Schulversuche mit der Soundkarte

Dokumentation

.Friedrich-List-Gymnasium Reutlingen

.Seminarkurs „Computereinsatz im naturwissenschaftlichen Unterricht“

.Schuljahr 2004 / 2005

.Projektgruppe „Soundkarte“

.Niklas Reisser
.Benedikt Rudolph
.Oliver Stickel

Inhaltsverzeichnis

Vorwort

Das vorliegende Dokument stellt die Dokumentation der Projektruppe „Schulversuche mit der Soundkarte“ im Seminarkurs 2004/2005 „Computereinsatz im naturwissenschaftlichen Unterricht“ unter der Leitung von Herrn Schich am Friedrich-List-Gymnasium Reutlingen dar. Dieser Kurs befasste sich damit, neue Möglichkeiten zu finden, den Computer als Unterrichtsmedium ins Gymnasium zu integrieren. Hierzu wurden, im Einklang mit der kultusministeriellen Vorgabe, dass ein Seminarkurs die „intensive Einübung studien- beziehungsweise berufsvorbereitender Arbeitsmethoden, vor allem selbstgesteuertes Lernen“ zum Ziel hat, im ersten Halbjahr Arbeits- und Präsentationstechniken, im Besonderen auch die Projektarbeit, gezielt erlernt und trainiert. Im zweiten Halbjahr folgte dann die Arbeit in einzelnen Projektgruppen mit selbstgewählten Themen passend zum Oberthema des Seminarkurses. Hier fand dann unsere Gruppe, bestehend aus Niklas Reisser, Benedikt Rudolph und Oliver Stickel zusammen und wir wählten das Thema „Messwerterfassung mit der Soundkarte“, was soviel bedeutet wie die Erstellung von physikalischen Versuchen mit anschließender Messwerterfassung an einem Computer, für uns aus. Die Schnittstelle, also den Übergang zwischen Versuchsaufbau und Computer stellt hierbei die Soundkarte dar. Das Ziel war es, in projektorientierter Arbeit ein Produkt zu erstellen, das sich anschließend im Physikunterricht auch wirklich einsetzen lässt und nicht, wie es an der Schule zu unserem Leidwesen so oft geschieht, ins Leere zu arbeiten. Unter diesem Leitgedanken entwickelten wir die Struktur unseres Projekts. Wir nahmen uns vor, ein Komplettpaket namens XESCOE – „x-tendable environment for soundcard operated experiments“, also sinngemäß „erweiterbare Arbeitsumgebung für Experimente auf Basis einer Soundkarte“ - zu erstellen. Diese sollte physikalische Versuche inklusive Anleitungen und Erklärungen, sowie ein komplett selbstgeschriebenes Messwerterfassungsprogramm für den PC umfassen. Diese Software sollte erweiterbar sein, so dass jeder interessierte Lehrer oder auch Schüler neue Versuche in die Umgebung integrieren kann. Damit ist gewährleistet, dass unsere Arbeit auch auf zukünftige Sicht nützlich sein kann und der Schule somit einen reellen Nutzen bietet. Über diese strukturelle Einteilung des Projekts kamen wir auch zu unserer Arbeitsteilung: Benedikt, der schon länger programmiert, kümmerte sich um die Software, Oliver um die grafische Umsetzung der Software und sonstige Designfragen, sowie um einen Teil der Versuche, Niklas war für die übrigen Versuche zuständig.
Diese Dokumentation ist eigentlich nur ein ausschnitthafter Überblick über unsere Arbeit, denn sie ist sehr auf das Ergbnis fokussiert. Die Entstehung dagegen ist in all ihren – teilweise auch sehr problembehafteten – Facetten in unseren individuellen Portfolios, das Protokolle unserer Arbeit, sowie den persönlichen Lernfortschritt beinhaltet, nachvollziehbar.

Hiermit versichern wir, diese Dokumentation eigenständig und nur mithilfe der angegebenen Quellen angefertigt zu haben.

Niklas Reisser Benedikt Rudolph Oliver Stickel

Reutlingen, den 9.6.2005

Software

Einleitung

Hinter der Idee, den PC als Erfassungsgerät für Messwerte einzusetzen, steckt die Tatsache, dass die Soundkarte in der Lage ist, analoge Signale unter hohen Frequenzen zu digitalisieren. Da heutzutage praktisch jeder PC mit einer solchen Soundkarte ausgestattet ist, besteht also eine gute Verfügbarkeit von notwendiger Hardware für die Messwerterfassung. Mit dem meistverwendeten Betriebssystem Microsoft Windows wird standardmäßig der Audiorekorder mitgeliefert, mit dem sich die Messungen der Soundkarte als gewöhnliche Audiodateien speichern lassen. Lediglich ein Programm zur Auswertung dieser Daten ist nicht vorhanden. Dafür lassen sich Audioprogramme wie etwa GoldWave oder der Nero Wave Editor zweckentfremden, indem man manuell innerhalb der Visualisierungen nach markanten Veränderungen der Messwerte (d.h. der gemessenen Amplitude) sucht und damit die für die Berechnungen nötigen Zeitintervalle bestimmt. Um diese etwas primitiv anmutende Technik unnötig zu machen, habe ich es mir zur Aufgabe gemacht, ein Programm zu schreiben, das die Auswertungen der Audiodaten automatisch durchführt. Ein Leitgedanke dabei ist die Erweiterung, also die Möglichkeit, neue Versuchsauswertungen in das Programm zu integrieren, ohne dabei den Programmcode selbst verändern zu müssen. Dadurch entsteht auch der Name der Software, XESCOE, was für eXtendable Environment for Sound Card Operated Experiments, zu Deutsch Erweiterbare Umgebung für Soundkarten-basierte Versuche steht.

Funktion und Aufbau

XESCOE soll, wie in der Einleitung erwähnt, auf der Ausgabe des Audiorekorders aufsetzen, muss folglich in der Lage sein, Audiodateien zu öffnen und darin diejenigen Stellen zu erkennen, bei denen der Versuchsaufbau deutliche Änderungen des Audiosignals liefert. Dabei gilt das Prinzip, das ein Versuch zwei mögliche Signale an die Soundkarte schickt, nämlich ein Geräusch bzw. Ton und kein bzw. ein sehr leises Signal, was unter eine bestimmte Toleranzgrenze fällt. Über eine Funktion wird die Audiodatei in Abschnitte ohne und mit Signal eingeteilt. Aus den resultierenden Wechselstellen lassen sich alle gesuchten Zeitintervalle berechnen.
Welche Intervalle aber wie interpretiert werden müssen ist versuchsbedingt. Genauso ist es unterschiedlich, welche Toleranzgrenze benutzt werden muss, da manche Versuche sehr saubere Signale ausgeben während andere Nebengeräusche und Störfrequenzen liefern. Um XESCOE erweiterbar zu halten dürfen diese Einstellungen nicht im Programmcode festgelegt werden, sondern müssen „von außen" veränderbar sein. Diese Möglichkeit wird über eine einfache Skriptsprache gegeben, die sich schnell erlernen lässt und die komplett verschiedene Versuchsauswertungen zulässt (s. nächster Abschnitt).
Letztlich stellt sich die Frage, wie diese Anforderungen in den Aufbau der Software umgesetzt werden sollen. Wir haben uns schließlich für eine sog. Wizard-Struktur entschieden, was in etwa bedeutet, dass sich dem Anwender hintereinander mehrere Seiten, auf denen Einstellungen vorgenommen werden, öffnen. XESCOE ist in drei Seiten aufgeteilt. Die erste Seite listet alle möglichen Versuche auf, die durchgeführt werden können. Versuch bedeutet in diesem Fall ein Auswertungsskript, das in einem Unterverzeichnis der Software (...\XESCOE\experiments) mit der Dateiendung .exp gespeichert wird. Im einen Fenster auf dieser Seite werden alle Versuchsdateien aufgelistet, im anderen erscheint eine Beschreibung des markierten Versuchs.
Ist ein Versuch ausgewählt, lässt sich über den „Versuch Starten"-Button die nächste Seite aufrufen. Sie enthält ein Eingabefenster für den Pfad der aufgezeichneten Audiodatei. Ich gehe davon aus, dass die meisten Anwender ihre Versuche mit dem Audiorekorder aufnehmen werden, was aber nicht unbedingt notwendig ist. XESCOE kann allerdings nur unkomprimierte WAVE-Dateien verarbeiten, je nach Versuch kann es sein, dass eine Stereo-Aufnahme (z.B. zur Messung der Schallgeschwindigkeit) benötigt wird. Ist der Pfad einer gültigen Datei eingegeben worden, so wird diese auf die Signalwechsel hin überprüft und anschließend die dritte Seite aufgerufen. Da hier die Auswertung der Daten stattfindet, ist der Aufbau und die Funktionalität der Seite komplett über das Skript gesteuert. Daher lässt sich über den Inhalt der Seite nicht viel sagen, wohl aber über die Möglichkeiten ihrer Gestaltung.

