Hochschule Darmstadt
- Fachbereich Informatik -

Integration und Test von Audio Bibliotheken für das Mixed Reality System CINeSPACE

Abschlussarbeit zur Erlangung des akademischen Grades
Bachelor of Science (B.Sc.)

vorgelegt von

Marco Münch

Referent : Prof. Dr. Hans-Peter Weber
Korreferent : Prof. Dr. Wolf-Dieter Groch

Projekt:

CINeSPACE

von

Abteilung: A2 – Industrielle Anwendung

Betreuer: Petro Santos, Dominik Acri

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, wurden als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Darmstadt, den 28.03.10 Marco Münch

Abstrakt:

Zweck dieser Arbeit ist eine geeignete Audio Bibliothek für das Mixed Reality System CINeSPACE zu finden. Mithilfe dieser Audio Bibliothek soll es bei dem Projekt möglich sein, sich zwischen zwei unabhängigen, mobilen CINeSPACE Head Mounted Display zu unterhalten. Des weiteren soll es damit möglich sein, ein Gespräch aufzuzeichnen und dieses auf einem externen Server dauerhaft zu speichern.

Dazu werden mögliche Alternativen von verschiedenen Herstellern auf dem Markt näher betrachtet, deren Möglichkeiten aber auch deren Grenzen aufgezeigt. Aufgrund dieser Vorarbeit werden danach verschiedene Audio Bibliotheken ausgewählt und bewertet. Hiernach erfolgt eine weitere Analyse verschiedener Audio Codecs.

Schlüsselwörter:

CINeSPACE, Audio Bibliotheken, RTAudio, Directshow, Audiocodec, MP3, OGG, Speex,

Inhaltsverzeichnis

1. Einleitung

In diesem Kapitel wird der Aufbau der Arbeit erklärt, das Projekt CINeSPACE vorgestellt und dessen Zielsetzung näher erläutert, die während des Praxissemesters gesetzt worden sind.

1.1 Aufbau der Arbeit

Im **zweiten** Kapitel werden die Grundlagen vermittelt, die für das Verständnis dieser Arbeit nötig sind.
Im **dritten** Kapitel werden die Probleme analysiert, die bei dem CINeSPACE Projekt auftreten können.
Das **vierten** Kapitel beschäftigt sich mit dem aktuellen Technikstand. Es werden einige Programme vorgestellt die einige Gemeinsamkeiten mit dem Projekt aufweisen und Gründe aufgeführt weswegen diese Programme ungeeignet sind.
Im **fünften** Kapitel schließlich werden einige ausgewählte Audio Bibliotheken vorgestellt und dann einigen Tests unterzogen.
Vorstellung der Audio Codecs und deren Test erfolgen dann im **sechsten** Kapitel der Arbeit.
Im **siebten** und letzten Kapitel wird dann das Ergebnis des Projekts betrachtet und deren Zukunft im Projekt CINeSPACE.
Im Anschluss an diese Kapitel befinden sich noch als Anhänge das Literaturverzeichnis, Abbildungsverzeichnis, Tabellenverzeichnis und das Glossar.
Auf der letzten Seite dieser Arbeit befindet sich noch eine CD mit Daten für diese Arbeit. Unter anderem befindet sich dort diese Arbeit als PDF-Dokument, die Sprachsamples und zwei Beispielprogramme.

1.2 Vorstellung von CINeSPACE

CINeSPACE ist eine Mixed Reality Hardware und Software Lösung für Filmtouristen die im Auftrag der Europäischen Städte Glasgow (Irland), San Sebastian (Spanien) und Venedig (Italien) entwickelt werden soll.
Mixed Reality ist ein Sammelbegriff und darunter versteht man entweder eine visuelle Erweiterung der Realität durch virtuelle Objekte (Augmented Reality) oder eine Erweiterung der Virtualität durch Reale Objekte (Augmented Virtuality) (*Siehe Abbildung 1*).

Während Augmented Reality, oder auch kurz AR genannt, mithilfe von Sichtshilfen virtuelle Objekte dem Menschen Zusatzinformation bieten soll, so wird die Augmented Virtuality dazu genutzt um reale Objekte in eine virtuelle Umgebung zu integrieren. [MILGRAM94]

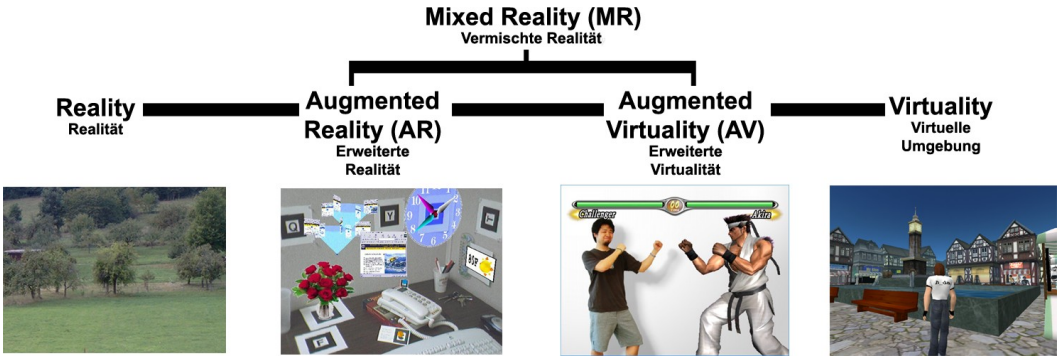


Abbildung 1: Mixed Reality nach Milgram und Takemura

Das Cinespace Projekt widmet sich nur einem Teilbereich der Mixed Reality und zwar der Augmented Reality. Diese Erweiterung der Realität wird mit Hilfe eines so genannten Head Mounted Display (HMD) erreicht. Diese meisten HMDs sind Groß, Schwer und Unflexibel (*Siehe Abbildung 2*) da sie meist mit Kabeln an Computern verbunden sind. Selbst bei den mobilen HMDs müssen selbst kleine Computer in einem Rucksack herum getragen werden.

Abbildung 2: Head Mounted Displays [HMDS08]

Das CINeSPACE Projekt soll es den Filmtouristen ermöglichen durch die Stadt zu laufen um sich vor Ort Filmsequenzen anzuschauen. Dafür sind herkömmliche HMDs ungeeignet. Deswegen wurde ein etwa zwei Kilogramm leichtes HMD entwickelt, welches sich am besten als ein größeres Fernglas beschreiben lässt (*Siehe Abbildung 3*).

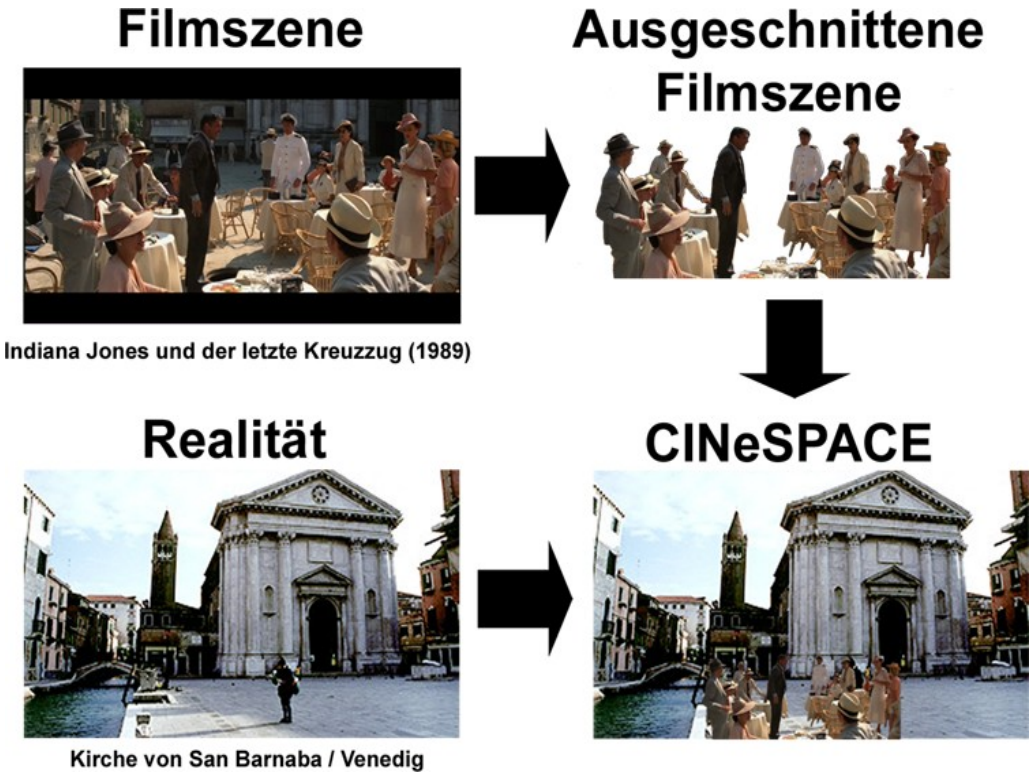
Dieses ist im Gegensatz zu herkömmlichen HMDs dank Wireless LAN unabhängig von Kabel. Ebenso besitzt das CINeSPACE HMD einen leistungsfähigen Akku der lange Erkundungstouren ermöglicht.

Mit dem CINeSPACE HMD sollen Filminteressierte mit Hilfe von Outdoor Positioning Technologies an vergangene Drehorte der jeweiligen Stadt geleitet werden. Diese Outdoor Positioning Technologies kann man mit einem Navigationsgerät vergleichen. Aber im Gegensatz zum Navigationsgerät können die Standortkoordinaten nicht von einem Satelliten erfasst werden, da in sehr engen Gassen kein Satellit das HMD erfassen kann. Aus diesem Grund muss man die Koordinaten mithilfe anderer Technologien erfassen.
Der Filmtourist kann vor Ort Informationen über den Film abrufen und entsprechende Filmsequenzen durch das HMD einblenden lassen. Dabei werden nur die Schauspieler in die Sequenz eingefügt, der Rest um die Schauspieler ist die Realität (*Siehe Abbildung 4*).
Im Endprodukt des CINeSPACE HMDs soll das Bild nicht mehr von einer Videokamera aufgezeichnet und mit LCD Display dem Auge vorgespielt werden, sondern das Bild soll eins zu eins durchgeleitet werden und nur an der entsprechenden Stelle, wo sich ein virtuelles Objekt befinden soll, werden die Pixel Licht undurchlässig geschaltet und das Objekt drauf projiziert. Im aktuellen Prototypen ist das aber noch nicht integriert.

Aber nicht nur Filmtouristen sollen mit CINeSPACE angesprochen werden, sondern es soll auch der normale Stadttourist angesprochen werden. So soll CINeSPACE nicht nur Drehorte gespeichert haben sondern auch Standorte und Informationen über Sehenswürdigkeiten,

Museen oder Restaurants.
Die Touristen sollen auch die Möglichkeit erhalten einen Beitrag für die nachfolgenden Touristen zu leisten. So können Sie damit auch Fotos schließen, kleine Videos aufnehmen und auch gesprochene Kommentare abgeben. Diese Zusatzinformationen sollen von jedem Nutzer angehört werden können.

Aber auch Filmregisseure sollen von diesem Gerät profitieren. So kann eine Filmcrew an vielen verschiedenen Orten gleichzeitig auf die Suche nach möglichen Drehorten gehen. Wenn jemand meint einen geeigneten Drehort gefunden zu haben, so kann er dann ein Foto schießen und es den Crewmitgliedern sofort zu senden, damit sie es sich anschauen und gleich entscheiden können, ob sich dieser Drehort für ihrenFilm eignen würde. Über das im System integrierten Mikrofon kann sich das Filmteam unterhalten und Meinungen austauschen.



Mit dem CINeSPACE Gerät kann man somit eine komplette Filmproduktion dokumentieren und diese dann Filmtouristen zugänglich machen.

1.3 Zielsetzung der Arbeit

Die Zielsetzung dieser Arbeit ist das Auffinden und Testen einer Audio Bibliothek und eines Audio Codecs für das CINeSPACE Projekt.
Diese soll es zum einen ermöglichen sich zwischen zwei CINeSPACE HMDs zu unterhalten und zum anderen die Möglichkeit erhalten, vor Ort eine Geschichte oder Informationen zu erzählen und diese direkt auf dem Server zu speichern. Damit soll man es anderen Leuten sie Gelegenheit geben diese Aufzeichnungen jederzeit anzuhören und somit zusätzliche Informationen bekommen.

Um das zu Realisieren wird zuerst einmal in den aktuellen IT-Markt umgeschaut ob es nicht schon eine Möglichkeit gibt, welche die gewünschten Anforderungen erfüllt. Aufgrund dieser Analyse werden einige Audio Bibliotheken ausgewählt und näher betrachtet. Wenn dies geschehen ist, werden einige Audio Codecs ausgewählt und auf die Eignung für das CINeSPACE Projekt getestet.

2. Grundlagen und Begriffserklärungen

Wie wir gesehen haben benötigen wir eine Audio Bibliothek, mit deren Hilfe wir eine Kommunikation zwischen zwei CINeSPACE HMDs realisieren sollen. Dazu benötigen wir einige Grundkenntnisse die in diesem Kapitel erklärt werden sollen.

2.1 Audio Bibliothek

Eine Audio Bibliothek ist eine Software die eine Verbindung von der Soundkarte zum eigentlichen Betriebssystem herstellen soll. Diese Audio Bibliotheken stellen eine Verbindung zum Gerätetreiber der Soundkarte her und der wiederum spricht die Soundkarte an. Über die Soundkarte kann man nun angeschlossene Lautsprecher und Mikrophone ansprechen und somit Geräusche mit dem Mikrophon aufnehmen und über die Lautsprecher wiedergeben (*Siehe Abbildung 5*).

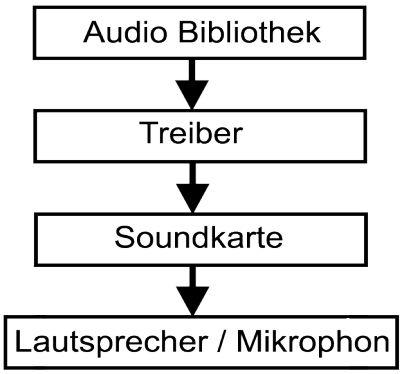


Abbildung 5: Aufbau eine Audio Bibliothek

Die Audio Bibliothek muss auch die Treiber des Betriebssystems zugreifen, daher muss die Audio Bibliothek speziell für jedes Betriebssystem angepasst sein.

2.2 Audiosignal

Wie wir im Kapitel eins erfahren haben soll ein Gespräch übertragen werden. Damit es überhaupt übertragen werden kann müssen diese akustische Signale erst einmal in einen für den Computer verarbeitbares Signal umgewandelt werden.
Für einen Menschen ist es einfach Audiosignale zu verarbeiten. Wenn sich zwei Menschen unterhalten so werden Schallwellen übertragen und vom Ohr von einer Person eingefangen. Die Schallwellen werden dann vom Trommelfell in ein Signal umgewandelt, daß das Gehirn

verarbeiten kann. Dadurch kann ein Mensch hören.

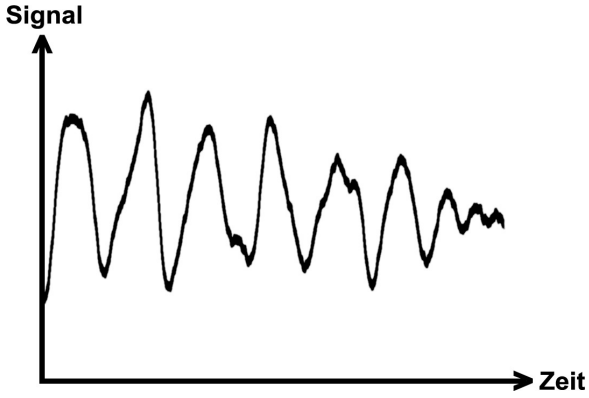


Abbildung 6: Analoges Signal

Diese Schallwellen kann man als analoges Signal darstellen (*Siehe Abbildung 6*). Ein Computer hat ein Problem diese kontinuierlichen Signale zu verarbeiten. Mithilfe eines Mikrophons können wir diese analoge Signale erfassen. Da aber ein Rechner nicht mit analogen Daten umgehen kann müssen diese Signale erst in ein digitales Signal umgewandelt werden. Dazu wird innerhalb eines gleich bleibenden Zeitabstandes die Werte gemessen (*Siehe Abbildung 7*).

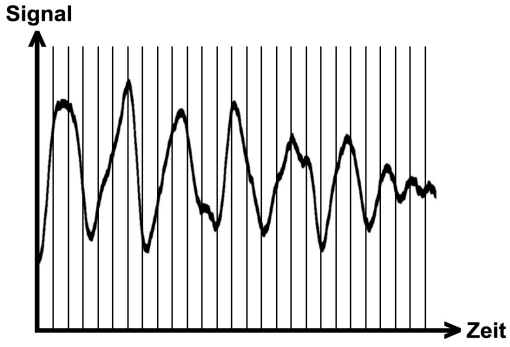


Abbildung 7: Analoges Signal mit Abtastrate

Der Zeitabstand von einer Signalmessung zu einer anderen Signalmessung heißt Abtastrate. Diese Abtastrate liegt bei Audiosignalen meist zwischen 8 kHz (Telefonqualität) und 44,1 kHz (CD Qualität). Eine Abtastrate von 1 kHz entspricht 1000 Werte pro Sekunde.
Für ein Digitales Signal benötigt man noch die Signalwerte. Dazu werden die Signalwerte auf einen endlich Bereich abgebildet. Dieser Bereich besteht meist aus 2^n Werten. Diesen Vorgang der Signalwerte auf einen endlichen Bereich abzubilden nennt man Quantisierung.

Durch diese Quantisierung entsteht aus dem analogen Signal ein digitales Signal, welches auch ihr Computer verarbeiten kann (*Siehe Abbildung 8*).

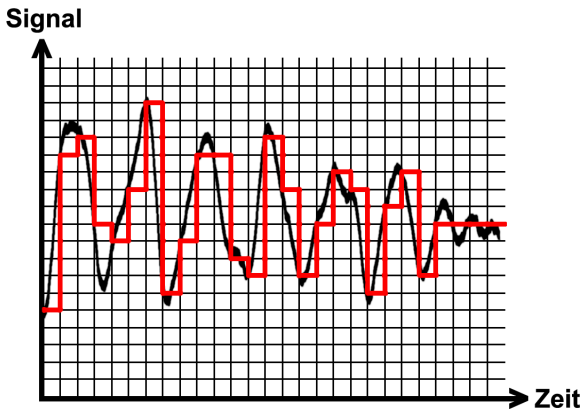
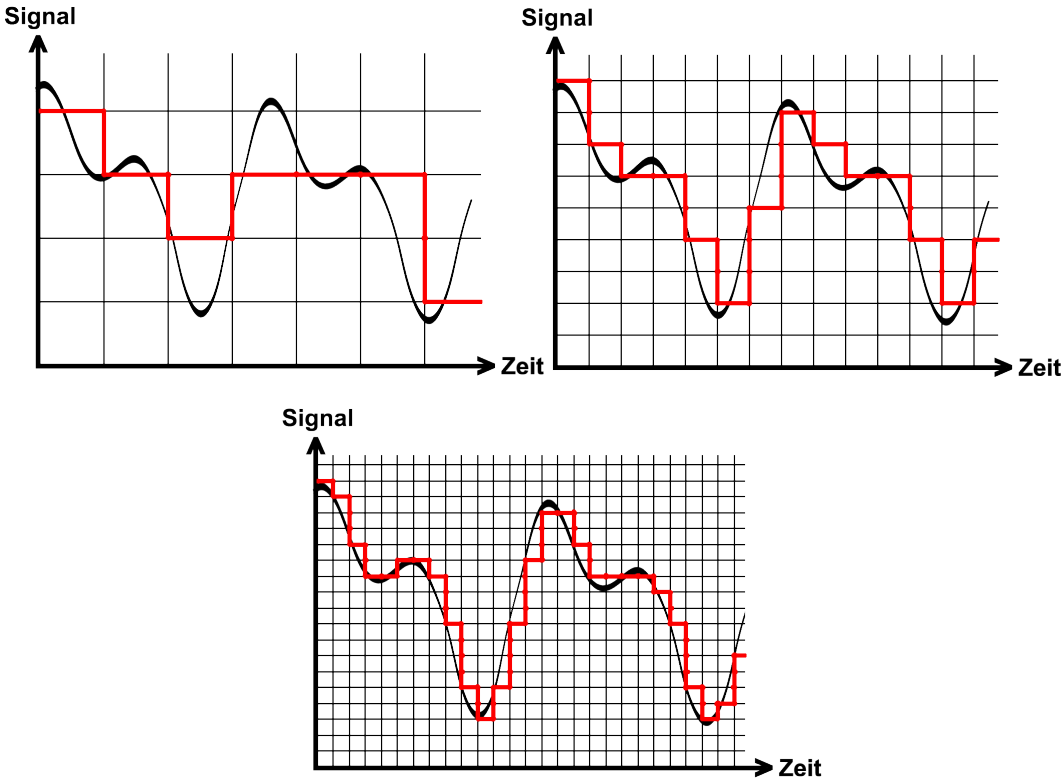


Abbildung 8: Analoges Signal in Digitales Signal umwandeln

Gerade gegen Ende des Analogen Signals von Abbildung 8 sehen wir, dass das Digitale Signal (Rot) gleich bleibt, das Analoge Signal sich aber ändert.
Damit solche schnellen Signaländerungen erfasst werden können muss die Abtastrate und die Quantisierung erhöht werden. Dadurch ergibt sich ein deutlich besseres digitale Signal (*Siehe Abbildung 8*).
Frequenzen bis etwa 20 kHz kann das menschliche Gehör wahrnehmen. Nach dem Nyquist-Shannon-Abtasttheorem, manchmal auch WKS-Sampling-Theorem genannt, muss eine Frequenz mit doppelter Abtastrate abgetastet werden, damit keinen Informationsverlust auftritt. Für den Menschen, der höchstens Frequenzen bis 20 kHz wahrnehmen kann, muss mit mindestens 40 kHz abgetastet werden, so das für ihn kein wahrnehmbarer Unterschied auftritt. In der Audiotechnik wird sich mit 44,1 kHz abgetastet. Dieser Wert entspricht der Qualität einer CD. Eine höhere Abtastrate bringt daher keine Qualitätsverbesserung mehr für das menschliche Gehör.[RECHENBG02]



Aber je mehr Werte für die Quantisierung benötigt wird und je die Abtastraten je Sekunden sind, desto höher ist natürlich auch der Speicherverbrauch. Also 2^2 Werte für die Quantisierung ergibt 4 Werte (binär: 00, 01, 10 und 11). Man benötigt also für jeden Abtastwert 2 Bit Speicherplatz. Für 2^4 sind es schon 4 Bit die man benötigt. Diese Art der Abbildung der Signalwerte auf einen Binärwert nennt man Pulse-Code-Modulation (PCM).

Um eine Vorstellung davon zu bekommen welche Datenmengen angefallen berechnen wir das an einem Beispiel.

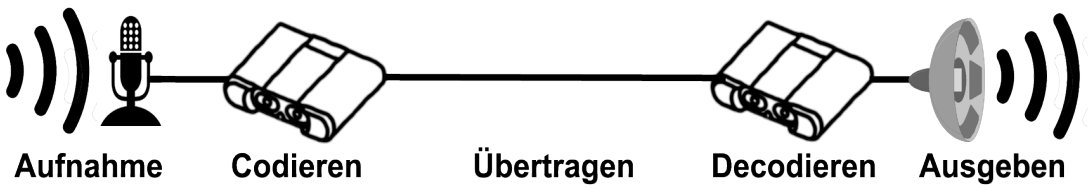
Abtastrate:	8 kHz
Quantisierung:	8 Bit/Wert, entspricht $2^8 = 256$ Werte
Bitrate:	8000 Werte/s * 8 Bit/Wert = 64000 Bits/s ~ 64 kBit/s

An diesem Beispiel sehen wir bei Telefonqualität und der niedrigen Quantisierung von 8 Bit auf einen Datendurchsatz von 64 kBit/s kommen. Wenn nun zwei Personen, die sich wie bei einem Telefongespräch gleichzeitig unterhalten wollen, benötigen die doppelte Bandbreite. In diesem Fall 128 kBit/s.

