

DIPLOMARBEIT

Konzeption und Realisierung einer multimedialen Präsentationsplattform für Themen der Computertechnik

Autor: Thomas Morawe

Studiengang: **Multimediatechnik**

Seminargruppe: **MK03w1**

Erstprüfer: **Prof. Dr.-Ing. Wilfried Schmalwasser**

Zweitprüfer: **Prof. Dr.-Ing. Frank Zimmer**

Mittweida, August 2009

Bibliografische Angaben

Morawe, Thomas

Konzeption und Realisierung einer multimedialen Präsentationsplattform für Themen der Computertechnik

87 Seiten, 31 Abbildungen, 8 Tabellen, 1 Anlage mit Quelltexten

Hochschule Mittweida (FH), Fachbereich Informationstechnik & Elektrotechnik

Diplomarbeit, 2009

Referat:

Diese Diplomarbeit befasst sich mit der Konzeption und der Realisierung von Struktur und Design einer Präsentationsplattform und ausgewählten Beispielanimationen für das Fach Computertechnik. Es werden Lösungsmöglichkeiten und Werkzeuge zur Erstellung aufgezeigt. Im Anhang befinden sich die ausführlichen Scripte, die zur Steuerung und Modularität beitragen.

Inhaltsverzeichnis

| | |
|---|----|
| Abkürzungsverzeichnis..... | 7 |
| Begriffserklärung..... | 8 |
| 1 Einleitung..... | 10 |
| 1.1 Motivation..... | 10 |
| 1.2 Kapitelübersicht..... | 11 |
| 2. Grundlagen..... | 12 |
| 2.1 Was ist Animation?..... | 12 |
| 2.2 Mögliche Animationssoftware..... | 12 |
| 2.2.1 Microsoft PowerPoint..... | 13 |
| 2.2.2 Director..... | 15 |
| 2.2.3 Autodesk 3D Studio Max..... | 16 |
| 2.2.4 SMIL..... | 17 |
| 2.2.5 Flash..... | 19 |
| 2.3 Entscheidung..... | 21 |
| 2.4 Hilfsprogramme..... | 23 |
| 3. Konzeption..... | 25 |
| 3.1 Forderungen..... | 25 |
| 3.2 Gedanken zur Gestaltung der Benutzeroberfläche..... | 25 |
| 3.2.1 Struktur..... | 25 |
| 3.2.2 Design..... | 26 |
| 3.2.3 Aufbau..... | 27 |
| 3.2.4 Bildrate der Nutzeroberfläche..... | 27 |
| 3.2.5 Farben..... | 28 |
| 3.4 Vorüberlegungen zu den Animationen..... | 28 |
| 3.4.1 Platzaufteilung..... | 28 |
| 3.4.2 Bildrate der Animationen..... | 29 |
| 3.4.3 Storyboards und interner Aufbau..... | 29 |
| 3.4.3.1 PIO Mode..... | 30 |
| 3.4.3.2 DMA Mode..... | 34 |
| 3.4.4 Farben..... | 37 |
| 4. Realisierung..... | 40 |
| 4.1 Die Benutzeroberfläche..... | 40 |
| 4.1.1 Grafische Erstellung der Benutzeroberfläche..... | 40 |
| 4.1.2 Aufbau der Menü-XML-Datei..... | 40 |
| 4.1.3 Funktionalitäten der Benutzeroberfläche..... | 41 |
| 4.1.3.1 Überblick..... | 41 |
| 4.1.3.2 Einlesen der XML-Daten..... | 44 |
| 4.1.3.3 Navigation..... | 46 |
| 4.1.3.4 Dozentenleiste..... | 54 |
| 4.1.3.5 Filmsteuerung..... | 55 |
| 4.1.3.6 Playlistensteuerung..... | 58 |
| 4.2 Die Animationen..... | 61 |
| 4.2.1 Grafische Erstellung der Animationen..... | 61 |
| 4.2.2 Aufteilung der Gesamtanimation in Teilabschnitte..... | 61 |
| 4.2.3 Verknüpfung der Animationen mit der Benutzeroberfläche..... | 62 |
| 4.2.4 Realisierung einer Bewegungsanimation..... | 62 |
| 4.2.5 Erstellung von scrollendem Text..... | 62 |
| 4.2.6 Bearbeitung der Textteile..... | 66 |

| | |
|--|----|
| 5. Zusammenfassung..... | 67 |
| 5.1 Ergebnisse und aktueller Stand..... | 67 |
| 5.2 Ausblick..... | 68 |
| 6. Das Handbuch..... | 69 |
| 6.1 Allgemeines..... | 69 |
| 6.2 Anlegen einer neuen Vorlage..... | 69 |
| 6.3 Erstellen von wiederverwendbaren Objekten..... | 70 |
| 6.4 Anpassung von Objekten..... | 71 |
| 6.5 Bewegungsanimation und Transparenz..... | 72 |
| 6.6 Scrollfunktion..... | 73 |
| 6.7 Pausen und Schleifen..... | 74 |
| 6.8 Formatierung der XML-Datei..... | 74 |
| 6.9 Änderung der Buttons..... | 75 |
| 6.10 Anlegen der Playlisten..... | 75 |
| 6.11 Abschluss..... | 76 |
| 7. Anhang..... | 77 |
| 8. Literaturverzeichnis..... | 86 |
| 9. Selbstständigkeitserklärung..... | 87 |

Abbildungsverzeichnis

| | |
|---|----|
| Abbildung 1: Lerntypenverteilung nach Vester [2]..... | 10 |
| Abbildung 2: Aufbau einer SMIL-Datei..... | 18 |
| Abbildung 3: Aufbau einer XML-Datei..... | 24 |
| Abbildung 4: Beispiel der Vorlesungsunterlagen..... | 26 |
| Abbildung 5: Storyboard Übersicht PIO Mode..... | 30 |
| Abbildung 6: Storyboard PIO Mode - Detailansicht..... | 30 |
| Abbildung 7: Storyboard PIO Mode - Cache laden..... | 31 |
| Abbildung 8: Storyboard PIO Mode - Transfer Cache → CPU-Register..... | 31 |
| Abbildung 9: Storyboard PIO Mode - Transfer CPU-Register → RAM..... | 32 |
| Abbildung 10: Storyboard PIO Mode - Transfer RAM → Register..... | 32 |
| Abbildung 11: Storyboard PIO Mode - Transfer CPU-Register → HD..... | 33 |
| Abbildung 12: Storyboard PIO Mode - HD-Cache leeren..... | 33 |
| Abbildung 13: Storyboard Übersicht DMA Mode..... | 34 |
| Abbildung 14: Storyboard DMA Mode - Detailansicht..... | 34 |
| Abbildung 15: Storyboard DMA Mode - Cache laden..... | 35 |
| Abbildung 16: Storyboard DMA Mode - Request Systembus..... | 35 |
| Abbildung 17: Storyboard DMA Mode - Transfer DVD-Cache → RAM..... | 36 |
| Abbildung 18: Storyboard DMA Mode - Transfer RAM → HD-Cache..... | 36 |
| Abbildung 19: Storyboard DMA Mode - Freigabe Systembus..... | 36 |
| Abbildung 20: Farbbeispiel für den PIO Mode..... | 38 |
| Abbildung 21: Übersicht über die Präsentationsoberfläche..... | 42 |
| Abbildung 22: Grafische Buttons..... | 47 |
| Abbildung 23: Filmsteuerung..... | 55 |
| Abbildung 24: Bearbeitung der Textteile..... | 66 |
| Abbildung 25: Zeitleiste..... | 70 |
| Abbildung 26: Dokumenteigenschaften..... | 70 |
| Abbildung 27: Symbolkonvertierung..... | 71 |
| Abbildung 28: Objekteigenschaften..... | 72 |
| Abbildung 29: Zeitleistenzeiger..... | 72 |
| Abbildung 30: Bewegungstween..... | 73 |
| Abbildung 31: Playliste..... | 75 |

Tabellenverzeichnis

| | |
|---|----|
| Tabelle 1: Zusammenfassung für Microsofts PowerPoint..... | 14 |
| Tabelle 2: Zusammenfassung für Adobe Director..... | 16 |
| Tabelle 3: Zusammenfassung für 3D Studio Max..... | 17 |
| Tabelle 4: Zusammenfassung für SMIL..... | 19 |
| Tabelle 5: Zusammenfassung für Adobe Flash..... | 21 |
| Tabelle 6: Auswertung der Endergebnisse..... | 21 |
| Tabelle 7: Farbtabelle für die Benutzeroberfläche..... | 28 |
| Tabelle 8: Farbtabelle für die Animationen..... | 38 |

Abkürzungsverzeichnis

| | |
|---------|---|
| 2D | zwei dimensional |
| 3D | drei dimensional |
| AVI | Audio Video Interleave |
| CGI | Computer Generated Imagery |
| CPU | Central Processing Unit (Hauptprozessor) |
| CS3 | Creative Suite 3 |
| DCR | Dokumentformat von Director |
| DMA | Direct Memory Access |
| DVD | Digital Versatile Disc |
| ECMA | European Computer Manufacturers Association |
| FLA | Dateiendung für Flash Quelldateien |
| GIF | Graphic Interchange Format |
| HD | Harddisk (Festplatte) |
| HTML | Hypertext Markup Language |
| MPEG | Moving Picture Expert Group |
| MS-DOS | Microsoft Disk Operating System |
| PIO | Programmed Input Output |
| RAM | Random Access Memory (Arbeitsspeicher) |
| RGB | Rot Grün Blau |
| ROM | Read Only Memory |
| SWF | Dateiendungen für Shockwave Flash Dateien |
| W3C | World Wide Web Consortium, ein Gremium zur Standardisierung von WWW Anwendungen |
| WYSIWYG | „What you see is what you get“ - Bezeichnung für Programme, bei denen das bearbeitete Dokument dem Ausgabedokument sehr ähnlich sieht |
| XML | Extended Markup Language |

Begriffserklärung

| | |
|------------------------|--|
| ActionScript | eine auf JavaScript basierende Programmiersprache der Firma Adobe, ursprünglich entwickelt von Macromedia |
| Adobe Director | Autorenprogramm zum Erstellen von Anwendungen und Animationen |
| Adobe Flash | Autorenprogramm, vorwiegend zur Erstellung von Animationen |
| Autodesk 3D Studio Max | Animationsprogramm zur Erzeugung von CGI-Filmen |
| Brickfilme | komplett aus Legosteinen erstellte Filme |
| Claymation | vorwiegend aus Knete gestaltete Filme |
| Container | sinngemäß für Behälter |
| Corporate Design | Teilbereich der Unternehmensidentität, umfasst das Erscheinungsbild einer Firma |
| DMA-Controller | Der DMA-Controller dient der Übertragung von Daten zwischen dem RAM und der Peripherie, um damit den Prozessor zu entlasten. |
| Impress | Präsentationssoftware der Firma Sun Microsystems |
| JavaScript | eine objektorientierte, aber klassenlose Programmiersprache |
| Layout | Entwurf, der den Eindruck des Endergebnisses vermitteln soll |
| Lingo | die Programmiersprache der Authoringsoftware Adobe Director |
| Linux | Betriebssystem der UNIX Gruppe |
| MacOS | Betriebssystem der als Mac bekannten Rechnersysteme von Apple |
| Media Player | eine Software zum Abspielen von verschiedenen Audio- und Videoformaten |
| Playlisten | Wiedergabelisten für Multimediadateien |
| PowerPoint | Präsentationssoftware der Firma Microsoft |
| RGB-Werte | Rot, Grün und Blau – über diese drei Werte wird eine Farbe am Computer dargestellt. |
| Scrollen | dt. Bildlauf – das Bewegen eines Dokumentenausschnitts |
| SMIL | Synchronized Multimedia Integration Language – eine |

| | |
|--------------------|--|
| Storyboard | Auszeichnungssprache für zeitsynchronisierte, multimediale Inhalte handgefertigte Zeichnungen, welche zur Übersicht einer Szene beim Film erstellt werden |
| Syntaxhighlighting | Fähigkeit eines Programms, bestimmte Textsegmente als Programmcode zu identifizieren und farbig zu kennzeichnen |
| Systembus | Daten-, Adress- und Steuerleitungen zur rechnerinternen Übertragung von Daten |
| Tween | ein Effekt zwischen zwei Schlüsselbildern (Flash) |
| Windows | Betriebssystem der Firma Microsoft |
| WWW | World Wide Web; das Internet |

Hinweis: Zahlen in eckigen Klammern sind Quellenangaben, welche im Literaturverzeichnis zu finden sind.

1 Einleitung

In diesem Teil wird kurz in die Thematik der Arbeit eingeführt und es erfolgt eine inhaltliche Vorstellung der einzelnen Kapitel.

1.1 Motivation

Zirka 60% der Bevölkerung sind visuell lernende Menschen. Auditiv¹ und scriptorisch² Lernende machen jedoch weniger als 10% der Bevölkerung aus [1]. Deshalb war es in der Lehre schon immer wichtig, den Studenten oder auch Schülern Wissen bildhaft zu vermitteln. Um dieses Wissen

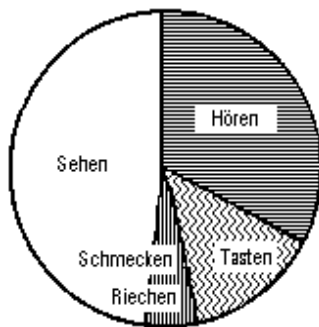


Abbildung 1: Lerntypenverteilung nach Vester [2]

optisch darzustellen, wurde früher ausschließlich die Tafel verwendet. Mit fortschreitender technischer Entwicklung kamen die Lichtprojektoren hinzu und später wurden Beamer eingeführt. Letztere bieten den Lehrenden die Möglichkeit, interaktive Präsentationen am Computer zu erstellen und als Anschauungsmaterial zu verwenden. Hierfür wird in vielen Fällen mit Microsofts PowerPoint gearbeitet. Diese Anwendung bietet jedoch nur begrenzte Möglichkeiten der

Animation. Beispielsweise müssen für das Fach Computertechnik Datenströme, Speicherfüllstände und Funktionsweisen verschiedenster Computerkomponenten demonstriert werden. Deswegen wurde nach einer Alternative zur vereinfachten Erstellung von vorlesungsbegleitenden Animationen gesucht.

Ziel dieser Arbeit ist die Konzeption und Gestaltung einer multimedialen Präsentationsplattform für Themen des Fachs Computertechnik. Es soll zudem eine Möglichkeit geschaffen werden, um Präsentationen effektiver zu erstellen und zu bearbeiten. Dazu ist eine Bibliothek mit vorgefertigten Elementen zu erstellen, welche bei Bedarf jederzeit erweitert und editiert werden kann. Des Weiteren ist ein modularer Aufbau erforderlich, um einen Einsatz auch in anderen Fachbereichen zu ermöglichen.

Die Arbeit ermittelt zunächst verschiedene Ansätze der Realisierung, um später eine konkrete Auswahl, die den Vorgaben gerecht wird, zu treffen und zu begründen. Zusätzlich werden anhand von zwei Beispielen die Möglichkeiten zukünftiger Animation dargestellt.

¹ auditiv = hörend lernen; ² scriptorisch = über Geschriebenes lernen

1.2 Kapitelübersicht

Das erste Kapitel, die **Einleitung** genannt, beinhaltet die Einführung in das Diplomthema und soll einen Überblick über die Inhalte der einzelnen Kapitel geben.

Der Abschnitt **Grundlagen**, das zweite Kapitel, klärt den Begriff der Animation. Des Weiteren wird eine Auswahl der möglichen Animationssoftware vorgestellt. Es erfolgt die Festlegung von Auswahlkriterien für die Software. Am Ende des Kapitels wird unter Berücksichtigung dieser Kriterien eine Entscheidung über die gewählte Software gefällt und begründet. Außerdem finden weitere verwendete Hilfsmittel Erwähnung.

Im dritten Kapitel **Konzeption** werden die Forderungen an die Anwendung und Animationen im Detail betrachtet. Dieses Kapitel beschreibt zusätzlich die Planungsphase und zeigt die Überlegungen auf, welche im Vorfeld der praktischen Arbeit angestellt wurden.

Der Abschnitt **Realisierung**, Kapitel vier, zeigt den Entstehungsprozess der allgemeinen Präsentationsoberfläche und deren modularen Aufbau. Des Weiteren wird die Erstellung der einzelnen Beispielanimationen für die Themengebiete PIO Mode und DMA Mode aufgezeigt.

Das Kapitel fünf, **Ergebnis**, reflektiert den gesamten Entstehungsprozess. Es werden die Ergebnisse der Arbeit vorgestellt, der aktuelle Projektstand zusammengefasst und Vorschläge für mögliche Erweiterungen und Verbesserungen aufgezeigt.

Das sechste Kapitel präsentiert abschließend ein **Handbuch**, welches nachfolgenden Projektbearbeitern den Einstieg in die Animationserstellung erleichtern soll.

2. Grundlagen

Dieses Kapitel befasst sich mit den Grundlagen der Arbeit. Es wird der Begriff der Animation erklärt, sowie eine Auswahl möglicher Software getroffen.

2.1 Was ist Animation?

Der Begriff Animation ist von dem lateinischen Wort animare abgeleitet, was soviel bedeutet wie „zum Leben erwecken“. Physikalisch stellt Animation das Zusammenwirken von Stroboskopeffekt und Nachbildwirkung (Trägheit des Auges) dar. Ab einer Frequenz von etwa fünfzehn Bildern pro Sekunde werden Bilder auf der Netzhaut des menschlichen Auges nicht mehr als Einzelbilder, sondern als filmische Bewegung wahrgenommen, die optisch einer realen Bewegung ähnelt. Eine Animation besteht in der Regel aus 24 einzelnen Bildern für eine Sekunde Film, damit die Illusion einer flüssigen Bewegung erzeugt wird. Dabei ist es irrelevant, ob die Bilder von Hand gezeichnet, fotografiert oder im Computer erstellt wurden. Animationen setzte man schon in der frühesten Zeit der Filmtechnik ein und verwendet sie bis heute. Es gibt verschiedenste Formen des 2D- und 3D-Objekt-Animationsfilms. Beispiele für 2D-Animationen wären Zeichentrick-, Öl-auf-Glas- oder die Legetricktechnik. Für die Objektanimation stehen die Puppentrick-, Claymation- oder Brickfilme. Aufgrund des zu hohen Arbeitsaufwandes der verschiedenen Techniken ist für die gegebene Aufgabenstellung der Diplomarbeit nur die 2D-Zeichentrickmethode oder 3D-CGI-Animation interessant.

Als Grundlage der Zeichentrickanimation stünden generell zwei Methoden zur Verfügung:

1. Die Animationen als animierte GIF-Dateien zu erstellen. Diese Dateien sind selbst laufend und nicht steuerbar, womit diese Methode entfällt .
2. Die andere Möglichkeit bestünde in animierten Filmen. Für die Erstellung solcher Filme gibt es zahlreiche Verfahren. Diese Arbeit stellt einige dieser Methoden vor und zeigt die Vor- und Nachteile der einzelnen Animationsprogramme auf.

2.2 Mögliche Animationssoftware

Als Anwendungen zur Animation standen zu Anfang der Arbeit verschiedene Programme bzw. Scriptsprachen zur Auswahl. Diese sind: Microsofts PowerPoint, Macromedia bzw. Adobe Director, Macromedia Flash 8 bzw. Adobe Flash CS3, Autodesk 3D Studio Max und SMIL.

Um eine passende Auswahl zu treffen, müssen verschiedene Kriterien aufgestellt und gewichtet werden. Es wurden drei Stufen der Gewichtung festgelegt: Sehr wichtig (Faktor 3), mäßig wichtig

(Faktor 2) und wenig wichtig (Faktor 1).

Im Folgenden sollen nun die Kriterien und ihre jeweiligen Faktoren aufgeführt werden:

- Verbreitung der Software (2)
- Plattformabhängigkeit für die Entwicklung (1)
- Plattformabhängigkeit für die Ausführung (1)
- Bedienbarkeit der Software (1)
- Art der Animation (2)
- benötigter Player (2)
- Speicherbedarf (2)
- Änderungsaufwand (2)
- Vielfältigkeit der Animationsmöglichkeiten (3)
- Modularität (3)

Anhand dieser Kriterien und den Anforderungen an die Anwendung trifft der Autor im Folgenden eine Auswahl. Das Ende eines jeden Abschnitts zeigt eine tabellarische Zusammenfassung auf, welche als Entscheidungsgrundlage dienen soll. Einige Kriterien (z.B. Änderungsaufwand) sind subjektive Einschätzungen des Autors, welche aufgrund von Erfahrungen mit der entsprechenden Software gewonnen wurden. Die Bewertung geschieht mittels eines Fünf-Sterne-System, wobei die bestmögliche Anzahl fünf Sterne sind. Am Ende jeder Auswertung wird der Durchschnitt aus den einzelnen Kriterien gebildet und der genaue Wert angegeben.

2.2.1 Microsoft PowerPoint

Das Programm PowerPoint ist die weltweit am meisten genutzte Präsentationssoftware. Laut dem Artikel „Papyrus to PowerPoint (P 2 P): metamorphosis of science communication“ [3] sind über 95% aller weltweit gestalteten Präsentationen mit Microsofts PowerPoint erstellt. Die große Verbreitung von Microsoft Office, in welchem PowerPoint, selbst in der günstigsten Paketvariante, enthalten ist, stellt einen entscheidenden Faktor für die große Popularität der Präsentationssoftware dar.

PowerPoint bietet eine einfache Lösung, um Präsentationen für die verschiedensten Gelegenheiten zu erstellen. Die Bandbreite reicht dabei von Firmenpräsentationen über die aktuelle Finanzsituation bis hin zu Geschichtsvorträgen über das Römische Reich in der Schule. Es ist für jeden Computeranwender sehr schnell erlernbar und leicht zu bedienen.

Auf Grund der recht übersichtlichen Benutzeroberfläche gestaltet sich der Einstieg in PowerPoint für jeden Computeranwender sehr einfach. Durch die veränderte Anwendungsoberfläche von

PowerPoint 2007, im Vergleich zu den Vorgängerversionen, wird die Nutzung für gewohnte Anwender jedoch etwas erschwert. Wegen der generellen Einfachheit des Programms ist es jedoch nur bedingt möglich, komplexe Animationen zu erstellen oder zu bearbeiten.

Die Animationen in PowerPoint sind voreingestellte Funktionen, welche auf einzelne Objekte der Präsentation angewendet werden können. Zur Erstellung von PowerPoint Präsentationen ist entweder Microsofts PowerPoint oder ein vergleichbares Programm, wie z.B. Impress von Sun, erforderlich. PowerPoint ist für jedes der drei gängigsten Systeme (Windows, Linux, MacOS) verfügbar. Dadurch erhält es die eingangs erwähnte hohe Verbreitung.

Um PowerPoint-Präsentationen zu nutzen, ist entweder Microsoft PowerPoint, ein vergleichbares Programm oder ein PowerPoint-Viewer erforderlich. Die Dateien werden als .ppt abgelegt und sind je nach Präsentationslänge und Inhalt bis zu mehreren Megabyte groß.

Die Fachpresse bewertet Microsoft PowerPoint als solides Präsentationsprogramm. Internetportale wie cnet [4] und PC Advisor [5] vergaben jeweils 3,5 von fünf möglichen Punkten. Die deutsche Zeitung Computerbild [6] bewertete PowerPoint 2007 mit der Note 1,99.

| Kriterium | Angabe | Bewertung | Faktor | Gesamt |
|------------------------------|-------------------------|------------------|---------------|---------------|
| Verbreitung der Software | Sehr hoch | ***** | 2 | 10 |
| Betriebssystem (Entwicklung) | Windows, Linux, Mac | ***** | 1 | 5 |
| Betriebssystem (Ausführung) | Windows, Linux, Mac | ***** | 1 | 5 |
| Bedienbarkeit der Software | Einfach | ***** | 1 | 5 |
| Art der Animation | PowerPoint Präsentation | *** | 2 | 6 |
| Benötigter Player | Power Point/Viewer | **** | 2 | 8 |
| Speicherbedarf | Mittel | *** | 2 | 6 |
| Aufwand bei Änderungen | Mittel | *** | 2 | 6 |
| Möglichkeiten der Animation | Gering | * | 3 | 3 |
| Modularität möglich | nein | * | 3 | 3 |
| Besonderheiten | | | | |
| Gesamt | | | 19 | 57 |
| Gesamtnote | | | | 3/5 |

Tabelle 1: Zusammenfassung für Microsofts PowerPoint

Für Geschäfts-, Vorlesungs- oder Vortragspräsentationen ist PowerPoint mehr als ausreichend ausgestattet. Komplexe Animationen zu erstellen ist nur bedingt möglich.

2.2.2 Director

Adobe Director ist ein Autorenprogramm zum Erstellen von interaktiven, multimedialen Anwendungen. Diese werden zum Beispiel als Menü oder multimediale Anwendungen für CDs und DVDs verwendet. Außerdem beinhaltet Director die objektorientierte Programmiersprache Lingo.

