufferplotter' Daten, die in zwei Arrays übergeben werden müssen, auf dem Bildschirm als Diagrammlinie aus. Dabei erwartet die Funktion im ersten Array 'buffer' die y-werte der zu zeichnenden Diagrammlinie und im Array 'freqbuffer' die zugehörigen x-werte.

Die verwendete Flashfunktion 'line.to' zeichnet dann beim Durchlaufen einer Schleife von der Länge des Buffers jeweils ein Linie zum nächsten Wertepaar.

Dabei ist zu beachten, dass die x-Werte vor der Ausgabe noch logarithmiert werden. Die logarithmische Darstellung ist für ein Frequenzspektrum geeignet, da sie besser mit der Empfindlichkeit des Gehöres übereinstimmt als eine lineare Darstellung.

25 Smith, Steven W. : The Scientist and Engineer's Guide to Digital Signal Processing, S. 227

5.2 Das Hauptprogramm, die Klasse Main.as

5.2.1 Die Funktion Buffer_playback

Diese Funktion ist der Kern des ablaufenden Programms. Sie wird immer dann aufgerufen, wenn Flash neue Samples für den Audiopuffer anfordert.

Zu Beginn eines Funktionsdurchlaufes werden zunächst aus einem bereits geladenen Soundobjekt, welches die zu spielende mp3-Datei enthält, die nächsten Samples extrahiert. Dabei wird eine, der Einstellung für den Audiopuffer entsprechende, Anzahl an Samples auf einmal übergeben. Gleichzeitig wird geprüft, ob das Dateiende erreicht wurde und falls nötig die Wiedergabe auf Anfang gesetzt.

Nun stehen die Samples in einem Array zur Verfügung. Dieses Array ist ein Byte-Array. Das heißt, ein Arrayelement speichert jeweils ein Byte. Für die Speicherung der Audiodaten bedeutet das, dass 1 Stereosample 8 Arrayelemente belegt, denn es handelt sich ja um zwei 32-bit Float-Zahlen. In den ersten 4 Bytes steht das Sample für den linken Kanal, in den zweiten 4 Byte das Sample für den rechten.

In einer Schleife, deren Durchlaufzahl der Anzahl Samples entspricht, werden nun mit jedem Durchlauf einzelne Samples aus diesem Array für den rechten und linken Kanal abgespalten.

Für diese einzelnen Samples wird jeweils die in EQ.as besprochenen 'compute' Funktion aufgerufen, welche das Filter berechnet.

Dann werden die gefilterten Samples wieder in ein Array geschrieben, um nach Ende der Schleife schließlich den gesamten berechneten Audiopuffer an Flash übergeben zu können.

Dieses Array wird erneut benutzt, diesmal um die Fouriertransformation für die enthaltenen Sampledaten zu berechnen, welche zur Anzeige des Spektrums benötigt wird.

5.2.2 Die Funktionen um auf Benutzer-Events zu reagieren

Die folgenden Bedienelemente wurden mit Hilfe von Flash-Components für die graphischen Benutzeroberfläche in der Flashentwicklungsumgebung angelegt und stehen damit in der Klasse Main.as zur Verfügung:

'eqctrl1.s1_f',

'eqctrl1.s2_gain',

'eqctrl1.s3_q',

'eqctrl1.bypassEQ_button',

'einaus_ctrl',

'antwort1_input',

'filectrl1.waveselectBox',

'filectrl1.fileup_button'.

Für die durch die Bedienelemente ausgelösten Events wurden Handler-Funktionen definiert, mit denen das Programm auf die Benutzereingaben reagiert und die entsprechenden Aktionen vornimmt.

Die Handler Funktionen der 'eqctrl.s'-Elemente setzen jeweils die Eigenschaften der Instanz der Klasse EQ.as, um die Koeeffizienten des Filter zu verändern.

Die 'eqctrl1.bypassEQ_button' und 'einaus_ctrl' Handler setzen jeweils eine Zustandsvariable. Die von 'eqctrl1.bypassEQ_button' gesetzte Variable wird vor dem Aufruf der Filterfunktion in buffer_playback geprüft. Die von 'einaus_ctrl' gesetzte Variable wird vor der Rückgabe der Samples an den Audiopuffer geprüft. Im Falle 'Aus' werden alle Samples auf den Wert 0 setzt.

'antwort1_input' dient zur Registrierung von Tastatureingaben und ruft außerdem die Antwortevaluierungsfunktion auf.

Die 'filectrl1.waveselectBox' und 'filectrl1.fileup_button' Handler rufen die 'sound_change'-Funktionen zum Ändern des momentan abgespielten Audiomaterials auf.

5.2.3 Sonstige Funktionen

Die 'Sound_change' Funktionen werden benötigt, um ein möglichst verzögerungs- und störgeräuschfreies Klangwechseln bei laufender Wiedergabe zu ermöglichen. Wechselt der Benutzter den Sound, wird dazu nicht direkt das Soundobjekt auf welches buffer_playback zugreift geändert, sondern zunächst ein anderes Soundobjekt mit der gewählten Datei gefüllt**.** Erst wenn Flash über ein Event meldet, dass der Ladevorgang abgeschlossen ist, wird dieses Soundobjekt in die Wiedergabe eingehängt.

Die 'timer' Funktionen dienen zum Aufruf der 'buffer_plotter.draw_buffer' Methode. Prinzipiell könnte man 'draw_buffer' auch für jeden abgearbeiteten Audiopuffer direkt in 'buffer_playback' aufrufen. Durch das Verwenden der Timerfunktion erreicht man, dass die Anzeige viel weniger Rechenleistung benötigt, da die Zeichenfunktion nur alle 100ms aufgerufen wird, was für die Darstellung völlig ausreichend ist.

5.2.4 Der Ablauf des Programms

Aufgrund der Event-basierten Funktionsweise und der objektorientierten Programmierung ist der Ablauf des Programms sehr übersichtlich.

Flash erzeugt eine Instanz der Klasse 'Main.as' und ruft anschließend deren Konstruktorfunktion 'Main()' auf.

In dieser Funktion werden den Eventhandlern ihre Handlerfunktionen zugewiesen und die Funktion zum Laden und Abspielen eines Sounds wird aufgerufen.

Damit läuft die Abspielroutine und Flash ruft regelmäßig 'buffer_playback' auf, um neue Samples zu erhalten.

Nun reagiert das Programm mit den entsprechenden Handler-Funktionen auf eintreffende Events.

6 Der Online Kurs

6.1 Lernziele

Die Lerninhalte des Onlinekurses sollen so gewählt werden, dass es einem Teilnehmer nach Durchführung des Kurses möglich ist einen Equalizer ergebnisorientiert einzusetzen.

Die nötigen Kenntnisse aus dem Bereich der Akustik und Audiotechnik sollen als Grundlage und Einführung in die Thematik behandelt werden. Nach der Einführung der theoretischen Grundbegriffe sollen die Kursteilnehmer in die Lage versetzt werden, physikalische Größen zu akustischen Reizen begrifflich zuzuordnen. Es soll also darum gehen, eine konkrete Vorstellung von beeinflussbaren Größen zu bekommen, um die gezielte Anwendung eines Equalizer zu ermöglichen.

Darauf aufbauend sollen praktische Übungen und Beispiele die Materie weiter verdeutlichen und der Gehörsinn trainiert werden. Dazu zählt das Zuordnen gehörter Klänge zu bestimmten Frequenzen.

Als Hilfsmittel soll der Umgang mit einem typischen Spektrumanalyser zur Messung und Veranschaulichung der durchgeführten Änderungen am Audiomaterial behandelt werden. Die Erkennung etwaiger Verfälschungen oder Unzulänglichkeiten des Signals durch die Abhörsituation und andere Faktoren und deren Minimierung sollen erläutert werden.

6.2 Strukturierung mittels Fragenkatalog

Um zu einer sinnvollen Struktur des Kurses zu gelangen soll der folgende Fragenkatalog für die Kursteilnehmer nach Durchführung des Kurses beantwortbar sein.

1. Was sind die von einem Equalizer beeinflussbaren Parameter eines akustischen Signals?
2. Wie ist ein Equalizer technisch realisiert?
3. Welche Bedienelemente eines Equalizers beeinflussen welche akustischen Parameter?
4. Wie sehen meine Veränderungen in einem Spektrumanalyser aus?
5. Wie finde ich die richtige Ansatzfrequenz?
6. Welche Frequenzbereiche klingen wie?
7. Wie soll mein Zielspektrum aussehen?
8. Welchen Einfluss hat mein Gehör auf das, was ich höre?
9. Welchen Einfluss hat meine Arbeitsumgebung auf das gehörte Signal?

Anhand dieser Fragen können nun die benötigten theoretischen Grundlagen aus Akustik und Audiotechnik zusammengetragen werden.

6.3 Die Kursinhalte

6.3.1 Schallausbreitung

Eine Schallquelle bringt die sie umgebenden Luftteilchen zum Schwingen. Dadurch entstehen Druckunterschiede, die sich auf die angrenzenden Teilchen übertragen. Die Luftteilchen schwingen dabei in Ausbreitungsrichtung um ihre Gleichgewichtslage herum. Es handelt sich bei einer Schallwelle, die sich in Luft ausbreitet, um eine longitudinale Dichtewelle.[26]

Die Ausbreitungsgeschwindigkeit c hängt vom Trägermedium und dessen Temperatur ab. In Luft beträgt die Schallgeschwindigkeit bei 20 Grad ca. 344 m/s. In Festkörpern ist die Schallgeschwindigkeit meistens sehr viel höher als in Luft.[27]

Die Zeit zwischen gleichen Schwingungszuständen eines bestimmten Luftteilchens bezeichnet man als Periodendauer T. Den Abstand zwischen Teilchen mit gleichem Schwingungszustand bezeichnet man als Wellenlänge λ. Die Anzahl der Periodendurchgänge pro Sekunde an einer Stelle bezeichnet man als Frequenz f = 1/T.[28]

6.3.2 Schalldruckpegel

Der absolute Schalldruckpegel wird als 20fach-logarithmiertes Verhältnis zu einem genormten Schalldruck $p_0 = 2 \cdot 10^{-5}$ Pa angegeben.

$$L_p = 20_{lg} \cdot \frac{p}{p_0} \tag{16}$$

Der genormte Druck p_0 entspricht etwa der Hörschwelle.