Je mehr Gesprächsteilnehmer es gibt desto größer muss die Bandbreite sein, denn es fallen dadurch auch mehr Daten an.
Daher hat man nur zwei Möglichkeiten diese technische Hürde zu überwinden: Entweder einer Vergrößerung der Bandbreite oder die Verringerung der Datenmenge. Die Vergrößerung der Bandbreite ist eine sehr aufwändige und teuere Investitionen, da man sie nur durch zusätzliche Hardware erreichen kann. Die Verringerung der Datenmenge durch Algorithmen ist dagegen eine billige Alternative. Diese Verringerung der Datenmenge nennt man komprimieren.

2.3 Komprimieren eines Audiosignals

Die Komprimierung eines Audiosignals ist nötig um die anfallende Datenmenge möglichst gering zu halten. Diese Reduzierung der Daten erfolgt mithilfe von verschiedenen Algorithmen, die je nach verwendeten Codec unterschiedlich ist. Ein Codec ist ein Kunstwort für **Co**dierer und **Dec**odierer, denn die Daten müssen vom hörbaren Signal in ein digitales Signal umgewandelt werden und danach codiert werden. Dieser Vorgang heißt Codieren. Wenn diese codierten Daten angehört werden möchten so müssen sie erst wieder in einen für den Computer verständliches Format umgewandelt werden. Diesen Vorgang heißt Decodieren.
Wie das im Falle des CINeSPACE Projektes aussieht erkennt man in Abbildung 10.



Die Komprimierung kann je nach verwendeten Codec unterschiedlich stark sein. Bei den Codecs kann man zwischen Verlustfrei und Verlustbehaftet unterscheiden.
Die Funktionsweise bei verlustbehafteten Codecs ist meist relativ ähnlich. Das menschliche Gehör erkennt in etwa Frequenzen zwischen 16 Hz und 20 kHz. Den Bereich unterhalb der

menschlichen Wahrnehmung heißt Infraschall und alles was darüber liegt Ultraschall beziehungsweise ab 1 Ghz Hyperschall. Diese Bereiche sind für den Menschen ohne Hilfsmittel nicht wahrnehmbar. Das Mikrophon nimmt diese Bereiche mit auf, obwohl diese der Mensch gar nicht hören kann. Diese verlustbehafteten Codecs entfernen die Audiosignale in diesem irrelevanten Bereich.

Verlustbehaftete Codecs gibt es viele und um einen Überblick zu bekommen sind einige in Tabelle 1 mit ihren minimalen und maximalen Bitraten aufgelistet. Diese Codecs stellen nur einen Bruchteil der Audio Codecs dar und sind bis auf MP3 und Vorbis alles Codecs die auf menschliche Sprache ausgelegt sind. MP3 und Vorbis sind auf die Komprimierung von Musik ausgelegt.

Name	Min. Birate	Max. Bitrate
GSM 6.06	13 kbit/s	13 kbit/s
G.723.1	5.3 kbit/s	6.3 kbit/s
G 729 A	8 kbit/s	8 kbit/s
MELP	0.6 kbit/s	2.4 kbit/s
MPEG 1 Layer 3 (MP3)	8 kbit/s	320 kbit/s
Vorbis	16 kbit/s	128 kbit/s
Speex	2 kbit/s	44 kbit/s

Tabelle1: Übersicht von verlustbehafteten Codecs [VOCAL07]

Bei den verlustfreien Codecs hingegen verliert man keinerlei Informationen, sondern wird mit verschiedenen Algorithmen kleiner gemacht. Dabei kann man höchstens je nach Codec zwischen 20% und 50% der ursprünglichen Dateigröße einsparen. Diese Codecs kommen dabei höchstens für Audiostudios oder zur Archivierung von Musik zum Einsatz. Auch die lange Komprimierungszeit machen den Einsatz von verlustfreien Codecs für das Projekt unmöglich.

Wie die Algorithmen bei jedem verlustfreien und verlustbehafteten Codecs im einzelnen funktioniert würde aber den Rahmen dieser Arbeit sprengen.
Im sechsten Kapitel werden einige verlustbehaftete Codecs aber etwas näher untersucht.

2.4 Übertragen von Sprache über ein Netzwerk

Wie wir die Sprache aufnehmen und wieder ausgeben können haben wir in den vorhergegangenen zwei Teilkapiteln gesehen. In diesem Kapitel geht es darum diese Codierten Audiosignale über ein Netzwerk zu übertragen.
Im Jahr 1995 wurde das erste Mal Sprache in Echtzeit über ein TCP/IP Netzwerk übertragen. Das wurde von der israelischen Firma VocalTec ermöglicht, wobei die Gesprächspartner nur abwechselnd sprechen konnten. Sie nannten dieses Verfahren Voice over Internet Protocol, abgekürzt auch Voice over IP oder VoIP genannt. Dieses Verfahren hat sich inzwischen durchgesetzt und wird nicht nur bei Gesprächen zwischen Personen über Computern, sondern auch über normale Telefone eingesetzt.
Dazu wird das Audiosignal in Stücke zerlegt und anschließend übertragen. Je größer diese Stücke sind desto länger muss der Hörer warten bis die Audiosignale komplett angekommen sind um abgespielt zu werden. Da diese Anwendung aber Echtzeit kritisch ist müssen diese Datenpakete möglichst klein gehalten werden, damit ein vernünftiges Gespräch gehalten werden kann.

Um das zu verdeutlichen kann man es sich an einem Beispiel anschauen:
Nehmen wir also mal an, dass diese Datenpakete jeweils 1 Minute Sprache enthalten, was in der Wirklichkeit natürlich nie der Fall sein wird. Die Person A begrüßt Personen B , und nun dauert es 1 Minute bis Person B überhaupt die Begrüßung hört. Da man aber annehmen kann das eine Begrüßung maximal 5 Sekunden dauert muss Personen A die restliche Zeit warten. Wenn nun Person B zurück antwortet dauert es ebenfalls 1 Minute bis diese Nachricht bei Person A ankommt. Man kann also davon ausgehen das Person A 1 Minute und 55 Sekunden warten muss. An diesem Beispiel kann man sehen das ein zu großes Datenpaket unproduktiv ist.
Daher muss man möglichst kleine Pakete nehmen. In der Voice over IP Technik haben sich Paketgrößen von 20 bis 30 Millisekunden in den meisten Anwendungen als ein vernünftiges Maß herausgestellt.
Kleine Datenpakete sichern aber noch lange keine fehlerlose Datenübertragung. Daher muss man bei Voice over IP Anwendungen einige Sachen beachtet werden. Diese kann man als Merkmale für die Güte eines Signals (Dienstgüte) ansehen [MASSOTH07]. Darunter versteht man:

- **Delay**: Verzögerung, Latenzzeit
- **Jitter**: Varianz der Verzögerung
- **Throuthput**: Durchsatz
- **Packet Loss**: Paketverlust

Unter der Verzögerung in einem Netzwerk versteht man die Zeit, welche man benötigt um einen Ereignis auszulösen. In unserem Fall ist die Zeit gemeint, welche man benötigt um einen Datensatz von einem Gerät zum anderen zu schicken.

Die Varianz der Verzögerung gibt an wie groß diese Verzögerung ist. Bei einen zu großen Jitter kann es passieren, dass einige Datenpakete erst später beim Empfänger ankommen. Da man also nicht die Pakete so abspielen kann wie man sie bekommt müssen sie erst in einem Puffer zwischengespeichert werden und in die richtige Reihenfolge gebracht werden. Das führt natürlich zu einer weiteren Vergrößerung in der Verarbeitung der Daten.

Die Menge an Daten, die man über das Netzwerk verschicken kann, nennt man Durchsatz. Dabei muss man unterscheiden zwischen der Datenübertragungsrate und den Datendurchsatz. Die Datenübertragungsrate ist die maximale Kapazität die ein Netzwerk übertragen kann. Wenn man von dieser Datenübertragungsrate alle Daten abzieht, welche nur für die Übertragung benötigt werden, so erhält man den für die eigene Anwendung verfügbaren Datendurchsatz. Dieser Datendurchsatz ist je nach verwendeten Netzwerk unterschiedlich. In Tabelle 2 sieht man verschiedene Netzwerktechniken und deren Übertragungsrate und den tatsächlichen Durchsatz.

Technik	Datenübertragungsrate (Brutto-Rate)	Datendurchsatz (Netto-Rate)

Fast Ethernet	100 Mbit/s	94 Mbit/s
Gigabit Ethernet	1000 Mbit/s	940 Mbit/s
WLAN 802.11g	54 Mbit/s	5-25 Mbit/s
WLAN 802.11n	300 Mbit/s	5-120 Mbit/s
Powerline	200 Mbit/s	15-85 Mbit/s

Tabelle2: Vernetzungstechniken [ATLERS07]

Als letzte Eigenschaft, die als Dienstgüte bewertet werden kann, ist der Paket Verlust. In einem großen Netzwerk, wie dass das für das CINeSPACE Projekt benötigt wird, können auch Datenpakete verloren gehen. Dieser Paket Verlust kann unter anderem auftreten wenn Leitungen gestört, Leitungen beschädigt oder die Datenpakete bei der Zwischenspeicherung in Routern verloren gehen.
Diese Probleme haben nichts mit der eigentlichen Anwendung zu tun, müssen aber für eine Netzwerkanwendung trotzdem beachtet und entsprechend darauf reagiert werden.

Auf diese Probleme kann man mithilfe so genannter Netzwerkprotokolle mildern. Diese Netzwerkprotokolle sind oberhalb der so genannten Internet Protokoll (IP) Schicht (*Siehe Abbildung 11*).

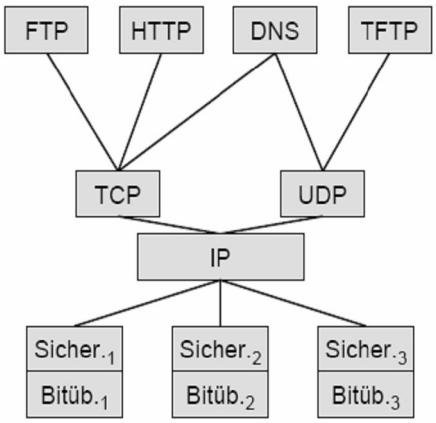


Abbildung 11: Internetarchitektur als Protokollgraph [MASSOTH07]

In der Netzwerktechnologie haben sich zwei Protokolle durchgesetzt. Das wären zum einen das verbindungsorientierte TCP und zum anderen das verbindungslose UDP. Schauen wir

uns aber diese Protokolle näher an.

2.4.1 Transmission Control Protocol (TCP)

TCP ist die Abkürzung für Transmission Control Protocol und realisiert eine sichere Verbindung. Das Transmission Control Protocol ist aus diesem Grund das mit am häufigsten verwendete Transportprotokoll. Es sorgt unter anderem dafür das keine Datenpakete verloren gehen, denn jedes Datenpaket ist durchnummeriert. Die Nummerierung erfolgt mit der so genannten Sequenznummer. Die sichere Verbindung wird mit dem so genannten Drei Wege Handschlag realisiert (*Siehe Abbildung 12*).

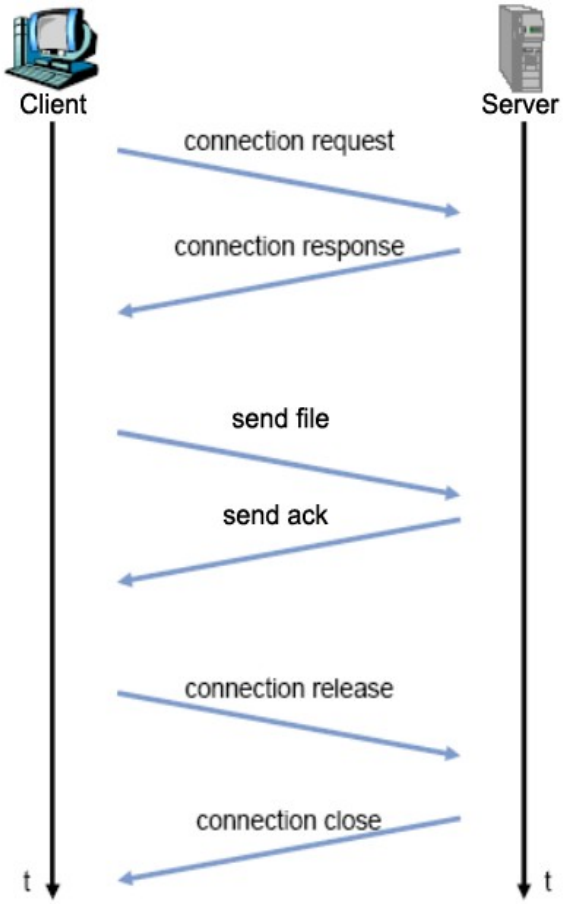


Abbildung 12: Drei Wege Handschlag [MASSOTH07]

Dazu sendet zuerst der Client eine Aufforderung zum Verbindungsaufbau (Connection Request) an den Server. In dieser Aufforderung befindet sich unter anderem auch die erste

Sequenznummer des Datenpakets, welches aus einem zufälligen Startwert besteht. Der Server bestätigt nun die Verbindung (Connection Response) und wartet auf weitere Datenpakete vom Client.

Dieser sendet nun die einzelnen Datenpakete (send file) in dem er die Sequenznummer für jedes Datenpaket um eins erhöht, dadurch entsteht eine bestimmte Reihenfolge. Wenn der Server nun ein Paket erhält, welches nicht die nachfolgende Sequenznummer des vorangegangenes Paket enthält, sendet der Server eine Mitteilung an den Client das ihn das Paket mit dieser Sequenznummer fehlt. Bei dieser ganzen Technik ist es egal ob das Paket einfach nur verloren gegangen ist oder ob das Paket einfach nur später kommt. Alle Pakete die der Server jetzt noch bekommt verfallen einfach. Der Client unterbricht auf diese Mitteilung vom Server die laufende Übertragung und sendet wieder ab dem entsprechenden Paket neu. Jedes erfolgreich übermittelte Paket quittiert der Server (send ack).

Wenn der Client alle Datenpakete gesendet hat, so sendet er hat dem Server eine Mitteilung das die Übertragung mit diesem Paket beendet ist (Connection Release). Somit weiß der Server nun das dies das letzte Paket ist. Daraufhin sendet er eine Mitteilung zurück an den Client dass er alle Datenpakete bekommen hat (Connection Close).

Um das alles zu ermöglichen müssen neben den anfallenden Daten zusätzliche Informationen mitgesendet werden. Dies steht am Anfang in den so genannten Headern, welches bei jedem Protokoll unterschiedlich ist. Wie der TCP Header aussieht kann man der Tabelle 3 nehmen.
Ein TCP Paket kann zwar maximal 65.515 Byte groß sein, da aber die Paketgröße bei einer Ethernetverbindung bei maximal 1500 Bytes (1500 Byte * 8 Bit = 12.000 Bit) liegt, kann ein TCP Paket diese Grenze nicht überschreiten. So kommt es das bei einer maximalen Übertragung von 1500 Bytes jeweils 20 Bytes für den IP-Header und nocheinmal 20 Bytes für den TCP Header abgezogen werden muss, so das man maximal 1460 Bytes pro Datenpaket über ein Ethernet Netz versenden kann.

1 Bit	16 Bit	17 Bit	32 Bit

Quell-Port			Ziel-Port
Sequenznummer			
Bestätigungsnummer			
Offset	Reserviert	Flags	Fenster
Checksumme			Urgent Pointer
Options (optional)			
Daten			

Tabelle3: TCP Header [MASSOTH07]

Wie man sieht entsteht durch TCP eine Menge an zusätzlichem Daten, auch Overhead genannt. Diese Daten müssen natürlich verarbeitet werden, wodurch eine höhere Verzögerung auftritt. Bei einer Echtzeit kritischen Anwendung sollen die Daten möglichst schnell verarbeitet werden, wodurch die Verwendung von TCP als Übertragungsprotokoll nicht besonders gut geeignet ist. [MASSOTH07]

2.4.2 User Datagram Protocol (UDP)

UDP steht für User Datagram Protocol und ist ein verbindungsloses Transportprotokoll. Die Entwicklung begann 1977 als man ein anderes Protokoll als TCP für die Übertragung von Sprache haben wollte. Das neue Protokoll musste einfach schneller sein und weniger Overhead erzeugen.
Der deutlich geringere Overhead kommt dadurch zu Stande, dass UDP keine Garantie dafür gibt das die Datenpakete überhaupt beim Empfänger ankommen, dass die Datenpakete in der richtigen Reihenfolge ankommen oder dass ein Paket nur einmal ankommt. Die Übertragung von Audiosignalen ist trotz dieser Probleme sehr gut möglich. Wenn diese Probleme nicht so extrem ausfallen bleiben die Gespräche trotzdem verständlich. Das es weniger Overhead hat erkennt man an dem UDP-Header (Tabelle 4), der nur 8 Bytes benötigt. TCP verbraucht für seinen Header 24 Bytes.
Ein Vorteil von UDP ist die schneller Datenversendung, da nicht wie TCP eine Verbindung aufgebaut werden muss. Die Daten werden nur mit einer Senderadresse, Empfängeradresse, der Länge des Datenpakets und einer Prüfsummen versehen und anschließend versendet.

1 Bit	16 Bit	17 Bit	32 Bit

Quell-Port	Ziel-Port
Länge	Prüfsumme
Daten	

Tabelle4: UDP Header [MASSOTH07]

Das User Datagram Protocol bringt bei der Übermittlung von Audiodaten mehr Vorteile als Nachteile, daher verwendet man UDP für die Übertragung des Audiodatenstroms.
Da aber zusätzliche Funktionen für die Steuerung des Datenstroms nötig wird, laufen diese wichtigen Steuerbefehle über TCP, da die Daten auf jeden Fall angekommen müssen.
Für die Übertragung von Audiodaten über ein Netzwerk sind aber noch einige andere Informationen nötig. Daher wurden neue Übertragungsprotokolle entwickelt.

2.4.3 Die Streaming Protokolle

Diese neuen Protokolle sitzen oberhalb der beiden Protokolle TCP und UDP, wie die Protokolle HTTP, FTP oder DNS in Abbildung 11. Diese neuen Protokolle wären:

- Real Time Streaming Protocol (RTSP)
- Real-Time Transport Protocol (RTP)
- Secure RTP (SRTP)
- RTP Control Protocol (RTCP)

Das Real Time Streaming Protocol kann oberhalb von TCP oder UDP sitzen, aber da es für die Steuerung des Audiodatenstroms zuständig ist, und deswegen eine gewisse Wichtigkeit hat, wird es meist über TCP übertragen. In diesem Protokoll werden solche Informationen wie der verwendete Codec, die Art des Übertragungsverfahrens oder Steuermethoden wie Abspielen, stoppen oder pausieren des Datenstroms.
Das Real-Time Transport Protocol ist für das Übertragung der eigentlichen Audioinhalte zuständig. Dieses Protokoll selbst setzt auf UDP auf und soll die Mängel von UDP etwas mildern, aber nicht so langsam wie TCP zu sein. Ein Mangel ist, wie in vorangegangenen Kapitel besprochen, dass es keine Sortierung der Pakete mithilfe einer Sequenznummer gibt. Dadurch fehlen Erkennungsmöglichkeiten für verloren gegangene Pakete und ob das Paket überhaupt noch notwendig ist. Wie man ihn Tabelle 5 sehen kann wurde das Problem mit der

Einführung einer Sequenznummer und eines Zeitstempels ausgemerzt. Zusätzlich wurde noch eine Synchronisation Source ID eingeführt, mit deren Hilfe man die einzelnen Datenströme von einander unterscheiden kann.

1 Bit	16 Bit	17 Bit	32 Bit
Flags		Sequenznummer	
Zeitstempel			
Synchronisation Source ID (SSRC)			
Contributing Source (optional)			
Daten			

Tabelle5: RTP Header [BADACH04]

Zusätzlich zu dem Real-Time Transport Protocol gibt es noch ein Secure RTP welches eine gesicherte Verbindung ermöglicht. Diese spielt im Rahmen dieser Arbeit keine besonders große Rolle, sie sollte nur zur Vollständigkeit genannt werden. Denn im CINeSPACE Projekt sollen keine sicherheitsrelevanten Gespräche geführt werden, so das eine genauere Untersuchung dieses Protokolls unnötig ist.
Das RTP Control Protocol wird dazu genutzt um die Verbindung zwischen Sender und Empfänger zu analysieren. Dazu werden Daten wie Anzahl der gesendeten Pakete, Anzahl der empfangenen Pakete oder Verzögerungen im Datenstrom zwischen den beiden ausgetauscht. Damit lassen sich Statistiken erstellen und man kann darauf reagieren.

3. Problemanalyse

In diesen Kapitel werden mögliche Probleme, die im CINeSPACE Projekt auftreten können, analysiert. Dazu müssen wir zwischen Problemen unterscheiden die wir auf Hardware Seite haben und zum anderen den Problemen die wir auf Software Seite haben.

Zuerst werden wir die technischen Ressourcen des CINeSPACE HMD näher betrachten. Das CINeSPACE HMD besitzt folgende Systemspezifikationen:

Samsung Q1 Ultra

- Intel Streatley 800 MHz
- 1 GB RAM
- HDD 60 GB
- W-LAN 802.11g

Da auf dem CINeSPACE HMD große Teile der GUI als 3D Anwendung ablaufen ist der Prozessor der Flaschenhals des Systems. Im weiteren Verlauf des Projektes soll der Samsung Q1 Ultra durch eine leistungsfähigere Hardware ersetzt werden. Momentan läuft er mit Windows XP.

Aber nicht nur die portable Hardware ist ein Problem sondern auch die Übertragungsgeschwindigkeit des WLAN. Die Übertragungsgeschwindigkeit beträgt bei guten Bedingungen maximal 54 Mbit/s. Durch ständige Abgleichen des eigenen Standortes mithilfe der HMD wird eine hohe Bandbreite für diese Outdoor Positioning Technologies benötigt. Aus diesem Grund muss die benötigte Übertragungsrate zwischen 10 und 20 kbit/s liegen, damit diese die anderen Anwendungen nicht stört.

Die wichtigste Anforderung auf Softwareseite ist die Plattformunabhänigkeit. Sie soll gewährleisten, dass man die Bibliothek einfach und ohne größere Anpassungen auf einem anderen Betriebssystem übernehmen kann. Besonders die am weitesten verbreiteten Betriebssysteme Windows XP, Linux und Mac OS X sollen unterstützt werden.
Des weiteren soll die Audio Bibliothek eine passende Lizenz besitzen. Für das CINeSPACE

Projekt darf höchstens eine Lizenz verwendet werden, die höchstens der GNU LGPL entspricht oder freier ist. Diese Lizenz besagt das Änderungen die an der Bibliothek, die unter der GNU LGPL veröffentlicht wurde, auch unter dieser Lizenz veröffentlicht werden muss. Andere Programmteile, die nichts mit dieser Bibliothek zu tun haben, können einer anderen Lizenz angehören. Bei der GNU GPL muss dagegen das komplette Projekt im Sourcecode veröffentlicht werden, das ist aber bei dem CINeSPACE Projekt nicht erwünscht. [GNULGPL]

Um weitere Anforderungen der Audio Bibliothek und des Audiocodecs zu erfassen wähle ich zwei Szenarien um weitere Probleme und Anforderungen herauszufinden:

Szenario 1:

Die zwei Freunde Alfred und Tobias fahren nach Venedig und möchten das neue CINeSPACE HMD ausprobieren. Da sie nur einen Tag zur Verfügung haben kommen sie auf die Idee, dass sie sich einen halben Tag lang trennen sollen und jeder einen Teil der Stadt abläuft, um die besten Plätze der Erweiterten Realität zu finden. Während sie unterwegs sind unterhalten sie sich über das CINeSPACE HMD. Dazu muss erst eine Verbindung aufgebaut werden. Diese Bindung muss auch bestehen bleiben, wenn es sich die beiden Freunde bewegen. Da aber Karneval in Venedig ist, spielt an vielen Ecken Musik.Durch das HMD Verabreden sie sich schließlich in einem Restaurant.

