Bachelorarbeit

TPHC

(Tutorial Programm for Hardware Change)

09. August 2013

Hochschule für angewandte Wissenschaft und Kunst
Fakultät Naturwissenschaften und Technik

Name:	Jannik Mewes	Paul Lindegrün
Matrikelnummer:	511081	508683
Studiengang:	Elektro- und Informationstechnik	Elektro- und Informationstechnik
Fachrichtung:	Medien- und Kommunikationssysteme	Medien- und Kommunikationssysteme
Erstprüfer:	Prof. Dr. Ing. Achim Ibenthal	Prof. Dr. Ing. Achim Ibenthal
Zweitprüfer:	Dipl. Ing. FH Tobias Bürmann	Dipl. Ing. FH Tobias Bürmann

Bachelorarbeit:

Vorwort 7

Glossar 8

1 Einleitung 10

1.1 Jannik Mewes 10
1.1.1 Aufgabenstellung 10
1.1.2 Einsatzmöglichkeiten 10
1.1.3 Vorteile gegenüber *YouTube*-Videos 10
1.1.4 3D-Modelle 11
1.1.5 Animation 11
1.1.6 Zusammenfassung des Kapitels 12
1.2 Paul Lindegrün 12
1.2.1 Programm 12
1.2.2 Einbindung der Videos in das Programm 12
1.2.3 Interface-Design 12
1.3 Kapitelübersicht 13

2 Systemüberblick 14

2.1 Jannik Mewes 14
2.1.1 Aufgabenaufteilung 14
2.1.2 Ablaufplan (Praxisprojekt) 15
2.1.3 Zeitplan 16
2.1.4 Zusammenfassung des Kapitels 17
2.2 Paul Lindegrün 18
2.2.1 Beschreibung des Projektumfeldes 18
2.2.2 Anforderungen an die Software 18
2.2.3 Zusammenfassung des Kapitels 19

3 Recherche 20

3.1 Jannik Mewes 20
3.1.1 Aktuelle Anschlusssports 20
3.1.2 Modellierungs-Programmauswahl 21
3.1.3 Videobearbeitungs-Programmauswahl 22
3.1.4 Bildbearbeitungs-Programmauswahl 23
3.1.5 Benötigte Arbeitsmaterialien 23

3.1.6 Kontaktaufbau zu Computer-Hardware-Unternehmen 24
3.1.7 Zusammenfassung des Kapitels 24
3.2 Paul Lindegrün 25
3.2.1 Auswahl der Programmiersprache 25
3.2.2 Auswahl der Entwicklungsumgebung 27
3.2.3 Design Patterns 28
3.2.4 Auswahl des Codecs 29
3.2.5 Auswahl des Videocontainers 30
3.2.6 Auswahl des Videoplayers 31
3.2.7 Interface Entwürfe 32
3.2.8 Zusammenfassung des Kapitels 34
4 Konzeptentwicklung 35
4.1 Jannik Mewes 35
4.1.1 Auswahl des Mustercomputers 36
4.1.2 Animationsentwürfe 37
4.1.2.1 Arbeitsspeicher 37
4.1.2.2 Festplatte 38
4.1.2.3 Grafikkarte 38
4.1.2.4 Mainboard 39
4.1.2.5 Prozessor 40
4.1.2.6 Netzteil 40
4.1.3 Schematischer Ablauf einer 3D-Modellierung 41
4.1.4 Orthogonale Ansicht 45
4.1.5 Ablaufplan (Erstellung von Anleitungsvideos) 48
4.1.6 Ablaufplan (Erstellung des Userinterface-Designs) 49
4.1.7 Probandentest 50
4.1.8 Zusammenfassung des Kapitels 50
4.2 Paul Lindegrün 51
4.2.1 Anwendungsfälle (use cases) 51
4.2.2 Aufbau des Interfaces 52
4.2.3 Schematischer Ablauf des Programms 54
4.2.4 Ablaufplan der Programmierung 56
4.2.5 Zusammenfassung des Kapitels 57
5 Implementierung 58
5.1 Jannik Mewes 58
5.1.1 Fotografieren und vermessen des zu modellierenden Computers 58
5.1.2 3D-Modellierung von PC-Bestandteilen 59

5.1.2.1 Beispiel der 3D-Modellierung von einer Gewindeschraube....59
5.1.2.2 Beispiel der 3D-Modellierung von eines geriffelten Objekts....61
5.1.2.3 Beispiel der 3D-Modellierung von Kabeln....61
5.1.2.4 Erstellung von Texturen für die Videos mit Photoshop....62
5.1.2.5 Mainboard....63
5.1.2.6 RAM....65
5.1.2.7 Grafikkarte NVIDIA....66
5.1.2.8 Netzteil....67
5.1.2.9 CPU....68
5.1.2.10 Festplatte (HDD)....68
5.1.2.11 DVD-Laufwerk....69
5.1.2.12 *Antec*-Gehäuse....70
5.1.3 Zusammenfügen der modellierten Einzelteile....71
5.1.3.1 Komplett-PC....71
5.1.3.2 Raumgestaltung....73
5.1.4 Lichteffekte und Schatten....74
5.1.4.1 Licht-Einstellungen....74
5.1.4.2 Schatten-Einstellungen....75
5.1.5 Animation der einzelnen Videos....75
5.1.5.1 RAM-Wechsel....79
5.1.5.2 HDD-Einbau....80
5.1.5.3 Grafikkarten-Einbau....80
5.1.5.4 DVD-Laufwerk-Wechsel....81
5.1.5.5 Netzteil-Einbau....81
5.1.5.6 Mainboard-Wechsel....82
5.1.5.7 Prozessor-Wechsel....83
5.1.5.8 Komplettbau....84
5.1.6 Rendern und Render-Einstellungen....85
5.1.7 Userinterface-Design....86
5.1.8 Video Bearbeitung....91
5.1.8.1 Videobearbeitung mit Adobe Premiere Pro CS6....91
5.1.8.2 Erstellung von Texturen für die Videos mit Photoshop....93
5.1.9 Problemstellungen sowie Lösungswege....93
5.1.9.1 Von 3ds-Max 2010 zu 3ds-Max 2014....93
5.1.9.2 Boolean-Löcher....94
5.1.10 Zusammenfassung des Kapitels....96
5.2 Paul Lindegrün....96
5.2.1 Installation der Software....96
5.2.2 Erstellung des Grundgerüstes....96

5.2.2.1 Steuerelemente aus Button-Klasse 97
5.2.2.2 Erstellung der MyButton-Klasse 99
5.2.3 Erweiterung des Programms 100
5.2.3.1 Erweiterung der TPHC-Klasse 100
5.2.3.2 Erstellung der InfoForm-Klasse 102
5.2.3.3 Erstellung der DoubleBufferPanel-Klasse 103
5.2.3.4 Erstellung der InfoPanel-Klasse 103
5.2.3.5 Einbindung von VideoLan DotNet-Bibliotheken 105
5.2.3.6 Erstellung der Volume-Klasse 106
5.2.3.7 Erstellung der Timeline-Klasse 107
5.2.3.8 Implementierung des Players 109
5.2.3.9 Erweiterung der MyButton-Klasse 109
5.2.3.10 Tastatureingaben 110
5.2.4 Userinterface-Design 112
5.2.4.1 Version 1 112
5.2.4.2 Version 2 114
5.2.4.3 Version 3 116
5.2.5 Zusammenfassung des Kapitels 118
5.3 Implementierung der Einzelergebnisse 118
6 Testphase 122
6.1 Jannik Mewes 122
6.1.1 Probandentest 122
6.1.1.1 Resultat Videoverständnis 124
6.1.1.2 Resultat Userinterface 125
6.1.2 Zusammenfassung des Kapitels 125
6.2 Paul Lindegrün 125
6.2.1 CPU-Auslastung 125
6.2.2 Speicher-Auslastung 126
6.2.3 Stabilität des Programmes 126
6.2.4 Starten unter verschiedenen Windows Versionen 127
6.2.5 Helligkeits-Kontrast-Sättigungs-Einstellungen 127
6.2.6 Zusammenfassung des Kapitels 129
7 Zusammenfassung 130
7.1 Jannik Mewes 130
7.1.1 Ergebnis 130
7.1.2 Ausblick 130
7.2 Paul Lindegrün 131

7.2.1 Ergebnis 131
7.2.2 Ausblick 132

8 Literaturverzeichnis 133

8.1 Jannik Mewes 133
8.2 Paul Lindegrün 134

Vorwort

Die Komplexität der elektrischen Schaltkreise in einem Computer wirkt auf viele Menschen abschreckend. Aus Sorge, den Computer zu beschädigen oder einen Stromschlag zu erhalten, trauen sich viele Menschen nicht, das Gehäuse zu öffnen und Bauelemente anzufassen.

Aus diesem Grund werden bereits bei kleineren Umbauten (z.B. Austausch des Arbeitsspeichers) PC-Werkstätten in Anspruch genommen, die sich diese Dienstleistungen teuer bezahlen lassen.

Um Menschen die Angst vor dem Innenleben eines Computers zu nehmen und zu zeigen, dass die allgemeine Struktur eines PCs immer gleich aufgebaut ist, beschloss ich (Jannik Mewes) ein Tutorial-Programm für den Umbau der einzelnen PC-Bestandteile im Rahmen einer Bachelorarbeit zu entwickeln.

Die Bachelorarbeit befasst sich mit der von uns erstellten Programmierung und Animation von 3D-Modellen sowie mit der Entwicklung eines benutzerfreundlichen Userinterfaces.

Glossar

Abkürzung	Erklärung
AGP	Accelerated Graphics Port
AMD	Advanced Micro Devices
ATX	Advanced Technology Extended
AVI	Audio Video Interleave
CPU	Central Processing Unit
CRF	Constant Rate Factor
DDR-RAM	Double Data Rate - Random Access Memory
DVD	Digital Versatile Disc
DVI	Digital Visual Interface
ECMA	European Computer Manufacturers Association
eSATA	External Serial Advanced Technology Attachment
FLV	Flash Video
FPS	Frames Per Second
GB	Giga Byte
HDD	Hard Disk Drive
IDE	Integrated Drive Electronics
JDK	Java Development Kit
JPG	Joint Photographic Experts Group
JRE	Java Runtime Environment
LED	Light-Emitting Diode
MP4	Kurzform von MPEG-4
MPC-HC	Media Player Classic – Home Cinema
MPEG	Moving Picture Experts Group
MSDN	Microsoft Developer Network
PC	Personal Computer
PCI	Peripheral Component Interconnect
PCI-Express	Peripheral Component Interconnect Express
PNG	Portable Network Graphics
PS/2	Personal System/2
RAM	Random Access Memory
SATA	Serial Advanced Technology Attachment
TGA	Truevision Advanced Raster Graphics Array
TPHC	Tutorial Programm for Hardware Change
USB	Universal Serial Bus
VGA	Video Graphics Array
VLC	VideoLAN ehem. VideoLAN Client
WinAPI	Windows Application Programming Interface
WMP	Windows Media Player
WPF	Windows Presentation Foundation

1 Einleitung

1.1 Jannik Mewes

Die folgende Einleitung soll einen kurzen Überblick über das gesamte Projekt geben. Es werden Aufgabenpunkte definiert und einleitende Erläuterungen zu den einzelnen Bestandteilen der Arbeit gegeben.

1.1.1 Aufgabenstellung

Realisiert werden soll ein selbstständig laufendes Programm, welches es ermöglicht, Kunden, die wenig Vorkenntnisse von PC-Hardware besitzen, zu unterstützen, Grundbestandteile des PCs zu erlernen und defekte Hardware auszutauschen oder aufzurüsten.

Hierzu soll ein Interface für den PC entstehen, welches der User ohne Installation starten kann. Es öffnet sich ein Interface mit verschiedenen Auswahlmöglichkeiten. Für definierte Problemstellungen stehen entsprechende Videos zur Verfügung.

1.1.2 Einsatzmöglichkeiten

Es gibt vielfache Einsatzmöglichkeiten für ein derartiges Programm. Denkbar wären:

- Mitgelieferte Software zur Anleitung, wie man ein Produkt richtig installiert.
- Nutzung für Werbezwecke, wobei z. B. ein Gehäuse-Hersteller sein Produkt als 3D-Modell einbindet und durch Einbauvideos Vorteile optisch präsentieren kann.
- Diese Videos sind für Allgemeinbildende Schulen oder als Unterstützung für Vorlesungen geeignet. Hierbei kann das Programm zur Visualisierung genutzt werden.
- Einsatz für das Praktikum des Vorlesungsfaches „Netzwerksysteme“.

1.1.3 Vorteile gegenüber *YouTube*-Videos

Ein solches Programm bietet gegenüber Anleitungsvideos von *Youtube* erhebliche Vorteile:

- die Sprachenunabhängigkeit (eine kleine Texttabelle genügt zum Übersetzen)
- der Kunde benötigt keinen Internetzugang
- hochwertigere und professionellere Darstellung als *Youtube*-Videos
- bessere Visualisierung kritischer Bereiche durch nahe Kamerafahrten und rot aufleuchtende Objekte

- Verwendung als Lehrmaterial in Universitäten und Schulen
- Einbindung firmeneigener Produkte möglich
- Anleitungsvideos als imagefördernde Produktbeigaben
- einheitliches Design
- Werbeplattform für Unternehmen, z.B. Gehäuse-Hersteller

1.1.4 3D-Modelle

Zuerst müssen viele, möglichst detailgetreue 3D-Modelle erstellt werden. Diese müssen einheitliche Größen aufweisen, um die Passgenauigkeit zu gewährleisten. Außerdem müssen 3D-Modelle aller gängigen Anschlüsse erzeugt werden. Hierzu sollten verschiedene Unternehmen angeschrieben werden, ob sie im Rahmen dieses Projektes bzw. dieser Bachelorarbeit eventuell schon fertige 3D-Modelle zur Verfügung stellen können oder gegebenenfalls Datenblätter mit Größeninformationen sowie Layouts der unbestückten Platinen. Erläutert wird ein grundlegender Ablauf der 3D-Modellierungen, im Kapitel 4.1.4. Die konkrete Modellierung der einzelnen Komponenten wird im Kapitel 5.1.2 beschrieben.

Für die Umsetzung eines 3D animierten Videos sind viele Bestandteile zu vermessen, z. B. das PC-Gehäuse. Die Vermessung der Bauteile wird im Kapitel 5.1.1 beschrieben.

Viele einzelne Bauteile, die auf die Platinen aufgesetzt werden, müssen ebenfalls erstellt werden.

1.1.5 Animation

Nach Fertigstellung der 3D-Modelle müssen sie sinnvoll zusammengesetzt und für jeden einzelnen Fall animiert werden. Kritische Bereiche sollen rot aufleuchten und die Kamerafahrten müssen zweckmäßig gestaltet werden. Dies wird im Kapitel 4.1.3 und Kapitel 5.1.5 beschrieben.

Es ist ein Evaluationsverfahren geplant. Probanden sollen mit Hilfe der Videos, Bauteile im PC[1] zu Testzwecken austauschen. Die Ergebnisse werden dokumentiert und ausgewertet. Anschließend werden die getesteten Videos gerendert und sinnvoll tabellarisiert. Auf den Probandentest wird im und Kapitel 6.1.1 eingegangen.

[1] PC (engl.: Personal Computer)

1.1.6 Zusammenfassung des Kapitels

In diesem Kapitel wurden die Aufgabenbereiche der Praxisprojektphase im Einzelnen definiert. Es wurde aufgezeigt, welche Anforderungen das Gesamtsystem stellt und nach welchen Prinzipien die Aufgabengebiete zu lösen sind. Im nächsten Kapitel werden die in diesem Abschnitt besprochenen Einzelaufgaben auf zwei Studenten aufgeteilt. Ebenfalls wird ein zeitlicher Ablauf erläutert.

1.2 Paul Lindegrün

1.2.1 Programm

Programmiert wird ein Interface zur Videoabspielung, das ohne Installation gestartet werden kann. Das Programm an sich stellt eine Art Videoplayer dar, welches ein ausgewähltes Video startet und passiert. Außerdem sollten innerhalb des Videos Textblöcke mit Zusatzinformationen erscheinen. Diese werden aus einer Tabelle herausgelesen, die sich mühelos in verschiedene Sprachen übersetzen lassen. Des Weiteren könnten Unternehmen Zusatzinformationen für Ihre Produkte implementieren. Die Implementierung des Programms wird im Kapitel 5.2 beschrieben.

Das Programm wird für das Betriebssystem *Windows* konzipiert.

1.2.2 Einbindung der Videos in das Programm

Die einzelnen Videos müssen möglichst im Programm selbst abspielbar sein. Zu programmieren ist also eine Art Mediaplayer innerhalb des Interfaces bzw. eine Interaktion mit einer anderen Abspielsoftware. Die Videos werden mit einer Auflösung von maximal 1280x720 Pixel dargestellt. Das Fenster muss in der Größe variierbar sein, entsprechend muss sich das Video anpassen. Eine Übersicht der in Frage gekommenen Player, gibt es im Kapitel 3.2.6.

Ist das Programm soweit fertig gestellt, schließt sich eine ausführliche Testphase an, in welcher das Programm auf seine Stabilität hin überprüft wird. Die Erfahrungen der Testpersonen sollen berücksichtigt werden und Verbesserungsvorschläge nutzbringend in das Programm einfließen. Die Ergebnisse der Testphase werden im Kapitel 6.2 erläutert.

1.2.3 Interface-Design

Zu wählen ist ein Interface, welches leicht zu bedienen und sehr übersichtlich gestaltet ist, da der Inhalt des Programms eine Zielgruppe mit wahrscheinlich eher geringeren Computer-Kenntnissen hat, ist das Interface entsprechend aufzubauen, sodass die Lösung eines Problems sehr einfach und ohne Fachwissen zu finden ist. Um ein vielseitiges und gutes Resultat beim

Design zu bekommen, wurde entschieden, dass von beiden Studenten ein eigenständiges Design entworfen wird und am Schluss die Vorteile beider Entwürfe zusammengefügt werden. Die Beschreibungen der einzelnen Entwürfe stehen im Kapitel 5.1.7 und im Kapitel 5.2.4 sowie die Zusammenfügung der Ergebnisse im Kapitel 5.3.

1.3 Kapitelübersicht

Im Kapitel Systemüberblick werden zunächst die Aufgabenaufteilung des Teams sowie die grundsätzlichen Anforderungen an die Software erläutert.

In dem Kapitel Recherche werden hauptsächlich die Recherche, der Vergleich und die Auswahl von Methoden und Tools beschrieben, die für eine erfolgreiche Durchführung des Projektes notwendig sind.

Das Kapitel Konzeptentwicklung erläutert die Konzepte der Implementierung, die auf Grundlage der Recherche und Aufgabenstellung erstellt wurden.

Im Kapitel Implementierung wird die konkrete Umsetzung der Konzepte beschrieben.

Das Kapitel Testphase beschreibt den Verlauf und die Umsetzung sowie die Auswertung der Ergebnisse des Probandentests. Ebenso werden Softwaretests durchgeführt.

Im Kapitel Zusammenfassung werden die erreichten Ergebnisse ausgeführt und evtl. Erweiterungsmöglichkeiten aufgezeigt.

2 Systemüberblick

2.1 Jannik Mewes

Im Folgenden werden die in Kapitel 1 festgelegten Einzelaufgaben auf P. Lindegrün und J. Mewes aufgeteilt. Die Arbeitsteilung muss abgestimmt und sinnvoll koordiniert werden. Überdies wird ein Zeitplan zur Bearbeitung der Einzelaufgaben erstellt.

2.1.1 Aufgabenaufteilung

Da das Projekt vom Aufgabenumfang zu umfassend für eine Bachelorarbeit ist, wurde beschlossen, es auf zwei Studenten aufzuteilen. Die folgende Grafik zeigt an, welche Aufgaben von wem bearbeitet werden.

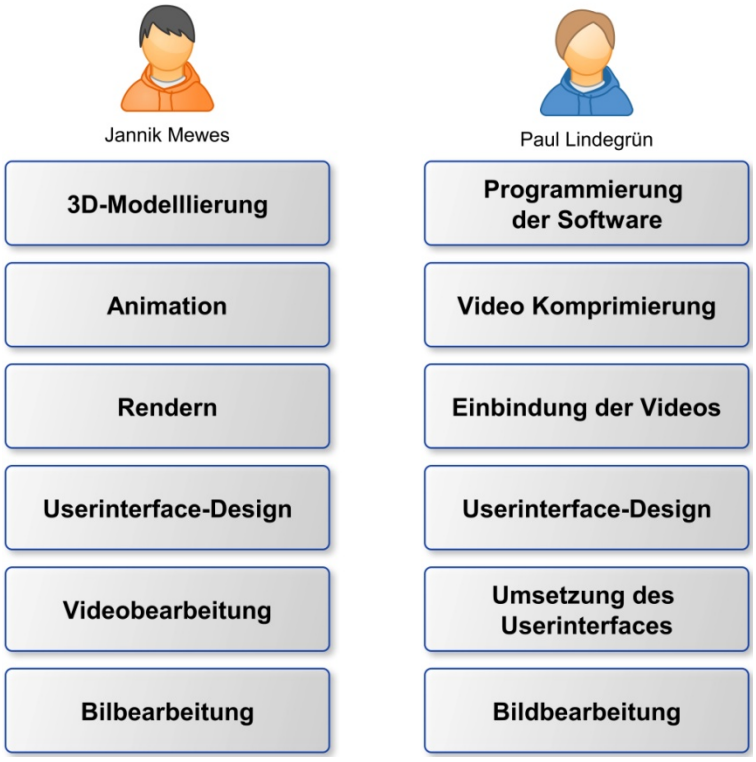


Abbildung 2.1: Aufgabenaufteilung

P. Lindegrün entwickelt vorwiegend die Implementierung des Programms, während J. Mewes die Anleitungsvideos sowie die Programmstruktur erstellt.

2.1.2 Ablaufplan (Praxisprojekt)

Wie schon erwähnt, wurde das Praxisprojekt in verschiedene Aufgabenbereiche unterteilt. In dem folgenden Ablaufplan sollen Einzelschritte des Projekts verdeutlicht werden.

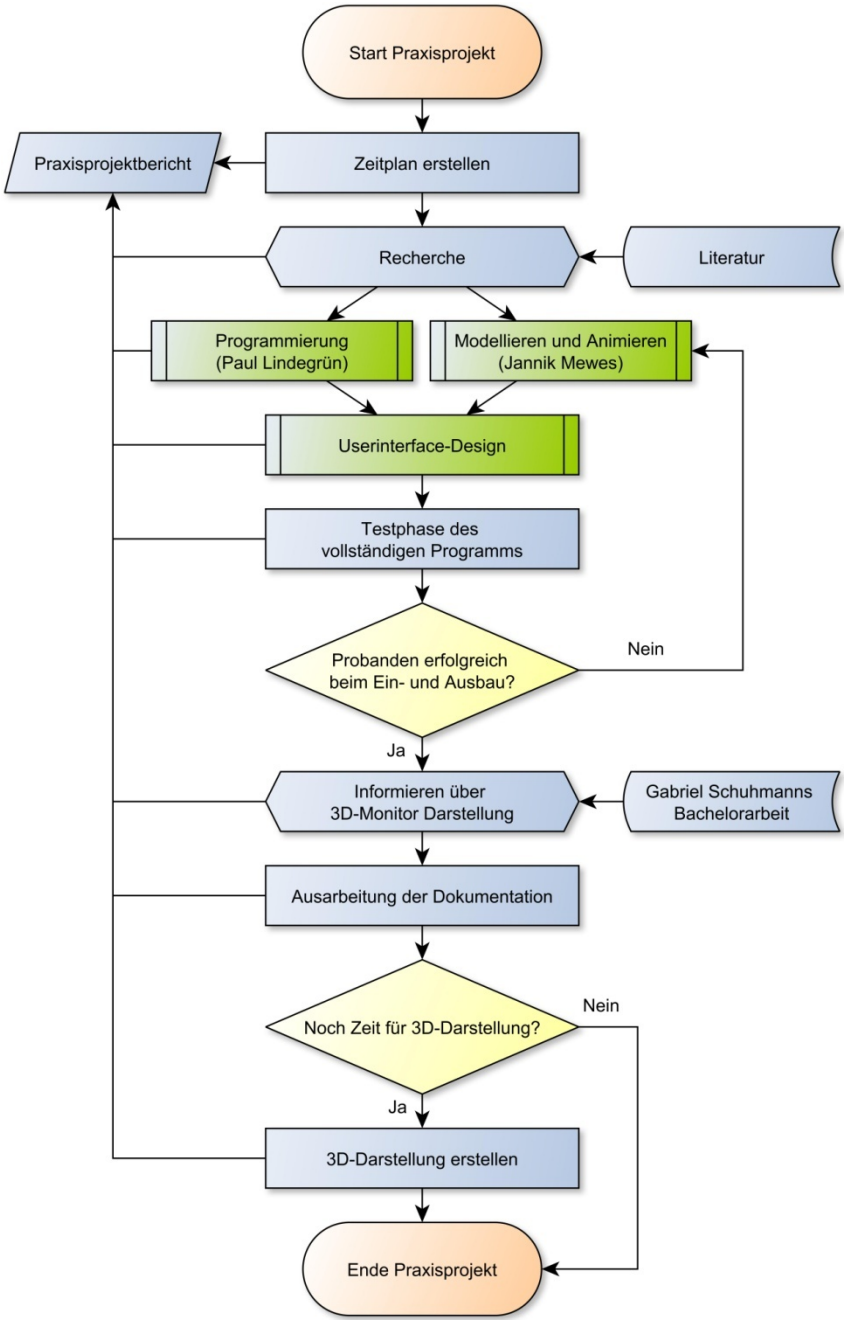


Abbildung 2.2: Ablaufplan Praxisprojekt

Zuerst werden die Aufgaben aufgeteilt und es wird festgelegt, bis wann diese zu erledigen sind. P. Lindegrün wird sich nach der Recherche der Programmierung zuwenden, während J. Mewes die Anleitungsvideos entwickelt.

Anschließend werden die Ergebnisse zusammengefügt und im Team das Userinterface-Design entworfen.

Die grün markierten Schritte in der Skizze 2.2 werden noch genauer in den jeweiligen Kapiteln erläutert.

- Ablaufplan der Programmierung: Kapitel 4.2.4

- Ablaufplan Erstellung von Anleitungsvideos: Kapitel 4.1.5

- Ablaufplan Userinterface-Design: Kapitel 4.1.6

2.1.3 Zeitplan

In der folgenden Übersicht ist dargestellt, wann und welche Aufgaben die Beteiligten für die Projektarbeit zu erfüllen hatten. Dieser Zeitplan dient als Übersicht und zur Strukturierung der Arbeit. Ein abgestimmtes Teamwork ist für die Durchführung dieses Projektes unerlässlich, damit die einzelnen Schritte sinnvoll ineinandergreifen können.

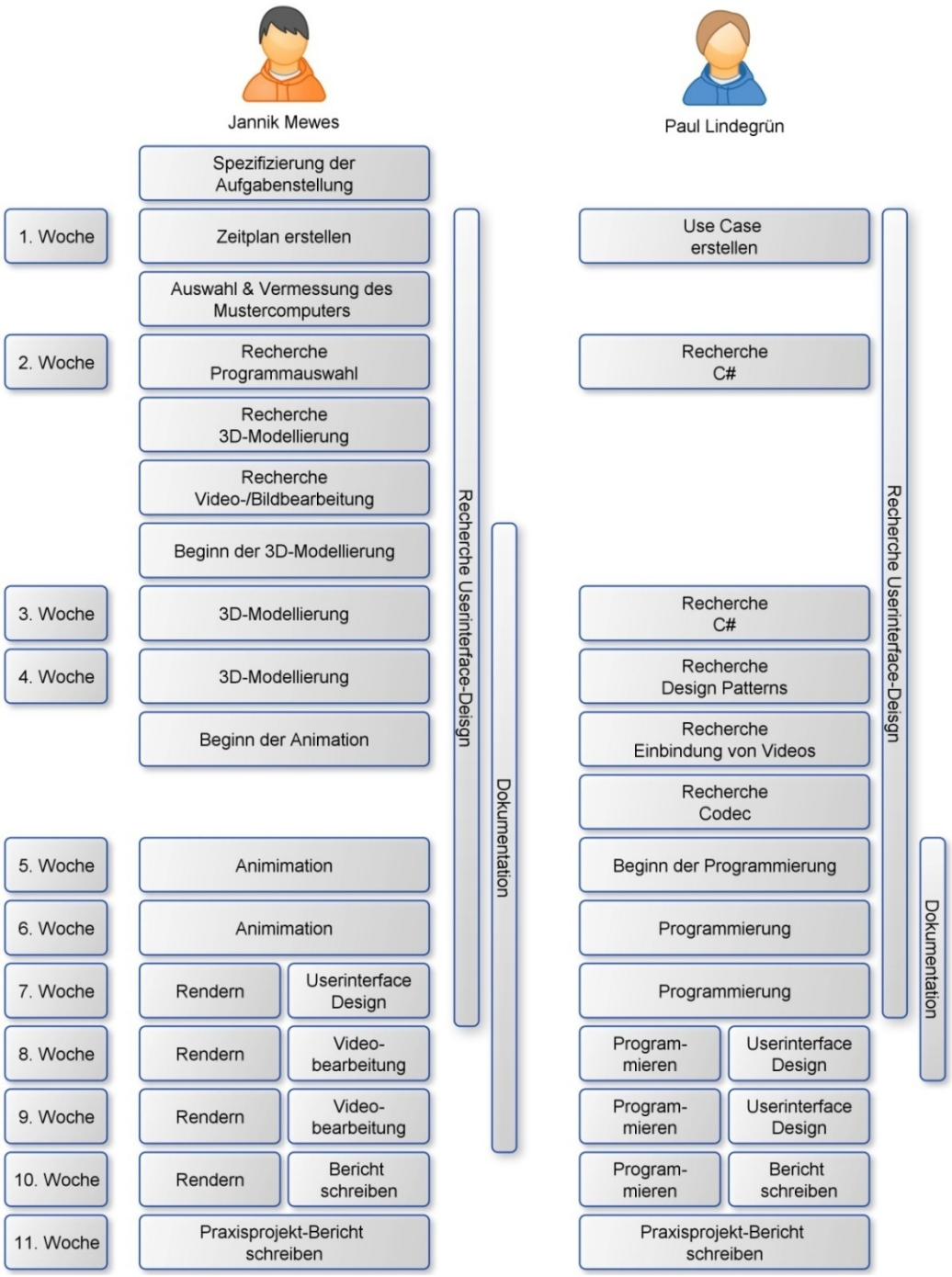


Abbildung 2.3: Zeitplan

2.1.4 Zusammenfassung des Kapitels

In diesem Kapitel wurden die Aufgaben aufgeteilt sowie die zeitlichen Abläufe festgelegt. Im 3. Kapitel werden die jeweiligen Recherchen erläutert, die zur Lösung des Projektes erforderlich sind.

2.2 Paul Lindegrün

In diesem Kapitel soll festgehalten werden, in welchem Umfeld das Projekt entsteht. Überdies werden aus der Aufgabenbeschreibung in Kapitel 1 Anforderungen an die zu erstellende Software abgeleitet.

2.2.1 Beschreibung des Projektumfeldes

Die Arbeit entsteht im Rahmen der Hochschule für angewandte Wissenschaft und Kunst in der Fachrichtung Medien- und Kommunikationssysteme. Die Schwerpunkte dieser Arbeit sind:

- 3D Modellierung
- Programmierung
- Userinterface-Design

Das während dieser Arbeit entstehende Programm soll zu Werbezwecken der Hochschule eingesetzt werden. Daher sollen auf Programmiersprachen, Techniken und Anwendungen zurückgegriffen werden, die an der Hochschule im Lehrplan stehen.

2.2.2 Anforderungen an die Software

Aus der beschriebenen Aufgabenstellung in der Einleitung, dass der Anwender das Programm ohne Installation und ohne vorherige Erfahrung starten kann, lassen sich folgende Anforderungspunkte für die Software festlegen:

- Das Programm muss alle benötigten Komponenten zum Starten der Anwendung und zum Abspielen der Videos bereitstellen und nicht auf externe Programme zugreifen, die der Anwender vorher installieren müsste.

- Es sollen möglichst wenige Systemressourcen verwendet werden, damit das Programm auch auf leistungsschwachen Systemen (z. B. Laptops, Netbooks oder älteren PCs) schnell startet und flüssig bedient werden kann.

- Es soll eine große Abwärtskompatibilität zu älteren Windows-Versionen bestehen (bis Windows XP), um von einem möglichst großen Anwenderfeld benutzt werden zu können.

- Das Programm soll eine Fensteranwendung sein, da die für das Interface und die Benutzerfreundlichkeit nötigen Steuerelemente sich nicht in einer Konsole darstellen lassen.