Bei 1000Hz liegt die Hörschwelle bei 4dB, die Schmerzgrenze bei ca. 130dB. Die empfundene Lautstärke durch ein menschliches Gehör beschreibt der Schalldruckpegel nur näherungsweise.[29]

---

26 Dickreiter, Handbuch der Tonstudiotechnik, Band 1, S. 2
27 Dickreiter, Handbuch der Tonstudiotechnik, Band 1, S. 3
28 Dickreiter, Handbuch der Tonstudiotechnik, Band 1, S. 4
29 Dickreiter, Handbuch der Tonstudiotechnik, Band 1, S. 9

6.3.3 Räume, Reflexion und Absorption

Sich ausbreitende Schallwellen werden frequenzselektiv an Gegenständen abgelenkt oder reflektiert. Außerdem werden Schallwellen beim Auftreffen auf Hindernisse absorbiert, d.h. der Schallwelle wird Energie entzogen.[30]

Reflexion findet statt, wenn die reflektierende Fläche die Größe mehrerer Wellenlängen der Schallwelle beträgt. Generell werden kleine Wellenlängen, also hohe Frequenzen oder Töne, bereits an kleinen Gegenständen reflektiert. Bei der Reflexion an geraden Flächen gilt für die Richtungsänderung Einfallswinkel = Ausfallswinkel. Reflexion an gekrümmten Flächen kann zu Schallstreuung oder Bündelung führen.[31]

Wird Schall in einem Raum erzeugt, treffen am Ohr, neben den direkt von der Schallquelle stammenden Wellen, durch Reflexion zeitlich verzögerte und in ihrer Frequenzzusammensetzung veränderte Schallwellen ein. Die Überlagerung aus allen diesen eintreffenden Wellen ergeben das tatsächlich wahrgenommene Signal. Entsprechend dem Verhältnis der Lautstärke des Originalsignals und dieses Nachhalls beeinflusst das Nachhallsignal mit seinem umgebungsspezifischen Frequenzgang und zeitlichem Abfall das Originalsignal.[32]

In Räumen mit parallelen Wänden kann es zudem zu stehenden Wellen kommen. Diese entstehen durch wiederholte Reflexion zwischen gegenüberliegenden Wänden. Überlagern sich reflektierte und originale Schallwellen, kommt es zu Verstärkung oder Abschwächung bestimmter Frequenzen. Für Frequenzen, deren halbe Wellenlänge oder ein Vielfaches davon genau in den Abstand zwischen zwei parallele Wände passt, führt diese Überlagerung zu festen Stellen im Raum, an denen diese Frequenz lauter oder leiser zu hören ist.[33]

Schallabsorbtion findet durch Reibung der schwingenden Luft mit ebenfalls schwingenden Materialien statt. Das bedeutet, dass es sozusagen den Luftteilchen schwieriger gemacht wird zu schwingen. Das passiert zum Beispiel in Faserstoffen, in denen zwar Luft noch schwingt, aber durch die Reibung am Material und dessen Anregung zum Schwingen Energie verliert. Ein Schallabsorber wirkt je nach seiner

30 Dickreiter, Handbuch der Tonstudiotechnik, Band 1, S. 15
31 Dickreiter, Handbuch der Tonstudiotechnik, Band 1, S. 11
32 Dickreiter, Handbuch der Tonstudiotechnik, Band 1, S. 28
33 Dickreiter, Handbuch der Tonstudiotechnik, Band 1, S. 12

Positionierung, Dimensionierung und Beschaffenheit auf einen bestimmten Frequenzbereich.[34]

In einem Raum entsteht also, je nach Größe und Beschaffenheit, ein Nachhall, der sich mit dem Originalsignal überlagert. Außerdem können Reflexionen zu Verstärkungen und Abschwächungen bestimmter Frequenzbereiche an bestimmten Stellen in einem Raum führen. Im Allgemeinen werden diese Beeinflussungen das von einer Quelle abgestrahlte Signal umso stärker verändern je lauter das Originalsignal ist.

6.3.4 Das Gehör

Am Ohr ankommende Schallwellen regen das Trommelfell zum Mitschwingen an. Durch die Gehörknöchel Hammer, Amboss und Steigbügel werden die vom Trommelfell registrierten Druckschwankungen transformiert und auf das ovale Fenster, den Eingang zur Schnecke, übertragen. Die Schnecke ist mit Lymphflüssigkeit gefüllt und in ihr befindet sich eine schwingende Trennwand auf der das Cortiorgan sitzt. Dieses Organ besteht aus etwa 25000 Sinneszellen, welche die mechanische Information in elektrische Impulse umwandeln. Diese wird über den Hörnerv an das Gehirn geleitet.[35]

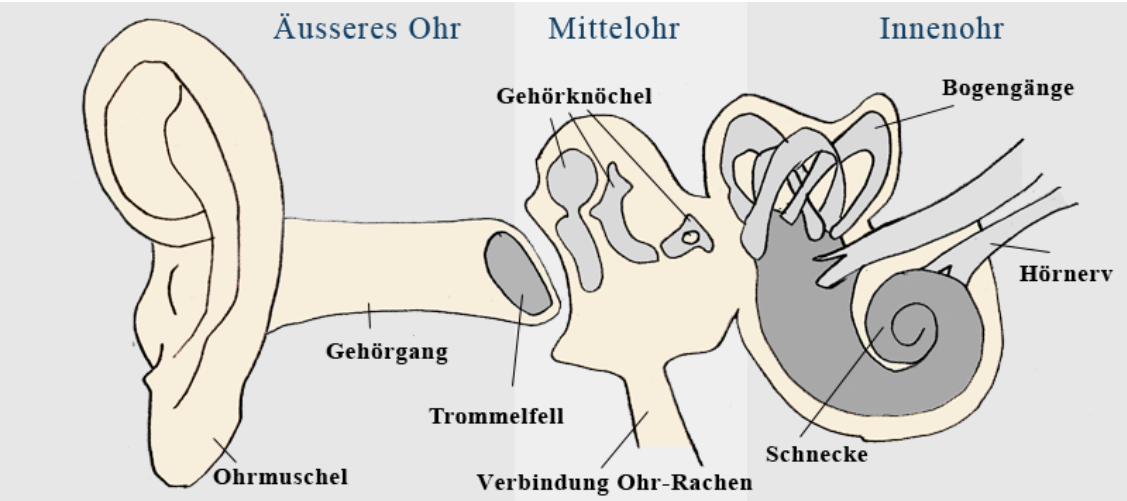

Abbildung 3: das menschliche Ohr

34 Dickreiter, Handbuch der Tonstudiotechnik, Band 1, S. 17

35 Dickreiter, Handbuch der Tonstudiotechnik, Band 1, S. 107

Die höchste vom menschlichen Ohr wahrnehmbare Frequenz nennt man Hörgrenze. Diese liegt je nach Individuum zwischen 16000-20000Hz. Mit steigendem Alter nimmt die Hörgrenze im Allgemeinen ab und kann auch bei etwas tieferen Frequenzen liegen.[36]

6.3.5 Lautstärke

Als Hörschwelle bezeichnet man die Lautstärke, bei der ein Geräusch gerade noch wahrgenommen wird. Als Schmerzschwelle bezeichnet man die Lautstärke, ab der Schmerzempfindung eintritt.

Das menschliche Gehör empfindet physikalisch gleiche Lautstärken verschiedener Frequenzen unterschiedlich laut. Zusätzlich ändert sich dieser Zusammenhang mit der Abhörlautstärke. Dieser Zusammenhang ist in den Kurven gleicher Lautstärke dargestellt.[37]

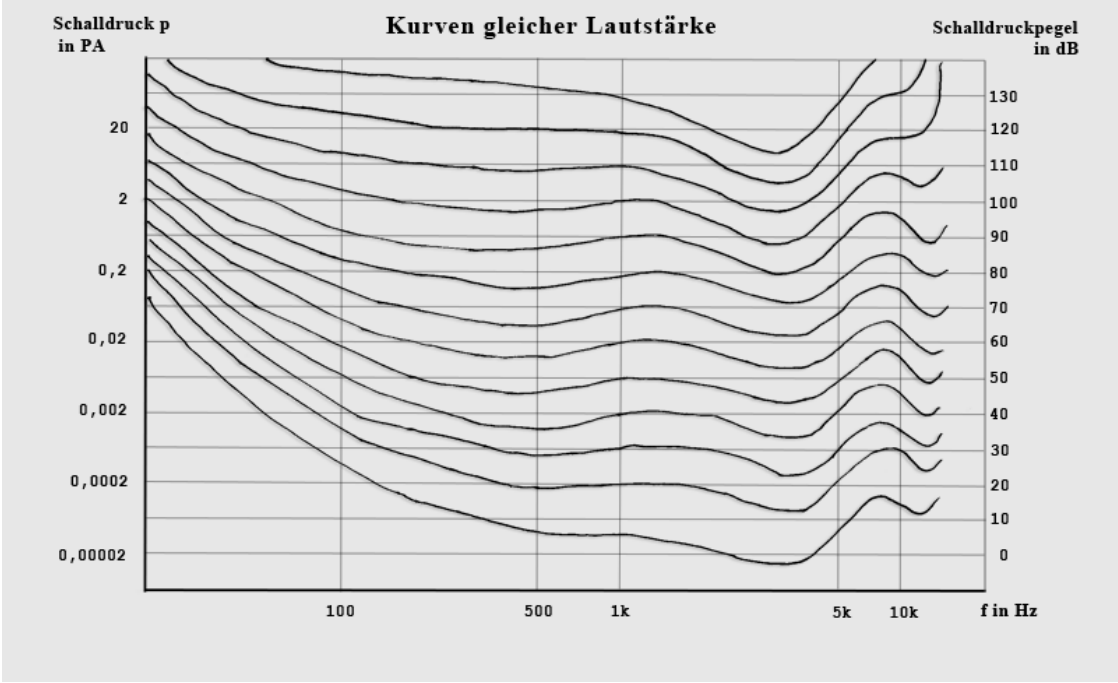

Abbildung 4: Kurven gleicher Lautstärke

36 Dickreiter, Handbuch der Tonstudiotechnik, Band 1, S. 110

37 Dickreiter, Handbuch der Tonstudiotechnik, Band 1, S. 111

Das bedeutet also, das bei einem Rausch-Signal, welches in allen Frequenzbereichen gleiche Pegel aufweist, ein Zuhörer tiefe Frequenzen oder Töne als leiser wahrnehmen wird als mittige Töne. Frequenzen im Bereich von 4000Hz dagegen wird er als lauter wahrnehmen.