Szenario 2:

Die 70 Jahre alte Alba, die in Venedig aufgewachsen ist, hat von dem neuen CINeSPACE Projekt gehört. Sie meldet sich freiwillig eine Tour durch Venedig zu machen und erzählt interessante Sachen über bekannte Orte. Das können geschichtliche Ereignisse, persönliche Erlebnisse oder aktuelle Geschehnisse sein. Diese werden auf einem zentralem Server System gespeichert.
Jahre später besichtigt Erwin Venedig. Während der Erkundungstour durch Venedig sieht er ein älteres Haus, welches sein Interesse auf sich zieht. Auf dem Display des CINeSPACE HMD sieht er, dass es zu diesem Gebäude eine Audioaufnahme vorhanden ist. Er spielt die

Audioaufnahme ab und hört dabei interessante Geschichten von der 70 jährige Alba über dieses Gebäude.

Dadurch ergeben sich folgende Aktionen welche das fertige Programm ausführen soll:

- Verbindungsaufbau
- Verbindungsabbau
- Daten aufnehmen
- gespeicherte Daten wiedergeben
- Daten persistent Speichern
- Daten an Hörer Versenden
- vom Sprecher gesendete Daten wiedergeben

Damit lassen sich die einzelnen Anforderungen zur besseren Übersicht den unterschiedlichen Benutzern zuordnen:

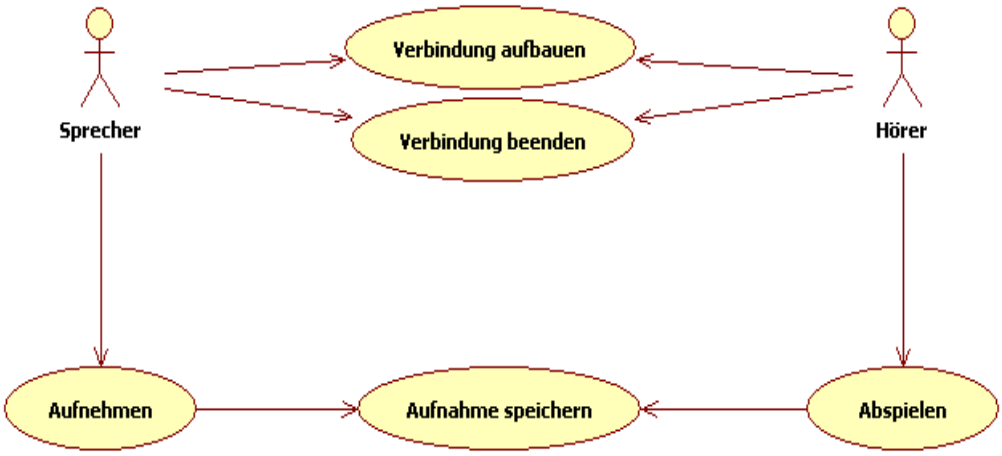
Sprecher

- Verbindungsaufbau
- Verbindungsabbau
- Daten aufnehmen
- Daten persistent speichern
- Daten an Hörer versenden

Hörer

- Verbindungsaufbau
- Verbindungsabbau
- gespeicherte Daten wiedergeben
- vom Sprecher gesendete Daten wiedergeben

Diese Anforderungen können wir alle Use Case Diagramm dargestellen (S*iehe Abbildung 12*).



Damit lassen sich nun folgende Anforderungen an das System ableiten:

- Plattformunabhängigkeit
- Lizenz: höchstens GNU LGPL
- benötigt nur wenige Bandbreite
- Verbindung zwischen zwei Usern
- Gespräche zwischen zwei Usern
- Daten persistent auf einem dezentralen Server speichern
- Daten vom dezentralen Server abspielen

Da nun die Anforderungen für das System stehen können wir nun einmal schauen ob es eventuell eine Software Applikation gibt, welche diese Anforderungen erfüllt.

4. Stand der Technik

In diesem Kapitel werden aktuelle Software Applikationen betrachtet und für das CINeSPACE Projekt ausgewertet. Damit sollen vorhandene Software näher betrachtet werden und eine treuere Eigenentwicklung vermieden werden. Wenn die Software nicht dafür geeignet ist, sehen wir zumindest was möglich ist.

Im Laufe der Recherche wurden insgesamt zwei verschiedene Arten an Programmen näher in Betracht gezogen. Dies während die Voice over IP Programme und die Streaming Programme.
Während Voice over IP Programm dazu benutzt werden um Echtzeitgespräche über das Internet oder LAN zu führen werden Streaming Programme dazu benutzt um Media Dateien von einem Host an viele Clients zu verteilen.

4.1 Voice over Internet Protocol Programme

Unter Voice over Internet Protocol, oder auch kurz VoIP genannt, versteht man die Sprachübertragung mithilfe des Internet Protokolls (IP). Das kann sowohl über das Internet geschehen oder in abgeschlossenen Netzwerken die das Internet Protokoll benutzen. In der heutigen Zeit wird Voice over IP fast ausschließlich für die Telefonie über das Internet benutzt. [SCHMITT04]

Vor Jahren war diese Art der Telefonie noch undenkbar gewesen. Damals war die Übertragungsrate des Internets noch nicht schnell genug gewesen, damit eine flüssige Übertragung der zeitkritischen Daten gewährleistet werden konnte. 2007 besitzen knapp 46% aller deutschen Haushalte einen Breitbandanschluss. Im Vergleich zu anderen Ländern ist das aber noch ein kleiner Prozentsatz (S*iehe Abbildung 14*). Da man mit Voice over IP über das Internet telefonieren kann erfreut sich diese Technik immer größerer Beliebtheit.

Man unterscheidet zwischen zwei Arten der Voice over IP Telefonie:

- über den Computer mithilfe von Software
- über das normale Telefon mithilfe von DSL Routern

In unserem Fall werden wir nur Computersoftware betrachten und analysieren.

4.1.1 Übersicht von VoIP-Programmen

Es gibt eine Reihe an freien aber auch kostenpflichtigen Voice over IP Programmen. Im Zuge dieser Arbeit wurden einige der bekanntesten Voice over IP Programmen angesehen und bewertet. Das sind Skype, Teamspeak, Mumble und Ventrilo.

4.1.1.1 Skype

Skype ist eine kostenlose Software, welche auf verschiedenen Betriebssystemen wie Windows, Linux oder Mac OS X läuft. Dieses Programm bietet viele Funktionen wie normale Telefonie, Texte schreiben und Konferenzschaltungen. Skype funktioniert nur über das Internet und nur über die Skype eigenen Server. Des weiteren benutzt Skype ein eigenes Protokoll zur Sprachübertragung. Dieses Protokoll ist kostenplichtig.
Aufgrund der guten Sprachqualität, der hohen Ausfallsicherheit und des günstigen Preises ist Skype bei Firmen und im privaten Bereich sehr beliebt.[SKYPE08]

4.1.1.2 Teamspeak

Teamspeak ist ebenso wie Skype eine proprietäre Software welche auch auf Windows, Linux und Mac OS X läuft. Dieses Programm wurde entwickelt um sich während Online Spielen zu unterhalten. Aus diesem Grund ist Teamspeak wurde auf kleine Übertragungsraten spezialisiert. Wie Skype benutzt es ein eigenes Übertragungsprotokoll und ist für VoIP Telefonate ungeeignet. Im Gegensatz zu Skype benötigt man einen eigenen Internetserver. Diesen kann man als Administrator beliebig konfigurieren. Man kann einen Server in beliebige Räume einteilen und diese auch mit Passwort schützen, so dass niemand unbefugtes in diesem Kanal kommt [TEAMSPEAK08].

4.1.1.3 Mumble

Genau wie Teamspeak kommt Mumble aus dem Computerspielbereich. Sie steht unter der GNU GPL und könnte deswegen für das CINeSPACE Projekt verwendet werden. Da Mumble aus dem Spielbereich kommt verfügt es über niedrige Latenzzeiten. Dies ist unter anderem dem Speex Codec zu verdanken. Wie die vielen anderen VoIP Programmen ist auch Mumble unter Windows und Linux lauffähig, nicht aber unter Mac OS X. Besonderheiten sind aber hierbei dass sich der Sprachverkehr komplett verschlüsseln lässt und die Möglichkeit besteht auf die Text-to-Speech API des Betriebssystems zuzugreifen. Unter Text-to-Speech versteht man die Möglichkeit geschriebene Texte in gesprochene Worte umzuwandeln. Unter Windows wird die Microsoft Text-to-Speech API und unter Linux Festival verwendet.[MUMBLE08]

4.1.1.4 Ventrilo

Ventrilo ist abermals eine proprietäre Software welcher aber nur unter Windows und Mac OS X funktioniert. Aber im Gegensatz zu den anderen proprietären Programmen gibt es von Ventrilo eine Freewareversion, die aber stark eingeschränkt ist. So gibt es eine Freewareversion, bei der man sich mit höchstens acht Personen pro Server unterhalten. Es besteht unter anderem die Möglichkeit die Pro Lizenz zu kaufen, allerdings kann diese Lizenz nur von großen Server Hosting Firmen erworben werden. [VENTRILO08]

4.1.2 Bewertung der VoIP-Programm

Für eine bessere Übersicht werden diese vier Programme in einer Tabelle aufbereitet und anschließend das Ergebnis besprochen.

Die ersten drei Zeilen sind die KO Kriterien. Wenn eine dieser drei nicht den Anforderungen entspricht können wir das Programm nicht für das CINeSPACE Projekt verwenden. Dieser drei Kriterien wären Lizenz, die Plattform und die Verbindungsart.

In den restlichen Zeilen werden sonstige Eigenschaften der Programme dargestellt.
Unter Konferenzen versteht man die Möglichkeit sich mit mehreren Nutzern gleichzeitig über das Programm zu unterhalten.
Mit der Verschlüsselung wird die Sprachübertragung Abhörsicher gemacht, in denen die einzelnen Pakete verschlüsselt und über ein sicheres Protokoll wie SRTP versendet wird.
Mit einer Videoübertragung kann man mithilfe einer angeschlossenen Videokamera Videoaufnahmen zwischen mehreren Rechnern versenden.
Text-to-Speech ist die Möglichkeit sich einen geschriebenen Text vom Computer vorlesen zu lassen.
Aufzeichnen befasst sich mit der Möglichkeit die geführten Gespräche sofort aufzuzeichnen und zu speichern.
Unter Bandbreite sind die durchschnittlich benötigten Bandbreiten der einzelnen Programme.
In der Zeile unterstützte Codecs werden die Audio Codecs aufgelistet, in dem Programm verwendet werden oder verwendet werden können.
Die letzte Angabe bezieht sich auf die Anforderungen die das Programm an den Computer stellt. Diese wurden nicht zu den KO Kriterien gezählt da der Samsung Q1 Ultra wohl in Zukunft durch eine leistungsfähigere Version ausgetauscht werden soll.

VoIP	**Skype 3.6**	**Teamspeak 2.0.32.60**	**Mumble 1.1.3**	**Ventrilo 3.0.1**
KO-Kriterien				
Lizenz	Proprietär ✗	Proprietär ✗	GNU GPL ✗	Proprietär ✗
Plattform	Windows, Linux, Mac OS X ✓	Windows, Linux, Mac OS X ✓	Windows, Linux ✗	Windows, Mac OS X ✗
Verbindungsart	Internet ✗	Internet, LAN ✓	Internet, LAN ✓	Internet, LAN ✓
Eigenschaften				
Konferenzen	Ja	Ja	Ja	Ja
Verschlüsselung	Ja	Nein	Ja	Nein
Videoübertragung	Ja	Nein	Nein	Nein
Text-to-Speech	Nein	Nein	Ja	Ja
Aufzeichnen	Nein	Nein	Nein	Nein
Bandbreite	3-16 kbit/s [1]	1-17 kbit/s	11 - 45.4 kbit/s	k.A.
Unterstütze Codecs	- SVOPC - AMR-WB - G.729 - G.711	- GSM 16.4kbit - CELP Windows 5.2kbit	- Speex	- GSM 6.10 - DSP True Speech - Speex
Anforderungen				
	1 GHz CPU 256 MB RAM 30 MB Festplatte	166 MHz CPU 64 MB RAM 6 MB Festplatte	Pentium III oder Athlon XP 1 GHz	k.A.

Tabelle6: Bewertung der VoIP-Programme

Wie man in Tabelle 3 sehen kann sind drei der vier Programme kommerzielle Anwendungen, nur eins befindet sich unter der GNU GPL. Da Skype, Teamspeak und Ventrilo von vornherein weg. Mumble wäre eine Alternative für das übertragen von Gesprächen, aber leider ist Mumble ist es nur unter Windows und Linux lauffähig nicht aber unter Mac OS X. Ein weiterer Grund, der gegen Mumble spricht ist die Hardware Anforderung. In dem CINeSPACE HMD ist ein Intel mit 800MHz eingebaut, Mumble benötigt aber 1 GHz.

Daher können wir keines dieser Programme für unser Projekt benutzen.

1 Keine offizielle Angabe

4.1.3 Audiodaten aufzeichnen

In Kapitel 3 haben wir unter anderen auch gesehen das wir für unser Projekt Audiodaten persistent auf einen Server aufzeichnen wollen. Bei den vier vorgestellten Voice over IP Programmen haben wir gleichzeitig darauf geachtet dass es eine Möglichkeit besteht de übertragene Sprache zu speichern.
Leider hat aber keines dieser vier Produkte daran gedacht einen Recorder zu integrieren.

4.2 Streaming Server

Unter einem Streaming Server versteht man einen Server der Streaming Media Daten über ein Netzwerk an mehrere Rechner, den so genannten Clients, versendet (*Siehe Abbildung 15*). Die Streaming Media Daten können Video Daten als auch Audio Daten sein. Dabei müssen wir zwischen Streaming-Media-Server und Streaming-Audio-Server als Serverprogramm unterscheiden. Während Streaming-Media-Server für die Übertragung von allgemeinen Media Inhalten, wie Video oder Audio verwendet werden, haben sich Streaming-Audio-Server auf die Übertragung von Audio Daten spezialisiert. Streaming Server werden dazu benutzt eine Media Datei gleichzeitig an viele Clients zu senden, damit diese Clients das Video oder die Audiodatei anziehen beziehungsweise anhören können.

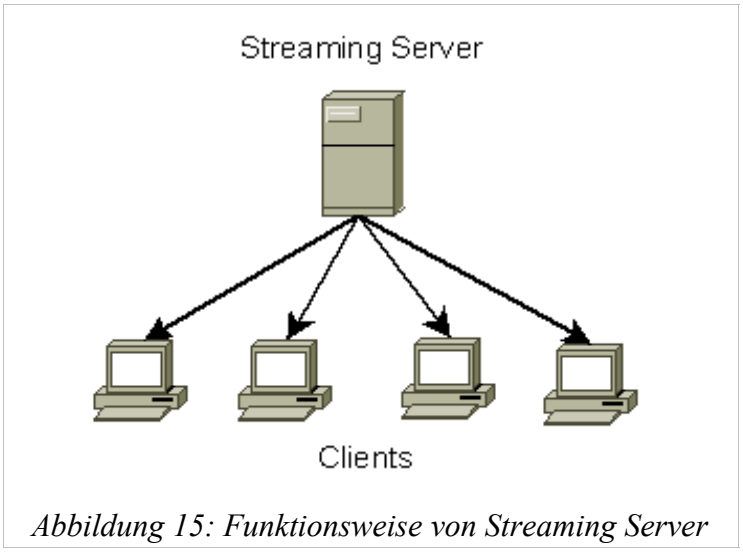


Abbildung 15: Funktionsweise von Streaming Server

Für das CINeSPACE Projekt ist das allerdings nicht nötig. Da sich viele Personen an unterschiedlichen Plätzen aufhalten, benötigten sie nicht alle gleichzeitig die gleiche Media Datei.

Da es sich aber um einen sehr ähnliches Verfahren handelt betrachten wir trotzdem einige Streaming Server, da vielleicht ein Streaming Server Programm unsere Anforderungen erfüllt.
In unserem Fall betrachten wir hier nur die Übertragung von Audioinhalten.

4.2.1 Übersicht von Streaming-Server Programmen

Um eine Übersicht zu bekommen werden wir zwei Streaming-Audio-Server und drei Streaming-Media-Server näher betrachten. Die Audioserver wären No23Live und Icecast. SHOUTcast, VLC media Player und Darwin Streaming Server sind die drei Media Server, die näher betrachtet werden.

4.2.1.1 No23Live

No23Live ist eine Software die mithilfe des Audio Codecs WMA von Microsoft das Streaming von Audiodateien ermöglicht. Dazu werden die Audiokanäle der Soundkarte ausgelesen und in das WMA Format umgewandelt. Dadurch entsteht der Nachteil, dass auf dem Server die Audiodatei über die Soundkarte abgespielt werden muss. Wenn nun zwei Personen zwei unterschiedliche Dateien wiedergeben wollen, so ist das nicht möglich. Der Empfänger kann mithilfe eines Audio Players diesen Audio Stream über das Internet empfangen und wiedergeben. Dieses Programm ist Freeware.[NO23LIVE08]

4.2.1.2 Icecast

Icecast ist ein freier Audio Streaming Server der von der Xiph.Org Foundation entwickelt wurde. Die Xiph.Org Foundation ist eine Organisation, welche sich dazu entschlossen hat freie Media Programme, Formate oder Protokolle zu entwickeln. Sie hat unter anderen die Audioformate Vorbis, Speex oder das Containerformat OGG entwickelt. Mit Icecast kann sowohl Audio über das Internet als auch über LAN übertragen. Um einen Streaming Server zu betreiben benötigt man einen Icecast Server und zum anderen eine Datenquelle, Ices genannt. Diese Ices sorgen dafür das die Audiodaten den Icecast Server übermittelt werden. Dadurch dass die Ices selbst programmiert werden können, kann man alle Audiodateien dem Icecast Server übermitteln. So gibt es Ices, Muse oder Darkice für Unix, Oddcast oder SAM2 für Windows und Nicecast für Mac OS X. Das Streaming erfolgt über das Audioformate Vorbis, Theora oder MP3.[ICECAST08]

4.2.1.3 SHOUTcast

SHOUTcast ist ein Streaming Server System der Firma Nullsoft. SHOUTcast ermöglicht das streamen von Audiodateien im Format MP3 und Videodateien im Format AAC+. Die Basisversion von ist SHOUTcast Server ist Freeware. Die einfache Verwendung sorgte dafür das SHOUTcast mittlerweile sehr weit verbreitet ist. Sehr viele Player unterstützen mittlerweile den Shoutcast Stream. Der SHOUTcast Server gibt es unter Windows und Linux, aber nich Mac OS X.[SHOUTCAST08]

4.2.1.4 VLC media Player

Der VLC Media Player wurde früher auch VideoLAN Client genannt. Inzwischen wurde er in VLC Media Player ungenannt und kann neben Media Dateien streamen auch als Abspielsoftware für alle möglichen Dateien benutzt werden. Dieses Projekt welches seit 1999 entwickelt wird steht unter der GNU GPL Lizenz und kann somit an eigene Entwicklungen angepasst werden. In diesem Media Player ist der so genannte Video LAN Server integriert, der eine große Anzahl verschiedenster Formate unterstützt und streamen kann.[VLCMEDIA08]

4.2.1.5 Darwin Streaming

Das Darwin Streaming Projekt ist ein von Apple entwickeltes Open Source Programm welche auf Apples Quicktime Streaming Server, kurz QTSS genannt, basiert. Das Programm wurde unter der APSL 2.0 Lizenz veröffentlicht und ist zur GNU GPL inkompatibel. Die Free Software Foundation, welche unter anderem die GNU GPL Lizenz beschrieben haben, sind der Meinung, dass man nach Möglichkeit keine eigenen Softwareentwicklungen unter dieser Lizenz herausbringen sollte.[FSFAPSL07] Das Darwin Streaming Projekt kann unter anderen Audio- und Videodateien in Formaten wie H.264/MPEG-4 AVC oder 3GP streamen. Während der Quicktime Streaming Server nur unter Mac OS X funktioniert, wurde das Darwin Streaming Projekt für andere Plattformen entwickelt, so zum Beispiel für Windows Server 2000 / Windows Server 2003 oder einigen Linux Distributionen wie Fedora. Außen vor bleibt aber Windows XP.[DARWINS08]

Für eine bessere Übersicht werden die Programme in einer Tabelle aufbereitet und anschließend besprochen.

4.2.2 Bewertung von Streaming Programmen

Hier werden alle fünf Streaming Programme in den Tabellen 7 und 8 auf den folgenden Seiten übersichtlich dargestellt und bewertet.

Wie auch bei den Voice over IP Programmen entsprechen die ersten drei Zeilen den KO Kriterien. Wenn eine dieser drei nicht den Anforderungen entspricht können wir das Programm nicht für das CINeSPACE Projekt verwenden. Dieser drei Kriterien wären Lizenz, die Plattform und die Verbindungsart.
Die Anzahl der Hörer werden als die maximale Anzahl angegeben die ein Media Stream gleichzeitig erreichen kann..
Unter Unicast versteht man eine Punkt-zu-Punkt Verbindung. Darunter versteht man dass der Streaming Server an einen einzigen Client die Streaming Daten versenden kann.
Das Gegenteil von Unicast ist der Multicast. Der Streaming Server versendet sie Audiodaten an alle Clients.
Unter Bandbreite stehen die durchschnittlichen Bandbreiten der einzelnen Programme.
Die Audiocodecs die unter Unterstütze Codecs stehen sind die Codecs, in dem die Audiodaten umgewnadelt und übertragen werden.

Für die Verwendung im CINeSPACE Projekt fallen von vornherein schon No23Live, SHOUTcast und Darwin Streaming aus. No23Live und SHOUTcast haben das Problem, dass sie nur als vorcompilierte Programme erhältlich sind. Bei Darwin Streaming ist die Lizenz inkompatibel zu unser geforderten Lizenz.

Die weiteren Alternativen wären Icecast und VLC Media Player. Gegen Icecast spricht dass es nicht komplett plattformunabhängig ist, sondern nur Windows und Linux unterstützt. Gegen den VLC Media Player spricht die geringe Zahl an Audio Codecs. Gerade MP3 und AC3 werden unterstützt und sind nicht gerade für gute Leistungen in Komprimierung für Sprache bekannt. Das wird in Kapitel 6 genauer besprochen.

Streaming	**No23Live 1.0.4.15**	**Icecast 2.3.1**	**SHOUTcast 1.9.8**
KO-Kriterien			
Lizenz	Freeware ✓	GNU GPL ✗	Freeware ✓
Plattform	Windows ✗	Windows, Linux ✗	Windows, Linux ✗
Verbindungsart	Internet ✗	Internet, LAN ✓	Internet ✗
Eigenschaften			
Anzahl der Hörer	Max. 50	k.A.	Über 10.000
Unicast	k.A.	k.A.	Nein
Multicast	Ja	Ja	Ja
Bandbreite	32 - 320 kbit/s	8 - 320 kbit/s	8 - 320 kbit/s
Unterstütze Codecs	WMA	Vorbis OGG MP3 Thedora	MP3 AAC+
Anforderungen			
	1 GHz CPU 512 MB RAM	k.A.	k.A.

Tabelle7: Bewertung der Streaming-Programme (I)

Streaming	**VLC Media Player 0.8.6e**	**Darwin Streaming 5.5.5**
KO-Kriterien		
Lizenz	GNU GPL 	APSL (Apple Public Source License) 
Plattform	Windows, Linux, Mac OS X 	Windows 2000 Server, Windows 2003 Server, Linux, Mac OS X Server 
Verbindungsart	Internet, LAN 	Internet 
Eigenschaften		
Anzahl der Hörer	k.A.	k.A.
Unicast	Ja	k.A.
Multicast	Ja	Ja
Bandbreite	8 - 320 kbit/s	8 - 320 kbit/s
Unterstütze Codecs	MP3 AC3	QuickTime MPEG-4 3GPP
Anforderungen		
	1 GHz CPU 512 MB RAM	k.A.

Tabelle8: Bewertung der Streaming-Programme (II)

5. Bestimmung einer geeigneten Audio Bibliothek

In diesem Kapitel werden einige Audio Bibliotheken näher betrachtet und anschließend bewertet.