- Das Interface sollte einfach aufgebaut und leicht verständlich sein, damit Anwender mit wenig oder keinen Vorkenntnissen sich schnell zurechtfinden.

- Der Videoplayer muss die gewohnten Funktionalitäten bieten:
  - Starten / Pausieren
  - Stoppen
  - Vorspulen / Zurückspulen
  - Lautstärkereglung

Ferner sollte das Programm samt der Videos nicht die Kapazität einer DVD[2] (4,7 GB[3]) überschreiten.

2.2.3 Zusammenfassung des Kapitels

In diesem Kapitel wurden das Projektumfeld sowie die Softwareanforderungen dargestellt. Im nächsten Kapitel werden die jeweiligen Recherchen erläutert, die zur Durchführung des Projektes notwendig sind.

[2] DVD (engl.: Digital Versatile Disc)
[3] GB (engl.: Giga Byte)

3 Recherche

3.1 Jannik Mewes

Das 3. Kapitel thematisiert hauptsächlich die Recherchen, die zur Bewältigung der im Kapitel 1 formulierten Aufgaben erforderlich sind. Es sollen Vergleiche zwischen Programmen sowie Technologien gezogen werden und - basierend auf den Anforderungen - passende Utensilien ausgewählt werden. Des Weiteren werden Unterschiede zwischen bestehenden Manuels und dem von uns entwickelten TPHC[4] aufgelistet und verglichen.

3.1.1 Aktuelle Anschlussports

Da die Zeit nicht ausreicht, um für alle Anschlüsse eines PCs eine eigene Anleitung zu erstellen, musste zunächst ermittelt werden, welche Anschüsse aktuell in den Computern der Zielgruppe verbaut wurden. Für die Modellierungen wurde ein Beispiel-PC der neueren Generation gewählt, da die Anleitungen auch in naher Zukunft noch Geltung besitzen sollen.

Die folgende Grafik verdeutlicht, welche Anschlüsse in welcher Generation üblich waren bzw. sind. Der Zeithorizont wird auf 1998 bis 2013 eingegrenzt, der für das Programm relevant ist [1].

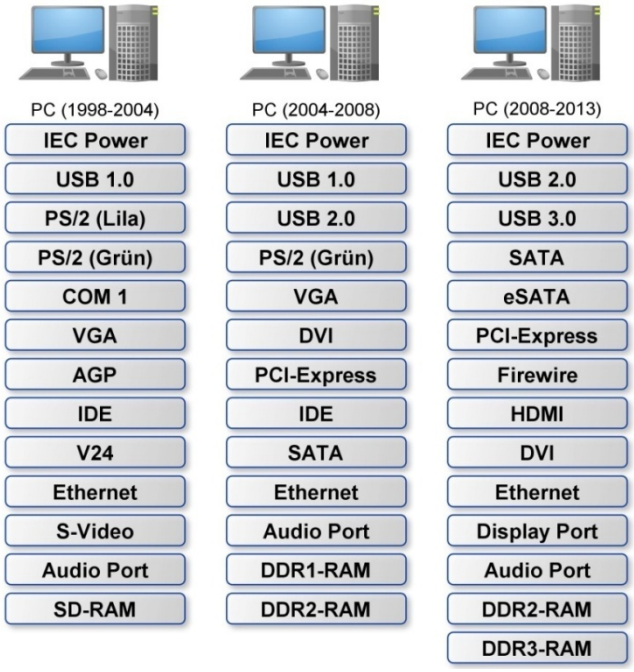


Abbildung 3.1: Generationsvergleich der PC-Anschlüsse

[4] TPHC (engl.: Tutorial Programm for Hardware Change

Für den Wechsel der Bauteile sind externen Anschlüsse nicht relevant. Im Kapitel 7.1.2 wird jedoch offensichtlich, weshalb dennoch Wert darauf gelegt worden ist.

Der zu Grunde gelegte Beispiel-PC besitzt die Anschlüsse PCI-Express[5], SATA[6] sowie DDR3-RAM[7]-Slots, die sich im Gehäuse befinden.

3.1.2 Modellierungs-Programmauswahl

In der Praxisphase muss der Beispiel-PC komplett digitalisiert werden. Hierzu gibt es verschiedene Möglichkeiten. Mangels eines 3D-Laserscanners sollten die 3D-Modelle von Hand aus modelliert werden. Die bekanntesten Entwicklungsumgebungen hierfür sind *Blender, Autocad, Maya* und *3ds-Max*.

Autocad ist eher für technische Zeichnungen geeignet. Es arbeitet mit Formeln und Berechnungen, bietet aber weniger Möglichkeiten „freihand“ zu animieren oder zu modellieren - zumindest nicht in dem Umfang und der Leichtigkeit wie die anderen drei Programme [2].

Blender hingegen ist aus diesem Winkel betrachtet das komplette Gegenteil von *Autocad*. Hier ist das Sculpten[8] sehr einfach und das Programm bietet einen guten Editor, um 3D-Sculpts in die gewünschte Form zu „kneten“. Doch ist *Blender* in der Bedienung sehr unübersichtlich und bietet wenige Möglichkeiten, schnell und einfach maßstabsgetreue Modelle zu erstellen [3].

3ds-Max hingeben ist sehr übersichtlich gestaltet und hat bis auf den Sculpt-Editor alles, was *Blender* auch anbietet. *3ds-Max* besitzt zudem einen sehr guten Animationseditor, welcher für das Projekt gut geeignet ist [4].

Das Programm *Maya* bietet eine sehr gute Mischung aus *3ds-Max* und *Blender*. Es vereint die Vorteile der beiden anderen Programme, ohne die Nachteile zu übernehmen [5].

[5] PCI-Express (engl.: Peripheral Component Interconnect Express)
[6] SATA (engl.: Serial Advanced Technology Attachment)
[7] DDR-RAM (engl.: Double Data Rate - Random Access Memory)
[8] Sculpten: Formveränderung durch „Kneten“ von Polygonnetzen

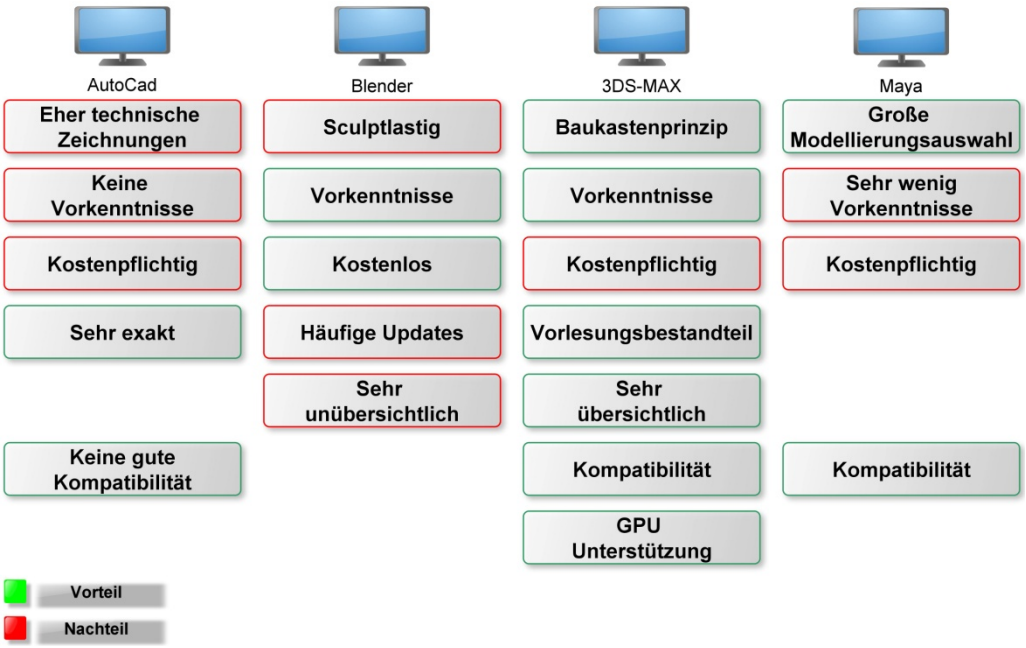


Abbildung 3.2: Vergleich der 3D-Modellierungsprogramme

Zum einen sieht die Aufgabe vor, eine maßstabsgetreue und einigermaßen exakte Kopie des Beispiels-PCs anzufertigen, zum anderen eine flüssige und verständliche Animation zu erstellen.

Die Wahl fiel auf *3ds-Max 2013* aus folgenden Gründen: Die HAWK-Göttingen besitzt bereits Lizenzen für *3ds-Max*, nicht aber für *Maya*. Zudem bietet sie einen Grundkurs für *3ds-Max* an. Da das Endprodukt auch als Werbeträger für die HAWK-Göttingen dienen soll, liegt es nahe, ein Programm zu benutzen, welches in dem Studiengang unterrichtet wird.

Zu einem späteren Zeitpunkt wurde auf *3ds-Max 2014* gewechselt. Die Gründe werden im Kapitel 5.1.9.1 näher erläutert.

3.1.3 Videobearbeitungs-Programmauswahl

Die Anleitungsvideos sollen folgende Anforderungen erfüllen:

1. Geringe Dateigröße:
   Das Programm sollte möglichst wenig Speicherplatz benötigen, da es als Download angeboten werden soll. Außerdem sollte es von der Größe nicht über 4,7 GB hinausgehen, um auf eine DVD zu passen.
2. Gute Videoqualität in der Auflösung 1920x1080 Pixel:
   Full-HD (1920x1080 Pixel) ist momentan bei den meisten PC-Usern die maximale Auflösung, die ihr Monitor unterstützt. Damit die Videos optisch gut aussehen, sollten die Bilder mit diesem Format gerendert werden.
3. Dauer einer Videosequenz:
   Sie sollte zwischen 1 und 5 Minuten liegen. Der Vorteil der 3D animierten Videos ist, dass, trotz kurzer Zeitspanne, ausreichend viele und klare Information übermittelt

werden können. Die Videos sollten kurz und einprägsam sein, um den Inhalt schnell zu erfassen.

4. Einblendungen und Hervorhebungen von wichtigen Elementen:
   Damit dem User wichtige Details nicht entgehen, ist es notwendig, bestimmte Elemente hervorzuheben, um sie nochmals zu verdeutlichen.
5. Videos in Video-Fenstern:
   Bei vielen Animationsparts ist es ratsam, das Gezeigte aus zwei verschiedenen Perspektiven zu präsentieren. Hierzu ist eine Video-in-Video-Funktion sehr hilfreich. Sich wiederholende Prozesse können besser und zeitsparender dargestellt werden.

Alle diese Anforderungen muss das Programm oder müssen die Programme bereitstellen. Es wurden viele Videobearbeitungsprogramme getestet, doch ergaben sich immer wieder Probleme. Die Wahl fiel letztlich auf das Programm *Adobe Premiere Pro* [6].

3.1.4 Bildbearbeitungs-Programmauswahl

Das Praxisprojekt verlangt die Bearbeitung von Texturen für die 3D-Modelle sowie Texturen und Bilder für die Einblendungen in den Videos. Außerdem musste das Design des Interfaces der Programmoberfläche entworfen werden.

Für die Bildbearbeitungen standen eine Reihe Programme zur Auswahl. Da die HAWK-Göttingen Lizenzen für *Adobe Photoshop* besitzt und es das benutze Bildbearbeitungsprogramm in der Vorlesung "Userinterface-Design" ist, fiel die Wahl auf die Programmversion *CS4*.

Im Gegensatz zu den Konkurrenten (z.B. *GIMP2.0*) ist *Photoshop* auch das umfangreichste Bilderverarbeitungstool [7].

Um die Icons, Buttons und andere Texturen für das Programm zu erstellen, wurde *Adobe Illustrator CS4* verwendet. Mit diesem Programm lassen sich die notwendigen Vektorgrafiken sehr gut entwickeln [8]. Somit ist gewährleistet, dass auch beim Skalieren die Texturen immer scharf abgebildet werden.

3.1.5 Benötigte Arbeitsmaterialien

Um 3D-Modelle mit hoher Detailstufe erstellen zu können, wird ein leistungsfähiger Computer benötigt. Des Weiteren sollte der Computer ein 64-Bit-Betriebssystem unterstützen. Da gerade die neueren 3D-Programmversionen scheinen technische Probleme mit 32-Bit-Systemen zu haben.

Die Mindestanforderungen für *Studio 3ds-Max* lauten [9]:

- Intel® 64 or AMD[9]64 - Prozessor

[9] AMD (engl.: Advanced Micro Devices)

- 4 GB[10] RAM (8 GB empfohlen)
- 4 GB Auslagerungsspeicher (84 GB empfohlen)
- 3 GB freier Festplattenspeicher
- Direct3D 10-Technologie, Direct3D 9 oder OpenGL-fähige Grafikkarte
- Grafikkarte mit 512 MB oder mehr Speicher (1 GB oder mehr empfohlen)
- Internet-Anschluss für Downloads und Zugriff auf Autodesk Subscription-Leistungen

Da zum Rendern der Animationen der 3D-Modelle nach Fertigstellung nicht mehr allzu viel Zeit bleibt, sollte es mindestens ein Computer mit folgenden technischen Spezifikationen sein [9]:

- Intel 64- oder AMD 64-Prozessor mit SSE2-Technologie
- 8 GB RAM
- 40 GB Auslagerungsspeicher
- 100 GB freier Festplattenspeicher
- Direct3D 10, Direct3D 9 oder OpenGL-fähige Grafikkarte
- Grafikkarte mit 1 GB oder mehr Speicher
- Internet-Anschluss für Downloads und Zugriff auf Autodesk Subscription-Leistungen

3.1.6 Kontaktaufbau zu Computer-Hardware-Unternehmen

Im Rahmen der Bachelorarbeit werden verschiedene Produkte großer Unternehmen der Computerbranche dargestellt. Um dies rechtlich abzuklären und zu sichern, damit es zu keinen Komplikationen kommt, habe ich die entsprechenden Unternehmen (*NVIDIA*, *Asus*, *Antec*) kontaktiert. Des Weiteren sollte herausgefunden werden, ob die Unternehmen fertige 3D-Modelle von Ihren Produkten besitzen und ob sie uns diese zur Verfügung stellen würden.

Die Antworten aller Unternehmen klangen recht identisch. Sie verfügten weder über 3D-Modelle noch über Vermessungsdaten ihrer Produkte. Dennoch bekundeten alle Unternehmen ihr großes Interesse am Ergebnis bzw. Fortschritt der Bachelorarbeit.

3.1.7 Zusammenfassung des Kapitels

In der Recherche wurde erläutert, aus welchen Gründen welche Entwicklungstools für das Projekt ausgewählt wurden. Die Wahl fiel auf das Modellierungsprogramm *Autodesk 3ds-Max* sowie das Bildbearbeitungsprogramm *Adobe Photoshop* und das Videobearbeitungsprogramm *Adobe Premiere Pro*. Außerdem wurden aktuelle Informationen über benötigte Anschlussports aufgelistet. Des Weiteren wurden auch die benötigten Arbeitsmaterialien beschrieben (z. B. Mindestanforderungen für den Modellierungs-Computer).

Im nächsten Kapitel werden aufgrund der im diesem Kapitel erstellten Recherchen Konzepte zur Umsetzung entwickelt.

[10] GB (engl.: Giga Byte)

3.2 Paul Lindegrün

In diesem Kapitel werden hauptsächlich die Recherche und die Auswahl von nötigen Methoden oder Tools behandelt, die zur Durchführung des Projektes notwendig sind. Diese werden untereinander anhand der im vorherigen Kapitel erstellten Anforderungen verglichen und ausgewählt.

3.2.1 Auswahl der Programmiersprache

Wie bereits im Kapitel 2.2.2 dargestellt, muss das Programm eine Fensteranwendung sein, um vom Anwender korrekt und sicher bedient werden zu können. Dadurch fallen von vornherein einige Programmiersprachen aus der Auswahlmöglichkeit heraus, da es mit ihnen sehr schwer oder unmöglich ist, eine für den Anwender ansprechende Fensteroberfläche zu erstellen (z. B. C, Assembler). Die gebräuchlichsten Programmiersprachen, die dafür verwendet werden, sollen im Folgenden kurz erläutert werden.

- C++ wurde 1979 von Bjarne Stroustrup entwickelt und ist eine Weiterentwicklung der Programmiersprache C. C++ ergänzt C um das Konzept der Objektorientierung [1] und ermöglicht dadurch einen höheren Abstraktionsgrad bei der Programmierung. Gleichzeitig sind die durch C++ erstellten Programme sehr hardwarenah und dadurch schnell und effizient bei der Ausführung. Durch das WinAPI[11] bietet sich auch die Möglichkeit, eine *Windows*-Fensteranwendung zu erstellen. Jedoch ist C++ nicht plattformunabhängig. Zwar lässt sich der C++ Quellcode für fast jede Plattform kompilieren, doch ein bereits kompiliertes Programm lässt sich nicht auf einem anderen System ausführen [2].

- Java wurde 1996 von *Sun Microsystems* veröffentlicht und ist eine vollständig objektorientierte Programmiersprache. Einer der größten Unterschiede von Java zu C++ ist, dass der Java-Quellcode nicht wie in C++ direkt in den Maschinencode kompiliert wird, sondern zuerst in einen Zwischencode. Der Zwischencode wird wiederrum in der JRE[12] zur Laufzeit in den Maschinencode übersetzt. Dadurch sind die Programme nicht so effizient und lassen sich nicht so gut optimieren wie bei C++. Jedoch wird dadurch eine Plattformunabhängigkeit der in Java geschriebenen Programme erreicht. Voraussetzung hierfür ist, dass auf dem System die JRE zur Verfügung steht. Zudem muss sich der Programmierer nicht um die Speicherverwaltung kümmern. Beim Erstellen eines Objektes wird automatisch Speicher allokiert und nicht mehr gebrauchte Objekte werden vom Garbage Collector wieder freigegeben. Durch die mitgelieferten Bibliotheken im JDK[13] lassen sich auch ansprechende Fensteranwendungen programmieren [3].

[11] WinAPI (engl.: Windows Application Programming Interface)
[12] JRE (engl.: Java Runtime Environment)
[13] JDK (engl.: Java Development Kit)

- C# wurde 2002 von *Microsoft* veröffentlicht und ist ein Konkurrenzentwurf bzw. eine Weiterentwicklung von Java. Ebenso wie Java ist C# vollständig objektorientiert, plattformunabhängig durch das bei der ECMA[14] standardisierte .Net Framework und besitzt einen Garbage Collector, der für die Speicherbereinigung zuständig ist. Jedoch steht das .Net Framework bei Weitem nicht für so viele Systeme zur Verfügung wie die JRE. *Microsoft* selbst bietet das .Net Framework nur für die verschiedenen *Windows*-Versionen an. Um C# Programme auf anderen Systemen nutzen zu können, bietet es sich an, das "Mono-Project" zu installieren."Mono-Project" [4] ist eine open source Implementierung des standardisierten .Net Frameworks und ermöglicht es dadurch C# Programme auf *Windows*, *Mac OS X* und verschiedenen *Linux* Distributionen auszuführen. Zum Erstellen von Fensteranwendungen bietet C# bzw. das .Net Framework zwei Möglichkeiten an: *Windows* Forms [5] und *W*PF[15] [6].

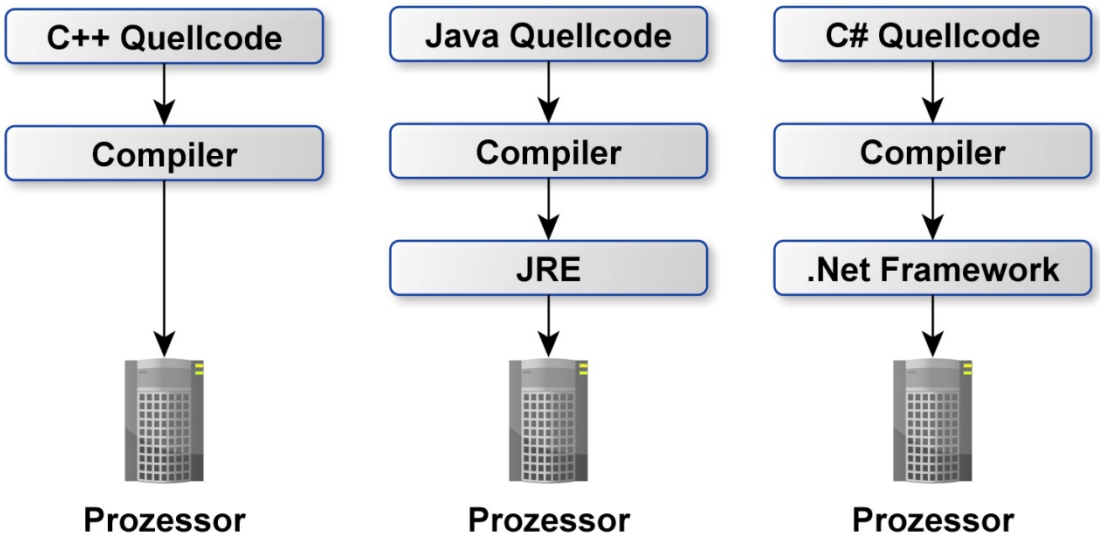


Abbildung 3.3: Vergleich C++, Java & C#

Die Wahl fiel auf C#, da es eine der modernsten Programmiersprachen ist und mit *Windows* Forms oder *W*PF sehr umfangreiche Bibliotheken zum Erstellen ansprechender Fensteranwendungen bietet. Außerdem ist das .Net Framework ab *Windows Vista* bereits vorinstalliert. Dadurch muss der Anwender im Idealfall keine weiteren Installationen durchführen und kann das Programm direkt starten. Bei Windows XP ist keine Version des .Net Frameworks vorinstalliert.

Es bleibt noch zu klären, ob die Fensteranwendung mit Hilfe von *Windows* Forms oder WPF implementiert wird. Hinsichtlich der Verwendung des Programms mit Hilfe des "Mono-Project" ergibt sich ein Punkt, der für *Windows* Forms spricht. Das "Mono-Project" bietet keine Unterstützung [4] für *W*PF. Somit bietet es sich an, auf *Windows* Forms zu setzen, weil dadurch

[14] ECMA (engl.: European Computer Manufacturers Association)
[15] WPF (engl.: Windows Presentation Foundation)

eine zukünftige Implementierung des Programms auf einem anderen Betriebssystems erleichtert wird.

3.2.2 Auswahl der Entwicklungsumgebung

Da nun C# als Programmiersprache für das Projekt feststeht, muss noch geklärt werden, welche Entwicklungsumgebung dafür zum Einsatz kommen soll.

Eine Entwicklungsumgebung ist eine Software, die es dem Programmierer ermöglicht, ein eigenes Programm zu erstellen. Sie vereint verschiedene Anwendungen in sich, die zum Entwickeln von Software notwendig sind.

Unter anderem sind dies:

- Der Texteditor: Im Texteditor wird der Quellcode bearbeitet. Er bietet zahlreiche Möglichkeiten, um die Übersicht im Quellcode zu gewährleisten, z. B. das Zusammenfassen von Quellcodeblöcken.

- Der Linker: Er führt alle nötigen Quellcodedateien zusammen, um daraus ein einziges ausführbares Programm zu erstellen.

- Der Compiler: Der Compiler übersetzt den im Texteditor geschriebenen Quellcode in eine Maschinensprache, um ihn auf dem Prozessor ausführen zu können.

- Der Debugger: Er dient dazu, Fehler im Quellcode aufzufinden, um diese zu beseitigen.

- Die Autovervollständigung: Sie bietet dem Programmierer während der Arbeit eine Auswahl an, um den Quellcode zu vervollständigen. Dies dient vor allem dazu, die von Hand geschriebene Quellcodemenge zu reduzieren und den Programmierer zu entlasten.

Einige der gängigsten Entwicklungsumgebungen sollen an dieser Stelle kurz vorgestellt werden.

- *NetBeans*: *NetBeans* ist eine open source Entwicklungsumgebung und steht für mehrere Betriebssysteme zur Verfügung. Sie wurde primär für die Programmiersprache Java entwickelt und bietet keine Unterstützung für C#. Daher ist *NetBeans* für das Projekt ungeeignet [7].

- *Eclipse*: *Eclipse* ist ebenfalls eine open source Entwicklungsumgebung und steht für mehrere Betriebssysteme zur Verfügung. Ebenso wie *NetBeans* wurde auch *Eclipse* primär für Java entwickelt, jedoch können durch ihre Erweiterbarkeit auch Programme für andere Programmiersprachen geschrieben werden. Es gibt mehrere Plug-ins, um die Funktionalität von *Eclipse* auf C# zu erweitern [8].

- *MonoDevelop*: *MonoDevelop* ist eine open source Multiplattform-Entwicklungsumgebung. Sie wurde primär für das "Mono-Project" entwickelt und unterstützt daher C# und andere .Net Programmiersprachen [9].

- Visual Studio: Visual Studio ist eine von *Microsoft* bereitgestellte Entwicklungsumgebung für *Windows*. Diese wurde primär für das .Net Framework entwickelt. Dadurch bietet sie eine optimale Unterstützung für C# und das .Net Framework [6].

Die Wahl fiel auf *Visual Studio 2010 Professional*, da es eine direkt von *Microsoft* stammende Entwicklungsumgebung für das .Net Framework ist. Dadurch ist sie optimal für das Projekt geeignet. Außerdem besitzt die Hochschule bereits Lizenzen für die Entwicklungsumgebung *Visual Studio 2010 Professional*, zudem wird die Entwicklungsumgebung in den angebotenen Vorlesungen mit programmiertechnischem Hintergrund verwendet. Da das zu entwickelnde Programm auch zu Werbezwecken der Hochschule eingesetzt werden soll, liegt es nahe, eine Entwicklungsumgebung zu benutzen, welche an der Hochschule unterrichtet wird.

3.2.3 Design Patterns

Design Patterns sind Entwurfskonzepte für die Softwarenwicklung. Sie stellen Schablonen zum Entwerfen von Klassenstrukturen zur Verfügung, um wiederkehrenden Problemen bei der Softwareentwicklung zu begegnen. Das Ziel der einzelnen Entwurfsmuster ist unterschiedlich, jedoch ist ihnen gemeinsam, dass der Quellcode möglichst oft wiederverwendbar sein soll.

Hinsichtlich der zukünftigen Weiterverwertung und der Erweiterbarkeit des Projektes ist es daher sinnvoll, sich mit Entwurfsmustern auseinanderzusetzen. Im Folgenden sollen einige Entwurfsmuster vorgestellt und erläutert werden:

- Strategy: Das Strategy-Muster definiert eine Familie von Algorithmen. Diese werden einzeln gekapselt und austauschbar gemacht. Dadurch wird es dem Algorithmus ermöglicht, unabhängig vom eingesetzten Client zu variieren. Das Strategy-Muster ist für das Projekt interessant, da es ermöglicht, Programmteile, die sich ändern können, von denen zu trennen, die konstant bleiben. So können die veränderlichen Programmteile ausgetauscht werden, ohne die konstanten Programmteile zu beeinflussen [10].

- Observer: Das Observer-Muster definiert eine Eins-zu-Viele-Abhängigkeit von Objekten. Wenn sich der Zustand eines Objektes (des Subjektes) ändert, werden die abhängigen Objekte (die Beobachter) darüber benachrichtigt und können dann ihre eigenen Prozeduren ausführen. Für das Projekt ist es, in Hinblick auf die Benutzeroberfläche des Programms, ein sehr interessantes Muster. So kann man auf die Zustandsänderung eines Steuerelementes reagieren und die entsprechende

Verarbeitungskette in Gang setzen. Außerdem ist dieses Muster bereits in C# in Form von Events integriert [10].

- Singleton: Das Singleton-Muster stellt sicher, dass es nur eine einzige Instanz einer Klasse gibt. Außerdem stellt es eine globale Zugriffsmöglichkeit für diese Instanz zur Verfügung. Das Singleton-Muster ist für das Projekt uninteressant, da es sich eher für die Ansteuerung von einzelnen Systemkomponenten eignet, z.B. Zugriff auf die Grafikkarte [10].

- Template Method: Das Template Method-Muster definiert in einer Basisklasse ein Grundgerüst für einen Algorithmus und lagert einzelne Teile des Algorithmus in Unterklassen aus. Dadurch können die Unterklassen ihre eigene Implementierung für den Algorithmus einbringen, ohne die Grundstruktur des Algorithmus zu verändern. Für das Projekt ist dieses Muster interessant, da es ermöglicht, einen Hauptalgorithmus zu definieren und Teile von ihm dynamisch zur Laufzeit zu verändern [10].

3.2.4 Auswahl des Codecs

Die aus den 3D-Modellen gerenderten Videos werden unkomprimiert sein und daher sehr viel Speicherplatz einnehmen. Bei einer Auflösung von 1920*1080 Bildpunkten mit 24 Bit pro Pixel (8 Bit-RGB), einer Bildwiederholungrate von 30 Hz und einer Lauflänge von 60 s, ergibt sich eine Dateigröße von:

$$1920 * 1080\ \text{Pixel} * 24\frac{\text{bit}}{\text{Pixel}} * 30\ \text{Hz} * 60\ \text{s} = 89{,}57952 * 10^9\ \text{bit} \approx 90\ \text{Gb}$$

Ca. 90 Gb pro Videominute sind eindeutig zu viel. Um die Größe der Videos zu verringern, müssen diese komprimiert werden. Die Komprimierung wird mit einem Codec (Kunstwort aus Encoder und Decoder) durchgeführt.

Dazu wird der H.264 Codec (auch MPEG-4/AVC genannt) verwendet. Es ist einer der modernsten Codecs, zeichnet sich durch sehr hohe Komprimierungsraten und gute Bildqualität aus. Zudem ist er weit verbreitet und wird von fast allen gängigen Videobearbeitungsprogrammen unterstützt. Ein weiterer Vorteil des H.264 Codecs, ist die Möglichkeit, durch einstellen des CRF[16], ein Video mit konstanter Qualität komprimieren zu lassen. Dadurch wird das Video mit einer variablen Bitrate kodiert. Jedoch lässt sich bei konstanter Qualität nicht die endgültige Dateigröße abschätzen, da diese dann vom Videoinhalt abhängt [11, 12].

[16] CRF (engl.: Constant Rate Factor)

3.2.5 Auswahl des Videocontainers

Da nun der Videocodec feststeht, bleibt noch zu klären, welcher Videocontainer zum Einsatz kommen soll.

Ein Videocontainer kann Audio- und Videodaten verschiedener Codecs in sich aufnehmen und diese strukturieren, um sie später auslesen zu können. Die Unterschiede zwischen den Containern liegen darin, wie sie die in ihnen enthaltenden Daten strukturieren und ob sie noch zusätzliche Daten, z.B. Untertiteldaten, aufnehmen können. Für das Projekt wird ein Container benötigt, der Audio- und Videodaten aufnehmen kann und mit dem ausgewählten Codec kompatibel ist. Im Folgenden werden die verbreitetsten Container kurz erläutert.

- Audio Video Interleave: Der AVI-Container ist einer der ältesten und verbreitetsten Container-Formate. Durch seine Verbreitung wird dieses Format von den meisten Videoplayern und Geräten unterstützt. Jedoch gibt es Probleme mit aktuellen Codecs, da das AVI Format z.B. nicht für variable Bitraten oder variable Frameraten konzipiert wurde [13].

- Flash Video: Das FLV-Format wird vor allem beim Übertragen von Videoinhalten über das Internet eingesetzt. Viele Videoplayer und Geräte sind kompatibel mit diesem Format [14].

- Matroska: Der Matroska-Container ist einer der aktuellsten und flexibelsten Container-Formate. So unterstützt das Matroska-Format fast alle gängigen Codecs. Die Komptabilität seitens der Videoplayer und Geräten ist jedoch beschränkt. Zum Beispiel unterstützten der *Windows Media Player* und einige Videobearbeitungsprogramme dieses Format nicht [13].

- MP4[17]: MP4 ist der offizielle Standard-Container für den H.264 Codec. Dabei ist MP4 die Kurzform des MPEG[18]-4 Standards. Er wird von den meisten Videoplayern, Videobearbeitungsprogrammen und Geräten unterstützt [13].

Die Wahl fiel auf den MP4-Container, da er einer der aktuellsten Formate ist und mit den meisten Videoplayern abgespielt werden kann. Zusätzlich unterstützen viele Videobearbeitungsprogramme dieses Format, was hinsichtlich der späteren Nachbearbeitung der gerenderten Videos ein entscheidendes Kriterium ist.

[17] MP4 (Kurzform von MPEG-4)
[18] MPEG (engl.: Moving Picture Experts Group)

3.2.6 Auswahl des Videoplayers

Da eine von Grund auf neue Programmierung eines Videoplayers den Zeitrahmen des Projektes sprengen würde, ist es sinnvoll, die Funktionalitäten eines bereits bestehen Players einzubinden. Es werden im Folgenden die gebräuchlichsten Videoplayer vorgestellt und es wird erläutert, ob und wie man diese in ein C# Projekt einbinden kann.