Es verhält sich außerdem so, dass ein als doppelt so laut empfundenes Geräusch nicht den doppelten Schalldruckpegel hat. Um das menschliche Lautstärkeempfinden besser abzubilden, führte man die Einheit Sone ein. Dabei bedeutet ein Lautstärkepegel von 1 Sone einen Schalldruckpegel von 40dB bei 1000Hz. Die als doppelt so laut empfundene Lautstärke liegt bei einem Schalldruckpegel von 50dB und das entspricht dann 2 Sone.[38]

Eine weitere Eigenschaft des Gehöres ist es, sich einem Schallpegel anzupassen. Das bedeutet, das z.B. ein permanent vorhandenes, diffuses Hintergrundgeräusch nach einer gewissen Gewöhnungszeit als 'Ruhe' wahrgenommen wird. Andersherum ist der Effekt bei aufmerksamem, leisem Musikhören zu beobachten. Anfangs kaum mehr als ein Gesamtgeräusch, nimmt das Gehör mit zunehmender Dauer viel mehr Details wahr und die anfangs eingestellte Lautstärke kann noch mehr zurückgefahren werden, ohne dass weniger Klanginformation wahrgenommen wird.

6.3.6 Verdeckung

Mit Verdeckung bezeichnet man die Tatsache, dass ein auf das Ohr wirkender Reiz die Empfindlichkeit für andere ankommende Reize absenkt. Dabei gilt allgemein, dass hohe Frequenzen naheliegende, tiefere Frequenzen verdecken. Tiefere Frequenzen hingegen können hohe Frequenzen verdecken, wenn ihre Intensität sehr viel größer als die der hohen Frequenzen ist.[39] Die allseits bekannte mp3-Kompression von Audiodaten beruht auf diesem Effekt. Zur Datenreduktion werden im Prinzip verdeckte und damit unhörbare Anteile des Audiosignals gelöscht und damit Information eingespart.[40]

38 Dickreiter, Handbuch der Tonstudiotechnik, Band 1, S. 112

39 Dickreiter, Handbuch der Tonstudiotechnik, Band 1, S. 113

40 Yiteng Huang und Jacob Benesty, Audio Signal Processing for Next-Generation Multimedia-Communication-Systems, S. 298

6.3.7 Tonhöhen und Frequenzen

Im Zusammenhang mit der Bearbeitung von Klangmaterial ist es oft hilfreich den Zusammenhang zwischen musikalischen Tönen oder Tonhöhen und ihren Frequenzen zu kennen.

Musikalische Tonhöhen sind zunächst nicht an bestimmte physikalische Frequenzen gebunden, sondern als Verhältnisse definiert. Eine Oktave z.B. entspricht einer Verdopplung der Frequenz.

Kennt man allerdings die Grundstimmung, an der sich alle anderen Töne orientieren, lassen sich die den Tonhöhen zugehörigen Frequenzen berechnen. Meistens wird als Bezugston der Kammerton A auf 440Hz festgelegt.[41]

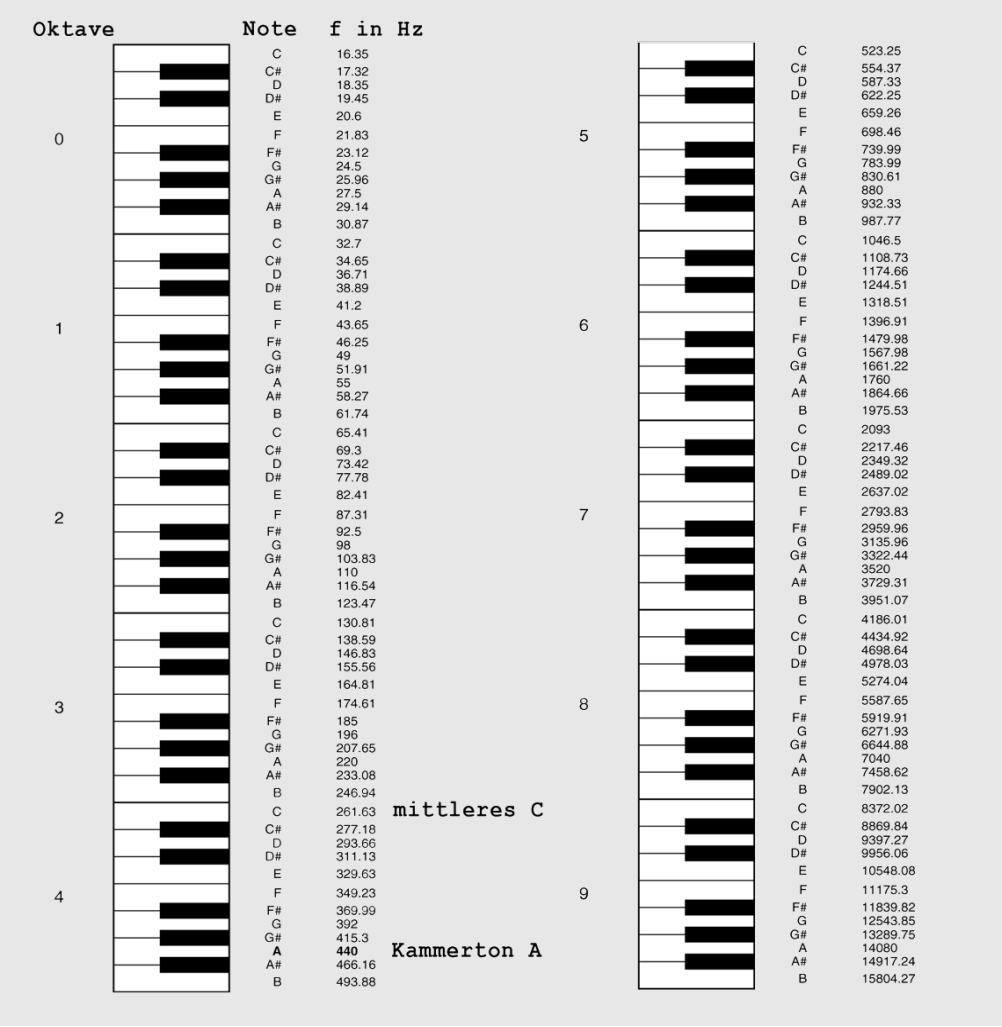

Abbildung 5: Frequenzen und Tonhöhen

41 Dickreiter, Handbuch der Tonstudiotechnik, Band 1, S. 78

6.3.8 Darstellung von Audiosignalen

Die einfachste und häufigste Darstellung von Audiomaterial ist die Auftragung des Pegels über der Zeitachse. Diese Darstellung wird in allen gängigen Audio-anwendungen benutzt und bietet genügend Informationen, um Audiomaterial zu schneiden und zu arrangieren. Um die klangliche Dimension graphisch darzustellen ist es nötig, zu einer Darstellung im Frequenzbereich zu gelangen. Dazu muss die zeitkontinuierliche Pegelfunktion nach Fourier in ihr Spektrum transformiert werden.[42] Die spektrale Darstellung gibt Auskunft über die zu einem Zeitpunkt vorhandenen Lautstärken einzelner Frequenzen im Gesamtsignal.

6.3.9 Was tut der Equalizer und wozu ist das gut

Mit einem Equalizer, auch Entzerrer genannt, lässt sich das Spektrum eines Audiosignals bearbeiten.[43] Das heißt die Lautstärke der im Audiosignal vorhandenen unterschiedlichen Frequenzanteile kann gezielt beeinflusst werden. An einer Consumer-Stereoanlage findet sich oft ein Equalizer in Form eines Höhen- und Tiefenreglers.

Mit einem Equalizer lassen sich ungewollte Nebengeräusche wie Netzbrummen absenken. Außerdem ist es möglich den Frequenzgang unausgewogener Aufnahmen nachträglich zu glätten. Diese können durch Nichtbeachten von Raumresonanzen bei der Mikrofonierung entstanden sein.

Oft ist es zum Beispiel auch sinnvoll das Frequenzspektrum für ein bestimmtes Wiedergabesystem zu optimieren. Soll beispielsweise eine Aufnahme in einer Telefonwarteschleife abgespielt werden, macht es Sinn Frequenzen unterhalb von ca. 300 Hz und oberhalb von ca. 7khz stark abzusenken, da sie von einem Telefonhörer sowieso nicht wiedergegeben werden können und höchstens zu ungewollten Verzerrungen führen.

Generell kann man sagen, dass der Frequenzgang einer Aufnahme immer möglichst glatt, also ohne große Lücken im Spektrum sein sollte. Damit erreicht man, dass das Ergebnis auf möglichst vielen Wiedergabesystemen möglichst ähnlich klingt.