Skripting

Die Aufgabe eines Versuchsskripts besteht darin, aus den zuvor erkannten Signalwechseln die notwendigen Zeitintervalle zu errechnen, evtl. weitere Eingaben vom Anwender anzunehmen, wie etwa die gemessene Länge einer Strecke, und aus diesen Angaben ein bestimmtes Ergebnis zu berechnen und auszugeben. Außerdem sollte das Skript eine kurze physikalische Erläuterung zum Versuch anzeigen können. Für die Umsetzung bedeutet dies, dass ein Skript verschiedene grafische Bedienelemente wie Textfelder, Buttons etc. auf der dritten Seite platzieren, die Wechselstellen aus der Audiodatei auslesen, Eingaben vom Benutzer entgegennehmen und schließlich rechnen können muss. Außerdem muss es die Toleranzgrenze bei den Signalanalysen festlegen und eine Kurzbeschreibung des Versuchs für die erste Seite liefern.
Um das Skripting einfach genug zu halten, benutze ich nur wenige mögliche Skritsprachenelemente. Ein Skript ist eine einfache Textdatei, die Zeile für Zeile von XESCOE ausgeführt wird. Variablen und mathematische Operatoren gibt es keine, nur Befehle. Jeder Befehl hat eine Liste von Parametern, die er verarbeitet. So würde man beispielsweise um Zahlen zu addieren, dem Additionsbefehl alle Summanden als Parameter übergeben. Jeder Befehl bildet einen Rückgabewert, der wiederum einem anderen Befehl als Parameter übergeben werden kann. Um also z.B. eine Summe mit einer Zahl zu multiplizieren, erhält der Multiplikationsbefehl als einen Parameter den Additionsbefehl mit den beiden Summanden. So lassen sich beliebig tiefe Verschachtelungen realisieren. Die einzelnen Parameter werden durch spitze Klammern eingeschlossen.

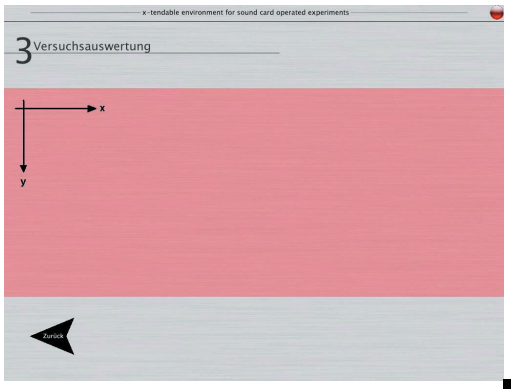
Beispiele:
`add<2><5><1>` entspricht 2 + 5 + 1 und gibt 8 zurück
`mul<add<2><5>><2>` entspricht (2 + 5) * 2 und gibt 14 zurück

Für die Berechnungen werden natürlich die Wechselstellen der Signale gebraucht. Die Befehle orientieren sich dabei an den Stellen ohne Signal. Durch `findzerostart` wird der Anfang und durch `findzeroend` das Ende eines Abschnitts ohne Signal abgefragt. Dabei wird die Zeit ab Beginn der Audiodatei in Millisekunden zurückgegeben. Die Befehle erhalten zwei Parameter, der erste bestimmt den Kanal (muss auch bei Monoaufnahmen angegeben werden), der zweite die Nummer der entsprechenden Wechselstelle. Für beide Parameter gilt, dass die 0 den ersten Kanal bzw. die erste Wechselstelle angibt.

Beispiel:
`sub<findzeroend<0><0>><findzerostart<0><0>>` gibt die Dauer des ersten Abschnitts ohne Signal zurück

Über Befehle lassen sich auch grafische Elemente auf dem Bildschirm ausgeben. Diese benötigen die Positionsangaben eines umgebenden Rechtecks. Dabei muss beachtet werden, dass nicht die ganze dritte Seite zugänglich ist. Nur auf dem rot unterlegten Bereich (s. Bild) können Elemente dargestellt werden. Als Einheit werden keine Pixel verwendet, sondern, um von der Bildschirmauflösung unabhängig zu bleiben, relative Koordinaten, wobei der rote Bereich 250 Einheiten breit und 100 Einheiten hoch ist. Die Koordinaten werden nach rechts und nach unten hin größer. Siehe Abb. 1.
Beispiel:
`winin<125><50><80>` erzeugt ein Feld zur Texteingabe durch den Benutzer mit der oberen linken Ecke in der Mitte des roten Bereichs und einer Breite von 80 Einheiten. Da Eingabefelder immer einzeilig sind, ist die Höhe festgelegt und kann nicht verändert werden.

Abb. 1



Um den gefordeten Inhalt eines Textfelds klarszustellen gibt es auch noch einen alternativen Befehl für die Erzeugung von Textfeldern mit Namen `descrwinin`. Dieser nimmt als vierten Parameter zusätzlich eine

Beschriftung entgegen die rechtsbündig unter dem Textfeld angezeigt wird.

Wird ein Textfeld wie im genannten Beispiel erzeugt, so wird ihm automatisch eine Nummer zugewiesen, die abhängig von der Anzahl der vorher erzeugten Textfelder ist. So erhält das im Skript als erstes benannte Feld die Nummer 0, das zweite die 1 usw. Über diese Nummer kann dann der Inhalt des Textfelds mithilfe des Befehls getwinin abgefragt werden.

Beispiel:
`mul<2><getwinin<0>>` multipliziert den Inhalt des ersten Textfelds mit 2

Alle bisher beschriebenen Befehle werden sofort nach Erscheinen der dritten Programmseite aufgerufen. Es kann aber sein, dass bestimmte Befehle erst später ausgeführt werden sollen, da zum Beispiel ein Textfeld am Anfang immer leer ist und der Andwender erst Zeit braucht, die entsprechenden Daten einzugeben. Um dann eine Berechnung zu starten ist es am Besten, wenn er auf einen Button klicken kann. Im Skript kennzeichnet man einen Befehl, der erst beim Klicken eines Buttons ausgeführt wird dadurch, dass man ihn einem Befehlsblock zuordnet. Dies geschieht über den Befehl `block`. Wie bei den Textfeldern erhalten die Blocks in der Reihenfolge ihrer Benennung Identifikationsnummern. Ein Befehlsblock kann, wie der Name bereits sagt, mehrere Befehle enthalten. In der Zeile unter dem block-Befehl muss eine Zahl stehen, die angibt, wieviele der folgenden Zeilen zum Block gehören sollen. Ein Button wird dann über den Befehl `button` erzeugt, der die Nummer des auszuführenden Blocks entgegennimmt.

Beispiel:
```
block
2
add<2><5>
mul<2><getwinin<0>>
button<20><10><Klick!><0>
```
erzeugt einen Button mit der Aufschrift „Klick!“ und ordnet ihm den erstgenannten Befehlsblock zu. Dieser besteht aus zwei Zeilen, nämlich einer Addition und einer Multiplikation mit der Abfrage eines Textfelds.

Für eine Versuchsauswertung fehlt jetzt lediglich noch die Möglichkeit, Text auf dem Bildschirm auszugeben. Der Befehl `textout` gibt einzeiligen Text mit festgelegter Maximalbreite aus.

Beispiel:
`textout<20><10><80><Das Ergebnis ist ><add<2><3>>`
Gibt „Das Ergebnis ist 5.00“ aus. Sollen Rechnungen in Text eingebaut werden, so muss dies über einen extra Parameter geschehen, ansonsten wird ein Befehl nicht ausgeführt, sondern ausgegeben.