- *WMP*[19]: Ist ein von Microsoft entwickelter Player und gehört zum Umfang des *Windows* Media Center, welcher auf den meisten *Windows* Versionen bereits vorinstalliert ist. Um ihn in ein C# Projekt einbinden zu können, muss er zuerst als ein ActiveX Steuerelement in die Toolbox von *Visual Studio* hinzugefügt werden. Von dort aus kann er, wie jedes andere Element aus der Toolbox, dem Programm hinzugefügt werden. Dazu muss jedoch der *Windows Media Player 9* oder eine höhere Version installiert sein. Das Aussehen der Bedienelemente des Players lässt sich nur schwer oder überhaupt nicht verändern [15].

- *MPC-HC*[20]: Es ist ein open source Media Player und ist in der Lage, fast alle gängigen Videodateiformate und Codecs abzuspielen. Es gibt jedoch keine Möglichkeit einer Einbindung in ein C# Projekt [16].

- *VLC*[21]: Der *VLC*-Player ist ebenfalls ein vielseitiger open source Mediaplayer, der so gut wie alle gängigen Container und Codecs unterstützt. Nachdem der Player installiert wurde, kann er als ein ActiveX Steuerelement der Toolbox von Visual Studio hinzugefügt werden. Von dort aus kann er, wie jedes andere Steuerelement, dem C# Projekt hinzugefügt werden. Um auf die Funktionalitäten des Players zugreifen zu können, muss dieser jedoch installiert sein [17].

- *VLC* Bibliotheken: Eine weitere Möglichkeit, um die Funktionalitäten des *VLC* Players zu nutzen, ist es seine Bibliotheken und Plug-ins in das Projekt einzubinden. Die Bibliotheken sind dann im Projekt vorhanden und eine Vorinstallation des *VLC*-Players entfällt dadurch. Jedoch kann man mit C# nicht auf diese Bibliotheken zugreifen, da diese nicht in C# geschrieben sind. Hierzu wird eine Adapter-Bibliothek benötig [18].

Die Wahl viel auf das Einbinden der *VLC* Bibliotheken, da der Player nicht vorinstalliert sein muss und somit die im Kapitel 2.2.2 gestellten Anforderungen erfüllt. Außerdem werden mit den Bibliotheken keine Bedienelemente mitgeliefert und müssen vom Programmierer selbst erstellt werden. Dadurch kann das Interface und das Aussehen einheitlich gestaltet werden. Somit ist diese Lösung optimal für das Projekt geeignet. Als Adapter werden die Bibliotheken des open source Projektes "VideoLan DotNet" [18] eingesetzt, die es ermöglichen, aus C# heraus auf die *VLC* Bibliotheken zuzugreifen.

[19] WMP (engl.: Windows Media Player)
[20] MPC-HC (engl.: Media Player Classic – Home Cinema)
[21] VLC (engl.: VideoLAN ehem. VideoLAN Client)

3.2.7 Interface Entwürfe

Wie bereits in der Einleitung beschrieben, soll das Interface des Programms übersichtlich und für den Anwender leicht bedienbar sein. Es gibt keine konkreten Vorgaben, um ein Interface für eine spezifische Aufgabe zu entwerfen. Beim Entwurf des Interfaces geht es nur um die allgemeine Anordnung der Anzeige- und Steuerelemente. Es geht nicht um die farbliche Gestaltung oder Formgebung, diese wird im Kapitel 5.1.7 und im Kapitel 5.2.4 näher erläutert. Hier sollen zunächst einige Überlegungen zu möglichen Entwürfen aufgeführt und erläutert werden.

- Auswahl durch Menüleisten: Die Auswahl der einzelnen PC-Bestandteile wird in eine Menüleiste, wie sie bei vielen Anwendungen vorkommt, ausgelagert. Dort werden die Komponenten zu Kategorien zusammengefasst (z.B. Arbeitsspeicher). Beim Auswählen einer Kategorie klappt ein Dropdown-Menü auf, aus dem eine konkrete Komponente ausgewählt werden kann (z.B. DDR2 oder DDR3 RAM-Riegel). Das Video sowie die Informationen zum Einbau der Komponenten werden jeweils in einem eigenen Anzeigeelement im Fenster ausgegeben. Die Video-Steuerelemente befinden sich unter dem Video-Anzeigeelement.

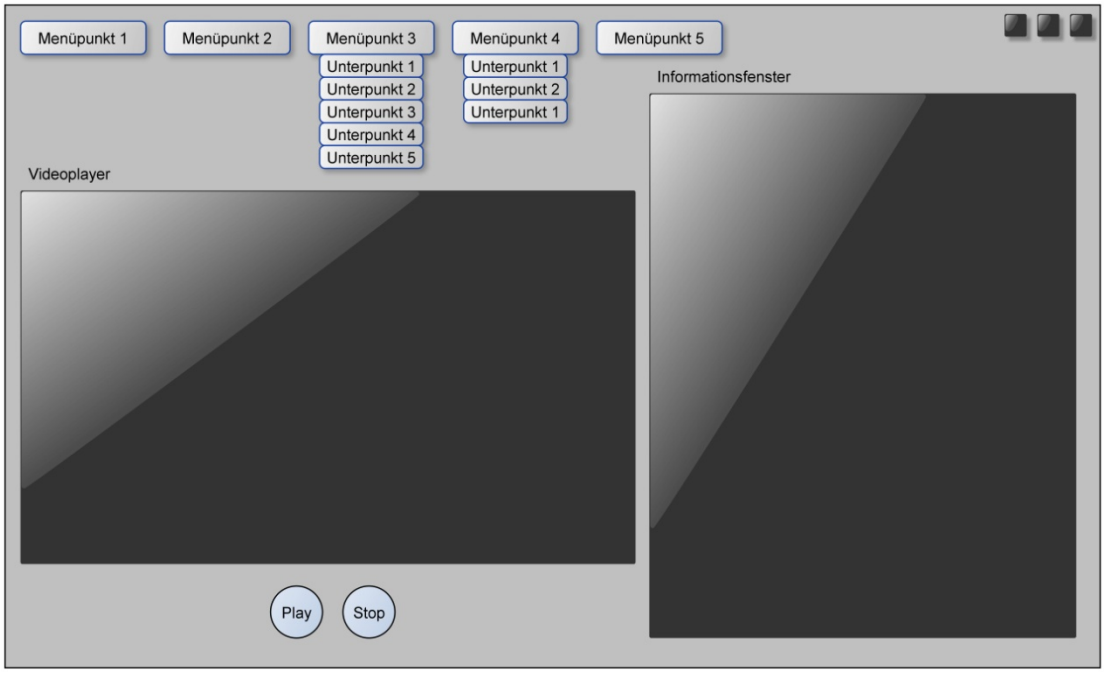


Abbildung 3.4: Entwurf: Auswahl durch Menüleisten

- Tabellarische Auswahl: Die PC-Bestandteile werden in einer Tabelle aus Steuerelementen aufgelistet und können dort ausgewählt werden. So wie beim Entwurf "Auswahl durch Menüleisten", werden die Informationen und das Video in jeweils eigenen Anzeigeelementen ausgegeben. Ebenfalls befinden sich die Video-Steuerelemente unter der Videoausgabe.

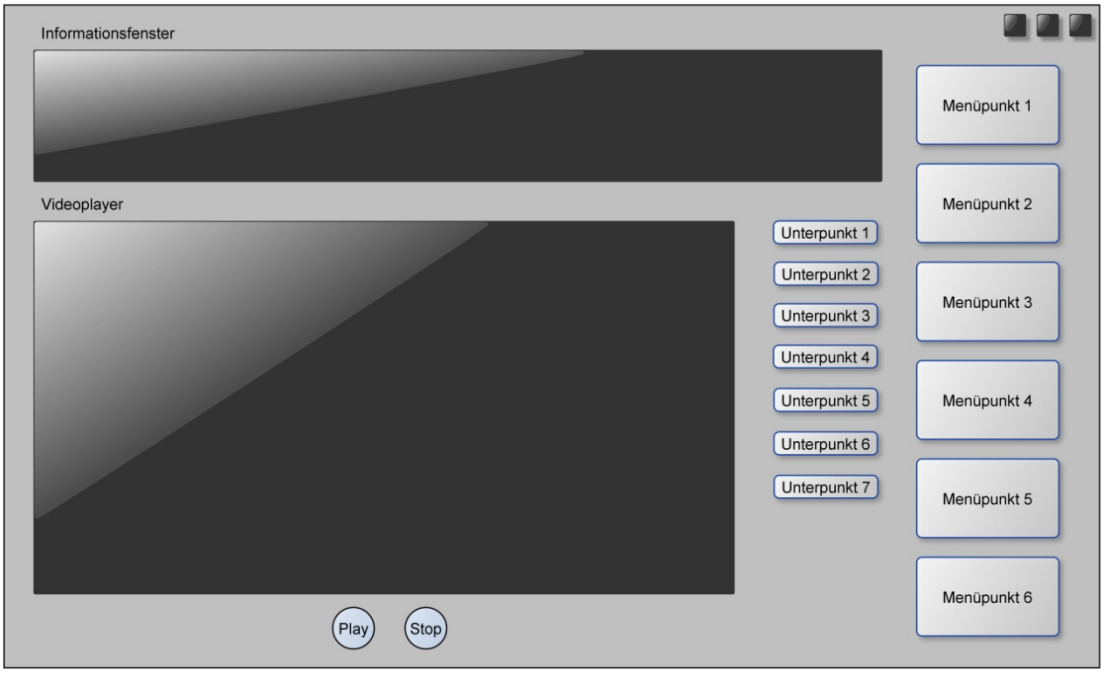


Abbildung 3.5: Entwurf: Tabellarische Auswahl

- Weiterleitung zur nächsten Ebene: Bei diesem Entwurf wird das Fenster in mehrere Ebenen aufgeteilt. Auf der ersten Ebene befindet sich die kategorische Auswahl der Komponenten (z.B. Festplatte oder CPU[22]). Nachdem eine Wahl getroffen wurde, wird zur nächsten Ebene weitergeleitet. Hier werden die Informationen zum Einbau angezeigt und es kann eine konkrete Komponente ausgewählt werden (z.B. DDR2 oder DDR3 RAM-Riegel). In der dritten Ebene befindet sich das Video-Anzeigeelement mit den dazugehörigen Video-Steuerelementen.

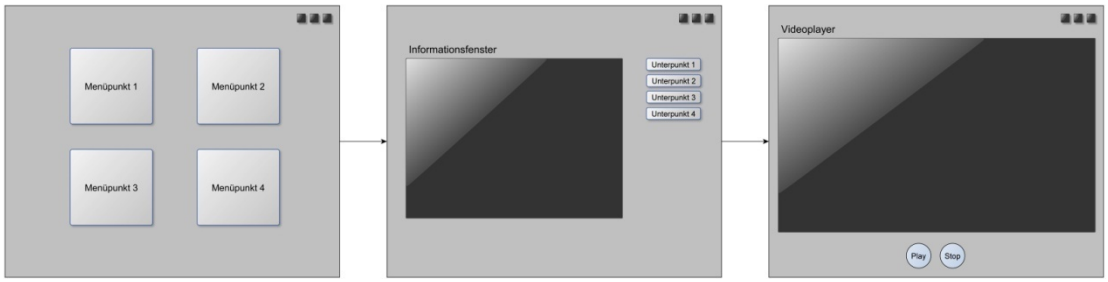


Abbildung 3.6: Entwurf: Weiterleitung zur nächsten Ebene

Die Entscheidung fiel auf den Entwurf "Weiterleitung zur nächsten Ebene", da dieser die gestellten Anforderungen im Kapitel 2.2.2 am besten erfüllt. Er verteilt die Anzeige- und Steuerelemente auf mehrere Ebenen. Dadurch wird der Anwender nicht durch das gleichzeitige Anzeigen aller Elemente überfordert und die Übersichtlichkeit wird gewährleistet. Außerdem wird dadurch eine schrittweise Führung durch das Programm ermöglicht.

[22] CPU (engl.: **C**entral **P**rocessing **U**nit)

3.2.8 Zusammenfassung des Kapitels

Es wurden verschiedene Lösungsansätze verglichen und die geeignetsten ausgewählt. So kommt C# als Programmiersprache und Visual Studio 2010 als Entwicklungsumgebung zum Einsatz. Als Codec wurden der H.264-Codec und das MP4-Format als Videocontainer ausgewählt. Zum Abspielen der Videos werden die Bibliotheken des *VLC*-Players in das Projekt eingebunden. Um auf die *VLC*-Bibliotheken zugreifen zu können, werden zusätzlich die Bibliotheken des "VideoLan DotNet" Projektes eingebunden. Das Interface wird in drei Ebenen aufgebaut sein, um eine schrittweise Führung durch das Programm zu ermöglichen.

Im nächsten Kapitel werden, auf Basis dieser Auswahl konkrete Konzepte zur Durchführung des Projektes entwickelt.

4 Konzeptentwicklung

4.1 Jannik Mewes

In den letzten Kapiteln wurde die Aufgabenstellung untersucht sowie die Aufteilung des Projektes. Anschließend wurden Recherchen durchgeführt, die für die Fertigstellung der Entwicklungsmethodiken benötigt werden.

In diesem Kapitel werden, basierend auf dieser Grundlage, Konzepte erstellt. Zum einen, wie die Animationen ablaufen sollen, und zum anderen, wie eine 3D-Modellation im schematischen Ablauf praktisch umgesetzt werden kann.

4.1.1 Auswahl des Mustercomputers

Im Rahmen des Projektes galt es einen Computer zu finden, der den Ansprüchen der meisten Hardware-Umbauten entspricht. Deshalb wurde zuerst ein Anforderungsprofil erstellt.

Anforderungsprofil:

- Es sollte ein handelsüblicher Computer sein, der in den meisten Haushalten zu finden ist, vorzugsweise bei Nutzern mit wenig Hardware-Erfahrung.

- Es sollten recht neuwertige Anschlüsse vorhanden sein (z. B.USB[23] 3.0, DVI[24], SATA), um auch das Klientel der Zukunft mit dem Programm zu erreichen. Allerdings sollten ebenfalls ein paar ältere Anschlüsse mit eingebunden sein (z.B. PS2, VGA[25], IDE[26]).

- Es sollte möglichst ein normgerechtes Gehäuse gewählt werden, was den meisten handelsüblichen im Aufbau ähnlich ist. (Netzteil oben links, Laufwerke rechts, Festplatten rechts unten, Mainboard in der Mitte).

- Das Gehäuse sollte ein ansprechendes Design aufweisen.

- Der PC sollte im Besitz der HAWK-Göttingen sein, um die Attraktivität bei Messeständen und Ausstellungen des Werbeträgers zu erhöhen. Für Schulungszwecke ist es vorteilhaft, wenn der dargestellte PC in den Anleitungsvideos mit einem vorhandenen PC der HAWK-Göttingen identisch ist.

Es erwies sich als schwierig, einen Rechner auszuwählen, der diesem Anforderungsprofil entsprach. Zumal die HAWK-Göttingen einen Rahmenvertrag mit dem Unternehmen *DELL* hat und somit eigentlich nur *DELL*-Rechner zur Verfügung stehen. *DELL*-Rechner sind aber grundlegend anders aufgebaut und wären für die Anleitungsvideos weniger geeignet gewesen.

Daher fiel die Wahl nach Absprache mit Professor Dr.-Ing. Ibenthal und Herrn Dipl.-Ing. Buermann auf einen PC, der bereits für eine andere Bachelorarbeit verwendet wurde. Dieser PC wird dank einer 3D-Technologie auf Messen präsentiert.

Diese Lösung war sehr zufriedenstellend. Das einzige Manko ist das von *Antec* designte Gehäuse, welches einige Unterschiede zu den anderen PC-Gehäusen aufweist und dass das verbaute Mainboard keinen IDE-Anschluss besitzt. Das Gehäuse unterscheidet sich allerdings nicht allzu grundlegend und ist, trotz des anderen Aufbaus, gut zu erkennen.

[23] USB (engl.: Universal Serial Bus)
[24] DVI (engl.: Digital Visual Interface)
[25] VGA (engl.: Video Grafhics Array)
[26] IDE (engl.: Intergrated Drive Electronics)

Auf den IDE-Anschluss wurde bewusst verzichtet, da das Kabel für den Anschluss sehr sperrig ist, zudem die Sicht blockiert und in aktuelleren Rechnern kaum mehr verbaut wird.

Die nachfolgenden Abbildungen verdeutlichen die Unterschiede zwischen einem *Apple*, einen Computer von *Dell* und einem PC.

Abbildung 4.1: *Apple* [10]

Abbildung 4.2: *DELL*

Abbildung 4.3: PC

4.1.2 Animationsentwürfe

4.1.2.1 Arbeitsspeicher

Bei der Animation des DDR-RAMs sind einige Punkte hervorzuheben. Es muss darauf hingewiesen werden, dass die beiden RAM-Riegel-Hebel nach unten geklappt werden müssen, bevor das Modul herausgelöst werden kann.

Ein RAM-Riegel hat sensible Stellen, die nicht berührt werden sollten. Bereits kleine statische Entladungen über die Haut können das gesamte Modul beschädigen. Daher wird im Video gezeigt, an welchen Stellen der User den RAM-Riegel gefahrlos berühren darf.

Beim Einstecken des neuen RAM-Riegels ist die eingestanzte Lücke zu beachten. Das Modul passt nur auf eine spezielle Art in den RAM-Slot.

Durch leichten Druck rasten die Hebel selbstständig in ihre Ausgangsposition ein. Wenn alle vollständig eingerastet sind, weiß man, dass sie richtig verbaut sind.

Falls zwei Module eingebaut werden sollen, müssen sie in die DUAL-Slot Konfiguration eingesetzt werden, um eine größere Leistungen zu erzielen. Diese sind meist farblich markiert. Wenn dies nicht der Fall ist, sind Slot 1 und Slot 3 sowie Slot 2 und Slot 4 gepaart [12].

4.1.2.2 Festplatte

Das *Antec*-Gehäuse unterscheidet sich in der Montage der Festplatte von anderen Gehäusen. Für das Video wurde die modernere *Antec*-Variante zu Grunde gelegt, da sie durch Transferleistung auf ein altes Gehäuse übertragbar ist und als zukunftsweisend gilt.

Bei der Montage der Festplatte muss darauf geachtet werden, dass sie richtig herum und erschütterungsfrei befestigt wird. Des Weiteren sollte gezeigt werden, wo die Festplatte verschraubt wird. Es sind zwei Anschlüsse erforderlich (der Stromanschluss des Netzteils sowie der SATA/IDE-Anschluss zum Mainboard) [12].

4.1.2.3 Grafikkarte

Die moderneren Grafikkarten unterscheiden sich in der Form, der Größe und im Aufbau ein wenig von den Grafikkarten der älteren Generationen. Sie sind im Allgemeinen sehr viel größer und haben eine große Kühlungsvorrichtung auf der Platine verschraubt und benötigen zusätzliche Stromanschlüsse.

Aus diesem Grund sollten zwei verschiedene Grafikkarten-Videos mit einer älteren und einer neueren Grafikkarte verfügbar sein.

Bei der neueren sollte explizit gezeigt werden, dass zusätzliche Stromanschlüsse anzuschließen sind.

Die Steckplätze der Grafikarten (AGP[27], PCI, PCI-Express) sind nur auf den ersten Blick ähnlich. Daher sollte in dem Video auf diesen Aspekt deutlich hingewiesen werden.

Beim Entfernen und Einstecken des Moduls in den Anschluss-Slot muss auf den Hebel geachtet werden, der sich nicht automatisch zurück schiebt. Auch dies sollte im Video hervorgehoben werden [12].

[27] AGP (engl.: Accelerated Graphics Port)

4.1.2.4 Mainboard

Der Einbau des Mainboards ist bei einem Computer das Komplizierteste. Im Video muss daher die richtige Reihenfolge der einzelnen Prozesse beachtet werden.

Zunächst muss gezeigt werden, wie die Blende für die Anschlüsse ausgewechselt wird. Diese Blenden haben eine Standardgröße, allerdings sind die ausgestanzten Öffnungen für die Anschlüsse nicht standardisiert. Daher wird bei jedem Mainboard eine passende Blende vom Hersteller mitgeliefert, die an das Gehäuse angebracht wird. Die vorherige Blende wird in Richtung Innenraum des Gehäuses herausgelöst und die neue von innen in die Form gedrückt, bis sie einrastet.

Die Gehäuse bestehen oft aus einem leitenden Material. Das Mainboard darf deshalb mit seinen Kontaktlötstellen nicht an das Gehäuse kommen. Damit dies nicht passiert, müssen am Gehäuse Abstandshalter eingeschraubt werden. Moderne Gehäuse haben leichte Ausbeulungen und können darauf verzichten. Je nachdem, ob es ein Mini ATX[28]- oder ein normales ATX-Mainboard ist, müssen 6 oder 9 Abstandhalter eingeschraubt werden. Aussehen und Ort sollten im Video deutlich hervorgehoben werden.

Nachdem die Abstandshalter angebracht wurden, wird das Mainboard mit bereits verbauten CPU und CPU-Lüfter (eventuell auch verbauten RAM-Riegel) an die richtige Position gelegt. Auch die Platine darf nur an bestimmten Punkten berührt werden. Im Video sollte gezeigt werden, dass man das Mainboard am CPU-Kühler am sichersten greift. Um Beschädigungen auf der Platinen-Unterseite zu vermeiden, sollte es zuerst mit den Anschlüssen in die Blende eingepasst und anschließend auf die Abstandshalter angelegt werden.

Wenn sich das Mainboard in der richtigen Position befindet, muss es mit den dafür vorgesehenen Schrauben auf die Abstandshalter montiert werden. In der Platine befinden sich Bohrungen in einem standardisierten Abstand.

Ist alles fest und ordnungsgemäß verschraubt, müssen das Netzteil und das Mainboard mit verschiedenen Anschlusskabeln verbunden werden. Hierzu gehört der Hauptstromanschluss (ATX-Power-Supply) und ein weiterer Stromanschluss für den Prozessor. Bei abweichenden Mainboards, wie in diesem Falle (6 PCI-Express Anschlüsse), benötigt das Mainboard einen zusätzlichen Stromanschluss (Molex 5V-Rail).

Abschließend müssen die am Gehäuse angebrachten Verbindungskabel an den ON/OFF Schalter und RESET Schalter, an die HDD[29]-LED sowie die Betriebs-LED[30] angeschlossen werden. Da diese bei jedem Mainboard anders aufgebaut sind, sollte drauf hingewiesen werden, dass diese Information in dem Handbuch des Mainboards zu finden ist [12].

[28] ATX (engl.: Advanced Technology Extended)
[29] HDD (engl.: Hard Drive Disk)
[30] LED (engl.: light-emitting diode)

4.1.2.5 Prozessor

Prozessoren sind sehr anfällig für elektrostatische Entladungen, daher sollte man sich vorher entweder erden oder einen Handschuh tragen. Diese Hinweise sollten in den Informationstexten mit eingebracht werden.

Der Prozessor muss in der richtigen Richtung eingesetzt werden. Dabei ist auf die Pfeil-Markierung am Rand jedes Prozessors zu achten.

Die neueren Prozessoren haben nur noch Kontaktstellen, die auf die Sockel gepresst werden. Ältere Generationen hingegen haben kleine Pins, die in die entsprechenden Öffnungen gedrückt werden müssen. Bei diesen ist es wichtig, den Prozessor ohne Kraftaufwand in den Sockel einzupassen. Wenn Kraft aufgewendet werden muss, sitzt er entweder falsch oder es sind bereits Pins verbogen.

Sitzt der Prozessor ordnungsgemäß an seinem Platz, wird ein Hebel nach unten gedrückt, der den CPU fest an den Sockel fixiert.

Anschließend muss der Lüfter eingesetzt werden. Dazu werden entweder Leitpaste oder „Templeitpads“ benötigt, die auf die Oberseite des Prozessors aufgetragen werden. Durch die Paste oder die Pads werden die Kühlrippen, die vom Lüfter gekühlt werden, wärmeleitend mit dem sich stark aufheizenden Prozessorkern verbunden.

Der Lüfter mit Kühlrippen wird auf das Mainboard um den CPU Sockel gesteckt und fest verschraubt [12].

4.1.2.6 Netzteil

Das Netzteil ist im Vergleich ein einfach ein- und auszubauendes Modul. Es wird an die dafür vorgesehene Öffnung im inneren hinteren Teil des Gehäuses angebracht. Mit vier Schrauben wird es von außen fixiert. Bei den Stromanschlüssen kann man nahezu nichts falsch machen, sie sind alle so geformt, dass sie nur für die vorgesehenen Anschlüsse passen. Im Video sollte daher hauptsächlich gezeigt werden, welche Anschlüsse notwendig sind und wo sie angebracht werden.

Es muss darüber hinaus auf den Hebel des Hauptstromkabels am Mainboard hingewiesen werden, da bei unsachgemäßer Bedienung der Stromanschluss nicht abgezogen werden kann.

4.1.3 Schematischer Ablauf einer 3D-Modellierung

Es gibt mehrere Methoden, um 3D-Modelle herzustellen. Die folgende schematische Darstellung beschreibt die beispielhafte Erstellung eines Mainbords.

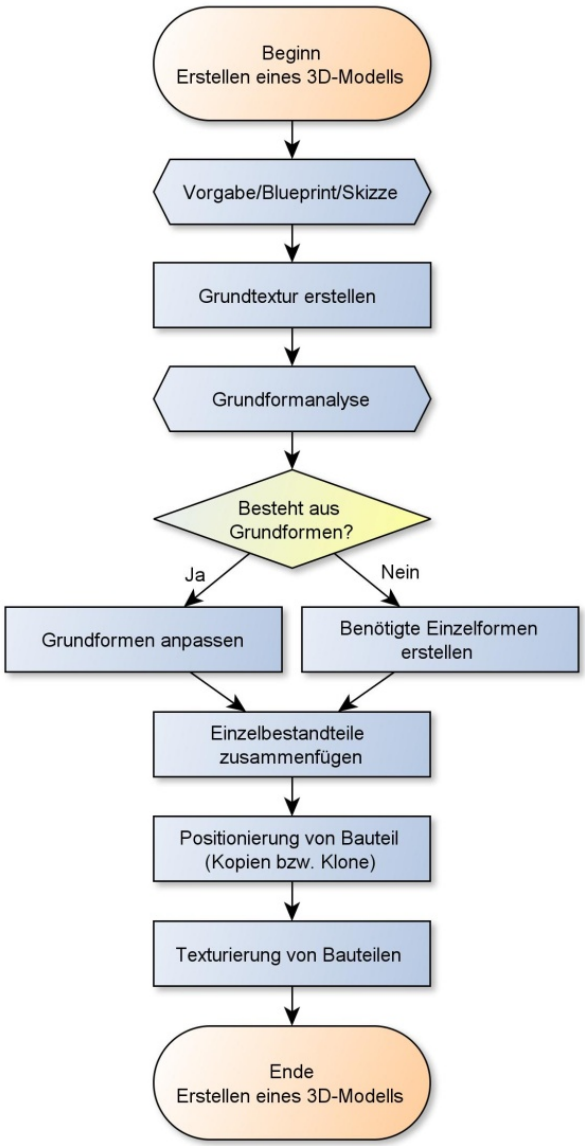


Abbildung 4.4: Ablaufplan zur Erstellung eines 3D-Modells

1. Es wird eine Vorgabe gesucht oder ein Modell entwickelt. In diesem Fall war das Mainboard vorgegeben (*Asus Maximum IV Extreme*) und musste vermessen werden. Hierzu wurden Skizzen angefertigt und das Mainboard fotografiert.

2. Im zweiten Schritt muss entschieden werden, welche Details und Bestandteile als 3D-Modell dargestellt werden und welche als Texturen nur den Eindruck eines 3D-Modells vermitteln sollen. Dies hängt meist davon ab, wie exakt die „Grundtextur“ gestaltet werden kann. Bei diesem Mainboard diente die Platine als Grundtextur. Realistischer wäre es gewesen, auf eine hochauflösende Textur von der unbestückten Platine

zurückzugreifen. Da *Asus* allerdings eine solche Textur nicht zur Verfügung stellen konnte und der Zeitaufwand, eine Textur zu zeichnen sehr aufwändig gewesen wäre, wurde das Mainboard mit den Bauteilen fotografiert. Dies funktioniert dann gut, wenn die entsprechenden 3D-Modelle die „aufgemalten" Bauteile verdecken.

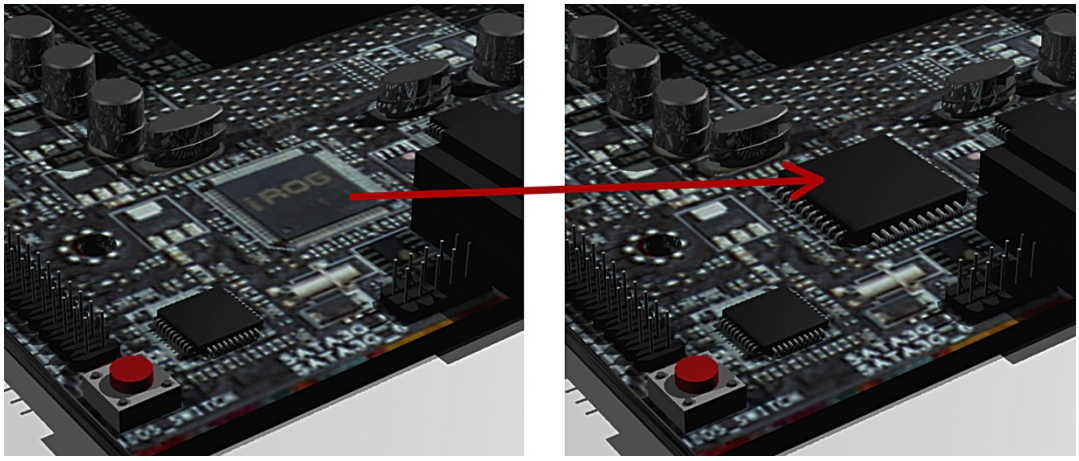


Abbildung 4.5: Veranschaulichung verdeckter Texturelemente

3. Im 3D-Programm (in diesem Fall *3ds-Max*) wird eine Grundplatte in den Maßen des Mainboards erstellt. Mit dem Material-Editor wird die Textur eingebunden und auf die Grundplatte gelegt. Bei einer rechteckigen Textur und einer rechteckigen Platte muss oft nur die Rotation angepasst werden. Bei runden Platten oder Modellen mit Volumen kann dies kompliziert werden. Dabei muss oft mit einem Bildbearbeitungsprogramm die Textur zerschnitten, verformt und einzeln eingebunden werden.
   Durch die Textur ist auch eine Art Blueprint gelegt worden, weshalb es ratsam ist, diesen Schritt immer als erstes vorzunehmen.

4. Als nächstes werden die häufigsten Bauteile modelliert. In diesem Fall sind es die Kondensatoren und Widerstände, wobei die Widerstände so klein und flach sind, dass es nicht unbedingt notwendig ist, sie zu modellieren. Der Kondensator wird zuerst auf die Form hin untersucht. Ist es eine sehr abstrakte Form oder folgt sie einer der Grundformen?
   Grundformen bei dem Programm *3ds-Max* sind: Box, Sphere, Cylinder, Torus, Cone, Geosphere, Tube, Pyramid, Hedra, CharmferBox, CharmferCylinder, Capsule, Gengo und Spindle.
   Entspricht das Bauteil einer dieser Formen, kann sie aus Einzelteilen zusammengefügt oder subtrahiert werden.

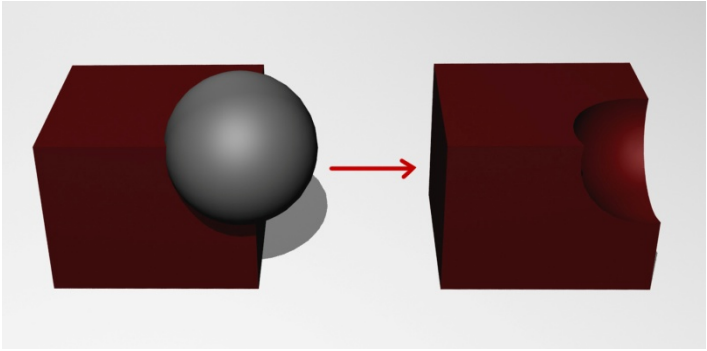


Abbildung 4.6: Veranschaulichung einer Subtraktion aus Grundformen

Ist dieses nicht der Fall, beginnt man mit einer Grundform und „zieht“ Polygonschicht für Polygonschicht heraus, um so den gewünschten Körper zu gestalten.