Die Bedienung der Software kann intuitiv erfolgen. Es ist für einfache Animationen bzw. Anwendungen keine tief gehende Kenntnis des Programms erforderlich. Um jedoch komplexere Anwendungen zu erstellen, benötigt man Kenntnisse über die Programmiersprache Lingo.

Zur vereinfachten Erstellung der Animationen werden Bibliotheken mit selbst gestalteten, vorgefertigten Objekten benötigt. Director bietet eine vorgefertigte Bibliothek mit Buttons, Textfeldern, Checkboxes und ähnlichen Objekten.

Außerdem besteht bei Director die Möglichkeit XML einzubinden. Dadurch würden leicht veränderbare Inhalte ermöglicht, die eine modulare Strukturierung der Anwendung gestatten.

Die Erstellung von Programmen erfordert in diesem Fall die Software Adobe Director, welche für Microsoft Windows und MacOS verfügbar ist. Die Dateien, welche mit Director erstellt werden, bezeichnet man als DCR-Dateien. Diese sind sehr klein, was eine Streamingfähigkeit auch für langsame Internetverbindungen ermöglicht. Ihre Größe beträgt meist nur wenige hundert Kilobyte. Zur Präsentation wird hierfür der Shockwave-Player benötigt, welcher für alle gängigen Webbrowser verfügbar ist und somit zu einer hohen Verbreitung der Anwendungen führt.

Die Pressestimmen zu Adobe Director 11 sind eher verhalten. Das Internetportal magnus.de [7] bewertet die Software nur mit einem „befriedigend“. Begründet wird das u.a. mit zu wenigen Neuerungen im Vergleich zur Vorgängerversion.

| Kriterium | Angabe | Bewertung | Faktor | Gesamt |
|------------------------------|------------------|-----------|--------|--------|
| Verbreitung der Software | Hoch | ***** | 2 | 10 |
| Betriebssystem (Entwicklung) | Windows, Mac | **** | 1 | 4 |
| Betriebssystem (Ausführung) | Windows, Mac | **** | 1 | 4 |
| Bedienbarkeit der Software | | *** | 1 | 3 |
| Art der Animation | Anwendung | **** | 2 | 8 |
| Benötigter Player | Shockwave Player | **** | 2 | 8 |
| Speicherbedarf | Gering | **** | 2 | 8 |
| Aufwand bei Änderungen | Mittel | *** | 2 | 6 |
| Möglichkeiten der Animation | Mittel | *** | 3 | 6 |
| Modularität möglich | Ja | ***** | 3 | 15 |
| Besonderheiten | | | | |
| Gesamt | | | 19 | 72 |
| Gesamtnote | | | | 3,8/5 |

Tabelle 2: Zusammenfassung für Adobe Director

2.2.3 Autodesk 3D Studio Max

Das 3D Studio Max ist eine Software zur Erstellung von 3D-Modellen, 3D-Animationen und visuellen Effekten. Es ist hervorragend geeignet, um grafisch ansprechende Animationen für jeden erdenklichen Bereich zu erstellen. Die Einsatzgebiete reichen von Architektur über Maschinenbau bis hin zu großen Kinofilmproduktionen und Videospiele. Es wurde 1990 das erste Mal für MS-DOS unter dem Namen 3D Studio DOS vorgestellt. 1996 erfolgte die Umbenennung in 3D Studio Max. Inzwischen kommt es seitdem fast jedes Jahr zur Veröffentlichung einer neuen Version von 3D Studio Max. Die aktuelle elfte Version heißt 3D Studio Max 2009 (Stand Mai 2008).

3D Studio Max bringt eine große Menge vorgefertigte Objekte und geometrische Grundelemente mit, die man lediglich in Größe und Farbe festlegen muss. Die Elemente können dann noch mit Texturen versehen werden, um ein realistischeres Aussehen zu erhalten.

Die Animation in 3D Studio Max erfolgt über Schlüsselbilder in einer Zeitleiste, welche eine Veränderung der erstellten Objekte ermöglichen. Die Bilder zwischen den Schlüsselbildern werden alle einzeln berechnet. Das Rendern der Animationen erfolgt ebenfalls in Einzelbildern, was einen erheblichen Speicherbedarf bewirkt. Mit 3D Studio Max erstellte Animationen werden in der Regel als selbst laufende Videos im AVI, MPEG oder anderen Videoformaten gespeichert. Für deren Präsentation kann nahezu jeder herkömmliche Media Player (z.B. Windows Media Player)

verwendet werden. Diese Videos belegen in der Regel sehr viel Festplattenspeicher. Je nach Lauflänge, Format und Komprimierungscodec können die Dateien Größen von mehreren hundert Megabyte Festplattenspeicher belegen. Die Quelldateien werden als MAX-Dateien abgespeichert. Diese können ebenfalls je nach Komplexität und Lauflänge der Animation mehrere Megabyte in Anspruch nehmen.

Eventuelle Änderungen an den Animationen gestalten sich als aufwändig, da in diesen Fällen große Teile der Vorlagen geändert werden müssen.

Durch seine Vielseitigkeit ist dieses Programm auf dem Animationssektor sehr weit verbreitet. Es findet in den verschiedensten Bereichen der Industrie Anwendung.

Die Zeitschrift DigitalArts [8] bewertet 3D Studio Max 2008 mit 4,5 von 5 möglichen Punkten. Begründet wird dies durch die verbesserten Modellierungs- und Charaktererstellungsmöglichkeiten sowie dem verbesserten mental ray Rendering im Vergleich zur Vorgängerversion 3D Studio Max 9.

| Kriterium | Angabe | Bewertung | Faktor | Gesamt |
|------------------------------|---------------------------|------------------|---------------|---------------|
| Verbreitung der Software | Hoch | ***** | 2 | 10 |
| Betriebssystem (Entwicklung) | Windows, Mac | **** | 1 | 4 |
| Betriebssystem (Ausführung) | Windows, Mac | **** | 1 | 4 |
| Bedienbarkeit der Software | Schwer | ** | 1 | 2 |
| Art der Animation | Gerenderte Videos | *** | 2 | 6 |
| Benötigter Player | z.B. Windows Media Player | ***** | 2 | 10 |
| Speicherbedarf | Sehr hoch | ** | 2 | 4 |
| Aufwand bei Änderungen | Sehr hoch | * | 2 | 2 |
| Möglichkeiten der Animation | Sehr gut | ***** | 3 | 15 |
| Modularität möglich | Bedingt | ** | 3 | 6 |
| Besonderheiten | | | | |
| Gesamt | | | 19 | 63 |
| Gesamtnote | | | | 3,3/5 |

Tabelle 3: Zusammenfassung für 3D Studio Max

2.2.4 SMIL

SMIL (Synchronized Multimedia Integration Language) ist ein Standard für eine

Auszeichnungssprache für multimediale Inhalte und wurde im Jahr 1998 erstmals von dem World Wide Web Consortium vorgestellt. Die aktuelle Version ist SMIL 3.0 und ist am 15. Januar 2008 veröffentlicht worden (Stand Mai 2008).

SMIL ist eine textbasierende Anwendung von XML. Über die Syntax wird das Layout der Präsentationen erstellt. Als Erleichterung gibt es verschiedene Texteditoren, die über entsprechende Implementierungen zur Erkennung der Sprache verfügen.

Abbildung 2: Aufbau einer SMIL-Datei

In Abbildung 2 ist der strukturelle Aufbau einer SMIL-Datei dargestellt. Da SMIL in seiner Struktur und in seinem Aufbau XML bzw. auch HTML sehr ähnlich ist, können entsprechende Parallelen gezogen werden. So beginnt eine HTML-Datei mit `<html>` und eine SMIL-Datei mit `<smil>`. Ebenso kann es einen `<head>` geben, der für entsprechende Metadaten oder Layoutelemente zur Verfügung steht. Wie bei HTML muss es auch bei SMIL einen `<body>` geben, in welchem die einzelnen Elemente der gewünschten SMIL-Datei eingebunden werden. Auf diese Art können unterschiedlichste Grafiken, Videos, Audiodateien o.ä. miteinander zu einer Präsentation verknüpft werden.

Ein Nachteil für die Arbeit mit SMIL besteht darin, dass es keine WYSIWYG-Editoren für diese Sprache gibt. Demzufolge müssen Benutzer die Kenntnisse für SMIL besitzen um damit arbeiten zu können. Als einzige Erleichterung gibt es einige Texteditoren, welche SMIL unterstützen. Auf diese Art kann die Syntax leichter auf Fehler überprüft werden und bestimmte Befehle werden farblich hervorgehoben. Dieses farbige Markieren und Hervorheben von Quellcode bezeichnet man auch als Syntaxhighlighting.

Trotz der verschiedenen Kombinationsmöglichkeiten von multimedialen Objekten hat sich SMIL

nie richtig durchgesetzt, weswegen dessen Verbreitung sehr gering ist.

Die auf diese Art zusammengestellten Präsentationen sind eine Art interaktiver Film. Zur Wiedergabe der Dateien, welche aufgrund des reinen Textinhaltes nur wenige Kilobyte groß sind, wird z.B. der RealPlayer verwendet. Die Gesamtspeichergröße einer SMIL-Präsentation hängt jedoch ebenso von den verwendeten Grafik-, Video- und Audioobjekten ab.

| Kriterium | Angabe | Bewertung | Faktor | Gesamt |
|------------------------------|--------------------------|-----------|--------|--------|
| Verbreitung der Software | Gering | * | 2 | 2 |
| Betriebssystem (Entwicklung) | Windows, Linux, Mac | ***** | 1 | 5 |
| Betriebssystem (Ausführung) | Windows, Linux, Mac | ***** | 1 | 5 |
| Bedienbarkeit der Software | Schwer | ** | 1 | 2 |
| Art der Animation | Interaktive Präsentation | *** | 2 | 6 |
| Benötigter Player | z.B. Real Player | *** | 2 | 6 |
| Speicherbedarf | Mittel | *** | 2 | 6 |
| Aufwand bei Änderungen | Mittel | *** | 2 | 6 |
| Möglichkeiten der Animation | Mittel | ** | 3 | 6 |
| Modularität möglich | Ja | **** | 3 | 12 |
| Besonderheiten | | | | |
| Gesamt | | | 19 | 56 |
| Gesamtnote | | | | 2,9/5 |

Tabelle 4: Zusammenfassung für SMIL

2.2.5 Flash

Flash ist eine Entwicklungsumgebung für multimediale Inhalte. Die damit erstellten Filme werden kurz Flashfilme genannt. Dateien, die mit Flash angefertigt wurden, liegen in einem auf Vektorgrafiken basierenden Grafik- und Animationsformat vor. Um selbige abspielen zu können, ist lediglich der Flash Player erforderlich, der für alle gängigen Webbrowser als Plugin erhältlich ist. Die daraus resultierende Plattformunabhängigkeit sorgt für einen Verbreitungsgrad von ca. 99% aller Internetnutzer [9].

1996 entwickelte die Firma FutureWave das Programm FutureSplash-Animator. Im selben Jahr übernahm Macromedia das Unternehmen und die Software sowie der Player werden unter den Namen Flash und Shockwave Flash Player weiterentwickelt.

Im Jahr 1997 veröffentlichte man die erste Version von Flash und dem dazugehörigen Shockwave Flash Player. Noch im selben Jahr erschienen die Version 2 und ein erweiterter Player. Mit dieser

Version ließen sich erstmals einfache Interaktionen umsetzen.

Flash 3 wurde 1998 präsentiert und brachte wesentliche Neuerungen für Aktionen bei Schlüsselbildern und Bildern. Außerdem wurde das Testen von Animationen aus der Entwicklungsumgebung heraus ermöglicht.

Im Jahr 2000 wurde das bereits vorhandene ActionScript stark verändert und an ECMA-Standard angepasst, auf dem auch JavaScript basiert.

Nach der Übernahme von Macromedia durch Adobe im Jahr 2005 wurden sämtliche Produkte in die Adobe Produktpalette übernommen. Die aktuelle Version heißt Adobe Flash CS4 und ermöglicht u. a. Importe von Adobe Photoshop und Adobe Illustrator (Stand Oktober 2008).

Flash bietet eine gut strukturierte und leicht zu bedienende Nutzeroberfläche, welche Neulingen den Einstieg in die Animation erleichtert.

Die Verknüpfung von XML und Flash ist möglich, was es erleichtert, einzelne Module, Playlisten oder andere Daten separat zu erstellen, zu bearbeiten und zu laden. Flash bietet außerdem eine Kantenglättung, die auch bei starker Vergrößerung auf einer Leinwand keine Stufenbildung bei nicht waagerechten oder senkrechten Linien hervorruft.

Die Animation mit Flash erfolgt über Schlüsselbilder, welche eine Veränderung der benötigten Objekte ermöglicht. Um einen bewegten Übergang zwischen den Schlüsselbildern zu erschaffen wird ein sogenannter Tween verwendet. Die einzelnen Zwischenbilder werden von Flash interpoliert. Es ist sogar möglich die Animationen zu „recompilieren“ und daraus wieder eine bearbeitbare Datei zu erstellen.

Die Entwicklungsdateien werden als FLA-Dateien abgespeichert und können je nach Komplexität und Lauflänge wenige Megabyte Größe annehmen. Die daraus erstellten SWF-Dateien sind jedoch nur wenige Kilobyte groß.

Änderungen in den FLA-Dateien gestalten sich sehr einfach, da man oftmals nur wenige Handgriffe benötigt, um den Inhalt einer Animation zu ändern.

Die Animationsplattform Flash 8 wurde von der PC Welt am 15.08. 2005 mit dem Testurteil „Gut“ bewertet. Der komplette Testbericht ist auf der Homepage der PC Welt nachzulesen [10]. Auch von anderen Internetplattformen und Zeitschriften (z.B. CNET, DigitalArts) wurde Flash mit 80% bewertet, was ebenfalls einem Qualitätsurteil „Gut“ entspricht.

| Kriterium | Angabe | Bewertung | Faktor | Gesamt |
|------------------------------|---------------------|-----------|--------|--------|
| Verbreitung der Software | Sehr Hoch | ***** | 2 | 10 |
| Betriebssystem (Entwicklung) | Windows, Mac | **** | 1 | 4 |
| Betriebssystem (Ausführung) | Windows, Linux, Mac | ***** | 1 | 5 |
| Bedienbarkeit der Software | Leicht, intuitiv | **** | 1 | 4 |
| Art der Animation | Flashfilme | ***** | 2 | 10 |
| Benötigter Player | Flash Player | ***** | 2 | 10 |
| Speicherbedarf | Gering | **** | 2 | 8 |
| Aufwand bei Änderungen | Gering | ***** | 2 | 10 |
| Möglichkeiten der Animation | Gut | **** | 3 | 12 |
| Modularität möglich | Ja | ***** | 3 | 15 |
| Besonderheiten | Kantenglättung | | | |
| Gesamt | | | 19 | 88 |
| Gesamtnote | | | | 4,6/5 |

Tabelle 5: Zusammenfassung für Adobe Flash

2.3 Entscheidung

In den vorangegangenen Abschnitten wurden die einzelnen Programme vorgestellt und nach bestimmten Kriterien bewertet. Es ist dabei zu bemerken, dass jede mögliche Software ihre Stärken und Schwächen hat. Eine Entscheidung, welches Programm in der weiteren Arbeit Verwendung findet, soll in diesem Kapitel gefällt und begründet werden.

Die folgende Tabelle fasst noch einmal sämtliche Ergebnisse zusammen.

| Software | Gesamtwertung |
|---------------------------|---------------|
| Microsoft PowerPoint | 3 |
| Adobe/Macromedia Director | 3,8 |
| 3D Studio Max | 3,3 |
| SMIL | 2,9 |
| Adobe/Macromedia Flash | 4,6 |

Tabelle 6: Auswertung der Endergebnisse

Durch die hohe Bewertung ergibt sich die Entscheidung für Adobe/Macromedia Flash. Die exakten Entscheidungsgründe werden im Folgenden ausführlich erläutert.

Als erstes sollen die Gründe für das Ausscheiden von Microsofts PowerPoint genannt werden. Die Möglichkeiten der Animation erschöpfen sich in einfach bewegten Texten, Grafiken oder bereits fertigen Videos. Es ist nur bedingt möglich, komplexe Zusammenhänge, wie die geforderten Themen, animiert darzustellen. Außerdem ist eine Bibliothekenerstellung oder Modularität nicht möglich. Als Anwendung, um Vorlesungen zu präsentieren, ist das Programm dennoch geeignet.

Adobe Director bietet zwar eine fertige Bibliothek mit Objekten, diese ist jedoch für die gewünschten Ergebnisse nicht ausreichend. Eigene Objekte lassen sich dieser Bibliothek nicht hinzufügen. Außerdem soll keine interaktive Anwendung geschaffen werden, sondern eine Präsentationsoberfläche für Animationen. Die Möglichkeiten der freien Animation sind ebenfalls sehr begrenzt. Director ist ein Programm, welches vorwiegend zur Erstellung von interaktiven Anwendungen oder Spielen verwendet wird. Außerdem verfügt Director über keine Kantenglättung, was bei einer Präsentation über den Beamer zu unschöner Stufenbildung bei diagonalen Linien führen würde. In der Fachwelt gilt Adobe Director trotz der vielseitigen Möglichkeiten inzwischen als überholt [7]. Es sind bisher keine Pläne bekannt, einen weiteren Nachfolger zu veröffentlichen.

Die Auszeichnungssprache SMIL ist ebenfalls für die vorgesehene Aufgabe nicht geeignet, da man für die Erstellung von SMIL Objekten weitreichende Kenntnisse der Sprache benötigt. Außerdem können mit SMIL keine eigentlichen Animationen erstellt werden, sondern lediglich verschiedene Objekte, wie Filme, Grafiken oder Audioaufzeichnungen, zu einer multimedialen Präsentation verknüpft werden. Da SMIL eine textbasierende Sprache ist, besteht keine Möglichkeit, Bibliotheken zur späteren Verwendung anzulegen. Des Weiteren ist der Verbreitungsgrad des RealPlayers mittlerweile nicht mehr ausreichend genug, um eine effektive Abdeckung für die Präsentationen zu erreichen. Der Änderungsaufwand bei SMIL Präsentationen ist bei unzureichenden Kenntnissen ebenfalls sehr hoch.

Das 3D Studio Max ist zwar für die Erstellung von Animationen hervorragend geeignet, allerdings gibt es keine Variante, eine Bedienungsfläche für Animationen in einem 3D-Programm zu erstellen. Es wäre allenfalls möglich, ein Design zu erarbeiten und dieses später als Hintergrund einzubinden. Der Aufwand für so eine Arbeit ist jedoch unverhältnismäßig hoch. Für die Bedienung von 3D Studio Max ist ein außerdem sehr tief gehendes Wissen des Programms erforderlich, weswegen es als Programm zur Erstellung der benötigten Animationen nur bedingt in Frage kommt. Des Weiteren ist die Erstellung einer Animation sehr aufwendig, da alles in drei Dimensionen abgestimmt werden muss. Für unerfahrene Benutzer gestaltet sich die Bedienung des Programms schwierig, da es für die gegebene Aufgabenstellung zu komplex ist.

Das 3D Studio bietet eine Unmenge an vorgefertigten Objekten, jedoch ist es nicht möglich, eigens

erstellte Objekte einer dauerhaften Bibliothek hinzuzufügen.

Die Flash Entwicklungsumgebung hat einen sehr hohen Verbreitungsgrad, welcher optimal ist, um jedem Studenten die Animationen zugänglich zu machen. Außerdem bietet sie eine intuitiv bedienbare Oberfläche. Es können Bibliotheken für selbst erstellte Objekte angelegt und bearbeitet werden. Des Weiteren ist es möglich, Vorlagen zu erstellen und weiterzuverwenden. Flash hat außerdem den praktischen Nebeneffekt der Kantenglättung, was sich positiv bei der Darstellung auf großen Flächen auswirkt. Die Verbindung mit XML bietet außerdem die Möglichkeit, die Animationen und auch die Nutzeroberfläche modular aufzubauen. Einzelne Änderungen in der Menüstruktur oder in der Reihenfolge der Animationen lassen sich auf diese Art sehr leicht vornehmen.

Der Autor der Arbeit hat sich dazu entschieden, Macromedia Flash 8 zu verwenden, da hier das an der Hochschule gelehrt ActionScript 2.0 Anwendung findet. Dieses wird auch von Adobe Flash CS3 noch unterstützt, was eine Bearbeitung in Zukunft weiterhin ermöglicht.

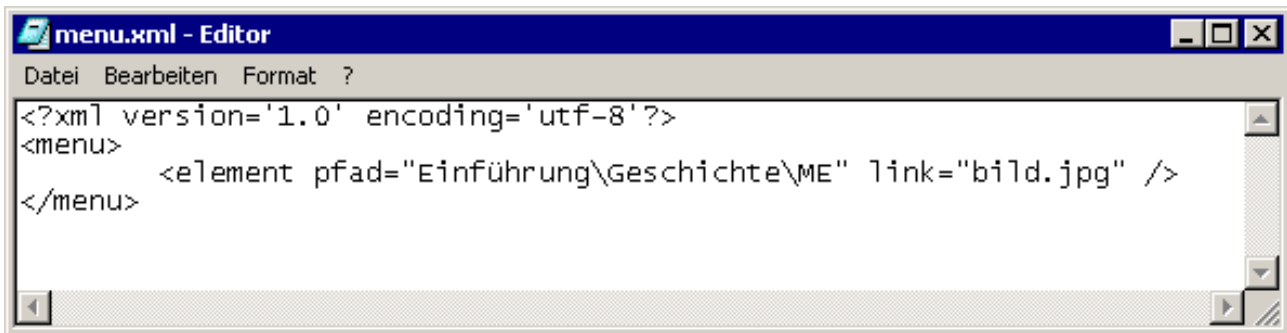
2.4 Hilfsprogramme

Zur Erfüllung der Aufgabenstellung war es notwendig, weitere Software zu verwenden. Diese finden an dieser Stelle jedoch nur Erwähnung, auf deren Funktionalitäten wird nicht näher eingegangen. Es ist für spätere Bearbeitung und Erstellung nicht erforderlich speziell diese Programme zu nutzen.

Da die gestalterischen Möglichkeiten von Flash begrenzt sind, musste man ein Grafikbearbeitungsprogramm hinzuziehen. Zu diesem Zweck wurde auf Adobe Photoshop 7 zurückgegriffen, mit welchem die Entwicklung des grafischen Designs der Benutzeroberfläche erfolgte.

Als Hilfsmittel für die modulare Erstellung der Anwendung und der Animationen wurde XML verwendet. Die Extensible Markup Language (XML) ist eine erweiterbare Auszeichnungssprache für hierarchische strukturierte Daten in Form von Textdateien. Die erste Spezifikation wurde 1998 vom W3C veröffentlicht, die aktuellste Version Fünf im November 2008 (Stand Mai 2009).

Durch die frei definierbaren Elemente eignet sich XML hervorragend als Container für die zu ladenden Module. Da es sich um eine Textdatei handelt, ist das Ändern des Inhalts sehr einfach. Man kann nahezu alle Inhalte dynamisch laden. Damit ermöglicht es für die Anwendung eine Modularität, wodurch sie leicht für andere Fachgebiete einsetzbar ist.

The image shows a screenshot of a text editor window titled "menu.xml - Editor". The window has a menu bar with "Datei", "Bearbeiten", "Format", and "?". The main text area contains the following XML code:

```
<?xml version='1.0' encoding='utf-8'?>
<menu>
  <element pfad="Einführung\Geschichte\ME" link="bild.jpg" />
</menu>
```

The code is displayed in a monospaced font. The window also features standard Windows window controls (minimize, maximize, close) in the top right corner and a scrollbar on the right side.

Abbildung 3: Aufbau einer XML-Datei

In Abbildung 3 wird die Grundstruktur einer XML-Datei dargestellt. Diese beginnt immer mit der XML- Definition. Danach folgen selbst gewählte Elemente, welche mit einem Start- und einem End-Tag versehen werden müssen. Sie können auch mehrfach ineinander verschachtelt werden. Das erste Element wird dabei als Wurzelement bezeichnet.

Die Bearbeitung der XML-Dateien wurde mittels des Standardtexteditors von Microsoft Windows vorgenommen. Für XML unerfahrene Nutzer empfiehlt sich hier ein Programmiereditor, der die XML-Elemente kennzeichnet und gegebenenfalls korrigiert, was die Bearbeitung erleichtert.

3. Konzeption

Nachdem im vorangegangenen Kapitel eine Auswahl der Software getroffen wurde, befasst sich dieses Kapitel mit den im Vorfeld der Realisierung notwendigen Überlegungen zur Entwicklung der Benutzeroberfläche und der Animationen. Außerdem erfolgt eine Formulierung der genauen Forderungen an die Anwendung und die Animationen.

3.1 Forderungen

Es soll eine Anwendung geschaffen werden, die es dem Nutzer ermöglicht, komplexe Animationen als begleitendes Vorlesungsmaterial zu verwenden. Diese soll einfach zu modifizieren und zu erweitern sein. Darüber hinaus sollen die Animationen als einzelne Module gestaltet werden, um sie für verschiedene Studiengänge unterschiedlich gestalten zu können.