Soll nicht nur eine einzelne Zeile, sondern ein ganzer Text ausgegeben werden, so steht der Befehl `winout` zur Verfügung. Er erzeugt ein Ausgabefenster ähnlich dem Versuchsbeschreibungsfenster auf der ersten Programmseite, allerdings mit variabler Überschrift. Der Inhalt des Ausgabefensters lässt sich formatieren. Ein „§“ im Text führt zu einem Zeilenumbruch, mit „$“, „#“ und „|“ wird der folgende Text kursiv, fett und fett-kursiv angezeigt. Ein Formatierungszeichen gilt bis zur nächsten Umstellung, bzw. bis zur Wiederholung des Zeichens, was den folgenden Text wieder in Normaldruck anzeigt.

Beispiel:
```
winout<20><10><100><70><Überschrift><Hier können $beispielsweise
#physikalische Erklärungen# angezeigt werden.>
```
Erzeugt ein Ausgabefenster mit der Breite 100, der Höhe 70, der Betitelung „Überschrift“ und dem Inhalt „Hier können *beispielsweise* **physikalische Erklärungen** angezeigt werden.“

Außerdem gibt es die Möglichkeit zur Veranschaulichung der Berechnungen eine Visualisierung der WAVE-Daten anzuzeigen. Diese Aufgabe wird vom Befehl `visu` übernommen. Die ersten vier Parameter werden für Position, Breite und Höhe gebraucht, der vierte Parameter nimmt den Kanal entgegen, die letzten beiden Parameter geben den Start und das Ende der zu visualierenden Daten innerhalb der Datei in Milisekunden an.

Beispiel:
`visu<10><10><100><30><0><1000><5000>`
Zeigt eine Visualisierung des ersten Kanals von Sekunde 1 bis 5 an.

Hier noch eine vollständige Übersicht über alle möglichen Befehle:
`add<x><y><z>` gibt x + y + z zurück
`sub<x><y><z>` gibt x - y - z zurück
`mul<x><y><z>` gibt x * y * z zurück
`div<x><y><z>` gibt (x / y) / z zurück
`quad<x><y>` gibt x^y zurück
`sqrt<x>` gibt die Wurzel aus x zurück

`findzerostart<x><y>` gibt den Beginn des Abschnitts y ohne Signal auf Kanal x zurück
`findzeroend<x><y>` gibt das Ende des Abschnitts y ohne Signal auf Kanal x zurück

`textout<x><y><w><text><...>` erzeugt an der Stelle x | y ein Textausgabefeld ohne Hintergrund mit maximaler Breite w und gibt „text...“ aus
`winout<x><y><w><h><Titel><text><...>` erzeugt an der Stelle x | y ein Ausgabefenster mit der Breite w, der Höhe h, der Überschrift „Titel“ und dem Inhalt „text...“. Über „§“ erreicht man einen Zeilenumbruch, ansonsten wird automatisch bei Leerzeichen umgebrochen.„$“ erzeugt kursiven,„#“ fetten und„ | “ fett-kursiven Text.
`visu<x><y><w><h><c><a><b>` erzeugt an der Stelle x | y eine Visualisierung des Kanals c mit der Breite w und der Höhe h von Milisekunde a bis b.
`winin<x><y><w>` erzeugt an der Stelle x | y ein Texteingabefeld der Breite w und automatischer Identifikationsnummer
`descrwinin<x><y><w><Beschriftung>` wie winin, nur mit zusätzlicher Beschriftung
`getwinin<x>` gibt den Inhalt des Textfelds x zurück
`block`
`x`
benennt einen Befehlsblock mit den x folgenden Zeilen und automatischer Identifikationsnummer
`button<x><y><Titel><z>` erzeugt einen Button mit der Aufschrift „Titel“ und führt bei Klick den Befehlsblock z aus
`analbutton<x><y><z>` erzeugt einen Button mit der Aufschrift „Auswertung“ und führt bei Klick den Befehlsblock z aus

`descr<Beschreibung des Versuchs>` legt den Inhalt des Versuchsbeschreibungsfensters auf der ersten Programmseite fest. Der Inhalt lässt sich wie bei einem winout formatieren.
`setlim<x><y><z>` legt die Toleranzgrenze auf x (Werte von 0 - 100), die minimale Dauer ohne Unterbrechnungen für einen Abschnitt ohne Signal auf y und den minimalen Prozentanteil an Signalen oberhalb der Toleranzgrenze für einen Abschnitt mit Signal innerhalb der Zeit y auf z fest.

Technische Aspekte

Um niemanden unnötigerweise zu langweilen, wird dieser Abschnitt entsprechend kurz gehalten und nur auf die wesentlichen Eigenschaften der Software und die Techniken zur Audioanalyse eingegangen, die als Herzstück des Programms angesehen werden können.
XESCOE ist natürlich ein grafisches Programm, das auch standardisierte grafische Elemente enthält, trotzdem aber kein gewöhnliches Windowsprogramm. Zum Verständnis dieser Problematik

ist es notwendig zu wissen, dass die Programmiersprache C++, in der XESCOE geschrieben ist, keine Möglichkeiten zur grafischen Darstellung besitzt (was unter anderem daran liegt, dass C++ älter ist als das Konzept der grafischen Bedienoberflächen). Daher ist es notwenig, eine zusätzliche Programmbibliothek, die vom Betriebssystem gestellt wird, herbeizuziehen. Wir verwenden eine Alternative dazu, nämlich eine Programmbibliothek namens Allegro, die eigentlich zur Erstellung von Computerspielen gedacht ist. Das bringt zwei Vorteile mit sich: Erstens ist Allegro plattformunabhängig, d.h. Allegro-Programme könnten auch unter Linux, MacOS o.ä. kompiliert werden, und zweitens können wir das Aussehen aller grafischen Elemente vom Button über die Schriftart bis zum Hintergrund selbst festlegen (Siehe Kapitel „Design“, Softwaredesign). Ein wesentlicher Nachteil soll aber nicht verschwiegen werden, diese Freiheit macht es nämlich auch notwendig, die Funktionalität aller Bedienelemente selbst zu implementieren, was einen großen Teil der Software und ihrer Entstehung ausmacht. Auf der anderen Seite widerum kann ich so das Verhalten der Skriptsprache ganz nach meinen Vorstellungen umsetzen.
Kern des Programms ist, jedenfalls logisch nach seiner Funktion betrachtet, die Methodik zur Einteilung der WAVE-Datei in Stellen mit und Stellen ohne Signal. Zuvor wird die Datei auf niedrigster Ebene direkt über die Bytes ausgelesen und dann jeder Kanal einzeln im Arbeitsspeicher mit den notwendigen Rahmendaten abgelegt. Eine WAVE-Datei ist eine Sammlung von Messungen eines analogen Audiosignals, auch Samples genannt. Bei CD-Qualität werden 44100 solcher Messungen pro Sekunde durchgeführt. Das Programm muss außerdem erfahren, wieviel Speicher pro Sample genutzt werden soll, bei CD-Qualität entspricht dies 2 Bytes (16 bit). Mit der Gesamtanzahl der Samples besitzt man dann alle notwendigen Rahmendaten zur Auswertung. Diese läuft so ab, dass das Programm die Audiodaten Sample für Sample abtastet, um festzustellen, ob der Wert über oder unterhalb der über `setlim` festgelegten Toleranzgrenze liegt. Der zweite Parameter von `setlim` gibt einen Zeitabschnitt an. Wird innerhalb dieses Intervalls kein Sample gefunden, das über der Toleranzgrenze liegt, so gilt der folgende Abschnitt als signallos. Ausgehend davon müssen mindestens so viele Samples innerhalb eines gleichlangen Zeitabschnitts liegen, wie vom dritten Parameter von `setlim` prozentual angegeben wird, um wieder in einen Bereich mit Signal zu kommen. Dort wird der Prozess dann wiederholt, bis schließlich das Ende der Audiodaten erreicht ist. Aus Gründen der Effektivität wird diese Prozedur nicht bei jeder Abfrage eines Signalwechsels durchgeführt, sondern nur einmal beim Aufrufen der dritten Programmseite, was bei größeren Dateien (und langsameren Rechnern) das kurze Stocken beim Klick auf den „Weiter-“Button zur Folge hat.

Design

Einleitung

Der Begriff *„Design"* steht für den Entwurf und die endgültige Formgebung einer Idee oder eines geistigen Bildes. Für uns ist es wichtig, ein solches Kapitel in diese Dokumentation einzubringen, da unser Projekt in seiner Gesamtheit ein Produkt darstellt, das es in dieser Form noch nie gab und wir deshalb bei der Gestaltung sämtlicher Elemente völlig freie Hand hatten, was wir auch ausgenutzt haben. Somit wurde das *Design* ein wichtiger Aspekt des Projektes und die unterschiedlichen Teilgebiete, in denen *Design* betrieben wurde, werden hier erläutert.