5.1 Übersicht von möglichen Audio Bibliotheken

In diesem Kapitel werden einige mögliche Audio Bibliotheken näher betrachtet. Da es aber eine ganze Reihe von Audio Bibliotheken, gibt ist es unmöglich alle im Rahmen dieser Arbeit zu untersuchen. Daher muss man eine Vorauswahl machen. Als Kriterium für diese Auswahl sind die Anforderungen, die im Kapitel 3 erarbeitet wurden. Als mögliche Audio Bibliotheken könnte man folgende Software nehmen:

- Media Control Interface (MCI)
- Directshow
- Simple Directmedia Layer (SDL)
- Port Audio
- Allegro Game Programming Library
- FMOD
- RtAudio
- OpenAL
- Oreka
- Open Source Audio Libaray Project
- BASS

Diese 11 Bibliotheken sind mit dem bekanntesten Audio Bibliotheken die man in einen C++ Projekt verwendet werden könnte. Daher muss man sie sich doch etwas näher anschauen um anschließend sagen zu können ob eine Bibliothek die groben Anforderungen erfüllen kann.

Die **Simple Directmedia Layer** wird nicht weiter untersucht, da sie nur aufnehmen kann. Die **Port Audio** uns die **Oreka** Bibliothek können wir in unserem CINeSPACE Projekt nicht verwenden darf sie der GNU GPL unterliegt

Die **Allegro Game Programming Library** wird außen vor gelassen, da der Programmcode komplett auf die Verwendung in Videospielen ausgelegt ist. Diese Bibliothek kann zwar Audio Aufnehmen und wiedergeben aber ist mit zusätzlichen Funktionen wie Grafik total überladen.
Das **Open Source Audio Libaray Project** wird leider nur unter Linux entwickelt und ist somit auch nur darunter lauffähig.
Die **BASS** als auch die **FMOD** Bibliothek sind zwar für nichtkommerzielle kostenlos, aber für kommerzielle Projekte müssen Lizenzgebühren bezahlt werden. Ein weiterer Nachteil von BASS ist die fehlende Unterstützung von Linux. Aus diesen Gründen werden auch diese beiden Bibliotheken nicht weiter betrachtet.
Somit bleiben nur noch vier Bibliotheken übrig: Media Control Interface, Directshow, RtAudio und OpenAL. **MCI** und **Directshow** sind für Windows implementiert so da sie auch ausscheiden.

Während der etwa vier monatigen Praxisphase wurde etwa sechs Wochen mit Directshow verbracht, dazu diesem Zeitpunkt die Anforderung der Plattformunabhängigkeit noch nicht gegeben war. Wegen diesem Grund uns dem interessanten Konzepts, welches Directshow mit den so genannten Filtergraphen darstellt, wird es näher betrachtet.

5.1.1 Directshow

Directshow ist ein von Microsoft entwickelte Multimediabibliothek, die neben Audiodaten auch Videodaten verarbeiten kann. Früher war Directshow einen Teil von DirectX und war dort unter dem Namen Direct Media bekannt. Erst im späteren Verlauf wurde Direct Media aus dem DirectX Paket herausgenommen und in Directshow umbenannt. Somit ist Directshow nicht mehr ein Teil des DirectX Paketes. Somit kann man Directshow nicht mehr mit dem DirectX SDK installiert werden, sondern befindet sich nun in den jeweiligen Windows Plattform SDK. In diesen SDKs befinden sich alle Bibliotheken die man benötigt eine Directshow Anwendung zu kompilieren.
Directshow basiert auf dem objektorientierten Component Object Model (COM), welches ebenfalls von Microsoft entwickelt wurde. Mit diesem Modell wird es ermöglicht das man unter Windows zwischen mehreren Prozessen kommunizieren kann und dass man dynamisch Objekte erzeugen kann.

Directshow hatte es sich zur Aufgabe gemacht sehr große Datenmengen schnell verarbeiten, Audiodaten und Videodaten zu synchronisieren, möglichst viele unterschiedliche Datenquellen verarbeiten und möglichst viele Formate abspielen zu können.
Das alles wird mit dem so genannten Filtergraphen realisiert. Davon gibt es drei verschiedene Arten: zum einen die Source Filter, die Transform Filter und zuletzt den Rendering Filter (*Siehe Abbildung 16*).

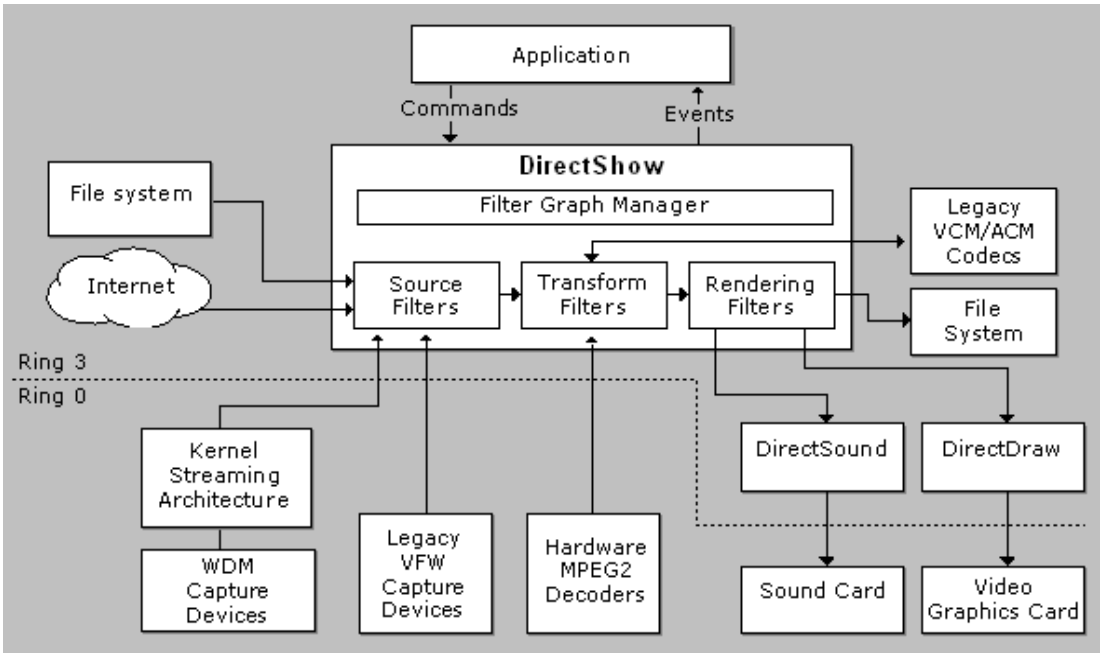


Abbildung 16: Kommunikation von Directshow Filtern [DS08b]

Damit wir diese Filter verwenden können muss in unserem Projekt zuerst ein so genannter Filter Graph Manager initialisiert werden. Dieser Manager überwacht sämtliche Schritte in den einzelnen Filtern, stellt eine Verbindung zur Applikation her und kann auf bestimmte Event reagieren und den Filtergraphen neu erstellen.
Wenn der Filter Graph Manager initialisiert ist, kann man einen oder mehrere Source Filter erstellen. Diese Filter ermöglichen es auf Dateien oder Datenströme zuzugreifen um sie im Transform Filter weiterverarbeiten zu können.
Diese Transform Filter wandeln die Datenströme beliebig um. Dazu muss nur ein entsprechender Filter vorhanden sein. Diese Umwandlung geschieht dadurch, dass ein Datenstrom einen oder mehrerer, hintereinander gelegte Transform Filter durchläuft. In diesen Filtern ist nun der Algorithmus wie die eingehenden Datenströme verarbeitet werden.

Nachdem die Daten verarbeitet worden sind, können sie noch an den Render Filter weitergeleitet werden.

In den Render Filter werden die Daten nun geschrieben oder ausgegeben. Das Schreiben auf die Festplatte erfolgt über so genannte File Writer. Für jedes Dateiformat muss ein eigener Writer geschrieben sein, der solche Zusatzinformationen wie den Dateikopf schreibt. Die Ausgabe aber die Soundkarte oder der Grafikkarte erfolgt mithilfe der DirectX eigenen Komponenten DirectSound und DirectDraw. [DS08a]

Die einzelnen Filter besitzen mindestens einen so genannten Pin. Dieser Pin kann sowohl ein Input- oder ein Outputpin sein. Die einzelnen Filter werden durch diese einzelnen Pins miteinander verbunden, jeweils ein Outputpin mit einem Inputpin. Jeder Filter gibt an in welchem Format der Datenstrom ausgegeben wird oder in welchen Datenformat er die Daten bekommen möchte. Somit ist es unmöglich zwei verschiedene Filter miteinander zu verbinden, die nicht über den gleichen Datentyp verfügen.

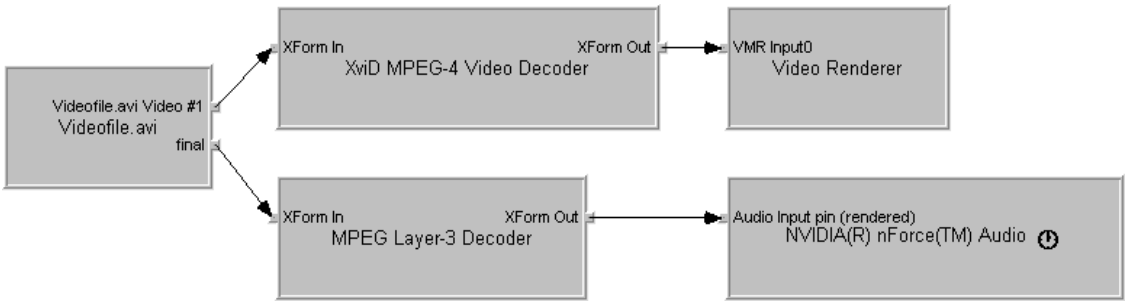


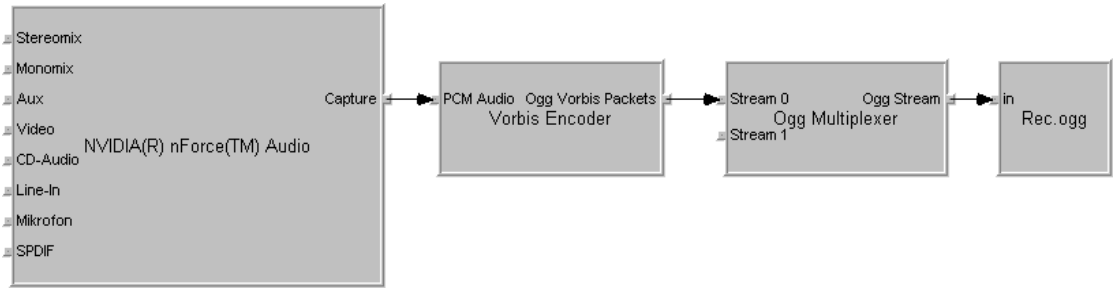
Abbildung 17: Videodatei anspielen mit Graphedit

In Abbildung 17 sieht man einen aufgebauten Filtergraphen in den Programm GraphEdit. Die Quelldatei wird mit dem Source Filter geladen und enthält ein Videostream und einen Audiostream. Diese müssen natürlich getrennt verarbeitet werden. Der Source Filter trennt diese beiden Ströme und übergibt sie jeweils einen Transform Filter. Dieser Filter decodiert nun diese eingehenden Datenströme und gibt sie anschließend einen Render Filter der sie dann der jeweiligen DirectX Komponente übergibt. Diese Komponenten sorgen dafür dass der Videostrom angezeigt und der Audiostrom abgespielt wird.

Da es für Directshow sehr viele verschiedene Filter gibt, ist es kaum möglich zu überblicken welcher Filter mit einem anderen Filter zusammenarbeiten kann. Dieses Problem hat

Microsoft schon sehr früh erkannt und aus diesem Grund das Programm GraphEdit entwickelt. Wie schon in Abbildung 17 zu sehen ist, kann man einen kompletten Graph aufbauen. Wenn man nun versucht zwei verschiedene Filter miteinander zu verbinden, die nicht zueinander kompatibel sind, so gibt das Programm eine Fehlermeldung aus. Wenn der komplette Graph aufgebaut ist kann man ihn testen. Wenn alles in Ordnung ist kann man diesen Graphen in C++ nachbauen.

Dies kann man an einem einfachen Beispiel demonstrieren:



Wir möchten über unsere Soundkarte das Mikrophon aufnehmen und es anschließend in Vorbis OGG abspeichern. In der Abbildung 18 sehen wir dazu den dazugehörigen Graphen. Der erste Kasten links zeigt dem Source Filter, der eine Verbindung zur Soundkarte erstellt. Der aufgenommene Audiostrom wird zum Vorbis Encoder weitergegeben, der die Audiodaten in das Vorbis Format umwandelt. Diese umgewandelten Daten werden nun dem Ogg Multiplexer übergeben, der die Daten in das Containerformat OGG packt. Wie man an dem Graphen erkennen kann Besitz der Ogg Multiplexer zwei Eingänge. An diesen Eingang könnte man eine Videoquelle, wie zum Beispiel eine Videokamera, integrieren, um damit eine OGM-Videodatei zu erstellen. Da mann diesen Eingang für eine Audiodatei nicht benötigt wird dieser Eingang einfach nicht belegt. Am Ende kann dieser Datenstrom einfach über einen File Writer Filter als OGG Datei gespeichert werden.

Die Programmierung ist, nachdem man diesen funktionstüchtigen Graphen erstellt hat, sehr einfach.

```
IGraphBuilder *pGraph;
IBaseFilter *pAudioInputFilter;
```

```
IBaseFilter *pVobEnc;
IBaseFilter *pOggMulti;
IBaseFilter *pFileWriter;
IFileSinkFilter *pSink;

// Initialize the COM library.
CoInitializeEx(NULL, COINIT_APARTMENTTHREADED);

// Create the filter graph manager and query for interfaces.
CoCreateInstance(CLSID_FilterGraph, NULL, CLSCTX_INPROC_SERVER,
                                          IID_IGraphBuilder, (void **)&pGraph);

//Create the Media Control Interface
pGraph->QueryInterface(IID_IMediaControl, (void **)&pControl);
```

Dazu wird zuerst der Filter Graph Manager erstellt werden. Dies geschieht durch das erzeugen des Objektes IGraphBuilder. Die einzelnen Filter sind vom Typ IBaseFilter. Für das Speichern wird noch ein Objekt vom Typ IFileSinkFilter benötigt.
Die erste Methode initialisiert die COM Schnittstelle. Der Filter Graph Manager wird der COM Schnittstelle in der zweiten Methode zugewiesenen. In der dritten Methode erfolgt eine Zuweisung auf das Media Control Interface, mit deren Hilfe man den Graphen steuern kann.

```
// Create AUDIO INPUT FILTER
EnumerateAudioInputFilters( (void **) &pAudioInputFilter);
EnumerateAudioInputPins(pAudioInputFilter);
pGraph->AddFilter(pAudioInputFilter, L"Capture");

// Create VORBIS ENCODER
AddFilterByCLSID(pGraph, CLSID_VorbEnc, L"Vorbis Encoder", &pVobEnc);
ConnectFilters(pGraph, pAudioInputFilter, pVobEnc);

// Create OGG MULTIPLEXER
AddFilterByCLSID(pGraph, CLSID_OggMultiplexer, L"Ogg Multiplexer", &pOggMulti);
ConnectFilters(pGraph, pVobEnc, pOggMulti);

// Create FILE WRITER
AddFilterByCLSID(pGraph,CLSID_FileWriter, L"File Writer", &pFileWriter);
pFileWriter->QueryInterface(IID_IFileSinkFilter, (void**)&pSink);
pSink->SetFileName(L"C:\\Rec.ogg", NULL);
ConnectFilters(pGraph, pOggMulti, pFileWriter);
```

Danach werden nur noch die einzelnen Filter erzeugt und miteinander verbunden. Zuerst wird dabei der Source Filter, in unserem Beispiel der Audio Input Filter der Soundkarte, aktiviert und mit dem Befehl pGraph->AddFilter(*IBaseFilter, LPCWSTR) in unseren Filter Graph Manager eingetragen. Der erste Parameter übergibt dabei den Zeiger auf unserer

Audio Input Filter und der zweite Parameter ist nur ein Bezeichner für einen Namen, den man beliebig wählen kann.
Danach werden die beiden Transform Filter integriert. Dies geschieht jeweils über die Funktion AddFilterByCLSID(*IGraphBuilder, const GUID& , LPCWSTR, *IbaseFilter). Diese Funktion ist keine Funktion die von Directshow zur Verfügung gestellt wird, sondern muss zusätzlich in das Programm eingebunden werden.[DS08c] Der erste Parameter gibt an bei welchem Filter Graph Manager der Filter integriert werden soll. Der zweite Parameter enthält eine Globally Unique Identifier (GUID) und ist eine 128 Bit große Zahl die weltweit eindeutig ist. Diese Bitfolge ermöglicht es dem Programm den richtigen Filter auszuwählen. Der Name des Filters kann im dritten Parameter angegeben werden und im vierten Parameter wird der Zeiger auf das Objekt übergeben.
Die nächste Funktion ConnectFilters(IGraphBuilder, *IBaseFilter, *IBaseFilter) dient dazu zwei Filter miteinander zu verbinden. Wie auch die vorherige Funktion muss sie in das Programm eingebunden werden. [DS08d] Der erste Parameter zeigt auf das Objekt des Graphen, der zweite und dritte Parameter zeigt auf die beiden Filter die miteinander verbunden werden sollen.
Zum Schluss wird noch der Render Filter, bei dem es sich hierbei um einen File Writer handelt, eingebunden. Dabei werden ebenso wie bei den Transform Filtern die beiden Funktionen benutzt um einmal den Filter in den Graphen zu integrieren und zum anderen den Filter mit dem letzten Transform Filter zu verbinden. Die beiden zusätzlichen sind dazu nötig um den File Writer den Speicher Ort mitzuteilen.

Damit wäre der komplette Graph erzeugt worden. Dieses Beispiel befindet sich noch in ausführlicherer Form auf der beiliegenden CD als ein Visual Studioprojekt unter Programme\DirektShow_Beispiel.
Problematisch bei Directshow wird es wenn kein Filter für einen Codec oder ähnlichem vorhanden ist, denn dann muss dieser selbst geschrieben werden. Das würde zu weit von dieser Arbeit abweichen, sollte aber trotzdem kurz erwähnt werden. In der Windows Plattform SDK als auch auf der MSDN Internetpräsenz von Microsoft sind Beispiele dafür zu finden.

5.1.2 RtAudio

Das RtAudio Projekt stammt von Gary P. Scavone, ein Professor Gehilfe für Musiktheorie an der berühmten McGill Universität in Kanada. Gegen Ende des 20. Jahrhunderts machte Scavone seinen Doktor in Computer basierte Musiktheorie an der Stanford Universität, Kalifornien. In seiner Freizeit arbeitet er an verschiedenen Softwareprojekten. So erstellte er das Programm RtFFT, RtMidi, The Synthesis ToolKit und eben RtAudio. Scavone stellt diese Programme mit dem Sourcecode komplett zur freien Verfügung.
RTAudio ist in C++ geschrieben und ist streng objektorientiert. Sie kann in Windows, Linux und Mac OS X eingesetzt werden.
In den jeweiligen Betriebssystemen unterstützt sie sogar verschiedene Soundschnittstellen. Unter Linux kann man RTAudio über OSS, ALSA oder Jack mit der Soundkarte kommunizieren lassen. In Mac OS X sind es noch CoreAudio und ebenfalls Jack. Bei Windows kann man über DirectSound und ASIO eine Aufnahme und Wiedergabe ermöglichen. RTAudio unterstützt sogar die simultane Verwendung zweier Audioschnittstelle.
Diese Bibliothek unterliegt keiner Lizenz. Die Lizenz besagt das man RTAudio komplett zur freien Verfügung hat. Man darf sie beliebig verwenden, kopieren, verändern oder veröffentlichen. Somit wäre diese Audio Bibliothek mit unserer gewünschten Lizenzvereinbarungen kompatibel. [RTAUDIO08]

Für die Bibliothek werden insgesamt nur drei Dateien benötigt. Diese wären:

- RtAudio.h
- Rtaudio.cpp
- RtError.h

Die RtError.h ist eine einfache Klasse zur Anzeige von Fehlern in den Programmen RtAudio und RtMidi. Die eigentliche Syntax befindet sich in den Dateien RtAudio.h und RtAudio.cpp.
Für die Portierung in ein anderes Betriebssystem oder einer Soundschnittstelle ist sehr einfach gehalten. Dazu muss in der RtAudio.h eines der folgenden Defines angelegt werden.

```
#define __UNIX_JACK__ true
#define __LINUX_ALSA__  true
#define __LINUX_OSS__  true
#define __WINDOWS_ASIO__  true
#define __WINDOWS_DS__  true
#define __MACOSX_CORE__ true
#define __RTAUDIO_DUMMY__ true
```

Die Soundschnittstelle Jack läuft sowohl unter Linux als auch Mac OS X. Das Define der Schnittstelle __RTAUDIO_DUMMY__ ist ein Platzhalter, der verwendet werden kann wenn man noch keine Schnittstelle ausgewählt wurde. Die Verwendung der eigentlichen Audio Bibliothek ist auch relativ einfach.

```
if (!DeviceExist())                 //search for Sound Device
        return false;               //if not exist, return false

//Input Device
iParams.deviceId = 0;               // first available device
iParams.nChannels = 1;              //set Number of Channels
//OutputDevice
oParams.deviceId = 0;               // first available device
oParams.nChannels = 1;              //set Number of Channels

try {
        adc.openStream(
                &oParams,                   // outputParameters,
                &iParams,                   // inputParameters,
                RTAUDIO_SINT32,             // format,
                44100,                      // SampleRate,
                &bufferFrames,              // bufferFrames,
                &inout,                     // CallbackFunktion
                (void *)&bufferBytes );     // userData

        bufferBytes = bufferFrames * 2 * 4;

        adc.startStream();                  //start the Stream
}
catch ( RtError& e ) {
        e.printMessage();
        return false;
}
```

Die Methode DeviceExist() überprüft ob mit der gewünschten Soundschnittstelle eine Soundkarte angesprochen werden kann.
Die beiden Objekte iParams und oParames sind ein von RtAudio konzipiertes Struct-Element. Die beiden stehen für den eingehenden und ausgehenden Datenstrom. Mit setzen

der deviceid auf 0 wird die erste freie Soundkarte angesprochen, die im System gefunden wird. Anschließend wird der Wert nChannels auf 1 gesetzt. 1 ist dabei für Mono und 2 für Stereo.
In der try-catch Anweisung erfolgt das Öffnen eines Datenstroms. Mit der Methode openStream(RtAudio::StreamParameters *outputParameters, RtAudio::StreamParameters *inputParameters, RtAudioFormat format, unsigned int sampleRate, unsigned int *bufferFrames, RtAudioCallback callback, void *userData, RtAudio::StreamOptions *options) werden die notwendigen Informationen übergeben.

Die ersten beiden Parameter sind die beiden Input und Output Ströme. Der dritte Parameter gibt das Audioformat an, in welchem die beiden Ströme übergeben werden. Für diesen Parameter kann man insgesamt sechs verschiedene Formate übergeben. Diese wären:

- RTAUDIO_SINT8 : 8-bit signed integer.
- RTAUDIO_SINT16 : 16-bit signed integer.
- RTAUDIO_SINT24 : Upper 3 bytes of 32-bit signed integer.
- RTAUDIO_SINT32 : 32-bit signed integer.
- RTAUDIO_FLOAT32 : Normalized between plus/minus 1.0.
- RTAUDIO_FLOAT64 : Normalized between plus/minus 1.0.

Der vierte Parameter ist die Abtastrate im Hz gemessen. Der fünfte Parameter ist ein Zeiger auf die Anzahl der Frames. Der sechste Parameter ist der eigentliche Aufruf der Verarbeitungsmethode. In dieser globalen Methode kann der Audiodatenstrom verarbeitet werden, also mithilfe eines Codecs codiert werden. Diese Methode wird im hinteren Teil dieses Kapitels näher erklärt. Der siebte Parameter ist ein Zeiger auf die von der Soundkarte eingegangenen Daten. Der achte und letzte Parameter ist optional. Dieser ist wie die beiliegende Input und Output Parameter ein von RtAudio definiertes struct Elemet. In diesen kann man den Audiodatenstrom einer Nummer zu weisen oder für die Soundschnittstelle Jack einen Namen.
Danach wird die Größe des Buffers berechnet. Sie setzt sich zusammen aus der Anzahl der Frames multipliziert mit Anzahl der Datenströme, in unserem Fall zwei einmal für Input und einmal für Output, und das wiederum multipliziert mit der Größe des Formates in Bytes, da

hier mit einem 32 Bit im Integer gearbeitet wird benötigen können wir 4 Bytes.