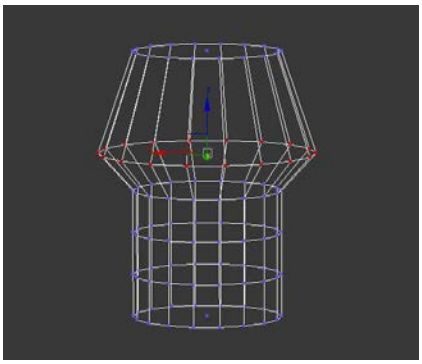


Abbildung 4.7: Polygonschichten

In diesem Fall kann ein Kondensator grob aus einem “CharmferCylinder“ (abgerundeter Zylinder) und drei normalen Zylindern geformt werden.

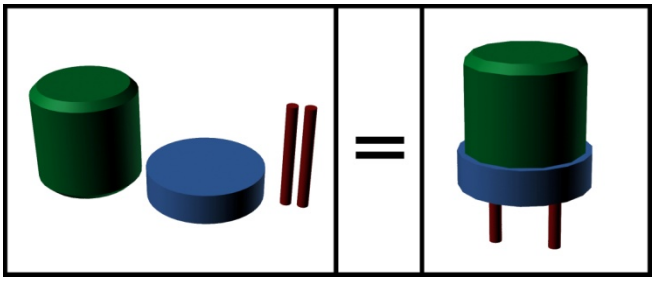


Abbildung 4.8: Kondensator aus Grundformen

Um Polygone zu sparen, sollte die Polygonzahl so eingestellt werden, dass das Objekt zwar echt aussieht, jedoch nicht zu detailreich abgebildet wird. Bei Booleans werden durch das Programm nahezu immer unwichtige Polygone hinzugefügt. Diese sollte man an dieser Stelle frühzeitig beseitigen, da eine nachträgliche Korrektur kaum möglich ist. Ein komplett selbst gestaltetes Modell ist polygonsparender. Modifikationen sind eine nützliche Hilfe, allerdings beinhalten sie oft auch "Nebenwirkungen".

Die einzelnen Grundformen werden nach Größe und Form an den vermessenen Kondensator angepasst und an die entsprechenden Plätze gezogen und gruppiert.

5. Das Modell des Kondensators wird an die richtige Stelle auf das Mainboard gesetzt und entsprechend durch Kopien vervielfältigt. Beim Vervielfältigen ist drauf zu achten, dass jede Kopie ein Klon des ersten Modells ist und nachträglich alle Änderungen des ersten Modells übernimmt. Dies ist für das spätere Texturieren sehr wichtig.

6. Diese Verfahrensweise wendet man nun auf alle Bauteile an. Um Rechenleistung zu sparen, sollten während des Modellierens bereits bestehende Bauteile ausgeblendet und nur zur Übersicht bzw. zum Gesamteindruck wieder eingeblendet werden.

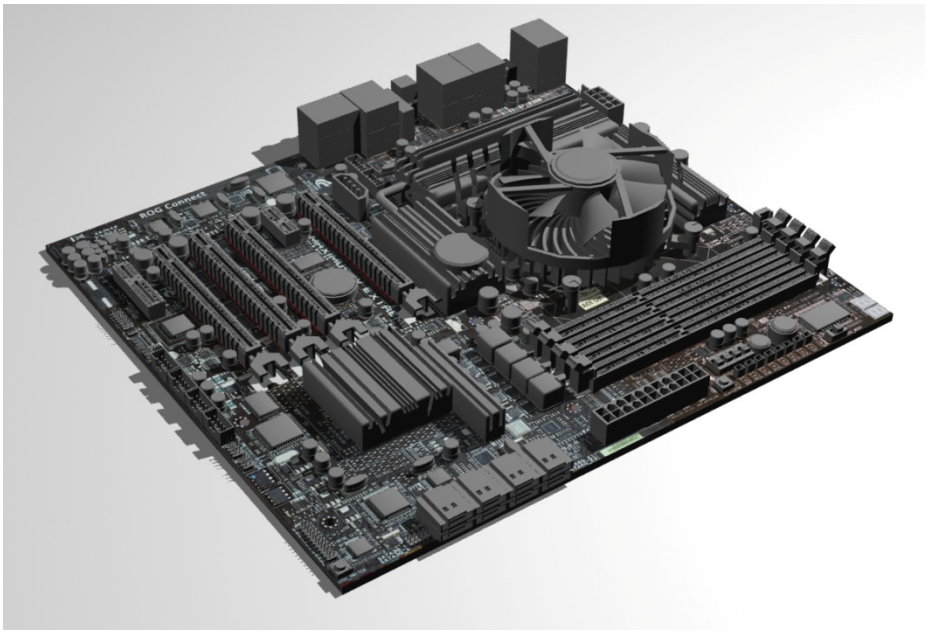


Abbildung 4.9: Mainboard mit positionierten Einzelteilen

7. Stehen alle 3D-Modelle an ihrem Platz, beginnt das Texturieren. Hierzu wird der Material-Editor geöffnet und für jede Textur und Farbe eines Bauteils ein Texturball erstellt. Die entsprechende Textur wird eingebunden und Luminanz, Glanz, Leuchten, Reflexion und die Formatierung der Textur eingestellt. Die Vorlage wird anschließend auf das erste Objekt der Klon-Reihe übertragen.

Abbildung 4.10: Mainboard mit texturierten Einzelteilen

8. Zum Schluss muss alles sinnvoll gruppiert und entsprechend bezeichnet werden. Das Mainboard kann nach Fertigstellung als ein komplettes Bauteil eingebunden werden.

4.1.4 Orthogonale Ansicht

Wie im Kapitel 4.1.3 beschrieben, muss von der Mainboard-Platine eine orthogonale Textur erstellt werden. Dies ist in der Umsetzung nur bedingt möglich, denn man kann sich dieser Sicht nur annähern.

Um zu verstehen, wie ein orthogonales Foto erzeugt wird, muss zunächst erläutert werden, was orthogonales Sehen bedeutet.

Kameras sowie Augen sehen durch eine Linse. Das bedeutet, sie haben einen Sehwinkel vom Mittelpunkt der Linse ausgehend. Dieses "verzerrt" das Bild der Umwelt und bewirkt beim Betrachter ein ansatzweise rundförmiges Sehen. Z. B. haben viele Tiere andere Winkelbreiten, in denen sie die Umwelt völlig anders wahrnehmen als Menschen [11].

Ein Vergleich der menschlichen Sicht mit der Sicht eines Turmfalken:

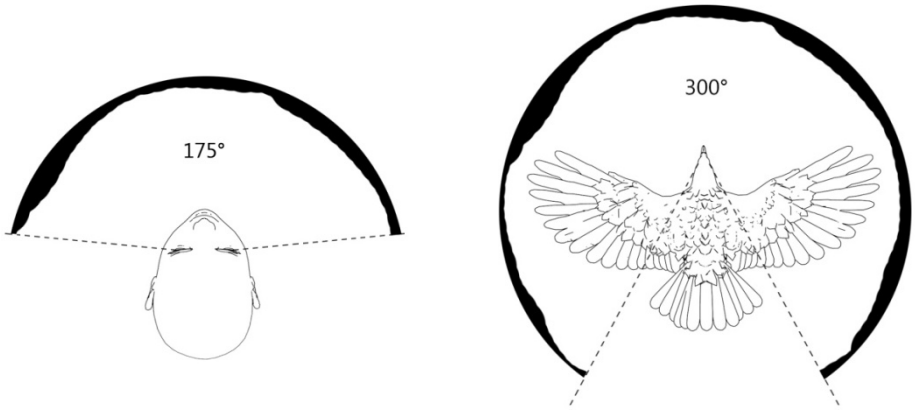


Abbildung 4.11: Vergleich: Sehwinkel zwischen Mensch und Turmfalke

Wenn Menschen orthogonal ihre Umwelt betrachten würden, ginge das perspektivische Sehen verloren. Bei einem Objekt sieht das Auge also nur einen einzigen Punkt in der Mitte orthogonal, alle anderen jedoch verzerrt [11]. In den Skizzen 4.13 und 4.14 ist der Unterschied dargestellt.

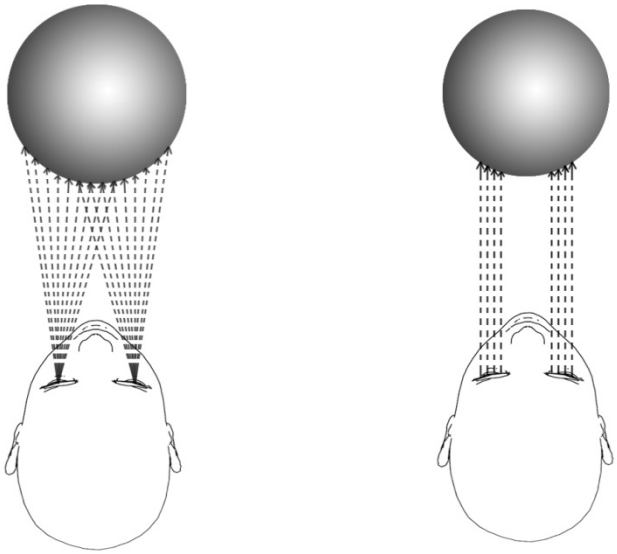


Abbildung 4.12: Vergleich: Perspektivisches Sehen und Orthogonales Sehen

Die Mitte wird also „gerade“ betrachtet und alle Bereiche des Objektes, die sich von diesem Punkt entfernen, perspektivisch. Diese Eigenschaft würde beim Fotografieren stören, da, wie bei unserem Beispiel, die Bauteile des Mainboards - von der Mitte aus - nach außen wegzeigen würden. Um dies zu verdeutlichen, ist das folgende Beispielobjekt einmal orthogonal und einmal perspektivisch aus einer Draufsicht dargestellt.

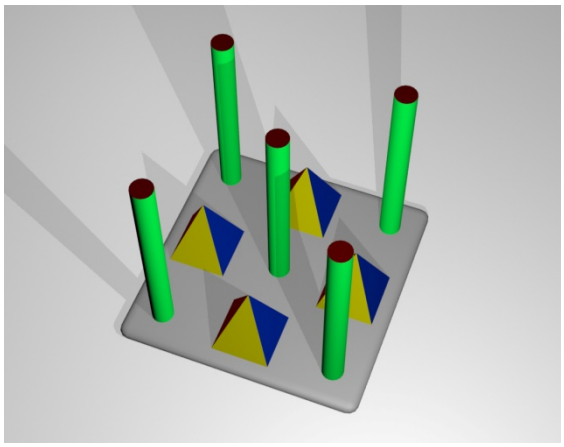


Abbildung 4.13: Veranschaulichungsmodell

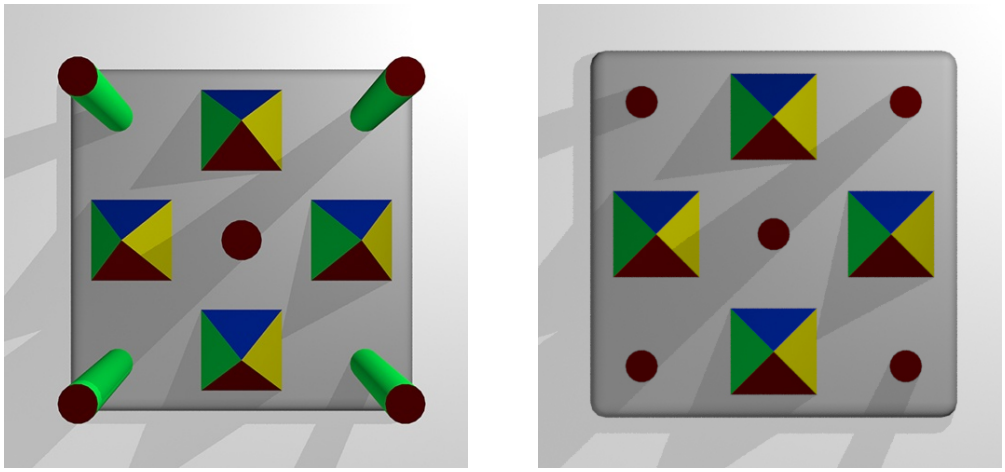


Abbildung 4.14: Vergleich: Perspektivisches Sehen und Orthogonales Sehen

Beim Vergleich der Bilder bemerkt man deutliche Unterschiede. Zum einen erscheinen die Zylinder, die in Richtung Kamera ragen, deutlich breiter im Durchmesser. Ebenfalls sind die Zylinderseiten zu erkennen (grün). Die Farben der Pyramidenseiten sehen nicht mehr gleichverteilt aus, bzw. die Spitzen scheinen nicht mehr zentrisch zu sein. Menschen sind in der Lage, dieses intuitiv zu interpretieren. Jedoch ist dieser Effekt unerwünscht für 2D-Texturen mit aufgesetzten 3D-Bauteilen. In unserem Beispiel würden Teile des Mainboards durch Bauteile verdeckt werden. Wenn anschließend Bauteile im richtigen Maßstab und der korrekten Position darauf appliziert werden, erscheinen die Texturen der Bauteile unter den 3D-Modellen hervor.

Um ein Foto zu erhalten, welches als Textur benutzt werden kann, muss das Motiv aus großer Entfernung mit einem Weitwinkel-Objektiv aufgenommen werden. Im Kapitel 5.1.1 wird noch einmal darauf eingegangen.

4.1.5 Ablaufplan (Erstellung von Anleitungsvideos)

Die Bearbeitungsschritte des 3D-Modellierens und 3D-Animierens wurden in dem vorhergehenden Kapitel ausführlich beschrieben. Jedoch reichen diese Schritte nicht aus, um ein vollständiges Video zu erstellen. In dem folgenden Schaubild ist der Verlauf von der Recherche bis zum fertigen Video abgebildet.

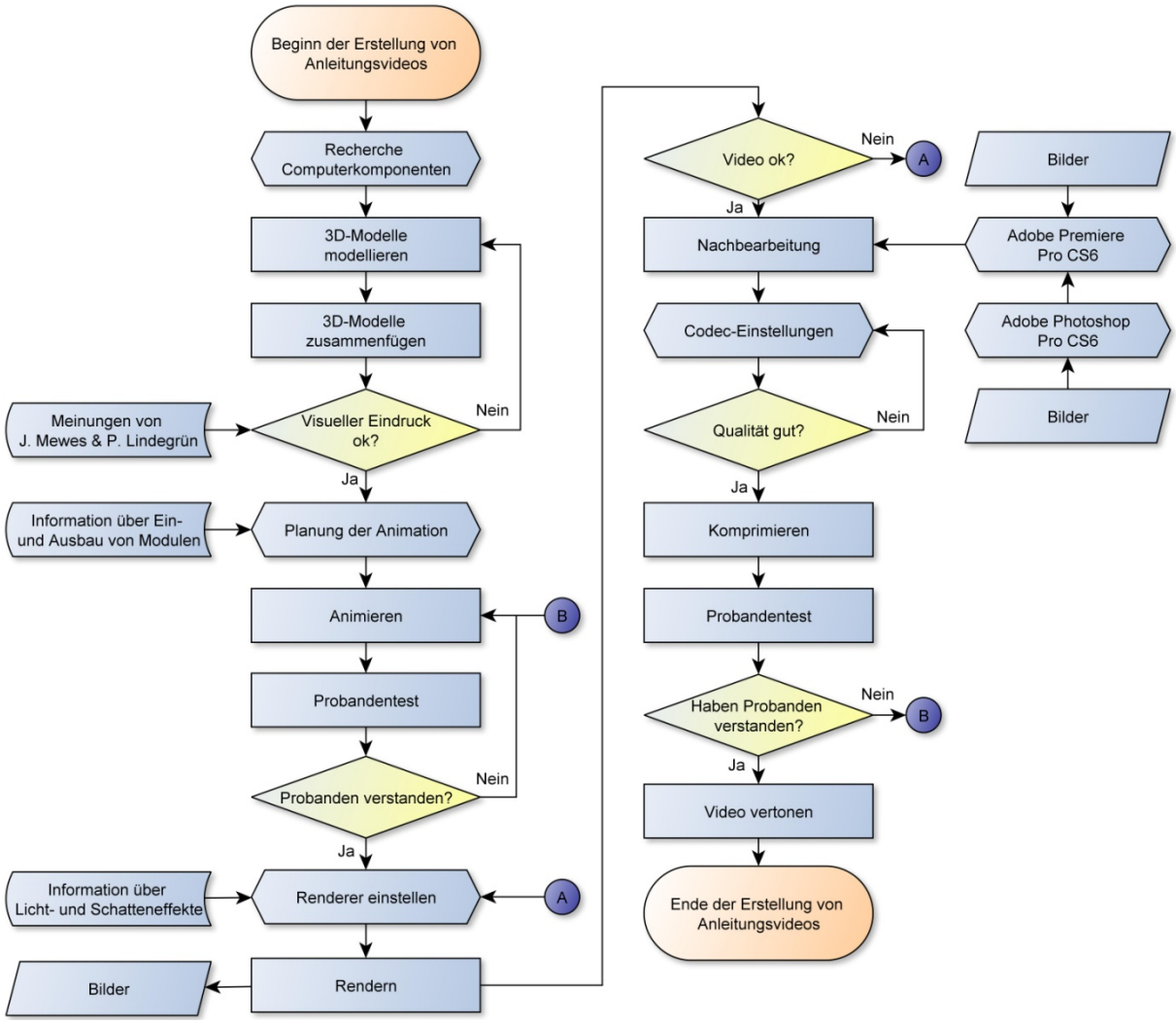


Abbildung 4.15: Ablaufplan "Erstellung von Anleitungsvideos"

Nach der Recherche werden die 3D-Modelle erstellt, die zusammengefügt und überprüft werden. Im Anschluss daran werden die Animationen geplant und umgesetzt (vgl. Kapitel 4.1.2). Nach einer Verständnisprüfung der Animationen durch Testpersonen werden diese gerendert und als Bilder gespeichert. Die einzelnen Bilder werden mit *Adobe Photoshop Pro CS6* und *Adobe Premiere Pro CS6* bearbeitet und zu einem Video zusammengefügt, codiert und komprimiert. Nach der Evaluation durch einen weiteren Probandentest werden die Videos durch Geräuscheffekte aufgewertet. Die fertigen Videofiles können dann in das TPHC Programm eingebunden werden.

4.1.6 Ablaufplan (Erstellung des Userinterface-Designs)

Das Userinterface wird von beiden Studenten als Team erstellt. Der Ablaufplan ist in der folgenden Grafik dargestellt.

Abbildung 4.16: Ablaufplan "Erstellung des Userinterface-Designs

Zunächst werden unabhängig voneinander zwei eigenständige Designs entwickelt. Diese werden jeweils von Probanden getestet und verbessert. Wenn beide Designs fertiggestellt wurden, werden Vor- und Nachteile der jeweiligen Lösungen untersucht und gegenübergestellt.

Die Vorteile werden kombiniert und Nachteile, wenn möglich, beseitigt. Das entworfene Interface wird anschließend im Team umgesetzt und implementiert. Die gemeinsame Lösung soll ein harmonisches Ganzes ergeben und zu einem stimmigen und anwenderfreundlichen Userinterface führen.

Als Hauptinformationsquelle diente dabei die Vorlesung "Userinterface-Design", die in der HAWK-Göttingen stattfindet.

4.1.7 Probandentest

Um das Projekt erfolgreich abzuschließen und am Ende ein Produkt zu erhalten, welches den Kundenwünschen entspricht, müssen Testläufe mit Probanden durchgeführt werden (Beta-Phase). Durch die Kritik der Testpersonen erhält man praktische Erfahrungsberichte, welche Ansätze verständlich und welche noch unverständlich waren. Es können zuvor nicht aufgefallene Fehler und Verständnisschwierigkeiten korrigiert und Verbesserungsvorschläge der Testpersonen in die Umsetzung integriert werden. Diese Phase verschafft einen Einblick, wie die Probanden mit dem Programm zurechtkommen. Diese Evaluation ist ebenfalls nützlich für zukünftige Projekte.

Das folgende Ablaufschema gibt Aufschluss darüber, welche Möglichkeiten dem Probanden und dem Beobachter während der Testphase eröffnet werden. So wird dem Probanden, ohne ihm konkrete Vorinformationen mitzuteilen, die Aufgabe gestellt, eine Hardware an einem PC auszutauschen. Ihm werden die entsprechenden Werkzeuge sowie ein Computer, auf dem das Lernprogramm gestartet wird, zur Verfügung gestellt. Der Proband muss sich anschließend allein im Programm zu Recht finden und versuchen, die Aufgabenstellung zu erfüllen. Anschließend beantwortet der Proband einen Fragebogen. Der Testbeobachter notiert ebenfalls, was ihm während der Testphase aufgefallen ist.

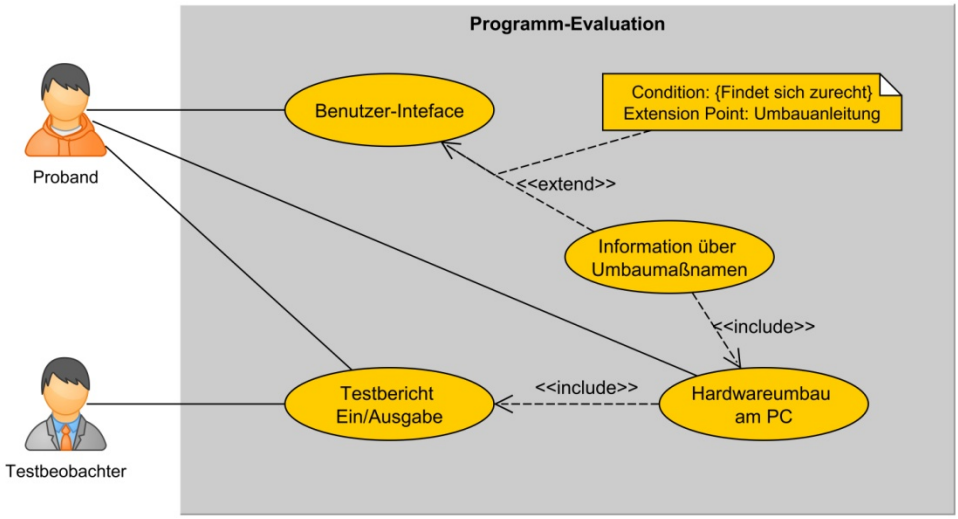


Abbildung 4.17: Use-Case TPHC Test

Nach dem Test werden alle Fragebögen ausgewertet und ggf. Änderungen am Programm bzw. Video vorgenommen. Die Ergebnisse und Auswertungen sind im Kapitel 6.1.1 dargelegt.

4.1.8 Zusammenfassung des Kapitels

In diesem Kapitel wurden die Konzepte der einzelnen Animationen ausführlich beschrieben. Des Weiteren wurde ein schematischer Ablauf zur Erstellung eines 3D-Modelles aufgezeigt. Ein

Muster-Computer wurde ausgewählt und die Theorie der orthogonalen Ansicht erklärt, um den im Kapitel 5 vorkommenden Aspekt „Fotografieren und vermessen des zu modellierenden Computers“ verstehen zu können. Anschließend wurde ein Ablaufplan zur Erstellung der Anleitungsvideos vorgestellt und erläutert.

Im Kapitel 5 wird genau beschrieben, wie die Konzepte und Recherchen im Einzelnen praktisch umgesetzt wurden.

4.2 Paul Lindegrün

Im letzen Kapitel wurden die nötigen Methoden und Tools für das Projekt ausgewählt. Anhand dieser sollen in diesem Kapitel Konzepte entwickelt werden, an denen sich während der Implementierung orientiert werden soll. Zum einen ist dies der konkrete Aufbau des Interfaces und zum andren der Programmablauf der zu erstellenden Software.

4.2.1 Anwendungsfälle (use cases)

Der Anwender kann die Hintergrundprozesse, die im Programm stattfinden, nicht sehen. Er kann lediglich auf die für ihn bereitgestellten Interaktionsmöglichkeiten zugreifen. Im Folgenden wird kurz erläutert, welche Möglichkeiten der Anwender für die Interaktion besitzt.

Der Anwender hat die Möglichkeit, aus mehreren Bauteilen das für ihn zutreffende auszuwählen. Daraufhin wird die Anzeige, um die für das Bauteil spezifischen Informationen, erweitert. Weiterhin ist es dem Anwender möglich, sich Informationen über das TPHC Programm anzeigen zu lassen. Beim Ansehen des Videos hat er die Wahl es wiederzugeben, zu pausieren, zu stoppen, zu einer beliebigen Stelle zu spulen und die Lautstärke zu verändern.

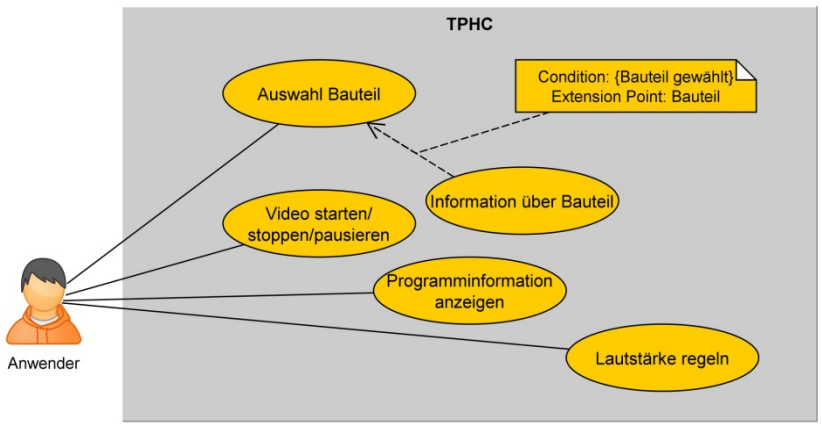


Abbildung 4.18: Anwendungsfälle

4.2.2 Aufbau des Interfaces

Im Kapitel 3.2.7 wurde bereits der Entwurf "Weiterleitung zur nächsten Ebene" ausgewählt. Nun soll dieser Entwurf konkretisiert werden. Dabei soll die genaue Anzahl an Steuer- und Anzeigeelementen sowie die Positionen festgelegt werden.

Ebene 1 (Auswahl-/Startebene): Auf dieser Ebene befindet sich die Auflistung der Komponenten. Jede Komponente wird dabei durch einen Button symbolisiert. Die Komponenten sind:

- Prozessor
- Mainboard
- Grafikkarte
- Arbeitsspeicher
- Netzteil
- DVD-Laufwerk
- Festplatte
- Komplettbau

Diese Buttons werden rasterartig in der Ebene angeordnet. Durch Klicken auf eine der Komponenten wird die entsprechende Information geladen und in der zweiten Ebene angezeigt. Zusätzlich befindet sich in der unteren rechten Ecke der Ebene ein Info-Button. Beim Klicken des Info-Buttons wird ein neues kleines Fenster mit Informationen über das Programm selbst und dessen Benutzung angezeigt. Dieser Button ist im allen drei Ebenen sichtbar.

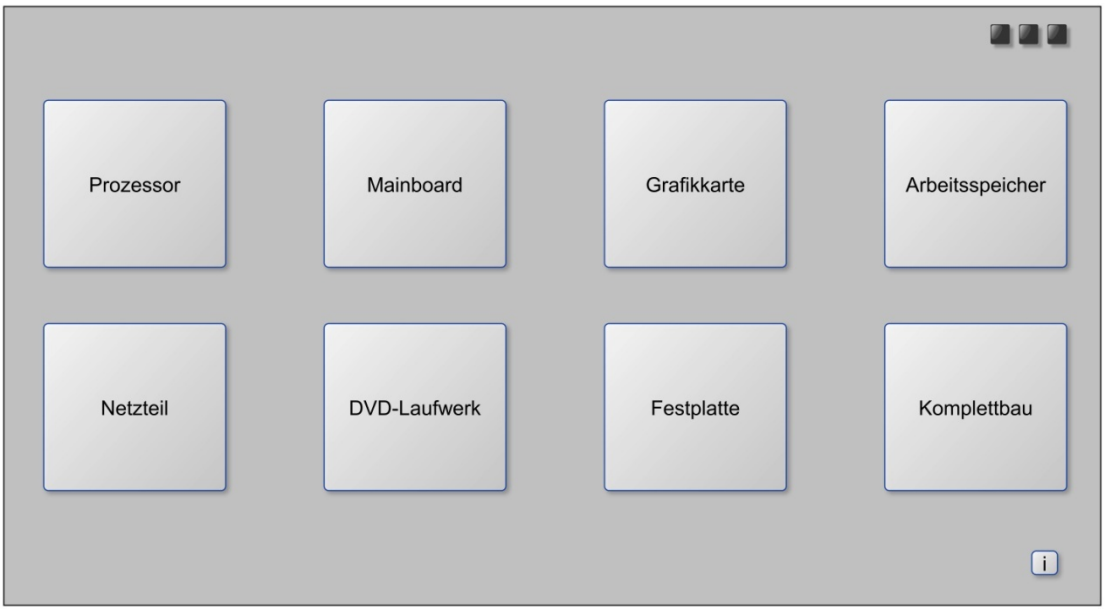


Abbildung 4.19: Konkreter Entwurf: Ebene 1

Ebene 2 (Informationsebene): In der rechten Hälfte dieser Ebene befindet sich ein Panel, welches die Informationen und Hinweise über das ausgewählte Bauteil anzeigt. Links in der Ebene befinden sich ein weiteres Panel, das ein Bild der ausgewählten Komponente anzeigt und ein

Button, der zur dritten Ebene weiterleitet und das Video startet. In der unteren linken Ecke gibt es einen Zurück-Button und einen Home-Button. Der Zurück-Button leitet immer zur vorherigen Ebene weiter, während der Home-Button zur ersten Ebene (der Startebene) weiterleitet.

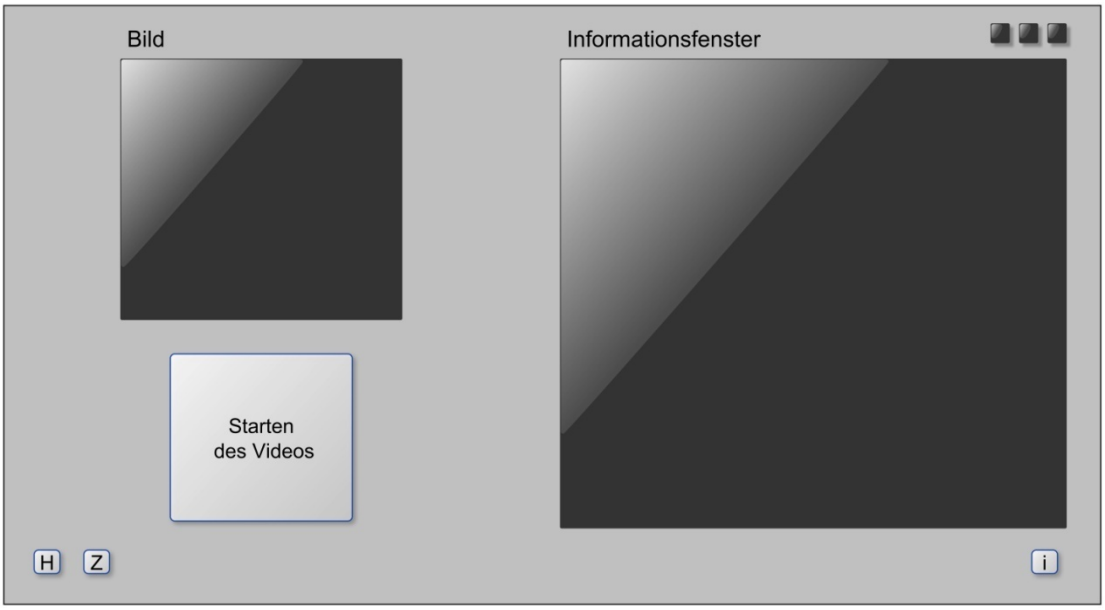


Abbildung 4.20: Konkreter Entwurf: Ebene 2

Ebene 3 (Videoplayerebene): Hier wird das abzuspielende Video in einem großem Panel dargestellt. Dieses erstreckt sich fast über die gesamte Fenstergröße. Eine Zeitleiste, welche den Video-Fortschritt anzeigt, befindet sich direkt unter dem Video-Panel. Die Steuerelemente für das Video werden mittig unter der Zeitleiste angeordnet. Es sind vier Buttons, die folgende Funktionen übernehmen:

- Wiedergabe/Pause
- Stopp
- Vorspulen
- Zurückspulen

In der rechten unteren Ecke, neben dem Info-Button, befindet sich die Lautstärkeregelung. Diese besteht aus 2 Steuerelementen:

- Button zum Stummschalten
- Leiste für die Lautstärke

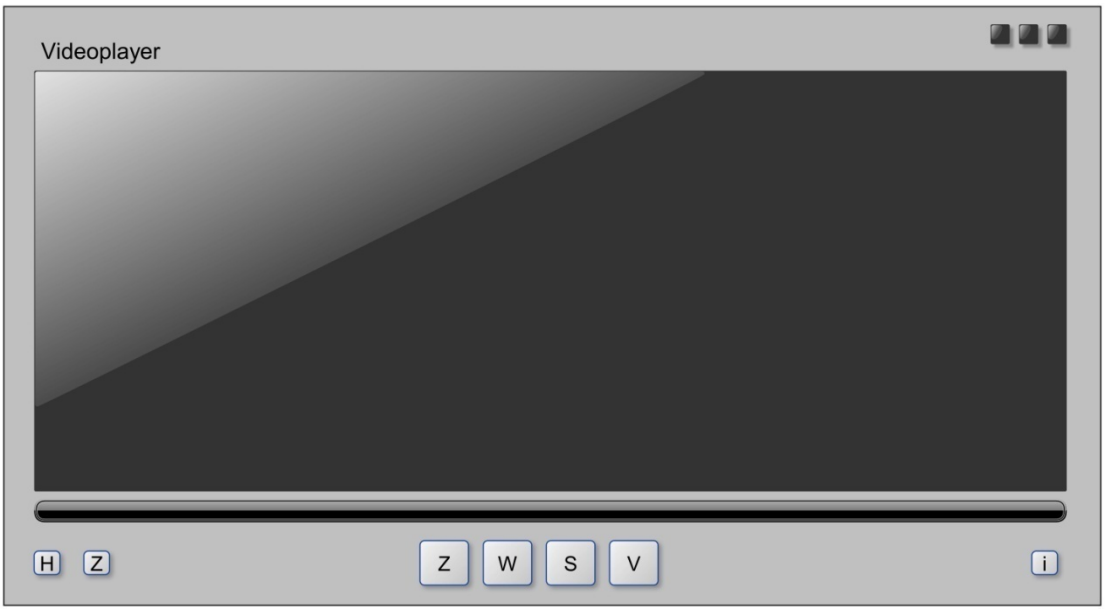


Abbildung 4.21: Konkreter Entwurf: Ebene 3

4.2.3 Schematischer Ablauf des Programms

Nach dem Starten des Programms wird zunächst das Hauptfenster initialisiert. Es wird die Größe, Startposition sowie die Hintergrundtextur festgelegt. Danach erfolgt die Initialisierung der Steuerelemente des Fensters. Das sind unter anderem Buttons, durch die der Anwender mit dem Programm interagieren kann und Panels als Anzeigeelemente, die dem Anwender Informationen ausgeben. Wie beim Hauptfenster wird auch bei den Steuerelementen die Größe, Startposition und gegebenenfalls eine Hintergrundtextur gesetzt.