Da es sich um eine Anwendung für den überwiegend mobilen Einsatz handelt, muss auch eine entsprechende Größe (Auflösung) eingehalten werden; unter Umständen könnten sonst Teile der Anwendung nicht sichtbar sein. Außerdem muss die Lesbarkeit der Texte unbedingt garantiert werden.

Die Steuerung der Laptops erfolgt in vielen Fällen mit der Maus oder durch eine Fernsteuerung, die eine Maus simuliert. Deshalb darf die Steuerung nicht ausschließlich auf Tastaturbefehle ausgelegt sein.

Des Weiteren sollen die Animationen leicht verständlich und zum größten Teil selbsterklärend sein. Die pädagogische Wirkung für Lernende muss außerdem zum Tragen kommen. Es darf nur ein Mindestmaß an erklärendem Text vorhanden sein um, die Aufmerksamkeit nicht zu stark von der Animation abzulenken.

3.2 Gedanken zur Gestaltung der Benutzeroberfläche

Dieser Abschnitt befasst sich mit den Gedanken, welche der Erstellung der Benutzeroberfläche vorangegangen sind. Zudem werden Überlegungen und Ideen aufgezeigt, welche im Vorfeld der praktischen Arbeit in Bezug auf die Gestaltung der Oberfläche angestellt wurden.

3.2.1 Struktur

Die Navigation ist das Hauptelement der Benutzeroberfläche. Deshalb muss die Navigation übersichtlich strukturiert und leicht zu erreichen sein. Es darf nicht jeder Menüpunkt als einzelner Button angelegt werden, da das Menü einfach zu erweitern und zu ändern sein soll. Aus diesem Grund fiel die Entscheidung zugunsten einer über XML baumartig aufgebauten Struktur der

Navigation. Die Menüpunkte sind deshalb vorläufig der Struktur des Vorlesungsaufbaus des Fachs Computertechnik entnommen worden. Eine Unterteilung erfolgt an dieser Stelle in grobe Themengebiete, welche wiederum in weitere Unterkategorien aufgeteilt sind. In der untersten Ebene werden die einzelnen Studiengänge aufgeführt, da es bei bestimmten Animationen verschiedene Ausführungen für verschiedene Studiengänge geben soll. Vorläufig wurde diese Art gewählt, damit eine Grundlage vorhanden ist, von der aus bestimmte Funktionalitäten der Anwendung ausgearbeitet und getestet werden können. Dies ist keine endgültig feste Lösung und es ist möglich die Struktur auch zukünftig weiter zu überarbeiten.

3.2.2 Design

Wichtig für eine Präsentationsoberfläche ist auch das Design. Es sollte einen hohen Wiedererkennungswert besitzen und trotzdem funktionell erscheinen. Das Design der Benutzeroberfläche ist an jenes der Vorlesungsunterlagen des Fachs Computertechnik angelehnt. In Abbildung 4 ist am unteren Rand der Name des Dozenten, sein Fachbereich sowie die zugehörige Lehranstalt zu finden. Am rechten oberen Rand ist ein Textfeld, in welchem die aktuelle Überschrift bzw. die Stelle in der Navigation eingeblendet werden soll.

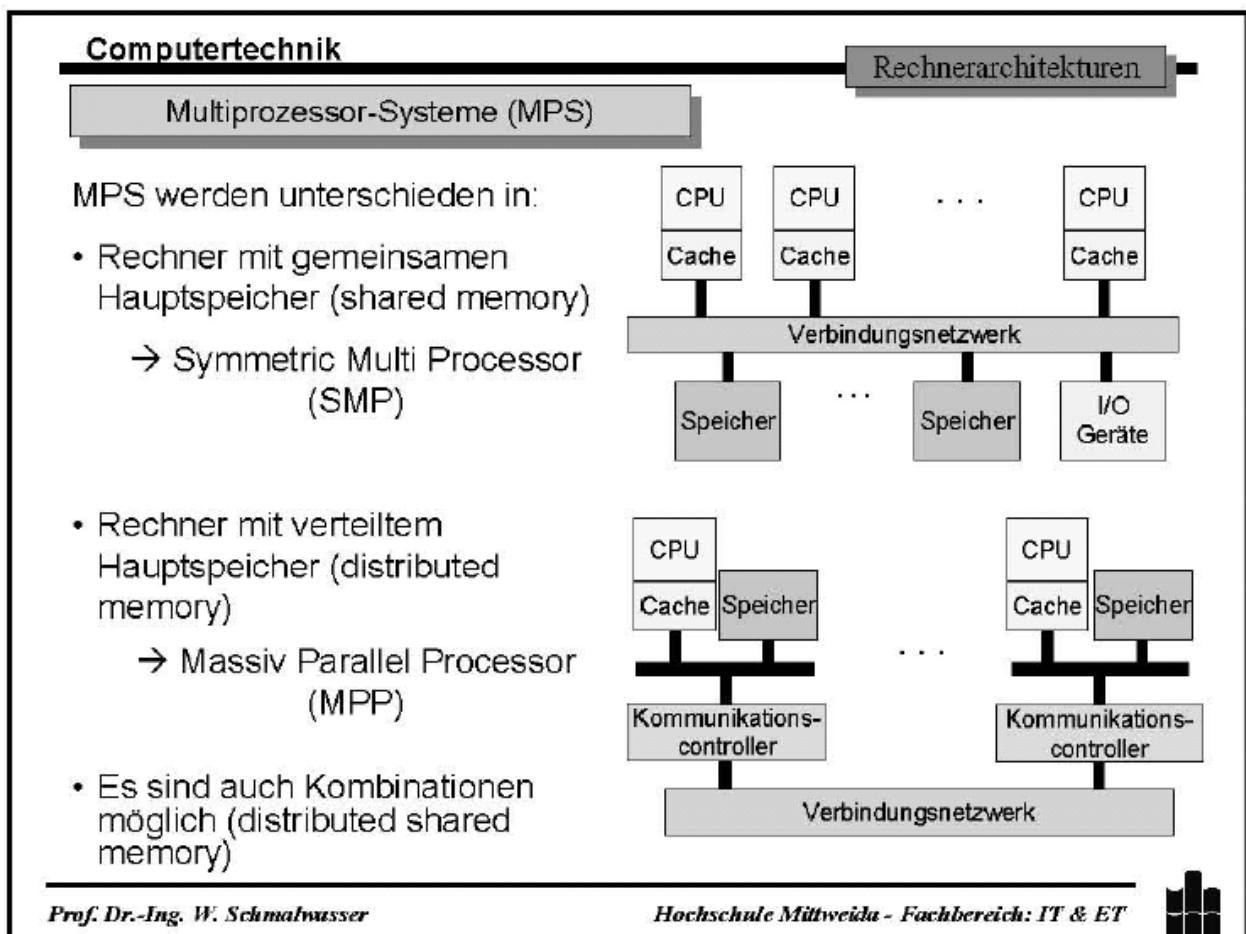


Abbildung 4: Beispiel der Vorlesungsunterlagen

Da es sich in vielen Bereichen bewährt hat, wurde entschieden, die Navigation auf der linken Seite unterzubringen und die entsprechenden Buttons im selben Design wie die Überschriftsleiste zu gestalten. Damit die einzelnen Buttons und die Überschrift auf der Nutzeroberfläche nicht unkoordiniert wirken, wurde ein verbindendes Element erstellt. Dieses sollte nicht zu starr, aber auch nicht zu dominant ausfallen. Die Wahl fiel auf einen geschwungenen Balken, der die Überschrift mit den Buttons verbinden soll.

Die Filmsteuerung ist dezent und nicht zu dominierend in die Benutzeroberfläche zu integrieren. Sie muss jederzeit zu sehen und zu erreichen sein, da sämtliche Animationen damit gesteuert werden.

3.2.3 Aufbau

Für eine Anwendung ist es entscheidend, auf welchen Systemen sie installiert und zu welchem Zweck sie verwendet werden soll. Daraus ergeben sich bestimmte Vorgaben für den Aufbau und die Dimensionen der Oberfläche. In diesem Fall ist es eine Präsentationsoberfläche für vorlesungsbegleitende Animationen. Das bedeutet, dass die Anwendung vorwiegend von Laptops aus gesteuert wird. Diese haben in der Regel eine maximale Auflösung von 1280 x 800 Pixel. Demzufolge darf die Anwendung nicht größer sein, weshalb man der Standardauflösung von 1024 x 768 Pixel als Abmessungen für die Nutzeroberfläche den Vorzug gab. Diese Werte sind ebenfalls als Auflösungen für 4:3 Monitore gedacht. Somit tritt keine Verzerrung oder ein Anzeigeverlust auf.

Weiterhin muss bedacht werden, dass auf der Anwendung ausreichend Platz vorhanden ist, um auch die Animationen einzubinden. Dabei ist darauf zu achten, dass ein gutes Verhältnis zwischen Navigation und Animation bestehen bleibt. Die Navigation darf nicht zu groß werden, da sonst der Rahmen für die Anzeigefläche zu gering wird. Ebenso wenig sollte die Navigation zu klein werden, weil die Lesbarkeit der Inhalte nicht mehr gegeben wäre.

3.2.4 Bildrate der Nutzeroberfläche

Ein entscheidendes Kriterium für den Aufbau ist die Bildrate. Wie in den Grundlagen bereits erwähnt, wird in der Filmtechnik eine Bildrate von 24 Bildern für eine Sekunde Film angesetzt. Da die Einteilung der Zeitleiste in Flash in Fünferschritten erfolgt, erhöhte man sich die Framerate auf 25 Bilder pro Sekunde. Dies ermöglicht eine leichtere Bearbeitung bei sich wiederholenden Animationssequenzen. Eine niedrigere Einstellung würde zu einer ruckartigen Animation führen. Sehr wichtig ist, dass die Nutzeroberfläche und die Animationen die gleiche Bildrate haben, da es sonst zu unterschiedlichen Ablaufgeschwindigkeiten bei den Animationen kommt.

3.2.5 Farben

Ein wichtiger Aspekt einer multimedialen Präsentation sind die Farben. Diese dürfen in keinem Fall zu aufdringlich oder schrill wirken, um nicht von den Lehrinhalten abzulenken. Die Entscheidung fiel auf die Verwendung der Farben des Corporate Design der Hochschule Mittweida.

Das Design besteht hier überwiegend aus zwei verschiedenen Blautönen. Die RGB-Werte für die entsprechenden Objekte befinden sich in der unten stehenden Tabelle. Des Weiteren werden die Farben vom Kontrast her so gewählt, dass eine Lesbarkeit gewährleistet ist. Für die Navigation und die Überschriften bedeutet das eine dunkle Schrift auf hellem Grund. Der stärkste Kontrast entsteht dabei mit Schwarz als Schriftfarbe.


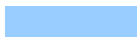
| Objekt | Rot | Grün | Blau | Farbe |
|------------------------|-----|------|------|---|
| Führungsleiste/Schrift | 51 | 102 | 153 |  |
| Buttons | 153 | 204 | 255 |  |

Tabelle 7: Farbtabelle für die Benutzeroberfläche

Eine Abbildung der fertigen Nutzeroberfläche mit den verwendeten Farben befindet sich in Abschnitt 4.1.3.1.

3.4 Vorüberlegungen zu den Animationen

In diesem Abschnitt werden die Gedanken des Autors aufgezeigt, welche vor Beginn der praktischen Arbeit angestellt wurden. Es erfolgt eine Vorstellung von Überlegungen zu Platzaufteilung, Storyboards und Farben. Die Konzepte werden erst allgemein, für alle Animationen gültig, verfasst und danach im Detail an den Beispielanimationen dargestellt.

3.4.1 Platzaufteilung

Sehr wichtig bei der Planung einer Animation ist die Platzaufteilung. Da der verfügbare Platz in den meisten Fällen sehr begrenzt ist, muss man sich über die genaue Aufteilung und Größe von Animation und anderen Elementen Gedanken machen.

In den vorliegenden Animationen wurde entschieden, die Anzeigefläche in drei grobe Abschnitte zu unterteilen. Im Detail bedeutet dies, dass im linken oberen Drittel die eigentliche Animation angeordnet ist. Direkt darunter wird die Befehlsbox platziert, welche jedoch nicht in jeder Animation benötigt wird. Im rechten verbliebenen Drittel ist die Informationsbox untergebracht, da sich die Navigation auf der linken Seite der Anwendung befindet und dies nicht zu einem optischen Konflikt mit der Navigationsleiste führt.

3.4.2 Bildrate der Animationen

Die Bildrate der Animationen ist ebenso wichtig wie die der Anwendung. Wenn sich die Rate von Oberfläche und Animation unterscheidet, kann es zu unschönen Nebeneffekten kommen. Eine zu geringe Bildrate würde eine ruckartige Animation hervorbringen. Eine zu hohe ist ebenfalls nicht sinnvoll, da das menschliche Auge den Unterschied nicht mehr wahrnehmen und sich der Speicherbedarf erhöhen würde.

Aus diesen Gründen verwendet man bis heute in der Filmtechnik und der Animation im allgemeinen eine Bildrate von 24 Bildern pro Sekunde. Wie bereits unter 3.2.3 erwähnt, hat man sich für die Erstellung der Oberfläche und Animationen unter Flash dazu entschieden, 25 Bilder für eine Sekunde zu benutzen. Dieselbe Bildrate ist auch in der Vorlagendatei enthalten. Das bedeutet, dass diese Einstellung für zukünftige Projekte einheitlich ist.

3.4.3 Storyboards und interner Aufbau

Der Ablauf der gewünschten Animationen muss im Voraus geplant werden. Zu diesem Zweck ist es dringend empfehlenswert, Storyboards anzufertigen, welche die Anordnung einzelner Komponenten innerhalb der Animationen und die Animationen in bestimmten Phasen verdeutlichen. Ein Storyboard ist eine grobe Skizze der gezeigten Szene. Es muss daher nicht akkurat gezeichnet sein. Es ist ebenso entscheidend, dass bei thematisch ähnlichen Animationen auch eine vergleichbare Art der Darstellung gewählt wird. Dadurch vereinfacht sich der Lernprozess.

Nachdem man nach ausreichenden Recherchen die technischen Abläufe für den PIO und DMA Mode erarbeitet hatte, fiel die Entscheidung auf die Darstellung als Blockschaltbild. Dieses ist in der Elektrotechnik und Elektronik sehr verbreitet und bietet eine gute und einfach strukturierte Übersicht. Für folgende Animationen zu anderen Themen müssen unter Umständen neue Darstellungsformen gewählt werden, um den entsprechenden Inhalt angemessen wiederzugeben. Auf den kommenden Seiten werden die einzelnen Storyboardbilder gezeigt und Erklärung zu deren Inhalten gegeben.

3.4.3.1 PIO Mode

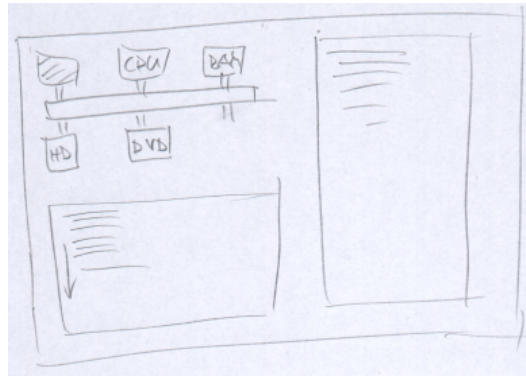


Abbildung 5: Storyboard Übersicht PIO Mode

Abbildung 5 zeigt den groben Überblick über den geplanten Aufbau der Animation zum PIO Mode. In der linken oberen Ecke ist die schematische Darstellung des Rechnersystems abgebildet. Diese soll während des Ablaufs animiert dargestellt werden. Am rechten Rand ist die Infobox zu erkennen, in welcher eine Kurzbeschreibung der aktuellen Animation zu lesen ist. Unten links soll die Befehlsbox für die CPU Befehle positioniert werden. Der Pfeil innerhalb dieser Box soll den geplanten Scrollvorgang der einzelnen CPU Befehle symbolisieren.

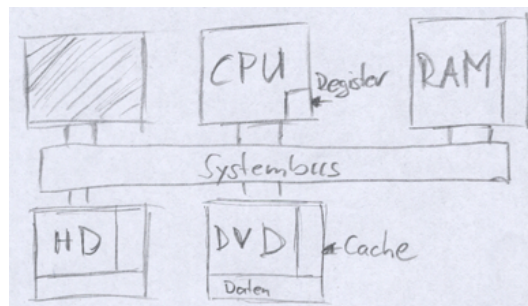


Abbildung 6: Storyboard PIO Mode - Detailansicht

In Abbildung 6 ist der detaillierte Aufbau der PIO-Mode-Animation dargestellt. Wie man erkennen kann, ist die CPU in der Mitte angeordnet, da sie das Zentrum eines jeden Rechners darstellt. Die Festdatenspeicher Festplatte (HD) und DVD-Laufwerk (DVD) sind unterhalb platziert; sie gehören zur gleichen Systemgruppe. Der dynamische Arbeitsspeicher (RAM) ist neben der CPU angeordnet, da er zu den internen Bausteinen eines Computersystems zählt. Am rechten Rand jeder Systemkomponente befindet sich ein Balken, welcher später für die entsprechenden Speicherfüllstände bzw. Register vorgesehen ist. Links oben befindet sich noch ein Platzhalter, um den Unterschied zum DMA Mode besser zu verdeutlichen.

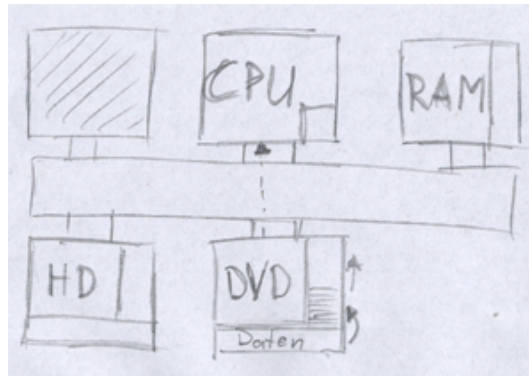


Abbildung 7: Storyboard PIO Mode - Cache laden

Die oben stehende Abbildung 7 zeigt den Transfer der Daten von der DVD in den internen Cache des DVD-Laufwerks. Der senkrechte Pfeil symbolisiert dabei den steigenden Füllstand des Laufwerkspuffers. Nachdem Füllen des Cache, wird ein Interruptsignal an die CPU gesendet.

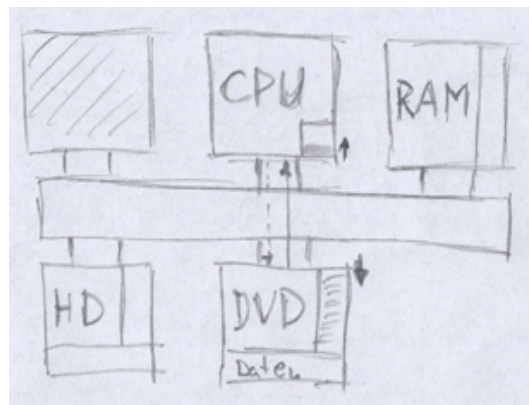


Abbildung 8: Storyboard PIO Mode - Transfer Cache → CPU-Register

Wie in Abbildung 8 zu sehen ist, unterbricht die CPU den laufenden Prozess und beginnt die Daten wortweise aus dem Cache in das interne CPU-Register zu übertragen. Dabei ist der durchgängige Pfeil für den Datentransfer und der gestrichelte Pfeil für die Adress- und Steuersignale gedacht. Da die Daten aus dem Cache gelesen werden, muss die Füllstandsanzeige gleichzeitig natürlich etwas verringert werden.

In der Befehlsbox werden zu diesem Zeitpunkt die ersten Befehle zu sehen sein. Für jeden Vorgang gibt es einen Befehlsholezyklus (Instruction Fetch) und einen Befehlsausführungszyklus (Execute). Der Befehl für die oben stehende Abbildung ist „mov reg, dvd“. Dies ist jedoch nur die vereinfachte Darstellung des Befehls, um das Prinzip dahinter zu verdeutlichen. In der Maschinensprache würden an Stelle von „reg“ und „dvd“ die entsprechenden Speicherzellen im Register bzw. im Cache des DVD-Laufwerks stehen. Die Reihenfolge der Speicherzellen bei den Befehlen ist immer „mov Ziel, Quelle“.

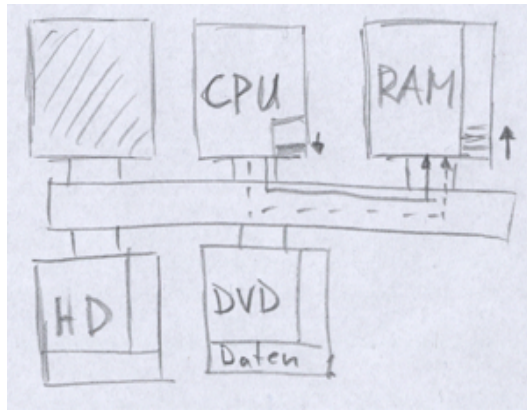


Abbildung 9: Storyboard PIO Mode -
Transfer CPU-Register → RAM

Nachdem das erste Datenwort im CPU-Register gespeichert wurde, sendet die CPU die Daten in den RAM. Die kleinen Pfeile symbolisieren auch hier wieder den steigenden oder fallenden Füllstand.

Für die Befehlsbox gibt es hier ebenfalls zwei Zyklen, welche erneut der Befehlshole- und der Befehlsausführungszyklus sind. Der Befehl selbst lautet für diesen Vorgang „mov ram, reg“.

Die Vorgänge aus den Abbildungen 8 und 9 werden so lange wiederholt, bis sämtliche Daten in den RAM übertragen wurden. Es ist geplant, diese Sequenz zum besseren Verständnis in der Animation als Schleife anzulegen.

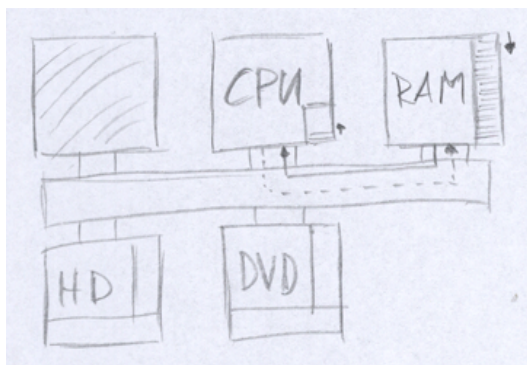


Abbildung 10: Storyboard PIO Mode -
Transfer RAM → Register

Nachdem sämtliche Daten in den RAM übertragen worden sind, beginnt der CPU-gesteuerte Transfer auf die Festplatte. Dabei werden die Daten ebenfalls wortweise aus dem Arbeitsspeicher in das Register der CPU geschrieben. Der Befehl dafür ist „mov reg, ram“.

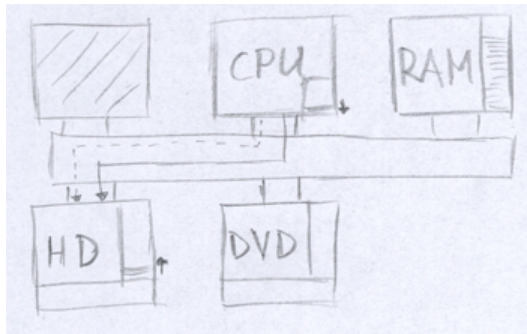


Abbildung 11: Storyboard PIO Mode - Transfer CPU-Register → HD

Vom Register der CPU aus werden die Daten in den Cache der Festplatte transferiert und von da auf die Festplatte geschrieben.

Der Befehl für diesen Transfer lautet „mov hd, reg“.

Die Vorgänge aus den Abbildungen 10 und 11 werden ebenfalls solange wiederholt, bis der RAM komplett geleert worden ist. Dieser Teil der Animation wird auch wieder als Schleife konzipiert, da die Befehle in recht schneller Folge angezeigt werden.

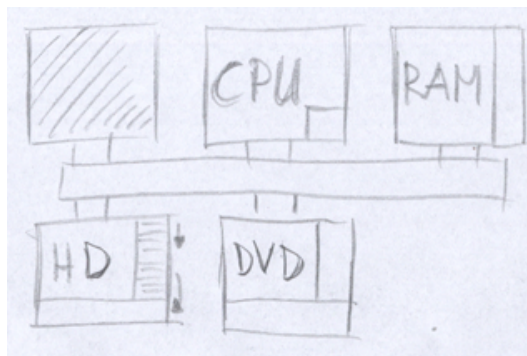


Abbildung 12: Storyboard PIO Mode - HD-Cache leeren

Nachdem die letzten Daten in den Cache der Festplatte übertragen worden sind, nimmt die CPU den unterbrochenen Prozess wieder auf.