Anschließend kann in der Methode startStream() die Aufnahme und/oder die Wiedergabe beginnen. Es wird jetzt jedes Mal wenn der Buffer voll ist die globale Methode aufgerufen, die in openStream() angegeben worden ist. In dieser Methode würde die Verarbeitung des Codecs erfolgen. Da aber noch kein Codec vorhanden ist wird eine einfache Funktion realisiert.
Um zu sehen ob die Audio Bibliothek funktioniert wird mit dem Mikrophon aufgenommenen und anschließend sofort wieder auf den Lautsprecher ausgegeben.

```
void inout( void *outputBuffer, void *inputBuffer, unsigned int nBufferFrames,
                    double streamTime, RtAudioStreamStatus status, void *data )
{

        if ( status ) cout << "Stream over/underflow detected." << endl;
        unsigned long *bytes = (unsigned long *) data;
        memcpy( outputBuffer, inputBuffer, *bytes );
}
```

Die if-Abfrage dient dazu Probleme mit dem Datenstrom zu erkennen. Gegebenenfalls kann man darauf reagieren. In der nächsten Zeile wird die Größe des Buffers in einem Zeiger des Typs unsigned long umgewandelt, damit wir ihn in der dritten Zeile verwenden können. Die Funktion memcpy() macht nun nichts mehr anderes als den inputBuffer auf dem outputBuffer zu schreiben.

```
try {
        adc.stopStream();                       // Stop the stream
 }
 catch (RtError& e) {
        e.printMessage();
 }

 if ( adc.isStreamOpen() ) adc.closeStream();       // Close the Stream
```

Mit den beiden Methoden stopStream() und closeStream() wird ie Anwendung angehalten. Zuerst bewirkt stopStream() das die Aufnahme und/oder die Wiedergabe beendet wird.der Datenstrom wird mit closeStream() beendet.

Das komplette Programm befindet sich als Visual Studio Projekt auf der CD in dem

Verzeichnis Programme\RTaudio_Beispiel.

5.1.3 OpenAL

Die von Loki Software entwickelte Audio Bibliothek OpenAL ist auf die Verwendung in 3D-Anwendungen spezialisiert. Sie kann mit einem 5.1 Lautsprechersystem echtem Raumklang wiedergeben. Um das zu ermöglichen sind insgesamt sechs Lautsprecher im Raum verteilt. Vier Lautsprecher befinden sich im vorderen und zwei im hinteren Bereich des Raumes. Anwendungen die diese Form der der Akustik sind hauptsächlich Filme und Computerspiele. Vor allem im Spielbereich ist OpenAL weit verbreitet und wird in Spielen wie Bioshock, Quake 4, Psychonauts oder Unreal 2 eingesetzt. Die Verwendung von Raumklang ist im CINeSPACE Projekt nicht vorgesehen. Die Bibliothek kann aber auch normale Stereo Audiodaten wiedergeben.
OpenAL liegt aktuell in der Version 1.1 vor und unterstützt die Systeme von Windows, Linux und Mac OS X.
Bei Windows kann OpenAL über die Soundschnittstellen DirectSound und DirectSound3D kommunizieren. Aber auch Hardware spezifische Schnittstellen wie NVIDIA nForce für Nvidia Soundkarten und Creative Audigy für Creative Soundkarten werden unterstützt.
Linux wird wird wieder über OSS und ALSA unterstützt und die Mac OS Betriebssysteme über Carbon Sound Manager und Core Audio.
Leider besitzt OpenAL keine einheitliche Lizenz. In Tabelle 9 kann man die Lizenzen für die einzelnen Schnittstellen entnehmen

Betriebssystem	Soundschnittstelle	Lizenz
Mac OS 8 / 9	Carbon Sound Manager	GNU LGPL
Mac OS X	Core Audio	Open Source (Apple)
Linux	OSS	GNU LGPL
	ALSA	GNU LGPL
Windows	DirectSound	GNU LGPL
	DirectSound3D	GNU LGPL
	NVIDIA nForce	GNU LGPL
	Creative Audigy	Creative Labs Inc.
	Creative Audigy 2, 4	Creative Labs Inc.

Tabelle9: Lizenzen für OpenAL [OPENAL08]

Wie man sieht sind alle wichtigen Schnittstellen unter der GNU LGPL. Nur die Schnittstelle Core Audio ist unter einer speziellen Apple Lizenz herausgegeben. Diese kann man unter Mac OS X beliebig verwenden.
OpenAL ist ursprünglich in C geschrieben, wurde aber auch in andere Programmiersprachen portiert.
Als Grundlage für den Programmierercode wurde ein Code Beispiele aus dem OpenAL 1.1 Programmer's Guide verwendet. In diesem Beispielmit eine WAV Datei geöffnet und abgespielt.

```
// Initialization
Device = alcOpenDevice(NULL); // select the "preferred device"
if (Device)
{
        Context=alcCreateContext(Device,NULL);
        alcMakeContextCurrent(Context);
}
```

Die Funktion alcOpenDevice(PALCdevice deviceName) öffnet eine Audioschnittstelle. Als Parameter kann der Name eines speziellen Audiogerätes mitangegeben werden. Mit alcOpenDevice('DirectSound3D') verwendet man die Schnittstelle von DirectX. Wenn kein Parameter übergeben wird, verwendet OpenAL das Standard Audiogerät. Als Rückgabewert bekommt man einen Zeiger auf das Gerät.
Wenn ein Audiogerät vorhanden ist wird in der if Anweisung mit der Funktion alcCreateContext(ALCdevice *device ,ALCint *attrlist) wird einen Kontext erzeugt in dem besondere Eigenschaften einem Audiogerät zugewiesen werden. Der erste Parameter ist der Zeiger auf das Gerät, der zweite Parameter ist eine Attributliste, mit der man einstellen kann ob es sich dabei um eine Stereo Quelle oder eine Mono Quelle handelt. Dieser Parameter kann auch Null sein.
Die Methode alcMakeContext(PALCcontext context) wird ein Kontext als aktueller Kontext gesetzt.

```
// Generate Buffers
alGetError(); // clear error code
alGenBuffers(1, g_Buffers);
if ((error = alGetError()) != AL_NO_ERROR)
{
        DisplayALError("alGenBuffers :", error);
        return;
```

```
}
```

Anschließend muss ein Buffer erzeugt werden in dem die Daten gespeichert werden.
Die Methode alGetError() setzt die Ausgabe für die Fehlermeldung zurück.
Mit glGenBuffer(ALsizei n, PALuint *buffers) wird ein Buffer erzeugt. Der erste Parameter steht für die Anzahl der zu erstellenden Buffer. Die Anzahl wird als Array gespeichert. Der zweite Parameter ist ein Zeiger in dem der Buffer gespeichert werden sollen.
Die Funktion alGetError(), die in der if Anweisung aufgerufen wird, gibt den aktuellen Status zurück. Wenn ein Fehler in der vorangegangenen Methode angefallen ist so würde sie einen Fehlercode zurückgeben.
DisplayALError(string n, TALCenum error) hat die Aufgabe den Fehler auf dem Display anzuzeigen. Da diese Methode in den folgenden Programmteilen immer wieder einmal auftaucht wird diese nicht mehr explizit erwähnt.

```
// Load test.wav
alutloadWAVFile("test.wav",&format,&data,&size,&freq,&loop);
if ((error = alGetError()) != AL_NO_ERROR)
{
        DisplayALError("alutLoadWAVFile test.wav : ", error);
        return;
}
```

Wenn der Buffer erzeugt ist kann die Wavedatei gelesen werden.
Die Methode alutloadWAVFile(ALbyte *filename, ALenum *format, void **data, ALsizei *size, ALsizei *frequency, ALboolean *loop) gehört nicht zur OpenAL Bibliothek, sondern zur Alut, der so genannten Audio Library Utility Toolkit.
Der erste Parameter ist der Pfad in dem die Wave Datei zu finden ist. Das Format dieser Datei wird als zweiter Parameter übergeben. Der Pointer der auf diese Datei zeigt wird als dritter Parameter verwendet. Die Größe der Datei wird im vierten Parameter und die Frequenz wird als fünfter Parameter übergeben. Der sechste Parameter ist zuständig für eine permanente Wiedergabe dieser Datei.

```
// Copy test.wav data into AL Buffer 0
alBufferData(g_Buffers[0],format,data,size,freq);
if ((error = alGetError()) != AL_NO_ERROR)
{
        DisplayALError("alBufferData buffer 0 : ", error);
        return;
```

```
}
```

Anschließend kann der Buffer mit den Audiodaten gefüllt werden. Das erfolgt über die Methode alBufferData(TALuint buffer, TALenum format, Pointer data, TALsizei size, TALsizei frequenz). Der Buffer wird als erster Parameter übergeben, wobei man ihn als Array übergibt. Danach wird das Format übergeben, in dem die Audiodaten vorliegen. Das können sein:

- AL_FORMAT_MONO8
- AL_FORMAT_MONO16
- AL_FORMAT_STEREO8
- AL_FORMAT_STEREO16

Als dritter Parameter wird ein Zeiger auf die Audiodaten verwendet. Die Größe dieser Audiodaten in Bytes ist der vierte Parameter und die Frequenz der fünfte Parameter.

```
// Unload test.wav
alutunloadWAV(format,data,size,freq);
if ((error = alGetError()) != AL_NO_ERROR)
{
        DisplayALError("alutUnloadWAV : ", error);
        return;
}
```

Nachdem die Wave Datei gelesen wurde und im Buffer gespeichert wurde, kann die Datei wieder geschlossen werden. Das erfolgt wie das Öffnen über die ALUT Methode alutunloadWAV(ALenum *format, void **data, ALsizei *size, ALsizei *frequency). Das Format wird als erster Parameter übergeben, der Zeiger auf die Audiodaten als zweiter, die Größe in Bytes als dritter und die Frequenz als vierter Parameter. Als nächstes muss der Buffer einer Quelle zugewiesen werden. Das ist notwendig da OpenAL eine 3D Audio Bibliothek ist und jede Quelle eine Position in einem drei dimensionalen Raum einnehmen muss.

```
// Generate Sources
alGenSources(1,source);
if ((error = alGetError()) != AL_NO_ERROR)
{
```

```
        DisplayALError("alGenSources 1 : ", error);
        return;
}
```

Ähnlich wie glGenBuffer(ALsizei n, PALuint *buffers) generiert die Methode alGenSources(ALsizei n, PALuint *sources) eine Quelle. Der erste Parameter ist die Anzahl der Quellen die erstellt werden und der zweite Parameter ist ein Zeiger auf ein Array von Quellen. Anschließend könne man die Quelle einen Standort im 3D Raum zu weisen, da das für das CINeSPACE Projekt nicht notwendig ist wird es darauf nicht näher eingegangen. Als nächstes weisen wir der Quelle einen Buffer zu.

```
// Attach buffer 0 to source
alSourcei(source[0], AL_BUFFER, g_Buffers[0]);
if ((error = alGetError()) != AL_NO_ERROR)
{
        DisplayALError("alSourcei AL_BUFFER 0 : ", error);
}
```

Die Methode alSourcei(ALuint source, ALenum param, ALint value) macht genau das. Der erste Parameter ist die Quelle, der zweite Parameter gibt an das es sich dabei um einen Buffer handelt, und der dritte Parameter ist der Buffer. Nun kann der die Datei abgespielt werden.

```
// Play WAV File
alSourcePlay(source[0]);
if ((error = alGetError()) != AL_NO_ERROR)
{
        DisplayALError("alSourceiPlay : ", error);
}
```

Die Wiedergabe der Audiodatei erfolgt über die Methode alSourcePlay(ALuint source) wo als Parameter die entsprechende Quelle angegeben wird. Nachdem die Audiodatei abgespielt wurden ist kann man das Programm beenden und den Speicher aufräumen.

```
// Exit
alcDestroyContext(Context);
alcCloseDevice(Device);
```

Dabei muss nur mit der Methode alcDestroyContext(PALCcontext context) der aktuelle Kontext zerstört werden uns mit der Methode alcCloseDevice(PALCdevice deviceName) die Audioschnittstelle geschlossen werden.[OPENAL08]

5.2 Wahl der Audio Bibliothek

Welche der beiden Audio Bibliotheken RtAudio und OpenAL für das CINeSPACE Projekt am besten geeignet ist müssten erst genauere Tests zeigen. Vor allem müsste beide Audio Bibliothek mit einem entsprechenden Audio Codec ausgestattet werden undauf ihre Leistung auf dem CINeSPACE HMD getestet werden.

Aufgrund von terminlichen Problemen konnten die beiden Audio Bibliotheken nicht mit einem Codec ausgestattet werden und somit konnte kein Test mehr ausgeführt werden.

Aufgrund des Sourcecodes der in den vorangegangenen Kapiteln erklärt wurde kann man eine Prognose abgeben.
RtAudio ist sehr kompakt und übersichtlich. OpenAL hingegen wirkt recht aufgebläht und mit sehr vielen Funktionen überladen. Diese Funktionen, wie zum Beispiel den 3D Sound, werden für das Projekt nicht benötigt und stellen somit überflüssigen Ballast da.
Daher scheint RtAudio für das CINeSPACE Projekt am ehesten geeignet.

6. Bestimmung eines geeigneten Audiocodec

In diesem Kapitel beschäftigen wir uns mit der Suche nach einem geeigneten Audio Codec. Dazu werden wir uns zuerst einmal eine Reihe von Codecs näher anschauen und anschließend einigen Tests unterziehen.

6.1 Übersicht der möglichen Audiocodecs

Im Laufe dieser Bachelorarbeit wurden insgesamt drei verschiedene Audio Codecs näher betrachtet. Es nur insgesamt zwei Codecs betrachtet, die auf menschliche Sprache zugeschnitten sind, und zwei Codecs die auf Musik spezialisiert sind. Zum einen fiel die Wahl auf das sehr bekannte und beliebte MP3 Format. Des weiteren wurde das OGG Format und das Speex Format der Xiph.Org Foundation näher untersucht. Auch wurde der GSM Audio Codec näher betrachtet. Er er wird heutzutage für die Sprachübertragung bei Handys verwendet.

6.1.1 MP3

MP3 ist ein verlustbehafteter Audio Codec, der auch unter dem Namen MPEG-1 Audio Layer 3 bekannt ist. Die ersten Grundlagenforschungen reichen bis ins Jahr 1982 zurück. Am Fraunhofer-Institut für Integrierte Schaltungen in Erlangen entwickelte Dr. Karl-Heinz Brandenburg in Zusammenarbeit mit den Firmen AT&T Bell Labs und Thomson dieses Audioformat. Dem Kürzel MP3 bekam dieser Audio Codec aber erst im Jahre 1995.
Die Fraunhofer-Gesellschaft und die Firma Thomson haben sich dieses Format patentieren lassen.
Die Kodierung ist asymmetrisch, das bedeutet das für die Kodierung mehr Aufwand betrieben werden muss, als für das decodieren.
Dieser Codec macht sich der der Tatsache zu Nutze, dass der Mensch wie in Kapitel 2.3 beschrieben nicht alle Töne wahrnehmen kann. Des weiteren verwendet MP3 Erkenntnisse der Psychoakustik. [HARTMANN99]

Das heißt auch Maskirung oder Überdeckung. Bei dem menschlichen Gehör gibt es eine Temporale Maskierung und eine Gleichzeitige Maskierung.
Nach einem sehr lauten Geräusch dauerte es etwa 50 bis 300 ms bis man leise Geräusche

wieder wahrnehmen kann. Das nennt man Temporale Maskierung.
Eine Schallquelle erzeugt ein kontinuierliches Signal mit einer Frequenz von 1 kHz. Ein etwas leiseres Geräusch, dass eine Frequenz von 1,1 kHz aufweist, wird von dem ersten Signal überlagert und ist für das menschliche Gehör nicht wahrzunehmen. Diesen Fall nennt man Gleichzeitige Maskierung, da die beiden Signale gleichzeitig existieren.
Bei dem MPEG Layer 3-Algorithmus ist aber nur die Gleichzeitige Maskierung implementiert.
Damit man diese unnötigen Daten erkennen kann wird zum Zeitpunkt t eine so genannte Maskierungskurve errechnet unter der alle nicht hörbaren Audiosignale liegen.

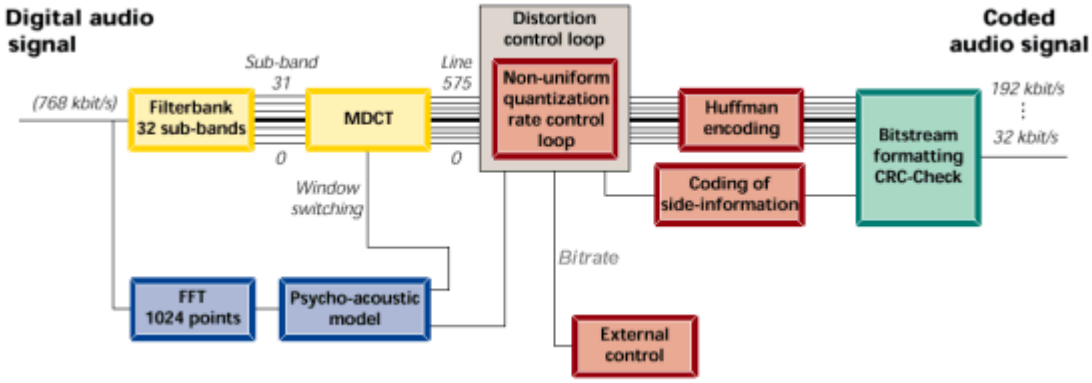


Abbildung 19: Aufbau des MPEG Layer 3 Encoder [BBURG00]

In Abbildung 19 sieht man den groben Aufbau des MPEG Layer 3 Encoders. Die gelben Blöcke sind für die Umwandlung in die Frequenzdarstellung zuständig. Die beiden blauen Blöcke berechnen das psychoakustische Modell. Die eigentliche Kodierung erfolgt in den vier roten Blöcken. Die abschließende Sicherung der Daten mittles CRC (Cyclic Redundancy Code) soll MP3 vor Datenfehlern schützen.

Das digitale Signal wird erst in so genannte Frames eingeteilt. Ein Frame bestehen aus 1152 Samples. Diese Signale werden durch eine Polyphase Filterbank in 32 parallele Subbänder aufgeteilt. Jedes dieser Subbänder filtert dabei einen bestimmten Frequenzbereich heraus. Dabei werden die ersten 32 Samples im Frequenzbereich umgewandelt. Danach die nächsten 32 und so fort, bis alle 1152 Samples auf die einzelen Bänder verteilt wurden. Dadurch entstehen 36 Frequenzspektren die in den 32 Subbändern stehen.

Anschließend werden die Frequenzspektren durch eine modifizierte diskrete Cosinustransformation (MDCT) umgewandelt. Diese MDCT ist eine Zeit-Frequenz-Transformation, die eindimensionalen Daten verarbeiten transformiert. Bei der MDCT überlappen sich an aufeinander folgende Frames um die Hälfte, dadurch sollen plötzliche Übergänge an den Transformationsblockgrenzen vermieden werden.

Die Frequenzlinie eines Subbandes wird in mehrere Frequenzlinien aufgespalten. Bei der Transformation macht man sich der Tatsache zunutze, das sich die Frames um die Hälfte überschneiden. Die 36 Frequenzspektren kann man auf 18 Spektren reduzieren, da man sie aus den vorangegangenen und nachfolgenden Frame verrechnen kann. Am Ende hat man nun ein Spektrum von 576 (= 32 Subbänder * 18 Frequenzspektren) Frequenzlinien.

Da das Frequensspektrum die Zeitspanne aller 1152 Eingangssamples enthält wird die Zeitauflösung entsprechend schlechter, die dann zu Folge haben kann das ein Pre-Echo-Effekt auftritt. Dieser Pre-Echo-Effekt sind Störgeräusche die durch die Transformation vor dem eigentlichen Signal entstehen. Es tritt auf, wie man in der Abbildung 20 erkennen kann, bei einer sehr starken Lautstärkeänderung. Es wird erzeugt wenn die Lautstärke gegen Ende eines Frames sprunghaft ansteigt. Dadurch berechnet der Encoder eine höhere Maskierungskurve für das komplette Frame und damit dieses Pre-Echo.

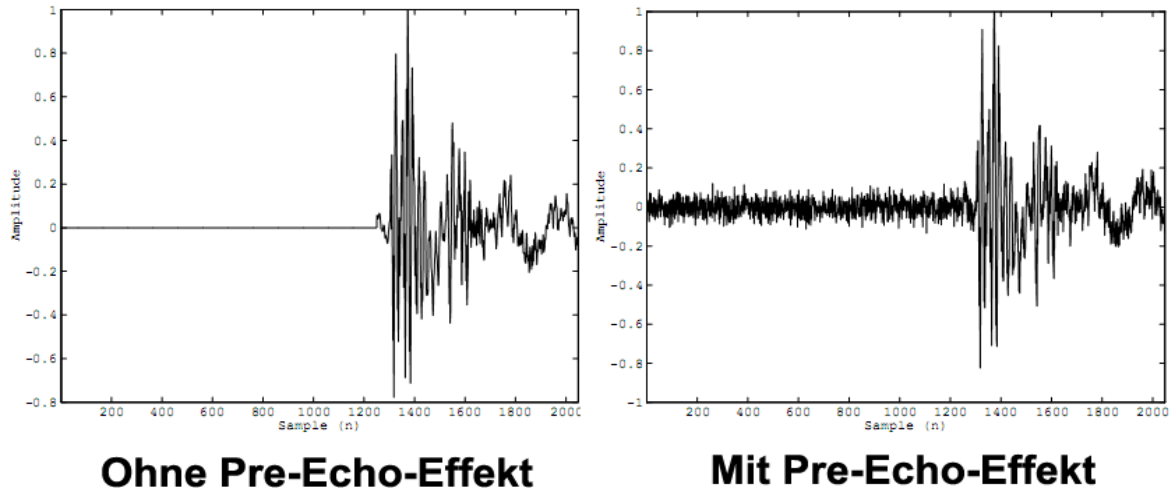


Abbildung 20: Der Pre-Echo-Effekt [PAINTER97]

Damit dieser Effekt nicht zu stark ausfällt verwendet die MDCT unterschiedliche Transformationsblocklängen. Bei sich wenig ändernden Signalen werden lange Blöcke verwendet, bei schnellen Änderungen werden kleinere Blöcke verwendet. Die MDCT erkennt diese Signal durch das psychoakustische Modell.
Bei einem langen Block werden die 36 Zeitsamples auf 18 Frequenzlinien abgebildet. Bei einem kurzen Block werden immer drei kurze Blöcke hintereinander gebildet. Jeder von ihnen enthält 6 Frequenzlinien. Sie entstehen aus jeweils 12 Subband Samples.
Am Ende dieser Prozedur werden noch vorhandene redundante Daten durch eine sogenannte Butterfly-Berechnung reduziert. Diesen Vorgang nennt man auch Aliasing-Reduzierung.
In Abbildung 21 sieht man die Umwandelung in den Frequenzbereich etwas übersichtlicher.

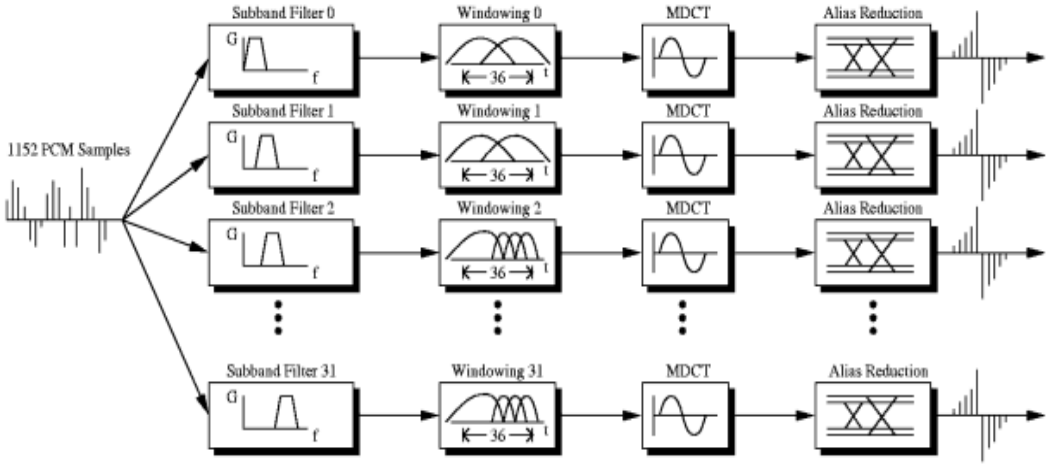


Abbildung 21: Übersicht der Umwandlung in den Frequenzbereich [KAPPES08]

Zeitgleich mit der Aufteilung in Subbänder werden die Samples durch eine Fast-Fourier-Transformation (FFT) für die Verwendung des psychoakustischen Modell umgewandelt. Dabei verwendet man eine 1024-Punkt Fast-Fourier-Transformation und vier 256-Punkt Fast-Fourier-Transformation um im Zeitbereich und im Frequenzbereich eine gute Auflösung zu erzielen. Anschließend können diese Daten für das schon besprochene psychoakustische Modell verwendet werden.