Nach der Initialisierung wird das Startfenster des Programms angezeigt. Hier kann der Anwender ein Bauteil auswählen, über das er sich informieren möchte. Hat er sich für ein Bauteil entschieden und dieses ausgewählt, werden die dazugehörigen Informationen aus einer Datei ausgelesen und in einem Panel angezeigt.

Nun befindet sich der Anwender im zweiten Fenster und kann sich Informationen über das ausgewählte Bauteil ansehen. Des Weiteren hat er die Möglichkeit zum Startfenster zurückzukehren, um ein anderes Bauteil auszuwählen oder ein Video zu dem entsprechenden Bauteil zu starten.

Sollte der Anwender sich dafür entschieden haben, das Video anzusehen, wechselt das Programm in die dritte Ansicht: den Videoplayer. Der Videoplayer bietet dem Anwender die grundlegenden Funktionen, um ein Video zu steuern. Das Video kann angehalten, gestoppt, wiedergegeben, zurück- und vorgespult werden. Außerdem kann die Lautstärke verändert werden.

Parallel dazu gibt es nach der Initialisierung die Möglichkeit, das Programm zu beenden. Dies geschieht durch Klicken auf das “X“ in der rechten oberen Ecke des Fensters.

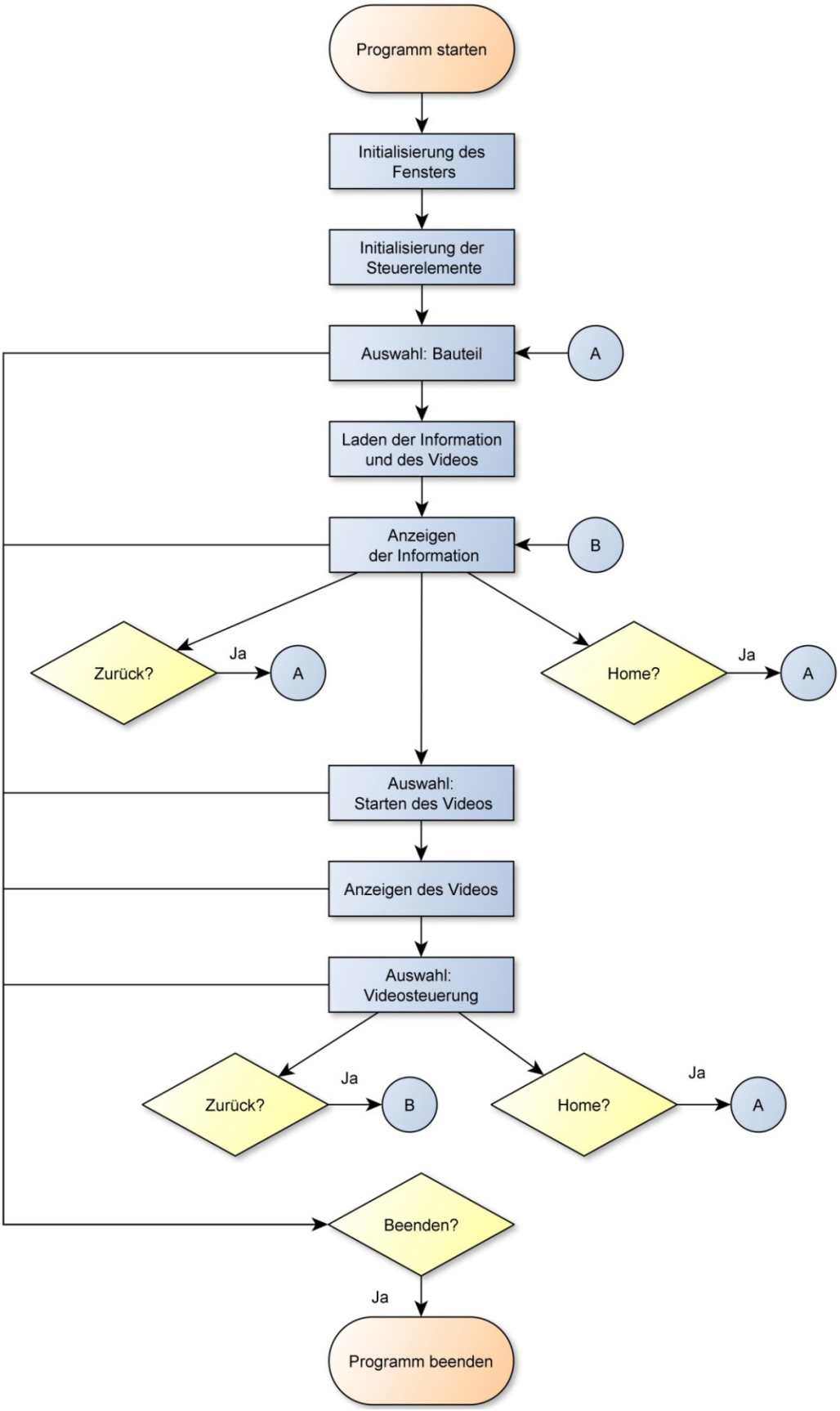


Abbildung 4.22: Ablaufplan des Programms

4.2.4 Ablaufplan der Programmierung

Da jetzt alles Wesentliche bezüglich des Interfaces und des Programmablaufs geklärt ist, muss noch ein Arbeitsplan für die Programmierung erstellt werden, an dem sich während der Projektarbeit orientiert werden soll.

Zuerst ist ein Grundgerüst zu erstellen, welches nur die erste Ebene enthält. Auf dieser Ebene befinden sich nur Buttons ohne Funktionen.

Danach wird das Grundgerüst, in einem iterativen Vorgang, um weitere Funktionalitäten erweitert. Zuerst wird die zweite Ebene, auf der sich die Informationsanzeige und die Weiterleitung zum Videoplayer befinden, hinzugefügt und um die Funktionalität erweitert, zwischen den beiden Ebenen zu wechseln. Als nächstes wird das Programm um die dritte Ebene, auf der sich die Anzeige- und Steuerelemente für den Videoplayer befindet, erweitert und die Möglichkeit hinzugefügt, zu den beiden vorherigen Ebenen zu wechseln.

Die Steuer- und Anzeigeelemente werden mit Texturen versehen, die während der Entwicklung des “Userinterface-Designs“ im Kapitel 5 entstehen.

Als nächstes werden der Videoplayer und die entsprechenden Funktionalitäten in das Programm integriert.

Zum Schluss wird das Programm getestet und etwaige Fehler ausgebessert.

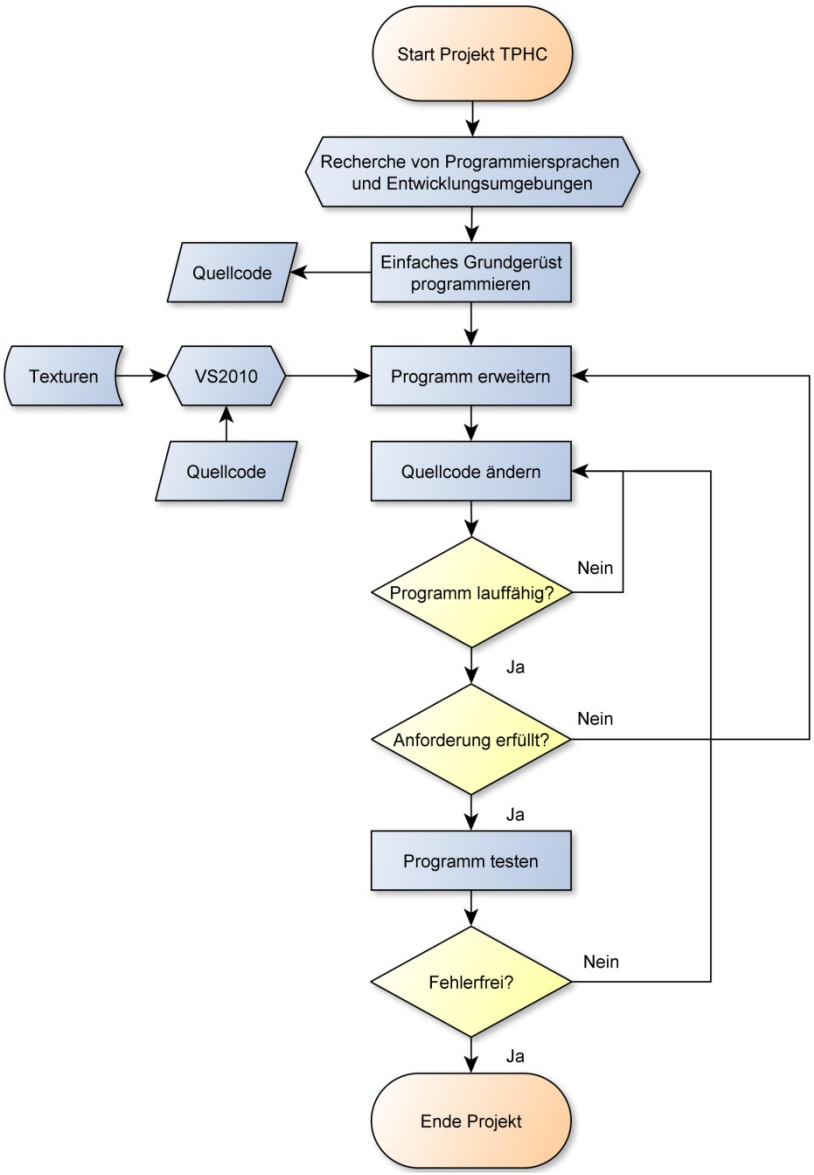


Abbildung 4.23: Ablaufplan der Programmierung

4.2.5 Zusammenfassung des Kapitels

In diesem Kapitel wurde das Interface konkretisiert. Es wurden der Aufbau und die Position der einzelnen Anzeige- und Steuerelemente des Programms festgelegt und ein konkreter Ablaufplan des Programms wurde erstellt. Zusätzlich wurde ein Ablaufplan aufgestellt, an dem sich während der bevorstehen Implementierung orientiert werden soll.

Im nächsten Kapitel wird die Implementierung detailliert beschrieben.

5 Implementierung

5.1 Jannik Mewes

In diesem Kapitel wird genau beschrieben, wie das erstellte Konzept einzeln umgesetzt worden ist.

5.1.1 Fotografieren und vermessen des zu modellierenden Computers

Um das 3D-Modell möglichst realistisch an dem Original zu halten und einen eindrucksvollen visuellen Eindruck zu gewinnen, wurden alle Bauteile vermessen.

Hierzu wurde der PC komplett auseinander gebaut und jedes Einzelteil per Hand mit einem Lineal vermessen und in Skizzen festgehalten.

Des Weiteren wurden die Platinen fotografiert. Da die beiden Unternehmen keine Platinen-Abbildungen des Mainboards und der Grafikkarte zur Verfügung stellen konnten und auch keine entsprechenden im Internet zu finden waren, fertigte ich eigene Bilder der bestückten Platinen (Vor- und Rückseite) an. Dazu benötigte ich eine hochauflösende Kamera, mit der ich aus weiter Entfernung Fotos von den jeweiligen Seiten aufnahm.

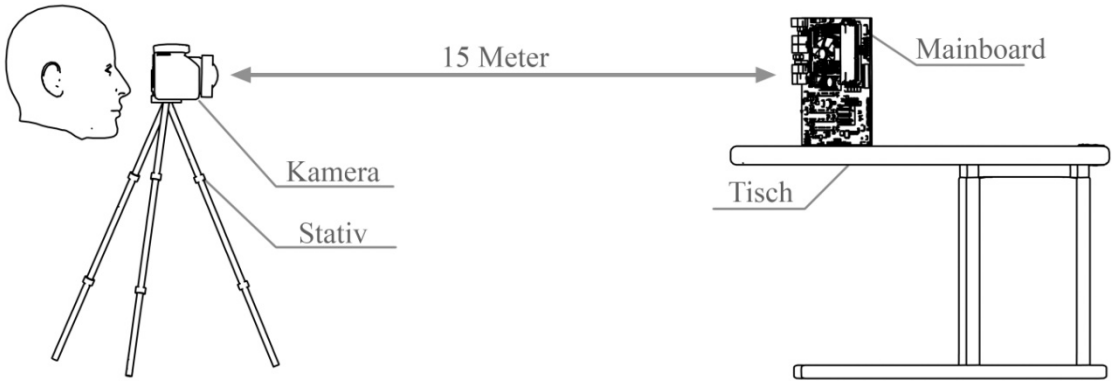


Abbildung 5.1: Aufbau des Aufnahmeprozesses

Benutzt wurde hierzu eine *Canon D600* mit einem 50 mm Festbrennweiten-Objektiv. Die Bilder wurden in einem Abstand von 15 m fotografiert, um eine möglichst orthogonale Aufnahme zu erzeugen (vgl. Kapitel 4.1.4).

Im Anschluss wurde das Bild mit *Photoshop* in Schärfegrad, Licht, Optik und Kontrast so bearbeitet, dass es gut auf ein 3D-Modell passte. Da es sehr schwierig war, auf 15 m Entfernung eine Platine exakt gerade aufzunehmen, wurde das Bild mit einer Photoshop-Funktion perspektivisch angepasst sowie ausgeschnitten. Näheres hierzu im Kapitel 5.1.2.4.

5.1.2 3D-Modellierung von PC-Bestandteilen

Nachdem die Bestandteile fotografiert und vermessen wurden, müssen die einzelnen 3D-Modelle erstellt werden.

Da sich beim Modellieren die Prozesse sehr oft wiederholen, sei hier auf die Methodik des vorgegangenen Kapitels 4.1.3 verwiesen. Dort wird allgemein erklärt, wie ein 3D-Modell erstellt wird.

An dieser Stelle werden exemplarisch an einigen Objekten einige besondere Modellierungsprozesse aufgezeigt.

5.1.2.1 Beispiel der 3D-Modellierung von einer Gewindeschraube

Um die Gewinde im Gehäuse, beim DVD-Laufwerk, für die Festplatte sowie bei den Abstandshaltern zu erzeugen, mussten zunächst die dazugehörigen Schrauben modelliert werden. Die Schrauben wurden aus den zwei Grundbausteinen "Zylinder" erstellt. Der Zylinder für das Außengewinde musste in 10 Höhensegmente und in 30 Umfangssegmente unterteilt werden. In jedem zweiten Höhensegment wurden alle Polygone ausgewählt und orthogonal voneinander entfernt. Diese Segmente erfuhren anschließend mit dem Smooth-Modifikator eine Glättung und wirken nun auf den ersten Blick wie ein Außengewinde. Der zweite Zylinder wurde in 2 Höhensegmente unterteilt und das obere Segment klein skaliert. Die Umfangssegmente mussten auf 6 reduziert werden, da der Kopf der Standard-Gehäuseschrauben eine Sechskantform aufweisen. Nun wurden 2 gleichförmige Quader über Kreuz in die obere schmalere Seite des Zylinders geschoben. Sie sollen später die Löcher für den Schraubendreher bilden. Dazu kommt noch ein Sechskantzylinder, der passgenau auf den vorherigen Sechskantzylinder aufgesetzt wurde und ein klein wenig vertieft ist. Die Rechtecke und der runde Zylinder wurden zu einem Objekt zusammengefasst und von dem Sechskantzylinder subtrahiert und somit ein Boolean erzeugt. Das Außengewinde wurde nun mittig auf der Unterseite des Sechskantzylinders gesetzt und beide Modelle miteinander verbunden. Das Ergebnis ist eine Gehäuseschraube. Das Gewinde ist zwar nicht spiralförmig, wird aber im Video nicht zu bemerken sein, daher konnte dies vernachlässigt werden. Abschließend musste der Schraube noch mit dem „Material-Editor“ das Material "Metall" zugewiesen und die Reflexionsgröße, Farbe sowie Spiegelungsrate eingestellt werden.

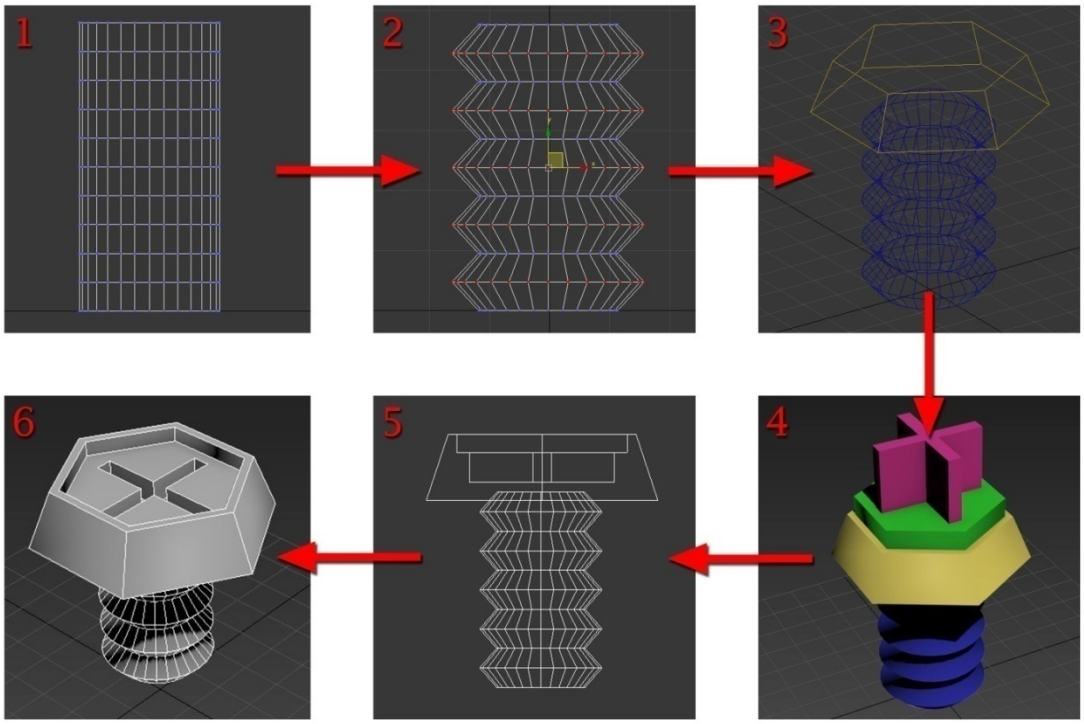


Abbildung 5.2: Ablauf der 3D-Modellierung einer Schraube

Um ein Gewinde z.B. in einem Gehäuse zu erstellen, wurde ein runder Zylinder mit einem etwas größeren Durchmesser als das des Außengewindes der Schraube erstellt. Dieser Zylinder wurde kopiert und an die entsprechende Stelle im Gehäuse platziert. An genau dieser Stelle wurde dann eine Kopie des Zylinders erstellt, da beim Subtraktionsprozess der erste Zylinder verloren geht. Der erste Zylinder wurde nun vom Gehäuse subtrahiert und der zweite passgenau in das vorhandene Loch geschoben. Nun musste eine Kopie des Außengewindes der Schraube in den Zylinder gesetzt werden und vom Zylinder subtrahiert. Auf diese Weise entstand ein passgenaues Gewinde für die Schraube.

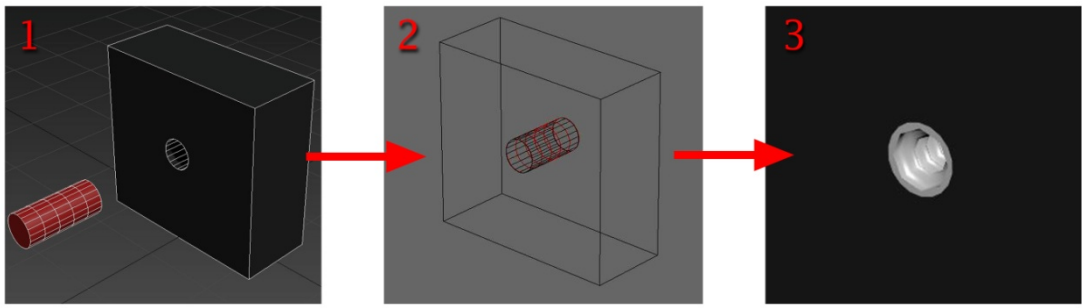


Abbildung 5.3: Ablauf der 3D-Modellierung eines Gewindes

5.1.2.2 Beispiel der 3D-Modellierung von eines geriffelten Objekts

Wie entsteht ein Objekt mit einer Riffelung? Um dies zu zeigen, wird das Beispiel einer Fingerschraube des Gehäuses aufgegriffen. Die Fingerschraube weist einen großen Kopf mit einer Riffelung auf, um diese auch per Hand auf- und zudrehen zu können. Wenn man sich den Kopf der Schaube genauer betrachtet, fällt auf, dass sie aus einem abgerundeten Zylinder besteht. Dieser Zylinder besitzt Einbuchtungen in Form eines Rechtecks. Erstellt wurde also ein "Charmfer Cylinder" und ein Rechteck, welcher sehr flach und in der Höhe ein wenig kleiner ist als die Außenkante des Kopfes. Das Rechteck wurde an die entsprechende Stelle im Kopf positioniert (nicht zu tief). Daraufhin wurde der Rotationspunkt genau in die Mitte des Kopfzylinders gelegt und im Abstand von 4° neunzig Mal kopiert. Die 4 Grad wurden deshalb gewählt, weil es an der Außenkannte ungefähr die Dicke des Rechtecks ergab. Dass es 90 Kopien kreisförmig um den Zylinder sein mussten, ergab sich aus der Rechnung: 360° / 4° = 90. Nun wurden alle Zylinder zu einem Objekt zusammengefügt und vom Kopfzylinder subtrahiert. Das resultierende Boolean bekommt analog zu der Gehäuseschraube noch eine Textur, eine Gewindestange sowie ein Schraubendreher-Kreuzloch.

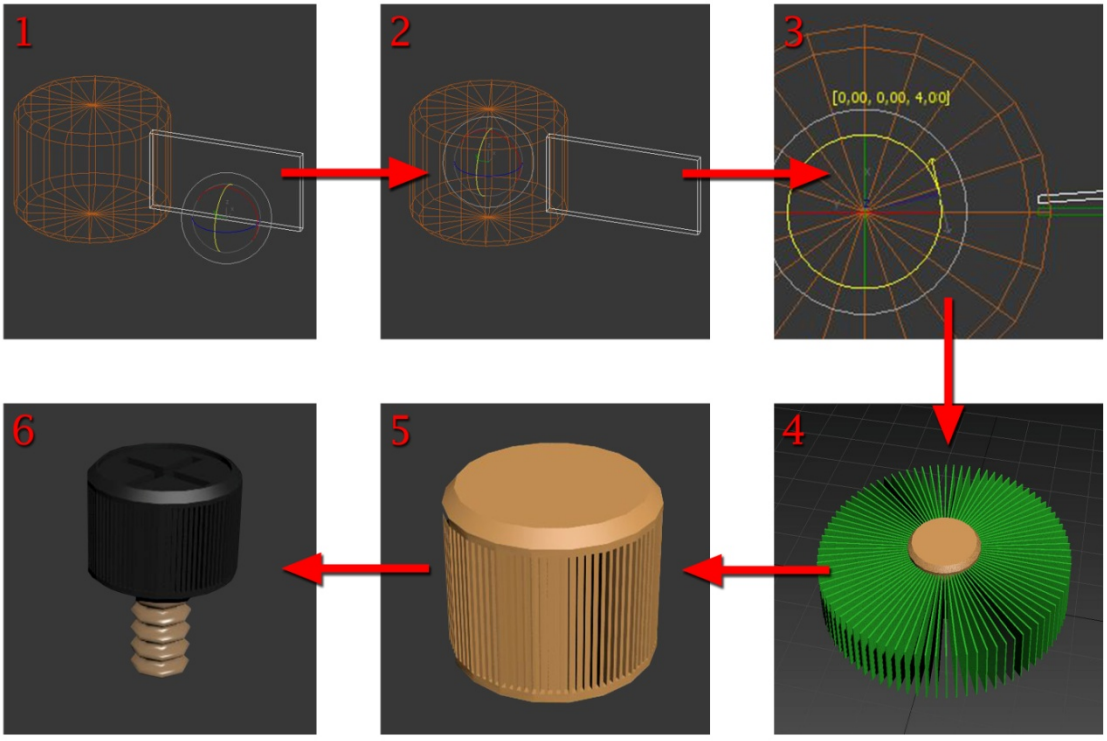


Abbildung 5.4: Ablauf der 3D-Modellierung einer Fingerschraube

5.1.2.3 Beispiel der 3D-Modellierung von Kabeln

Auf dem ersten Blick besteht ein Kabel aus einem reinen Zylinder, den man durch Polygonverschiebung in Form gebracht hat, jedoch wäre diese Vorgehensweise sehr mühselig und zeitintensiv. Daher wurden bei der Erstellung aller Kabel Splines benutzt. Splines sind 2D-

Linien, die sich im dreidimensionalen Raum befinden. Im Gegensatz zu den bisher benutzen Meshes besitzen sie kein Volumen und sind nur im Editor Menü sichtbar, jedoch nicht beim gerenderten Video/Bild. Um einen Spline "kabelartig" in einen 3D-Raum zu positionieren, wurde der Verlauf erst in einer 2D Ebene gezeichnet. Durch einen Modifikator wurden danach zusätzlich Polygonpunkte längs der Linie hinzugefügt, um auch Polygonpunkte für die anderen Ebenen zu erhalten. Nun wurde in eine andere 2D-Ebene um 90° in den 3D-Raum gewechselt, um auch dort die Polygone auf die richtige Bahn zu positionieren. Anschließend, wurde wieder in die 3D perspektivische Sicht gewechselt und kleinere Feinheiten am Verlauf korrigiert. Da das Spline durch die einzelnen Polygone noch sehr kantig aussah, wurde der Spline-Smooth Modifikator benutzt, der die Kanten abrundet. Um den 2D-Spline ein Volumen zu geben, wurde ein zweites Spline in Form eines Kreises mit dem Durchmesser des Kabels erstellt. (Beim SATA-Kabel wurde hier z.B. ein abgerundetes Rechtecke anstatt eines Kreises gewählt). Der Kreis-Spline wird nun via Modifikator orthogonal dem Verlauf des ersten Splines entlang gelegt und gefüllt. So ergab sich ein perfekt gefüllter Zylinder in Kabelform. Um die Polygonanzahl, die theoretisch nun bei unendlich liegt, zu reduzieren, kam ein Polygonreduzierungs-Operator zum Einsatz. Hier muss eine gute Mischung aus Polygonanzahl und Rundung gefunden werden.

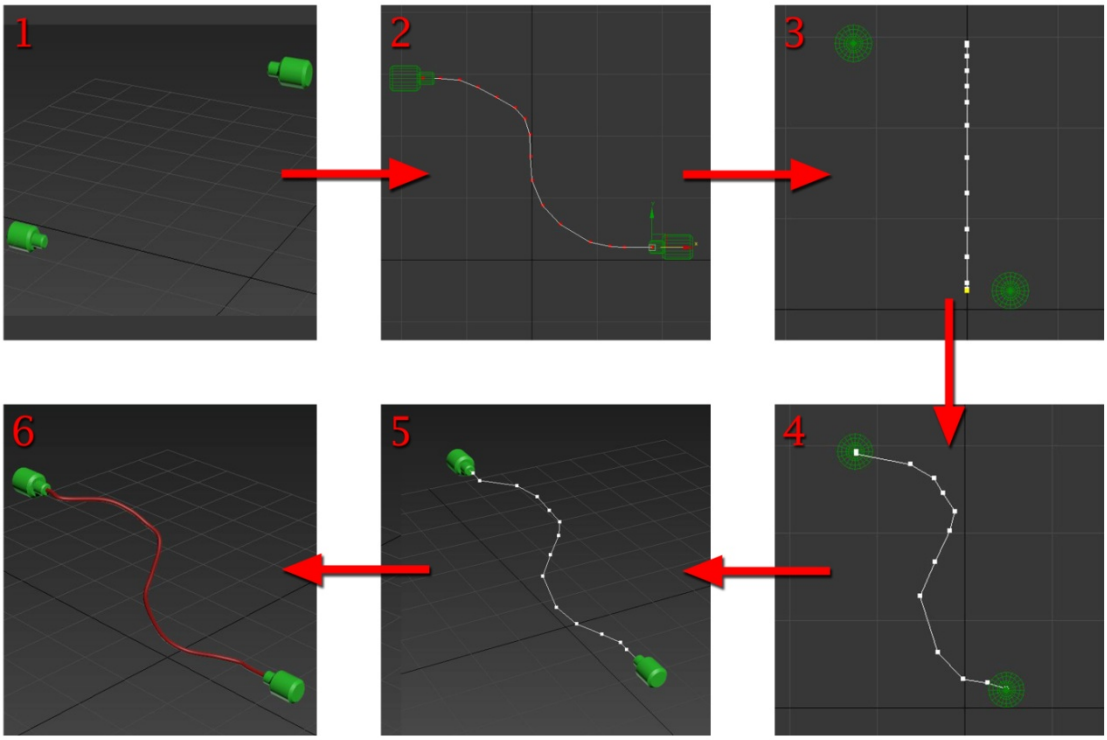


Abbildung 5.5: Ablauf der 3D-Modellierung eines Kabels

5.1.2.4 Erstellung von Texturen für die Videos mit Photoshop

Um eine Untergrundtextur zu erstellen, wurde wie in Kapitel 5.1.1 beschrieben, ein Foto z. B. von der Mainboard-Platine gemacht. Da das resultierende Bildergebnis allerdings ein wenig

verzerrt, nicht komplett scharf (Tiefenschärfe) und mit unerwünschten Lichteffekten versehen war, musste es nachträglich noch mit dem Programm *Photoshop* bearbeitet werden.

Zuerst wurden die Verzerrungen mit Hilfe der Funktion "Perspektivisch Transformieren" so verändert, dass die Textur gerade erscheint. Hierzu muss zuvor erst ein Verzerrungsgitter erstellt werden. Um das Gitter zu generieren, wurden Linien entlang der Kanten des Mainboards gelegt. Auf diese Weise erfährt man, wo die Fluchtpunkte des Bildes liegen. Nun wurde mit Hilfe der Funktion die Gitterlinien gerade gezogen und das Bild perspektivisch entzerrt. Der Rahmen des Bildes wies danach keine rechteckige Form mehr auf, weshalb das Mainboard erst nach diesem Schritt passgenau ausgeschnitten wird.