3.4.3.2 DMA Mode

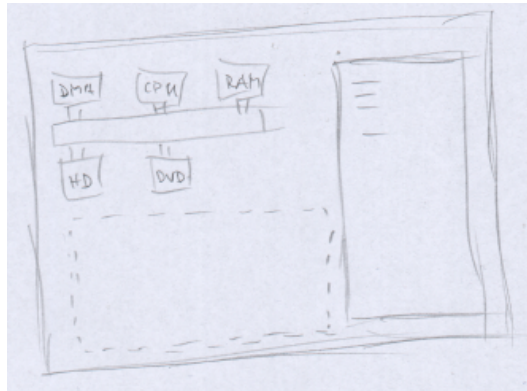


Abbildung 13: Storyboard Übersicht
DMA Mode

Abbildung 13 zeigt die Platzaufteilung der Animationselemente für den DMA Mode. Vom Grundaufbau her ist es dasselbe wie beim PIO Mode. Der Unterschied besteht nur im Fehlen der Befehlsbox, welche hier gestrichelt dargestellt ist.

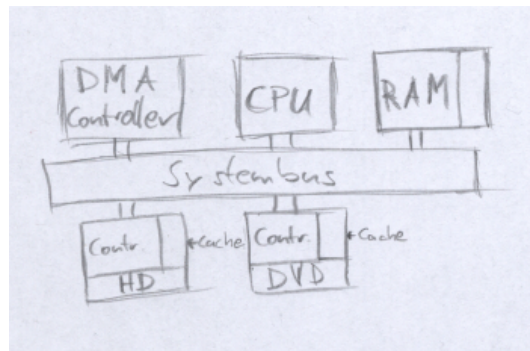


Abbildung 14: Storyboard DMA Mode -
Detailansicht

In der oben stehenden Abbildung ist der detaillierte Aufbau der Animation für den DMA Mode dargestellt. Oberhalb des Systembusses sind von links nach rechts der DMA-Controller, die CPU und der RAM angeordnet. Unterhalb des Busses werden die Festplatte (HD) und das DVD-Laufwerk angezeigt. Die leeren Balken an den rechten Seiten von RAM, HD und DVD stellen den Cache der Komponenten dar.

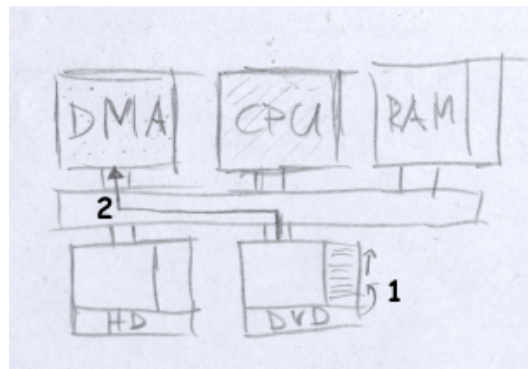


Abbildung 15: Storyboard DMA Mode - Cache laden

In Abbildung 15 sieht man den Beginn des DMA Mode. Zuerst werden die Daten von der DVD in den Cache des Laufwerks übertragen (1). Ist der Cache voll, sendet das Laufwerk einen DMA Request an den DMA-Controller (2). Der gepunktete DMA-Controller symbolisiert in diesem Fall eine Inaktivität. Die CPU hingegen stellt mit ihrer Streifung den aktiven Zustand dar.

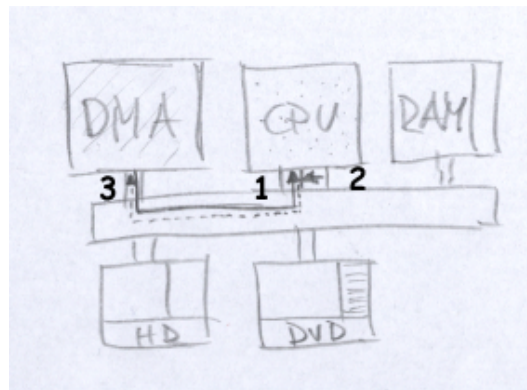


Abbildung 16: Storyboard DMA Mode - Request Systembus

Der DMA-Controller sendet einen HOLD-Request zur alleinigen Nutzung des Datenbusses an die CPU (1). Daraufhin trennt sich die CPU vom Datenbus (2) und sendet einen HOLD-Acknowledge-Befehl an den DMA-Controller (3). Danach wechselt der aktive Zustand auf den DMA-Controller und der inaktive auf die CPU.

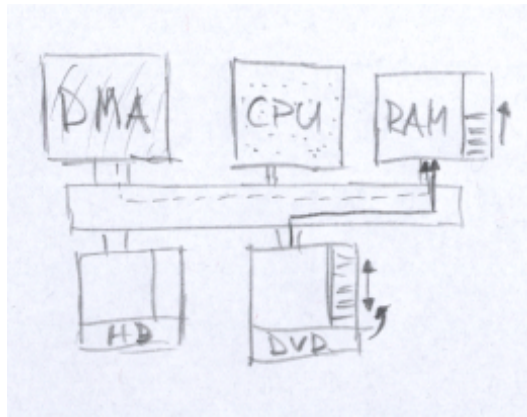


Abbildung 17: Storyboard DMA Mode - Transfer DVD-Cache → RAM

Nachdem der DMA-Controller die Buskontrolle erhalten hat, beginnt die Übertragung der Daten aus dem Cache des Laufwerks in den Arbeitsspeicher des Systems. Dabei steuert der DMA-Controller sämtliche Adressierungsvorgänge. Der beidseitige Pfeil symbolisiert hier das fortlaufende Füllen und Entleeren des Laufwerkscaches. Der gestrichelte Pfeil stellt die Adressvorgänge dar, während der durchgezogene Pfeil den Datentransfer symbolisiert.

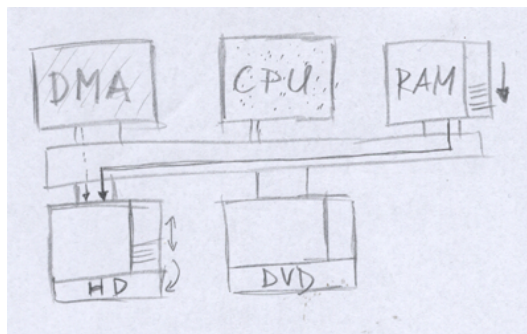


Abbildung 18: Storyboard DMA Mode - Transfer RAM → HD-Cache

Nachdem sämtliche Daten in den RAM übertragen wurden, beginnt die DMA-Controller-gesteuerte Übertragung auf die Festplatte. Der beidseitige Pfeil in Abbildung 18 symbolisiert auch hier das stetige Füllen und Entleeren des Festplattencaches.

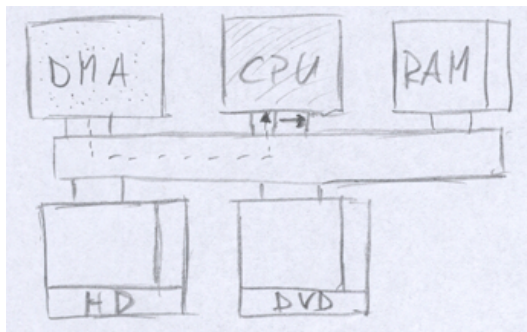


Abbildung 19: Storyboard DMA Mode - Freigabe Systembus

Wenn die Datenübertragung abgeschlossen ist, übergibt der DMA-Controller die Bussteuerung

wieder an die CPU. Der aktive Zustand wechselt nun wieder von dem DMA Controller auf die CPU (Farbwechsel).

3.4.4 Farben

Die Welt ist nicht schwarzweiß. Deshalb werden auch in Animationen unterschiedliche Farben verwendet. Die Wahl der Farben ist von entscheidender Bedeutung für das Verstehen der Animationen. Aus diesem Grund muss darauf geachtet werden, dass weder zu schrille noch zu blasse Farben gewählt werden. Zu schrill würde unter Umständen bedeuten, dass die Aufmerksamkeit auf einen falschen Punkt gelenkt wird. Eine zu blasser Farbe kann in bestimmten Fällen das genaue Gegenteil bewirken. Hier besteht dann die Gefahr, dass einem bestimmten Element zu wenig Aufmerksamkeit beigemessen wird.

Für jede neue Animation muss berücksichtigt werden, welche Farben bei der Nutzeroberfläche Verwendung fanden.

Im Falle der PIO- und DMA-Mode-Animationen wurde die Informationsbox der Einheitlichkeit wegen mit demselben Hellblau, wie jenes aus dem Design der Hochschule, ausgestattet.

Das Herzstück eines jeden Computers ist die CPU und somit sehr wichtig. Deshalb wurde für dieses Objekt die Farbe Rot gewählt. Rot gilt im Allgemeinen als Signalfarbe oder um etwas Wichtiges hervorzuheben.

Für den DMA-Controller gilt etwas Ähnliches. Wenn er aktiv ist, übernimmt er die Steuerung des Datentransfers, deshalb ist auch dafür eine kräftige Farbe zu wählen. Es wurde ein leuchtendes Violett gewählt. Bei den inaktiven Varianten der CPU- und des DMA-Controllers rückt die Wichtigkeit in den Hintergrund. Aus diesem Grund wurde die Farbe verändert. Über die jeweiligen Objekte ist eine Graufäche mit denselben RGB-Werten wie das Leerfeld und einem Alphawert von 70% gelegt worden, um einen blässeren Farbton zu erzeugen

Das Leerfeld dient als Platzhalter für verschiedene Komponenten und sollte daher nicht in den Vordergrund gerückt werden. Ein grauer Farbton steht immer für etwas Unwichtiges.

Für Datenspeicher jeglicher Art (Festplatte, DVD-ROM, RAM) wurde eine einheitliche Farbe gewählt. In diesem Fall die Farbe Gelb. Da ein zu grelles Gelb zu viel Aufmerksamkeit auf sich ziehen würde, hat man sich für ein blässeres Gelb entschieden.

Die Farbe der Cachefüllstandsanzeigen ist dem Ladebalken der meisten Microsoft-Windows-Versionen entnommen, um beim Betrachter sofort eine Assoziation zu einem Fortschrittsbalken herzustellen. Es handelt sich hierbei um ein dunkles Blau.

Der Systembus muss sich von allen anderen Komponenten abheben, deswegen wurde für diesen ein sehr helles Blau gewählt. Es ist jedoch ein anderes Blau, als jenes für die Informationsbox, um den

Diese sind in den jeweiligen Farben der Quelle gehalten um eine Zuordnung zu erleichtern. Der Pfeil für die transportierten Daten ist Blau wie die Füllstandsanzeige und der Pfeil für die Adress- und Steuersignale ist in dem selben Rot gehalten wie die CPU. Eine Ausnahme bilden die Steuerpfeile beim DMA Controller. Diese sind aufgrund der Einheitlichkeit und der gleichen Aufgabe ebenfalls Rot.

4. Realisierung

Im vorangegangenen Kapitel wurde ausführlich die Konzeption der Anwendung und der Animationen erläutert. Kapitel vier befasst sich mit der detaillierten Umsetzung der Nutzungsoberfläche sowie der beiden Beispielanimationen zum DMA und PIO Mode. Es werden die Probleme und deren Lösungsansätze erläutert, welche bei der Erstellung aufgetreten sind.

4.1 Die Benutzeroberfläche

Dieser Abschnitt befasst sich mit der Realisierung der Benutzeroberfläche. Dabei wird insbesondere auf das ActionScript Wert gelegt. Es erfolgt ebenfalls die Erläuterung der einzelnen Verfahren zur Erstellung bestimmter Objekte und eine Untermalung mit Screenshots.

4.1.1 Grafische Erstellung der Benutzeroberfläche

Zu Beginn der Arbeit wurde die grafische Benutzeroberfläche erstellt, wobei primär Photoshop 7 Verwendung fand. Als erstes erfolgte die Festlegung der Dimensionen der Oberfläche, wobei für diese Anwendung 1024 x 768 Pixel gewählt wurden (vgl. 3.2.3). Als Platzhalter für die Animationen entschied man sich, eine Fläche von 725 x 600 Pixel zu reservieren. Dadurch erzielte man ausreichend Platz für Navigation und Überschriften.

Die maximale Breite für die Navigation beträgt jetzt 299 Pixel. Da es allerdings angebracht ist, links und rechts der Navigationsbuttons etwas Platz zu lassen, wurde für die Navigation eine Breite von 190 Pixeln festgelegt. Damit ist die Breite der Buttons ausreichend für unterschiedliche Textinhalte. Auf die genaue Erstellung und den Aufbau der Buttons wird im Abschnitt Navigation präziser eingegangen.

Die Erstellung der Buttons und des Hintergrunds geschah auf unterschiedlichen Ebenen. Ursprünglich wurde auch die „Dozentenleiste“ in Photoshop vorgefertigt. Durch die dynamische Generierung der Informationen dieser Anzeige entfernte man diese Option später wieder.

4.1.2 Aufbau der Menü-XML-Datei

Der strukturelle Aufbau der XML-Datei ist von entscheidender Bedeutung für die Einfachheit der Gestaltung. Je weniger Unterstrukturen vorhanden sind, desto übersichtlicher ist die gesamte Datei. Am Anfang der XML-Datei steht immer die Dokumententypdefinition. Diese sieht wie folgt aus:

```
<?xml version='1.0' encoding='utf-8'?>
```


Die UTF-8 Codierung bedeutet, dass innerhalb der XML-Datei sämtliche Unicodezeichen dargestellt und weiterverarbeitet werden können.

Es muss immer ein Wurzelement vorhanden sein, darunter können die Elemente verschachtelt sein. Bei dieser Anwendung ist die Verschachtelung nicht notwendig, da es den Verarbeitungsprozess unnötig verkomplizieren könnte. Das Wurzelement dieser XML ist `<menu>`. Sämtliche Elemente der XML-Datei haben leicht zu identifizierende Namen, um die Bearbeitung zu erleichtern.

Danach folgt eine Trennzeichendeklaration für die Pfadelemente. Man hat sich für den Backslash als Trennzeichen entschieden, da Flash bzw. ActionScript bei einem normalen Slash Probleme verursachen könnte. Es würde z.B. bei „/n“ eine neue Zeile starten, was allerdings nicht beabsichtigt wäre. Als Variable für das Trennzeichen ist der Einfachheit halber „tz“ gewählt worden.

Als nächstes kommt es zum Laden einzelner Objekte der Struktur, was in diesem Fall die Überschrift für die entsprechende Lehrveranstaltung, der jeweilige Dozent und dessen Fachbereich wären. Hierfür wurden die Variablen „hl“, „doz“ und „fach“ festgelegt. Diese dienen in der Anwendung später zur eindeutigen Identifizierung.

Zum Schluss werden noch die Links für die Buttons und somit die Verzeichnisstruktur definiert. Dafür findet die Variable „pfad“ Verwendung.

4.1.3 Funktionalitäten der Benutzeroberfläche

Der folgende Abschnitt erläutert die Entstehung der speziellen Funktionen der Nutzeroberfläche. Zu diesem Zweck wurden einige Screenshots angefertigt und Scriptauszüge im Detail beschrieben.

4.1.3.1 Überblick

Um dem Leser die folgenden Abschnitte zu erleichtern, möchte der Autor an dieser Stelle die Nutzeroberfläche zunächst im Detail vorstellen. In Abbildung 20 ist die Oberfläche für das Fach Computertechnik zu sehen.

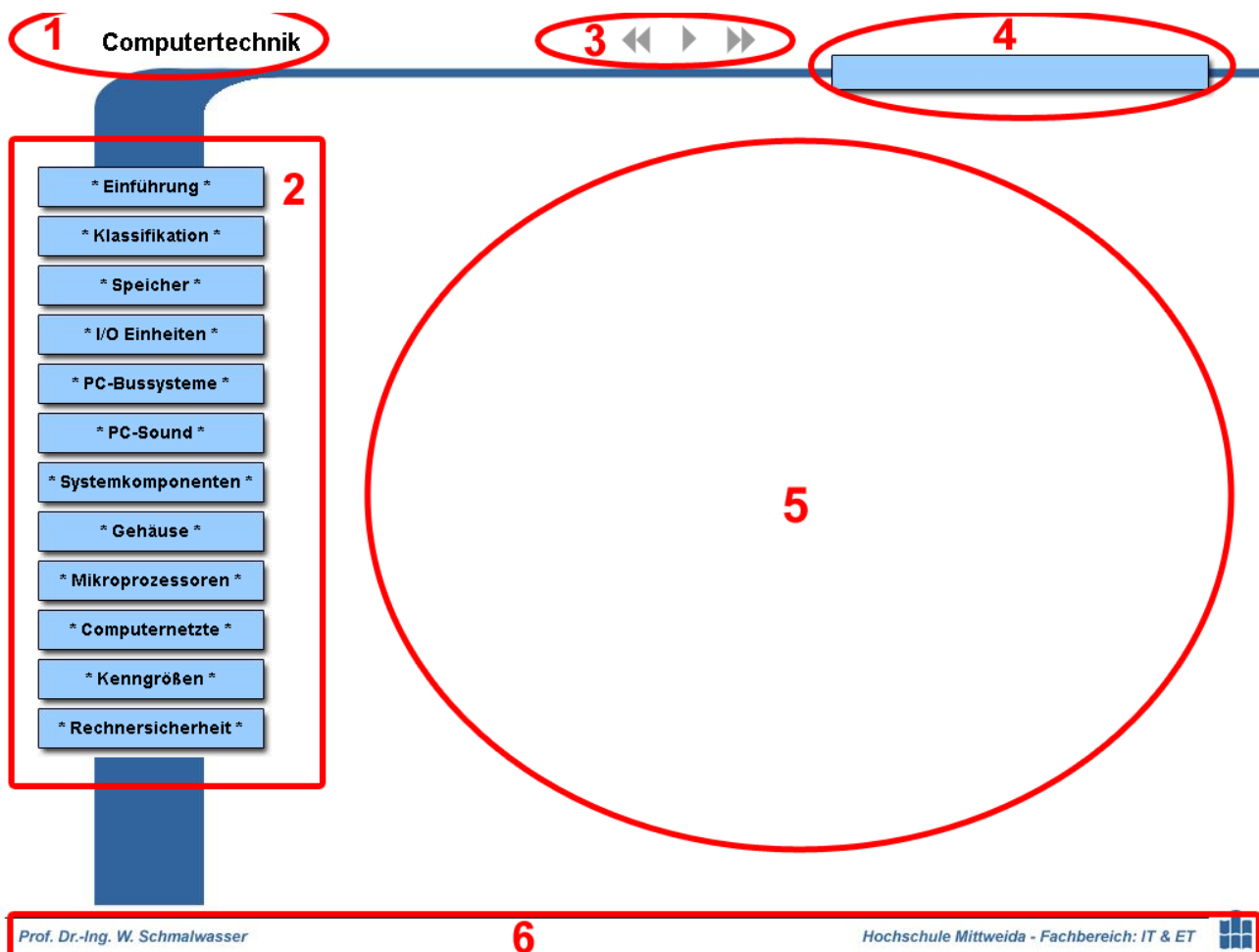


Abbildung 21: Übersicht über die Präsentationsoberfläche

Es befinden sich verschiedene Elemente auf der Oberfläche, welche zum größten Teil über XML generiert werden. Dies wären die Hauptüberschrift (1), die Navigation (2), die Themenüberschrift (4) und die „Dozentenleiste“ (6), welche den jeweiligen Dozenten, dessen Fachbereich und die zugehörige Lehranstalt anzeigt. Außerdem befindet sich am oberen Rand die Steuerung (3) für die Animationen, die in der leeren Fläche (5) der Abbildung angezeigt werden.

Die komplette Nutzeroberfläche befindet sich in der Datei menu fla. Sie besteht aus fünf einzelnen Frames. Im ersten Frame werden im ActionScript die globalen Variablen festgelegt und die XML-Daten eingelesen (siehe 4.1.3.2).

Im Folgenden werden die verwendeten Variablen und deren Bedeutung aufgeführt. Zuerst erfolgt die Erstellung eines leeren MovieClip-Objekts, in welchem später der Playlistcontainer geladen wird. Für den Vorgang findet der „createEmptyMovieClip“-Befehl Anwendung, welcher für die dynamische Erstellung von MovieClips zuständig ist. Der Container ist für die Darstellung der Animation erforderlich.

```
var Content:MovieClip = this.createEmptyMovieClip("Content",
this.getNextHighestDepth());
```

Das erstellte Objekt wird nun an die gewünschte Position verschoben, um das Verdecken der Navigation oder der Überschriften durch die Animationen zu verhindern.

```
Content._x = 250; Content._y = 100;
```

„_x“ steht für den horizontalen Wert und „_y“ für den vertikalen.

Als nächstes wird der Container für die Navigation angelegt. Dieser ist ebenfalls ein leeres MovieClip-Objekt. Damit das Menü immer auf der nächsthöheren Ebene der Anwendung abgelegt wird, findet die Methode „getNextHighestDepth“ Anwendung.

```
var Menuleiste:MovieClip=this.createEmptyMovieClip("Menuleiste",  
this.getNextHighestDepth());
```

Die folgende Variable ist für die Navigation zuständig und wird zu Beginn leer erstellt. In diesem Fall wäre das die Anfangsstruktur für das Menü.

```
var MenuPfad = "";
```

Als nächstes muss ein Array (ein Feld) für die Gesamtheit der XML-Elemente angelegt werden. Aus diesen extrahiert man später die entsprechenden Pfade.

```
var Elemente = new Array();
```

Die nun aufgeführten Variablen sind für die Steuerung der Playlisten notwendig.

Zuerst wird die zur späteren Identifizierung einer Playliste benötigte Variable erstellt.

```
var Playlistpfad;
```

Als nächstes folgt die Variable für die aktuelle Animationsnummer in der Playliste.

```
var Nummer:Number;
```

Danach schließt sich die Kontrollvariable an, welche zur Überprüfung dient, ob die Playliste durchlaufen wurde oder nicht. Zu Beginn setzt man diese auf „false“, da zum jetzigen Zeitpunkt noch keine Liste abgearbeitet worden ist.

```
var Fertig:Boolean = false;
```

Die folgende ist eine Kontrollvariable, welche überprüft, ob eine Animation gerade abläuft oder ob sie gestoppt ist. Auch diese Variable setzt man zu Anfang auf „false“, da ebenfalls noch keine Animation abgespielt wurde.

```
var PlayYN:Boolean = false;
```

Als nächstes wird eine Variable benötigt, welche zur Überprüfung eines Schleifenabbruchs notwendig ist. Sie muss zu Beginn auf „true“ gesetzt werden. Während eines Schleifendurchlaufs ist diese Variable auf „false“ zu setzen. Sobald man jedoch die Taste zum Abbruch der Schleife betätigt, wird sie wieder auf „true“ gesetzt.

```
var Schleife:Boolean = true;
```

Am Schluss erfolgt noch die Erstellung eines KeyListener-Objekts, welches später die Steuerung per Tastaturbefehle ermöglichen soll.

```
var KeyListener:Object = new Object();
```

Von Frame zwei bis vier ist ein Zeitpuffer für langsamere Rechner eingefügt worden. Dies ist zu empfehlen, da es sonst passieren könnte, dass die „Animation“ weiter läuft, obwohl die Daten aus der XML-Datei noch nicht vollständig eingelesen wurden. Der fünfte Frame dient der XML-Datenverarbeitung. Hier erfolgt die Umwandlung der Pfade aus der XML-Datei mittels ActionScript in entsprechende Verzeichnisstrukturen und Buttons. Ein späteres Kapitel erläutert diesen Vorgang ausführlich.

Um die Anwendung nutzen zu können, muss sie vorher kompiliert werden. Dies geschieht über die „Film testen“-Funktion von Flash. Dabei wird eine SWF-Datei erstellt, die auf jedem System, welches den Flash Player installiert hat, ausführbar ist.

4.1.3.2 Einlesen der XML-Daten

Es war bekannt, dass sich Flash mit XML Daten kombinieren lässt. Das Problem bei der Erstellung bestand darin, wie sich XML und Flash miteinander verknüpfen lassen. Nach längeren Recherchen tat sich die Lösung in einem Flashforum [11] auf.

Das Einlesen der XML-Daten in die Menüdatei ist der essentielle Bestandteil der Anwendung. Dieser Vorgang erfolgt über ActionScript zum einen in der Datei menu fla und zum anderen in der Datei playlistcontainer fla.

Als erstes muss ein leeres XML-Objekt für Flash erstellt werden. Dies geschieht über die folgende Befehlszeile:

```
var XMLDaten = new XML();
```

„XMLDaten“ ist dabei der Name der Variablen für das XML-Objekt.

Danach wird festgelegt, dass ActionScript alle Leerzeichen und Tabulatoren ignorieren soll. Selbige sind jedoch zur besseren Übersichtlichkeit innerhalb der XML-Datei notwendig. Die Scriptzeile dafür sieht wie folgt aus:

```
XMLDaten.ignoreWhite=true;
```

Als nächstes muss angegeben werden, welche XML-Datei zu laden ist. Dies erfolgt über den Befehl `XMLDaten.load("menu.xml");` für die Datei menu.xml.

Die nachfolgenden Funktionen und Schleifen sollen nur ausgeführt werden, wenn das Laden der

XML-Daten erfolgreich war. Dazu ist die Ereignisprozedur `onLoad` erforderlich. Der vollständige Befehl für den Test des erfolgreichen Ladens sieht folgendermaßen aus:

```
XMLDaten.onLoad = function(success).
```

„`success`“ ist eine Booleanvariable, was bedeutet, dass die folgende Funktion nur auszuführen ist, wenn diese den Wert „`true`“, also „wahr“, besitzt.