42 Hayes, Monson H. : Schaum's Outline of Theory and Problems of Digital Signal Processing, S. 55

43 Dickreiter, Handbuch der Tonstudiotechnik, Band 1, S. 351

6.3.10 Wann kommt es überhaupt zu Veränderungen des Spektrums

Betrachtet man den Weg von der Signalaufzeichnung bis zur Wiedergabe entstehen an mehreren Punkten Beeinflussungen des Spektrum, die eine nachträgliche Bearbeitung erfordern können. Die Frequenzgänge bzw. Übertragungsfunktionen der einzelnen Punkte in der Verarbeitungskette zu kennen, ist wichtig, um das gehörte Ergebnis richtig bewerten zu können und eventuelle Veränderungen und Anpassungen vornehmen zu können.

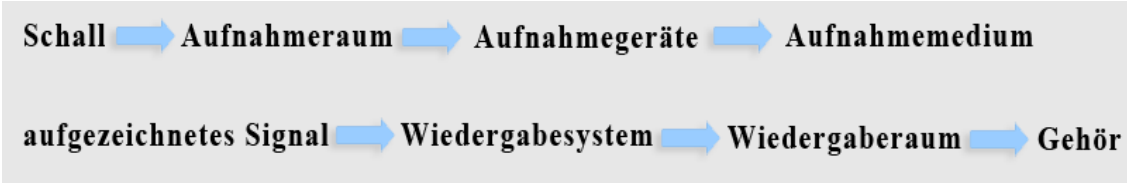

Abbildung 6: Beeinflussungen des Spektrum

Zunächst breitet sich der Schall im Aufnahmeraum aus. Hierbei entstehen Resonanzen und Nachhall, die sich störend auswirken können. Als nächstes passiert das Signal die Aufnahmegeräte und wird auf dem Aufnahmemedium gespeichert. Mikrofone beispielsweise haben oft drastisch frequenzgangverändernde Eigenschaften. Aufzeichnungsgeräte selbst fügen dem Signal heutzutage meistens vor allem Rauschen hinzu. Durch die Eigenschaften des Aufnahmemedium oder -Formats kann der Frequenzbereich eines Signals weiter beschnitten werden, obwohl das seit einigen Jahren für die Praxis eher unrelevant ist.

Beim Abspielen eines Signales wird dieses erneut durch diese Beeinflussungskette verändert, diesmal in umgekehrter Reihenfolge. Das heißt, Eigenschaften des Wiedergabegerätes, besonders der Boxen und des Abhörraumes, verändern das tatsächlich gehörte Signal.

6.3.11 Arten und einstellbare Parameter von Equalizern

Ein EQ hat mindestens ein sogenanntes Band. Ein Band besteht aus einem Filter mit einer bestimmten Ansatzfrequenz und einer bestimmte Flankensteilheit, auch Q-Faktor oder Güte genannt. Die Güte regelt wie breit das beeinflusste Frequenzband ist. Über die Verstärkung schließlich lässt sich die Lautstärke eines Bandes anheben oder absenken.

Equalizer lassen sich in zwei Kategorien unterteilen. Bei Parametrischen Equalizern sind die Ansatzfrequenzen und Q-Faktoren der Bänder fest voreingestellt. Bei den vorhanden Bändern lassen sich dann die jeweiligen Verstärkungen einstellen. Bei Graphischen Equalizern hingegen sind Ansatzfrequenz und der Q-Faktor der Bänder frei wählbar.

6.3.12 Finden der Ansatzfrequenz

Um einen ausgewogenen Frequenzgang zu erzeugen muss man erkennen, welche Frequenzbereiche angehoben oder abgesenkt werden sollten. Beim Hören eines Klanggemisches ist es allerdings oft nicht einfach zu erkennen, welche Frequenzen genau über- oder unterbetont sind. Abgesehen von Erfahrung und einer Abhöranlage mit möglichst linearem Frequenzgang hilft einem die im folgenden erläuterte Vorgehensweise zu guten Ergebnissen zu gelangen.

Die Methode beruht auf der Eigenschaft eines Equalizers, das bei einem hohen Q-Faktor, also einem engen Filter und gleichzeitig hoher Verstärkung, etwaige Ungleichmässigkeiten im Spektrum überproportional hervortreten. Das heißt, fährt man die Frequenzen mit hoher Verstärkung und kleinem Q-Faktor durch, treten vorhandene Ungleichmässigkeiten noch stärker hervor und können dadurch besser erhört werden. Heutzutage oft verwendete digitale Equalizer haben außerdem die eigentlich ungewollte Eigenschaft bei hohen Verstärkungen enger Frequenzbänder einen pfeifenden Ton zu erzeugen. Fährt man mit einem solchen Equalizer durch den Frequenzbereich, deutet das deutliche lauter werden dieses Pfeifen eine Überbetonung im zugehörigen Frequenzbereich an. Die zunächst extreme Einstellung des EQ erleichtert also das Auffinden der zu bearbeitenden Frequenzbereiche.

6.3.13 Frequenzbänder nach Gehörempfinden

Bestimmte Teile des Spektrums rufen oft ähnliche klangliche Assoziationen hervor, unabhängig vom bearbeiteten Audiomaterial.

Frequenz	Beschreibung	Überbetonung
16-60	Subbass, Gefühl von Druck	Klingt dumpf
60-250	Perkussiver und tonaler bass, basisbildende Noten der Rhytmusgruppe	Klingt matschig
300-1000	Enthält die musikalische Tonhöhen und Harmonieinformation	Klingt nach Blech
1000-4000	Sprachverständlichkeitsbereich	Verstärkt Näheeindruck, ermüdend
4000-9000	Klarheit von Stimme und Instrumenten	erzeugt schneidende Schärfe
6000-16000	Brillianz	wird hauchig, seidig

Tabelle 3: Frequenzen nach Hörempfinden

6.3.14 Weitere Tips zum Umgang mit einem Equalizer

Die folgende Liste führt Vorgehensweisen auf, die beim Bearbeiten und Beurteilen von Audiomaterial und beim Benutzen eines Equalizers in der Praxis hilfreich sein können:

- Indem die Abhörlautstärke öfter verändert wird, lässt es sich vermeiden, dass das Ohr sich an eine Lautstärke gewöhnt und ermüdet.
- Beim Anheben oder Absenken von Frequenzen muss darauf geachtet werden, wie sich die Lautstärkeverhältnisse der einzelnen Teilklänge ändern.
- Durch Absenkung störender Frequenzen kann der Klang verbessert werden. Eine Veränderung des Klanges kann durch Frequenzanhebung herbeigeführt werden.

- Es ist von Vorteil, wenn das Ergebnis auf möglichst vielen verschiedenen Wiedergabesystemen angehört wird.
- Das Anhören professioneller Produktionen, auch während der Bearbeitung von anderem Audiomaterial, kann wichtige Anhaltspunkte geben.
- Um zu vermeiden, dass die durch den Equalizer mitbedingte Veränderung der Lautstärke das Urteilsvermögen beeinflusst, sollte man darauf achten, nach der Einstellung des Equalizers die Lautstärke anzupassen, sodass beim Betätigen des Bypass-Schalters kein Lautstärkeunterschied zu hören ist.
- Wenn der Equalizer mit geschlossenen Augen bedient wird, hilft dies, sich auf den Klang zu konzentrieren. Außerdem wird das Gehirn nicht durch visuellen Input abgelenkt.
- Beim Zusammenmischen verschiedener Signale sollte darauf geachtet werden, dass diese sich nicht überdecken. Dies gelingt, indem ein Signal immer wieder auf 'solo' geschaltet und überprüft wird, ob es sich dabei im Klang verändert. Für den Fall, dass es von einem anderen Signal verdeckt wird, sollten beide Signale mit einem EQ bearbeitet werden.

6.4 Die interaktiven Elemente

Da die Materie theoretisch nur zum Teil fassbar ist, soll mit interaktiven, visuell-akustischen Elementen ein praktischer Bezug zur behandelten Theorie hergestellt werden.

Das im ersten Teil der Arbeit erstellte Audiotool soll nun für konkrete Aufgabenstellungen im Zusammenhang mit der Kursthematik angepasst werden. Es sollen zu den Punkten der Theorie, die sich dafür eignen, entsprechende Aufgaben erstellt werden. Je nachdem, was durch die Aufgabe gefordert ist, wird das benötigte Audiomaterial aufbereitet und mit der Flash-Authoring-Umgebung ein spezielles, auf diese Aufgabe abgestimmtes Audiotool erstellt.

6.4.1 Das verwendete Audiomaterial

Das Audiomaterial unterteilt sich in Testsignale und Musikstücke. Dabei wurden die Testsignale, also die Sinusschwingungen, Sweeps und das Rauschen mit einer Testversion der Software DSSF3[44] erstellt. Die musikalischen Inhalte stammen aus von mir durchgeführten Produktionen aus den Bereichen der Pop- und Filmmusik und können dadurch lizenz- und GEMA-gebührenfrei verwendet werden.

44 http://www.ymec.com/products/dssf3e/

6.4.2 Darstellung Audiosignale

Diese Aufgabe dient zur Heranführung an den Analyser und die spektrale Darstellung eines Audiosignals. Einige typische Testsiggnale sowie Sprache und Musik können gewählt werden und ihre zugehörigen, typischen Frequenzverläufe können betrachtet werden.

6.4.2.1 Audiotool

Die Equalizer-Steuerung wurde ausgeblendet, um die Oberfläche möglichst einfach zu halten.

Als Klangelemente stehen ein Musik- und ein Stimmsignal, sowie Sinus und Sägezahnwellen zur Verfügung. Zusätzlich gibt es die Möglichkeit über das Datei-Menü eigene mp3-Dateien spektral darzustellen.