Softwaredesign

Jeder von uns benutzt fast täglich einen PC und damit auch Software - und wenn es nur das installierte Betriebssystem ist. Jeder von uns ist es gewohnt, dass er nach dem hochfahren auf den „Start"-Button klickt, woraufhin sich das Startmenü öffnet, Programme gestartet werden können, etc. Doch wer macht sich schon einmal Gedanken darüber, wie das alles funktioniert? Hinter jeder Software mit grafischer Bedienoberfläche steckt natürlich erstens ein Programmierer, der das Programm selbst schreibt und zweitens auch ein Designer, der sich Gedanken darüber macht, wie das Programm auf dem Bildschirm des Benutzers aussehen soll. In unserem Falle heißt die Software XESCOE, der Programmierer Benedikt und der Designer bin ich.
Meine Aufgabe war es also, unsere Messwerterfassungs- und -verarbeitungssuite zu visualisieren. Hierzu mussten wir im Team erst die grundlegende Struktur des Programmes festlegen: Es ist wie ein sogenannter Wizard aufgebaut, führt den User also von Schritt zu Schritt. In unserem Falle sind es deren drei, jeder auf einer eigenen Seite innerhalb des Programmes, die ich später alle noch einzeln erläutern werde. Was allerdings alle drei Seiten gemeinsam haben, sind Hintergrund und Exit- Button:

Hintergrund:
Programme mit einfachem weißem, grauem, oder Ähnlichem Hintergrund sehen schlichtweg langweilig aus. XESCOE soll natürlich nicht so wirken, weshalb ich einen aufwändigeren Hintergrund erstellen musste. Es würde zu weit führen, auf alle Möglichkeiten einzugehen, die ich ausprobiert habe, deswegen sei die endgültige Textur, nämlich Brushed Metal hervorgehoben: Brushed Metal, also gebürstetes Metal oder Stahl ist ein Überbegriff für alles, was in irgendeiner Weise so aussieht, als ob man mit einer Drahtbürste an etwas Silbernes gegangen sei. Das Ganze hat den Vorteil, dass es relativ elegant und vor allem neutral aussieht. Außerdem kann man es so gestalten, dass es nicht zu dunkel wirkt. In Kurzfassung wird eine solche Textur über ein Schwarzes Bild, auf das man erst einen Wolkenfilter anwendet und dann mittels Störungs- und Weichzeichnerfiltern eine gebürstete Struktur aufträgt, erstellt. Für meinen Hintergrund habe ich diese Grundlage ein wenig abgeändert und Filter teilweise weggelassen bzw. verändert, um das Ganze neutraler zu halten. Das Ergebnis ist in Abb. 1 zu sehen. Auf jeder Seite ist oben der Name des Programms zu lesen, außerdem sind noch die seitenspezifischen Header (Überschriften) eingearbeitet.

Exit-Button:
Wie der Name schon sagt, beendet dieser Button das Programm. Er sitzt in der rechten oberen Ecke des Bildschirms und hat, wie alle Buttons in XESCOE zwei Phasen: Ungeklickt (Abb. 2) und Geklickt (Abb.3)

Abb. 1

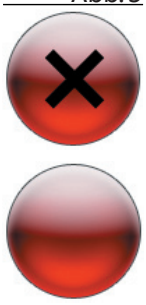


Abb. 2

Abb. 3

Die oben schon kurz angerissenen drei Seiten, in die sich die Software gliedert, sind folgende:

1. Versuchsauswahl:
Hier kann der User auswählen, welchen Versuch er durchführen will und bekommt die Versuche auch kurz im Überblick vorgestellt. Wesentliche Elemente dieser Seite sind folgende:

- *2 Fenster:* Eines in dem die verfügbaren Versuche angezeigt werden und eines, in dem der selektierte Versuch erläutert wird. Elementare Anforderungen an ein solches Fenster sind die Abgrenzung eines bestimmten Bereiches vom Hintergrund durch das gesamte Fenster, sowie eine interne Unterscheidung zwischen Überschrift und Inhalt des Fensters.
- *Markierungsbalken:* Dieser Balken zeigt an, welcher Versuch gerade selektiert ist.
- *Versuch starten-Button:* Hiermit wird der gewählte Versuch gestartet und der User auf Seite 2 weitergeleitet.

Das gesamte Design ist in Abb. 4 zu sehen.

2. Wave-Import:
Auf dieser Seite muss der User nun die Wave-Audiodatei, die er für den Versuch aufgezeichnet hat, in das Programm importieren, indem er den Dateipfad angibt, unter dem die Datei zu finden ist. Ich habe diese Seite in zwei Teile aufgesplittet: Zum ersten die rechte Seite, auf der der User nichts verändern oder anklicken kann. Sie enthält eine genaue Erklärung, was auf dieser Seite getan werden muss und auf welche Weise ein Dateipfad angegeben wird. Zweitens die linke Seite, auf der der Dateipfad im Texteingabefeld angegeben wird. Grafische Elemente sind folgende:

- *Texteingabefeld:* In dieses Feld wird der Dateipfad eingegeben. Um es auch optisch als auszufüllendes Feld zu akzentuieren, muss es sich deutlicher vom Hintergrund abgrenzen als beispielsweise ein normales Fenster, sollte aber trotzdem ins allgemeine Layout passen. Ich habe dies durch eine weiße Basis, die durch einen Farbverlauf mit bereits in anderen Elementen verwendeten Farbwerten, aufgelockert wird, gelöst.
- *Vor-/ Zurück-Buttons:* Bringen den User entweder nach Angabe des Pfades einen Schritt weiter, also auf die nächste Seite, oder zurück auf Seite 1, falls doch ein anderer Versuch ausgewählt werden soll. Optisch sind sie recht schlicht gehalten, passend auch zum Design des Buttons auf Seite 1.

Das gesamte Design ist in Abb.5 zu sehen.

3.Versuchsauswertung:
Dies ist die einzige Seite, die layouttechnisch von mir nicht genau festgelegt wird, da sie versuchsspezifisch ist und somit über die Skriptsprache von demjenigen, der den betreffenden Versuch in die Software einbaut, festgelegt wird. Um alle möglichen Fälle abzudecken, sind folgende grafische Elemente notwendig:

- *Texteingabefeld:* Hier werden in dieses Feld die o.g. Werte eingegeben, das Feld selbst ist das gleiche wie auf Seite 2.
- *Auswertungs-Button:* Dieser Button startet die Versuchsauswertung.
- *Blank-Button:* Gleiches Design wie der Auswertungs-Button, nur ohne Beschriftung. So kann der skriptende User eine eigene Beschriftung festlegen.
- *1 Fenster:* In diesem Fenster, welches das gleiche ist, wie auf Seite 1, werden Erklärungen ausgegeben.
- *Scrollbuttons:* Um innerhalb des Fensters scrollen zu können, braucht es noch zwei Up- und Down-Scrollbuttons. Diese sind klein und schlicht und haben als einzige Buttons in XESCOE keine zwei Phasen, da sie optisch nicht so akzentuiert werden müssen wie die größeren Buttons
- *Zurück-Button:* Um auf die vorherige Seite zurückzugelangen.
- *Rahmen:* Ein Rahmen, also eine optische Abgrenzung für die Visualisierung der Audiodatei. Er ist sehr dezent, da die Visualisierung eigentlich nur ein netter Zusatz ist und das Hauptaugenmerk auf den Ergebnissen und Erklärungen liegen sollte.

Ein mögliches Design ist in Abb. 6 zu sehen.