Zunächst muss die Gesamtanzahl der in der XML-Datei vorhandenen Elemente bestimmt werden. An dieser Stelle bezieht man das Trennzeichen, die Dozentenangaben, die Fachüberschrift und die Pfadangaben mit ein. Um die Anzahl zu bestimmen, wird die „`length`“-Eigenschaft der „`XMLDaten`“-Variable ausgelesen. Das Ergebnis wird in der Variable „`AnzahlElemente`“ gespeichert.

```
AnzahlElemente = XMLDaten.firstChild.childNodes.length;
```

Als nächstes werden mittels einer „`for`“-Schleife die einzelnen Elemente identifiziert und sortiert. Dabei soll die Schleife solange durchlaufen, wie es Elemente in der XML-Datei gibt. Der Maximalwert dafür wurde zuvor in der Variable „`AnzahlElemente`“ gespeichert.

```
for (var i = 0; i < AnzahlElemente; i++)
```

Die Zählvariable ist „`i`“, welche man bei jedem Schleifendurchlauf um eins erhöht.

Zuerst werden die einmaligen Elemente heraus gefiltert. Das wären in diesem Fall das Trennzeichen, die Überschrift, der Dozent und der Fachbereich/Hochschule (vgl. 4.2.3.4 Die Dozentenleiste).

Jetzt erweitert man das „Element“-Array als zählbares Objekt, um Eigenschaften für die einzelnen Pfade bzw. Buttons hinzuzufügen. Die Nummerierung erfolgt dabei über eine Zählvariable, welche unabhängig von der Schleife zählt, da noch nicht bekannt ist, wie viele Objekte sich innerhalb der XML-Datei befinden. Als nächstes wird das Attribut „`pfad`“ aus der XML-Datei in die Flashvariable „`Pfad`“ eingelesen.

```
Pfad = XMLDaten.firstChild.childNodes[i].attributes.pfad;
```

Danach wird der String „`Pfad`“ nach dem letzten darin enthaltenen Trennzeichen durchsucht und seine Position in der Variable „`LastTz`“ gespeichert.

```
var LastTz = Pfad.lastIndexOf(Tz);
```

Der nächste Schritt ist das Bestimmen der einzelnen Buttonbeschriftungen anhand der zuvor eingelesenen Pfadangaben aus der XML-Datei. Für diesen Vorgang wird die Funktion „`substring`“,

welche innerhalb eines Strings einen bestimmten Teil ermittelt, verwendet.

```
Elemente[j].Pfad = Pfad.substring(0, LastTz + 1);
```

Die Werte in den Klammern der Funktion „substring“ geben hierbei die Start- und Endpositionen der Suche an. Die Ziffer „0“ steht dabei für den Anfang und „LastTZ + 1“ für das letzte zu durchsuchende Zeichen des Strings.

Nun müssen die Bezeichnungen für die sogenannten Linkbuttons bestimmt werden, wobei für diesen Vorgang auch hier wieder die „substring“-Funktion Anwendung findet.

```
Elemente[j].Bezeichnung = Pfad.substring(LastTz + 1, Pfad.length);
```

In diesem Fall bedient man sich der Positionen „letztes Trennzeichen + 1“ und „Gesamtlänge des Strings“.

Als letztes müssen die Verknüpfungen der Linkbuttons aus der XML-Datei eingelesen werden, welche über die Attribute „link“ definiert sind.

```
Elemente[j].link = XMLDaten.firstChild.childNodes[i].attributes.  
link;
```

Zum Schluss ist die Zählvariable „j“ noch zu erhöhen, damit die einzelnen Arrayelemente nicht überschrieben werden.

Die Weiterverarbeitung der eingelesenen Daten erfolgt in Frame 5 der Datei „menu fla“, welches im folgenden Abschnitt beschrieben wird.

Der komplette Quellcode ist im Anhang unter Script 1 zu finden.

4.1.3.3 Navigation

Die Navigation ist das Hauptelement der Nutzeroberfläche. Dafür wurde mittels Photoshop ein Button erstellt, welcher für drei verschiedene Zustände verfügbar ist. Diese wären die Standardanzeige (1), das Drücken des Buttons (2) und für die Aktivierungsbuttons der „Markiert“-Zustand (3). Als Standardanzeige wurde ein einfacher Button erstellt und mit einem Schlagschatten versehen um das Menü plastischer wirken zu lassen. Die Maße inklusive Schatten betragen 40 Pixel in der Höhe und 190 Pixel in der Breite. Der sichtbare Button hat Abmessungen von 33 Pixel in der Höhe und 183 Pixel in der Breite. Der gedrückte Button ist mit dem sichtbaren von den Maßen her identisch. Aufgrund der geringeren Größe und des fehlenden Schattens entsteht die Illusion, dass der Knopf gedrückt wurde. Die letzte Möglichkeit ist der Button, wenn er aktiviert bleibt. Diese Option

In der nachstehenden Abbildung sind die drei Zustände der einzelnen Buttons zu sehen.

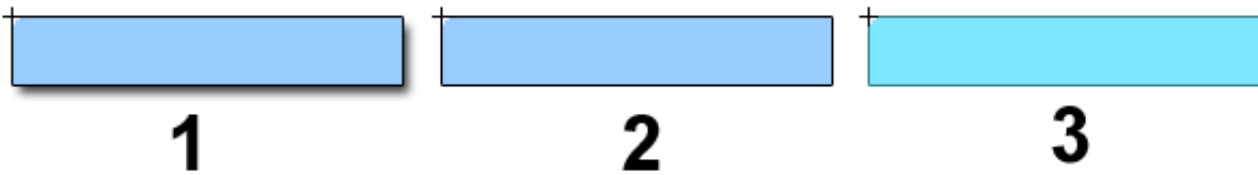


Abbildung 22: Grafische Buttons

Die drei grafischen Elemente wurden in einem separaten MovieClip namens Button1 mit drei Frames kombiniert. Jeder dieser Frames enthält eine Stop-Anweisung, damit die Zustände nicht zufällig wechseln, während die Anwendung läuft. Auf die drei Frames wird später mittels ActionScript zugegriffen um die benötigten Zustände im richtigen Moment darzustellen. Des Weiteren benötigt man ein Textfeld mit der Eigenschaft dynamischen Text darzustellen. Um später den Text für die Buttons diesem Feld zuzuweisen, wurde ihm die Variable „Textfeld“ gegeben.

Die Daten für die Navigation werden komplett aus der XML-Datei eingelesen. Dieser Vorgang erfolgt über das ActionScript in der Datei menu fla. Die entsprechenden Verlinkungen für die Buttons sind in der Datei menu.xml zu finden.

Als erstes wird die Navigationsleiste aus Sicherheitsgründen entfernt, um eventuelle Überlagerungen von älteren Buttons zu vermeiden. Direkt danach erstellt man die Leiste wieder neu, damit die neuen Buttons angezeigt werden können. Die beiden nachstehenden Befehlszeilen bewirken genau dies.

```
Menuleiste.removeMovieClip();  
var Menuleiste:MovieClip = this.createEmptyMovieClip("Menuleiste",  
this.getNextHighestDepth());
```

Die Navigation besteht aus Verzeichnissen, Unterverzeichnissen und den Aktivierungsbuttons. Letztere sind für das Aufrufen der Animationen oder anderer Inhalte, wie Grafiken, nötig. Bei allen drei Arten ist zu Beginn der Animation noch nicht bekannt, wie viele Buttons dafür benötigt werden. Um die Vielzahl an Möglichkeiten effektiv abzudecken, wurden drei Arrays erstellt. Das erste beinhaltet die Aktivierungsbuttons, auch Linkbuttons genannt. Für die Buttons der Verzeichnisse wurde ebenfalls ein Array angelegt. Zuletzt wurde noch ein Array für die grafischen Buttons benötigt. Dies geschieht mit den folgenden drei Befehlen:

```
var Linkbuttons = new Array(); für die Aktivierungsbuttons,  
var Kategorien = new Array(); für die Verzeichnisse/Kategorien und  
var Buttons = new Array(); für die grafischen Buttons.
```

Nun müssen sämtliche aus der „menu.xml“-Datei eingelesene Daten den entsprechenden Arrays

zugewiesen werden. Hierzu ist eine differenzierte Auswertung der Daten erforderlich. Da bereits sämtliche Zeilen zuvor in dem Array „Elemente“ gespeichert wurden, muss dieses jetzt nach spezifischen Kriterien durchsucht werden. Dieser Vorgang wird am einfachsten mittels einer „for“-Schleife realisiert. Diese soll solange zählen, wie es Objekte innerhalb des Elemente Arrays gibt.

```
for (i = 0; i < Elemente.length; i++)
```

Zur einfacheren Weiterarbeit weist man als nächstes den Inhalt der Variable „Element[i].Pfad“ der Variable „ElementPfad“ zu.

```
ElementPfad = Elemente[i].Pfad;
```

Als erstes muss überprüft werden, ob der Inhalt der Variable „MenuPfad“ in der Variable „ElementPfad“ enthalten ist. Um einen Vergleichswert zu besitzen verwendet man die „indexOf“-Methode. Diese gibt die Position des gefundenen Strings zurück.

```
if (ElementPfad.indexOf (MenuPfad) == 0)
```

Da der Inhalt von „MenuPfad“, wenn eine Übereinstimmung gefunden wurde, immer an der Position „0“ des Inhalts von „ElementPfad“ beginnt, gibt „indexOf“ eine „0“ zurück.

Nun muss der Restpfad ermittelt werden. Dafür wendet man die „substring“-Methode auf den vollständigen Elementpfad an.

```
var RestPfad = ElementPfad.substring (MenuPfad.length, ElementPfad.length);
```

Die Start- und Endpositionen bestimmt man aus den Längen der Strings „MenuPfad“ und „ElementPfad“.

Jetzt folgt die Überprüfung, ob weitere Kategorien vorhanden sind, die dem „Kategorien“-Array hinzugefügt werden müssen oder bereits vorhanden sind. Dies kann nur der Fall sein, wenn die Variable „Restpfad“ nicht leer ist.

```
if (RestPfad != "")
```

Nun wird der entsprechende Kategorienname ermittelt und der Variable „Kategorie“ zugewiesen. Dafür benutzt man erneut die „substring“-Methode.

```
var Kategorie = RestPfad.substring (0, RestPfad.indexOf (Tz));
```

Die benötigten Parameter sind die erste Position des Strings „RestPfad“ und die Position des nächsten Trennzeichens innerhalb des Restpfads.

Als nächstes erstellt man die Kontrollvariable „NeueKategorie“ und setzt diese auf „Ja“.

```
var NeueKategorie = "Ja";
```

Diese Variable dient der Überprüfung, ob es sich um eine neue Kategorie oder eine bereits vorhandene handelt.

Nun folgt die eigentliche Prüfung, ob die Kategorie des aktuellen Elements bereits vorhanden ist

oder nicht. Dies realisiert man mit einer „for“-Schleife.

```
for(j = 0; j < Kategorien.length; j++)
```

Die Schleife soll bis zur Anzahl der Elemente im „Kategorien“-Array zählen. Dabei werden alle bereits vorhandenen „Kategorien“-Elemente überprüft.

Innerhalb der Schleife wird die aktuelle Kategorie mit dem „Kategorien“-Element verglichen.

```
if(Kategorie == Kategorien[j])
```

Wenn die beiden Variablen übereinstimmen, setzt man die Variable „NeueKategorie“ auf „Nein“ und die „for“-Schleife wird mittels der „break“-Anweisung abgebrochen.

```
NeueKategorie = "Nein";
```

```
break;
```

Hat die Variable „NeueKategorie“ den Wert „Ja“, fügt man den Wert der Variable „Kategorie“ mittels der „push“-Methode dem „Kategorien“-Array hinzu.

```
if(NeueKategorie == "Ja")
```

```
Kategorien.push(Kategorie);
```

Zum Schluss muss noch die Möglichkeit abgedeckt werden, dass die Variable „RestPfad“ leer ist. In diesem Fall verschiebt man den aktuellen Wert der Variable „i“ in das „Linkbuttons“-Array.

```
Linkbuttons.push(i);
```

Als nächstes erfolgt die dynamische Erstellung der einzelnen Buttons der Navigation. Für die korrekte Darstellung müssen drei verschiedene Arten von Buttons generiert werden, welches der „Zurück“-Button, die Kategoriebuttons und die Linkbuttons sind. Als erstes wird für die Zählung der Buttons eine separate Zählvariable festgelegt. Anhand dieser werden die Angaben aus dem „Button“-Array der Reihenfolge nach in die Navigation eingefügt.

Der Autor möchte an dieser Stelle zuerst die Erstellung des „Zurück“-Buttons näher erläutern. Dieser soll nur sichtbar sein, wenn man sich in einem Untermenü befindet. Die Überprüfung erfolgt über den Zustand der Variable „MenuPfad“, welche nur im Hauptmenü leer ist.

```
if(MenuPfad != "")
```

Nachdem die Überprüfung erfolgreich war, wird der grafische Button mittels der „attachMovie“ an den MovieClip „Menuleiste“ angehängt.

```
Buttons[i] = Menuleiste.attachMovie("Button1", "Zurueck",  
Menuleiste.getNextHighestDepth());
```

Für die genaue Bestimmung werden drei Parameter benötigt. Der erste ist die Bezeichnung des anzuhängenden MovieClips, der zweite ist die eindeutige Bezeichnung für den anzuhängen MovieClip und der letzte ist die Angabe der Tiefenebene auf der der Clip abgelegt werden soll. Als nächstes folgt die Beschriftung für den „Zurück“-Button. Dafür wird dem Textfeld mit der Variable

„Textfeld“ innerhalb des MovieClips „Button1“ ein entsprechender String zugewiesen.

```
Buttons[i].Textfeld = "<< Zurück";
```

Zuletzt muss für den Button noch die Funktion für das Loslassen der Maustaste implementiert werden.

```
Buttons[i].onRelease = function()
```

Da der Button aus drei Frames besteht, wird als nächstes festgelegt auf welchen er beim Loslassen der Taste springen soll.

```
this.gotoAndPlay(1);
```

Gleichzeitig soll auch der Name des neuen aktuellen Untermenüs in der Anzeige dargestellt werden.

Hierfür spaltet man den aktuellen „MenuPfad“ in ein Array auf.

```
MenuPfad = MenuPfad.split(Tz);
```

Als nächstes müssen die letzten beiden Arrayelemente entfernt werden, da diese keine Kategorien enthalten.

```
MenuPfad.splice(MenuPfad.length - 2);
```

Nun erfolgt die Ausgabe der aktuellen Kategorie in Textfeld „Anzeige“. Um das zu erreichen ermittelt man das den Wert des letzten Arrayelements und weißt es der Anzeige zu.

```
Anzeige = "*" + MenuPfad.slice(-1) + "*";
```

Zum Schluss wird das Array „MenuPfad“ wieder in die Stringvariable „MenuPfad“ umgewandelt.

Dafür verwendet man die „join“-Methode und das Trennzeichen.

```
MenuPfad = MenuPfad.join(Tz) + Tz;
```

Wenn man sich wieder im Hauptmenü befindet, besteht der „MenuPfad“ nur aus dem Trennzeichen.

Für diesen Fall müssen die Variable „MenuPfad“ und „Anzeige“ geleert werden.

```
if(MenuPfad == Tz)
```

```
MenuPfad = "";
```

```
Anzeige = "";
```

Als nächstes werden die Buttons für einzelnen Unterkategorien generiert. Für diesen Vorgang verwendet man eine „for“-Schleife.

```
for(j = 0; j < Kategorien.length; j++)
```

Die dafür vorgesehene Zählvariable „j“ soll in diesem Fall bis zur Gesamtanzahl der Einträge im Kategorienarray zählen. Nun werden die grafischen Buttons wieder dem MovieClip „Menuleiste“ mittels der „attachMovie“ hinzugefügt.

```
Buttons[i] = Menuleiste.attachMovie("Button1", "Button1" + i,  
Menuleiste.getNextHighestDepth());
```

Dabei wird die eindeutige Bezeichnung diesmal zusätzlich mit der Zählvariablen für die Buttons

versehen. Danach weist man den Buttons die Beschriftungen zu. Zur Unterscheidung mit den Linkbuttons entschied man für die Kategorien eine Hervorhebung einzufügen.

```
Buttons[i].Textfeld = "*" + Kategorien[j] + "*";
```

Die Kategorien erscheinen nun innerhalb von zwei Sternen auf den Buttons und heben sich von den Linkbuttons ab.

Nun muss die aktuelle Kategorie des Buttons dauerhaft gespeichert werden. Dafür hängt man an das „Button“-Objekt die Eigenschaft „Kategorie“ an und weist ihr den Wert der Variablen „Kategorien[j]“ zu. Damit ist jedem Button die entsprechende Kategorie zugewiesen in der er sich befindet.

```
Buttons[i].Kategorie = Kategorien[j];
```

Zum Abschluss muss auch hier wieder definiert werden, was bei einem Mausklick geschehen soll.

```
Buttons[i].onRelease = function()
```

Auch hier soll die Ansicht des Buttons wieder in Frame 1 springen.

```
this.gotoAndPlay(1);
```

Als nächstes überträgt man den aktuellen Kategorienamen des Buttons auf das Anzeigetextfeld.

```
Anzeige = this.Textfeld;
```

Nun muss der neue „MenuPfad“ erstellt werden. Dies geschieht durch Aneinanderreihung der alten Variable „MenuPfad“, der aktuellen Kategorie des Buttons und dem Trennzeichen

```
MenuPfad = MenuPfad + this.Kategorie + Tz;
```

Danach wird die Anwendung durch einen Sprungbefehl in Frame 4 gezwungen, um die Buttons im Untermenü zu generieren.

Die letzten generierten Buttons sind die Linkbuttons, über welche man die Animation aufruft. Auch hier verwendet man wieder eine „for“-Schleife, die bis zur Anzahl der Elemente im „Linkbuttons“-Array zählen soll.

```
for(j = 0; j < Linkbuttons.length; j++)
```

Als erstes hängt man auch hier wieder den grafischen Button an den MovieClip „Menuleiste“ an.

```
Buttons[i] = Menuleiste.attachMovie("Button1", "Button1" + i, Menuleiste.getNextHighestDepth());
```

Die eindeutige Bezeichnung der Buttons wird erneut über die Zählvariable „i“ vorgenommen.

Jetzt fügt man dem Button die Beschriftung hinzu.

```
Buttons[i].Textfeld = Elemente[Linkbuttons[j]].Bezeichnung;
```

Der Parameter „Linkbuttons[j]“ gibt die Nummer des Buttonelements innerhalb des „Elemente“-Arrays an.

Danach wird dem aktuellen Button die Eigenschaft „Link“ hinzugefügt und der entsprechende Wert

aus dem „Elemente“-Array mit der Eigenschaft „link“ zugewiesen.

```
Buttons[i].Link = Elemente[Linkbuttons[j]].link;
```

Abschließend müssen auch für diese Buttons die Funktionalitäten implementiert werden.

```
Buttons[i].onRelease = function()
```

Jetzt werden sämtliche Linkbuttons zurückgesetzt. Für den Fall, dass bereits ein anderer Linkbutton innerhalb der selben Kategorie gedrückt wurde, setzt man sie mit den folgenden Scriptzeilen auf ihre Ausgangsansicht:

```
for(k = 0; k < Buttons.length; k++)
```

```
Buttons[k].gotoAndStop(1);
```

Danach setzt man den aktuell gedrückten Button auf die „gedrückt“-Ansicht. Diese ist im Frame 3 des „Button1“-MovieClips enthalten.

```
this.gotoAndPlay(3);
```

Nun fragt man den Link ab, ob es sich um eine XML-Datei handelt oder nicht. Dazu ermittelt man die letzten drei Zeichen des Links, welche mit dem gedrückten Button verknüpft ist.

```
var Endung = this.Link.substr(-3,3);
```

Die Parameter für die „substr“-Methode bedeuten den Beginn und die zu lesende Länge des Strings. Für diesen Fall bedeutet das, dass die Zählung ab dem drittletzten Zeichen beginnen und drei Zeichen umfassen soll. Den ermittelten Wert weist man der Variable „Endung“ zu.

Hat die Variable den Wert „xml“ handelt es sich um eine Playliste und der Playlistcontainer muss geladen werden. Außerdem setzt man die Variable „Playlistpfad“ auf den Wert des Links.

```
if(Endung == "xml")
```

```
Playlistpfad = this.Link;
```

```
_root.Content.loadMovie("playlistcontainer.swf")
```

Besitzt die Variable „Endung“ einen anderen Wert als „xml“, lädt man einfach den Link, welcher mit dem Button verknüpft ist.

```
_root.Content.loadMovie(this.Link);
```

Wenn bereits eine andere Animation lief müssen die beiden Kontrollvariablen „Fertig“ und „PlayYN“ wieder neu gesetzt werden.

```
_root.Fertig = false;
```

```
_root.PlayYN = true;
```

Zum Schluss des Ganzen muss man die Zählvariable „i“ noch um eins erhöhen, damit die Zählung der Buttons korrekt erfolgen kann.

Bei der dynamischen Erstellung der einzelnen Buttons treten einige Scriptelemente wiederholt auf. Dies ist erforderlich, da für jede Art Button bestimmte Funktionen immer gelten. So sind die

Scriptzeilen für die Positionierung und die Funktionen für Mauseaktionen wie „Drag Out“, „Release Outside“ und „Press“ für jeden Button dieselben.

Die Ausrichtung der Buttons erfolgt in jedem Fall zuerst nach der x-Achse und als zweites nach der y-Achse. Da die grafischen Buttons bereits einen Schatten integriert haben, ist es nicht notwendig einen Abstand zwischen den einzelnen Menüelementen festzulegen.

Die folgenden Zeilen bestimmen die Ausgabe der Buttons auf der Nutzeroberfläche:

```
Buttons[i]._x = 30; .
```

Diese dient der Festlegung der Entfernung zum linken Anwendungsrand:

```
Buttons[i]._y = 125 + ( i * 40 ); .
```

Die oben stehende Anweisung wird zur Bestimmung der Buttons untereinander eingesetzt, wobei der oberste bei 125 Pixeln zu sehen ist.

Als nächste Anweisungen folgen die Befehle für verschiedene Mauseaktionen. Die hierbei bedachten Aktionen sind das Drücken der Maustaste und Bewegen des Mauszeigers außerhalb des Buttons, das Loslassen der gedrückten Maustaste außerhalb des Buttons und das einfache Drücken der Maustaste über dem Button.

Für die genannten Funktionen existieren in Flash vorgefertigte Befehle.

```
Buttons[i].onDragOut = function()  
this.gotoAndPlay(1);
```

Die obenstehende Funktion bewirkt, dass der Button wieder auf seine Ausgangsansicht zurück gesetzt wird, wenn man bei gedrückter Maustaste die Buttonfläche verlässt.

```
Buttons[i].onReleaseOutside = function()  
this.gotoAndPlay(1);
```

Diese Funktion setzt den Button ebenfalls auf seine Ausgangsansicht zurück, wenn die Maustaste außerhalb des Buttons losgelassen wird.

```
Buttons[i].onPress = function()  
this.gotoAndPlay(2);
```

Die letzte gemeinsame Funktion der Buttons ist die „onPress“-Funktion. Diese ermöglicht es, dass der Button den gedrückten Zustand annimmt, wenn man die Maustaste über einem Button betätigt. Im Anhang sieht man, an welcher Stelle diese Scriptteile eingefügt werden müssen. Das komplette Script ist dort unter Script 2 zu finden.

4.1.3.4 Dozentenleiste

Diese Leiste dient der eindeutigen Festlegung auf einen bestimmten Dozenten, dessen Fachbereich und der Hochschule. Die Daten für diese Angaben werden in der menu.xml-Datei festgelegt und zusammen mit den Daten der Navigation eingelesen.

Die Leiste besteht im Wesentlichen aus zwei Textfeldern, eins linksbündig und eins rechtsbündig angeordnet. Als Schriftart wurde Arial mit einer Größe von zwölf Punkten gewählt. Zur besseren optischen Wirkung erfolgte die Darstellung der Schrift fett und kursiv.

Das linke Textfeld dient der Anzeige des Dozentennamens. Hierfür wurde die Variable „Prof“ festgelegt. Analog dazu ist das rechte Textfeld für die Darstellung des Fachbereichs und der Hochschule konzipiert worden. Die dafür vorgesehene Variable ist „Fach“. Variablen für Textfelder werden im Eigenschaftsmenü von Flash vergeben.

Im ersten Bild der Nutzeroberfläche werden die Daten aus der XML-Datei heraus gelesen (siehe 4.1.3.2). Die Befehlszeilen für die Identifizierung des Dozenten lauten wie folgt:

```
if (XMLDaten.firstChild.childNodes[i].attributes.doz) {  
Doz = XMLDaten.firstChild.childNodes[i].attributes.doz;
```

Letztere Anweisung wird nur ausgeführt, wenn das Attribut „doz“ innerhalb der XML-Datei vorhanden ist.