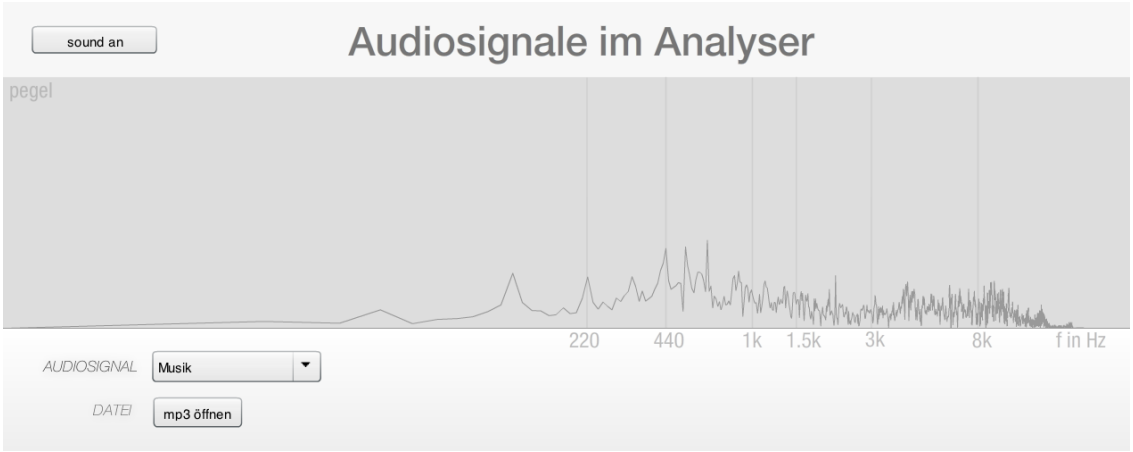

Abbildung 7: Audiotool, Darstellung Signale

6.4.2.2 Aufgabentext

„Schauen Sie sich die Frequenzzusammensetzung verschiedener Audiosignale im Analyser an. Mit dem Dropdownmenü 'Audiosignale' können Sie die verschiedenen Signale anwählen. Um eine Ihrer eigenen mp3-Dateien in den Analyser zu laden klicken Sie 'mp3 öffnen'.“

6.4.3 Hörgrenze

Über ein Sweepsignal, welches langsam aus dem hörbaren Bereich fährt, soll eine Vorstellung davon vermittelt werden, wie die physikalischen Frequenzwerte sich tatsächlich anhören.

6.4.3.1 Audiotool

Um eine genauere Zuordnung der Frequenzen zu ermöglichen, ist die Frequenzskala besser aufgelöst.

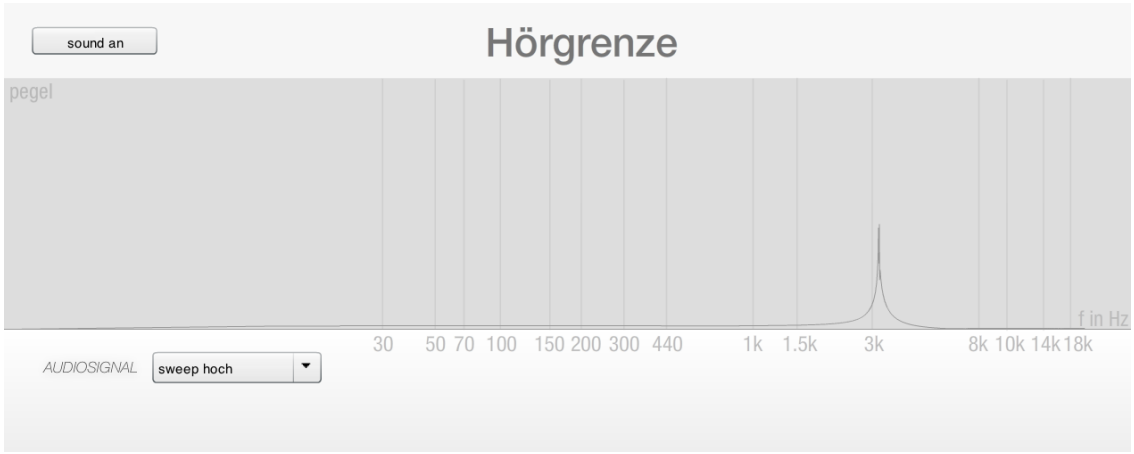

Abbildung 8: Audiotool, Hörgrenze

6.4.3.2 Aufgabentext

„Überprüfen Sie ihre Hörgrenze. Schalten Sie dazu den Sound an und hören sich den ansteigenden Ton an. Im Analyser können sie ablesen welche Frequenz der Ton im Moment hat. Die Frequenz, ab der Sie den Ton nicht mehr wahrnehmen, ist Ihre Hörgrenze.“

6.4.4 Überprüfen der Fähigkeiten des Wiedergabesystems

Diese Übung dient allgemein zur Heranführung an die Thematiken Abhörsysysteme und Beeinflussung des Spektrum bei der Audiobearbeitung.

Dazu ist es möglich sich mit Hilfe eines tieffrequenten Sweepsignales einen Eindruck von der unteren Grenzfrequenz des Abhörsystems zu verschaffen.

6.4.4.1 Audiotool

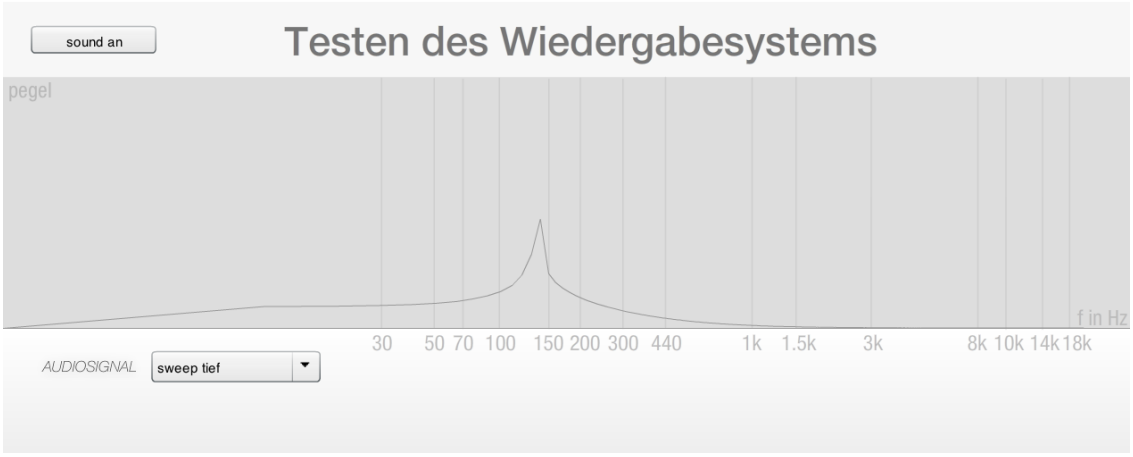

Abbildung 9: Audiotool, Wiedergabesystem

6.4.4.2 Aufgabentext

„Um einen ungefähren Eindruck der Fähigkeiten ihrer Abhöranlage zu bekommen und sich mit dem Spektrumanalyser vertraut zu machen, können Sie nun die tiefste Frequenz, die von ihrem Wiedergabesystem noch dargestellt werden kann, ermitteln. Man benutzt dazu ein sogenanntes Sweepsignal. Das ist ein Sinussignal, welches kontinuierlich durch alle Frequenzen fährt.

Schalten Sie zunächst das Audiosignal an und hören sich den Sinussweep an. Der Sweep beginnt bei 20Hz und endet bei 800 Hz. Sie werden feststellen, dass Sie erst ab einer bestimmten Tonhöhe wirklich etwas hören. Diese Frequenz ist dann die tiefste von ihrem System darstellbare Frequenz. Lesen Sie diese Frequenz am Analyser ab. Benutzen Sie einen Subwoofer sollten Sie schon ab ca. 30Hz einen Ton vernehmen. Benutzen Sie beispielsweise herkömmliche Laptopboxen sollten Sie erst ab ca. 100-200Hz etwas hören. Mit dem Audiosignal-Dropdownmenü können Sie auch einen festen Sinuston auswählen und so ihre Lautsprecher überprüfen.“

6.4.5 1-Band-Graphic-EQ

Dieses Audiotool ermöglicht es, sich mit den Bedienelementen eines Equalizers vertraut zu machen und die gewählten Einstellungen optisch am Analyser und auch akustisch nachzuvollziehen.

6.4.5.1 Audiotool

Als Klangquellen wurde Musik verschiedener populärer Stile gewählt. Die Auflösung der Frequenzskala wurde verringert, da es zunächst wichtig ist die mit dem EQ bewirkten Veränderungen einem groben Frequenzraster zuzuordnen. Genauere Angaben würden das erschweren und ein sehr gut trainiertes Gehör erfordern.