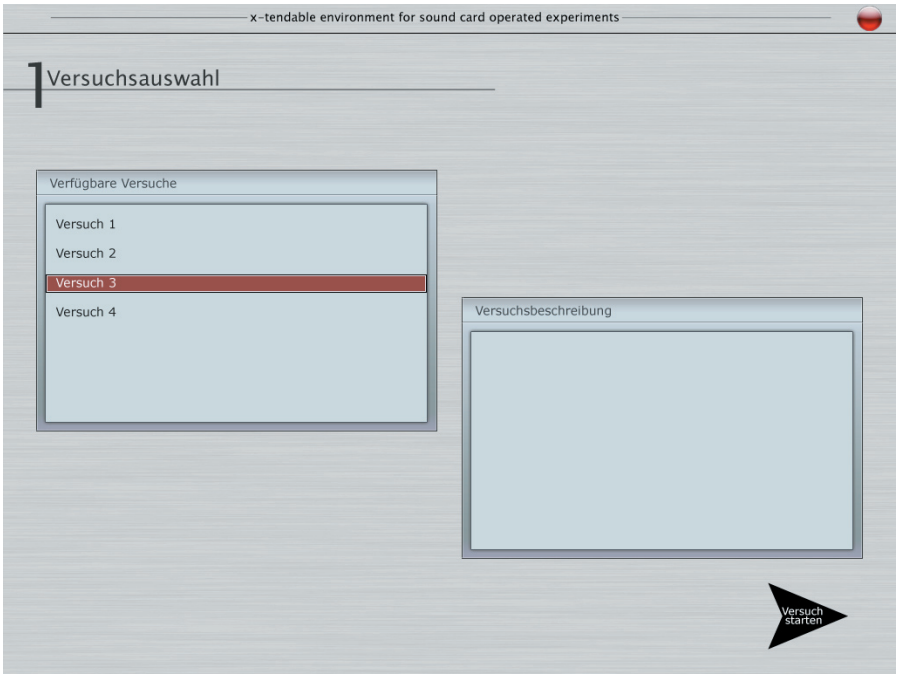


Abb. 4

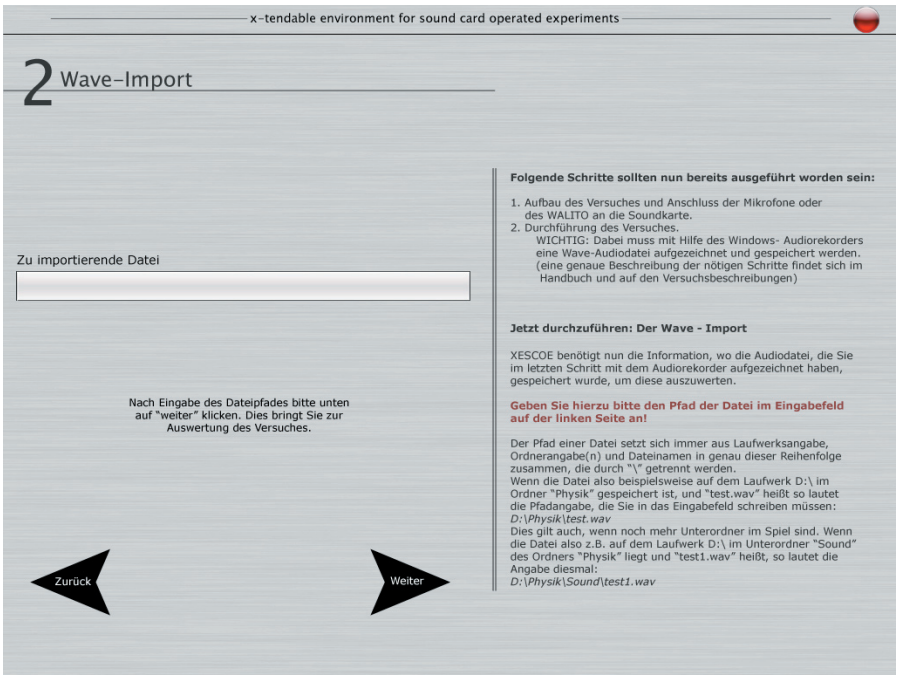


Abb. 5

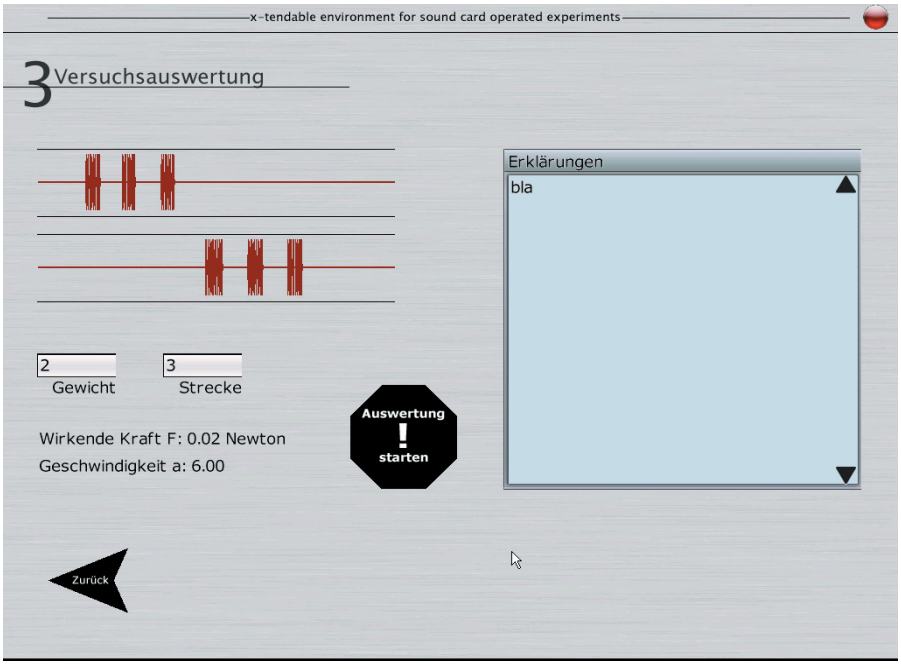


Abb. 6

WALITO

Auch hinter dem WALITO stecken einige Designgedanken. So soll ein solches Gerät natürlich möglichst übersichtlich sein. Deshalb habe ich die Anschlüsse für die beiden Lichtschranken sowie deren Statusanzeigen auf dem Deckels des Gerätes nach außen geführt, farblich getrennt nach Lichtschranke 1 und 2. Die Stromversorgung und der Anschluss an die Soundkarte befinden sich an einer Stirnseite. Eine anschauliche Beschriftung ist per bedruckter Klebefolie auf dem Deckel angebracht (Abb. 7)

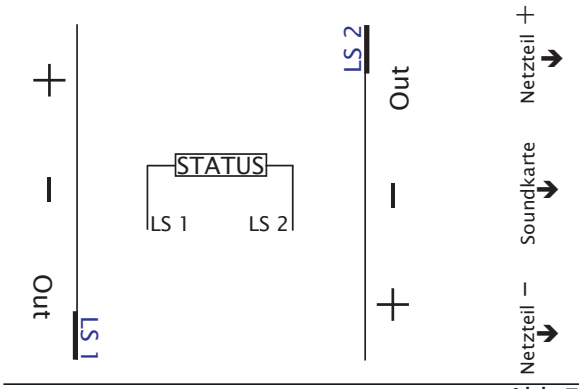


Abb. 7

Handbuch und Dokumentation

Die vorliegende Dokumentation ist als Dokument zur Abgabe und zur Benotung schlicht und übersichtlich gehalten. Dies habe ich umgesetzt, indem ich konsequent bei einer Schriftfamilie (mit Ausnahme des Skriptcodes) und möglichst wenigen verschiedenen Schriftgraden geblieben bin. Bilder sind dem Text an Priorität untergeordnet. Die eigentlichen Inhalte (abgesehen von meinen eigenen natürlich) verfassten Benedikt und Niklas als reinen Text, woraufhin es meine Aufgabe war, diesen ins Design der Dokumentation einzupassen.
Das Handbuch orientiert sich an der Dokumentation, ist aber etwas weniger schlicht gehalten, da es sich hierbei um keine wissenschaftliche Arbeit handelt. Auch das Format, nämlich DIN A5 grenzt es von einer „normalen" Facharbeit o.Ä. ab. Mit dem X-Logo als Hauptelement ist garantiert, dass man auf den ersten Blick bemerkt, dass beide Dokumente zum gleichen Projekt gehören. Abb. 8 zeigt das

Präsentation

Zum Zeitpunkt der Drucklegung dieser Dokumentation ist die Präsentation noch unvollständig, deswegen kann ich nur die bisherigen Gedanken dazu widergeben:
- Schlank (wissenschaftlich! Eine Schriftfamilie, wenige Schriftgrade)
- Professionell (nur unterstützend, weniger Inhalte als bei Unterrichtspräsentationen)
- Farbschema wie das Logo (weißer Hintergrund, schwarze Schrift, Farben wenn dann in Blautönen)
- Erste Seite nur mit Logo (um sie bei Begrüßung + Einleitung hinter uns zu projizieren)
- Zwei Beamer (Arbeiten mit Kamera + Präsentation, evtl. auch verknüpft)
- Vor dem Laienpublikum ausführlicher Einleiten / Erklären, evtl. auch mit einer zusätzlichen Präsentationsseite (grafisch darstellen)

CD-ROM

Die CD-ROM, auf der sich Dokumentation, Handbuch, sowie die eigentliche Software befinden, ist dem Handbuch in einem Extrakapitel beigefügt, so dass der Endbenutzer sie gleichzeitig mit dem Handbuch griffbereit hat. Da beides so eng verbunden ist, ist eine Extrabeschriftung der CD unnötig, deshalb hat sie nur ein einziges Designelement, nämlich das X-Logo.