Beispielbild:

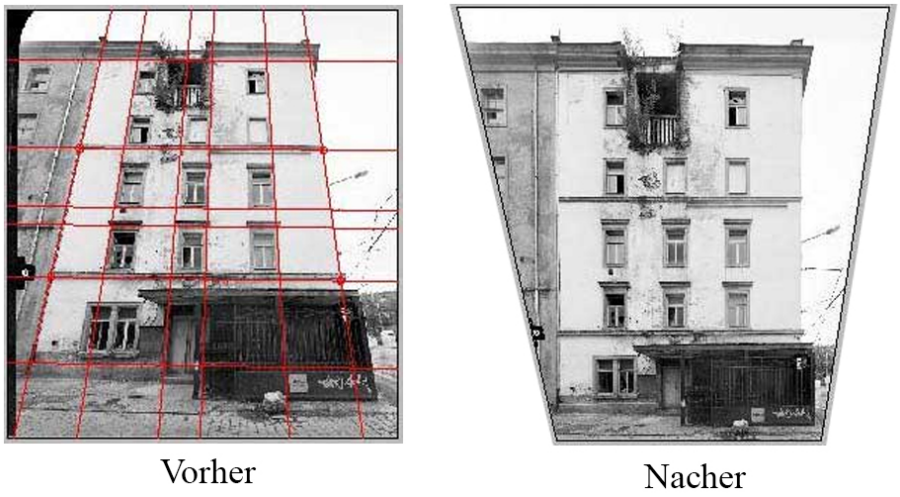


Abbildung 5.6: Beispiel einer perspektivischen Entzerrung [13]

Das ausgeschnittene Mainboard wurde als neues Bild in *Photoshop* eingefügt und der Kontrast, die Sättigung sowie die Helligkeit angepasst und als PNG[31]-Datei abgespeichert.

5.1.2.5 Mainboard

Das Mainboard wurde, wie alle anderen 3D-Modelle, komplett modelliert und texturiert. Es besitzt eine Grundtextur, die die Platine darstellt, und eine weitere für das *NVIDIA* Emblem. Um die Textur zu erstellen, wurde das Original-Mainboard fotografiert und mit *Photoshop* entsprechend bearbeitet (vgl. Kapitel 5.1.2.4).

[31] PNG (engl.: Portable Network Graphics)

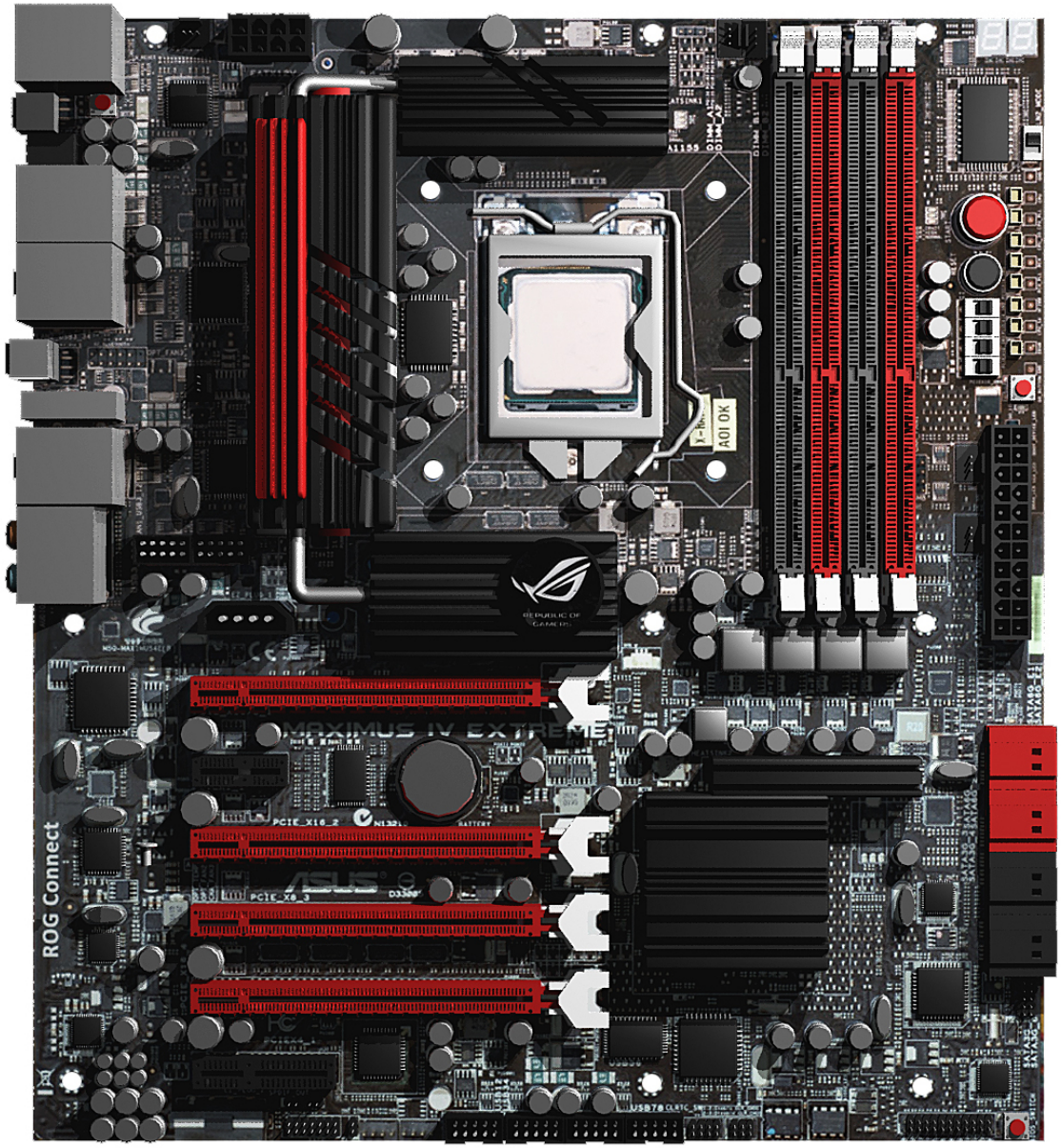


Abbildung 5.7: Fertiges 3D-Modell vom Mainboard

Name:	Asus Maximus IV Extreme
Anzahl Polygone:	821.541
Anzahl Vertikals:	416.529

In der Folgenden Tabelle, sind die Einzelbauteile vom Mainboard aufgelistet, die 3D-Modelliert worden sind.

Anzahl	Bauelement		Anzahl	Bauelement
5x	Schwarze Kühlrippe		2x	Graue RAM-Slot
1x	Rote Kühlrippe		1x	Prozessordeckel
2x	Kühlrohr		1x	Prozessorhebel
1x	Kühlrippe *NVIDIA* Symbol		12x	Eckiger Widerstand

6x	10ner Anschlusspinbuchse		8x	Großer Kondensator
1x	*NVIDIA* Emblem		60x	Mittlerer Kondensator
36x	Doppelanschlusspin		7x	Kleiner Kondensator
8x	Blaue USB-Buchse (USB 3.0)		11x	Ovaler Kondensator
2x	eSATA-Buchse		3x	Taster
1x	Schwarze USB Buchse (USB 2.0)		4x	Rote SATA-Buchse
15x	Mikrochip		4x	Schwarze SATA-Buchse
8x	Schalter		8x	3er Anschlusspin
4x	Roter PCI-Express-Slot		1x	Optical Audioport
1x	Kleiner PCI-Slot		1x	PS/2 Port
1x	Mittlerer PCI-Slot		2x	Netzwerkport
4x	Kipphebel für PCI-Express-Slot		6x	Klinkenaudioport
2x	Druckschalter		1x	Mainboard-Batterie

5.1.2.6 RAM

Auf dem RAM-Modul sitzt im Original PC ein RAM-Kühler. Deshalb wurde zuerst der RAM-Riegel modelliert und anschließend der Kühler, der sich im Video später auf den RAM-Riegel schiebt. Das RAM-Modul besteht hauptsächlich aus einer dünnen Platte mit einer aufgelegten Textur, die vorher fotografiert worden ist. Die Kühlrippen hingegen wurden hauptsächlich mit Booleans bearbeitet.

In den Animationen wurde auf die Darstellung der Kühlrippen (nicht jedoch in der RAM-Riegel-Anleitung) verzichtet, da das Bauteil vom User so besser auf dem Mainboard zu erkennen ist.

Bild des fertig modellierten RAM-Riegels mit Kühler:

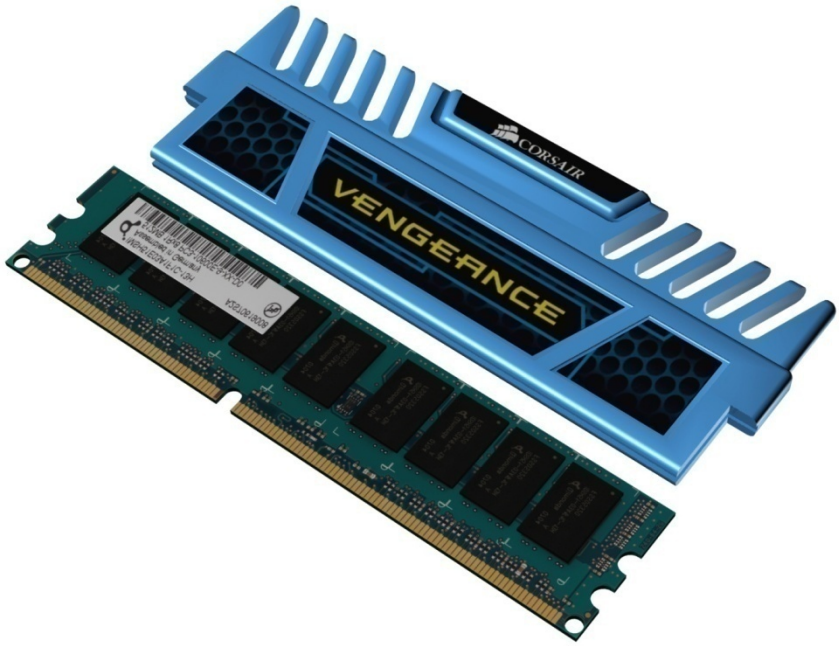


Abbildung 5.8: Fertiges 3D-Modell vom RAM-Modul

Name:	Vegeance 16GB Quad Channel DDR3 Memory Kit
Anzahl Polygone:	636
Anzahl Vertikals:	360

5.1.2.7 Grafikkarte NVIDIA

Die Grafikkarte besitzt ein schwarzes Plastikgehäuse um die Oberseite der Platinen. Somit ist von der Oberseite der Platine relativ wenig zu sehen. Die Unterseite hat keine Bauteile, lediglich sehr flache Lötstellen. Somit wurden nur die vorhandenen Schrauben modelliert und eine Texturplatte eingesetzt. Diese wurde entsprechend geformt und das Gehäuse via Boolean-Funktion angepasst.

Das Bild zeigt die fertig modellierte Grafikkarte:

Abbildung 5.9: Fertiges 3D-Modell der Grafikkarten-Oberseite

Abbildung 5.10: Fertiges 3D-Modell der Grafikkarten-Unterseite

Name:	GeForce GTX 680 2048 GDDR5
Anzahl Polygone:	66.078
Anzahl Vertikals:	34.978

5.1.2.8 Netzteil

Das Netzteil besteht aus einem normalen Quader, von dem andere Formen subtrahiert wurden. Im Deckel des Netzteiles befindet sich ein rundes Loch mit einem Gitter. Dahinter sitzt ein Lüfter und lässt in das Innenleben des Netzteils blicken. Für den Lüfter wurde wieder der bereits modellierte Gehäuse-Lüfter verwendet. Da das Innenleben kaum in den Videos zu sehen sein wird, wurde einfach eine Textur einer Netzteilplatine angefertigt und hinter den Lüfter gelegt.

An der Vorderseite des Netzteils befindet sich ein weiterer Lüfter sowie ein Schalter und die Stromanschlussbuchse. Dahinter sind die Stromkabel für die PC-Bauteile angebracht und als Knäuel mit einem Kabelbinder verbunden.

Abbildung 5.11: Fertiges 3D-Modell des Netzteils

Name:	be quiet Dark Power Pro 10 850 Watt
Anzahl Polygone:	517.766
Anzahl Vertikals:	267.629

5.1.2.9 CPU

Die CPU besteht aus den Grundbausteinen "Rechteck" und "abgerundetes Rechteck". Um eine realistische Abbildung zu erreichen, wurde eine *Intel*-Textur erstellt. Hierzu wurde mit dem Programm *Photoshop* das *Intel*-Logo nachgezeichnet und der verbaute Prozessortyp (i7) aufgebracht. Insbesondere musste drauf geachtet werden, dass der Markierungspfeil an der richtigen Stelle für den User gut sichtbar wird.

Jede CPU benötigt einen Kühler. Dieser besteht meistens aus Kühlrippen und einem Lüfter. Als Vorlage diente der bereits modellierte Gehäuse-Propeller, der entsprechend angepasst wurde. Für die Kühlrippen musste der Rotationspunkt, wie im Kapitel 5.1.2.2 beschrieben, verschoben und das gebogene Mesh kopiert werden.

Das folgende Bild zeigt die fertig modellierte CPU einschließlich Kühler:

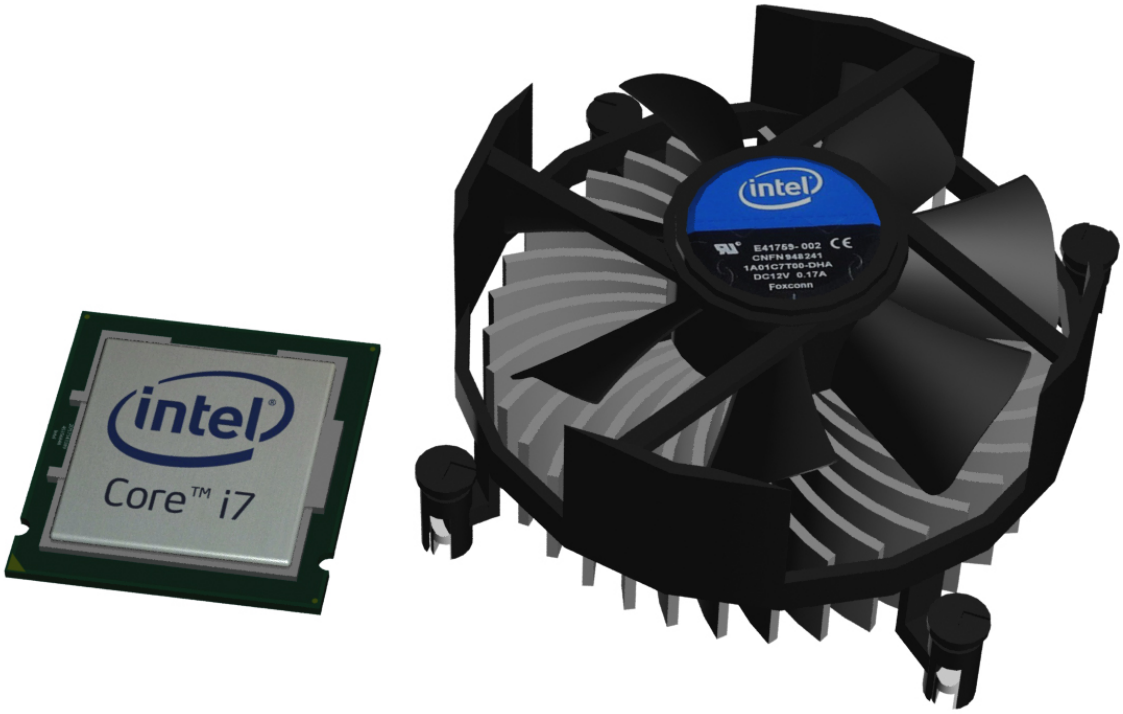


Abbildung 5.12: Fertiges 3D-Modell vom Prozessor und CPU-Kühler

Name:	IntelCore i7-3770K
Anzahl Polygone:	14.216
Anzahl Vertikals:	8.645

5.1.2.10 Festplatte (HDD)

Für die HDD wurden hauptsächlich abgerundete Quader sowie eine selbst erstellte Platte aus Polygonnetzen für die Metallplatte oben verwendet. Die Beschriftung wurde von einer vorhandenen Festplatte abfotografiert und als Textur auf ein Rechteck gelegt.

Bild der modellierten Festplatte (links Oberansicht, recht Unteransicht):

Abbildung 5.13: Fertiges 3D-Modell der Festplatte (links: Oberseite, rechts: Unterseite)

Name:	Keine Angabe
Anzahl Polygone:	4.852
Anzahl Vertikals:	2.472

5.1.2.11 DVD-Laufwerk

Das DVD-Laufwerk besteht hauptsächlich aus einem abgerundeten Quader, von dem andere Grundbausteine subtrahiert wurden.

Abbildung 5.14: Fertiges 3D-Modell des DVD-Laufwerks

Name:	Keine Angabe
Anzahl Polygone:	4.016
Anzahl Vertikals:	2.042

5.1.2.12 *Antec*-Gehäuse

Bei der Erstellung des Gehäuses musste besonders sorgfältig modelliert werden. Es musste z. B. von vorherein auf die exakte Einhaltung des Maßstabes geachtet werden, da es später, u. a. wegen der Bohrungen, nicht mehr durch leichtes Verzerren ausgleichbar ist. Zudem setzt sich das Gehäuse aus sehr vielen Booleans zusammen, die jeweils sehr oft subtrahiert wurden. Das Grundgerüst besteht aus sechs Rechtecken, die würfelförmig angeordnet sind. Die einzelnen Lüfter-Gitter wurden hergestellt, indem man in eine Gengon-Platte viele Löcher durch Subtraktion „einstanzt“. Diese Löcher wurden exakt wie im Original-Lüfter-Gitter angeordnet.

Die Lüfter waren die schwierigsten Einzelbauteile des Gehäuses. Sie wurden hauptsächlich Polygonschicht für Polygonschicht heraus modelliert (vgl. Kapitel 4.1.3).

Die Gewinde wurden erst nach Fertigstellung des Mainboards in die Rückwand-Abdeckung gestanzt. Dazu wurde das Mainboard an die richtige Position gebracht und anhand der Positionen der Bohrlöcher des Mainboards die vorgesehenen Gehäuselöcher durch Subtraktion mit Zylindern heraus gestanzt.

Die folgenden zwei Bilder zeigen das fertig modellierte Gehäuse:

Abbildung 5.15: Fertiges 3D-Modell des PC-Gehäuses (von der linken Seite)

Abbildung 5.16: Fertiges 3D-Modell des PC-Gehäuses (von der rechten Seite)

Name:	Antec P280
Anzahl Polygone:	87.973
Anzahl Vertikals:	59.595

5.1.3 Zusammenfügen der modellierten Einzelteile

5.1.3.1 Komplett-PC

Nachdem alle Einzelbauteile erstellt worden sind, mussten sie in ein einziges großes Projekt zusammengefügt werden. Da die Bauteile von vornherein mit den Originalmaßen angefertigt worden sind, verlief dieses ohne große Probleme und der PC ließ sich digital gut zusammenbauen.

Die einzelnen Objekte wurden sinnvoll gruppiert und anschließend benannt, um den späteren Animationsprozess einfacher zu gestalten. Erst jetzt war es möglich, Details wie die Verkabelung vom Netzteil zu modellieren, da der Abstand zwischen Netzteil und den Stromanschlüssen der

Einzelteile zu sehen war. Diese wurden dann mit der im Kapitel 5.1.2.3 gezeigten Technik erstellt.

Bild des modellierten Computers:

Abbildung 5.17: Komplettansicht des 3D-Modells des fertigen PCs

5.1.3.2 Raumgestaltung

Nun sollte noch ein virtueller Raum um den PC herum entstehen. Auch sollte der Computer auf einem Tisch stehen. Dazu wurde eine vordefinierte Holztextur von *3ds-Max* genommen und auf eine große Platte gelegt und als Boden ausgerichtet. Drum herum wurden Wände mit einer Tapetentextur erstellt und mit einer kleinen Bodenleiste versehen.

Als Tisch wurde der in der HAWK-Göttingen befindliche Schreibtisch aus dem PC-Pool modelliert. Da dieser aber ein wenig zu groß war, wurde er in der Breite auf die Hälfte reduziert.

Die Raummaße wurden so gewählt, dass die Kamera sich gut und frei darin bewegen konnte, ohne außerhalb der Wände zu gelangen. Des Weiteren wurde für einen schöneren Schattenverlauf gesorgt indem der Tisch in der Mitte dieses Raumes mit einem deutlichen Abstand zu den Wänden platziert worden ist.

Bild des Raumes mit PC:

Abbildung 5.18: Gesamtansicht der Szene

5.1.4 Lichteffekte und Schatten

Die Beleuchtung sowie Schattierung ist bei einem 3D-Projekt eines der drei großen Prozesse (Modellierung, Beleuchtung/Schatten, Render-Einstellungen). In diesem Bereich sind eigentlich keine Grenzen gesetzt. Man könnte immer noch eine etwas bessere Beleuchtung erstellen und an Details Schatten hinzufügen. Die Grenze setzt dabei allein die Bearbeitungszeit.

Um den modellierten Raum gut auszuleuchten und dadurch realistischer zu gestalten, wurde zuerst ein Omni-Licht[32] als Deckenlampe eingefügt. Des Weiteren wurde ein gerichtetes Licht (Lichtverlauf ähnelt einer Taschenlampe) direkt an die Kamera angefügt. Auf diesem Weg ist gewährleistet, dass immer dort, wohin die Kamera schaut, auch ein wenig Licht vorhanden ist. Da das Omni-Licht den Computer von vorne schräg oben beleuchtet, wurde ein weiteres sehr schwaches Omni-Licht in die Nähe der Rückseite des Computers erstellt. Das Gehäuse sowie der Raum waren recht gut beleuchtet, jedoch legte das Gehäuse einen Schatten über sämtliche Bauteile im Gehäuse, sodass sie selber keinen Schatten warfen. Auch waren diese dadurch sehr dunkel und schwer zu erkennen. Um dem entgegenzuwirken, wurden zwei weitere gerichtete Lampen und ein Omni-Licht im Gehäuse selber hinzugefügt. Diese wurden so ausgerichtet, dass das Innenleben gut beleuchtet und realistisch schattiert war.

5.1.4.1 Licht-Einstellungen

Nun, da die Positionen der einzelnen Lichter feststand, musste noch die jeweilige Intensität und Farbe deklariert werden, um ein optisch schönes Resultat zu erzielen. Die Deckenlampe wurde als stärkstes Licht definiert und sie sollte die Hauptlichtquelle sein. Das Omni-Licht auf der Rückseite des Gehäuses wurde sehr schwach eingestellt, da noch Lichtreflexionen der Tapetenwand hinzugefügt wurden. Diese wurden zusammen mit der Kameraleuchte solang getestet, bis ein angenehmes Resultat entstand. Des Weiteren wurde die Lichtfarbe ein wenig ins Gelbliche verstellt, um die Lichtreflexion des Bodens zu simulieren. Die Kameraleuchte sowie die 3 Leuchten im Gehäuse blieben in der Farbe Weiß.

In der folgenden Tabelle sind die Spezifikationen der einzelnen Lampen ersichtlich:

Name:	Intensität Multiplier:	Farbe:
Deckenleuchte	75	weiß
Kameraleuchte	40	weiß
Backleuchte	28	leicht gelblich
Innen-Omni-Licht	60	weiß
Innenstrahler 1	30	weiß
Innenstrahler 2	25	weiß

[32] Omni-Licht: Lichtquelle, die nach allen Seiten gleich leuchtet

5.1.4.2 Schatten-Einstellungen

Nachdem alle Lichter positioniert und ihre Werte angepasst wurden, konnten die Schattierungen deklariert werden. Bisher warfen alle Lampen gleichstarke Schatten. Dadurch entstanden pro Objekt sechs Schatten. Außerdem warfen die Lampen im Gehäuse auch Schatten an die Raumwände durch die Gehäuse-Öffnungen. Zuerst wurden bei allen Lampen, bis auf die Deckenleuchte, die Schatteneffekte ausgeschaltet. Allerdings waren die Schattierungen ein durchaus erwünschter Effekt bei den Bauteilen im Gehäuse. Deshalb wurden die Schatten der drei Leuchten im Gehäuse nur für die Bauteile eingeschaltet, für die Schatten generiert werden sollten (z.B. beim Strahler die Bauteile auf dem Mainboard). Nachdem alle Schatten richtig eingestellt und justiert waren, mussten noch die jeweiligen Intensitäten angepasst und definiert werden. Zudem musste entschieden werden, welcher Schattentyp verwendet werden sollte. Mit Schattentypen ist z.B. gemeint, ob es scharfkantige oder Verläufe sein sollen, zudem mit welchen Render-Einstellungen sie voreingestellt sind. Bei allen Schatten wurde der Typ "Ray Traced Shadow" gewählt.

Die folgende Tabelle zeigt die Spezifikationen der einzelnen Lampen:

Name:	Intensität Multiplier:	Schatten:
Deckenleuchte	40	An
Kameraleuchte	30	An
Backleuchte	----------	Aus
Innen-Omni-Licht	60	An
Innenstrahler 1	----------	Aus
Innenstrahler 2	----------	Aus

5.1.5 Animation der einzelnen Videos

Nach Fertigstellung der einzelnen Modelle und Gestaltung des Raumes wurden die Animationen und Kamerafahrten erstellt. Die zu zeigenden Bestandteile des Ein- und Ausbaus der Einzelanimationen wurden in Kapitel 4.1.2 beschrieben. Deshalb wird an dieser Stelle auf die technischen Aspekte, also auf die Reihenfolge, Kamerafahrten und Animationstechniken eingegangen.

Um Wiederholungen zu vermeiden wird übergreifend erklärt, wie die Glättung der Animationseckpunkte umgesetzt wurde.

Animationen erstellt man im Allgemeinen, indem man zwei Punkte fest definiert und eine Zeit zwischen diesen vorgibt. Das Animationsprogramm (in diesen Fall ebenfalls *3ds-Max 2014*) berechnet dann den direkten Weg zwischen diesen beiden Punkten, teilt den Weg in äquidistante Zeitbereiche ein und erstellt selbstständig Animationspunkte beim Rendern. Auf diesem Weg muss man nicht per Hand jedes einzelne Polygon bewegen. In dem Schaubild wird an solch einem Punkt die Rotation, Position und Verformung festgehalten. Es lassen sich jedoch

noch Objekt, Parameter, Attribute, Modifikationen, Materialien sowie eigens definierte Eigenschaften eines Objekts (z.B. das Gewicht) vorgeben.

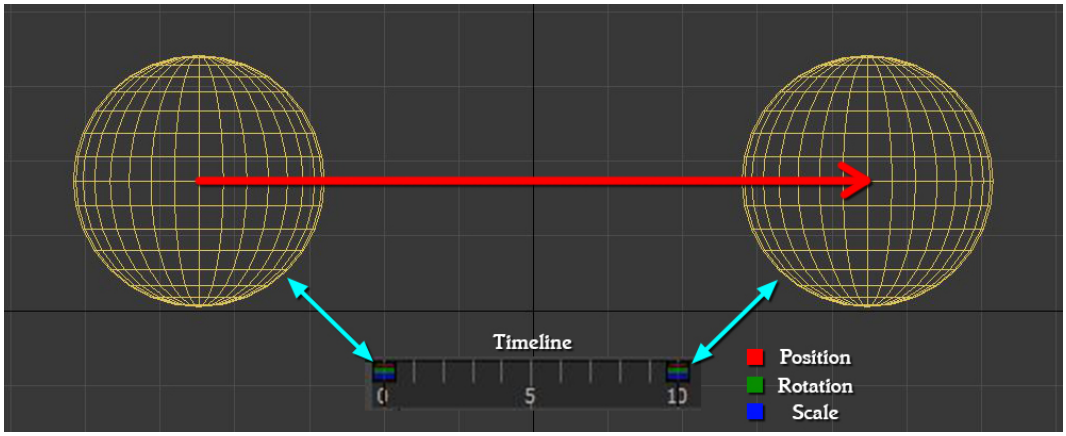


Abbildung 5.19: Beispiel für Animationspunkte

Jedoch würden die Bewegungen sehr abgehackt und stoßhaft/ruckelig aussehen, wenn nur zwei Animationspunkte fest definiert würden. Dies ist für das Auge unangenehm und wirkt auch sehr unnatürlich, weil das physikalische Gesetzt der Massenträgheit außer Kraft gesetzt ist. Um diesem Effekt entgegenzuwirken, wurden die Anfänge und Enden jeder Bewegung exponentiell gestaltet. Dies bedeutet, dass das Objekt sich nicht sprungartig mit einer gleichförmigen Geschwindigkeit bewegt, sondern fließend beschleunigt, die Geschwindigkeit beibehält und schließlich langsam wieder abbremst.

Im folgenden Bild sieht man den Animationsverlauf: links ohne Glättung, rechts mit Glättung. In beiden Fällen sei die Y-Achse die Position s[m] und die X-Achse die Zeit t[s]. Die Steigung des Graphen gibt die Geschwindigkeit v[m/s] an.

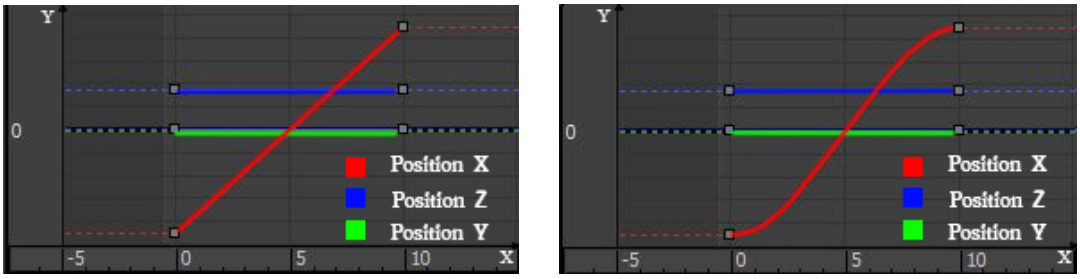


Abbildung 5.20: Vergleich zwischen einem harten (links im Bild) und einem weichen (rechts im Bild) Animationsverlauf

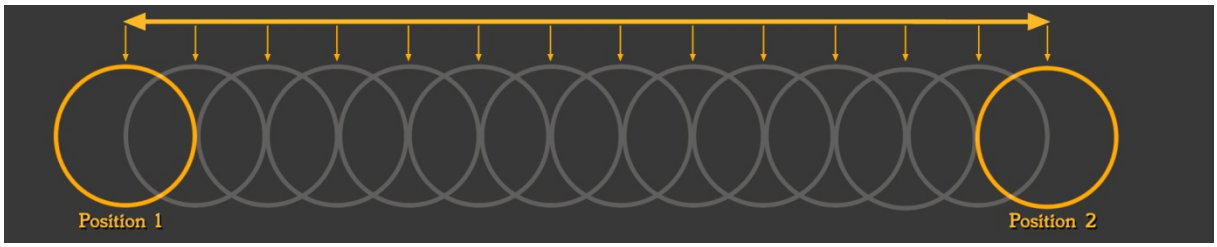


Abbildung 5.21: Bild von äquidistanten Abständen

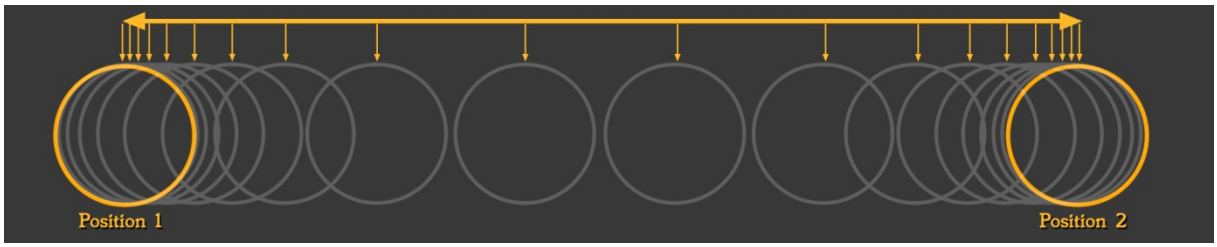


Abbildung 5.22: Bild von mit beschleunigten und gebremsten Abständen

Um die Kamerafahrt nicht allzu hektisch oder gradlinig zu gestalten, wurden Kurven und Rotationen eingebaut, die nicht zwingend notwendig waren, jedoch den Gesamteindruck des Videos deutlich verbesserten. Hierzu wurden zunächst ein Start- und Endpunkt festgelegt sowie die Kanten geglättet. Ein Zeitpunkt zwischen den beiden Eckpunkten wurde definiert und wiederum ein neuer Animationspunkt erstellt. Von diesem Punkt aus wurde das Objekt an eine andere Stelle bewegt (außerhalb des gradlinigen Verlaufs) und ggf. noch eine Rotation hinzugefügt.

