(http://www.htw-dresden.de/)

Fakultät Informatik/Mathematik (https://www.htw-dresden.de/hochschule/fakultaeten/info-math)

Audio-, Video- und Grafikprogrammierung

Die Lehrveranstaltung "*Audio-, Video- und Grafikprogrammierung*" findet im 5. Semester des Bachelorstudiengangs Medieninformatik bzw. im 7.Semester des Diplomstudiengangs Medieninformatik im Umfang von 2/0/2 SWS statt. Es werden solide Programmierkenntnisse der Programmiersprache C++ vorausgesetzt.

- Inhalt der LV
- Praktikumsaufgaben
- Zeitplan
- Skripte (externer Zugriff nur via VPN) (https://www2.htw-dresden.de/~bruns/Skripte/)
- Legale Beispielmedien für Ihre Programme finden Sie auf dem Laufwerk P: im Ordner bruns oder auf dieser Seite (https://samplelib.com/).

Die Lehrveranstaltung wird mit Hilfe von zwei Teilleistungen bewertet. Im Laufe des Semesters ist entsprechend der Aufgabenstellung Software zu entwickeln, dabei können zu den Praktika (siehe Zeitplan) entsprechende Punkte erworben werden (Vorführen der Software - jeweils 5 Punkte pro Aufgabe (PVL), Beantwortung von Fragen - jeweils weitere 5 Punkte pro Aufgabe/Themengebiet (APL)). Wenn die Hälfte der geforderten Punkte (siehe Aufgabenstellung unten) erbracht wurde, gilt die PVL als bestanden.

Aus der erreichten Gesamtpunktzahl ergibt sich die Abschlussnote des Moduls.

Themen der Lehrveranstaltung:

- einführender Theorieteil
- Das Media Control Interface und die Windows Presentation Foundation
- Grundlagen zur DirectX-API
    - COM-Programmierung ist wirklich nicht schwer
    - DirectSound - Bytefolgen zeitsynchron anhören :-)
    - Direct3D - direkter Zugriff auf die Grafikhardware
    - DirectInput - Eingabegeräte direkt auslesen
    - DirectShow - Multimedia-Datenströme abspielen, erzeugen und konvertieren
- Dateiformate
    - dynamische Dateiformate (RIFF - am Beispiel von wav-Dateien)
      lesen und schreiben von RIFF-Blöcken, wav-Dateien in DirectSound einbinden
    - Pixelgrafiken
        - statische Dateiformate am Beispiel des bmp-Dateiformates
        - eine jpg-Komprimierung einbinden (lesen und schreiben)
        - Grafikfilter selbst programmmieren
- die Fast Fourier Transformation (FFT)
    - die FFT verstehen
    - und zeitsynchron anwenden
- Managed Multimedia mit dem .NET-Framework und der Windows Presentation Foundation (WPF)

Praktikumsaufgaben

- Vorbemerkungen
- MCI- und WPF-Anwendungen
- DirectX (DirectSound, Direct3D, DirectShow)
- wav-Dateien
- Pixelgrafiken
- Fast Fourier Transformation
- Managed Multimedia

Vorbemerkungen

Die vorliegenden Praktikumsaufgaben sollen dem Studenten helfen, das in der Vorlesung vermittelte Wissen zu festigen. Die Aufgaben lehnen sich an die in der Vorlesung besprochenen Stoffgebiete an. Das Praktikum findet in einem Labor mit Windows-Rechnern statt, wobei Visual C++ (und das DirectX-SDK) als Entwicklungsplattform verwendet wird.

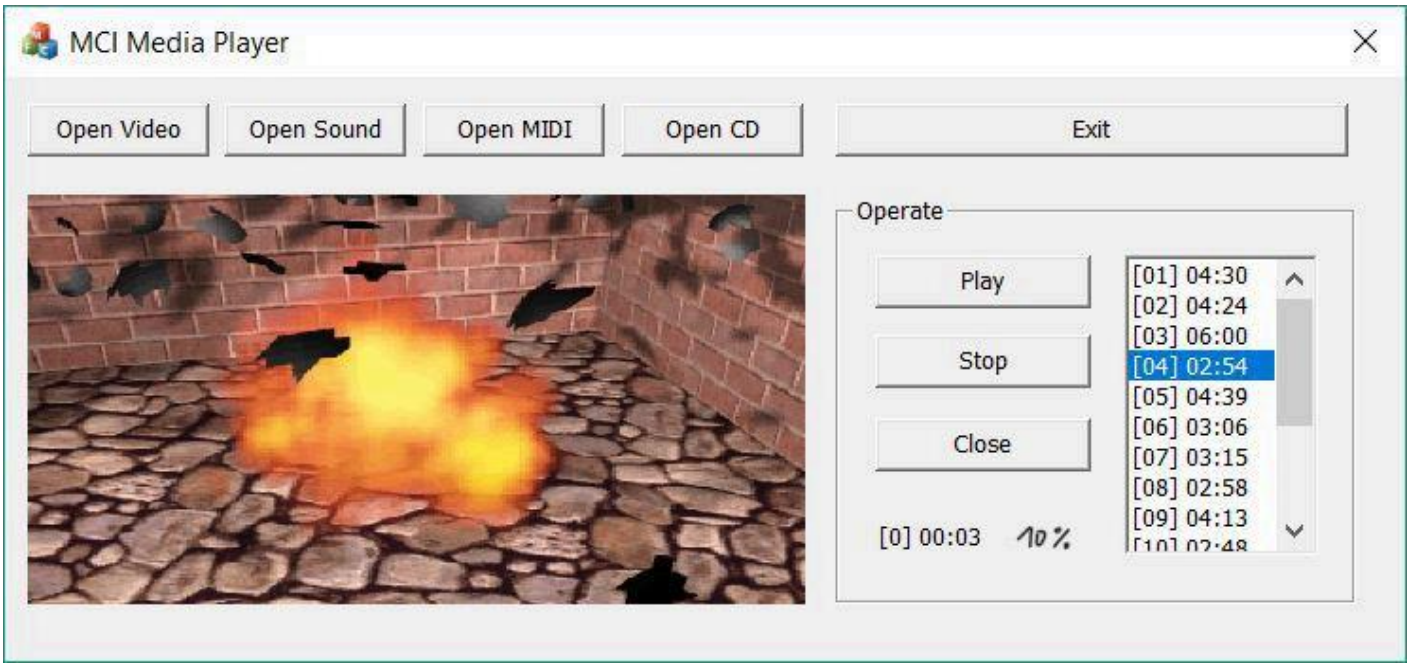
MCI- und WPF-Anwendungen

Aufgabe 1: Audio-/Midi-/Video-Player (mit der CMCIObject-Klasse)

Implementieren Sie ein Programm das mit Hilfe der Klasse CMCIObject die Ausgabe einer Audiodatei im wav- oder mp3-Format ermöglicht **[1P]**. Erweitern Sie das Programm um zwei weitere Schaltflächen, mit deren Hilfe der Nutzer eine Ihnen im Praktikum zur Verfügung gestellte MIDI- und eine Video-Datei (im aktuellen Fenster) abspielen kann **[1P]**.