X-Logo

Das Logo (Abb. 10) unserer Suite basiert auf dem „X" von XESCOE, von dem ausgehend stilisierte Schallwellen in den Raum streben. Auf der linken Seite ist der Eigenname „XESCOE" an die Schallwellen angekoppelt. Das Logo als Ganzes soll seriös aber dennoch nicht bieder wirken, um weder Schüler noch Lehrer abzuschrecken.

Abb. 8

Abb. 9

Abb. 10

Versuche

g-Versuch

Aufbau:
In diesem und dem folgenden Abschnitt wird erklärt wie der g-Versuch aufgebaut und durchgeführt wird. Um den Versuch durchzuführen bracht man neben dem eigentlichen Versuch (Batteriebox + Wäscheklammer) noch folgendes:
- Einen Zollstock
- Verschiedene Stativteile: 1 Stange, 1 Tischhalterung, 1 Schraubklemme, 1 Kreuzverbindung
- Ein Mikrofon (z.B. die aus dem Schallmessungsversuch (ebenfalls mit der Soundkarte)
- Einen Computer mit Line In Anschluss, auf dem ein Programm zum aufnehmen von Wave-Dateien (z.B. Windows Audiorekorder) und XESCOE installiert ist.

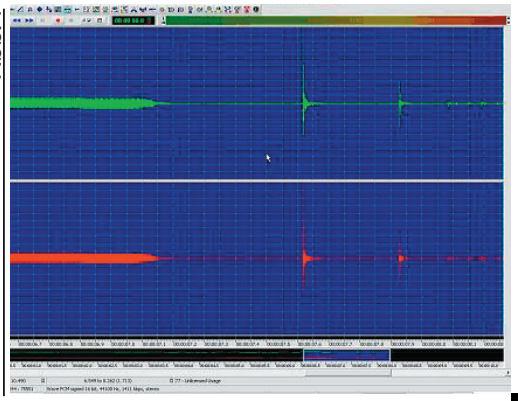
Hat man diese Teile zusammen wird nun der Versuch aufgebaut. Dazu befestigt man mit der Tischhalterung die Stativstange und mit der Kreuzverbindung die Klemme an der Stativstange. Jetzt klemmt man die Wäscheklammer mit dem daran befestigten Holzstück in die Klemme und fixiert so die Wäscheklammer. Als nächstes muss das Mikrofon an den Line-In Anschluss der Soundkarte des Versuchs-PCs angeschlossen werden. Dann wird der Computer gestartet und das gewählte Aufnahme-Programm geöffnet werden. (beim Windows Audiorekorder: Start – Programme – Zubehör – Unterhaltungsmedien – Audiorekorder) Das Mikrofon legt man zusammen mit der Summerbox neben das Stativ, also neben die Aufschlagstelle der Kugel. Klemmt man die beigelegte Kugel in die Wäscheklammer, müsste aus der Batteriebox ein Summton ertönen. Der Versuch ist jetzt einsatzbereit.

Durchführung:
Hat man den Versuch aufgebaut, folgt die Durchführung. Die ist ganz einfach man steckt die Kugel in die Wäscheklammer, aktiviert das Mikrofon und lässt dann die Kugel fallen. Danach stoppt man die Aufnahme und speichert sie als Wavedatei. Dabei sollte man den Dateipfad entweder selbst wählen, oder ihn sich zumindest merken. Denn als nächstes sollte das Programm „XESCOE“ zur Versuchsauswertung gestartet werden, und dieses braucht den Dateipfad (und natürlich auch den Namen), um die Datei zu finden und den Versuch auszuwerten (s. auch die Erläuterungen im Programm). In XESCOE lässt sich der Versuch dann auswerten und die g-Konstante berechnen (wie das funktioniert steht im Teil „physikalischer Hintergrund“).
Hier einige Beispiel-Messergebnisse in guter Übereinstimmung mit allen anderen Werten:

Versuch nr.	*Gemessener Weg*	*Gemessene Zeit*	*Berechneter g-Wert*
1	59,9 cm	0,348 s	10,24
2	52,2 cm	0,338 s	9,138
3	79 cm	0,41 s	9,399

Funktionsweise:
Der Versuch funktioniert, indem bei gegebenem Wert s (=Strecke), den man selbst gewählt hat ein freier Fall durchgeführt und die Fallzeit t gemessen wird. Dazu wird in der Wäscheklammer bei eingehängter Kugel ein Stromkreis geschlossen, der einen Summer betreibt. Wird die Wäscheklammer geöffnet, beginnt die Kugel zu fallen und auch der Stromkreis öffnet sich, weshalb der Summer abgeschaltet wird. Fällt die Kugel auf den Boden entsteht wiederum ein Geräusch. Nimmt man diesen Vorgang per Mikrofon und einem geeignetem Programm auf (Audiorecorder) und wertet die aufgenommene Datei mit XESCOE aus, so ist die Fallzeit in der visuellen Darstellung deutlich als Lücke zwischen Summersignal und Aufschlag zu erkennen.

Abb. 1



Physikalischer Hintergrund:
Als erstes ist zu erwähnen, dass die Bezeichnung „freier Fall" für diesen Versuch nicht ganz korrekt ist. Eigentlich handelt es sich um einen „Fall im lufterfüllten Raum", dessen Berechnung und Auswertung ungleich komplizierter ist. Allerdings ist beim Fall einer Eisenkugel der Luftwiderstand bei hoher Gewichtskraft G relativ gering und aufgrund des geringen Fallweges vernachlässigbar. (Aber auch hier würde sich bei lang genug gewähltem Fallweg eine konstante Endgeschwindigkeit einstellen, da der Luftwiderstand exponentiell wächst.)
Der freie Fall lässt sich am besten über die Kräftegesetze nach Newton erklären. So wirkt auf den fallenden Körper – als einzige Kraft – die konstante Gewichtskraft $G = m*g$. g ist der Ortsfaktor, der bei unserem Versuch gemessen werden soll. (der tabellarische Wert für unsere Breiten ist $g = 9{,}81\ m/s^2$)
Der Ortsfaktor g entspricht im freien Fall der Beschleunigung a, den Beweis dafür liefert Newtons Grundgesetz, wobei F durch G ersetzt werden kann (da G die einzig wirkende Kraft F ist):

$a = F/m = G/m = (g*m)/m = g$

Auch die Einheiten entsprechen sich. Ein anschaulicher Beweis dafür ist die Trägheit: Ein Körper hat die zehnfache Masse und erfährt somit auch eine zehnmal so starke Gewichtskraft wie ein beliebiger anderer, müsste also zehnmal so stark beschleunigt werden. Allerdings hat er aufgrund der höheren Masse auch proportional höhere Trägheit. Diese wirkt aber der Beschleunigung genau entgegen, und gleicht so die höhere Masse aus. Also erfährt ein Körper mit viel Masse genauso viel Beschleunigung wie ein Körper mit wenig Masse.
Beim freien Fall handelt es sich also um einen Sonderfall der beschleunigten Bewegung. Somit gelten also auch die Gesetze der beschleunigten Bewegung. Für die Berechnung der g-Konstanten ist dafür besonders ein Gesetz wichtig: $s = ½\ a\ t^2$ (aus der Berechnung der Fläche unter einem v-t Diagramm) Diese Formel lässt sich nach a (bzw. g, s.o.) umstellen und heißt somit: $g = 2\ (s/t^2)$. In dieser Formel können nun der gemessene Weg und die gemessene Zeit eingesetzt werden (was in XESCOE automatisch passiert).