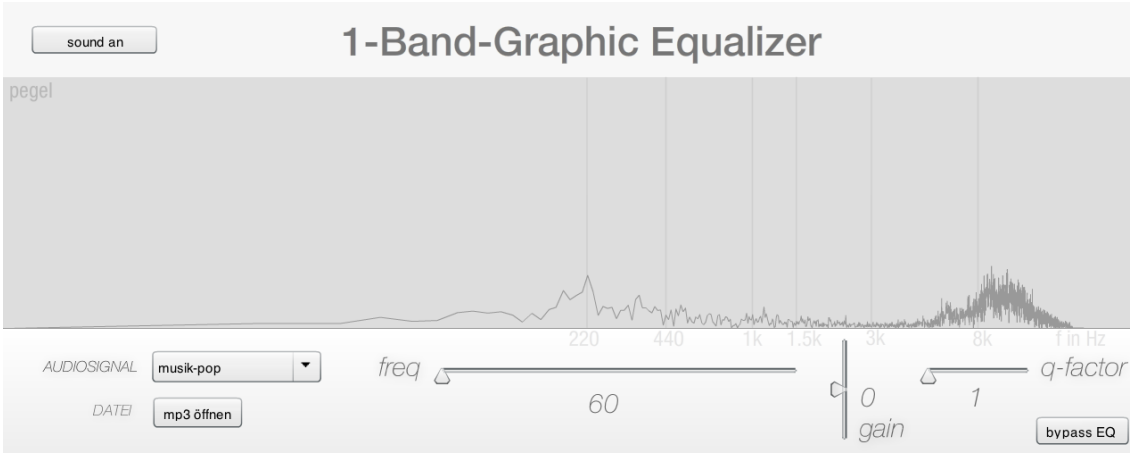

Abbildung 10: Audiotool, 1-Band-EQ

6.4.5.2 Aufgabentext

„Stellen Sie am 1-Band-EQ einen Q-Faktor von 5 und eine Verstärkung von 10 ein. Fahren Sie dann mit dem Frequenzregler durch das Spektrum und hören Sie sich die Veränderungen an! Versuchen Sie andere Veränderungen! Versuchen Sie ein anderes Audiosignal!“

6.4.6 Finden der Ansatzfrequenz

Nach Einführung der grundlegenden Paramter dient dieses Tool zur Demonstration einer wichtigen praktischen Methode im Umgang mit einem Equalizer.

6.4.6.1 Audiotool

In dieses Audiotool wurde zusätzlich die Möglichkeit einer Lösungsüberprüfung integriert.

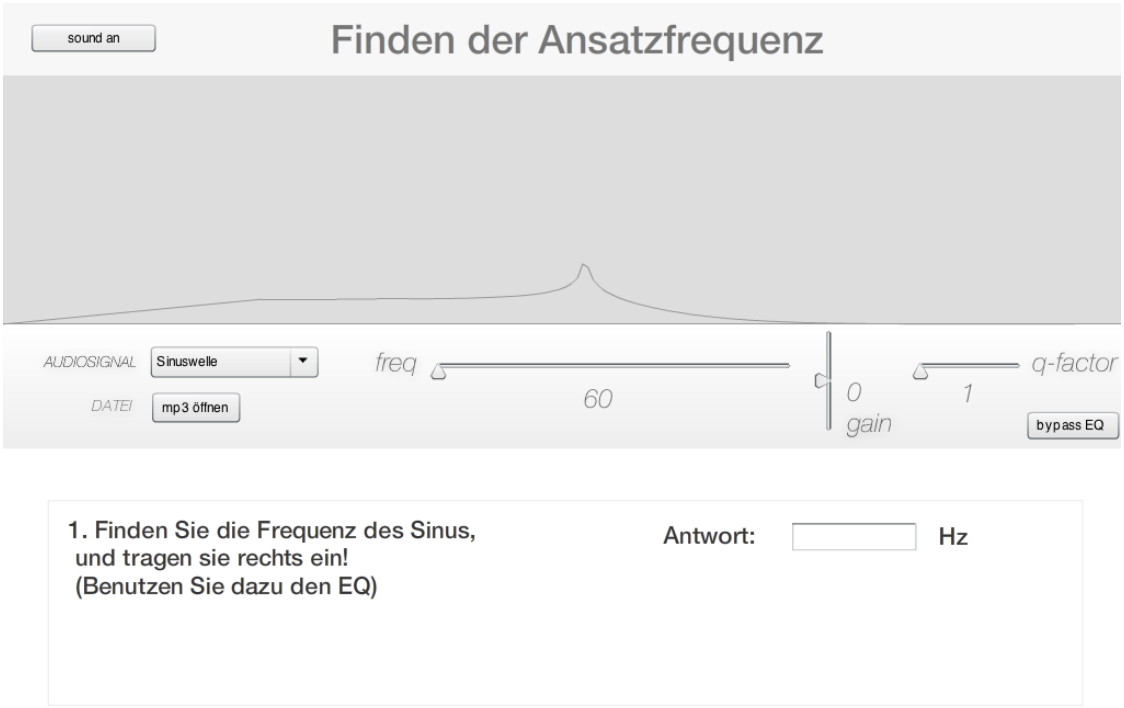

Abbildung 11: Audiotool, Ansatzfrequenz

6.4.6.2 Aufgabentext

„Im Beispiel hören Sie zunächst einen Sinuston, also ein maximal unausgewogenes Frequenzspektrum. Stellen Sie eine hohe Filtergüte und eine Verstärkung von 10 ein. Fahren Sie dann mit dem Frequenzregler von den tiefen zu den hohen Frequenzen.

Bei einer bestimmten Frequenz können Sie hören und im Spektrumanalyser beobachten, dass Sie den Ton deutlich beeinflussen. Diese Frequenz ist die Frequenz des Sinustones. Diese Methode kann man auch benutzen, um in einem komplexen Signal über- oder unterbetonte Frequenzbereiche aufzufinden.“

6.4.7 Hörempfinden

Diese Übung dient zum weiteren Gehörtraining und zur Anwendung der Methode zum Finden der Ansatzfrequenz.

6.4.7.1 Audiotool

Das integrierte Klangmaterial wurde so vorbereitet, dass typische Frequenzbereiche überbetont sind.

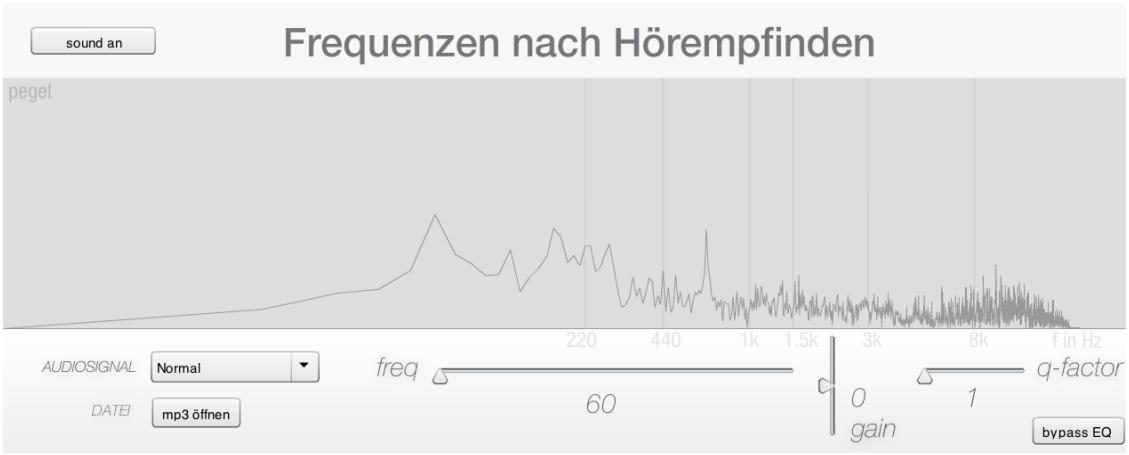

Abbildung 12: Audiotool, Frequenzen nach Hörempfinden

6.4.7.2 Aufgabentext

„Hören Sie sich zunächst für eine Weile das Musikstück an. Wählen Sie dann eine der Versionen mit einer Überbetonung aus und versuchen Sie die qualitative Beschreibung aus der Tabelle zuzuordnen.

Nun können Sie versuchen den Equalizer zu benutzen, um die Überbetonung zu glätten. Dazu können Sie die oben vorgestellte Methode zum Finden der richtigen Ansatzfrequenz benutzen. Stellen Sie dazu eine Verstärkung von 10 und einen Q-Faktor von 1 oder 2 ein. Fahren Sie dann mit dem Frequenzregler durch das Spektrum. An der Stelle im Spektrum, an der eine Überbetonung vorliegt, sollten sie eine deutliche Klangveränderung wahrnehmen. Senken Sie nun an dieser Stelle die Verstärkung soweit ab, dass die Überbetonung nicht mehr negativ auffällt.

Vergleichen Sie bearbeitetes und unbearbeitetes Signal durch Klicken des Bypass EQ-Button. Hören Sie sich erneut das Originalsignal ('Normal') an.“

7 Der Rollout des Kurses für das Bildungsportal

Im Bildungsportal wird zur Präsentation der E-Learning Inhalte ein Content-Management-System benutzt, welches es auf sehr angenehme Weise ermöglicht neue Kurse einzupflegen. Die Inhalte müssen dazu in einer Word-Datei vorliegen, die mit den Word-eigenen Werkzeugen gegliedert ist. Die Struktur wird über Formatvorlagen für Überschriften hergestellt.

Eine Thema kann in bis zu 10 Unterpunkte aufgeteilt werden. Die Unterpunkte müssen dabei im Word-Dokument als Überschrift der Ebene 1 markiert werden.

Die textlich erscheinende Überschrift eines Unterpunktes muss als Überschrift der Ebene 1.1 markiert sein. Ab der Ebene 1.1.1 dient die Word-Gliederung zur übersichtlicheren Darstellung des Textes.

Der Equalizer-Kurs soll in ein umfassenderes E-Learning Angebot zu Audiotechnik- und Audiobearbeitung einfließen. Durch die bestehende Gliederung dieses Angebotes in grundlegende und erweiterte Lernabschnitte war es sinnvoll auch das Material zum Equalizerkurs entsprechend in 2 Module aufzuteilen. Im ersten Modul wurden die Grundlagen zusammengefasst, die auch für viele weitere Anwendungsfälle zum Verständnis wichtig sind. Erst im zweiten Teil wird dann auf den speziellen Fall Equalizer eingegangen.

Zusätzlich sind in den Strukturvorgaben des Content-Management-Systems weitere Einheiten wie z.B. 'Anmerkung', 'Definition', 'Zitat', 'Verweis', 'Hinweis' oder 'Schlussfolgerung' vorgesehen, welche jedoch im Zusammenhang mit den erstellten Modulen nicht benötigt wurden.