Die Klasse CMCIObject ist um die Methoden SetAviPosition(...), TMSFSeek(...) und GetTMSFPosition(...) mit der in der Vorlesung besprochenen Funktionalität zu erweitern. Die Oberfläche Ihres Programms ist funktional entsprechend des hier dargestellten Screenshots zu gestalten **[1P]**.

Erweitern Sie Ihr Programm dahingehend, das Schaltflächen zum Starten, Stoppen und Schließen der Wiedergabe einer Audio-CD, sowie zum gezielten Wechsel auf einzelne Titel der Audio-CD bereitstellt werden. Alle Titel der eingelegten Audio-CD sind ein einer ListBox mit Angabe der Titellänge aufzuführen. Außerdem ist während des Abspielvorgangs die aktuelle Titel-Nummer und die laufende Abspielzeit in einem geeigneten Intervall auszugeben **[1P]**.



Erweitern Sie die Klasse CMCIObject um eine zusätzliche Methode zur Ermittlung der Länge einer Mediendatei (analog zu GetTrackLength). Ergänzen Sie nun die Darstellung der Zeitanzeige zum Abspielfortschritt des aktuellen Abspielvorgangs (z.B. einer Prozentanzeige) **[1P]**. Testen Sie Ihre Anwendung und machen Sie sich die Grenzen des Media Control Interface deutlich.

Termine: Bearbeitung und Abnahmen siehe Zeitplan. Die Punkte sollten nach Möglichkeit während der regulären Praktika erworben werden. Weitere 5 Punkte können durch die korrekte Beantwortung der Fragen bei der Abnahme erarbeitet werden.

Aufgabe 2: (Audio-) Video-Player (mit der Windows Presentation Foundation)

Implementieren Sie mit C# eine WPF Anwendung unter Nutzung der in der Vorlesung erarbeiteten Wrapperklasse. Hierzu ist die Grundfunktionalität (Play-, Pause-, Stopp) zum Abspielen einer Videodatei bereitzustellen.
Eine weitere Schaltfläche dient zum Umschalten in den Vollbildmodus, der dann zum Beispiel mit der linken Maustaste wieder verlassen werden kann.

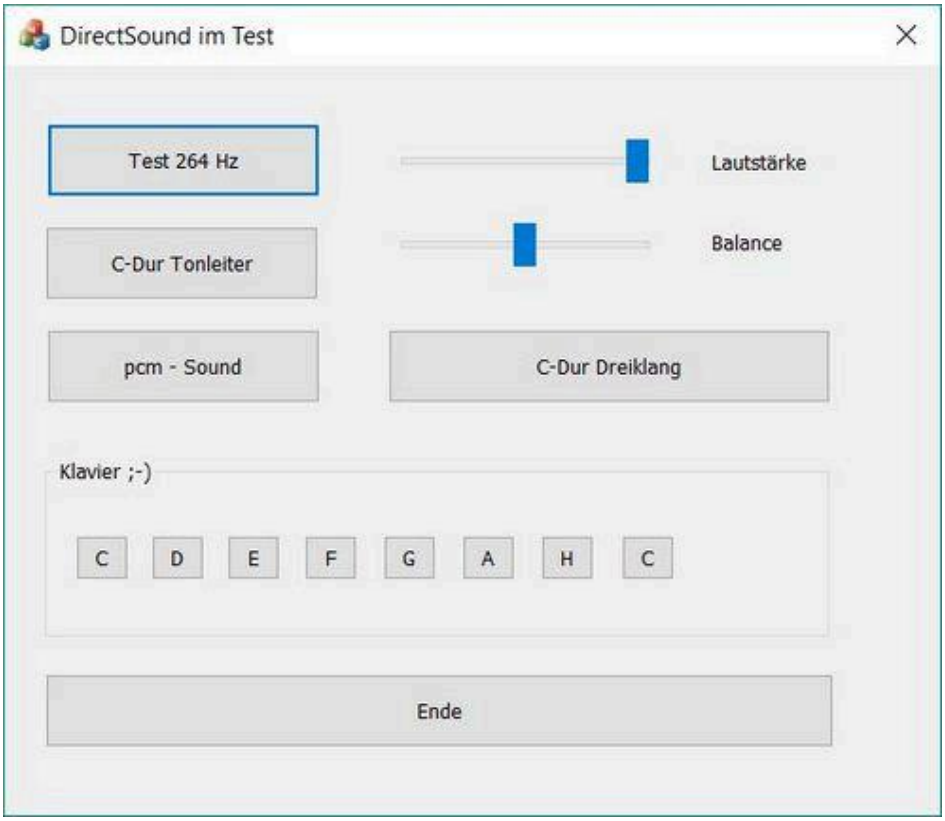
Die aktuelle Abspielposition ist als Zeitangabe (aktuelle Abspielzeit / Gesamtlänge) auszugeben.
Ein mitlaufender Slider dient zum Navigieren innerhalb des Videos. Zwei weitere Slider dienen der Einstellung von Lautstärke und Balance.