Schallgeschwindigkeitsmessung

Aufbau:
In diesem und dem nächsten Teil wird erklärt wie der Schallmessungsversuch aufgebaut und durchgeführt wird. Zum Aufbau braucht man neben dem eigentlichen Versuch (zwei Mikrofone im Stativ mit Batteriebox und Anschluss an die Soundkarte) noch folgendes:
- Eine Messleiste aus der Optik sowie zwei dazugehörige Wägelchen mit jeweils zwei Schrauben zum fixieren.
- Eine Schallquelle (dem Versuch liegen „Knackenten" bei)
- Einen Computer mit Line-In Anschluss, auf dem ein Programm zum Aufnehmen von Wave-Dateien (z.B. Windows Audiorekorder) und XESCOE installiert ist.
- Ein Lineal, Geodreieck oder etwas Ähnliches

Die Mikrofone werden nun in die Wägelchen gesteckt und so gedreht, dass die Membran des Mikrofons bündig mit der einen Seite des Wägelchens abschließt (dazu dient das Lineal) und dann mit der Stellschraube fixiert. Die Wägelchen kommen auf die Messleiste (die Mikrofone sollten in die gleiche Richtung zeigen) und der gewünschte Abstand wird eingestellt, was dank der vorherigen Einstellung kein Problem mehr sein sollte. Als letztes muss das Mikrofon angeschlossen werden, und das Aufnahmeprogramm gestartet werden.

Durchführung:
Als erstes wird die Aufnahme gestartet. Dann wird mit dem gewählten Instrument ein Geräusch verursacht. Die Geräuschquelle muss sich dabei in gerader Linie zu den Mikrofonen befinden um ein optimales Ergebnis zu erhalten. Nachdem die Mikrofone das Geräusch aufgenommen haben kann die Aufnahme gestoppt und als Wavedatei gespeichert werden. Dabei ist es wichtig darauf zu achten, dass sich das Gerät am Line-In Eingang befindet und der Audiorecorder auch am Line-In Eingang aufzeichnet. Wie das genau geht: Im Audiorecorder unter Bearbeiten - Audioeigenschaften - Bereich Soundaufnahme - Lautstärke können unter den gewünschten Eingang Häkchen gesetzt werden. Außerdem sollte man auf den Namen und den Pfad der Datei achten, da diese für XESCOE, das Versuchsauswertungsprogramm benötigt werden. In XESCOE kann dann die Schallgeschwindigkeit berechnet werden (s, physikalischer Hintergrund).

Funktionsweise:
Bei diesem Versuch werden zur Schallgeschwindigkeitsbestimmung zwei Mikrofone benutzt, die in unterschiedlichen Abstand zu einer Schallquelle stehen. Dieser Abstand wird bei dem Versuch selbst eingestellt. Diese Mikrofone werden an einen Line–In Anschluss angeschlossen. Wird der Versuch durchgeführt, empfängt das der Schallquelle näher stehende Mikrofon zuerst das Signal, das andere Mikrofon dagegen erst später. Da das Ganze aber in Stereo, also beide Mikrofone getrennt voneinander aufgezeichnet wird, kann man bei der Auswertung ziemlich leicht die Zeit zwischen dem Signalbeginn von Mikrofon 1 und dem Signalbeginn von Mikrofon 2 messen.
Als Signalquelle eignen sich logischerweise am Besten solche Signale die einen plötzlichen Beginn haben, was eine einfachere Messung ermöglicht.
Abb. 1 zeigt eine visualisierte Wavedatei, Abb. 2 dieselbe Datei, nur vergrößert.

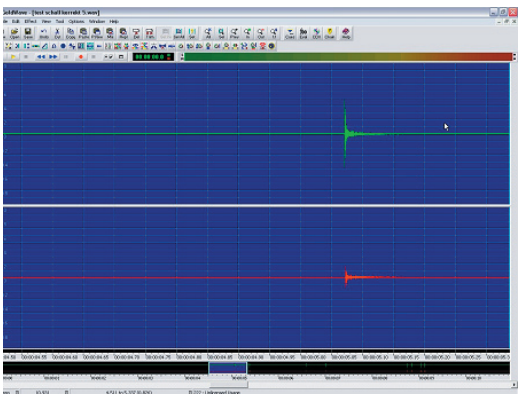


Abb. 1

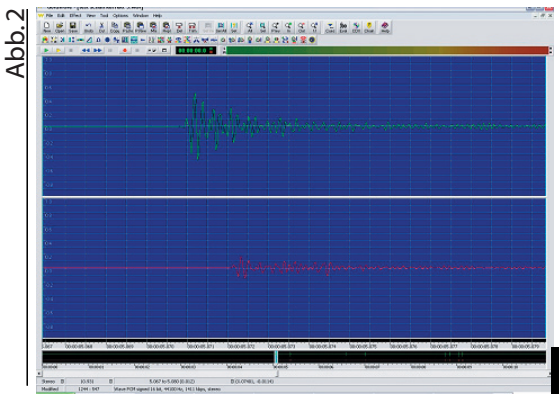


Abb. 2

Physikalischer Hintergrund:
Dieser Versuch misst die Geschwindigkeit des Schalls in der Luft. Bei Schall handelt es sich schließlich um nichts anderes als eine Schwingung, die sich durch jeden beliebigen Körper und natürlich auch durch Luft ausbreiten kann. Somit erfordert die Weiterleitung dieser Schwingung eine gewisse Zeit, und genau diese Zeit wird mit dem Versuch gemessen. Je nach Material breitet sich der Schall unterschiedlich schnell aus (so zum Beispiel in Wasser viel schneller als in Luft). Aus der benötigten Zeit t kann dann mit der Formel v = s/t die Geschwindigkeit ausgerechnet werden. Hier sind einige Beispielmessungen aufgeführt. Der tabellarische Wert beträgt 340 m/s, man sieht also, dass der Versuch ziemlich präzise misst, denn diese Werte stimmen auch mit allen anderen gemessenen gut überein.

Versuch nr.	*Weg s in m*	*Zeit t in s*	*Ergebnis V in m/s*
1	0,8	0,00248	322,58
2	1,0	0,003	333,33
2	0,4	0,0012	333,33

WALITO - das Gerät

Wie schon gesagt, ist der moderne Computer mit seiner präzisen internen Zeitintervallmessung ein optimales Messgerät. Bei etwas komplizierteren Versuchen wird nun aber der Wunsch nach anspruchsvolleren Inputgeräten als einem oder zwei Mikrofonen aufkommen - das Gerät der Wahl wäre hier natürlich eine Lichtschranke, da sie die Tür zu diversen neuen Versuchen öffnet. Im Prinzip ist das ein schöner Gedanke, nur stellt sich sehr bald ein elementares Problem: Eine Soundkarte ist ein Gerät zur Erfassung von - natürlich - Sound, also Wechselspannungssignalen bis ca. 20kHz. Eine Lichtschranke dagegen gibt eine quasistationäre Gleichspannung aus, die eine Soundkarte im Prinzip nicht verarbeiten kann.
Daraus ergibt sich die Notwendigkeit, ein Gerät zu bauen, welches in der Lage ist, die Outputsignale einer Lichtschranke zu verarbeiten und soundkartenverträgliche Wechselspannungssignale auszugeben. OStR. Klaus-Dieter Grüninger hat ein solches Gerät, den WALITO (Wandler Licht-Ton) konzipiert und eine Bauanleitung dazu auf dem Landesbildungsserver Baden-Württemberg veröffentlicht. Ich selbst hätte dank mangelnder Elektronikkentnisse ohne Anleitung niemals etwas entsprechendes bauen können. Herr Grüninger, mit dem ich im Laufe des Baus auch per E-Mail in Kontakt trat, war sehr hilfsbereit, beantwortete alle meine Laienfragen ausführlich und ich halte es für fraglich, ob der Nachbau des WALITO ohne seine Hilfe gelungen wäre. Deshalb möchte ich ihm an dieser Stelle im Namen unserer ganzen Projektgruppe unseren Dank aussprechen. Sämtliche folgenden Erläuterungen zur Funktionsweise des Gerätes sind inhaltlich von ihm übernommen.