Dies ist zudem eine gute Methode, um zu verhindern, dass Objekte sich während einer Animation überschneiden und ineinander geraten.

Die folgenden Grafiken verdeutlichen diesen Unterschied:

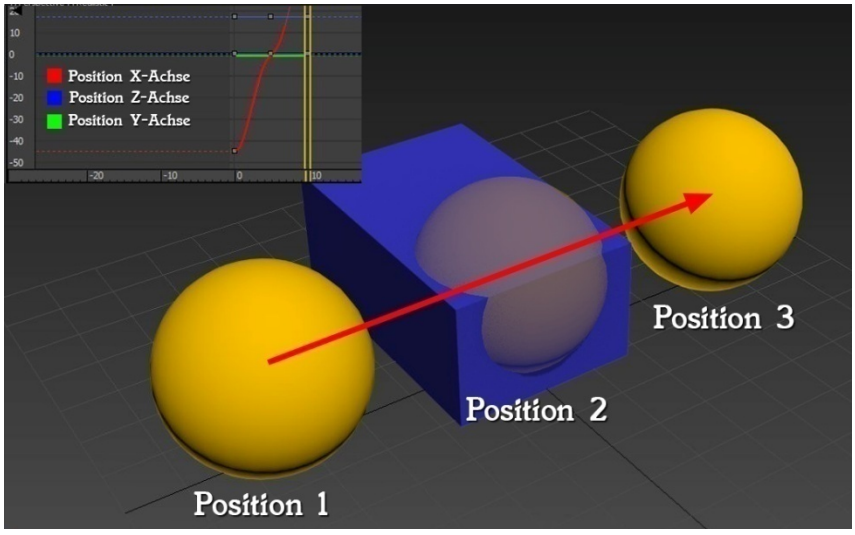


Abbildung 5.23: Animationsverlauf zwischen zwei Animationspunkten

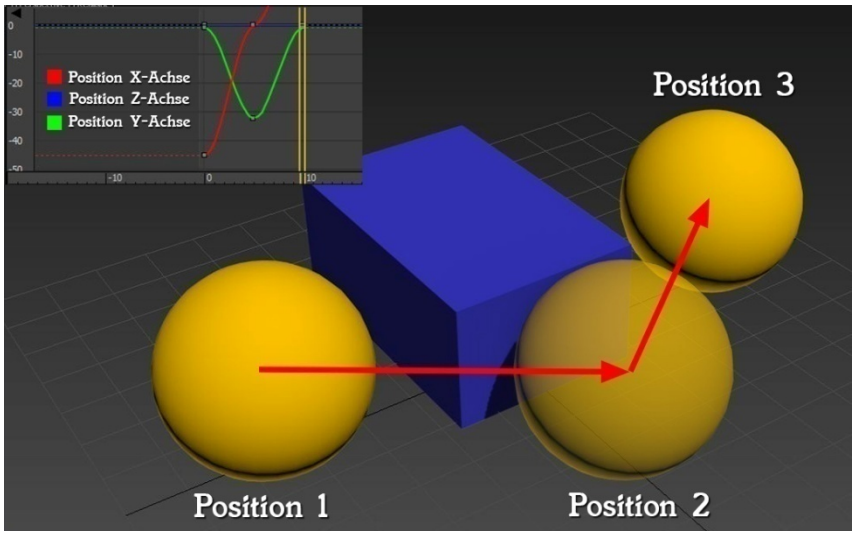


Abbildung 5.24: Animationsverlauf mit zwei Animationspunkten mit nachträglich erstelltem Zwischenpunkt

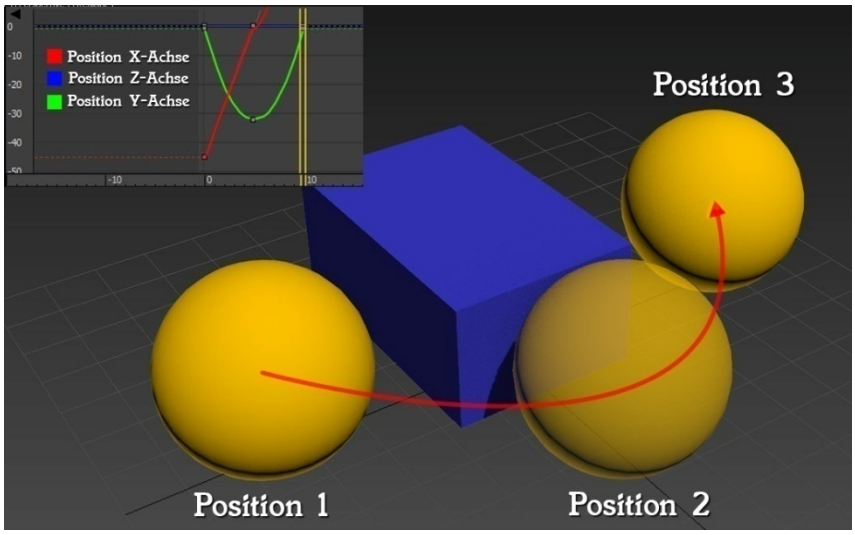


Abbildung 5.25: Animationsverlauf von den drei Animationspunkten mit Glättung der Animationskurven

Weiterhin wurde, kurz bevor ein Objekt in eine Steckverbindung eingeklinkt wurde, eine kurze Pause von jeweils 10 Frames eingelegt. Dadurch wirken die Bewegungen realistischer und lassen sich vom User besser verfolgen.

Die folgende Grafik zeigt beispielhaft die Positionskoordinaten der Kamera bei der Animation des Mainboard-Einbaus:

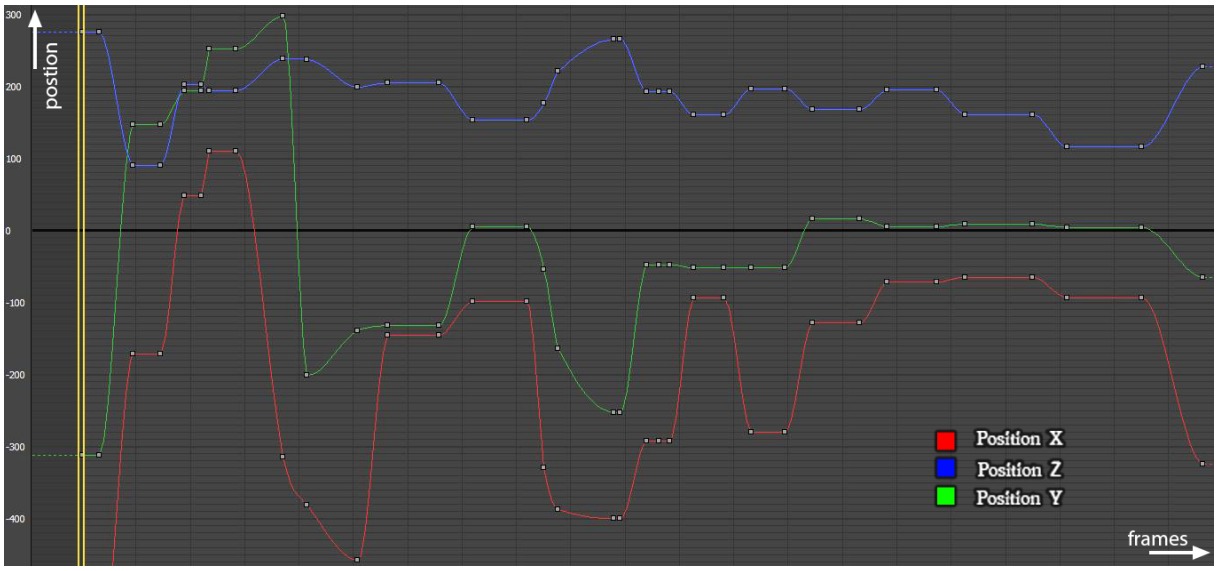


Abbildung 5.26: Positionsverlauf der Kamerafahrt beim Mainboard-Einbau

Im Folgenden sollen lediglich die Abläufe der einzelnen Animationen sowie die Positionen der Kamera erläutert werden.

5.1.5.1 RAM-Wechsel

Bei Umbau der Ram-Riegel ist darauf zu achten, dass sich die Austauschmodule von den verbauten optisch ein wenig unterscheiden. Um dies zu verdeutlichen, wurden die nicht notwendigen Kühlrippen von den einzubauenden Modulen entfernt. Der Schraubendreher sowie die Austauschmodule befinden sich zu Beginn der Videosequenz zusammen mit dem kompletten PC auf einem Tisch.

Die Kamera zeigt zunächst einen Gesamteindruck und fährt anschließend zur Rückwand des Computers. Der Schraubendreher wird an die Gehäuse-Schrauben herangeführt und dreht sie auf. Die Kamera bewegt sich in einem Bogen wieder in Richtung Ausgangsposition. Dabei erscheint eine Hand, die zeigt, wie die Seitenwand des Gehäuses zu öffnen ist. Die Seitenwand schwebt mit der Hand aus dem Sichtbereich der Kamera wieder heraus.

In der anschließenden Sequenz nähert sich die Kamera den RAM-Riegeln des Mainboards und zoomt vorerst nur an die RAM-Slot-Hebel heran. Um deutlich zu machen, dass diese vorher gekippt werden müssen, blinken diese vor der Bewegung rot auf. Dies wurde ermöglicht, weil im zeitlichen Verlauf Farbpunkte rot und weiß abwechselnd deklariert worden sind. Jetzt zoomt die Kamera wieder auf die gesamten RAM-Riegel und begleitet diese, bis sie auf dem Tisch abgelegt werden. Die Information, an welcher Position ein RAM-Riegel angefasst werden darf, wird links oben im Bild mit Hilfe eines Videobearbeitungsprogrammes eingeblendet.

Die neuen RAM-Module beginnen sich zu bewegen. Die Kamera erfasst sie und folgt ihnen wieder zum Mainboard, wo sie mit Druck in die Slots eingesteckt werden.

Abschließend fährt die Kamera wieder in die Ausgangsposition zurück und zeigt das Ergebnis des Umbaus in einer Vollansicht.

5.1.5.2 HDD-Einbau

Da der Ein- und Ausbau der Festplatte analog zueinander verlaufen, wurde nur der Einbau animiert. Am Anfang befindet sich die Festplatte neben dem PC auf dem Tisch.

Die Gehäusewand wird entfernt (vgl. Kapitel 5.1.5.1).

Die Kamera zeigt, wie das HDD-Schiebemodul aus dem Gehäuse entfernt wird. Anschließend bewegt sich die Festplatte auf das Schiebemodul zu und das gesamte System wird um 180° gedreht. Der Schraubendreher sowie vier Einzelschrauben „fliegen“ über die entsprechenden Schraubenlöcher und finden ihre jeweilige Position. Jetzt befestigt der Schraubendreher die Schrauben einzeln durch das Schiebemodul mit der Festplatte und fügt beide Komponenten fest zusammen. Der Schraubendreher entfernt sich wieder und das Gesamtsystem dreht sich abermals um 180°.

Nun fahren die Kamera sowie das System aus Schiebemodul und Festplatte zu dem vorgesehenen Steckplatz. Das Schiebemodul rastet ein und ist nun mit dem Gehäuse fixiert. Dann zoomt die Kamera an den Strom- und SATA-Anschluss der Festplatte, wo sich die Anschlusskabel mit Steckbewegungen in die definierten Anschlüsse schieben. Um zu zeigen, wo sich genau der SATA Anschluss des Mainboards befindet, bewegt sich die Kamera zum Mainboard, wo sich das andere Ende des SATA-Kabels einsteckt.

Abschließend fährt die Kamera wieder in die Ausgangsposition zurück und zeigt den Gesamteindruck der verbauten Festplatte.

5.1.5.3 Grafikkarten-Einbau

Aus Zeitgründen wurde nur eine Grafikkarte modelliert und nur der Einbau animiert. Der Ausbau verläuft komplett umgekehrt analog zum Einbau. Die Grafikarte sowie der Schraubendreher liegen wieder zu Beginn der Videosequenz neben dem PC auf dem Tisch.

Zuerst wird die Gehäuseabdeckung entfernt (vgl. Kapitel 5.1.5.1).

Die Kamera fährt zunächst auf zwei Erweiterungskartenblenden zu. Diese werden mit dem eingeblendeten animierten Schraubendreher aufgeschraubt und gelöst. Die Blenden gleiten auf den Tisch und werden dort abgelegt, während die Kamera die Grafikkarte abholt. Kamera und Grafikkarte bewegen sich nun gemeinsam zum PCI-Express Slot des Mainboards. Dabei dreht sich die Grafikkarte um 180° um die X-Achse, sodass die Platine nach oben zeigt. Der PCI-Express-Hebel klickt auf und die Grafikkarte gleitet in den Slot hinein. Anschließend rastet der Hebel wieder ein.

Die Schrauben der Blenden bewegen sich samt Schraubendreher zu den Schraublöchern und fixieren die Grafikkarte am Gehäuse. Die Kamera fährt zum anderen Ende der Grafikkarte und zeigt, wie sich die Stromkabel des Netzteils mit den entsprechenden Stromanschlüssen der Grafikkarte verbinden.

Die Kamera fährt in die Ausgangsposition zurück und zeigt ein Übersichtsbild der verbauten Grafikkarte.

5.1.5.4 DVD-Laufwerk-Wechsel

Das DVD-Laufwerk sowie ein Schraubendreher und vier Schrauben befinden sich beim Start der Animationssequenz in der Ausgangssituation neben dem PC auf dem Tisch.

In einem ersten Schritt wird das Gehäuse geöffnet. Hierzu wird zunächst die Gehäusewand des PCs entfernt (vgl. Kapitel 5.1.5.1).

Die Kamera fährt zusammen mit dem DVD-Laufwerk um den PC herum, während die Fronttür des Gehäuses geöffnet wird. Der Schraubendreher bewegt sich derweil in das Gehäuse und drückt die Laufwerksblende nach vorne heraus. Eine zweite Kamera positioniert sich ins Innere des Gehäuses und zeigt genauer, wie der Schraubendreher die Blende vorsichtig aufdrückt. Blende und Schraubendreher bewegen sich zum Tisch und werden dort abgelegt.

Das DVD-Laufwerk positioniert sich nun vor den Laufwerksschacht und bewegt sich in diesen langsam hinein. Die Kamera fährt wieder zur linken Seitenwand zurück und zoomt auf die Gewinde des DVD-Laufwerksschachts. Schraubendreher samt Schrauben positionieren sich vor die Gewinde und fixieren das DVD-Laufwerk am Gehäuse.

Anschließend bewegt sich die Kamera, zusammen mit dem Schraubendreher, wieder hinter das Gehäuse und demonstriert, wie die zweite Gehäusewand geöffnet wird (analog zur ersten). Nachdem auch diese entfernt wurde, bewegen sich Schraubendreher und Kamera zum Laufwerksschacht und verschrauben das DVD-Laufwerk von der anderen Seite.

Zum Abschluss der Sequenz fährt die Kamera noch einmal um den gesamten PC und zeigt in einer Totalen den Gesamteindruck des verbauten DVD-Laufwerks.

5.1.5.5 Netzteil-Einbau

Beim Einbau des Netzteils kommt es hauptsächlich auf die korrekte Verkabelung an. Da es technisch sehr aufwendig ist, ein Kabel realistisch zu animieren, wurde aus Zeitgründen ein Trick angewandt. Zunächst sind die Kabel mit einem Kabelbinder zusammengebunden. Später zeigt die Kamera - immer nur stark vergrößert - den entsprechenden Anschluss, mit dem sich der Stecker mit dem Kabel verbindet. So müssen nicht alle Kabel animiert werden, sondern sie können als starre Objekte zum Anschluss hingeführt werden. Das Netzteil, vier

Gehäuseschrauben sowie der Schraubendreher liegen zu Beginn der Videosequenz neben dem PC auf dem Tisch.

Da der Einbau innerhalb des PCs stattfindet, wird zunächst der Gehäusedeckel entfernt (vgl. Kapitel 5.1.5.1).

Vorab bewegt sich die Kamera zum Netzteil auf den Tisch und präsentiert das Einbauteil durch eine kleine Pause. Kamera samt Netzteil bewegen sich anschließend zum Gehäuse hin, wo das Netzteil an seine vorgesehene Stelle platziert wird. Die Kamera fährt danach zur Rückseite des Gehäuses und demonstriert, wie der Schraubendreher die vier Gehäuseschrauben mit dem Netzteil verschraubt. Das Netzteil ist nun fest mit dem Gehäuse verbunden. Abschließend zoomt die Kamera auf die einzelnen Stromanschlüsse der Bauteile des Computers.

Animiert werden die Kabel, wie oben beschrieben, mit einem Trick, um sich die sehr aufwendige realistisch Darstellung der Kabel zu sparen.

Die Reihenfolge der Aktionen wurde wie folgt festgelegt:

1. Hauptstromanschluss des Mainboards
2. CPU Stromanschluss am Mainboard
3. Zusatzstromsupport für die PCI-Express-Bänke
4. Stromanschluss des DVD-Laufwerks
5. Stromanschluss der Festplatte

Zum Abschluss der Sequenz bewegt sich die Kamera wieder in die Ausgangsposition zurück und vermittelt einen Gesamteindruck des verbauten Netzteils. Das Kabelknäuel des Netzteils ist, während es von der Kamera nicht erfasst wurde, weit im unteren Raum verschwunden. Die Einzelkabel kamen während der Animation von außerhalb des Raumes „hereingeflogen“. Hierbei wurde für die Animation besonders Sorgfalt auf Schatteneffekte und Tiefenwirkung gelegt, um den Benutzer ein klareres optisches Bild zu vermitteln und um ihn nicht zu verwirren.

5.1.5.6 Mainboard-Wechsel

Das Mainboard einzubauen, dürfte wohl der komplizierteste und schwierigste Teil aller Komponenten sein. Aus diesem Grund sind eine Vielzahl von Einzelaspekten für die Kamerafahrten zu zeigen und separat zu animieren. Die Aufgabenstellung ist es, ein Mainboard in einer Animation in das vorgegebene Gehäuse einzubauen. Es liegt daher, zusammen mit der Mainboardblende, zu Beginn der Animation auf dem Tisch neben dem Computer.

Bevor der Einbau visualisiert wird, muss zunächst wieder die Gehäusewand des PCs entfernt (vgl. Kapitel 5.1.5.1) werden.

Nachdem das Gehäuse geöffnet wurde, holt die Kamera die Mainboardblende vom Tisch ab. Gemeinsam bewegen sich Kamera und Blende zum Mainboardblendensteckplatz. Die alte Blende löst sich vom Gehäuse und wird auf dem Tisch abgelegt, während sich die neue an den

Steckplatz des PCs platziert. Anschließend zoomt die Kamera an ein Gewinde der inneren Rückenwand des Gehäuses heran, wo ein Abstandshalter eingeschraubt wird. Um redundante Informationen zu vermeiden und Zeit zu sparen, platzieren sich die restlichen 8 Abstandshalter, während sich die Kamera gleichzeitig langsam zum Mainboard bewegt. Später wird an dieser Stelle mit dem Videobearbeitungsprogramm ein zusätzliches Informationsvideo eingefügt und es werden Richtungspfeile gesetzt (vgl. hierzu Kapitel 5.1.8.2).

Das Mainboard wird durch die Kamera anvisiert und zum Gehäuse geführt. Es wird durch Kombination aus Rotation und Position demonstriert, dass das Mainboard exakt in das Gehäuse eingepasst und einzuklinken ist. Um dies zu verdeutlichen, erscheint eine zweite Kamera, die den Vorgang von hinten (außerhalb des Gehäuses) gleichzeitig dokumentiert.

Die Kamera zoomt auf ein Gewinde und einen Schraublock des Mainboards heran und filmt, wie der Schraubendreher das Mainboard durch eine Schraube am Gehäuse fixiert. Analog zum Abstandshalter wird dies nur an einer Schraube als Beispiel demonstriert. Die restlichen Befestigungsschrauben werden allesamt gleichzeitig an ihre Zielposition bewegt. Wie beim Abstandshalter wird später an dieser Stelle ein Zusatzvideo eingeblendet, das mit entsprechenden Pfeilen den Vorgang veranschaulicht.

Nun werden alle Stromanschlüsse mit dem Animationstrick (vgl. Kapitel 5.1.2.3) an das Mainboard angeschlossen. Die Start-, Reset- sowie HDD-Kabel werden im Anschluss am vorgesehenen Platz befestigt.

Zum Abschluss fährt die Kamera an ihre Ursprungsposition zurück und zeigt in einem Übersichtsbild das verbaute Mainboard.

5.1.5.7 Prozessor-Wechsel

Der Prozessor wird im Allgemeinen bereits verbaut, bevor sich das Mainboard im Gehäuse befindet. Aus diesem Grund liegen Mainboard, CPU, CPU-Kühler und Schraubendreher am Anfang der Animation neben dem Gehäuse. Das Gehäuse ist zu Beginn geöffnet und verweist somit noch deutlicher darauf, dass das Mainboard bereits ausgebaut wurde.

Die Kamera zeigt zunächst den kompletten Tisch mit allen zum Einsatz kommenden Elementen. Sie fährt auf das Mainboard zu und filmt, wie sich das Verdeck des CPU-Slots öffnet. Anschließend bewegt sie sich zum CPU und bringt diesen an die richtige Position über den entsprechenden Slot.

Der Prozessor positioniert sich zunächst mit Absicht falsch über den CPU-Slot. Dies wurde deswegen vorausgeschickt, damit der User durch die anschließende korrekte Positionierung versteht, dass der CPU nur mit einer bestimmten Ausrichtung in den Slot gesteckt werden darf. Deshalb befindet sich auf jedem Prozessor ein gelber Pfeil, der im Video ebenfalls zu sehen ist. Dieser Pfeil markiert die richtige Rotation und Ausrichtung. Nachdem der Prozessor sich im

Video selbstständig korrekt ausgerichtet hat, wird er in den Slot hineingesetzt, worauf sich die CPU-Slot-Klappe schließt.

Um ein wenig mehr Dynamik in das Video zu bekommen, bewegt sich die Kamera an eine andere Position und filmt nun aus dieser neuen Perspektive, wie sich der CPU-Kühler über das Mainboard positioniert. Er wird auf den Sockel am Mainboard gesteckt. Anschließend kommt der Schraubendreher zum Einsatz und dreht einzeln die Fixierbeine des Kühlers mit einer 45° Drehung in die Gewindebohrungen hinein.

Die Kamera zoomt an den Stromanschluss für den CPU-Kühler auf dem Mainboard und filmt, wo und wie sich das Kabel selbstständig mit dem Anschluss verbindet.

Abschließend fährt die Kamera wieder an die Ausgangsposition der Videosequenz zurück und zeigt das Ergebnis des verbauten Prozessors.

5.1.5.8 Komplettbau

Um den Usern verschiedene Winkel und Eindrücke des Einbaus der einzelnen Bauteile zu vermitteln, sollte der Komplettbau nicht aus Teilvideos der bereits erstellten Animationen bestehen. Es ist für die User an dieser Stelle wichtig, die Reihenfolge des Austausches kennenzulernen und dass sich bestimmte Arbeitsschritte wiederholen. Daher wurden Schraubbewegungen sowie Steckverbindungen vernachlässigt, da diese in den Einzelvideos eingehend erläutert werden. Primär wird also in diesem Video auf die Reihenfolge und Platzierung der Einzelkomponenten Wert gelegt. Die Reihenfolge stellt sich dabei wie folgt dar:

1. CPU auf Mainboard befestigen
2. Lüfter mit Kühlrippen auf den CPU verschrauben
3. Ram-Riegel auf das Mainboard stecken
4. Mainboard-Blende am hinteren Teil des Gehäuses austauschen
5. Abstandshalter einschrauben
6. Mainboard in das Gehäuse einstecken und verschrauben
7. Netzteil in das Gehäuse verbauen
8. Grafikkarte in das Mainboard einstecken
9. DVD-Laufwerk und HDD (Festplatte) einbauen
10. HDD und DVD-Laufwerk an das Mainboard anschließen
11. Gehäuse-Kabel an das Mainboard anschließen

12. Netzteil-Verkabelung anbringen

Diese Reihenfolge ist wichtig und sollte beim Kompletteinbau der Einzelkomponenten unbedingt eingehalten werden. Andernfalls kann es Komplikationen geben, die einen erneuten Einbau nötig machen [12].

Die Kamera fährt grob zu den einzelnen Bauteilen und filmt, wie sich diese an Ihre Position begeben. Die Verkabelung wird nicht detailliert animiert, auch wird wie in den speziellen Animationen auf das Heranzoomen verzichtet. Die Komponenten werden einfach von einem Frame zum nächsten an die korrekte Position gebracht. Hierzu wurde eine scharfkantige Animationen verwendet und ohne ein Verlauf zwischen Punkt (A) und Punkt (B). Sie werden später mit dem Videobearbeitungsprogramm langsam eingeblendet.

Zum Schluss fährt die Kamera wieder in die Ausgangsposition zurück und vermittelt den Gesamteindruck des zusammengebauten Computers.

5.1.6 Rendern und Render-Einstellungen

Um die Render-Einstellungen richtig deuten und verstehen zu können, muss erst einmal verstanden werden, was der Renderprozess überhaupt macht. Mit den heutigen leistungsfähigen Computern ist es möglich, bereits bei der Modellierung eine recht gute 3D-Ansicht des späteren Modells zu erhalten. In den Anfangszeiten der 3D-Modellierung wurde hauptsächlich mit 3D-Gittern oder mit in der Qualität deutlich reduzierten Modellen gearbeitet, die erst beim Rendern vollständig berechnet wurden. Selbst heute gibt es noch gravierende Unterschiede zwischen dem Erstellungsprozess und dem nachher gerenderten Bild.

Eigentlich berechnet der Renderer nur jedes Einzelbild (Frame) mit den Parametern, die man vorher eingestellt hat. So kann man z.B. eine Textur beim Bearbeitungsprozess entweder ausblenden oder mit einer starken Komprimierung sehen. Beim späteren Rendern jedoch wird die volle Qualität beachtet.

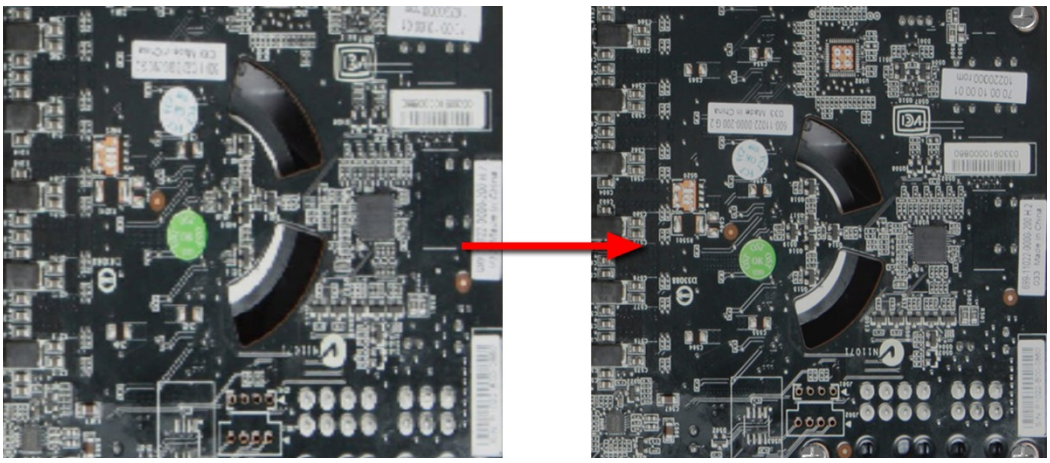


Abbildung 5.27: Qualitätsänderung beim Rendern von Texturen

Das Gleiche gilt auch für Rundungen, Kantenglättung, Spiegelungen, Reflexionen, Schattierung, Antialising usw. Der Computer, an dem gearbeitet worden ist, benötigte ca. 20 bis 90 Sekunden, um ein Bild zu rendern. Die hängt stark davon ab, wie viel auf dem Bild zu sehen ist und ob z. B. Spiegelungen berechnet werden müssen.

Um dies zu bewerkstelligen, bietet *3ds-Max* verschiedene Rendertypen. Man kann sich auch weitere aus dem Internet besorgen. Für das Projekt wurde der MetalRay Renderer benutzt, da er, im Gegensatz zu den anderen, eine deutliche kürzere Renderzeit benötigte und die Schattierung besser zur Geltung kamen.

Render-Einstellungen:

Renderer	Mental.ray.no.gi
Auflösung	1920x1080
Speicherart	Frames als Einzelbilder im Bildformat TGA[33] Speichern

Alle weiteren Einstellungen wurden auf Standard belassen. Es bestände auch die Möglichkeit, einen eigenen Renderer zu programmieren, welches aber viel zu aufwendig wäre.

Da der Videokomprimierer von *3ds-Max* - selbst bei angeblich verlustfreier Komprimierung - starke Qualitätsverluste beim Video erzeugte, fiel die Wahl auf unkomprimierte Einzelbilder, die nachträglich mit einem anderen Programm zu einem Video zusammengefügt wurden.

Hierzu wurde vorerst das Programm *VirtualDub* verwendet, das aus den Einzelbildern ein unkomprimiertes Video herausgab. Danach wurde es mit dem Programm *XMedia Recode* komprimiert. Das Ergebnis war einigermaßen zufriedenstellend, bis auf den Lichtverlauf an der Gehäusewand.

Für die Testvideos war dies zunächst ausreichend. Jedoch wurden später bei der Videobearbeitung alle Schritte direkt vereint und die Programme *Xmedia Recode* sowie *VirtualDub* nicht mehr benutzt.

5.1.7 Userinterface-Design

Das Userinterface-Design war eine der Hauptpunkte der Praxisphase. Im Kapitel 4.2.2 wurde beschrieben, wie das Layout und die einzelnen Positionen aussehen sollten. In diesem Abschnitt wird dargestellt, wie das Farb- und Formdesign der einzelnen Elemente gewählt und erstellt worden ist. Um ein vielseitiges und gutes Resultat beim Design zu bekommen, wurde vorab entschieden, dass von beiden Studenten ein eigenständiges Design entworfen wird und am Schluss die Vorteile beider Entwürfe zusammengefügt werden. Im Folgenden soll der Entwurfsprozess von J. Mewes erläutert werden. Der Entwurfsverlauf von P. Lindegrün ist im Kapitel 5.2.4 aufgeführt.

[33] TGA (engl.: Truevision Advanced Raster Graphics Array)

Für die Entwürfe für das Userinterface-Design wurden verschiedene *Photoshop* Techniken angewandt. Zur Verdeutlichung soll exemplarisch aufgezeigt werden, wie das Schattenmuster des ersten Interfaceentwurfs erstellt worden ist.

Das erste Design sah vor, alle Buttons mit einem Schattierungsrahmen abzugrenzen. Hierzu wurde ein Pinsel mit einer weichen Verlaufsform gewählt. Das bedeutet, dass der Mittelpunkt schwarz ist und kreisförmig nach außen hin immer durchsichtiger wird. Mit diesem Pinsel wurde auf einem transparenten Hintergrund ein Rechteck in Form des Buttons gezeichnet. Um den Button so wirken zu lassen, als würde er gerade über dem Hintergrund schweben, wurde ein Auswahlrechteck in die Mitte des Pinselstriches gezogen und anschließend die ausgewählte Fläche gelöscht. Daraufhin wurde der noch zu kräftige Rahmen etwas transparenter gestaltet. Anschließend wurde der Text, der auf dem Button stehen soll, in den Kasten hinein geschrieben. Der fertige Button wurde mit transparentem Hintergrund als PNG abgespeichert.

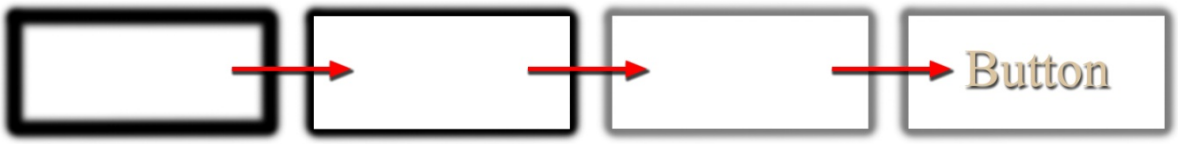


Abbildung 5.28: Ablauf der Erstellung eines Buttons mit Schattenmuster

Der erste Entwurf sah ein sehr dunkles Design mit einem grauen Farbverlauf vor, um die Augen nicht zu blenden und das Video deutlich vom dunklen Hintergrund abzutrennen. Für den Hintergrund wurde eine dunkelgraue Fläche mit einem sehr leichten Verlauf erstellt. An den unteren Ecken wurden Aussparungen mit einem Schattenmuster für die Buttons erzeugt, die durch ein leicht helleres Grau hervorgehoben wurden. Um das Gesamtbild ein wenig ruhiger für das Auge erscheinen zu lassen, wurde dem Gesamtbild ein dünnes schwarzes Gitternetz hinzugefügt.