DirectX

Aufgabe 1: DirectSound

Implementieren Sie eine einfache Dialoganwendung und binden Sie die in der Vorlesung besprochene Klasse CDirectSound ein. Initialisieren Sie ein Objekt dieser Klasse und erzeugen Sie einen "Secondary DirectSoundBuffer" mit folgenden Eigenschaften:

- PCM-Format,
- stereo,
- 16 Bit pro Sample,
- 22.05 kHz Samplingrate,
- 4 Sekunden Pufferkapazität



Schreiben Sie 2 Sekunden Sound mit einer Sinusamplitute und einer Frequenz von 264 Hz ("Ton c") in den Soundpuffer und spielen Sie diesen Soundpuffer zyklisch ab **[1P]**.

Starten Sie mit Hilfe einer weiteren Schaltfläche einen Timer mit einer Periode von 700 Millisekunden. Die Timerereignisse sind derart zu behandeln, dass jeweils 2 Sekunden des Soundpuffers im Sinne der Doppelpuffertechnik verwendet werden. Während der eine Pufferteil abgespielt wird, werden Sinusamplituten der C-Durtonleiter mit steigender Frequenz in den jeweils anderen Pufferteil geschrieben. Nach einer Oktave endet die Wiedergabe **[1P]**.

Versuchen Sie mit Hilfe der Doppelpuffertechnik und der von Ihnen programmierten Klasse CDirectSound den Sound aus einer im Praktikum zur Verfügung gestellten pcm-Datei bis zum Ende abzuspielen. Beachten Sie, dass am Ende der Puffer wieder mit 0-Bytes zu füllen ist. Legen Sie sich für das Abspielen von PCM-Dateien eine geeignete weitere Methode der Klasse an **[1P]**.

Wenn auf einem Instrument, wie z.B. einer Gitarre, Geige oder einem Klavier, ein Ton erzeugt wird, dann besteht dieser nicht nur aus einer einzigen Frequenz, sondern es schwingen noch verschiedene Obertöne mit. Man kann jedoch im Allgemeinen diesem Ton eine Grundfrequenz zuordnen. So ist dem Kammerton 'A' die Frequenz 440Hz zugeordnet oder dem Ton 'C' die Frequenz 264 Hz. Weitere Töne können über die folgende Tabelle bestimmt werden.

Die Musiktheorie besagt, dass die Tonabstände innerhalb einer Tonleiter in die Intervalle Prime, Sekunde, Terz, Quarte, Quinte, Sexte, Septime und Oktave eingeteilt sind. Nachfolgend ist als Beispiel eine C-Dur Tonleiter (reine Stimmung) und die dafür notwendigen Frequenzverhältnisse aufgeführt. Ähnliches gilt natürlich für Tonleitern in Moll.

Ton	C-Dur	Frequenzverh.	Ton	C-Moll	Frequenzverh.
Prime	c	1	Prime	c	1
Sekunde	d	9/8	Sekunde	d	9/8
gr.Terz	e	5/4	kl.Terz	es	6/5
Quarte	f	4/3	Quarte	f	4/3
Quinte	g	3/2	Quinte	g	3/2
Sexte	a	5/3	kl. Sexte	as	8/5
Septime	h	15/8	kl. Septime	b	9/5
Oktave	c	2	Oktave	c	2

Um z.B. einen C-Dur Dreiklang zu bekommen, nimmt man die Prime `c', die große Terz `e' und die Quinte `g'. Für einen C-Moll Dreiklang wird entsprechend die Prime, die kleine Terz und die Quinte verwendet.

Beispiel : Berechnung der Frequenzen für einen C-Dur Dreiklang. Der Ton `c' hat die Frequenz 264 Hz. Der Ton `e' hat somit die Frequenz 264*(5/4)=330 Hz und der Ton `g' eine Frequenz von 264*(3/2)=396 Hz.

Fügen Sie nun Ihrem Programm Schaltflächen (Buttons) für die einzelnen Töne einer Tonleiter hinzu und behandeln Sie das BN_CLICKED-Ereignis dahingehend, dass der jeweilige Ton kurz angespielt wird (je Ton ein eigener Soundpuffer). **[1P]**.

Geben Sie nun dem Nutzer die Möglichkeit, über weitere Schaltflächen den C-Dur Dreiklang (als Akkord mit Hilfe von 3 Soundpuffern) abzuspielen.**[1P]**.

Optional: Sie können Zusatzpunkte erwerben. Implementieren Sie in Ihrem Programm die Möglichkeit, Lautstärke und Balanceeinstellungen zu beeinflussen **[1ZP]**. Wir haben Ihnen im Praktikumsordner noch Gitarrentöne im pcm(raw)-Format (44.1 kHz Samplingrate, 16 Bit pro Sample, mono) bereitgestellt. Der Nutzer soll über eine geeignete Schaltfläche entscheiden können, ob er die oben genannte Funktionalität mit generiertem Sound oder Gitarrensound abgespielt bekommt **[1ZP]**.

Termine: Bearbeitung und Abnahmen siehe Zeitplan. Die Punkte sollten nach Möglichkeit während der regulären Praktika erworben werden. Weitere 5 Punkte sind bei korrekter Beantwortung der Fragen während der Abnahme möglich.

Aufgabe 2: 3D-Sound mit DirectSound

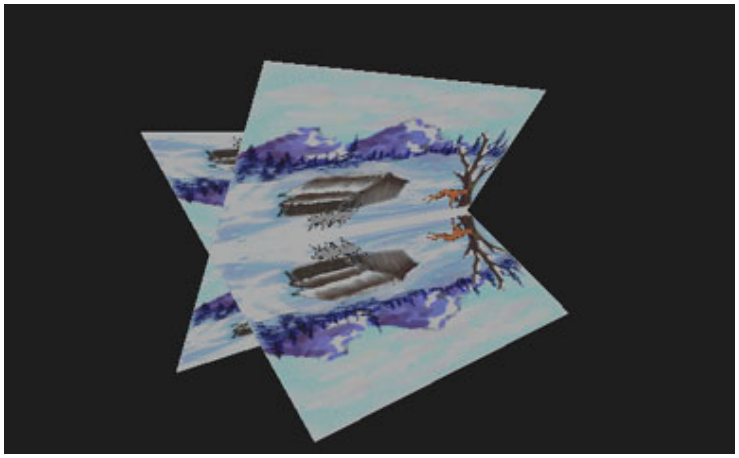
Zur einfacheren Nutzung der DirectSound-API haben wir im Rahmen der Lehrveranstaltung eine CDirectSound-Klasse entwickelt. Implementieren Sie nun eine Klasse CDirect3DSound mit 3D-Soundpuffern und einem entsprechenden 3D-Listener, die über entsprechende Methoden im Raum positioniert werden können.