Zum Lern-Material gehörende Grafiken und das Audiotool müssen separat zu den gegliederten Word-Dokumenten vorhanden sein. Sie werden mit Html-Tags in den Content eingebunden.

8 Zusammenfassung

Ziel dieser Diplomarbeit war es, die zur Benutzung eines Audioequalizers nötigen Kenntnisse in einem Online-Kurs aufzubereiten und mit Hilfe eines dafür entworfenen interaktiven Lerntool auch die praktische und akustische Nachvollziehbarkeit der theoretischen Lerninhalte zu ermöglichen.

Das interaktive Tool stellt Möglichkeiten zur Verfügung, Audiosignale abzuspielen und in einem Echtzeitspektrumanalyser darzustellen. Die Audiosignale können über ein digitales Filter, dass einem analogen, graphischen Equalizer nachempfunden ist, klanglich in Echtzeit bearbeitet werden.

Das in Adobe Flash / Actionscript 3.0 implementierte Audiotool kann von jedem Rechner mit Internetanschluss aus genutzt werden. Zur Benutzung ist keine Softwareinstallation notwendig.

An dieser Stelle ist es erwähnenswert, dass die Realisierung eines Spektrumanalysers und eines digitalen Filters, die Echtzeitfunktionalitäten unter einer betriebssystem-unabhängigen Oberfläche zur Verfügung stellen, erst seit einigen Monaten möglich ist. Bei der mittlerweile an jedem Computerarbeitsplatz verfügbaren Rechenleistung und dem Entwicklungsstand der Adobe-Flash-Umgebung kann man davon ausgehen, dass in Zukunft sehr viele Softwareapplikationen über ein Browser-Interface zur Verfügung stehen werden, deren Funktionalität an die von herkömmlichen Desktopanwendungen heranreicht. Vielleicht steht uns der Anfang der 1990er Jahre postulierte NetPC nun langsam bevor.

Auf der Grundlage des Audiotools wurden mehrere interaktive Übungen in den Kursablauf eingebunden.

Mit Hilfe der interaktiven Teile konnte das theoretisch nur begrenzt vermittelbare Gebiet auch auf der ihm eigenen akustischen Ebene transportiert werden, was dazu beiträgt den Verständnis- und Lernprozess zu beschleunigen.

Abschließend wurden die Kursmaterialien für die Einpflege in das Content-Management-System des Bildungsportals Sachsen vorbereitet.

9 Anhang

9.1 Quelltext der Klasse EQ.as

```
Package {

public class EQ {

private var xh0:Number;
private var xh1:Number;
private var xh2:Number;

private var yh0:Number;
private var yh1:Number;
private var yh2:Number;

private var a0,a1,a2:Number;
private var b0,b1,b2:Number;

public var f0,dBgain,q:Number;


      public function EQ() {

            xh0=0;
            xh1=0;
            xh2=0;

            yh0=0;
            yh1=0;
            yh2=0;

            f0=1000;
            dBgain=0;
            q=1;
      }

      public function compute(sample:Number):Number {

            var a:Number;
            var w0,cosw0,sinw0,alfa,Fs:Number;
            Fs=44100;

            a = Math.pow(10,(dBgain/40));
            w0=2*Math.PI*f0/Fs;

            cosw0=Math.cos(w0);
            sinw0=Math.sin(w0);
            alfa = sinw0/(2*q);

            b0=1+alfa*a;
            b1=-2*cosw0;
            b2 = 1-alfa*a;
            a0=1+alfa/a;
            a1=-2*cosw0;
            a2=1-alfa/a;
```

```
            xh0=sample;

            yh0 = (b0/a0)*xh0 + (b1/a0)*xh1 + (b2/a0)*xh2 - (a1/a0)*yh1 //
            - (a2/a0)*yh2;//biquad filter

            xh2=xh1;//rekursiv buffer updaten
            xh1 = xh0;
            yh2=yh1;
            yh1=yh0;

            return (yh1);
      }
}
}
```

9.2 Quelltext der Klasse FFTransformer.as

```
Package {

import flash.utils.ByteArray;

public class FFTransformer {

private var sample_buffer_REX:Array = new Array();
private var sample_buffer_IMX:Array = new Array();
private var sample_buffer_REX_polar:Array = new Array();

var wr,wi,arg,temp,tr,ti,ur,ui:Number;
var i,bitm,j,lee,lee2,k,logN:int;

var fftBuffer:Array = new Array();
var mag_buffer:Array = new Array();
var freq_buffer:Array = new Array();

var p1,p2,p1r,p1i,p2r,p2i:int;


      public function FFTransformer() {
            }

      public function compute(sample_buffer:Array,fftsize:int):Array {

            for (i=0; i<fftsize; i++) {
                        fftBuffer[(i*2)]=sample_buffer[i];
                        fftBuffer[(i*2+1)]=0;
            }

            logN =(Math.log(fftsize)/Math.log(2.)+.5);
            for (i = 2; i < 2*fftsize-2; i += 2) {

                  for (bitm = 2, j = 0; bitm < 2*fftsize; bitm <<= 1) {
                              if (i&bitm) {
                                    j++;
                              }
                        j<<=1;
                  }

                  if (i<j) {
```

```
                        p1=i;
                        p2=j;
                        temp=fftBuffer[p1];
                        fftBuffer[p1++]=fftBuffer[p2];
                        fftBuffer[p2++]=temp;
                        temp=fftBuffer[p1];
                        fftBuffer[p1]=fftBuffer[p2];
                        fftBuffer[p2]=temp;

                  }
            }

            for (k = 0, lee = 2; k < logN; k++) {

                  lee<<=1;
                  lee2=lee>>1;
                  ur=1.0;
                  ui=0.0;
                  arg = Math.PI / (lee2>>1);
                  wr=Math.cos(arg);
                  wi = (-1)*Math.sin(arg);

                  for (j = 0; j < lee2; j += 2) {

                        p1r=j;
                        p1i=p1r+1;
                        p2r=p1r+lee2;
                        p2i=p2r+1;

                        for (i = j; i < 2*fftsize; i += lee) {

                              tr=fftBuffer[p2r]*ur-fftBuffer[p2i]*ui;
                              ti=fftBuffer[p2r]*ui+fftBuffer[p2i]*ur;
                              fftBuffer[p2r]=fftBuffer[p1r]-tr;
                              fftBuffer[p2i]=fftBuffer[p1i]-ti;
                              fftBuffer[p1r]+=tr;
                              fftBuffer[p1i]+=ti;
                              p1r+=lee;
                              p1i+=lee;
                              p2r+=lee;
                              p2i+=lee;
                        }

                        tr=ur*wr-ui*wi;
                        ui=ur*wi+ui*wr;
                        ur=tr;
                  }
            }

            for (k=0; k<fftsize; k++) {
                  mag_buffer[k] = Math.LOG10E*20*Math.log( 2* (Math.sqrt(Math.//
                  pow (fftBuffer[k*2],2)) + Math.pow(fftBuffer[k*2+1],2))+1);
                  freq_buffer[k]=k*44100/fftsize;
            }
            return (mag_buffer);
      }
}
}
```

9.3 Quelltext der Klasse Buffer_plotter.as

```
package {

import flash.display.*;
import flash.utils.*;
import flash.text.*;

public class Buffer_plotter extends Sprite {

      var i:int;
      var l,r:Number;
      private var _tf:TextField;
      var xscale:int;
      var yscale:int;

      public function Buffer_plotter() {

            graphics.lineStyle(1, 0x000000, 1, false, LineScaleMode.NONE,//
            CapsStyle.SQUARE);
            this.graphics.moveTo(0,260);
            xscale=200;
            yscale=1;
      }

      public function draw_buffer(buffer:Array,freqbuffer:Array,len:int) {

            this.graphics.clear();
            graphics.lineStyle(1, 0x999999, 1, false, LineScaleMode.NONE,//
            CapsStyle.SQUARE);
            this.graphics.moveTo(0,260);
            this.graphics.lineTo(900,260);
            this.graphics.moveTo(0,260);

            for (i=1; i<len; i++) {
                  this.graphics.lineTo(xscale*Math.LOG10E* Math.log(Math.round//
                  (freqbuffer[i])),260-buffer[i]*yscale);
            }
      }
}
}
```