Nachdem nun die Daten etwas reduziert wurden, können diese Daten codiert werden. Dazu werden die 576 Frequenzlinien aus der MDCT , die Maskierungskurve aus dem psychoakustischem Modell und die eingestellte Bitrate. Die Frequenzlinien werden

quantisiert, wobei mithilfe der Maskierungskurve die Güte der Quantisierung überprüft wird. Die Bitrate wird für jedes Frame einzeln gespeichert. Das wird für die variable Bitrate einer MP3 Datei ausgenutzt.

Mit der Quantisierung wird weiter Daten reduziert. Zuerst wird für jedes Frame die globale Verstärkung bestimmt. Dadurch wird die Größe der Quantisierung bestimmt. Bei einer großen Verstärkung wird die Quantisierung mit einem großen Wert vollzogen, bei einer geringen Verstärkung wird ein kleiner Quantisierungswert verwendet.
Der Quantisierer im MPEG Layer 3 ist ein sogenannter Powerlaw-Quantisierer. Er besitzt die Eigenschaft kleine Frequenzanteile genauer zu quantisieren. Dadurch soll vermieden werden, dass diese zu schnell gegen Null laufen. Bei größeren Frequenzen wird mit größerer Quantisierung gearbeitet. Man nimmt dadurch Rundungsfehler in Kauf. Bei dieser Art spricht man auch von einer nichtlinearen Quantisierung. Bei der nichtlinearen Quantisierung entstehen im hohen Frequenzbereich neue Frequenzen, die der Mensch als Rauschen wahrnimmt. Man nennt sie Quantisierungsgeräusche.

Nach der Quantisierung werden die 576 Frequenzlinien in 24 Blöcke eingeteilt. Jedem dieser Blöcke wird ein so genannter Skalierungsfaktor zugeordnet. Damit sollen entstandene Quantisierungsgeräusche ausgeglichen werden.

Zum Schluss werden die Daten noch nach dem Huffman Verfahren kodiert. Dieses Verfahren benutzt kurze Codewörter für kleine Werte im Frequenzspektrum, da diese häufiger Vorkommen. Man hat in MPEG Layer 3 Standard 32 Huffman Tabellen vorgegeben, aus denen der Kodierer das Beste auswählen kann. Anschließend wird das Frame in kleinere Abschnitte eingeteilt, auf die jeweils die 32 Huffman Tabellen verwendet werden. Damit ermöglicht man eine feinere Auflösung der Frequenz.[KAPPES08]

Das Dekodieren erfolgt dadurch, das man die Schritte in umgekehrter Reihenfolge durchführt.

6.1.2 Vorbis

Der Vorbis Audio Codec wurde von der Xiph.Org Foundation entwickelt und zur kostenlosen Verwendung freigegeben. Dieses Audioformat wird in der Regel im Containerformat OGG gespeichert, so dass meistens von Vorbis OGG gesprochen wird. Das Vorbis Modell ist wie eine Pipeline aufgebaut. Der Datenstrom wird in fünf Pipelinestufen bearbeitet. In der Abbildung 22 sieht man den Aufbau des Vorbis Codec.

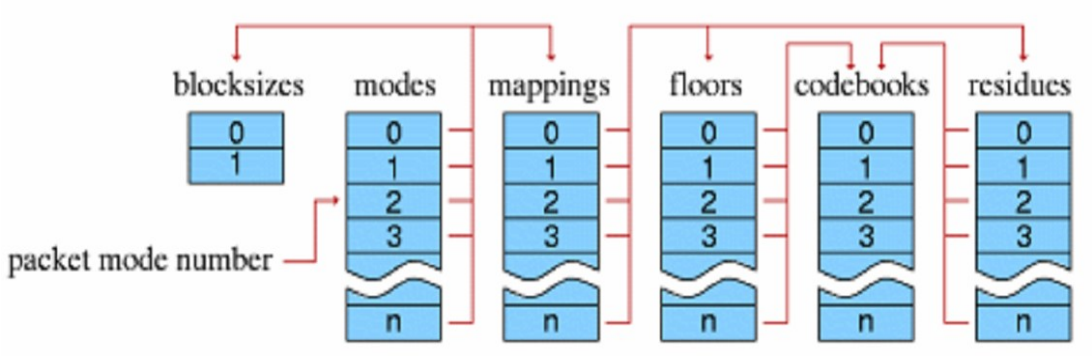


Abbildung 22: Aufbau des Vorbis Codec [VORBIS07]

Der Eingehende PCM Audiostrom wird in der Pipelinestufe Modes in Frames eingeteilt. Die Frames werden auf eine oder mehrere Vorlagen angewendet, die am besten zu diesem Frame passt. Diese Vorlagen enthalten wie sich ein Frame zu einem anderen Frame verändert. Die Nummer die dieses Frame erhält wird für die spätere Kodierung benötigt. Sie enthält die Frame Größe, den Fenstertyp (in der aktuellen Version von Vorbis I ist er 0), den Transformationstyp (in der aktuellen Version von Vorbis I ist er 0, was der MDCT entspricht) und eine Mapping Nummer. Diese wird im nächsten Schritt der Pipeline benötigt.

Nachdem der Datenstrom in Frames eingeteilt wurde werden sie in der nächsten Pipelinestufe, Mapping genannt, weiter bearbeitet. Die Mapping Nummer dient dabei für die Konfiguration dieser Pipelinestufe. Hier wird für jeden Kanal eine Submap errechnet. Vorbis analysiert das Frequenzspektrum und begrenzt das Spektrum in dem es dem Frame aus einer Liste ein bestimmte Submap zuweist.

Anschließend werden die Daten der nächsten Pipelinestufe übergeben. In diesem Floor werden die Daten von jedem Frame quantisiert. Das geschieht mit dem Floor Filter, der dieser Stufe ihren Namen gab.
Dieser Floor Filter kennt zwei Typen. Der erste Typ (Floor 0) benutzt eine Line Spectral Pair (LSP) Repräsentation des Signals. Der zweite Typ (Floor 1) benutzt eine Lineare Interpolation des Signals. Floor 0 liefert bessere Hörergebnisse an als Floor 1, deswegen verwendet Vorbis nur die LSP Repräsentation.

Etwa Zeitgleich mit der Floor Kodierung erfolgt die Bearbeitung der Überreste des Audiosignals in der Residue Stufe. Audiodaten die nicht von der dritten Pipelinestufe bearbeitet wird, weil sie außerhalb der Submap liegen, werden hier verarbeitet.
Mit diesen beiden Stufen soll das Signal möglichst naturgetreu wiedergegeben werden.

Zum Schluss werden die Daten aus der Floor Stufe und der Residue Stufe mit Huffman Codiert.[VORBIS07]

Bei der Dekodierung der Audiodaten wird wie bei MP3 alle Schritte in umgekehrter Reihenfolge abgearbeitet.

6.1.3 Speex

Speex ist wie der Vorbis Audio Codec von der Xiph.Org Foundation entwickelt und ist spezialisiert auf menschliche Sprache. Speex unterstützt drei Abtastraten: 8 kHz, 16 kHz und 32 kHz. Alle anderen werden Abtastraten sind zwar Möglich aber nur durch einen erhöhten Rechenaufwand.
Er benutzt immer eine Variable Bitrate, das heißt der Codec reagiert auf die momentane Geräuschkulisse und senkt sie wenn möglich auf ein Minimum. Speex ist auch mit einer Voice Activity Detection (VAD) ausgestattet, damit erkennt der Codec ob überhaupt ein Signal vorhanden ist das man kodieren muss. Dadurch werden überflüssige Datenübertragungen vermieden.
Darüber hinaus kann Speex auch noch die Discontinuous Transmission (DTX). Dabei werden Frames nicht gesendet wenn nichts zu senden ist, also Sprechpausen. Bei der Erkennung hilft die Voice Activity Detection.

Eine weitere Besonderheit gegenüber MP3 und Vorbis ist die Rauschunterdrückung. Dabei werden Frequenzen, die nicht der Menschlichen Stimme entsprechen, ausgeblendet und gehen nicht in die Quantisierung mit ein.

Die Frame Größe bei Speex ist fest auf 20 ms eingestellt. Das entspricht 160 Samples bei 8 kHz. Jedes dieser Frames wird dabei noch einmal in vier Subframes mit jeweils 40 Samples unterteilt. Die Codierung von Speex läuft in zwei Schritten ab, eine Analyse des Gesamten Frames und eine Analyse der vier Subframes.

Das gesamte Frame wird zuerst mit Line Spectral Pairs (LSP) analysiert. Hierbei wird das Signal herunter gerechnet, so das man es linear interpolieren kann. Damit erhält man ein grobes Muster des ursprünglichen Signal. Dieses neue Signal wird mit einer Vektor Quantisierung mit 30 Bit für Hohe Qualität und 18 Bit für Niedrige Qualität abgetastet.

Anschließend werden die vier Subframes einer Analysis-by-synthesis (Abs) unterzogen.

.....

Speex benutzt für die Subframes ebenfalls eine Vektor Quantisierung, aber mit einem festen Codebuch. Dazu wird jedes Subframe in Subvektoren mit 5 bis 20 Samples unterteilt.

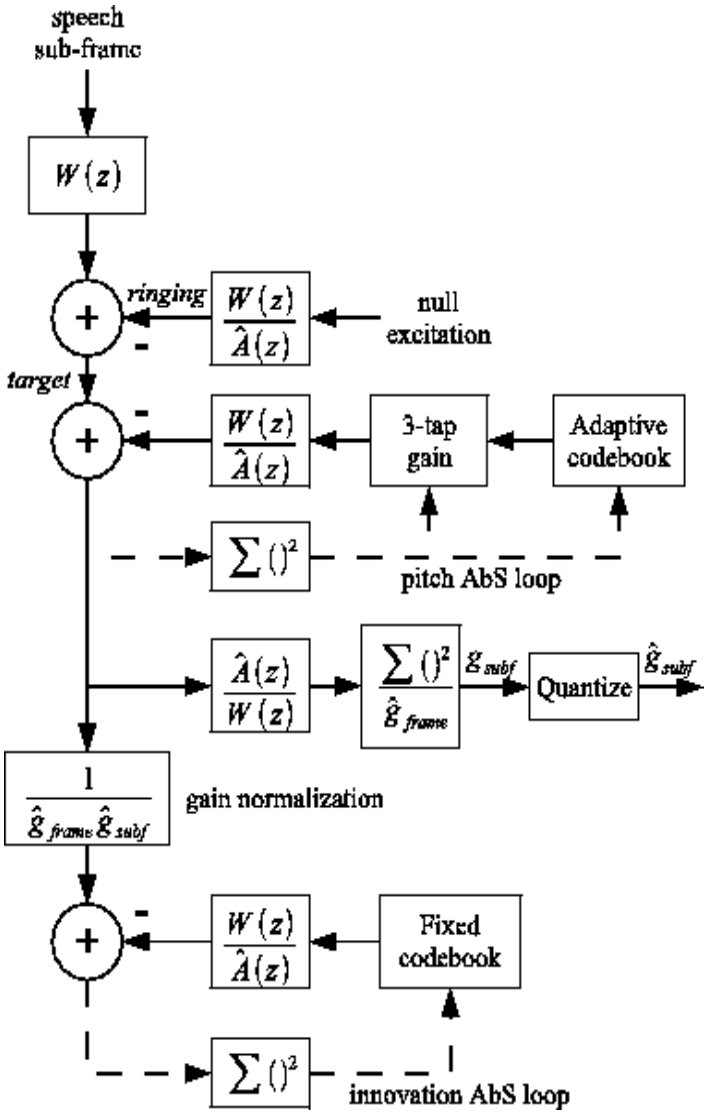


Abbildung 23: Analysis-by-synthesis bei einem Subframe [SPEEX06]

[SPEEX06]

6.1.4 GSM 6.10

Der GSM Codec ist wie Speex ein verlustbehafteter Audio Codec zur Sprachübertragung.

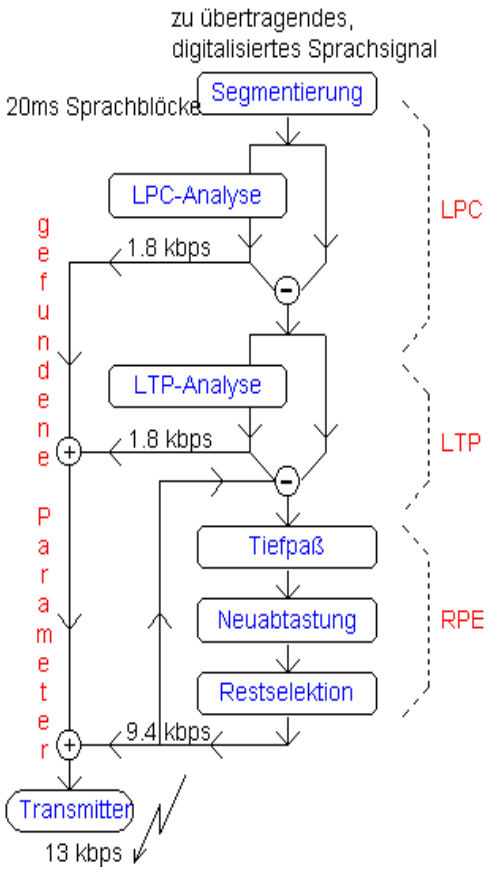
GSM ist die Abkürzung für Group System Mobile. Dieser Code ist einer der ältesten Audio Codecs für die Sprachübertragung und wurde im Jahre 1992 veröffentlicht. Dank seiner guten Eigenschaften wurde die Technik in die Handys integriert und verwendet ihn bis heute.
Seit 1992 hat sich dieser Audio Codec weiterentwickelt, gerade in Hinblick auf die Handy Technologie, und wurde mit zusätzlichem Funktionen wie SMS Übertragung ausgestattet [GSM08]. Im Rahmen dieser Arbeit betrachten wir nur denen Audio Codec in der Version 6.10. Dieser Codec ist in den neuen Windows Version wie Windows XP Windows Vista schon integriert und kann somit verarbeitet werden.

GSM besitzt auch die von Speex bekannten Voice Activity Detection (VAD) und Discontinuous Transmission (DTX).
Das Signal wird wie bei Speex in 20ms große Frames zerlegt. Bei 8kHz sind das 160 Samples.
Man verwendet dabei die Kodierung mithilfe des Linear Prediction Coefficients (LPC), der Long Term Prediction (LTP) und der Regular Pulse Excitation (RPE). Abgekürzt heißt sie RPE/LTP-LPC. In Abbildung 24 kann man sehen wie die Kodierung des RPE/LTP-LPC Algorithmus im GSM 6.10 aufgebaut ist.
Das Frame wird zuerst mithilfe der LPC ein Muster im Signal zu erkennen und damit eine grobe Struktur des ursprünglichen Signals zu erhalten. Die Struktur erhält man durch jeweils 8 Samples. Diese gewonnen Informationen werden übertragen und von dem Ursprungssignal abgezogen und der LZP Stufe übergeben.
Hier werden länger dauernde Abhängigkeiten wie Silben erkannt. Diese gewonnen Signale können auch übertragen werden. Das Signal wird von dem aktuellen Signal abgezogen und das neu entstandene Signal der RPE Stufe übergeben.
Das Restsignal wurden die wichtigsten Sprachteile aus den vorangegangenen Stufen entfernt, so dass das Amplitudenspektrum nur noch sehr flach und wenig Änderungen aufweist. In der RPE Stufe wird das Signal durch eine Tiefpaßfilterung geglättet und anschließend neu abgetastet.

Dabei wird das Signal insgesamt dreimal abgetastet. Das erste Mal fängt er beim ersten Wert an und liest nur noch jeden dritten Wert. Danach tastet er das Signal noch einmal ab, fängt

aber bei zweiten Wert an und tastet auch jeden dritten Wert ab

Im dritten Durchlauf beginnt die RPE Stufe beim dritten Wert und liest jeden dritten Wert. Diese drei Werte werden nun mit dem ursprünglichen Restsignal verglichen und das Signal, welches am besten dazu passt wird behalten [ROHRBR95].



6.2 Vergleichen der Audio Codecs

Im vorangegangenen Kapitel wurden einige Audio Codecs näher vorgestellt, damit aber eine Aussage getroffen werden kann, welcher Codec für unser Projekt geeignet ist müssen einige Tests gemacht werden.

Der Audio Codec, der im CINeSPACE Projekt verwendet werden soll, muss über ein gutes Verhältnis zwischen Audioqualität und der Datenrate haben.

6.2.1 Audioqualität

Zuerst wird die Qualität der verschiedenen Audio Codecs betrachtet. Jeder Mensch hat eine subjektive Wahrnehmung, die je nach Alter und Geschlecht unterschiedlich sein können. Aus diesem Grund kann nicht eine Person allein über die Qualität entscheiden.
Aus diesem Grund wurde einer ausgesuchten Gruppe, verschiedenen Alters, unterschiedliche Audiobeispiele vorgespielt. Die subjektiven Eindrücke werden anhand der MOS-Skala bewertet [BADACH04]. MOS (Mean Opinion Score) ist wie man ihn Tabelle 10 sehen kann eine Bewertung zwischen eins und fünf, wobei fünf für die beste Qualität steht. Diese Skala solle eine Aussage geben, die für die Sprachqualität wichtig sind:

- Verständlichkeit der Sprache
- Akzeptanz der Lautstärke
- Akzeptanz der Laufzeitschwankungen und Echos

MOS-Wert	Bedeutung
5 = exzellent	keine Anstrengung zum Verständnis der Sprache nötig, totale Entspannung möglich
4 = good	keine Anstrengung nötig, Aufmerksamkeit nötig
3 = fair	leichte, moderate Anstrengung nötig
2 = poor	merkbare, deutliche Anstrengung nötig
1 = bad	Trotz Anstrengung keine Verständigung

Tabelle10: MOS-Skala [BADACH04]

Der Versuchsaufbau ist für alle Testpersonen die gleiche. Ihnen wird nacheinander verschiedene Audiostücke vorgespielt, die anhand der MOS-Skala bewertet werden soll. Die Personen bekommen nicht gesagt um welchen Codec es sich handelt damit eine neutrale Bewertung möglich ist.
Als Bewertungsgrundlage dienen dabei fünf Beispiele. Um eine möglichst gute Ausgangs Grundlage zu haben wurden dabei Quellen von CD und DVD benutzt. Die ersten beiden Samples stammen dabei aus einem Hörspiel. Bei diesen sind leichte Hintergrundgeräusche zu hören. Damit sollen überprüft werden wie die einzelnen Codecs auf leichte Störgeräusche reagieren. Das dritte Beispiel ist aus einem Hörbuch und soll verdeutlichen wie die einzelnen

Codecs generell mit menschlicher Sprache umgehen können. Im vierten Beispiel wird die Situation eines Dialoges zwischen einer Frau und einem Mann in einem Bus dargestellt. Das fünfte Sample soll aufzeigen wie die Codecs sich bei sehr lauten Umgebungsgeräuschen verhalten.
Pro Sample wurden insgesamt 34 verschiedene Beispieldateien in Mono erzeugt. 12 für MP3, 9 für Vorbis, 9 für Speex und 4 für GSM. Diese insgesamt 170 Dateien wurden 20 verschiedenen Personen unterschiedlichen Alters und unterschiedlichen Geschlechtes vorgespielt und anhand der MOS-Skala bewertet. Die durchschnittlichen MOS-Werte kann der Tabelle 11 entnehmen.

Wie man erkennen kann ist der Speex Codec bei einer Abtastrate von 8 kHz von diesen vier ausgesuchten Codecs der Beste. Sowohl in der Qualitätsstufe 5 und 10 wurde er mit 4.38 am angenehmsten empfunden. Erst danach kommen Vorbis mit 3.44 ohne dass MP3 mit 3.42. GSM ist nur bei der einer niedrigsten Qualität mit 2.56 bewertet worden.
Ein ähnliches Bild kann man auch bei 11.025 kHz feststellen. So ist Speex mit Qualitätsstufe 10 mit 4.72 vor Vorbis und MP3. Vorbis hat dabei 4.19 und MP3 4.09 bekommen. Wie schon bei 8 kHz überzeugt GSM nur auf gegenüber den anderen Codecs mit der niedrigsten Qualität, muss sich dabei aber auch dem Vorbis Codec geschlagen geben.
Bei 22.050 kHz muss sich der Speex Codec mit 4.80 dem Vorbis Codec mit 4.82 geschlagen geben. Wieder auf dem dritten Platz ist MP3 mit 4.55 erreichten Punkten. GSM kann sich hier wieder nur gegenüber der niedrigsten Qualität behaupten.
Bei einer Abtastrate von 44.100 kHz ist MP3 mit 4.73 vor Vorbis mit 4.70. Mit 4.52 Punkten ist der GSM Codec schon auf dem dritten Platz. Speex fehlt hier bei dieser Abtastrate weil Speex sie nicht verarbeiten kann.

In Abbildung 25 werden diese MOS Werte aus Tabelle 11 übersichtlicher dargestellt. Die dafür verwendeten Daten befinden sich in Anhang A in einer übersichtlichen Tabelle.