Testen Sie diese Klasse mit einer kleinen Dialoganwendung, mit der verschiedene Soundquellen vom Nutzer gestartet und gestoppt werden können. Diese Soundquellen sind von links nach rechts vor einem virtuelen "Listener-Interface" zu positionieren. Auch den "Doppler-Effekt" können Sie testen, indem Sie eine Soundquelle während des Abspielens im virtuellen Raum bewegen. Bringen Sie hierzu Kopfhörer mit ins Praktikum :-)

Aufgabe 3: Direct3D

Erstellen Sie ein C++-Projekt (Dialoganwendung) mit Direct3D-Anbindung. Zur Initialisierung der Direct3D-Interfaces können Sie die auf dem Praktikumsordner (P:/bruns/DirectX/Direct3D) bereitgestellten Klassen in der aus der Vorlesung bekannten Beispielanwendung nutzen. Machen Sie sich hierzu deren Aufbau und Nutzung in dem Beispielprojekt deutlich, bevor Sie die Klassen in Ihr eigenes Projekt übernehmen. Erzeugen Sie ein einfaches 3D-Objekt aus 2 Rechtecken wie unten abgebildet.

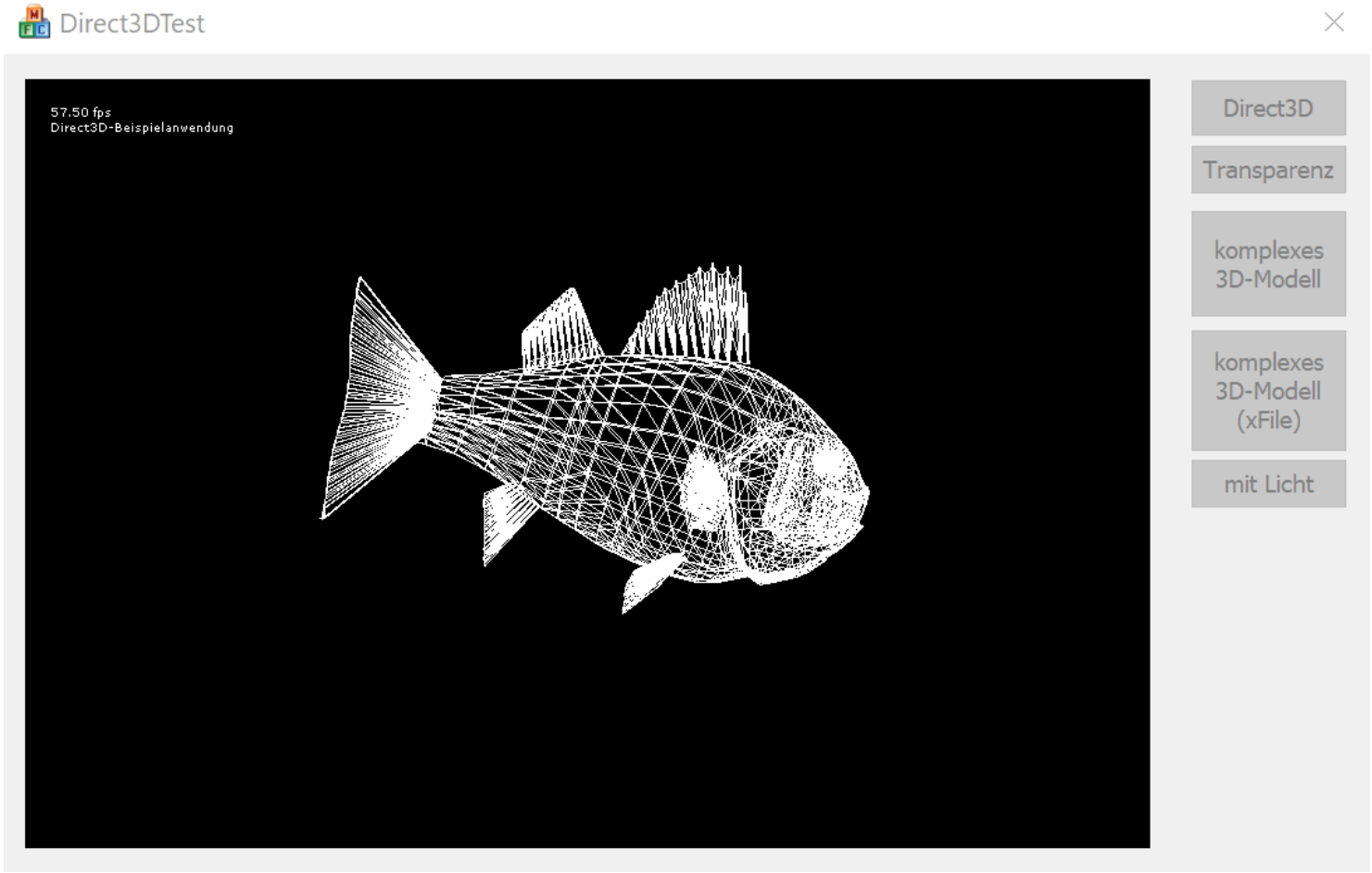
1. Zunächst soll dieses Objekt mit beliebiger Farbe dargestellt werden. Realisieren Sie über Transformationsmatrizen eine langsame Drehung (Rotation) um die x- und y-Achse.
2. Messen Sie die aktuelle Framerate und zeigen diese im Fenster (entweder permanent oder auf Tastendruck) an.
3. Ermöglichen Sie dem Nutzer, das Objekt auch mit der Maus (bei gedrückter linker Maustaste) zu bewegen. Nutzen Sie hierzu die in der Klasse `CDirect3DObject` implementierten Transformationsmatrizen.



Erweitern Sie Ihr Modell mit Texturkoordinaten und laden Sie eine beliebige Textur auf die entsprechenden Rechtecke.