Zur Identifizierung des Fachbereichs erfolgt die die Befehlserstellung analog . Hierbei wird nach dem Attribut „fach“ gesucht.

```
if (XMLDaten.firstChild.childNodes[i].attributes.fach) {  
Fb = XMLDaten.firstChild.childNodes[i].attributes.fach;
```

Die Variable „XMLDaten“ enthält sämtliche Informationen der geladenen XML-Datei. Mit diesen Befehlen werden die einzelnen Zeilen der XML-Datei nach den Attributen „doz“ und „fach“ durchsucht. Sind diese vorhanden, erfolgt die Zuweisung zu den ActionScript-Variablen „Doz“ und „Fb“.

Im fünften Frame der Oberfläche werden die Inhalte der ActionScript Variablen den oben genannten Variablen der entsprechenden Textfelder zugewiesen. Dies sind die Befehle „Prof = Doz;“ und „Fach = Fb;“.

Analog zur Erstellung der Dozentenleiste geschieht das Einfügen der Fachüberschrift. Der Unterschied besteht hier lediglich in den verwendeten Variablen, welche „Headline“ für das

entsprechende Textfeld und „hl“ für die zugehörige XML-Variable wären.

4.1.3.5 Filmsteuerung

Die Steuerung von Präsentationen durch den Lehrenden erfolgt entweder über eine Funkmaus oder über die Tastatur des Laptops. Anfangs entschied man sich bei der Realisierung dafür eine Tastatursteuerung umzusetzen, welche ohne Steuerungselemente ausgekommen wäre. Nach weiteren Überlegungen wurde die Steuerung um eine Maussteuerung erweitert. Es sind nun je nach Bedarf beide Versionen gleichzeitig möglich.

Als erstes möchte der Autor die Implementierung der Maussteuerung vorstellen.

Zuerst wurden die einzelnen Steuerelemente grafisch erstellt, wozu man die von Videogeräten her bekannten Elemente „rückwärts“, „abspielen“, „vorwärts“, „Pause“ und „Stopp“ verwendete. Die Gestaltung erfolgte als Textelemente mit der Schriftart Webdings als separate Schaltflächen. Dabei steht die Ziffer „4“ für die Playtaste, die „7“ für rückwärts, die „8“ für vorwärts, das „:“ für Pause, „<“ steht für Stopp und der „:“ für die Schleifenanzeige. Durch die Schriftart werden die genannten Zeichen in die in Abbildung 26 dargestellten Symbole umgewandelt. Als Schriftgröße für die Tasten wurde 50 festgelegt.

Als nächstes müssen die einzelnen Felder als Schaltflächen definiert werden. Dies funktioniert über das Eigenschaftenmenü von Flash. Die gesamte Steuerung wird zu dem MovieClip „PlayController“ zusammengefasst und als solcher weiterverwendet, welcher in Abbildung 22 zu sehen ist.

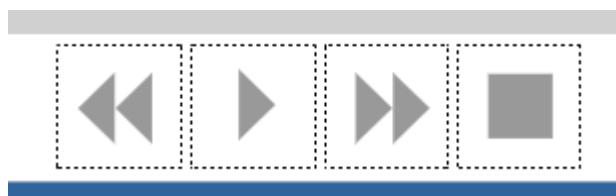


Abbildung 23: Filmsteuerung

Die Implementierung der Steuerung erfolgt über das ActionScript. Hierfür werden Variablen benötigt, welche in Bild 1 der Anwendung zu definieren sind. Am wichtigsten ist hierbei die „PlayYN“-Variable. Diese überprüft den aktuellen Zustand einer Animation, das heißt, ob diese gerade läuft oder nicht. Anhand des Zustandes der Variable werden die Buttons verändert. Das gleiche gilt für die „Schleife“-Variable. Hierbei erfolgt die Überprüfung, ob sich eine Animation in einer Schleife befindet oder nicht. Daraufhin werden die Symbole der Steuerung entsprechend verändert. Die Vorgehensweise ist dabei sehr simpel. Es wird anhand einer „if-else“-Anweisung der Zustand der Kontrollvariable geprüft und danach der Inhalt der Steuerungstextfelder auf den gewünschten Wert gesetzt. Dazu wies man den Textfeldern Variablen zu, welche per ActionScript

ansprechbar sind. Um die Variablen in Bild 1 des Menüs zu erreichen, muss der absolute Pfad für diese angegeben werden. Dies erfolgt über die Angabe „_root“, welches auf die oberste Ebene innerhalb der Animation verweist. Nötig ist dies deshalb, da die Steuerung ein eigener MovieClip namens „PlayController“ ist, der in die Nutzeroberfläche eingebettet ist und somit eine Ebene „tiefer“ liegt. Da nur die Play-Pause-Taste und der Schleifenbutton verändert werden, müssen der „Vor“- und „Zurück“-Taste keine Variablen zugewiesen werden. Für die beiden erstgenannten Felder wurden „PlayPause“ und „CtrlSchleife“ (für Controll Schleife) als Variablen verwendet.

Als nächstes werden die Funktionalitäten der Buttons implementiert. Der Play-Pause-Button fungiert einerseits als normaler Pausebutton, während eine Animation läuft, und andererseits als Schleifenabbruchbutton, wenn sich die Animation innerhalb einer Schleife befindet. Damit ActionScript auf einen Mausklick reagiert, muss man die Funktion „on“ verwenden. Diese benötigt noch einen genauen Parameter, wann die Funktion ausgelöst werden soll. In den folgenden Fällen hat man sich entschieden, die „release“-Eigenschaft anzuwenden. Damit registriert ActionScript den Mausklick beim Loslassen der Maustaste. Der komplette Befehl sieht folgendermaßen aus: `on(release)` .

Nun muss abgefragt werden, ob es sich um die Taste für den Schleifenabbruch handelt, welche betätigt wurde. Wenn sich die Animation innerhalb einer Schleife befindet, sieht dieser Button aus, wie das Playsymbol während einer Pause, da sich die Animation gewissermaßen in einer animierten Pause befindet. Hierzu verwendet man die im Bild 1 der Menü-Datei erstellte Booleanvariable „Schleife“. Zur Überprüfung findet erneut eine „if“-Anweisung Anwendung. Wenn die Variable den Wert true besitzt, werden die folgenden Befehle ausgeführt. Zuerst prüft man anhand der „PlayYN“-Variable, ob die Animation läuft oder pausiert ist. Je nach Ergebnis kommt es zum Stoppen oder Weiterlaufen der Animation. Des Weiteren muss der Wert der „PlayYN“-Variable neu gesetzt werden. Die Befehle für den Stop-Vorgang sehen wie folgt aus:

`_root.Content.stop()`; Dieser Befehl dient der Sicherheit und stoppt den Playlistencontainer.

`_root.Content.Content.stop()`; Hier wird die eigentliche Animation, welche in den Playlistcontainer eingebettet ist, gestoppt.

`_root.PlayYN = false`; An dieser Stelle setzt man den Wert der Variable „PlayYN“ auf „false“, damit die Animation bei nochmaligem Drücken der Taste wieder weiterlaufen kann.

Analog dazu folgen hier die Befehle für das Fortführen der Animation:

`_root.Content.play()`; Hier erfolgt das erneute Starten des Playlistcontainers.

`_root.Content.Content.play()`; Dieser Befehl bewirkt das Weiterlaufen der eigentlichen

Animation.

`_root.PlayYN = true;` Die Anweisung dient ebenso dazu, den Wert der „PlayYN“-Variable umzukehren, damit die Animation beim wiederholten Druck auf den Button angehalten werden kann.

Für den Fall einer animierten Schleife wurde der vierte Button eingebaut. Dieser ermöglicht es, die Animation zu pausieren, ohne die Schleife zu verlassen. Die Befehle für das Pausieren und Abspielen sind dieselben wie bei dem Play-Pause-/Schleifenabbruchbutton. An dieser Stelle jedoch ohne die Abfrage, ob sich die Animation innerhalb einer Schleife befindet.

Abschließend folgt die Erläuterung der Funktionen der „Vor“- und „Zurück“-Tasten, welche zum Überspringen oder Wiederholen einzelner Animationsabschnitte gedacht sind. Sie werden ebenfalls über die „on(release)“-Funktion aktiviert. Bei der Vorwärtstaste erhöht man die Variable „Nummer“ innerhalb der Playliste um „1“, während man bei der Zurücktaste den Wert um „1“ verringert. Die genaue Auswertung der entsprechenden Listen erfolgt in der Playlistcontainer-Datei. Auf deren Funktionen wird im Abschnitt 4.1.3.6 näher eingegangen. Die Befehle für die beschriebenen Vorgänge lauten wie folgt:

`_root.Nummer++;` dient dem Überspringen einer Animation beim Vorwärtsbutton,
`_root.Nummer--;` ermöglicht das Wiederholen einer vorangegangenen Animation beim Zurückbutton,
`_root.Content.gotoAndPlay(2);` lässt den Playlistcontainer zu Bild 2 springen und ablaufen.

Als zweites möchte der Autor die Implementierung der Tastatursteuerung vorstellen.

Damit die Anwendung auf Tastatureingaben reagiert, muss eine „KeyListener“-Funktion erstellt werden. Diese soll alle benötigten Befehle zur Steuerung der Anwendung beinhalten, welche eine „Vorwärts“- und eine „Rückwärts“-Taste, ein „Schleifenabbruch“-Button und eine „Play/Pause“-Taste wären.

Der Befehl zur Erstellung der Funktion lautet wie folgt:

```
KeyListener.onKeyDown = function().
```

Als nächstes werden die benötigten Befehle zur Steuerung implementiert. Für die „Play/Pause“-Taste, welche leicht erreichbar sein muss, ist hier die Leertaste gewählt worden. Nun muss überprüft werden, welche Taste man auf der Tastatur gedrückt hat. Dazu nutzt man den Befehl „getCode“. Der Code für die Leertaste entspricht „SPACE“. Außerdem muss eine Prüfung erfolgen, ob die Animation nicht bereits zu Ende ist. Für diesen Vorgang hat man die „Fertig“-Variable

eingeführt. Zum Abgleich eignet sich eine „if“-Anweisung am besten.

Daraus ergibt sich die nachstehende Codezeile:

```
if ( (Key.getCode () == Key.SPACE) && (!_root.Fertig) ).
```

Die sich nun anschließenden Befehle entsprechen den bereits oben genannten für den „Play/Pause“-Button.

Für den Schleifenabbruch hat man die „Pfeil-nach-unten“-Taste gewählt. Hier muss ebenfalls der Code für die dafür vorgesehene Taste ausgewertet werden. In diesem Fall lautet der Code „DOWN“.

Der vollständige Befehl sieht wie folgt aus:

```
if (Key.getCode () == Key.DOWN) .
```

Auch die nach diesem Befehl folgenden Zeilen sind identisch mit den bereits oben erwähnten für den Schleifenabbruch.

Abschließend werden noch die beiden Tasten für „Vorwärts“ und „Rückwärts“ benötigt, wofür die „Links“- und die „Rechts“-Taste gewählt wurden. Auch an dieser Stelle muss man eine Überprüfung durchführen. Für die beiden Tasten lauten die benötigten Codes „LEFT“ und „RIGHT“. Die kompletten Befehle für diese Überprüfung sehen folgendermaßen aus:

```
if (Key.getCode () == Key.LEFT)
```

```
if (Key.getCode () == Key.RIGHT) .
```

Hiernach folgende Befehle entsprechen den bereits genannten für „Vorwärts“ und „Rückwärts“.

Für die vorgestellten Funktionen finden sich die vollständigen Scripte im Anhang unter Script 1, 3, 4, 5, 6 und 7.

4.1.3.6 Playlistensteuerung

Die Auswertung und Steuerung der Playlisten wird von entsprechendem ActionScript in der Datei Playlistcontainer fla übernommen. Diese Datei lädt man als Content innerhalb der menu fla.

Da die Datei nur als Ladeoberfläche dienen soll, definiert man auch hier die Fläche für die Anzeige der Animationen. Das funktioniert mit derselben Befehlskette wie innerhalb der menu fla.

```
var Content:MovieClip = this.createEmptyMovieClip ("Content",  
this.getNextHighestDepth ());
```

Der Container hat die genauen Abmessungen der zukünftigen Animationen. Somit muss das leere MovieClip-Objekt an der linken oberen Ecke platziert werden.

```
Content._x = 0; Content._y = 0;
```

Als nächstes sind einige Variablen zu erstellen. Das wären ein XML-Objekt zum Laden der Daten aus der Playlisten-XML-Datei, ein Array, welches die genaue Menge an Animationsabschnitten

aufnimmt und die Zählvariable zur Steuerung der Animationsabschnitte. Letztere wird zu Beginn auf „0“ gesetzt, um bei jeder Animationsreihe mit der ersten Animation zu beginnen. Danach lädt man die XML-Daten. Das Prinzip zum Laden der XML-Daten verläuft analog zur Datei menu fla, weswegen an dieser Stelle nicht noch einmal explizit darauf eingegangen wird. Ein Unterschied besteht lediglich in der Angabe der zu ladenden XML-Datei. Der Befehl für diese Datei lautet in diesem Fall wie folgt:

```
XMLDaten.load(_root.Playlistpfad);
```

Die Dateiangabe im Playlistcontainer wird aus der Variable „Playlistpfad“ aus der menu.xml im Stammverzeichnis übernommen.

Nach erfolgreichem Laden bestimmt man die Anzahl der einzelnen Animationen in der Playliste. Für diesen Vorgang wird die Gesamtanzahl der in der geladenen XML-Datei enthaltenen Elemente ermittelt. Der genaue Befehl dafür lautet wie folgt:

```
AnzahlMovies = XMLDaten.firstChild.childNodes.length;
```

Als nächstes müssen die Inhalte der XML-Datei in ein entsprechendes Array geschrieben werden, um sie später weiterverarbeiten zu können. Dafür verwendet man erneut eine „for“-Schleife, welche von Null bis zum Wert von „AnzahlMovies“ zählen soll. Der Befehl, welcher dazu verwendet wird, die Daten in das Array zu schreiben, sieht wie folgt aus:

```
Mov[i].movie = XMLDaten.firstChild.childNodes[i].attributes.movie;
```

Nachdem sämtliche Animationsverknüpfungen in das Array geladen wurden, springt der Playlistcontainer auf Frame 2, wo man die aktuellen Nummern aus der Steuerung prüft, ob diese überhaupt zulässig sind. Zu diesem Zweck erfolgt ein Vergleich des Inhalts der Variablen „Nummer“ und „AnzahlMovies“. Dabei wird zuerst geprüft, ob der Wert von „Nummer“ den von „AnzahlMovies“ überschritten hat oder gleichwertig ist. Ist dies der Fall, muss der Wert von „Nummer“ auf den von „AnzahlMovies - 1“ gesetzt werden. Dies ist notwendig, da „Nummer“ nie größer oder gleich dem Wert von „AnzahlMovies“ sein darf. Der Umstand ergibt sich aus der unterschiedlichen Zählweise der beiden Variablen. Während „Nummer“ bei „0“ anfängt, nimmt „AnzahlMovies“ in der Regel einen Wert größer „0“ an.

Die benötigten Scriptzeilen sehen wie folgt aus:

```
if(_root.Nummer >= AnzahlMovies)
_root.Nummer = AnzahlMovies - 1;
```

Als zweites erfolgt eine Prüfung, ob der Wert von „Nummer“ kleiner als „0“ ist. Da es keine negativen Animationszahlen geben kann, darf „Nummer“ auch nicht kleiner als „0“ werden. Die nachfolgenden Befehlszeilen sorgen dafür, dass dieser Fall nicht eintritt:

```
if(_root.Nummer < 0)
```

```
_root.Nummer = 0;
```

Zum Schluss lädt man die aktuelle Animation in das Content-Feld des Playlistcontainers. Dafür verwendet man den nachstehenden Befehl:

```
Content.loadMovie (Mov[_root.Nummer].movie);
```

Der Befehl „loadMovie“ bezieht seine Informationen aus dem „Mov“-Array, welches durch den aktuellen Wert der Variable „Nummer“ bestimmt wird.

Nachdem das Laden der gewünschten Animation erfolgreich war, muss die laufende Animation überwacht werden. Die dafür benötigten Scriptroutinen befinden sich in Frame 3 des Playlistcontainers.

Zunächst werden hier die Gesamtanzahl der innerhalb der laufenden Animation vorhanden Frames und der aktuelle Frame bestimmt. Die folgenden Befehle ermöglichen diese beiden Vorgänge:

```
gesamt = Content._totalframes;
aktuell = Content._currentframe;
```

Als nächstes vergleicht man die soeben ermittelten Werte miteinander, um zu überprüfen, ob die aktuelle Animation das Ende erreicht hat oder nicht. Für diesen Vorgang findet erneut eine „if“-Anweisung Verwendung. Wenn die beiden Werte übereinstimmen und die aktuelle Animation somit ihr Ende erreicht hat, soll die nächste Animation abgespielt werden. Dazu erhöht man den Wert der Variable „Nummer“ um eins. Für diesen Vorgang sind die folgenden Befehle erforderlich:

```
if (aktuell == gesamt)
_root.Nummer++;
```

Danach prüft man, ob der Wert der Variable „Nummer“ unter der Länge der geladenen Playliste liegt. Solange das der Fall ist, soll der Playlistcontainer auf Frame 2 springen, um erneut die Prüfung der Variablen vorzunehmen und dann die entsprechende Animation abzuspielen. Anderenfalls soll der Playlistcontainer auf Frame 5 springen, um die Animation zu beenden. Für diese beiden Vorgänge sind die folgenden Befehlszeilen verantwortlich:

```
if (_root.Nummer < Mov.length)
gotoAndPlay(2);
else
gotoAndPlay(5);
```

Der Sprungbefehl in Frame 4 sorgt dafür, dass die aktuelle Framenummerüberprüfung in Frame 3 fortlaufend stattfindet.

Nach der Überprüfung in Frame 3, ob weitere Animationen abgespielt werden sollen, stoppt man per Script in Frame 5 alle Teilanwendungen und setzt die Variable „Fertig“ auf „true“.

```
_root.Fertig = true;
```

```
Content.stop();
```

```
stop();
```

Die doppelte „stop“-Anweisung sorgt dafür, dass einerseits der Playlistcontainer und andererseits die darin befindliche Animation gestoppt werden.

Die kompletten Scripte sind im Anhang unter den Nummern 8 bis 11 zu finden.

4.2 Die Animationen

Im folgenden Abschnitt erläutert die Erstellung der Beispielanimationen im Detail. Es werden Probleme und deren verschiedene Lösungsansätze aufgezeigt.

4.2.1 Grafische Erstellung der Animationen

Bevor man Effekte und Funktionen der Animationen erstellt, müssen die grafischen Elemente kreiert werden. Die Mehrzahl der Objekte in den Beispielen bestehen aus Rechtecken. Diese wurden mittels des Rechteckwerkzeugs von Flash generiert. Danach markiert man die einzelnen Elemente der Objekte mit dem Auswahlwerkzeug und wandelt sie per rechter Maustaste und der Funktion „In Symbol konvertieren“ in ein zusammenhängendes Symbol um. Dieses kann man je nach Bedarf färben, skalieren und kopieren. Dadurch werden wiederverwendbare Objekte möglich, die man in einer Bibliothek zusammenfassen kann. Innerhalb der Animation benötigte Pfeile hat man mit dem Linienwerkzeug erstellt.

4.2.2 Aufteilung der Gesamtanimation in Teilabschnitte

Um eine bessere Modularität zu erhalten, entschied man sich dazu, die Animationen in Teilabschnitte aufzuspalten. Zum einen wären dies die einleitenden Texte und die genaueren Spezifikationen und zum anderen die eigentliche erklärende Animation. Bei späteren Animationen können dies durchaus noch weitere Abschnitte werden, die mittels einer Playliste zusammenzufügen sind. Diese besteht aus einer einfach strukturierten XML-Datei, welche in jedem Animationsverzeichnis abgelegt ist. Darin festgelegt sind die genauen Pfadangaben der Animationen und die entsprechende Reihenfolge, in der das Abspielen erfolgen soll. Die Verarbeitung der XML-Daten verhält sich analog zum Einlesen der Daten in der Nutzeroberfläche. Lediglich die Auswertung der eingelesenen Daten erfordert andere Angaben. Der Vorgang erfolgt allerdings in der Datei „playlistcontainer fla“, welche als Anzeigeobjekt für die Animationen Verwendung findet.

4.2.3 Verknüpfung der Animationen mit der Benutzeroberfläche

Da die Animationen über die Benutzeroberfläche gesteuert werden sollen, muss man sie auch mit dieser verbinden. Dieser Vorgang erfolgt über entsprechende globale Variablen im ActionScript. Hierzu werden innerhalb der Animation an den gewünschten Haltepunkten die entsprechenden Variablen gesetzt. Wenn die Animation angehalten werden soll, muss man die Variable PlayYN auf „false“ setzen und danach einen stop-befehl aufrufen. Das Script dafür sieht wie folgt aus:

```
_root.PlayYN = false; damit die Tastensteuerung funktioniert  
stop(); für das Stoppen der Animation
```

Es gibt auch Stellen innerhalb von Animationen, welche sich als selbst wiederholende Schleifen besser eignen, als ein Standbild. In diesen Fällen werden andere Befehle benötigt. Zu Beginn einer Schleife wird die Variable für den Schleifenabbruch auf „false“ gesetzt, damit die Schleife wiederholt, bis die Taste für den Abbruch gedrückt wird.

```
_root.Schleife = false;
```

Am Ende einer Schleife erfolgt eine Abfrage über den Status der „Schleife“-Variable. Besitzt diese den Wert „false“ ist der Sprungbefehl auf den Anfang der Animation auszuführen.

Die dafür benötigten Befehle lauten wie folgt:

```
if(!_root.Schleife) Abfrage des Schleifenstatus  
gotoAndPlay(201); Sprungbefehl, in diesem Fall auf den Frame 201
```

Mit diesen beiden Funktionen sind die häufigsten Möglichkeiten der Steuerung abgedeckt.

4.2.4 Realisierung einer Bewegungsanimation

Es gibt verschiedene Formen der Animation in den beiden bearbeiteten Beispielen. Darin sind sowohl eine Zoomanimation als auch eine Füllstandsanimation vorhanden. Auch wenn man zuerst annehmen würde, dass es sich um sogenannte Formtweens handelt, wurden beide als Bewegungstweens kreiert. Bei Formtweens tritt häufig ein Morphingeffekt auf, welcher die gesamte Gestalt zwischen der Anfangs- und Endphase verändert. Ein solcher Tween wird in der Regel zwischen zwei Schlüsselbildern erstellt. Dies realisiert man, indem man die Animation zu Beginn und am Ende der Veränderung gestaltet. Danach wählt man den Beginn aus und fügt über das Eigenschaftenmenü von Flash den Bewegungstween hinzu.

4.2.5 Erstellung von scrollendem Text

Für die Animation des PIO Mode war es erforderlich, eine weitere Textbox einzubinden, welche die einzelnen Kommandos während der Datenübertragung darstellt. Da diese Box nur eine begrenzte

Größe hat, war es notwendig, den Text nach unten scrollen zu lassen.

Diese Aufgabe bereitete etliche Probleme. Die Grundlösung der Situation bestand im ActionScript. Zuerst fügte man an jedem Scrollpunkt Schlüsselbilder ein und gab die maximal zu sehenden Texte in die Textfelder ein. Mithilfe der „MaxScroll“-Eigenschaft wurde bei jedem Schlüsselbild bis zum Ende des Textes in der Box gescrollt. An dieser Stelle trat das erste Problem zu Tage. Der Text scrollte unregelmäßig. Dieses Phänomen trat durch die ungerade Anzahl an Frames für eine Scrolleinheit auf. Es wurden fünfzehn Frames für eine Einheit festgesetzt, da die Animation sonst merklich länger geworden wäre und die Standardeinteilung der Zeitleiste von Flash in Fünferschritten eingeteilt ist. Das Problem löste man durch einen zweiten Frame innerhalb des Textctrl-Movies, welcher die „MaxScroll“-Eigenschaft ein zweites Mal enthielt. Dadurch wurde nun auch während der geraden Frames der Scrollbefehl ausgeführt. Da die gesamte Erstellungs- und Änderungsprozedur für diese Art zu aufwendig war, verwarf man sie später wieder.

Die endgültige Lösung besteht in einem Textarray, welches sämtliche anzuzeigenden Textzeilen enthält, und einer selbsterstellten Scrollfunktion. Die Ersterstellung ist zwar komplizierter als die Ursprungsversion, allerdings lassen sich Änderungen am Anzeigetext wesentlich einfacher vornehmen.

Bevor die Notwendigkeit der Scrollfunktion innerhalb der Animation gegeben ist, werden für die beiden Textfelder die entsprechenden Arrays in einem Frame erstellt. Es sind zwei Arrays nötig, da es ebenfalls zwei Textfelder sind, welche einen tabellenartigen Aufbau simulieren sollen. Als nächster Schritt sind die Arrays mit den erforderlichen Daten zu füllen. Die dafür verwendeten Befehle lauten wie folgt:

```
var AktionsDaten = new Array(); .
```

Mittels dieses Befehls wird das Array für die Aktionsdaten erstellt. Der Befehl für die Befehlsdaten sieht entsprechend ähnlich aus:

```
var BefehlsDaten = new Array(); .
```

Die folgende Zeile wird solange wiederholt, bis sämtliche Daten in das Array geschrieben wurden:

```
AktionsDaten.push("1) Instruction Fetch"); .
```

Die „push“-Funktion hängt den in Klammern stehenden Wert als Element an das gewünschte Array an.