9.4 Quelltext der Klasse Main.as

```
package {

import flash.display.*;
import flash.events.*;
import flash.utils.*;
import flash.media.Sound;
import flash.net.URLRequest;
import fl.controls.ComboBox;
import fl.controls.Slider;
import fl.events.*;

import org.audiofx.mp3.MP3FileReferenceLoader;
import org.audiofx.mp3.MP3SoundEvent;
import flash.net.FileFilter;
import flash.net.FileReference;

public class Main extends MovieClip {

private var loader:MP3FileReferenceLoader;
private var fileReference:FileReference;

public static const BUFFER_SIZE:int=4096;
public static const SAMPLE_RATE:int=44100;

private const _out_snd:Sound=new Sound;
private var _loop_snd:Sound=new Sound;
private var sound_array:Array=new Array;

private var _samples:ByteArray;
private var _samples_processed:ByteArray;

private var _FFTransformer:FFTransformer=new FFTransformer;
private var _buffer_plotter:Buffer_plotter=new Buffer_plotter;

private const _eqL:EQ=new EQ;
private const _eqR:EQ=new EQ;
private var bypassEQ:Boolean;
private var einaus:Boolean;

var floatsample_array_l:Array=new Array;
var floatsample_array_r:Array=new Array;
var freqbuffer:Array=new Array;

var l:Number,r:Number,k:int,i:int;


      public function Main():void {

            addChild(_buffer_plotter);
            Draw_timer();
            filectrl1.waveselectBox.addEventListener(Event.CHANGE,wavechosen);
            eqctrl1.s1_f.addEventListener(SliderEvent.CHANGE,slider_freq);
            eqctrl1.s2_gain.addEventListener(SliderEvent.CHANGE,slider_gain);
            eqctrl1.s3_q.addEventListener(SliderEvent.CHANGE,slider_q);
            filectrl1.fileup_button.addEventListener(MouseEvent.CLICK,//
            fileup_click);
```

```
        eqctrl1.bypassEQ_button.addEventListener(MouseEvent.CLICK,//
        bypassEQ_click);
        bypassEQ=false;
        einaus_ctrl.addEventListener(MouseEvent.CLICK, einaus_click);
        einaus=false;
        antwort1_input.addEventListener(KeyboardEvent.KEY_DOWN, keyHandler);
        loadSound('sinuswelle.mp3');
        loader=new MP3FileReferenceLoader();
        loader.addEventListener(MP3SoundEvent.COMPLETE,//
        mp3LoaderCompleteHandler);
        fileReference=new FileReference();
        fileReference.addEventListener(Event.SELECT,
        //fileReferenceSelectHandler);
    }

    public function wavechosen(e:Event):void {
        pb_sound_change(filectrl1.waveselectBox.selectedItem.data);
    }

    public function Draw_timer() {

        var myTimer:Timer=new Timer(100,0);
        myTimer.addEventListener("timer",timerHandler);
        myTimer.start();
    }

    public function timerHandler(event:TimerEvent):void {
        _buffer_plotter.draw_buffer(floatsample_array_r,//
        freqbuffer,1900);
    }

    private function loadSound(_soundURI:String):void {
        _loop_snd=new Sound  ;
        _loop_snd.addEventListener(Event.COMPLETE,loadSoundComplete);
        _loop_snd.addEventListener(IOErrorEvent.IO_ERROR,loadSoundError);
        _loop_snd.load(new URLRequest(_soundURI));
    }

    private function loadSoundComplete(event:Event):void {
        sound_array[0]=_loop_snd;
        startPlay();
    }

    private function startPlay():void {
        _out_snd.addEventListener("sampleData",buffer_playback);
        _out_snd.play();
    }

    private function loadSoundError(event:Event):void {
        trace("loadError");
    }

    private function pb_sound_change(_soundURI:String) {
        _loop_snd=new Sound;
        _loop_snd.addEventListener(Event.COMPLETE,//
        sound_change_loadSoundComplete);
        _loop_snd.addEventListener(IOErrorEvent.IO_ERROR,loadSoundError);
        _loop_snd.load(new URLRequest(_soundURI));

        }
```

```
private function sound_change_loadSoundComplete(event:Event):void {
      sound_array[0]=_loop_snd;
}

private function buffer_playback(event:SampleDataEvent):void {

      _samples=new ByteArray  ;
      var len:Number=sound_array[0].extract(_samples,BUFFER_SIZE);

      i=0;
      if (len<BUFFER_SIZE) {
            len+=_loop_snd.extract(_samples,BUFFER_SIZE-len,0);
      }

      _samples.position=0;

      while (i<BUFFER_SIZE) {

            l=_samples.readFloat();
            r=_samples.readFloat();

            if (! bypassEQ) {
                  l=_eqL.compute(l);
                  r=_eqR.compute(r);
            }

            floatsample_array_l[i]=l;// samples in Array speichern
            floatsample_array_r[i]=r;

            if (!einaus) {
                  l=0;// ruhe
                  r=0;
            }

            event.data.writeFloat(l);
            event.data.writeFloat(r);
            i++;
      }

      floatsample_array_r = _FFTransformer.compute(//
      floatsample_array_r,BUFFER_SIZE);

      for (k=0; k<BUFFER_SIZE; k++) {
                  freqbuffer[k]=k*44100/BUFFER_SIZE;
            }
      }

function slider_freq(event:SliderEvent):void {

      _eqL.f0=Math.round(Math.pow(10,event.value));
      _eqR.f0=Math.round(Math.pow(10,event.value));
      eqctrl1.freq_text.text=String(Math.round(Math.pow(10,//
      event.value)));
}

function slider_gain(event:SliderEvent):void {
      _eqL.dBgain=event.value;
      _eqR.dBgain=event.value;
      eqctrl1.dBgain_text.text=String(event.value);
}
```

```
	function slider_q(event:SliderEvent):void {

		_eqL.q=event.value;
		_eqR.q=event.value;
		eqctrl1.q_text.text=String(event.value);
	}

	function fileup_click(e:Event) {
		fileReference.browse([new FileFilter("mp3 files","*.mp3")]);
	}

	function weiter_click(e:Event) {
		this.gotoAndPlay(this.currentFrame+1);
	}

	private function fileReferenceSelectHandler(ev:Event):void {
		loader.getSound(fileReference);
	}

	private function mp3LoaderCompleteHandler(ev:MP3SoundEvent):void {
		sound_array[0]=ev.sound;
	}

	function bypassEQ_click(e:Event) {
		bypassEQ=! bypassEQ;
	}

	function einaus_click(e:Event) {
		trace("clicked");
		einaus=!einaus;
	}

	function antwort1_eval() {

		if ((int(antwort1_input.text) >200) && //
		(int(antwort1_input.text)<240)) {

			korrektur_eval.text="Richtig!";
		} else {
			korrektur_eval.text="Falsch!";
		}
	}

	private function keyHandler(evt:KeyboardEvent):void {

		if (evt.keyCode==13) {
			antwort1_eval();
		} else {
			korrektur_eval.text="";
		}
	}
}
}
```

9.5 CD mit Kursmaterialien

Auf der beigelegten CD befinden sich die Kursmaterialien, sowie eine im Browser benutzbare Version des Kurses.

Die CD ist mit Windows- und Macintoshcomputern kompatibel.

10 Literaturverzeichnis

10.1 Fachbücher

Allen, Chris: The Essential Guide to Open Source Flash Development – New York, USA: Springer, 2008

Ballou, Glen (Hrsg.), Handbook for Sound Engineers – 2. Auflage - Indiana, Alaska, USA: Sams & Co, 1987

Braunstein, Roger: ActionScript 3.0 Bible – Indianapolis, USA: Wiley, 2008

De Donatis, Antonio: Advanced ActionScript Components - Mastering the Flash Component Architecture - New York, USA: Springer, 2006

Dickreiter, Michael: Handbuch der Tonstudiotechnik Band 1. - 5. Auflage – München: Saur, 1987

Dickreiter, Michael: Handbuch der Tonstudiotechnik Band 2. - 5. – München: Saur, 1987

Douglass, Bruce: Real-Time Design Patterns - Robust Scalable Architecture for Real-Time Systems – Boston, USA: Addison Wesley, 2003

Everest , F. Alton: The Master Handbook Of Acoustics – 4. Auflage - New York, USA: McGraw-Hill , 2001

Hayes, Monson H. : Schaum's Outline of Theory and Problems of Digital Signal Processing - New York, USA: McGraw-Hill , 1999

Izhaki, Roey: Mixing Audio - Concepts, Practices and Tools - Oxford, GB: Elsevier, 2008

Kahrs, Mark und Brandenburg, Karlheinz: Applications of Digital Signal Processing to Audio and Acoustics - New York, USA: Kluwer Academic Publishers, 2002

Lott, Joey und Patterson, Danny: Advanced ActionScript 3 with Design Patterns - Berkeley, CA, USA: Peachpit Press, 2007

Lott, Joey und Schall, Darron: ActionScript 3.0 Cookbook - Sebastopol, CA, USA: O'Reilly Media, 2006

Madisetti, Ed. Vijay K. and Williams, Douglas B.: Digital Signal Processing Handbook - Boca Raton, USA: CRC Press LLC, 1999

Moock, Colin: Essential ActionScript 3.0 - Sebastopol, CA, USA: O'Reilly Media, 2007

Owsinski, Bobby: The Mixing Engineer's Handbook – Valleijo, CA, USA: Mixbooks, 1999

Rocchesso, Davide: Introduction to Sound Processing – Firenze, I: Phasar, 2003

Rorabaugh, C. Britton: Digital Filter Designer's Handbook – 2. Auflage – New York, USA: McGraw-Hill, 1997

Smith, Steven W. : The Scientist and Engineer's Guide to Digital Signal Processing – 2. Auflage - San Diego, USA: California Technical Publishing, 1999

Watkinson, John: An Introduction to Digital Audio – Oxford, England: Elsevier, 1994

White, Steve: Digital Signal Processing - A Filtering Approach – Florence, USA: Delmar Cengage Learning, 2000

Winder, Steve: Analog and Digital Filter Design – 2. Auflage – Woburn, USA: Elsevier Science (USA), 2002

Zölzer, Udo: Digital Audio Signal Processing – 2. Auflage – Southern Gate, GB: John Wiley & Sons, 2008

10.2 Firmenschriften

Bores Signal Processing, Introduction to DSP – Surrey, GB, 2000

SevenWoodsAudio, Analog Audio Tone Controls and Equalizers – Belmont, USA, 2002

10.3 wissenschaftliche Arbeiten

o. V.: Theorie zur Digitalen Signalverarbeitung und Regelung. - 2003. - 70 S.

Göttingen, Georg-August-Universität, Drittes Physikalisches Institut, Skript, 2001

10.4 Internet

Bernsee, Stephan M. : The DFT "à Pied", URL:
http://www.dspdimension.com/admin/dft-a-pied/, verfügbar am 3.8.2009

o.V. : Running locally loaded MP3 files through an fx chain, URL:
http://www.flexiblefactory.co.uk/flexible/?p=46, verfügbar am 12.9.2009

o.V.: ActionScript 3.0 Language and Components Reference, URL:
http://livedocs.adobe.com/flash/9.0/ActionScriptLangRefV3/, verfügbar am 10.8.2009

11 Erklärung

Ich erkläre, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Mittweida, den 27. Oktober 2009