Aufgabe 4: Direct3D und DirectInput

Ihnen ist das 3D-Modell eines Fisches mit ca. 6000 Dreiecken in Form einer Textdatei gegeben. Machen Sie sich mit dem Aufbau dieser Datei vertraut und erklären Sie sich die Methode `CDirect3DObject::BuildFromObtFile(...)`, die dieses 3D-Modell in einen Direct3D-Modellpuffer lädt. Implementieren Sie die Anzeige für dieses Modell mit einer anderen Farbe als weiss und die Möglichkeit für den Nutzer, das Modell mit der Maus zu drehen.



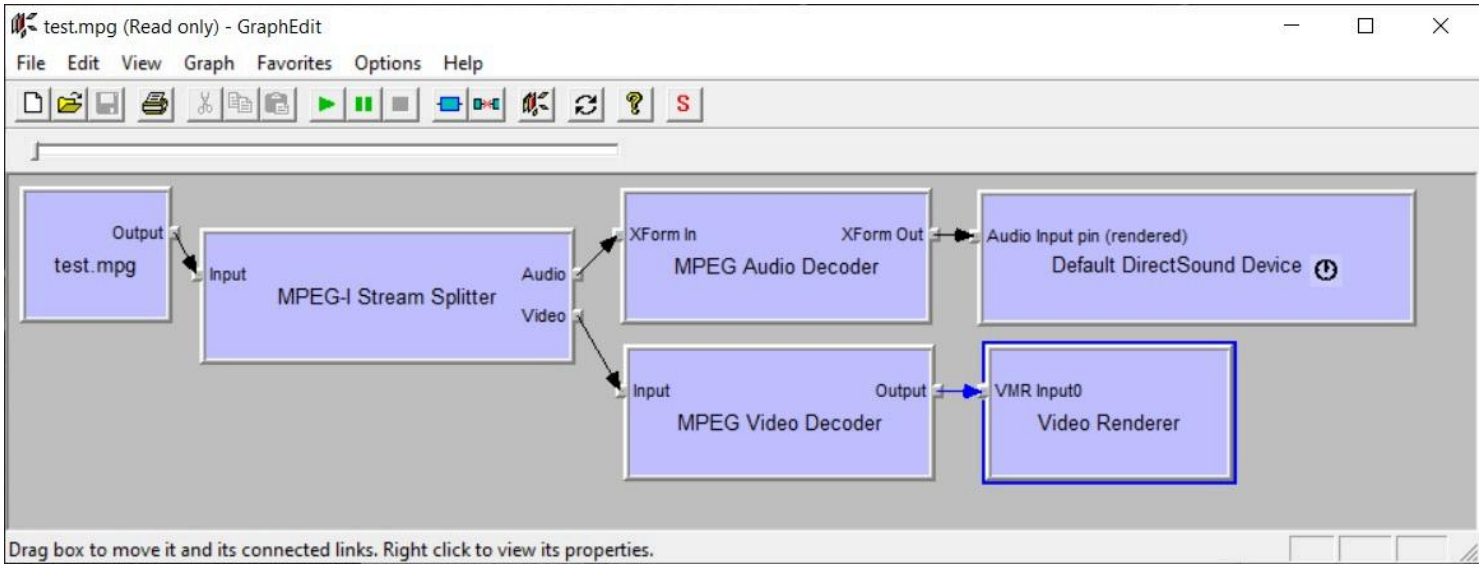
Erweitern Sie Ihr Projekt um die DirectInput-Funktionalität für Tastatureingaben. Wählen Sie jeweils eine Taste (Hotkey) für die Eigenrotation des Fisches um die x-, y-, oder z-Achse und Zoomfunktionalität. Beachten Sie dabei, dass die Tastaturabfrage

Bestandteil der Animationsschleife ist. Testen Sie, ob bei gleichzeitiger Betätigung mehrerer Tasten, die entsprechende Funktionalität auch gleichzeitig ausgeführt wird.

Termine: Bearbeitung siehe Zeitplan.

Aufgabe 5: DirectShow-Graphen erstellen und mit dem Programm GraphEdit testen

Starten Sie das Werkzeug "GraphEdit" und ziehen Sie "per Drag & Drop" eine avi-Datei (oder mpeg-Datei) in diese Anwendung. Dadurch wird automatisch ein Filtergraph erzeugt. Machen Sie sich diesen Filtergraphen plausibel und testen Sie diesen Graphen (Play, Pause, Stop).



Aufgabe 6: DirectShow - Wiedergabefunktionalität

Erstellen Sie ein Abspielprogramm für Videodateien. Die folgende Funktionalität ist dabei zu implementieren:

- Der Dateiname ist über eine Dateidialogbox (CFileDialog) auszuwählen (oder via "Drag and Drop" an das Programm zu übermitteln) **[1P]**.
- Das Abspielen erfolgt asynchron und das Video ist im aktuellen (Dialog)Fenster darzustellen **[1P]**.
- Die aktuelle Abspielposition ist anzuzeigen (Textfeld und Schieberegler). Die aktuelle Abspielposition kann über den Schieberegler eingestellt werden. **[1P]**
- Die Ausgabe kann in den Vollbildmodus geschalten werden. Der Vollbildmodus kann mit der linken Maustaste wieder verlassen werden. Am Ende des Abspielvorgangs schaltet die Anwendung (automatisch) wieder in den Fenstermodus und an den Anfang des Videos. **[1P]**
- Kapseln Sie die Funktionalität von DirectShow sinnvoll und wiederverwendbar in einer eigenen Klasse **[1P]**.