Analog dazu lautet die Befehlszeile für die Befehlsdaten wie folgt:

```
BefehlsDaten.push("mov reg, dvd"); .
```

Auch diese Zeile ist ebenfalls solange wiederholt einzutragen, bis die gewünschten Daten vollständig in das Array geschrieben worden sind. Für das gewählte Beispiel sind für jedes Textfeld 80 „push“-Befehlszeilen erforderlich.

Einen Auszug des Scripts findet man im Anhang unter der Nummer 12.

Im darauf folgenden Frame wird die Scrollfunktion zur weiteren Verwendung definiert. Dies ermöglicht es, bei jedem Scrollvorgang lediglich den selbst definierten Befehl zu verwenden. Dadurch sinkt der Arbeitsaufwand für Änderungen erheblich.

Zuerst werden dafür entsprechende globale Variablen definiert. Es sind mehrere Kontrollvariablen erforderlich. Die erste ist „ScrollJN“, welche zur Überprüfung dient, ob der Text gescrollt werden soll oder nicht. Dann folgt „Scrollposition“ zur Bestimmung der aktuellen Scrolltextposition und „Scrollanzeige“ zum Speichern der aktuellen Scrollzeile. Da der Scrollvorgang beim ersten Aufruf der Funktion starten soll, werden beide Variablen zu Beginn auf „1“ gesetzt. Die letzten beiden Variablen „ScrollArrayPos“ und „ScrollArrayAnz“ dienen zur Speicherung der genauen Scrolldaten für die Schleifen, welche in diesen Fällen die aktuelle Scrollposition und welche Zeilen gerade angezeigt werden sollen, wären.

Danach wird die eigentliche Funktion definiert. Der Wiedererkennbarkeit halber wird sie schlicht „Scrollen“ genannt.

```
function Scrollen()
```

Als nächstes überprüft man, ob für den aktuellen Frame bereits ein ScrollArray1 vorhanden ist. Diese Arrays werden bei jedem Scrollvorgang angelegt und sind nach dem Anlegen mit den Nummern der entsprechenden Frames versehen. Dafür wird der Befehl „_currentframe“ verwendet. Die komplette Befehlskette sieht wie folgt aus:

```
if(ScrollArrayPos[_currentframe] == undefined) .
```

Wenn das entsprechende Arrayelement noch nicht vorhanden ist, schreibt man die aktuellen Daten der Variablen „Scrollposition“ und „Scrollanzeige“ in die beiden vorgesehenen Arrays.

```
ScrollArrayPos[_currentframe] = Scrollposition;
```

```
ScrollArrayAnz[_currentframe] = Scrollanzeige;
```

Sobald die Animation in eine Schleife eintritt, ist die obige Abfrage nicht mehr zutreffend, da das Array bereits vorhanden ist, und der „else“-Zweig wird aktiv.

```
Scrollposition = ScrollArrayPos[_currentframe];
```

```
Scrollanzeige = ScrollArrayAnz[_currentframe];
```

An dieser Stelle erfolgt eine umgekehrte Zuweisung. Die benötigten Daten werden aus den bereits vorhandenen Arrays ausgelesen und in die Variablen „Scrollposition“ und „Scrollanzeige“ eingetragen.

Als nächstes folgt die Ausgabe für die Textfelder innerhalb der Animation. Für diesen Vorgang werden die beiden zuvor erstellten Arrays „AktionsDaten“ und „BefehlsDaten“ und die darin enthaltenen Werte benötigt. Die Textfelder besitzen die Instanznamen „Instruction1“ und

„Instruction2“. Innerhalb des Feldes „1“ sind die Aktionen aufgeführt und in dem Feld „2“ die entsprechenden Befehle. Um diesen Vorgang zu ermöglichen werden zwei Methoden miteinander verknüpft. Dies ist zum einen die „slice“-Methode, welche es ermöglicht vorbestimmte Inhalte aus einem Array zu extrahieren. Zum anderen ist das die „join“-Methode, die wiederum in der Lage ist, Arrays zu einem String zu verbinden.

Die folgende Befehlszeile bewirkt eine Ausgabe der Daten von Position 0 bis „Scrollanzeige“ aus dem „AktionsDaten“-Array innerhalb des Textfeldes „Instruction1“ und danach einen Zeilenumbruch, welcher durch die Angabe von „\n“ erfolgt.

```
Instruction1.text=AktionsDaten.slice(0, Scrollanzeige).join("\n");
```

Danach steht eine Zeile, welche bewirkt, dass der Text innerhalb von Textfeld 1 an die angegebene Position gescrollt werden soll.

```
Instruction1.scroll = Scrollposition;
```

Die beiden nachstehenden Befehlszeilen wurden analog für das Textfeld „Instructuion2“ erstellt und geben die Daten aus dem „BefehlsDaten“-Array aus.

```
Instruction2.text=BefehlsDaten.slice(0, Scrollanzeige).join("\n");
```

```
Instruction2.scroll = Scrollposition;
```

Am Ende der Scrollfunktion erfolgt, mittels einer „if“-Anweisung, eine Überprüfung der „ScrollJN“-Variable, ob diese immer noch gültig ist.

Wenn dies der Fall ist, werden die beiden Variablen „Scrollposition“ und „Scrollanzeige“ jeweils um eins erhöht.

Zum Abschluss erfolgt eine Überprüfung, ob der gesamte Scrollvorgang abgeschlossen ist oder nicht. Dazu wird der Inhalt der Variable Scrollanzeige zuerst mit der Länge des „AktionsDaten“-Arrays und danach mit der Länge des „BefehlsDaten“-Arrays angeglichen. In beiden Fällen wird die „ScrollJN“-Variable auf „false“ gesetzt, wenn die Zahlenwerte übereinstimmen oder gar größer sind. Die dafür benötigten Befehle lauten wie folgt:

```
if(Scrollanzeige >= AktionsDaten.length)
```

```
ScrollJN = false;
```

```
if(Scrollanzeige >= BefehlsDaten.length)
```

```
ScrollJN = false;.
```

Das komplette Script für die in diesem Abschnitt beschriebene Scrollfunktion befindet sich im Anhang unter der Nummer 13.

4.2.6 Bearbeitung der Textteile

Zu Beginn jeder Animation werden einige einleitende Wörter zu dem entsprechenden Thema präsentiert. Diese sind als Stichpunkte untereinander angeordnet. Als Abschluss zu jedem Thema werden noch einmal spezifische Daten zu der vorangegangenen Animation angezeigt. Die Realisierung der Texteinblendungen erfolgt über Textfelder.

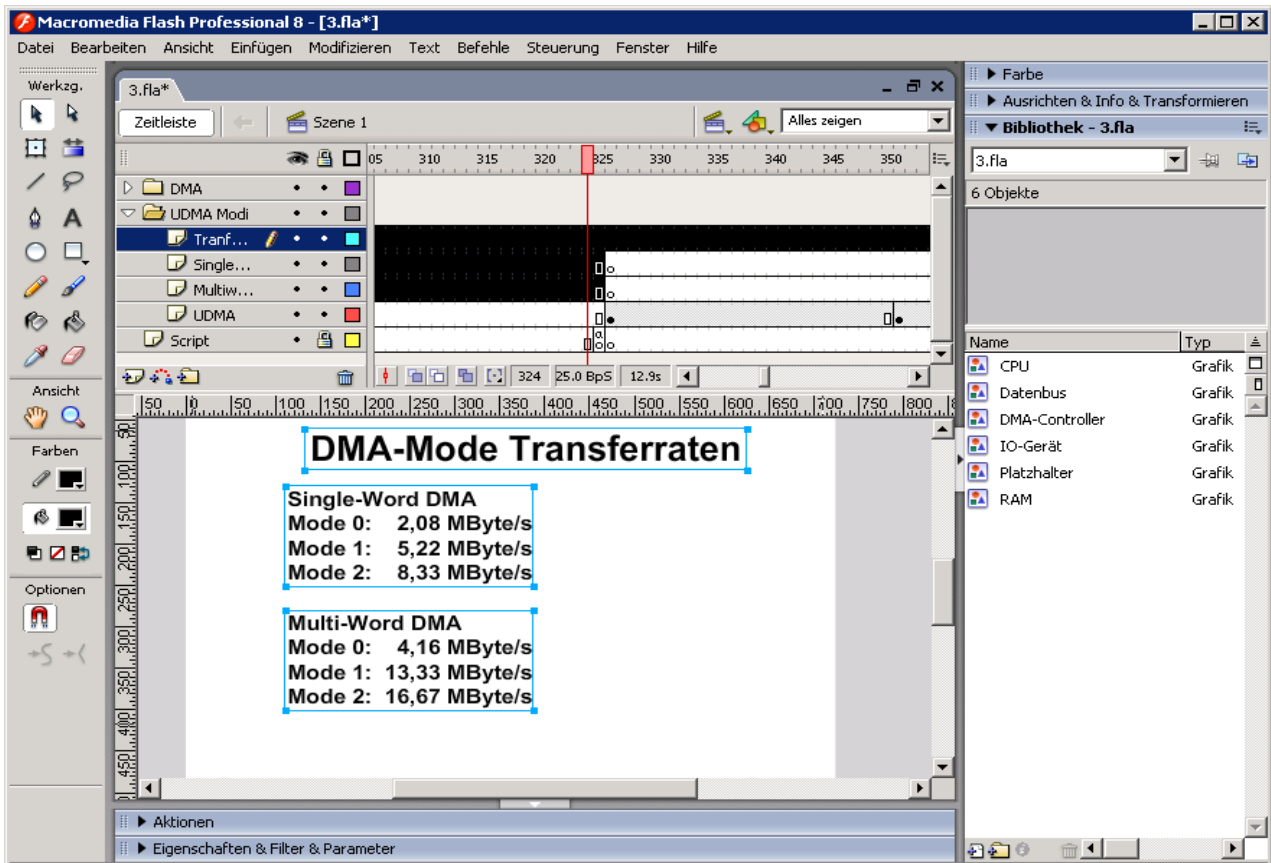


Abbildung 24: Bearbeitung der Textteile

Bei jedem Schlüsselbild wird das Textfeld erweitert, sodass es beim Abspielen zu der Illusion kommt, der Text würde automatisch nachrücken. Für den scrollenden Text am Ende der PIO- und DMA-Mode-Animation wäre es ebenfalls möglich gewesen, eine ähnliche Verfahrensweise wie bei dem zweiten Teil der PIO-Mode-Animation zu nutzen. Dies wurde jedoch aufgrund der geringen Anzahl von Scrollvorgängen nicht durchgeführt. Die Scriptmethode ist für größere Datensätze besser geeignet.

5. Zusammenfassung

Dieses Kapitel befasst sich mit den erzielten Ergebnissen der vorliegenden Arbeit und den Möglichkeiten von Erweiterungen und Verbesserungen für die Anwendung.

5.1 Ergebnisse und aktueller Stand

Ziel dieser Diplomarbeit war die Konzeption und Erstellung einer Präsentationsplattform für vorlesungsbegleitende Animationen. Des Weiteren sollten zwei Beispielanimationen erstellt werden, an denen der Schaffensprozess für nachfolgende Bearbeiter nachvollziehbar dargestellt werden kann.

Es wurden zunächst verschiedenste Möglichkeiten und Programme für Animationen vorgestellt, analysiert und anschließend eine Auswahl für die Realisierung getroffen.

Eine Erarbeitung von Konzepten über die Realisierung der Nutzeroberfläche und der Animationen schloss sich an. Dazu wurden die genauen Forderungen an die Anwendung im einzelnen formuliert, Storyboards angefertigt und eine Auswahl für die Farben getroffen.

Der praktische Teil der Arbeit befasste sich mit der Umsetzung der Oberfläche und der Animationen mittels Flash. Infolgedessen wurde eine Plattform geschaffen, die für die Präsentation von Vorlesungsunterlagen und Animationen geeignet ist. Es folgte das Anlegen einer Bibliothek mit vorgefertigten Objekten, deren Erweiterung jederzeit möglich ist. Es wurden außerdem zwei Beispielanimationen zum Thema PIO und DMA Mode erstellt, anhand derer der Schaffensprozess nachvollziehbar ist. Im Anschluss daran entstand ein Handbuch, das bestimmte Handlungsabfolgen und besondere Spezifikationen für spätere Projektbearbeiter aufzeigt.

Die Bibliothek ist zum Zeitpunkt der Arbeit noch begrenzt. Sie enthält bis jetzt überwiegend die Elemente für die Beispielanimationen, welche jedoch für weitere Animation im Stil des Blockschaltbildes Verwendung finden können. Ergänzt wurde diese durch Objekte zum Thema Speichermedien. Sobald weitere Animationen konzipiert und erstellt werden, erweitert sich der Umfang der Bibliothek.

Abschließend ist zu sagen, dass eine Anwendung geschaffen wurde, die in ihrem jetzigen Stadium vollständig genutzt werden kann. Inhaltlich sind jedoch Ergänzungen und Erweiterungen in Form von weiteren Animationen notwendig. Dafür wurde den Nutzern ein Handbuch mitgegeben, welches die wichtigsten Aspekte der Erstellung von Animationen noch einmal zusammenfasst.

5.2 Ausblick

Wie bereits im vorhergehenden Abschnitt erwähnt wurde, ist eine Anwendung inklusive zweier Beispielanimationen zu Themen der Computertechnik geschaffen worden. Für die Zukunft ist vorgesehen, dass man noch weitere Animationen zum Fach Computertechnik erstellt.

Durch den modularen Aufbau der Präsentationsoberfläche ist es leicht möglich, diese auch für andere Fächer (z.B.: Digitale Schaltungstechnik) zu modifizieren.

In diesem Zusammenhang kann die bisher bestehende Bibliothek mit den jeweiligen Inhalten erweitert werden, wodurch bei folgenden Animationen ein immer größerer Ressourcenpool verfügbar ist.

Da sich das Angebot möglicher Animationsprogramme in der schnelllebigen Computerbranche ständig verändert, ist nicht abzusehen welche Möglichkeiten zukünftig zur Verfügung stehen werden und inwieweit sich XML oder Flash entwickeln. Es ist ratsam, zukünftig zu prüfen, welche neuen Möglichkeiten der Animation sich bieten und eventuell Vereinfachungen oder Verbesserungen entsprechend umzusetzen.

6. Das Handbuch

Eine Komplett Einführung in das Programm Flash ist mit diesem Handbuch nicht beabsichtigt. Grundkenntnisse mit der Arbeit von Ebenen und der Zeitleiste von Flash werden hierbei vorausgesetzt. Man möchte kommenden Bearbeitern ein Werkzeug liefern, um ähnliche Animationen, wie die oben gezeigten, innerhalb kurzer Einarbeitungszeit erstellen zu können. Es wird schrittweise aufgezeigt, wie bei der Erstellung einer Animation vorzugehen ist und worauf in besonderem Maße zu achten ist.

6.1 Allgemeines

Als erstes müssen Vorüberlegungen zur gewünschten Animation angestellt werden. Dazu zählen die Anordnung der Objekte, Farben und andere Faktoren. Die Vorgehensweise dafür ist detailliert in Kapitel 3 beschrieben.

Um den Einstieg für die Erstellung neuer Animationen zu erleichtern, wurde bereits eine Vorlagendatei namens „Präsentation.fla“ angelegt. Es ist dringendst zu empfehlen, immer eine aktuelle Sicherungskopie dieser Datei anzulegen. Darin sind alle bereits vorgefertigten Elemente enthalten. Diese Elemente können somit auch für weitere Animationen verwendet werden, sofern sie dafür geeignet sind.

Bevor eine neue Animation erstellt wird, ist eine Kopie dieser Datei über die Funktion „Speichern unter“ im Dateimenü mit einem neuen Namen an der gewünschten Pfadposition abzuspeichern. Entsprechende Verzeichnispfade müssen vorher angelegt werden, um eine gewisse Ordnung unter den einzelnen Animationen zu erhalten. Des Weiteren dienen diese Pfade auch der Modularität mit der XML-Datei.

Bei der Erstellung weiterer Animationen sind die neuen Elemente, welche möglicherweise auch für andere Animationen von Nutzen sein können, in die Vorlagendatei zu übertragen, um sie für einen späteren Gebrauch verfügbar zu machen. Diese Vorgehensweise dabei ist wie folgt sehr einfach:

Man kopiert das gewünschte Element. Danach öffnet man die Vorlagendatei und fügt das Objekt in die Bibliothek ein. Das neue Element ist jetzt in die Vorlage übertragen worden. Danach ist die Vorlagendatei zu speichern.

6.2 Anlegen einer neuen Vorlage

Für den Fall, dass die Vorlagendatei nicht verfügbar ist, muss ein neues Flashdokument angelegt werden.

Für dieses sind eine Bildrate von 25 Bilder pro Sekunde (BpS) und eine Fläche von 725 x 600 Pixel für die einzelnen Animationen einzuhalten. Das ist notwendig, da die Präsentationsoberfläche ebenfalls mit dieser Framerate arbeitet und es sonst zu Verzögerungen oder Beschleunigungen während des Abspielens kommen würde. Die exakte Bildfläche ist nötig, da genau diese Größe als Anzeigefläche für die Animationen innerhalb der Anwendung reserviert wurde. (vgl. 3.2.3 und 4.1.1)

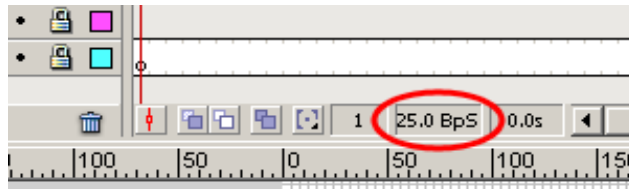


Abbildung 25: Zeitleiste

Um die entsprechenden Einstellungen vorzunehmen, ist ein Doppelklick mit der linken Maustaste auf die in Abbildung 25 gezeigte Fläche (25.0 BpS) erforderlich. Daraufhin erscheint das Dokumenteigenschaftenfenster, welches in Abbildung 26 zu sehen ist. Dort kann unter „Größe“ die Breite und Höhe der Animation auf die oben angegebenen Werte festgelegt werden. Ebenfalls in diesem Menü ist die Bildrate auf 25 BpS einzustellen.

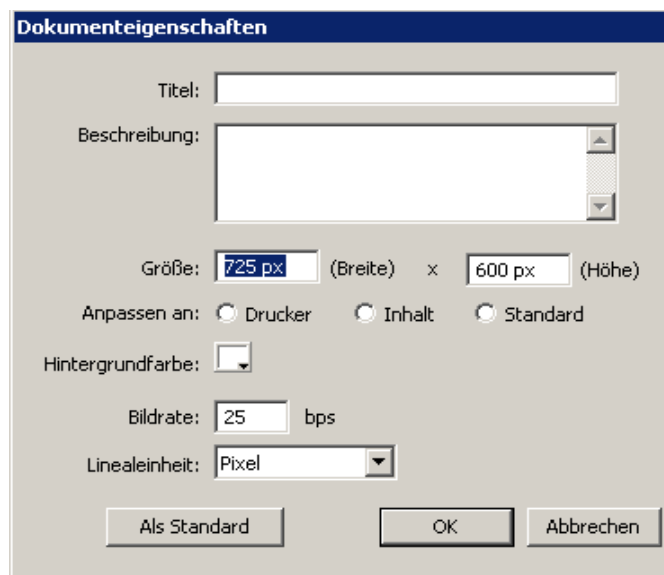


Abbildung 26: Dokumenteigenschaften

Anschließend werden die Einstellungen bestätigt und das Ergebnis in dem gewünschten Pfad auf der Festplatte gespeichert.

6.3 Erstellen von wiederverwendbaren Objekten

Um das Zusammenstellen neuer Animationen zu erleichtern wurde eine Bibliothek angelegt. Diese

ist beliebig erweiterbar. Das Vorgehen dabei ist bereits in Abschnitt 4.2.1 beschrieben worden.

Zuerst zeichnet man sich mit den von Flash gegebenen Werkzeugen das gewünschte Objekt. Dieses kann auch mehrere Ebenen enthalten (z.B. Grafik, Text) um später eine Bearbeitung zu erleichtern. Danach markiert man das gesamte Objekt und betätigt die rechte Maustaste darüber. Hier wählt man dann den Menüpunkt „In Symbol konvertieren“ aus. Darauf erscheint das Auswahlmenü, welches in Abb. 27 zu sehen ist. In diese Eingabemaske wird zuerst ein eindeutiger Name für das gewünschte Objekt festgelegt. Als nächstes muss man bestimmen, ob es sich um einen eigenständigen Movieclip, eine Schaltfläche oder eine Grafik handeln soll. Da die Objekte fast ausschließlich für die Animationen verwendet werden sollen, sind es in den meisten Fällen Grafiken. Man kann noch weitere Einstellungen vornehmen, die aber für die Erstellung der geplanten Animationen nicht relevant sind.

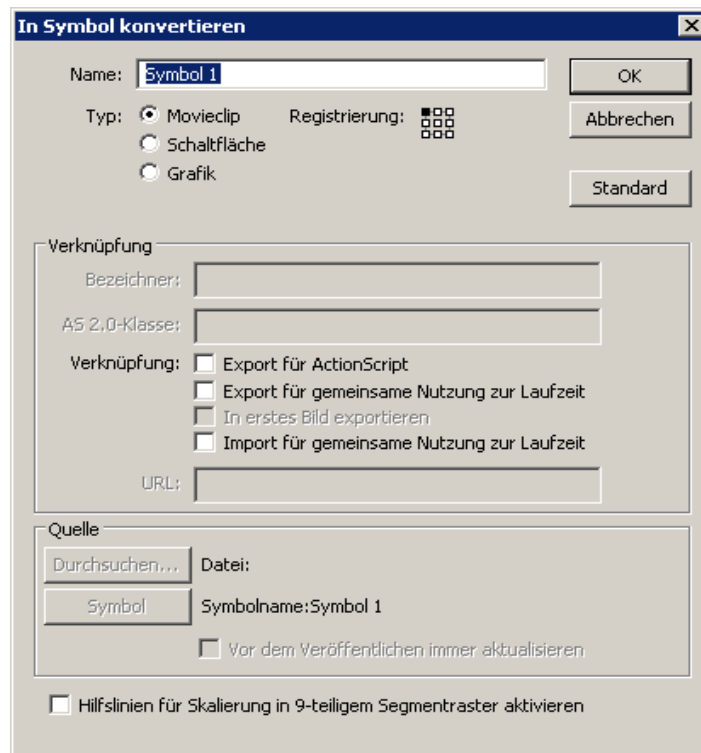


Abbildung 27: Symbolkonvertierung

Die gewählten Einstellung werden mit „OK“ bestätigt und das Objekt befindet sich jetzt in der Bibliothek und man kann es von dort jederzeit erneut bearbeiten.

6.4 Anpassung von Objekten

Vorgefertigte Objekte, welche als Symbole in der Bibliothek abgelegt sind, kann man ohne großen Aufwand verändern. Möglich sind hierbei die Größe, Farbe, Helligkeit und Transparenz

(Alphawert). Das Originalobjekt bleibt dabei unverändert. Dazu wählt man das Objekt auf der Bühne an und wählt das Eigenschaftenfenster (Abb. 28) aus.

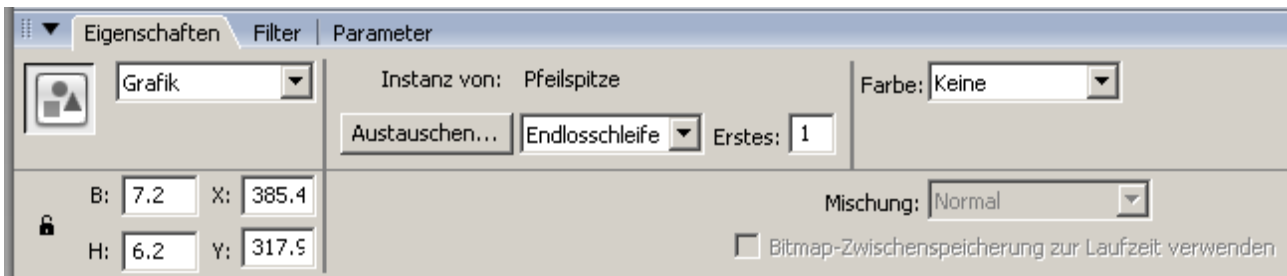


Abbildung 28: Objekteigenschaften

Über die Dropdownmenüs lassen sich die Eigenschaften des Objekts bestimmen. Das wichtigste hierbei ist das „Farbe“-Menü. An dieser Stelle werden die Helligkeit, der Farbton und die Transparenz (Alphawert) eingestellt. Es gibt auch einen „Erweitert“-Modus. In jenem können auch verschiedene Farbkombinationen zwischen Schrift und Hintergrund gewählt werden.