	MP3			Vorbis		
		MOS	Varianz		MOS	Varianz
8.0 kHz	8 kps	1.68	0.22	8 kps	1.88	0.71
	32 kps	2.43	0.25	42 kps	3.44	0.7
	96 kps	3.12	0.77	---	---	---
	160 kps	3.42	0.55	---	---	---
11.025 kHz	8 kps	1.63	0.24	12 kps	3.59	0.49
	48 kps	3.17	0.55	50 kps	4.19	0.54
	96 kps	4.09	0.61	---	---	
22.050 kHz	8 kps	1.50	0.25	16 kps	3.66	0.50
	56 kps	4.25	0.45	58 kps	4.24	0.55
	96 kps	4.55	0.41	96 kps	4.82	0.39
44.100 kHz	48 kps	4.38	0.38	48 kps	4.33	0.48
	96 kps	4.73	0.20	96 kps	4.70	0.46
	Speex			GSM 6.10		
		MOS	Varianz		MOS	Varianz
8.0 kHz	Qual. 1	1.62	0.49	8 kps	2.56	0.69
	Qual. 5	3.78	0.76	---	---	---
	Qual. 10	4.38	0.49	---	---	---
	---	---	---	---	---	---
11.025 kHz	Qual. 1	1.70	0.48	8 kps	3.42	0.55
	Qual. 5	3.87	0.61	---	---	---
	Qual. 10	4.72	0.45	---	---	---
22.050 kHz	Qual. 1	1.28	0.45	8 kps	3.96	0.78
	Qual. 5	4.31	0.53	---	---	---
	Qual. 10	4.80	0.40	---	---	---
44.100 kHz	---	---	---	8 kps	4.52	0.5
	---	---	---	---	---	---

Tabelle11: MOS-Skala für Audio Codecs

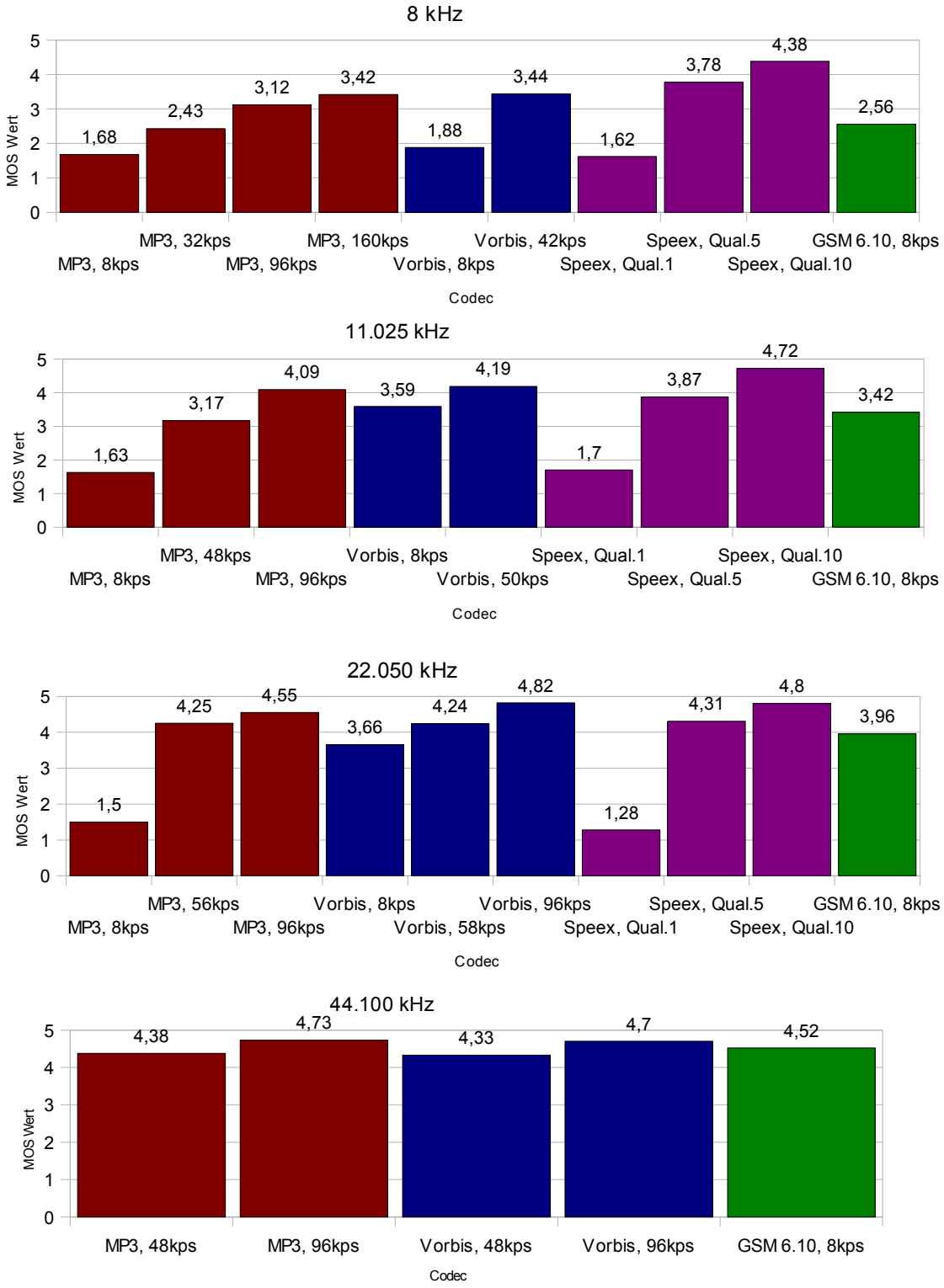


Abbildung 25: Übersicht MOS Werte

Also Speex hat sich bei dem subjektiven MOS Test als der beste Codec herausgestellt. Für eine Voice over IP Anwendung ist aber nicht nur der Höreindruck entscheidet sondern es muss auch noch die durchschnittliche Datenrate beachtet werden.

6.2.2 Datenrate

Um eine Aussage treffen zu können welcher Codec das beste Verhältnis hat von Qualität zur Datenrate benötigen wir noch die einzelnen Datenraten. Aus Tabelle 12 kann man die Durchschnittliche Datenrate entnehmen. In Anhang B befinden sich alle Informationen, die zu der Durchschnittlichen Datenrate führten.

	MP3		Vorbis		Speex		GSM	
		KB/s		KB/s		KB/s		KB/s
8.0 kHz	8 kps	1.03	8 kps	1.61	Qual. 1	0.58	8 kps	1.64
	32 kps	4.11	42 kps	4.91	Qual. 5	1.49	---	---
	96 kps	12.30	---	---	Qual. 10	3.22	---	---
	160 kps	20.55	---	---	---	---	8 kps	2.26
11.025 kHz	8 kps	1.02	12 kps	2.31	Qual. 1	0.79	---	---
	48 kps	6.13	50 kps	6.49	Qual. 5	2.05	---	---
	96 kps	12.25	---	---	Qual. 10	4.43	---	---
22.050 kHz	8 kps	1.02	16 kps	3.34	Qual. 1	1.14	8 kps	4.52
	56 kps	7.11	58 kps	9.48	Qual. 5	3.03	---	---
	96 kps	12.18	96 kps	14.33	Qual. 10	6.61	---	---
44.100 kHz	48 kps	6.08	48 kps	5.30	---	---	8 kps	9.05
	96 kps	12.15	96 kps	11.34	---	---	---	---

Tabelle12: Datenraten der Audio Codecs

Die Angabe über die durchschnittliche Datenrate für einen Codec kann man noch keine Aussage darüber treffen wie gut er ist. Bei 8 kHz bei Qualitätsstufe 1 benötigt der Speex Codec gerade einmal 0.58 KB die Sekunde. Gegenüber allen anderen Codec ist das ein sehr guter Wert. Wenn wir aber die Qualität nachdem MOS Wert miteinbeziehen, der bei 1.62 liegt, nützt diese geringe Datenrate nichts, da man sich fast nicht verständigen kann.

6.2.3 Einbeziehung von Datenrate und MOS Wert

Um die Ergebnisse zu vergleichen teilen wir die KB/s aus diesem Kapitel mit den MOS Werte aus dem vorangegangenen Kapitel. So erhält man die KB/s für einen MOS Wert. Je niedriger dieser Wert ausfällt desto besser ist dann dieses Ergebnis.

	MP3		Vorbis		Speex		GSM	
		KB/s / MOS		KB/s / MOS		KB/s / MOS		KB/s / MOS
8.0 kHz	8 kps	0,61	8 kps	0,86	Qual. 1	0,36	8 kps	0,64
	32 kps	1,69	42 kps	1,43	Qual. 5	0,39	---	---
	96 kps	3,94	---	---	Qual. 10	0,74	---	---
	160 kps	6,01	---	---	---	---	8 kps	0,66
11.025 kHz	8 kps	0,63	12 kps	0,64	Qual. 1	0,11	---	---
	48 kps	1,93	50 kps	1,55	Qual. 5	0,53	---	---
	96 kps	3,00	---	---	Qual. 10	0,94	---	---
22.050 kHz	8 kps	0,68	16 kps	0,91	Qual. 1	0,89	8 kps	1,14
	56 kps	1,67	58 kps	2,24	Qual. 5	0,70	---	---
	96 kps	2,68	96 kps	2,97	Qual. 10	1,38	---	---
44.100 kHz	48 kps	1,39	48 kps	1,22	---	---	8 kps	2,00
	96 kps	2,57	96 kps	2,41	---	---	---	---
Durchschnitt		2,23		1,58		0,67		1,11

Tabelle13: Datenrate / MOS Wert der Audio Codecs

Wie man der Tabelle 13 entnehmen kann ist der Speex Codec die beste Wahl. Kein anderer Codec schafft es mit einer niedrigen Datenrate so ein gutes Hörergebnis zu liefern wie mit dem Speex Codec.

Auch wenn wir von jedem Codec das beste Ergebnis mit der Datenrate vergleichen, erhalten wir ein ähnliches Bild.
Wie in Tabelle 14 zu sehen ist hat Speex mit seinem Besten MOS Wert die niedrigste Datenrate. Danach folgt auch schon GSM 6.10, wobei er von diesen vier Codecs das schlechteste Hörempfinden aufweist. MP3 folgt mit der fast doppelt so großen Datenrate wie der Speex Codec. Den besten Höreindruck konnte zwar der Vorbis Codec erreichen, aber das auch nur auf Kosten der Datenrate.

Codec	MOS Wert	KB/s
MP3 44.100 kHz, 96 kps	4,73	12,10
Vorbis 22.050 kHz, 96 kps	4,82	14,33
Speex 22.050 kHz, Qual.10	4,80	6,61
GSM 6.10 44.100 kHz	4,52	9,05

Tabelle14: Gegenüberstellung der besten Codecs

Aus den Gründen sollte man den Speex Codec im CINeSPACE Projekt verwenden. Denn es sprechen die guten Hörergebnisse und die geringen Datenraten klar für den Speex Codec.

7. Ergebnisse und Ausblick

Im Laufe dieser Arbeit wurden mehrere Audio Bibliotheken wie die MCI , Directshow, Simple Directmedia Layer, Port Audio, RtAudio und OpenAL näher betrachtet. Davon kamen schließlich nur noch zwei Bibliotheken, RtAudio und OpenAL, für das CINeSPACE Projekt in Frage. Diese beiden Bibliotheken erfüllen die Anforderungen für die Plattformunabhängigkeit und stehen zur freien Verfügung, so dass sie im CINeSPACE Projekt problemlos eingesetzt werden können.

Bevor auf dieser beiden Audio Bibliotheken näher eingegangen wurde, wurde zuerst nach Alternativen im Voice over IP Bereich und im Streaming-Server Bereich gesucht. In keinem dieser beiden Bereiche konnte eine überzeugende Alternative gefunden werden.

Aus diesem Grund wurden die beiden Audio Bibliotheken RtAudio und OpenAL näher betrachtet. Vom Programmieraufwand ist RtAudio deutlich geringer als der von OpenAL, wobei dieser auch nicht allzu groß ist.

Bei der Wahl für einen passenden Audio Codec wurden Tests an 20 Personen durchgeführt. Anhand der MOS- Skala musste mehrere Audiodateien bewertet werden. Den Probanden wurden fünf verschiedene Beispieldateien in 34 Variationen vorgespielt, die jeweils mit einem unterschiedlichen Audio Codecs kodiert waren. Bei den verwendeten Codecs handelte es sich um MP3, Vorbis, Speex und GSM. Bei diesen Hörtest stellte sich Speex als ein sehr guter Audio Codec heraus. Auch im Hinblick auf die anfallende Datenrate hat sich Speex deutlich vor GSM postiert.

Im weiteren Verlauf des Projektes sollte der Speex Codec in beide Audio Bibliotheken implementiert werden und mit Leistungstest direkt auf dem CINeSPACE HMD getestet werden. Ohne diese Leistungstests kann man leider keine Aussage darüber machen, welche dieser beiden Audio Bibliotheken für das Projekt am Besten ist.

Anhang

Anhang A : Übersicht der MOS Werte

Anhang B : Übersicht der Datenraten

Anhang I : Literaturverzeichnis

Anhang II : Abbildungsverzeichnis

Anhang III : Tabellenverzeichniss

Anhang IV : Glossar

Anhang A: Übersicht der MOS Werte

Dieser Anhang Zeit alle Werte der Codec Bewertung, die im Kapitel 6.2.1 mehr erläutert wird. Die entsprechenden Sample Dateien, die bewertet wurden, befinden sich auf der CD unter AudioCodecTest\Sample*. In der Liste werden diese Samples mit S1 bis S5 abgekürzt.

MP3		Person 1					Person 2					Person 3					Person 4					Person 5				
		S1	S2	S3	S4	S5	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5
8.0 kHz	8 kps	2	2	2	2	1	2	2	2	2	1	2	2	2	2	1	2	2	2	2	1	1	2	1	2	1
	32 kps	2	3	3	3	3	2	2	2	2	3	2	2	2	2	3	2	3	3	3	3	2	2	2	3	3
	96 kps	3	3	5	3	3	2	2	4	3	3	2	2	4	3	3	3	3	5	3	3	3	2	5	3	3
	160 kps	4	3	5	3	4	3	2	4	3	3	3	2	4	3	3	4	3	5	3	4	4	3	4	3	4
11.025 kHz	8 kps	2	1	1	2	2	2	1	1	2	2	2	1	1	2	2	2	1	1	2	2	2	1	1	2	2
	48 kps	3	3	4	4	4	2	2	3	3	3	2	2	3	3	3	3	3	4	4	4	2	3	4	4	4
	96 kps	4	4	5	5	5	3	3	4	4	4	3	3	3	4	4	3	4	5	5	5	4	4	5	5	5
22.050 kHz	8 kps	2	1	2	1	1	2	1	2	1	1	2	1	2	1	1	2	1	2	1	1	2	1	2	1	1
	56 kps	4	4	5	5	5	3	3	4	4	4	3	3	4	5	4	5	5	5	5	5	4	4	5	5	4
	96 kps	5	5	5	5	5	4	5	4	5	5	3	4	4	5	5	5	5	5	5	5	5	5	4	5	4
44.100 kHz	48 kps	5	4	5	5	5	4	3	5	4	4	4	3	5	5	5	5	4	5	5	5	4	4	4	4	5
	96 kps	5	4	5	5	5	5	4	5	5	5	5	4	5	5	5	5	4	5	5	5	5	4	5	4	5
Vorbis																										
8.0 kHz	8 kps	2	2	3	2	1	2	2	3	2	1	2	2	3	2	1	2	1	3	2	1	2	1	3	1	1
	42 kps	3	3	4	5	3	3	3	3	4	3	3	3	3	4	3	3	3	4	5	3	3	3	4	5	3
11.025 kHz	12 kps	3	3	4	4	4	3	4	3	4	4	3	4	3	4	4	3	3	4	4	4	3	3	4	4	4
	50 kps	4	4	5	5	4	4	4	4	4	4	4	4	4	4	4	4	4	5	5	4	3	4	5	5	4
22.050 kHz	16 kps	4	3	4	4	3	4	4	4	4	3	4	4	4	4	3	4	3	4	4	3	4	3	4	4	3
	58 kps	5	5	5	4	4	4	4	4	4	4	4	4	4	4	4	5	5	5	4	4	5	4	5	3	4
	90 kps	5	5	5	5	5	4	4	5	5	5	5	5	5	5	5	5	5	5	5	5	4	5	5	4	5
44.100 kHz	48 kps	4	4	5	5	5	4	4	4	4	4	4	4	4	4	4	5	5	5	5	5	4	4	4	4	5
	96 kps	5	5	5	5	5	4	4	5	5	5	4	4	5	5	5	5	5	5	5	5	5	4	5	4	5
Speex																										
8.0 kHz	Qualität1	1	2	2	2	1	1	1	1	2	1	1	2	1	2	1	2	2	2	2	2	1	2	2	2	1
	Qualität5	3	3	5	5	4	3	3	4	4	4	3	3	4	4	4	3	3	5	5	4	3	3	5	5	4
	Qualität10	4	5	5	5	5	4	4	4	4	5	4	4	4	4	5	4	5	5	5	5	4	4	4	4	5
11.025 kHz	Qualität1	1	2	2	2	2	1	1	1	2	2	1	2	1	2	2	1	2	2	2	2	1	2	2	2	2
	Qualität5	3	4	4	5	4	3	4	4	5	4	3	4	4	5	4	3	4	4	5	4	3	4	3	4	4
	Qualität10	5	5	5	5	5	4	5	5	4	5	4	5	5	4	5	5	4	5	5	5	5	4	5	5	5
22.050 kHz	Qualität1	1	1	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	1	1	1	2	2	1
	Qualität5	4	4	5	5	5	3	4	4	4	4	4	4	4	4	4	5	4	4	5	5	4	4	5	5	5
	Qualität10	5	5	5	5	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	4	5	5	5	5	4
GSM 6.10																										
8.0 kHz		3	3	3	4	2	2	3	2	2	2	2	3	2	2	2	3	3	3	4	2	3	2	3	4	2
11.025 kHz		4	3	4	4	3	3	3	4	4	3	3	3	4	3	3	4	3	4	4	3	4	3	4	4	3
22.050 kHz		4	4	5	5	5	3	4	4	4	4	3	3	4	4	4	4	4	5	5	5	4	3	4	4	5
44.100 kHz		5	4	4	5	5	4	4	4	5	5	4	4	4	5	5	5	4	4	5	5	5	4	4	5	4

MP3		Person 6					Person 7					Person 8					Person 9					Person 10				
		S1	S2	S3	S4	S5	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5
8.0 kHz	8 kps	2	2	1	2	1	1	1	2	2	1	1	2	2	1	1	2	2	2	2	1	2	2	2	2	1
	32 kps	2	2	2	3	2	2	2	2	2	3	2	3	3	3	3	2	3	3	3	3	2	3	3	3	3
	96 kps	3	2	5	3	2	3	2	4	4	3	3	3	5	3	3	3	3	5	3	3	3	3	5	3	3
	160 kps	4	3	4	3	4	3	2	4	3	3	4	3	5	3	4	4	3	5	3	4	4	3	5	3	4
11.025 kHz	8 kps	2	1	1	2	2	2	1	1	2	2	2	2	2	2	2	2	1	1	2	2	2	1	1	2	2
	48 kps	2	3	4	4	4	2	2	3	3	3	4	3	4	4	4	3	3	4	4	4	3	3	4	4	4
	96 kps	4	4	5	5	5	3	3	3	4	4	4	4	5	4	5	3	4	5	5	5	3	4	5	5	5
22.050 kHz	8 kps	2	2	2	2	2	2	1	2	1	1	2	2	2	1	1	2	1	2	1	1	2	1	2	1	1
	56 kps	4	4	5	5	4	4	4	4	5	4	4	4	5	5	5	5	5	5	5	5	5	5	5	5	5
	96 kps	5	5	4	5	4	3	4	4	5	5	5	5	5	4	5	5	5	5	5	5	5	5	5	5	5
44.100 kHz	48 kps	4	4	4	4	5	4	3	5	5	5	4	4	5	5	5	5	4	5	5	5	5	4	5	5	5
	96 kps	5	4	5	4	5	5	4	4	5	5	5	4	5	5	5	5	4	5	5	5	5	4	5	5	5
Vorbis																										
8.0 kHz	8 kps	2	1	3	1	1	2	2	3	2	1	2	2	3	2	1	2	1	3	2	1	2	1	3	2	1
	42 kps	3	3	4	5	3	3	3	3	4	3	3	3	4	5	3	3	3	4	5	3	3	3	4	5	3
11.025 kHz	12 kps	4	3	3	4	4	3	4	3	4	4	3	3	4	4	4	3	3	4	4	4	3	3	4	4	4
	50 kps	3	4	5	5	4	4	4	4	4	4	4	4	5	5	4	4	5	5	5	4	4	4	5	5	4
22.050 kHz	16 kps	4	3	4	4	3	4	4	4	4	3	3	3	4	3	3	4	3	4	4	3	4	3	4	4	3
	58 kps	5	4	5	3	4	4	4	4	4	4	4	4	5	4	4	5	5	5	4	4	5	5	5	4	4
	90 kps	4	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	4	4	5	5	5	5	5	5	5
44.100 kHz	48 kps	4	4	4	4	5	4	4	4	4	4	4	4	5	5	5	4	4	5	4	5	5	5	5	5	5
	96 kps	4	4	5	4	5	4	4	5	5	5	5	4	5	5	5	5	4	5	5	5	5	5	5	5	5
Speex																										
8.0 kHz	Qualität1	1	2	2	2	1	1	2	1	2	1	1	2	2	2	1	2	2	2	2	2	2	2	2	2	2
	Qualität5	3	3	5	5	4	3	3	4	4	4	3	3	5	5	4	3	3	5	5	4	3	3	5	5	4
	Qualität10	4	4	4	4	4	4	4	4	4	5	4	5	5	5	5	4	5	5	5	5	4	5	5	5	5
11.025 kHz	Qualität1	1	2	2	2	2	1	2	1	2	2	1	2	2	2	2	1	2	2	3	2	1	2	2	2	2
	Qualität5	3	4	3	4	4	3	4	4	4	4	3	4	4	5	4	4	4	4	5	4	3	3	4	5	4
	Qualität10	4	4	5	5	5	4	5	5	4	5	5	5	5	5	5	5	4	5	5	5	5	4	5	5	5
22.050 kHz	Qualität1	1	1	2	2	1	1	1	1	1	1	2	1	2	2	2	1	1	2	2	1	1	1	2	2	1
	Qualität5	4	4	5	5	5	4	4	4	4	4	4	4	5	5	5	5	4	4	5	5	5	4	4	4	5
	Qualität10	5	4	5	5	4	4	5	5	4	5	5	5	5	5	4	5	5	5	5	4	5	5	5	5	4
GSM 6.10																										
8.0 kHz		3	2	3	4	2	1	3	2	2	2	3	3	3	4	2	3	2	3	3	2	3	3	3	4	2
11.025 kHz		4	3	4	3	3	3	3	4	3	3	4	3	4	4	3	4	3	4	4	3	4	3	4	4	3
22.050 kHz		4	3	3	4	5	3	3	4	3	4	4	5	5	5	5	5	4	5	5	5	4	2	5	5	5
44.100 kHz		5	5	4	5	5	4	4	4	5	5	5	4	4	5	5	5	4	5	5	5	5	4	4	5	5

MP3		Person 11					Person 12					Person 13					Person 14					Person 15				
		S1	S2	S3	S4	S5	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5
8.0 kHz	8 kps	1	2	1	2	1	2	2	1	2	1	2	2	2	2	1	1	2	2	2	2	2	1	2	2	1
	32 kps	2	2	2	3	3	2	2	2	3	2	2	2	2	2	3	2	2	2	2	3	2	3	3	3	3
	96 kps	3	2	5	3	3	3	2	5	3	2	2	2	4	3	3	3	3	3	4	3	3	3	5	3	3
	160 kps	4	3	4	3	4	4	3	4	3	4	3	2	4	3	3	3	2	3	3	3	4	3	5	3	4
11.025 kHz	8 kps	2	1	1	2	2	2	1	1	2	2	2	1	1	2	2	2	1	1	2	2	2	2	1	2	2
	48 kps	2	3	4	4	4	2	3	4	4	4	2	2	3	3	3	3	2	3	3	3	3	3	4	4	4
	96 kps	4	3	5	5	5	4	4	5	5	5	3	3	4	4	4	3	3	3	4	4	4	4	5	5	5
22.050 kHz	8 kps	2	1	2	1	1	2	2	2	2	2	2	1	2	1	1	2	1	2	1	1	2	1	2	1	1
	56 kps	4	4	5	4	4	4	4	5	4	4	3	3	4	4	4	4	4	4	4	4	4	4	4	5	3
	96 kps	5	5	3	4	4	5	4	4	5	4	4	5	4	5	5	3	3	4	3	4	5	5	5	5	5
44.100 kHz	48 kps	4	4	4	4	5	4	4	4	4	5	4	3	5	4	4	4	4	4	4	4	5	4	4	4	5
	96 kps	5	4	5	4	5	5	4	5	4	5	5	4	5	5	5	5	4	5	4	5	5	4	5	5	5
Vorbis																										
8.0 kHz	8 kps	2	1	3	1	1	2	1	3	1	1	2	2	3	2	1	2	2	3	2	2	2	2	3	2	2
	42 kps	3	3	4	5	3	3	3	4	5	3	3	3	3	4	3	3	3	3	4	3	3	3	4	5	3
11.025 kHz	12 kps	3	3	4	3	4	4	3	3	4	4	3	4	3	4	4	3	4	3	4	4	3	3	4	4	4
	50 kps	3	4	5	5	4	3	4	5	5	4	3	4	4	4	4	3	4	4	4	4	5	4	5	5	4
22.050 kHz	16 kps	4	3	4	4	3	4	3	4	4	3	4	3	4	5	3	4	4	4	4	3	4	3	4	4	3
	58 kps	4	4	4	3	3	5	4	5	3	4	4	4	4	4	4	4	4	5	4	4	5	5	5	5	5
	90 kps	4	4	5	4	5	4	4	5	5	5	5	4	5	5	5	5	5	5	5	5	5	5	5	5	5
44.100 kHz	48 kps	4	4	4	4	5	4	4	4	4	5	4	4	4	4	4	4	4	4	4	4	5	5	5	5	5
	96 kps	5	5	5	5	5	4	4	5	4	5	4	4	4	5	5	4	4	5	5	5	5	5	5	5	5
Speex																										
8.0 kHz	Qualität1	1	2	2	2	1	1	2	2	2	2	1	1	1	2	1	1	2	1	2	2	2	2	2	2	2
	Qualität5	3	3	5	5	4	3	3	3	4	4	3	3	4	4	4	3	3	4	4	4	3	3	5	5	4
	Qualität10	4	4	4	4	5	4	4	4	4	4	4	4	4	4	5	4	4	4	4	5	4	5	5	5	5
11.025 kHz	Qualität1	1	2	2	2	2	1	2	2	2	2	1	1	1	2	2	1	2	1	2	2	1	2	2	2	2
	Qualität5	3	4	3	4	4	3	4	3	4	4	3	4	4	5	4	3	4	4	4	4	3	4	4	5	4
	Qualität10	5	4	5	5	5	4	4	5	5	5	4	5	5	4	5	4	5	5	4	5	5	5	5	5	5
22.050 kHz	Qualität1	1	1	2	2	1	1	1	2	2	1	1	1	1	1	1	1	2	1	2	1	1	1	2	2	1
	Qualität5	4	4	5	5	5	4	4	5	5	5	3	4	4	4	4	4	4	4	4	4	4	4	5	5	5
	Qualität10	5	5	5	5	4	5	4	5	5	5	5	4	5	4	5	4	5	4	4	5	5	5	5	5	4
GSM 6.10																										
8.0 kHz		3	3	3	4	3	2	2	3	3	2	2	3	2	2	2	1	2	2	2	2	3	3	3	4	2
11.025 kHz		4	3	4	4	3	3	3	4	3	3	3	3	4	4	3	3	3	4	3	3	4	3	4	4	3
22.050 kHz		3	3	4	3	5	4	3	3	4	5	3	4	4	4	4	3	3	4	3	4	3	4	5	4	4
44.100 kHz		5	4	4	5	4	4	5	4	4	5	4	5	4	5	5	5	4	4	5	5	5	4	4	5	5