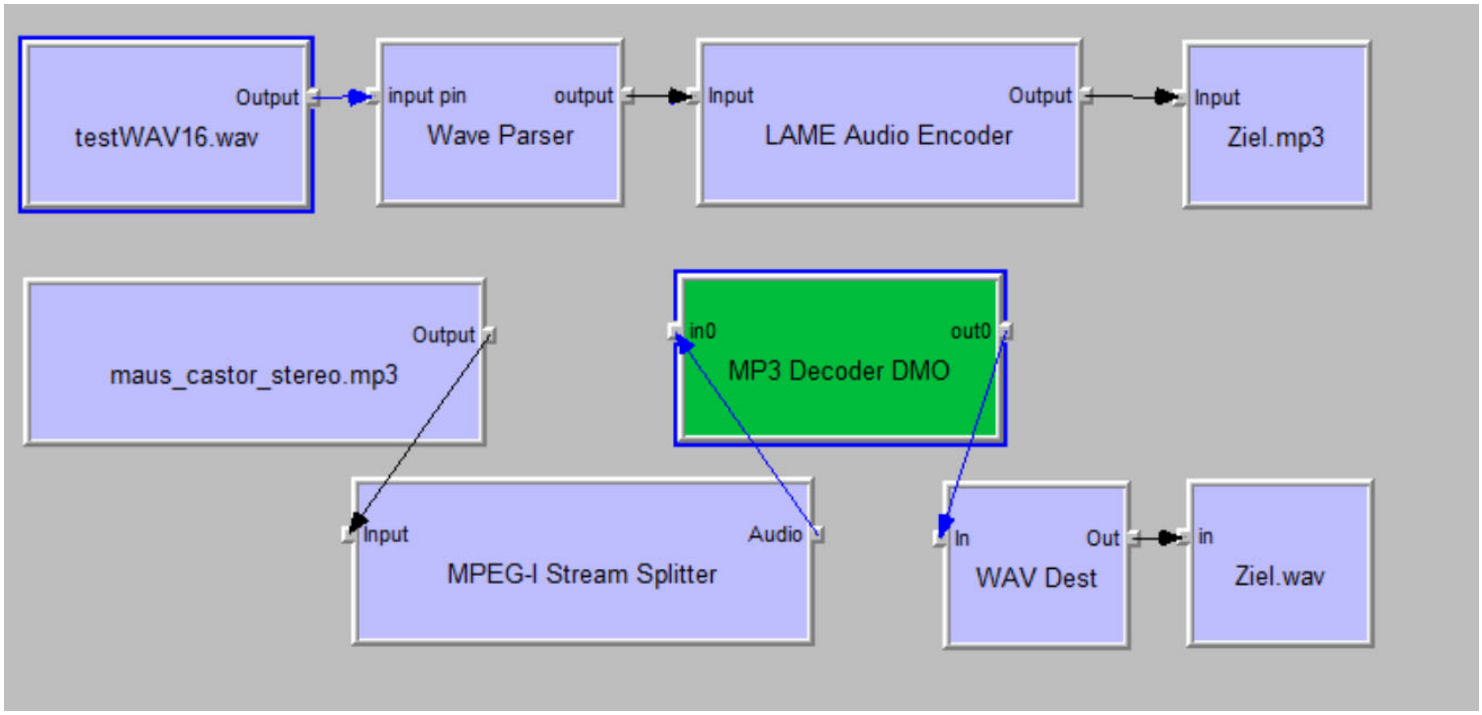
Nutzen Sie hierfür das in der Vorlesung entwickelte Beispiel und lassen Sie sich den Filtergraphen automatisch mit Hilfe der Methode "RenderFile" erstellen. Testen Sie Ihre Anwendung.

Aufgabe 7: DirectShow - mp3_to_wav und wav_to_mp3 Encoder

Erstellen Sie eine Dialoganwendung, welche 2 Dateinamen als Eingabe erhält und wahlweise eine mp3-Datei in eine wav-Datei oder umgekehrt umwandelt. Sie können als Vorlage auch eine Kopie Ihres Programms aus der vorangegangenen Aufgabe verwenden.

Ereugen Sie die entsprechenden Filtergraphen manuell. Die Dateinamen für Quell- und Zieldatei sind über ein `IFileSourceFilter` -Interface (bzw. über ein `IFileSinkFilter` -Interface) zu benennen. Fügen Sie die erzeugten Filter dem Filtergraphen hinzu und

verbinden Sie die entsprechenden Pins wie in der nachstehenden Grafik angegeben (achten Sie dabei auf die genaue Pin-Bezeichnung!).



Wenn die Filter in den Graphen eingefügt sind, können Sie die Filter-Interfaces wieder freigeben (Release) und den Graphen testen. Vergessen Sie nicht, ggf. die für diese Aufgabe zusätzlich notwendigen DirectShow- Filter zu registrieren. Die Filter finden im Praktikumsordner auf P:\bruns für Ihren Rechner daheim. An der Hochschule sollte alles richtig installiert/registriert sein. Dies könnten Sie aber auch aus Ihrem Programm heraus zur Laufzeit tun:

```
RegisterFilterDll("DUMP.AX",true);
RegisterFilterDll("lame_dshow.ax",true);
RegisterFilterDll("WAVDEST.AX",true);
```

Steuern Sie den Filtergraphen über die in der Vorlesung besprochenen COM-Interfaces. Zerstören Sie den Filtergraphen im Ereignishandler EC_COMPLETE bzw. EC_USERABORT.

Der mp3-Encoder unterstützt einen Einstellungsdialog, nutzen Sie diesen.

Termine: Bearbeitung und Abnahmen siehe Zeitplan. Die Punkte sollten nach Möglichkeit während der regulären Praktika erworben werden. Weitere 5 Punkte sind bei korrekter Beantwortung der theoretischen Fragen während der Abnahme möglich. Die Aufgabe 7 können Sie zusätzlich lösen und damit das Grundverständnis zu DirectShow festigen. Bei auftretenden Problemen helfen wir Ihnen auch hier :-)

RIFF-Dateien

Aufgabe 1: Erzeugen einer wav-Datei

Schreiben Sie ein Programm (oder erweitern Sie Ihr Programm aus dem DirectSound-Komplex) mit folgender Funktionalität:

- Es ist eine wav-Datei für 16-Bit Samples und 44100 Hz Abtastfrequenz zu erzeugen, die einen Audiokanal enthält. Generieren Sie in diese Datei zwei Sekunden Sinussound mit einer Frequenz von 5 kHz.
- Legen Sie eine weitere wav-Datei mit den gleichen Eigenschaften an, die jedoch mit 8-Bit Samples arbeitet.