Die vier nebeneinander liegenden Eingabefelder dienen zur Skalierung und Positionierung der Objekte. Links untereinander stehen die Werte für die Breite (B) und die Höhe (H) des Objekts. Rechts daneben befinden sich die Eingabefelder für die Werte der horizontalen (X) und vertikalen (Y) Ausrichtung. Die grobe Positionierung kann mittels der Maus und Drag & Drop vorgenommen werden. Dazu selektiert man das gewünschte Objekt in der Bibliothek, hält die linke Maustaste gedrückt und zieht das Objekt auf die Bühne an die gewünschte Position.

6.5 Bewegungsanimation und Transparenz

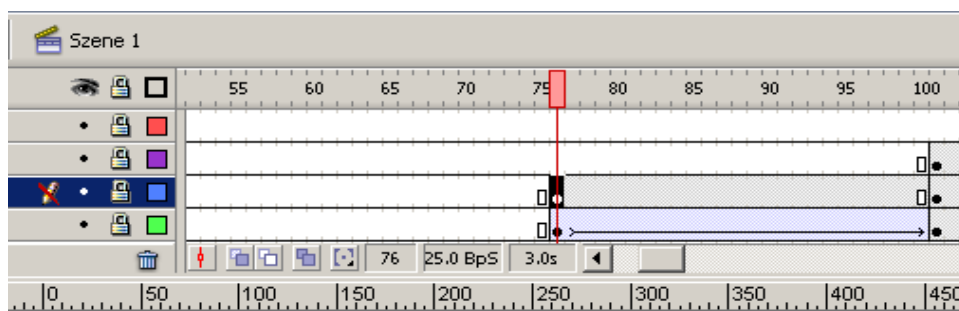


Abbildung 29: Zeitleistenzeiger

Um einen Pfeil, ein Quadrat oder etwas ähnliches über den Bildschirm zu bewegen, bedarf es weniger Handgriffe. Zuerst erstellt man das gewünschte Objekt. Danach fügt man an der Endposition des Objekts ein Schlüsselbild in der Zeitleiste ein und verschiebt das Objekt an die gewünschte Stelle. Nun setzt man den Zeitleistenzeiger auf die erste Position des gewünschten Objekts (Abb. 29) und wählt im Eigenschaftenmenü von Flash unter dem Menüpunkt Tween „Bewegung“ aus (Abb. 30). Danach interpoliert Flash automatisch den Bewegungsablauf von der

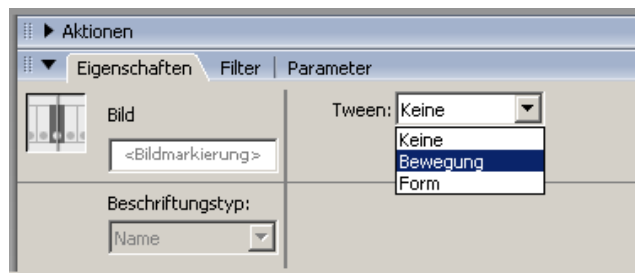


Abbildung 30: Bewegungstween

ersten Position zur gewünschten Endposition. Unter einen Bewegungstween fällt auch eine Änderung der Größe und der Transparenz eines Objektes wie es in den Beispielanimationen gezeigt wird. Um die Transparenz eines Objektes einzustellen, muss man den Alpha-Wert des Objekts ändern. Auch dies geschieht erneut über das Eigenschaftenmenü von Flash.

6.6 Scrollfunktion

Um die Scrollfunktion, welche innerhalb der PIO-Mode-Animation Verwendung findet, auch in anderen Animationen nutzen zu können, müssen einige Anpassungen daran vorgenommen werden. Die in der Arbeit verwendete Funktion ist auf zwei Textfelder ausgelegt. Nun ist es jedoch nicht immer erforderlich zwei Felder scrollen zu lassen. Es könnten es auch eins oder drei Felder sein, welche zu scrollen sind. Für diese Fälle müssen einige Scriptzeilen modifiziert werden. Das komplette Script ist im Anhang unter der Nummer 13 zu finden, weswegen hier nur Auszüge aufgeführt werden.

Für die Scrollfunktion sind die Textfelder in denen der Text zu scrollen ist, mit eindeutigen Namen zu versehen. Diese Namen müssen auch in der Scrollfunktion an den entsprechenden Stellen geändert werden. In den folgenden Scriptzeilen sind die zu ändernden Elemente fett markiert.

```
Instruction1.text = AktionsDaten.slice(0, Scrollanzeige).join("\n");
```

```
Instruction1.scroll = Scrollposition;
```

„Instruction1“ ist hierbei der Name des Textfelds. „Aktionsdaten“ ist das Array, in dem die zu scrollenden Textzeilen enthalten sind. Wie diese Daten in das Array geschrieben werden, ist in Abschnitt 4.2.5 ausführlich beschrieben.

Die folgende Scriptzeile ist auch anzupassen.

```
if(Scrollanzeige >= AktionsDaten.length) { ScrollJN = false; }
```

Man muss, um die Scrollfunktion an den entsprechenden Stellen zu aktivieren, Schlüsselbilder setzen. Es ist empfehlenswert für die Scrollschlüsselbilder eine separate Ebene zu erstellen, da es

sonst nicht ohne erheblichen Aufwand möglich ist, den Namen des Textfeldes zu ändern. Im praktischen Teil der Arbeit wurde dies aufgrund der Entwicklung leider nicht berücksichtigt.

Es ist auch möglich Textfelder unterschiedlich scrollen zu lassen. Dafür muss man lediglich eine zweite Funktion mit anderem Namen anlegen und auf das zweite Textfeld anpassen und die Funktion den gewünschten Stellen aktivieren. In der Beispielanimation sieht der Befehl zum aktivieren wie folgt aus: `Scrollen()` ; .

6.7 Pausen und Schleifen

Um Pausen oder Schleifen in die Animationen einzubinden, sind kurze Scriptbefehle an den gewünschten Stellen nötig. Dafür erstellt man idealerweise eine separate Ebene in der Zeitleiste. Dort werden an den entsprechenden Stellen Schlüsselbilder gesetzt und in diese die Scripte eingetragen. Für eine einfache Pause genügen die Befehle „`_root.PlayYN = false`“ und „`stop()`“. Danach bleibt die Animation an dieser Stelle stehen und läuft erst nach einem Druck auf die Pausetaste in der Anwendung weiter.

Für eine Schleife muss man zwei Schlüsselbilder bearbeiten. Zuerst fügt man am Beginn der Schleife das Script zum Setzen der Variable ein. „`_root.Schleife = false`“. Als zweites wird am Schleifenende eine Abfrage für diese Variable eingesetzt. „`if(!_root.Schleife) { gotoandplay(x) }`“. Das „`x`“ steht hierbei für eine ganze Zahl größer „0“. Auch hier kann die Schleife nur durch einen Druck auf die Schleifenabbruchtaste verlassen werden.

6.8 Formatierung der XML-Datei

Es ist notwendig, die Formatierung der XML-Dateien exakt einzuhalten. Die Verzeichnispfade müssen unbedingt mit einem Schrägstrich (Slash) als Trennzeichen versehen werden. Das ist nötig, da einerseits der Slash als Trennzeichen definiert ist und andererseits ein Backslash unter Umständen falsche Befehle in Flash ausführen könnte. Zum Beispiel bewirkt der Befehl `\n` in einem Textfeld den Umbruch einer neuen Zeile, was aber in den seltensten Fällen erwünscht ist. Aus dem selben Grund ist es erforderlich das Trennzeichen nicht innerhalb von Buttonbeschriftungen zu verwenden, da es dann zu einer ungewollten Trennung des Textes kommt.

Der Übersichtlichkeit halber ist es ratsam, mit Tabulatoren zu arbeiten. Diese beeinflussen die Nutzung der Datei nicht, da innerhalb des ActionScript die Leerzeichen und Tabulatoren per Befehl ignoriert werden können.

Es wird außerdem empfohlen, für Variablen und XML-Elemente eindeutige Namen zu verwenden,

um späteren Nutzern die Arbeit damit zu erleichtern.

Abgelegt werden die XML-Dateien im Stammverzeichnis für die „menu.xml“ bzw. in den jeweiligen Unterverzeichnissen der Animationen.

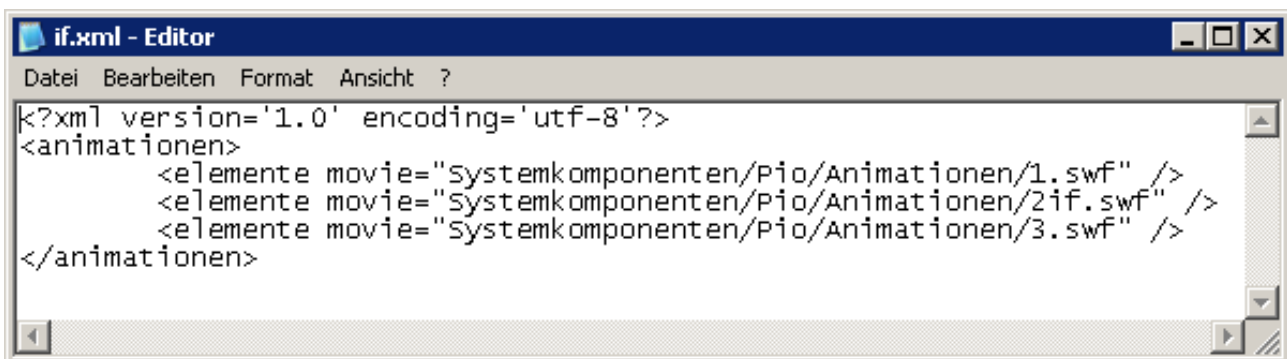
6.9 Änderung der Buttons

Da die Buttons innerhalb der Anwendung nur vorübergehend diese Anordnung haben, soll hier noch einmal erläutert werden, wie diese zu ändern ist.

Um die Änderungen vornehmen zu können, muss die „menu.xml“-Datei geöffnet werden. Darin sind sämtliche Verlinkungen und Buttons eingebettet. Die Buttons selbst sind die „Elemente“. Diese sind unterteilt in „Pfade“ und „Links“. Die Pfade dienen der Anwendung zur Erstellung des grafischen Buttons und die Links der Funktionalität. Um einen Button zu entfernen muss lediglich die entsprechende Zeile gelöscht werden. Wenn man einen neuen Button hinzufügen will, muss man umgekehrt einfach einen passenden Eintrag verfassen. Die Pfade sind dabei den Verzeichnispfaden auf der Festplatte nachempfunden und die Links sind die relativen Pfadangaben auf der Festplatte.

6.10 Anlegen der Playlisten

Die Playlisten der Animationen werden ebenfalls als XML-Dateien erstellt. In diesen sind jedoch nur direkte Links zu den Animationen abgelegt. In Abbildung 31 ist ein Beispiel für eine Playliste abgebildet.



```
if.xml - Editor
Datei Bearbeiten Format Ansicht ?
<?xml version='1.0' encoding='utf-8'?>
<animationen>
  <elemente movie="systemkomponenten/Pio/Animationen/1.swf" />
  <elemente movie="systemkomponenten/Pio/Animationen/2if.swf" />
  <elemente movie="systemkomponenten/Pio/Animationen/3.swf" />
</animationen>
```

Abbildung 31: Playliste

Die Listen sind in den Oberverzeichnissen der Animationen abgelegt und werden in der „menu.xml“ auch von da verlinkt.

6.11 Abschluss

Um eine Animation in der Anwendung nutzen zu können, muss man diese in eine SWF-Datei umwandeln. Dafür gibt es zwei Möglichkeiten. Zum einen wird bei jedem Test der Animation über das Steuerungsmenü von Flash eine solche Datei angelegt. Als zweite Möglichkeit kann man auch über das Dateimenü die Option „Veröffentlichen“ wählen. Hier wird ohne Vorschau eine SWF-Datei im Speicherordner der Bearbeitungsdatei erstellt.

Die Animation ist nun fertig und kann in der Anwendung abgespielt werden.

7. Anhang

Die folgenden Scripte werden in den Kapiteln 4.1.3.2 ff. ausführlich erläutert.

Script 1 – menu.fla – Frame 1

```
var      Content:MovieClip      =      this.createEmptyMovieClip("Content",
this.getNextHighestDepth());
Content._x = 250; Content._y = 100;

var      Menuleiste:MovieClip   =      this.createEmptyMovieClip("Menuleiste",
this.getNextHighestDepth());
var MenuPfad = "";
var Elemente = new Array();
var Playlistpfad;
var Nummer:Number;
var Fertig:Boolean = false;
var PlayYN:Boolean = false;
var Schleife:Boolean = true;
var KeyListener:Object = new Object();
KeyListener.onKeyDown = function() {
if((Key.getCode() == Key.SPACE) && (!_root.Fertig)) {
    if (PlayYN) {
        _root.Content.stop();
        _root.Content.Content.stop();
        PlayYN = false;
    } else {
        _root.Content.play();
        _root.Content.Content.play();
        PlayYN = true;
    }
}
}
if (Key.getCode() == Key.DOWN) {
    Schleife = true;
}
if (Key.getCode() == Key.LEFT) {
    Nummer--;
    Content.gotoAndPlay(2);
}
if (Key.getCode() == Key.RIGHT) {
    Nummer++;
    Content.gotoAndPlay(2);
}
```

```

    }
};
Key.addListener(KeyListener);

var XMLDaten = new XML();
XMLDaten.ignoreWhite=true;
XMLDaten.load("menu.xml");
XMLDaten.onLoad = function(success) {
    if (success){
        AnzahlElemente = XMLDaten.firstChild.childNodes.length;
        var j = 0;
        for (var i = 0; i < AnzahlElemente; i++) {
            if (XMLDaten.firstChild.childNodes[i].attributes.tz) {
                Tz = XMLDaten.firstChild.childNodes[i].attributes.tz;
            }
            else if (XMLDaten.firstChild.childNodes[i].attributes.hl) {
                Hl = XMLDaten.firstChild.childNodes[i].attributes.hl;
            }
            else {
                Elemente[j] = new Object();
                Pfad = XMLDaten.firstChild.childNodes[i].attributes.pfad;
                var LastTz = Pfad.lastIndexOf(Tz);

                Elemente[j].Pfad = Pfad.substring(0, LastTz +
1).split(Tz);
                Elemente[j].Bezeichnung = Pfad.substring(LastTz + 1,
Pfad.length);
                Elemente[j].link=
XMLDaten.firstChild.childNodes[i].attributes.link;
                j++;
            }
        }
        gotoAndStop(5);
    }
}
stop();

```

Script 2 – menu.fla – Frame 5

```
Menuleiste.removeMovieClip();
var Menuleiste:MovieClip = this.createEmptyMovieClip("Menuleiste",
this.getNextHighestDepth());
Headline = H1;
var Linkbuttons = new Array();
var Kategorien = new Array();
var Buttons = new Array();

for (i = 0; i < Elemente.length; i++) {
    ElementPfad = Elemente[i].Pfad;
    if (ElementPfad.indexOf(MenuPfad) == 0) {
        var RestPfad = ElementPfad.substring(MenuPfad.length,
ElementPfad.length);
        if (RestPfad != "") {
            var Kategorie = RestPfad.substring(0, RestPfad.indexOf(Tz));
            var NeueKategorie = "Ja";
            for (j = 0; j < Kategorien.length; j++) {
                if(Kategorie == Kategorien[j]) {
                    NeueKategorie = "Nein";
                    break; }
            }
            if (NeueKategorie == "Ja") {
                Kategorien.push(Kategorie); }
        } else { Linkbuttons.push(i); }
    }
}
var i = 0;
if (MenuPfad != "") {
    Buttons[i] = Menuleiste.attachMovie("Button1", "Zurueck",
Menuleiste.getNextHighestDepth());
    Buttons[i].Textfeld = "<< Zurück";
    Buttons[i]._x = 30;
    Buttons[i]._y = 125 + ( i * 40 );
    Buttons[i].onDragOut = function() {
        this.gotoAndPlay(1);}
    Buttons[i].onPress = function() {
        this.gotoAndPlay(2);}
    Buttons[i].onRelease = function() {
        this.gotoAndPlay(1);
        MenuPfad = MenuPfad.split(Tz);
```

```

        MenuPfad.splice(MenuPfad.length - 2);
        Anzeige = "*" + MenuPfad.slice(-1) + "*";
        MenuPfad = MenuPfad.join(Tz) + Tz;
        if(MenuPfad == Tz) {
            MenuPfad = "";
            Anzeige = "";}
        gotoAndPlay(4);    }
    i++; }
for (j = 0; j < Kategorien.length; j++) {
    Buttons[i] = Menuleiste.attachMovie("Button1", "Button1" + i,
Menuleiste.getNextHighestDepth());
    Buttons[i].Textfeld = "* " + Kategorien[j] + " *";
    Buttons[i].Kategorie = Kategorien[j];
    Buttons[i]._x = 30;
    Buttons[i]._y = 125 + ( i * 40 );
    Buttons[i].onDragOut = function() {
        this.gotoAndPlay(1); }
    Buttons[i].onPress = function() {
        this.gotoAndPlay(2); }
    Buttons[i].onRelease = function() {
        this.gotoAndPlay(1);
        Anzeige = this.Textfeld;
        MenuPfad = MenuPfad + this.Kategorie + Tz;
        gotoAndPlay(4);    }
    i++; }
for (j = 0; j < Linkbuttons.length; j++) {
    Buttons[i] = Menuleiste.attachMovie("Button1", "Button1" + i,
Menuleiste.getNextHighestDepth());
    Buttons[i].Textfeld = Elemente[Linkbuttons[j]].Bezeichnung;
    Buttons[i].Link = Elemente[Linkbuttons[j]].link;
    Buttons[i]._x = 30;
    Buttons[i]._y = 125 + ( i * 40 );
    Buttons[i].onDragOut = function() {
        this.gotoAndPlay(1); }
    Buttons[i].onPress = function() {
        this.gotoAndPlay(2); }
    Buttons[i].onRelease = function() {
        for(k = 0; k < Buttons.length; k++) {
            Buttons[k].gotoAndStop(1); }
        this.gotoAndPlay(3);
        var Endung = this.Link.substr(-3,3);

```



```

        if(Endung == "xml") {
            Playlistpfad = this.Link;
            _root.Content.loadMovie("playlistcontainer.swf") }
            else {_root.Content.loadMovie(this.Link);}
        _root.Fertig = false;
        _root.PlayYN = true; }
    i++; }
stop();

```

Script 3 – menu.fla – PlayController - zurück

```

on(release) {
    _root.Nummer--;
    _root.Content.gotoAndPlay(2);}

```

Script 4 – menu.fla – PlayController - Play/Pause

```

on(release) {
    if (!_root.Schleife) {
        _root.Schleife = true;
    } else {
        if (_root.PlayYN) {
            _root.Content.stop();
            _root.Content.Content.stop();
            _root.PlayYN = false;
        } else {
            _root.Content.play();
            _root.Content.Content.play();
            _root.PlayYN = true;}
        }
}

```

Script 5 – menu.fla – PlayController - vorwärts

```

on(release) {
    _root.Nummer++;
    _root.Content.gotoAndPlay(2);}

```

Script 6 – menu.fla – PlayController - Animationspause/play

```
on(release) {  
    if (_root.PlayYN) {  
        _root.Content.stop();  
        _root.Content.Content.stop();  
        _root.PlayYN = false;  
    } else {  
        _root.Content.play();  
        _root.Content.Content.play();  
        _root.PlayYN = true;}  
}
```

Script 7 – menu.fla – PlayController

```
if (_root.PlayYN) {  
    _root.PlayPause = " ";  
    else { _root.PlayPause = "4";}  
  
if (!_root.Schleife) {  
    _root.PlayPause = "4";}  
  
if (_root.Schleife) {  
    _root.CtrlSchleife = " ";  
    else {  
        if (_root.PlayYN) {  
            _root.CtrlSchleife = "<";    }  
        else {  
            _root.CtrlSchleife = ":";    }  
    }  
}
```

Script 8 – playlistcontainer fla – Frame 1

```
var Content:MovieClip = this.createEmptyMovieClip("Content",
this.getNextHighestDepth());
Content._x = 0; Content._y = 0;

var XMLDaten = new XML();
var Mov = new Array();
_root.Nummer = 0;

XMLDaten.ignoreWhite=true;
XMLDaten.load(_root.Playlistpfad);
XMLDaten.onLoad = function(success) {
    if (success){
        AnzahlMovies = XMLDaten.firstChild.childNodes.length;
        for (var i = 0; i < AnzahlMovies; i++) {
            Mov[i] = new Object;
            Mov[i].movie
XMLDaten.firstChild.childNodes[i].attributes.movie;
        }
        gotoAndPlay(2);
    }
}
stop();
```

Script 9 – playlistcontainer fla – Frame 2

```
if (_root.Nummer >= AnzahlMovies) { _root.Nummer = AnzahlMovies - 1; }
if (_root.Nummer < 0) { _root.Nummer = 0; }
Content.loadMovie(Mov[_root.Nummer].movie);
```

Script 10 – playlistcontainer fla – Frame 3

```
gesamt = Content._totalframes;
aktuell = Content._currentframe;

if (aktuell == gesamt) {
    _root.Nummer++;
    if (_root.Nummer < Mov.length) {
        gotoAndPlay(2);    }
    else { gotoAndPlay(5); }
}
```

Script 11 – playlistcontainer.fla – Frame 5

```
_root.Fertig = true;
Content.stop();
stop();
```

Script 12 – PIO Mode 2if.fla – Frame 151

Da sich die Programmzeilen stetig wiederholen, wird hier nur ein Ausschnitt dargestellt.

```
var AktionsDaten = new Array();
var BefehlsDaten = new Array();
AktionsDaten.push("1) Instruction Fetch");
AktionsDaten.push("2) Execute");
... //Fortführung bis zur letzten Befehlszeile (in diesem Fall insgesamt 80 Zeilen)
AktionsDaten.push("80) Execute");
BefehlsDaten.push("mov reg, dvd");
BefehlsDaten.push("mov reg, dvd");
... //Fortführung bis zur letzten Programmcodezeile (ebenfalls 80 Zeilen)
BefehlsDaten.push("mov hd, reg");
```

Script 13 – PIO Mode 2if.fla – Frame 152

```
_global.Scrollposition = 1;
_global.Scrollanzeige = 1;
_global.ScrollJN = true;
_global.ScrollArrayPos = Array();
_global.ScrollArrayAnz = Array();

function Scrollen() {
    if (ScrollArrayPos[_currentframe] == undefined) {
        ScrollArrayPos[_currentframe] = Scrollposition;
        ScrollArrayAnz[_currentframe] = Scrollanzeige;
    } else {
        Scrollposition = ScrollArray1[_currentframe];
        Scrollanzeige = ScrollArray2[_currentframe];
    }

    Instruction1.text = AktionsDaten.slice(0, Scrollanzeige).join("\n");
    Instruction1.scroll = Scrollposition;
    Instruction2.text = BefehlsDaten.slice(0, Scrollanzeige).join("\n");
    Instruction2.scroll = Scrollposition;
```

```
if (ScrollJN) {
    Scrollposition++;
    Scrollanzeige++;
}
if (Scrollanzeige >= AktionsDaten.length) {
    ScrollJN = false;
}
if (Scrollanzeige >= BefehlsDaten.length) {
    ScrollJN = false;
}
}
```

8. Literaturverzeichnis

- [1] Tony Stockwell (2007) "Was für ein Lerntyp sind Sie?" erschienen im NeulandMAGAZIN Frühling 2007 <http://neuland.ch>
- [2] Dr. Werner Stangl (2009)
<http://www.stangl-taller.at/ARBEITSBLAETTER/LERNEN/Lerntypen.shtml>
- [3] Ronald E. LaPorte, Faina Linkov, Tony Villaseñor, Francois Sauer, Carlos Gamboa, Mita Lovalekar, Eugene Shubnikov, Akira Sekikawa and Eun Ryoung Sa (2002) "Papyrus to PowerPoint (P 2 P): metamorphosis of scientific communication"
- [4] Elsa Wenzel (2007) http://reviews.cnet.com/presentation/microsoft-powerpoint-2007/4505-3525_7-32143058.html
- [5] PC Advisor (2007) <http://www.pcadvisor.co.uk/reviews/index.cfm?reviewid=643>
- [6] ComputerBild (2007) http://media.computerbild.de/downloads/2084587/Testtabelle_Powerpoint_gegen_Impress.png
- [7] Tim Kaufmann (2008) <http://software.magnus.de/tools/artikel/adobe-director-11.html>
- [8] Michael Burns (2007) <http://www.digitalartsonline.co.uk/reviews/index.cfm?reviewID=802>
- [9] ADOBE (2009) http://www.adobe.com/products/player_census/flashplayer/
- [10] PC Welt (2005) <http://www.pcwelt.de/index.cfm?pid=1506&pk=117745>
- [11] FlashForum (2007) <http://www.flashforum.de/forum/showthread.php?t=138323>

9. Selbstständigkeitserklärung

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

Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Mittweida, den 28. August 2009