Prinzipiell arbeitet der WALITO nach einer „Modulationsmethode". Sprich: Er erzeugt immer genau dann, wenn die Lichtschranke verdunkelt ist, über einen Generator ein Wechselspannungssignal, das an die Soundkarte geht. Es können maximal zwei Lichtschranken an den WALITO angeschlossen werden, welche dann über das Gerät auch gleich mit Strom versorgt werden. Der grundlegende Aufbau des Geräts ist: Lichtschranke - Invertierlogik - Timer - Soundkarte. Die Invertierlogik sorgt dafür, dass sowohl Lichtschranken, die bei Unterbrechung 0V, als auch solche, die dann 5V ausgeben, angeschlossen werden können. Beide Arten von Lichtschranken sind auf dem Markt erhältlich. Der Timer schwingt (erzeugt das gewünschte Wechselspannungssignal), sobald an seinem Eingang 5V anliegen, sind es 0V, ist er in Ruhe. Somit kann eine Soundkarte über einen ganz normalen Klinkenstecker auf beiden Kanälen (eine Lichtschranke pro Kanal) angesteuert werden. Sowohl die Invertierlogik als auch die beiden Timer sind über jeweils einen bestimmten IC realisiert.

Der Bau des WALITO war eine recht knifflige Sache, was mir auch von Anfang an klar war. Trotzdem habe ich mich dazu entschieden, das Gerät zu bauen, und das aus drei Gründen: Erstens sollte der WALITO eine Art Prestigeobjekt sein, also unsere Fähigkeit demonstieren, auch etwas kompliziertere Versuchsaufbauten zu realisieren, zweitens hat das Gerät wirklich einen Sinn für den Einsatz im Unterricht, besonders hinsichtlich der Möglichkeit, neue Versuche damit zu konzipieren und mithilfe von XESCOE auszuwerten, und drittens war der Bau schlichtweg eine Herausforderung und beinhaltete viele für mich neue Arbeitsschritte wie zum Beispiel das Ätzen von Platinen oder das Löten von komplizierteren elektronischen Schaltungen. Dies alles sind Dinge, die man im normalen Gymnasialunterricht nicht einmal annähernd lernt und deshalb war im Rückblick meiner Meinung nach die doch sehr lange Zeit, die der von diversen Problemem begleitete Bau des Gerätes gebraucht hat, keine vergeudete.

WALITO - Abhängigkeit von Kraft und Beschleunigung

Der WALITO an sich ist schön und gut, aber da unser Projekt ja auf die tatsächliche Verwendung der Endprodukte im Unterricht abzielt, brauchen wir auch noch entsprechende Versuche. Hierfür habe ich in Absprache mit Herrn Schich als erstes die Abhängigkeit von Kraft und Beschleunigung ausgesucht, da dieser Versuch nicht zu kompliziert ist. An zweiter Stelle unserer Liste stand der unelastische Stoß, doch dieser muss aus Zeitgründen leider wegfallen. Ich denke jedoch, dass ein Versuch genügt, um die Fähigkeiten des WALITO zu demonstrieren und hoffe, dass das Gerät in Verbindung mit der Software XESCOE unter den Physiklehrern Anklang findet und vielleicht einige bereit sein werden, neue Versuche mit dem WALITO zu konzipieren. Doch nun zum Versuch:

Zu S.30/31 Dorn-Bader: Physik 11

Fragestellung: 1. Wie hängt die nötige Kraft *F* bei ein und demselben beschleunigten Fahrzeug von der gewünschten Beschleunigung *a* ab?

2. Wie hängt die Kraft *F* vom Fahrzeug und seiner Beladung ab, wenn sich *a* nicht ändern soll?

Zu Frage 1: Versuch 1

Benötigte Teile:
- Schiene (Mechanik)
- Wagen + Stange zum Aufstecken + Gewichte
- Abrollvorrichtung + Schnur
- 2 Lichtschranken
- 2 Stativplatten + Aufbauteile

a) Auf einer Schiene wird ein Wagen festgehalten, an dem eine beliebige Anzahl von Wägestücken der Masse *m* zieht. Auf dem Wagen wird die entsprechende Stange befestigt, die dazu dient, Wägestücke aufzunehmen und die Lichtschranken zu unterbrechen. Außerdem sind mithilfe von Stativen über der Schiene in beliebigem (nicht zu kleinem) Abstand *s* 2 Lichtschranken angebracht (Siehe Abb. 1). Die Lichtschranken werden über den WALITO an die Soundkarte angeschlossen und eine entsprechende Audiodatei aufgezeichnet (Siehe Kapitel „Software", Funktion und Aufbau). In XESCOE muss nun nach deren Import noch das beschleunigende Gewicht in Gramm (wird intern in Newton umgerechnet) und die Strecke s zwischen den Lichtschranken in cm angegeben werden, anschließend kann der Versuch ausgewertet werden und XESCOE liefert Ergebnisse und Erklärungen.

Abb. 1

b) Nun soll die wirkende Kraft geändert werden, das Gesamtgewicht aber konstant bleiben. Dazu wird eines der Wägestücke weggenommen und auf die Stange auf dem Wagen aufgesteckt (Abb. 2). So wird es weiterhin mitbeschleunigt ohne dass seine Gewichtskraft noch mitzieht. Anschließend erfolgt widerum die Auswertung in XESCOE. Man kann diese Messung natürlich beliebig oft mit verschiedenen Gewichten und Strecken durchführen.

Abb. 2

Beispiele:

F in N:	0,08	0,06	0,04	0,02
a in m/s:	0,392	0,294	0,196	0,098

Wir sehen also: Bei gleich bleibender Masse ist bei n-facher Kraft die Beschleunigung ebenfalls n-mal so groß. Damit ist die Beschleunigung, die ein bestimmter Körper erfährt, der an ihm angreifenden Kraft proportional, also **F ~ a**.

Intern:
Die Audiodatei dieses Versuches ist recht einfach aufgebaut: Ruhezustand - Signal - Ruhezustand - Signal. Im ersten Ruhezustand steht der Wagen in Ausgangsposition. Der Versuch wird gestartet, der Wagen fährt los und es folgt die Unterbrechung der ersten Lichtschranke durch die Stange auf dem Wagen (Signal). Der folgende Ruhezustand in der Audiodatei bedeutet, dass der Wagen zwischen LS1 und LS2 unterwegs ist. Beim zweiten Signal hat er LS2 erreicht. Damit ist eine Zeit *t* bekannt, die der Wagen für die Strecke *s* benötigt. Mithilfe des t-s-Gesetzes $s=1/2at^2$ errechnet XESCOE daraus die Beschleunigung *a* und gibt diese aus.

Zu Frage 2: Versuch 2

Benötigte Teile:
Wie bei Versuch 1, nur kommen ein zusätzlicher Wagen + Gewichte hinzu.

Frage:
Wie hängt die Kraft *F* vom Fahrzeug und seiner Beladung ab, wenn sich *a* nicht ändern soll?

Diese Frage stellt sich deshalb, da z.B. Raketen auch im schwerelosen Raum zum Beschleunigen Kräfte benötigen, die von ihrer Beladung abhängen, obwohl diese dort nichts wiegt. Die Raketen sind also auch im Weltall träge.
Deshalb wollen wir mit folgendem Versuch überprüfen, ob ein Körper doppelter Masse für die gleiche Beschleunigung auch die doppelte Kraft benötigt, also doppelt so träge ist.

Versuch:
Der Aufbau ist nahezu der gleiche wie bei Versuch 1b), nur wird diesmal noch ein zweiter Wagen gleicher Masse an den ersten angehängt und gleichzeitig die Gewichte am Ende der Schnur verdoppelt (Siehe Abb. 3) Nach der Auswertung in XESCOE sehen wir, dass die Beschleunigung gleich bleibt wie in Versuch 1b).

Damit kommt man zu dem Schluss, dass die Kraft *F*, die man für eine bestimmte Beschleunigung braucht zu der Masse *m* des zu beschleunigenden Körpers proportional ist. Also ist **F~m**.

Intern:
Wie bei Versuch 1.

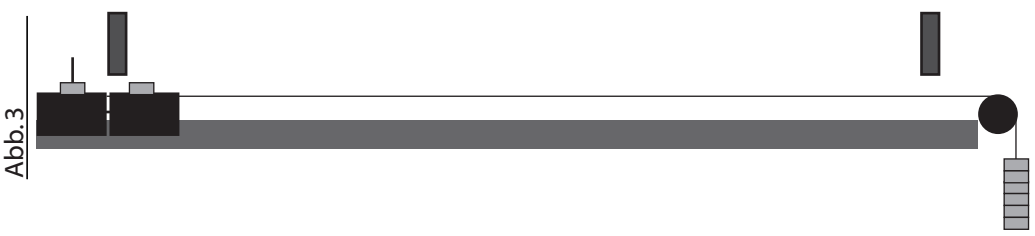


Abb. 3