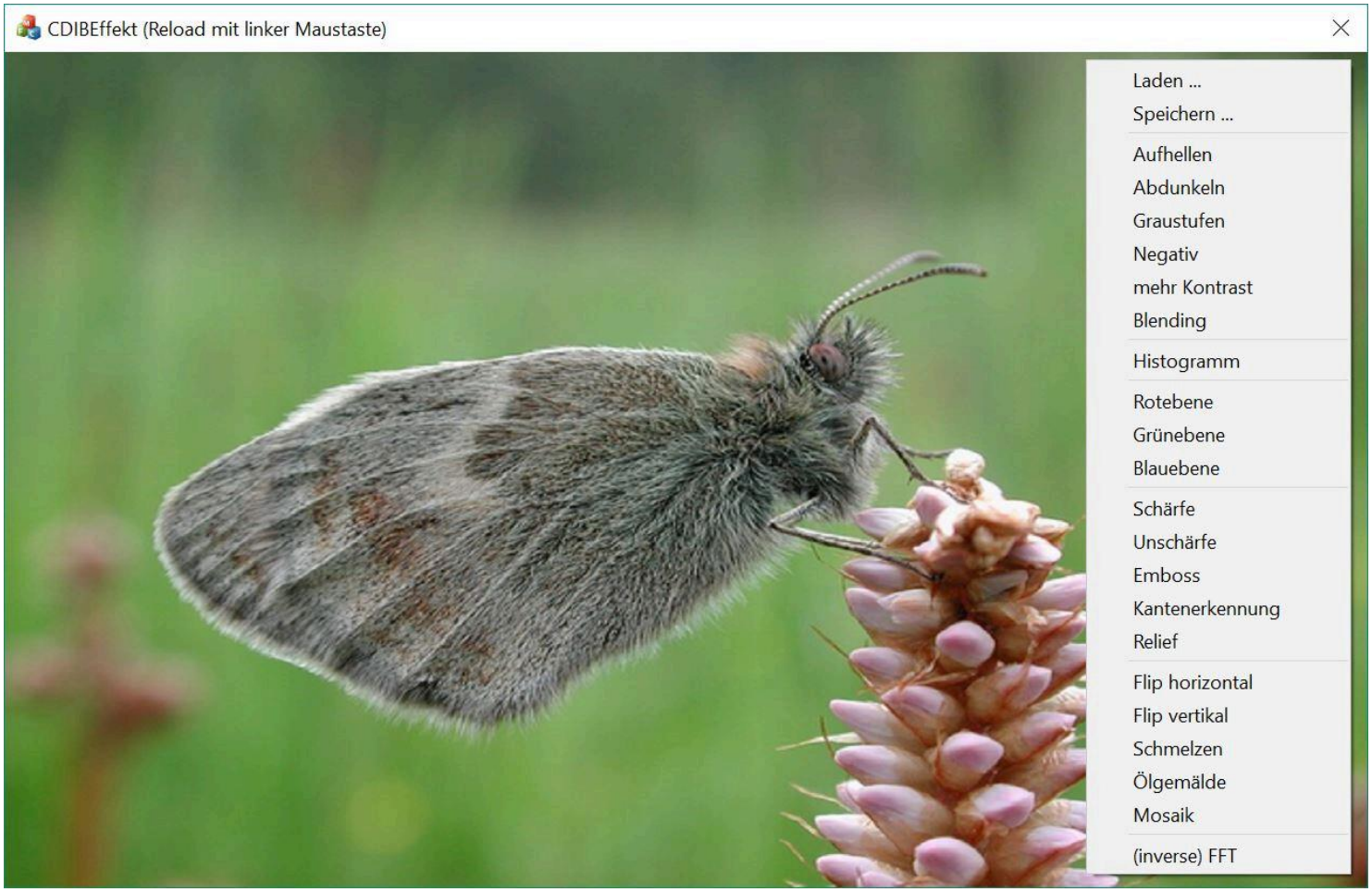
Lassen Sie sich das Ergebnis grafisch (d.h. den Sampleverlauf) und akustisch ausgeben. Sollten Sie das Fach MFC-Windowsprogrammierung besucht haben, realisieren Sie bitte entsprechende Steuerelemente, um die Audioeigenschaften (d.h. Abtastfrequenz, Anzahl der Audiokanäle, usw.), die zu generierende Frequenz und den Dateinamen direkt eingeben zu können.

Pixelgrafiken

Aufgabe 1: Darstellung einer Bilddatei

Erstellen Sie ein neues MFC-Projekt. Die dialogbasierte Applikation hat eine Schaltflächen mit der Aufschrift "Zeige das Bild". Wenn diese Schaltfläche gedrückt wird, zeigt Ihr Programm eine Bilddatei im bmp-Format. Nutzen Sie die in der Vorlesung erarbeitete Klasse CDIB **[1P]**.

Erweitern Sie die Klasse CDIB um eine Methode DrawRect, die es Ihnen ermöglicht, die Bitmap bei der Ausgabe zu skalieren. Fügen Sie Ihrer Applikation zwei weitere Schaltflächen hinzu, die zur Vergrößerung bzw. zur Verkleinerung der Darstellung dienen **[1P]**. Gern können Sie auch ein in seiner Grösse variables Dialogfenster nutzen und das Ereignis WM_SIZE entsprechend behandeln. Siehe hier:



Aufgabe 2: Das jpg-Dateiformat

Ihr Projekt aus der vorangegangenen Aufgabe ist zu erweitern. Ziel dieser Erweiterung ist das Laden und Speichern von Pixelgrafiken im jpg-Dateiformat. Nutzen Sie hierzu die in der Vorlesung entwickelten Methoden der Klasse CDIB. Ein Schieberegler für die Qualität der Komprimierung ist ebenfalls einzufügen.

Testen Sie das Programm indem Sie ein Bild im bmp-Format laden und im jpg-Format abspeichern bzw. umgekehrt. Versuchen Sie, die von Ihnen erstellte Datei in einem anderem Grafikprogramm anzuzeigen **[1P]**.