MP3		Person 16					Person 17					Person 18					Person 19					Person 20				
		S1	S2	S3	S4	S5	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5
8.0 kHz	8 kps	2	2	2	2	1	2	2	2	2	1	2	2	2	2	1	2	2	1	2	1	2	2	2	2	1
	32 kps	2	3	3	3	3	2	2	2	2	3	2	2	2	2	3	2	2	2	3	2	2	2	2	2	3
	96 kps	3	3	5	3	3	2	2	4	3	3	2	2	4	3	3	3	2	5	3	2	2	2	4	3	3
	160 kps	4	3	5	3	4	3	2	4	3	3	3	2	4	3	3	4	3	4	3	4	3	2	4	3	3
11.025 kHz	8 kps	2	1	1	2	2	2	1	1	2	2	2	1	1	2	2	2	1	1	2	2	2	1	1	2	2
	48 kps	3	3	4	4	4	2	2	3	3	3	2	2	3	3	3	2	3	4	4	4	2	2	3	3	3
	96 kps	3	4	5	5	5	3	3	4	4	4	3	3	3	4	4	4	4	5	5	5	3	3	3	4	4
22.050 kHz	8 kps	2	1	2	1	1	2	1	2	1	1	2	1	2	1	1	2	2	2	2	2	2	1	2	1	1
	56 kps	5	5	5	5	5	3	3	4	4	4	3	3	4	5	4	4	4	5	4	4	3	3	4	5	4
	96 kps	5	5	5	5	5	4	5	4	5	5	3	4	4	5	5	5	4	4	5	4	3	4	4	5	5
44.100 kHz	48 kps	5	4	5	5	5	4	3	5	4	4	4	3	5	5	5	4	4	4	4	5	4	3	5	5	5
	96 kps	5	4	5	5	5	5	4	5	5	5	5	4	5	5	5	5	4	5	4	5	5	4	5	5	5
Vorbis																										
8.0 kHz	8 kps	2	1	3	2	1	2	2	3	2	1	2	2	3	2	1	2	1	3	1	1	2	2	3	2	1
	42 kps	3	3	4	5	3	3	3	3	4	3	3	3	3	4	3	3	3	4	5	3	3	3	3	4	3
11.025 kHz	12 kps	3	3	4	4	4	3	4	3	4	4	3	4	3	4	4	4	3	3	4	4	3	4	3	4	4
	50 kps	4	4	5	5	4	4	4	4	4	4	4	4	4	4	4	3	4	5	5	4	4	4	4	4	4
22.050 kHz	16 kps	4	3	4	4	3	4	4	4	4	3	4	4	4	4	3	4	3	4	4	3	4	4	4	4	3
	58 kps	5	5	5	4	4	4	4	4	4	4	4	4	4	4	4	5	4	5	3	4	4	4	4	4	4
	90 kps	5	5	5	5	5	4	4	5	5	5	5	5	5	5	5	4	4	5	5	5	5	5	5	5	5
44.100 kHz	48 kps	5	5	5	5	5	4	4	4	4	4	4	4	4	4	4	4	4	4	4	5	4	4	4	4	4
	96 kps	5	5	5	5	5	4	4	5	5	5	4	4	5	5	5	4	4	5	4	5	4	4	5	5	5
Speex																										
8.0 kHz	Qualität1	2	2	2	2	2	1	1	1	2	1	1	2	1	2	1	1	2	2	2	2	1	2	1	2	1
	Qualität5	3	3	5	5	4	3	3	4	4	4	3	3	4	4	4	3	3	3	4	4	3	3	4	4	4
	Qualität10	4	5	5	5	5	4	4	4	4	5	4	4	4	4	5	4	4	4	4	4	4	4	4	4	5
11.025 kHz	Qualität1	1	2	2	2	2	1	1	1	2	2	1	2	1	2	2	1	2	2	2	2	1	2	1	2	2
	Qualität5	3	3	4	5	4	3	4	4	5	4	3	4	4	5	4	3	4	3	4	4	3	4	4	5	4
	Qualität10	5	4	5	5	5	4	5	5	4	5	4	5	5	4	5	4	4	5	5	5	4	5	5	4	5
22.050 kHz	Qualität1	1	1	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	1	1	1	1	1	1
	Qualität5	5	4	4	4	5	3	4	4	4	4	4	4	4	4	4	4	4	5	5	5	4	4	4	4	4
	Qualität10	5	5	5	5	4	5	5	5	5	5	5	5	5	5	5	5	4	5	5	5	5	5	5	5	5
GSM 6.10																										
8.0 kHz		3	3	3	4	2	2	3	2	2	2	2	3	2	2	2	2	2	3	3	2	2	3	2	2	2
11.025 kHz		4	3	4	4	3	3	3	4	4	3	3	3	4	3	3	3	3	4	3	3	3	3	4	3	3
22.050 kHz		4	2	5	5	5	3	4	4	4	4	3	3	4	4	4	4	3	3	4	5	3	3	4	4	4
44.100 kHz		5	4	4	5	5	4	4	4	5	5	4	4	4	5	5	4	5	4	4	5	4	4	4	5	5

Anhang B: Übersicht der Datenraten

In diesem Anhang befindet sich die Datenraten der untersuchten Codecs. Die entsprechenden Sample Dateien befinden sich auf der CD unter AudioCodecTest\Sample*.

Vorbis		Sample 1		Sample 2	
		Dateigröße KB	KB/s	Dateigröße KB	KB/s
8.0 kHz	8 kps	40,5	1,50	15,2	1,69
	42 kps	127,0	4,70	46,7	5,19
11.025 kHz	12 kps	59,7	2,21	22,4	2,49
	50 kps	168,0	6,22	62,0	6,89
22.050 kHz	16 kps	85,0	3,15	31,9	3,54
	58 kps	224,0	8,30	88,4	9,82
	90 kps	368,0	13,63	132,0	14,67
44.100 kHz	48 kps	168,0	6,22	62,6	6,96
	96 kps	282,0	10,44	105,0	11,67

		Sample 3		Sample 4	
		Dateigröße KB	KB/s	Dateigröße KB	KB/s
8.0 kHz	8 kps	26,6	1,77	16,5	1,65
	42 kps	75,1	5,01	50,0	5,00
11.025 kHz	12 kps	34,7	2,31	24,2	2,42
	50 kps	99,4	6,63	65,7	6,57
22.050 kHz	16 kps	50,3	3,35	34,5	3,45
	58 kps	158,0	10,53	97,9	9,79
	90 kps	233,0	15,53	144,0	14,40
44.100 kHz	48 kps	11,0	0,73	66,1	6,61
	96 kps	203,0	13,53	111,0	11,10

		Sample 5			Durchschnitt
		Dateigröße KB	KB/s		KB/s
8.0 kHz	8 kps	43,1	1,44		1,61
	42 kps	139,0	4,63		4,91
11.025 kHz	12 kps	64,0	2,13		2,31
	50 kps	185,0	6,17		6,49
22.050 kHz	16 kps	96,0	3,20		3,34
	58 kps	269,0	8,97		9,48
	90 kps	402,0	13,40		14,33
44.100 kHz	48 kps	180,0	6,00		5,30
	96 kps	298,0	9,93		11,34

MP3		Sample 1		Sample 2	
		Dateigröße KB	KB/s	Dateigröße KB	KB/s
8.0 kHz	8 kps	26,9	1,00	9,7	1,08
	32 kps	107,0	3,96	38,8	4,31
	96 kps	323,0	11,96	116,0	12,89
	160 kps	538,0	19,93	194,0	21,56
11.025 kHz	8 kps	26,9	1,00	9,6	1,07
	48 kps	161,0	5,96	57,6	6,40
	96 kps	322,0	11,93	115,0	12,78
22.050 kHz	8 kps	26,8	0,99	9,5	1,06
	56 kps	187,0	6,93	66,6	7,40
	96 kps	321,0	11,89	114,0	12,67
44.100 kHz	48 kps	160,0	5,93	56,9	6,32
	96 kps	321,0	11,89	113,0	12,56

		Sample 3		Sample 4	
		Dateigröße KB	KB/s	Dateigröße KB	KB/s
8.0 kHz	8 kps	15,6	1,04	10,4	1,04
	32 kps	62,4	4,16	41,6	4,16
	96 kps	187,0	12,47	124,0	12,40
	160 kps	312,0	20,80	208,0	20,80
11.025 kHz	8 kps	15,5	1,03	10,4	1,04
	48 kps	93,1	6,21	62,1	6,21
	96 kps	186,0	12,40	124,0	12,40
22.050 kHz	8 kps	15,4	1,03	10,3	1,03
	56 kps	108,0	7,20	72,0	7,20
	96 kps	185,0	12,33	123,0	12,30
44.100 kHz	48 kps	92,4	6,16	61,5	6,15
	96 kps	184,0	12,27	123,0	12,30

		Sample 5			Durchschnitt
		Dateigröße KB	KB/s		KB/s
8.0 kHz	8 kps	29,5	0,98		1,03
	32 kps	118,0	3,93		4,11
	96 kps	354,0	11,80		12,30
	160 kps	590,0	19,67		20,55
11.025 kHz	8 kps	29,5	0,98		1,02
	48 kps	176,0	5,87		6,13
	96 kps	353,0	11,77		12,25
22.050 kHz	8 kps	29,4	0,98		1,02
	56 kps	205,0	6,83		7,11
	96 kps	352,0	11,73		12,18
44.100 kHz	48 kps	176,0	5,87		6,08
	96 kps	352,0	11,73		12,15

Speex		Sample 1		Sample 2	
		Dateigröße KB	KB/s	Dateigröße KB	KB/s
8.0 kHz	Qualität1	15,0	0,56	5,4	0,60
	Qualität5	39,2	1,45	14,0	1,56
	Qualität10	84,9	3,14	30,2	3,36
11.025 kHz	Qualität1	20,6	0,76	7,4	0,82
	Qualität5	53,9	2,00	19,2	2,13
	Qualität10	116,0	4,30	41,4	4,60
22.050 kHz	Qualität1	29,8	1,10	10,7	1,19
	Qualität5	79,9	2,96	28,4	3,16
	Qualität10	198,0	7,33	70,3	7,81

		Sample 3		Sample 4	
		Dateigröße KB	KB/s	Dateigröße KB	KB/s
8.0 kHz	Qualität1	8,7	0,58	5,9	0,59
	Qualität5	22,6	1,51	15,1	1,51
	Qualität10	48,8	3,25	32,6	3,26
11.025 kHz	Qualität1	11,9	0,79	8,0	0,80
	Qualität5	31,1	2,07	20,7	2,07
	Qualität10	67,3	4,49	44,8	4,48
22.050 kHz	Qualität1	17,2	1,15	11,5	1,15
	Qualität5	46,0	3,07	30,6	3,06
	Qualität10	46,0	3,07	75,9	7,59

		Sample 5			Durchschnitt
		Dateigröße KB	KB/s		KB/s
8.0 kHz	Qualität1	16,5	0,55		0,58
	Qualität5	43,0	1,43		1,49
	Qualität10	93,1	3,10		3,22
11.025 kHz	Qualität1	22,6	0,75		0,79
	Qualität5	59,1	1,97		2,05
	Qualität10	128,0	4,27		4,43
22.050 kHz	Qualität1	32,7	1,09		1,14
	Qualität5	87,6	2,92		3,03
	Qualität10	217,0	7,23		6,61

GSM 6.10	Sample 1		Sample 2	
	Dateigröße KB	KB/s	Dateigröße KB	KB/s
8.0 kHz	43,4	1,61	15,2	1,69
11.025 kHz	59,8	2,21	21,1	2,34
22.050 kHz	119,0	4,41	42,3	4,70
44.100 kHz	239,0	8,85	84,5	9,39

	Sample 3		Sample 4	
	Dateigröße KB	KB/s	Dateigröße KB	KB/s
8.0 kHz	24,9	1,66	16,6	1,66
11.025 kHz	34,4	2,29	22,8	2,28
22.050 kHz	68,7	4,58	45,7	4,57
44.100 kHz	137,0	9,13	91,3	9,13

	Sample 5			Durchschnitt
	Dateigröße KB	KB/s		KB/s
8.0 kHz	47,7	1,59		1,64
11.025 kHz	65,6	2,19		2,26
22.050 kHz	131,0	4,37		4,52
44.100 kHz	262,0	8,73		9,05

Anhang I: Literaturverzeichnis

[ATLERS07] Ahlers, Ernst : *Perfekt vernetzt* In: c´t 12/07, S.120 – 123

[BADACH04] Badach, Anatol:*Voice over IP : Die Technik*. Hanser Verlag, München, 2004

[BBURG00] Brandenburg, Karl-Heinz [Popp, Harald] : *An Introduction to MPEG Layer 3*. http://www.mp3-tech.org/programmer/docs/trev_283-popp.pdf, 2000, zuletzt besucht am 09.05.2008

[CINE08] - : *CINeSPACE : technology*. www.cinespace.eu, 2007, zuletzt besucht am 18.02.2008

[DARWINS08] Apple : *Darwin Streaming Server*. http://developer.apple.com/opensource/server/streaming/index.html, 2008, zuletzt besucht am 26.03.2008

[DS08a] Microsoft Corporation : *DirectShow*. http://msdn2.microsoft.com/en-us/library/ms783323(VS.85).aspx, 2008, zuletzt besucht am 18.04.2008

[DS08b] Microsoft Corporation : *DirectShow : DirectShow System Overview*. http://msdn2.microsoft.com/en-us/library/ms783354(VS.85).aspx, 2008, zuletzt besucht am 18.04.2008

[DS08c] Microsoft Corporation : *DirectShow : Add a Filter by CLSID*. http://msdn2.microsoft.com/en-us/library/ms778952(VS.85).aspx, 2008, zuletzt besucht am 18.04.2008

[DS08d] Microsoft Corporation : *Directshow : Connect Two Filters*. http://msdn2.microsoft.com/en-us/library/ms782274(vs.85).aspx, 2008, zuletzt besucht am 18.04.2008

[EISY07] Eisy : *Deutschland hat beim Breitband Nachholbedarf*. http://www.yoome.de/news.877.html, 2007, zuletzt besucht am 20.03.2008

[FSFAPSL07] Free Software Foundation :
FSF's Opinion of the Apple Public Source License (APSL) 2.0.
http://www.gnu.org/philosophy/apsl.html, 2007,
zuletzt besucht am 29.03.2008

[GNULGPL] GNU : *GNU LESSER GENERAL PUBLIC LICENSE*.
http://www.gnu.org/licenses/lgpl-3.0.html, 2007,
zuletzt besucht am 27.04.2008

[GSM08] GSM Association : *GSM World*.
http://www.gsmworld.com/index.shtml, 2008,
zuletzt besucht am 04.05.2008

[HARTMANN99] Hartmann, Nico : *MP3 Grundlagen: Psychoakustik*.
http://www.tecchannel.de/test_technik/grundlagen/401060/, 1999,
zuletzt besucht am 04.05.2008

[HMDS08] Virtual Realities, Inc. : *Head Moundted Displays*.
http://www.vrealities.com, 2008, zuletzt besucht am 04.03.2008

[ICECAST08] Xiph.Org Foundation : *Icecast*. http://www.icecast.org/, 2008,
zuletzt besucht am 26.03.2008

[KAPPES08] Kappes, Andre : *Die Audiokodierung mp3*.
http://www.mp3encoding.de, 2008, zuletzt besucht am 06.05.2008

[MASSOTH07] Massoth, Michael : *Vorlesungsskript Telekommunikation* , 2007,
Hochschule Darmstadt, Sommersemester 2007

[MILGRAM94] Milgram, Paul [Takemura, Haruo] : *Augmented Reality:*
A class of displays on the reality-virtuality continuum.
http://vered.rose.utoronto.ca/index.php, 1994,
zuletzt besucht am 10.03.2008

[MUMBLE08] Das Mumble-Team : *Mumble*. http://mumble.sourceforge.net/, 2008,
zuletzt besucht am 21.03.2008

[NO23LIVE08] Bischof, Ivan : *no23Live*. http://www.no23.de, 2008,
zuletzt besucht am 26.03.2008

[OPENAL08] - : *OpenAL*. http://www.openal.org/, 2008,
zuletzt besucht am 19.04.2008

[PAINTER97] Painter, Ted [Spanias, Andreas] : *A Review of Algorithms for Perceptual Coding of Digital Audio Signals*. http://www.svendtofte.com/university/extern/dsp_codec_review.ps, 1997, zuletzt besucht am 10.05.2008

[RECHENBG02] Rechenberg, Dr. Peter [Pomberger, Dr. Gustav]:*Informatikhandbuch*. Hanser, München, 2002

[ROHRBR95] Rohrbacher, Kai : *Die GSM-Technik Ecke : Sprachcodierung*. , 1995, zuletzt besucht am 13.05.2008

[RTAUDIO08] Scavone, Gary P. : *RtAudio*. http://www.music.mcgill.ca/~gary/rtaudio/, 2008, zuletzt besucht am 19.04.2008

[SCHMITT04] Schmitt, Maik [Jochem, Rainer] : *Voice over IP unter Verwendung von Open Source-Software*. http://graphics.cs.uni-sb.de/VoIP/fopra.pdf, 2004, zuletzt besucht am 20.03.2008

[SHOUTCAST08] Nullsoft, Inc. : *SHOUTcast*. http://www.shoutcast.com/, 2008, zuletzt besucht am 26.03.2008

[SKYPE08] Skype Limited : *Skype*. http://www.skype.de, 2008, zuletzt besucht am 20.03.2008

[SPEEX06] Xiph.Org Foundation : *Speex*. http://www.speex.org/, 2006, zuletzt besucht am 01.05.2008

[TEAMSPEAK08] TeamSpeak Systems GmbH : *Teamspeak*. http://www.goteamspeak.com/, 2008, zuletzt besucht am 21.03.2008

[VENTRILO08] Flagship Industries, Inc. : *Ventrilo*. http://www.ventrilo.com/, 2008, zuletzt besucht am 21.03.2008

[VLCMEDIA08] The VideoLAN Team : *VLC media Player*. http://www.videolan.org/, 2008, zuletzt besucht am 26.03.2008

[VOCAL07] VOCAL Technologies : *Audio Vocoders*. http://www.vocal.com/index.html, 2007, zuletzt besucht am 04.04.2008

[VORBIS07] Xiph.Org Foundation : *Vorbis*. http://www.xiph.org/vorbis/, 2007, zuletzt besucht am 01.05.2008

Anhang II: Abbildungsverzeichnis

Anhang III: Tabellenverzeichnis

Anhang IV: Glossar

3GP	ein Video Codec spezialisiert für Handy und PDA
AAC+	Abkürzung für MPEG-4 High Efficiency Advanced Audio Coding und ist ein Audio Codec
ALSA	Soundschnittstelle unter Linux
APSL	Abkürzung für Apple Public Source License, welche von Apple als Open Source Lizenz veröffentlicht wurde
Augmented Reality	englische Bezeichnung für Erweiterte Realität
Augmented Virtuality	englische Bezeichnung für erweiterte Virtualität
Bitrate	bezeichnet das Verhältnis von Datenmenge zu einer Zeiteinheit
Codec	Kunstwort aus **Co**dierer und **Dec**odierer
FSF	Abkürzung für Free Software Foundation und ist eine gemeinnützige Organisation zur Förderung freier Software
GNU GPL	eine von der Free Software Foundation herausgegebenen Lizenz für freie Software
GNU LGPL	eine von der Free Software Foundation herausgegebenen Lizenz für freie Software
GPL	Abkürzung für General Public License (Siehe auch GNU GPL)
H.264/MPEG-4 AVC	ein verlustbehafteter Video Codec

Head Mounted Display	ist ein am Kopf befestigtes Anzeigegerät, mit einem oder zwei Displays vor dem Auge
HMD	Abkürzung für Head Mounted Display
IP	Abkürzung für Internet Protocol
LPC	Abkürzung für Linear Prediction Coefficients, einem Algorithmus zu Erkennung von Mustern in Signalen
LSP	Abkürzung für Line Spectral Pairs, ein Algorithmus zu Erkennung von Mustern im Signal, besser als LPC
Mixed Reality	Bedeutet eine Vermischung zwischen Realität und einer künstlichen Welt
MP3	ein verlustbehafteter Audio Codec, ist aber auch unter der Bezeichnung MPEG-1 Audio Layer 3 bekannt
OGG	ein freies Containerformat der Xiph.Org Foundation, in denen Audiodaten, Videodaten oder Texte abgelegt sein können
OSS	Soundschnittstelle unter Linux
Overhead	der Mehraufwand der für eine Datenübertragung nötig ist
PCM	Abkürzung für Puls-Code-Modulation und wird unter anderem für die Digitalisierung von Audiodaten
Proprietär	Eigentum, Software gehört jemanden

QTSS	Abkürzung für Quicktime Streaming Server
Stream	ist eine kontinuierliche Übertragung von Daten
TCP	Abkürzung für Transmission Control Protocol und ist ein Transportprotokoll
Text-to-Speech	die Möglichkeit Textinhalte mithilfe des Computers in menschliche Sprache umzuwandeln
UDP	Abkürzung für User Datagram Protocol und ist ein Transportprotokoll
VoIP	Abkürzung für Voice over IP
Voice over IP	Abkürzung für Telefonie über das Internet oder LAN mithilfe des Internet Protokolls
Vorbis	ein verlustbehafteter Audio Codec der Xiph.Org Foundation, der meist im Containerformat OGG verwendet wird
Vorbis OGG	Bezeichnung für den Audio Codec Vorbis der im Containerformat OGG gespeichert ist
WMA	Abkürzung für Windows Media Audio und ist ein verlustbehafteter Audio und Video Codec

Danksagung

Ich danke hiermit allen die mich während meiner Praxisphase und bei meiner Bachelorarbeit unterstützt haben. Allen voran meiner Mutter Erna und meinem Vater Gebhard, ebenso meinen beiden Brüdern Thorsten und Christian.
Dank gebührt auch Prof. Dr. Elke Hergenröther die mir dabei geholfen hat in das Fraunhofer IDG zu kommen und bei diesen interessanten Projekt mitzuwirken.
Auch allen Mitarbeitern beim IDG Abteilung A2 möchte ich mich bedanken. Besonders Pedro Santos und Dominik Acri die mir bei Problemen geholfen haben und wichtige Hinweise auf die Funktionsweise von verschiedenen Programmen und allgemein zu Programmen gegeben haben.
Des weiteren möchte ich den 20 Testpersonen danken, die sich die Zeit genommen haben die Codecs zu bewerten.
Den größten Dank haben die ganzen Korrekturleser, die (hoffentlich) alle Fehler gefunden haben, die in dieser Arbeit entstanden sind.