Aufgabe 3: Einfache Grafikfilter

Erweitern Sie die Klasse CDIB mit den in der Vorlesung besprochenen Methoden zur Bildbearbeitung und die Anzeige des Histogramms (siehe Screenshot - außer Schmelzen, Ölgemälde und Mosaik). Für das Überblenden zweier Bilder ist ein Slider einzufügen (ggf. als nichtmodaler Dialog), der die prozentuale Überblendung zwischen 0 und 100 Prozent einstellbar macht. Programmieren Sie zusätzlich eine Methode zum Abdunkeln eines Bildes. Implementieren Sie einen Framework zum Testen der Klasse (zum Beispiel in der oben dargestellten Form als Kontextmenu) **[2P]**. Arbeiten Sie schrittweise, indem Sie mit einfachen Funktionen beginnen. ;-)

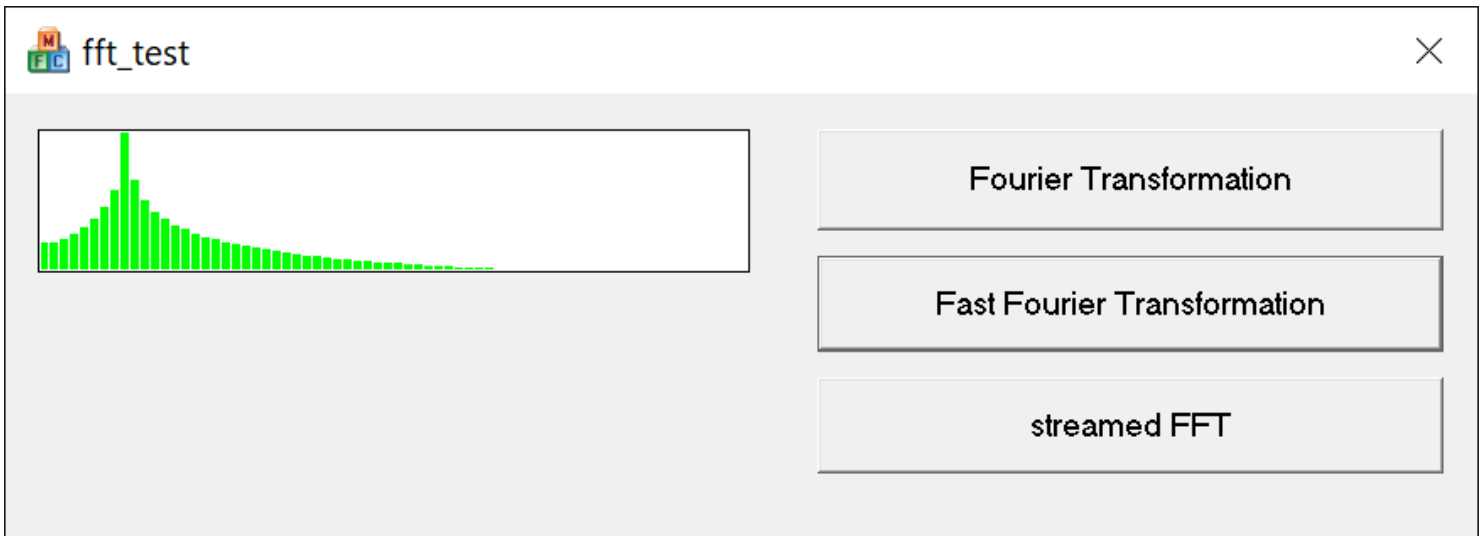
Termine: Bearbeitung und Abnahmen siehe Zeitplan. Die Punkte sollten nach Möglichkeit während der regulären Praktika erworben werden. Weitere 5 Punkte sind bei korrekter Beantwortung der Fragen während der Abnahme möglich.

Fast Fourier Transformation

Aufgabe 1: eine FFT (zeitsynchron) berechnen

Erweitern Sie Ihre Praktikumslösung zum "Abspielen von Sounddaten mit DirectSound" um die in der Vorlesung behandelte FFT-Funktionalität.

Fügen Sie hierzu ein neues Timer-Ereignis in die Applikation ein, das in einem Intervall <50ms eine FFT über den jeweils aktuell abgespielten Teil des Soundpuffers berechnet und grafisch darstellt.



Der FFT-Algorithmus ist (wiederverwendbar) als neue Methode der DirectSound-Klasse zu implementieren.

Termine: Die Aufgabe kann freiwillig gelöst werden und hat keine Prüfungsrelevanz.

Der **Zeitplan** für die Praktika im **Wintersemester 2024/25** sieht wie folgt aus:

- **40. und 41. KW:** es finden nur die Vorlesungen statt
- **42. und 43. KW:** Media Control Interface (MCI)
- **44. und 45. KW:** DirectSound
- **46. KW:** noch offene Abnahmen APL (alternative Prüfungsleistung) für MCI und DirectSound
- **47. und 48. KW:** Direct3D
- **49. und 50. KW:** DirectShow
- **51. und 2. KW:** Pixelgrafiken
- **3. und 4. KW:** noch offene Abnahmen APL (alternative Prüfungsleistung) für DirectShow und Pixelgrafiken

Direkteinstieg

Programmierung I (prog1.html)
Programmierung II (prog2.html)
Fotografie/Bildgestaltung (fotografie.html)
Windowsprogrammierung (winprog.html)
Audio-/ Video-/ Grafikprogrammierung (mmprog.html)
Programmierung mit Swift (swiftprog.html)

Publikationen (publikationen.html)
Abschlussarbeiten (diplom.html)

studentisches Praktikum (praktikum.html)
weitere Infos für Studierende (infos.html)

Direkteinstieg

Studienmöglichkeiten (http://www.htw-dresden.de/studium/studieninteressierte/studienmoeglichkeiten.html)
Bibliothek (http://www.htw-dresden.de/bib.html)
Rechenzentrum (http://www.htw-dresden.de/rz.html)
Sprachenzentrum (http://www.htw-dresden.de/sprachenzentrum.html)
StuRa (http://www.htw-dresden.de/hochschule/stura.html)
Veranstaltungskalender (http://www.htw-dresden.de/service/veranstaltungen/veranstaltungskalender.html)
HTW Aktuell (http://www.htw-dresden.de/hochschule/htw-aktuell.html)
Seitenüberblick (http://www.htw-dresden.de/servicemenue/seitenueberblick.html)
Impressum (http://www.htw-dresden.de/servicemenue/impressum.html)

Kontakt

Hochschule für Technik und Wirtschaft Dresden
Fakultät Informatik/Mathematik
Friedrich-List-Platz 1
D-01069

Postanschrift:
PF 120701, D-01008 Dresden

Tel.: +49 (0)351 462 3546
E-Mail: bruns@informatik.htw-dresden.de (mailto:bruns@informatik.htw-dresden.de)