Die Grafik zeigt die erste Hintergrundtextur:

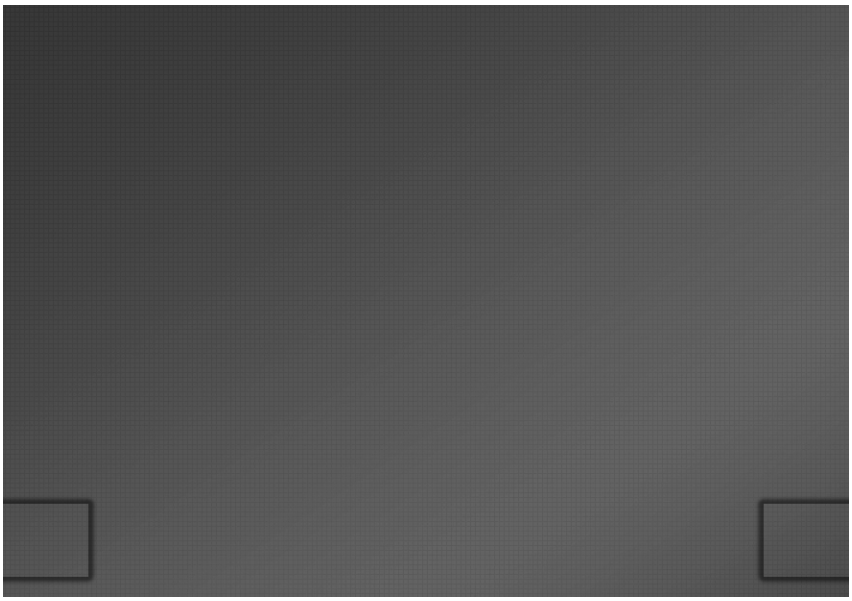


Abbildung 5.29: Erste Entwurf der Hintergrundtextur

Die Buttons sollten in einem einheitlichen Stil zusammen mit dem Hintergrund erscheinen. Daher wurden sie auch mit einem Schattenmuster erstellt. Es wurden die allgemein bekannten Formen als Grundgerüst gewählt, um den Wiedererkennungswert zu steigern. Diese wurden mit der Schattentechnik nachgezeichnet.

Das Ergebnis zeigt das folgende Bild:

Abbildung 5.30: Erste Entwurf der Steuerelemente

Dieser erste Entwurf wurde einigen Personen gezeigt sowie in der Vorlesung Userinterface-Design präsentiert, um Kritik und Feedback zu bekommen. Die Kritikpunkte waren:

- zu starker Verlauf
- nicht genug Abgrenzung von Steuerelementen
- wirkt zu eintönig
- zu viele Rahmen um die Steuerelemente
- ähnelt zu sehr einem Fernseher

Im zweiten Entwurf wurde an den Kritikpunkten im Design gearbeitet. Am Rand wurden ovale Linien mit einem Schattenmuster hinzugefügt und hinter den ovalen Linien und den Aussparungen wurde die graue Farbe ein wenig aufgehellt. Des Weiteren wurden die Aussparungen etwas in der Form verändert, um das ovale Muster zu vervollständigen und ein wenig mehr Dynamik hineinzubekommen. Die Buttons wurden lediglich mit einem dunkleren Hintergrund versehen, um sie mehr vom Hintergrund abzugrenzen.

Zweiter Entwurf des Hintergrundbildes mitsamt der Videoplayeraussparung und den Steuerelementen:

Abbildung 5.31: Zweiter Entwurf des Videoplayerfensters

Für das erste Fenster mussten nun die Auswahlbilder erstellt werden. Um ein wenig Farbe und Helligkeit in das Design zu bekommen, wurde sich für gerenderte Farbbilder von den 3D-Modellen entschieden. Dadurch konnten von vornherein die 3D-Modelle begutachtet werden und der User konnte sein Bauteil, welches er vor sich liegen hatte, zuordnen. Als Hintergrund für die 3D-Modelle wurde die Holztextur des Raumbodens benutzt, da die kräftige Holzfarbe einen guten Kontrast zum dunklen Grau darstellte. Als Beschriftung wurde mit einem dickeren Font die Farbe Beige ausgesucht und mit einem schwarzen Schatten versehen.

Die Grafik zeigt das fertige Startseitenbild (1. Fenster)

Abbildung 5.32: Zweiter Entwurf des Auswahlmenüs

Im 2. Fenster wurden ebenfalls die gerenderten Bilder und der Font verwendet.

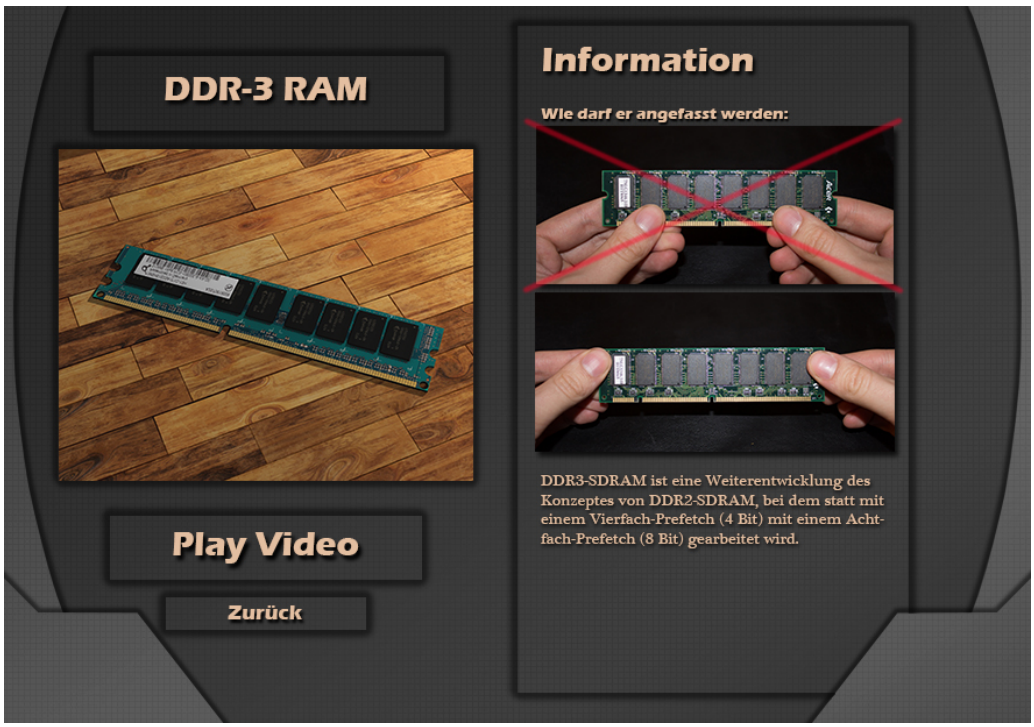


Abbildung 5.33: Zweiter Entwurf des Infofensters

Nachdem das Design ein weiteres Mal dem Userinterface-Design Kurs präsentiert wurde und es weitgehend als in sich stimmig beurteilt wurde, war das Design damit komplett. Im Kapitel 5.3

wird das endgültige Design aus der Mischung dieses und des von P. Lindegrün erstellten erläutert.

5.1.8 Video Bearbeitung

Wenn die gerenderten Einzelbilder zu einem Video zusammengefasst sind, enthalten sie zwar ausreichend Informationen und Hinweise für einen Hardware-Umbau, jedoch wäre es hilfreich, auf einzelne Passagen deutlicher hinzuweisen. Ebenso sollte der Name des Erstellers sowie ein paar Informationen zum Ein-/Ausbau eingeblendet werden.

Um dies zu bewerkstelligen, wurde, wie im Kapitel 3.1.3 beschrieben, das Programm *Adobe Premiere Pro CS6* ausgewählt.

5.1.8.1 Videobearbeitung mit Adobe Premiere Pro CS6

In das Programm *Adobe Premiere Pro CS6* wurden zuerst die Einzelbilder im Bildformat TGA eingelesen. Danach wurde eine neue Sequenz erstellt, wo die Teilbilder in der richtigen Reihenfolge hintereinander angeordnet wurden. Als nächstes musste noch die Länge der einzelnen Frames angepasst werden, sodass das Video eine, für den allgemeinen Betrachter, angenehme Geschwindigkeit hat und zeitlich nicht zu lang wird. Nun wurde die erste Passage herausgesucht, wo Zusatzinformationen in das Video eingeblendet werden sollte. Beispielhaft soll dies am Mainboard-Video erläutert werden. Hier wird z.B. beim Teilbild 1418 ein Standbild eingeblendet, wo neun Pfeile auf die neun Abstandshalterschrauben im Gehäuse zeigen. Kurz darauf soll ein kleines Video oben rechts im Bild eingeblendet werden, das den Einschraub-Prozess der Abstandhalter zeigt. Dadurch werden redundante Informationen im Video vermieden und die Anweisungen werden für den User verständlicher dargestellt.

Zuerst muss das Teilbild 1418 in das Bildbearbeitungsprogramm *Photoshop* geladen und die Pfeile eingefügt werden. Dieser Prozess wird separat im nächsten Kapitelpunkt 5.1.8.2 beschrieben.

Nachdem das Bild erstellt und in *Abobe Premiere Pro* geladen worden ist, musste eine Pause zwischen dem Teilbild 1418 und 1419 in der Timeline erzeugt werden. Dies wurde bewerkstelligt, indem alle Teilbilder ab 1418 etwas weiter nach rechts verschoben worden sind, wodurch eine Lücke entstand. Danach wurde eine zweite Videoebene über der ersten erstellt und das aus *Photoshop* importierte bearbeitete Teilbild 1418 (ab jetzt 1418b genannt) mit den Pfeilen in die zweite Videoebene geschoben. Sie wurde genau einen Frame hinter das Originalteilbild 1418 geschoben. Die Länge des Bildes 1418b wurde anschließend so gewählt, dass es genau zwischen die Lücke passt. Damit die Pfeile im späteren Video nicht einfach nur von einer Millisekunde zur nächsten auftauchen und vermutlich den Betrachter verwirren, sollten sie langsam eingeblendet werden. Um dies zu erreichen, wurden, ähnlich wie bei den

Animationspunkten der 3D-Animation, vier Punkte in der Zeitebene gesetzt, die die Transparenz regeln.

Abbildung der Zeitleiste mit der Transparenz:

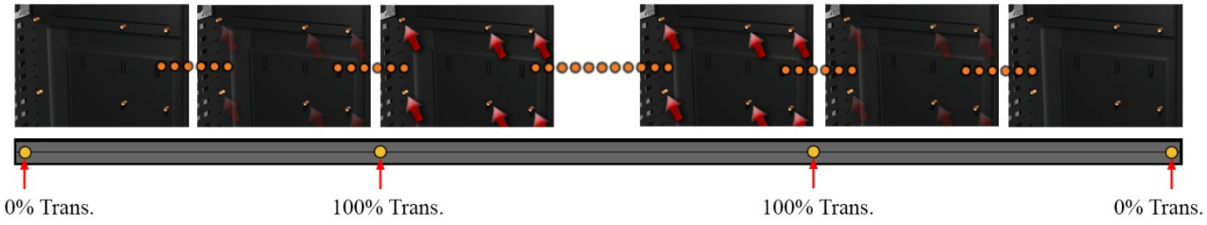


Abbildung 5.34: Ein- und Ausblendung der Pfeile

Das Programm errechnet analog zum 3D-Animationsprogramm den Verlauf selbst.

Als nächstes müssen noch zwei weitere Videoebenen erstellt werden. In der Ebene über dem langen Teilbild 1418b wurde eine mit Photoshop erstellte Hintergrundtextur geladen. Sie hat einen roten Verlauf und einen dünnen schwarzen Rahmen. Dies soll eine Art Rahmen für das eingeblendete Video sein, damit es sich ein wenig vom hinteren Video abhebt.

Um das Teilvideo in Größe, Geschwindigkeit, Helligkeit, Kontrast sowie Transparenz einstellen zu können, musste es vorher als einzelne Sequenz definiert werden. Deshalb wurde eine neue Sequenz erstellt, in die die Teilbilder des Einschraub-Prozesses eingefügt wurden.

In die oberste Ebene der ursprünglichen Sequenz wurde nun das Teilvideo eingebunden. Um die Attribute Größe und Transparenz einstellen zu können, wurden die entsprechenden Modifikatoren hinzugefügt. Nun wurden das Video sowie der Rahmen in Größe und Länge angepasst und an die richtige Position (oben rechts) gesetzt. Analog zu den Pfeilen wurde ebenfalls die Transparenz der beiden Videoebenen so definiert, dass sie langsam in das Video ein- und ausgeblendet werden.

Die folgende Grafik zeigt die Ebenen in der Bearbeitungstimeline von *Adobe Premiere Pro*:

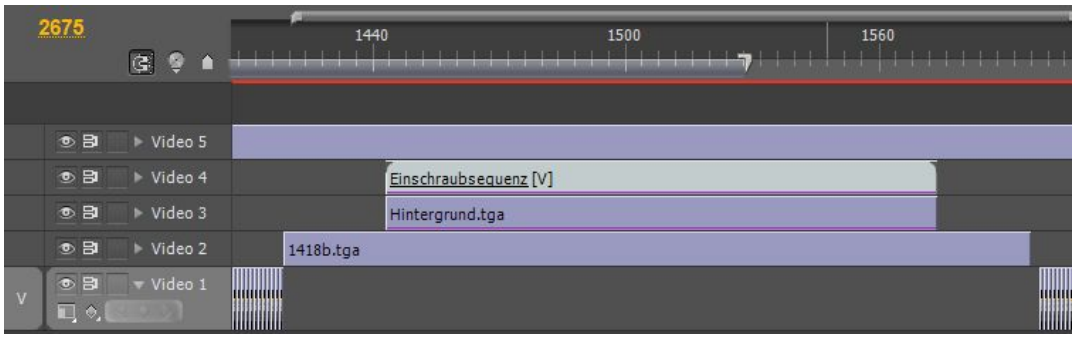


Abbildung 5.35: Ausschnitt der Bearbeitungstimeline mit mehreren Ebenen

Dieser Bearbeitungsprozess wurde bei allen Stellen des Videos, wo etwas eingeblendet werden sollte, wiederholt. Nachdem das Video komplett war, wurde noch ein Schrifttext unten rechts in der Ecke mit dem Namen des Erstellers eingefügt, der über das ganze Video sichtbar bleibt.

Da *Adobe Premiere Pro CS6* auch über gute Video-Komprimierer verfügt, wurden auch die Videos mit den in der Tabelle spezifizierten Einstellungen als Video exportiert.

Name:	Mainboard Anleitungsvideo
Format:	H.264
Auflösung:	1920x1080
Fernsehnorm:	NTSC
Framerate (fps):	30
Pixel-Seitenverhältnis:	Widerscreen 16:9
Bildcodierung:	VBR, 1-Pass
Ziel-Bitrate [Mbit/s]:	32
Maximale Bitrate [Mbit/s]:	40
Größe [MB]:	320

5.1.8.2 Erstellung von Texturen für die Videos mit Photoshop

Für die Videos mussten, wie im vorherigen Kapitel beschrieben, manche Einzelbilder bearbeitet werden. Im Allgemeinen waren das andere Bilder, die auf dem Grundbild eingebunden werden sollten (z.B. Pfeile auf dem Bild 1418). Hierzu wurde das Bild zunächst in *Photoshop* importiert und eine neue Ebene darüber erstellt. Auf dieser Ebene konnte dann gearbeitet werden, ohne das Grundbild zu verändern. Anschließend wurde auf der neuen Ebene ein Pfeil an die vorgesehene Stelle gezeichnet. Für das Design des Pfeiles wurde ein roter Verlauf mit einem schwarzen Rahmen gewählt. Dieser Pfeil wurde nach Fertigstellung achtmal kopiert und an die entsprechenden Stellen gerückt. Abschließend wurde das Bild als TGA mit der Bezeichnung 1418b abgespeichert.

Um die Pfeile auch in den späteren Bildern einheitlich darzustellen, wurde der Pfeil noch herausgeschnitten und in einem Extrabild eingefügt. Diese Bild besteht ausschließlich aus dem Pfeil und einem durchsichtigen Hintergrund. Um den transparenten Hintergrund auch später noch zu haben, durfte die Datei nicht als JPG[34] abgespeichert werden, da sonst der Hintergrund mit Weiß gefüllt würde. Gewählt wurde wieder das Bildformat TGA. Dieses Bild konnte später bei anderen Teilbildern importiert und positioniert werden.

5.1.9 Problemstellungen sowie Lösungswege

5.1.9.1 Von 3ds-Max 2010 zu 3ds-Max 2014

Leider musste mitten in der Fertigstellung der 3D-Modelle auf das Programm *3ds-Max 2014* gewechselt werden, da es größere Probleme beim Rendern mit der Version 2010 gab. So schien es, dass der Mental Ray-Renderer häufig Bugs hineinprojizierte. Auch konnten die Lichteffekte nicht richtig verarbeitet werden. Die Recherche ergab, dass es Komplikationen zwischen der *3ds-Max-Studentenversion* und *Windows7 64Bit* gab.

[34] JPG (engl.: Joint Photografic Experts Group)

Auch dauerte es ca. 2 Minuten, um ein einzelnes Bild zu rendern. Bei anfänglichen 4800 Bildern/Minute hätte das Rendern für jede Animation 160 Stunden bzw. fast 7 Tage benötigt. Dies war bei Weitem zu lang und ließ sich nur mit sichtbarem Qualitätsverlust verringern.

Zur Lösung wurde auf die neueste Version des Programmes *Studio 3ds-Max 2014* gewechselt. Diese neuere Version besitzt im Gegensatz zum Vorgänger eine GPU-Unterstützung. Da der Rechner, an dem ich arbeitete, mit einer sehr leistungsfähigen Grafikkarte (GeForce GTX 680 2048 GDDR5) ausgestattet war, wurde die Zeit für das Rendern deutlich verkürzt.

Nach Importieren der 3D-Modelle in das neue Programm traten sehr viele Fehler auf. Zum Beispiel beim Lüftergitter. Wie auf dem folgenden Bild zu sehen ist, kahm es insbesondere dort zu Polygonfehlern, wo sich die bool´schen Löcher befinden (Boolean = Formveränderung von Objekten durch Fusion oder Subtraktion).

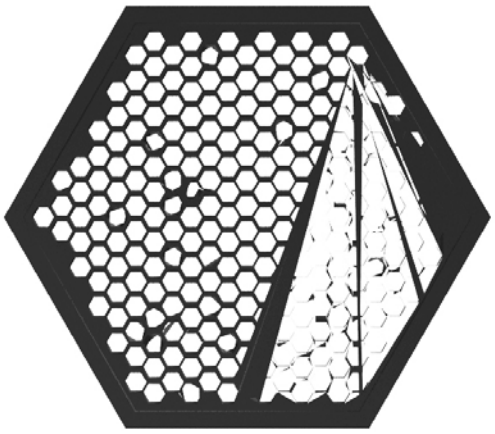


Abbildung 5.36: Lüftergitter mit Polygonfehlern

Ungefähr bei der Hälfte aller Boolean gab es Fehler im Polygonnetz. Diese waren auch nicht zu beheben. Da die Vorteile der 2014er Version eindeutig überwogen, mussten viele Objekte ein zweites Mal modelliert werden.

Licht/Schatten/Render-Einstellungen wurden neu konfiguriert und erbrachten sichtbar bessere Resultate.

5.1.9.2 Boolean-Löcher

Wie schon beschrieben, wurden bei der Modellierung der Bestandteile der Szene viele Booleans verwendet. Bei dem Boolean-Algorithmus berechnet das Programm eigenständig die benötigte Verformung und erstellt neue Polygonnetze. Das Problem dabei ist nur, dass dieses nicht immer zu 100% klappt. Zum Beispiel versucht das Programm immer entlang der Subtraktion eine neue Wand zu errichten. Allerdings funktioniert dieses nur selten, wenn man aus einem Objekt viele andere Objekte subtrahiert. In diesem Fall entsteht ein Loch ohne Innenwand.

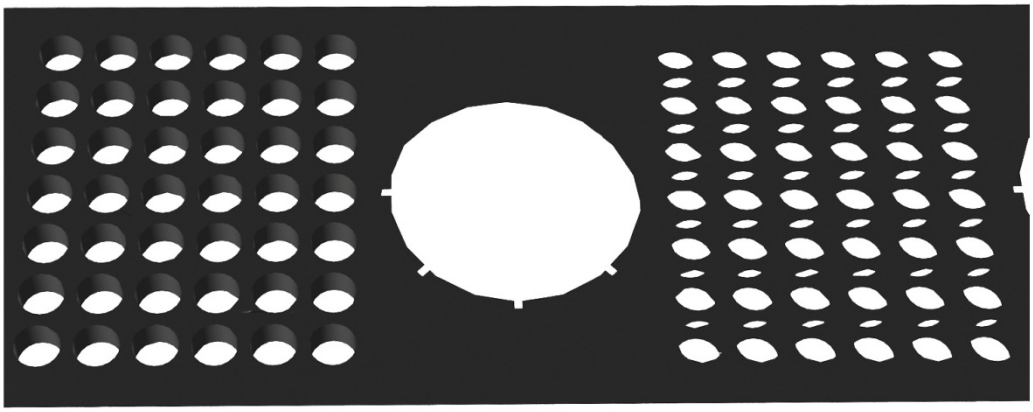


Abbildung 5.37: Vergleich Boolean-Löcher (links mit Innenwand, rechts ohne Innenwand)

Eine Möglichkeit, dieses zu verhindern, wäre es, alle Objekte die subtrahiert werden sollen, vorher zu einem Objekt zusammenzufügen. Allerdings funktioniert das ähnlich wie die Subtraktion. Das Programm stellt eine unendliche dünne Linie zwischen den frei "fliegenden" Objekten her, wodurch sie verbunden werden. Wenn man nun viele Objekte hat und diese zu einem Mesh transformiert, sieht man erst einmal keinen Unterschied. Jedoch berechnet das Programm bei der Subtraktion diese Hilfspolygone mit, wodurch es zu ungewollten Verzerrungen im Boolean kommt.

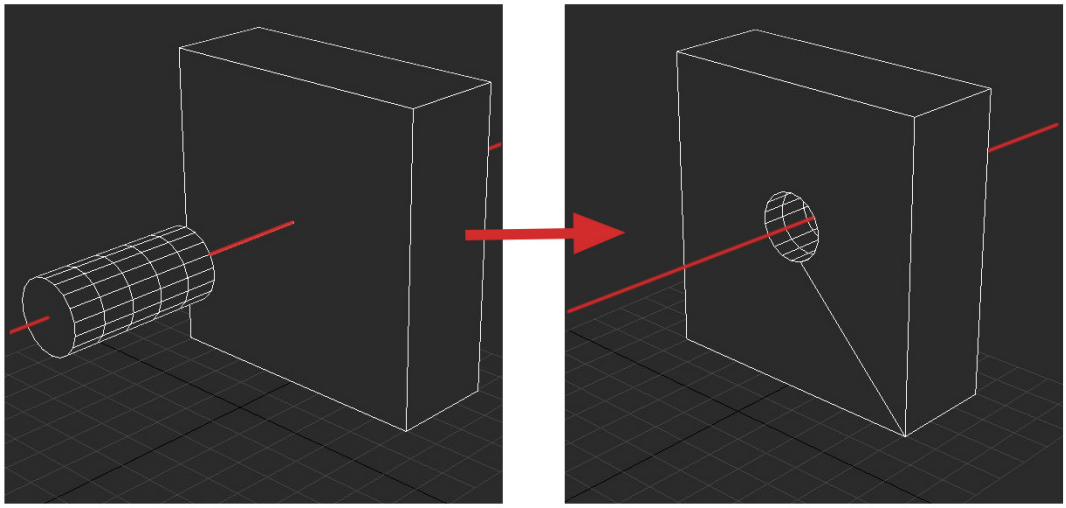


Abbildung 5.38: Beispiel einer Polygonhilfsline nach Subtraktion eines Zylinders vom einem Quader

3ds-Max 2014 bot eine verbesserte Version des Boolean Algorithmus an, der sogenannte "ProBoolean". Dieser hat eine deutliche höhere Erfolgsquote. Leider wurde dieses erst während des Projektes herausgefunden, sodass "defekte" Löcher nur teilweise neu erstellt wurden. Da es zu zeitaufwendig gewesen wäre, das Gehäuse neu zu modellieren, wurden die teilweise defekten Löcher hingenommen.

5.1.10 Zusammenfassung des Kapitels

In diesem Kapitel wurde aufgezeigt, wie die einzelnen 3D-Modelle entstanden sind und wie sie letztendlich aussehen. Des Weiteren wurde gezeigt, wie diese Bauteile animiert wurden und welche Arbeitsweise dafür benutzt worden ist, um ein bestimmtes Ergebnis zu erzielen. Außerdem wurde erläutert, welche Licht-/Schatten- und Render-Einstellungen zu welchem Zweck gewählt wurden. Im Abschnitt Userinterface-Design erfuhr man, wie der Entstehungsprozess des Designs und Layouts des Programms war.

Im nächsten Kapitel wird das bis hierhin Erstellte anhand von Probanden getestet und die Ergebnisse ausgewertet.

5.2 Paul Lindegrün

Im vorangegangenen Kapitel wurden die Konzepte besprochen, die bei der Implementierung zum Einsatz kommen sollten. In diesem Kapitel werden die Implementierungen des Programms, des Players und das Userinterface-Design beschrieben.

5.2.1 Installation der Software

Wie bereits im Kapitel 3 beschrieben, kamen *Visual Studio 2010* Professional als Entwicklungsumgebung, *Adobe Illustrator CS4* als Bildbearbeitungsprogramm, *VLC*-Bibliotheken samt Plug-ins als Videoplayer und VideoLan DotNet-Bibliotheken als Bindeglied zwischen dem .Net-Framework und dem *VLC*-Player zum Einsatz.

Die Installation und Inbetriebnahme der Programme verlief ohne Probleme. Lediglich das Einbinden der VideoLan DotNet-Bibliotheken gestaltete sich als schwierig. Näheres dazu im Kapitel 5.2.3.8.

Entwickelt wurde auf dem Betriebssystem *Windows 7 Professional x64*.

5.2.2 Erstellung des Grundgerüstes

Wie bereits im Kapitel 4.2.4 erwähnt, wurde zunächst ein Grundgerüst erstellt. Dazu wurde in *Visual Studio* ein neues Projekt zum erstellen einer *Windows*-Fensteranwendung angelegt. V*isual Studio* generierte dabei einige vorgefertigte Quellcode-Dateien. Zu einem waren dies Form1.cs, welche die Klasse Form1, die von der im .Net Framework definierten Klasse Form erbt und zum einen die Program.cs, welche den Einstiegspunkt der Anwendung, also die Main-Methode, enthält. Eine ausführliche und aktuelle Beschreibung der im .Net Framework definierten

Klassen, findet man auf der MSDN[35]-Homepage [19]. Damit war die Fensteranwendung bereits lauffähig und beim Ausführen wurde ein kleines *Windows*-Fenster ohne Inhalt angezeigt.

Nicht benötigte ***using***-Direktiven wurden aus dem Quellcode entfernt und die Klasse Form1 wurde umbenannt in TPHC, um dem Projektnamen zu entsprechen. Im Konstruktor der Klasse wurde die Breite des Fensters auf 1110 Pixel und die Höhe auf 770 Pixel gesetzt.

Als nächstes wurden dem Fenster einige Steuerelemente hinzugefügt.

5.2.2.1 Steuerelemente aus Button-Klasse

Das Windows-Fenster musste als nächstes mit Steuerelementen bestückt werden, welche später die Ansicht auf die nächste Ebene weiterleiten und veranlassen, sodass entsprechende Informationen, Bilder und Videos geladen werden. Das .Net Framework bietet eine Fülle von bereits vordefinierten Steuer- und Anzeigeelementen. Das gebräuchlichste Steuerelement, um solche Aufgaben zu bewältigen, ist das Button-Steuerelement. Es ist in der gleichnamigen Klasse definiert und kommt in fast allen Fensteranwendungen zum Einsatz. Zum Testen der Eigenschaften und Funktionsweise der Button-Klasse wurde ein Button der Fensteranwendung hinzugefügt und ausgeführt. Als nächstes wurde dem Button, mit Hilfe der Eigenschaft ***BackgroundImage*** und der Methode ***FromFile*** der Image-Klasse, eine Hintergrundtextur zugewiesen, da das Aussehen der Buttons später durch eigens erstellte Texturen bestimmt werden soll. Das Ergebnis war nicht zufriedenstellend. Der Button behielt seine Form und sein vordefiniertes Aussehen bei. Die Textur wurde lediglich zum Teil angezeigt, da sie größer war als der Button. Zwar ließ sich die Textur durch die Eigenschaft ***BackgroundImageLayout*** auf die Größe des Buttons einpassen, wurde dabei jedoch verzerrt. So musste die Größe des Buttons angepasst werden, bis die Textur nicht mehr verzerrt war.

Als nächstes musste das vordefinierte Aussehen des Buttons verändert werden, da es im Zusammenhang mit der Hintergrundtextur sehr störend war. Dazu wurde die Eigenschaft ***FlatStyle*** auf ***Flat*** gesetzt. Dadurch wurde der Button nun flach dargestellt und die Textur war dadurch besser zu erkennen.

Befand sich der Mauszeiger über dem Button, wurde der Hintergrund des Buttons hellgrau. Ebenso wurde der Hintergrund dunkelgrau, wenn eine Maustaste über dem Button betätigt wurde. Um dieses Verhalten zu unterbinden, wurde die Farbe der Eigenschaften ***MouseOverBackColor*** und ***MouseDownBackColor*** (die wiederum selbst Eigenschaften der Eigenschaft ***FlatAppearance*** sind), auf transparent gesetzt. Des Weiteren tauchte ein Rahmen um den Button auf, sobald er den Fokus verlor. Dieser wurde beseitigt, indem die Eigenschaft ***BorderSize*** (welche ebenfalls eine Eigenschaft der Eigenschaft ***FlatAppearance*** ist) auf 0 gesetzt wurde.

[35] MSDN (engl.: Microsoft Developer Network)

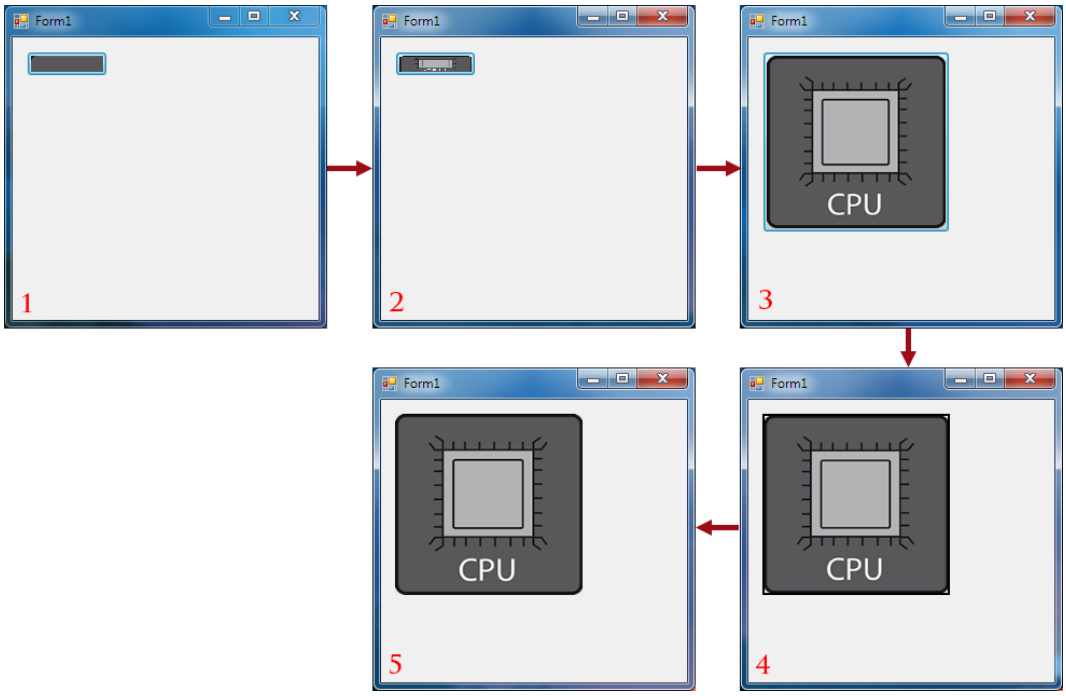


Abbildung 5.39: Ablauf der Anpassung eines Buttons

Nachdem die Textur ohne die störenden Eigenschaften des Buttons angezeigt wurde, musste dem Anwender ein optisches Feedback gegeben werden, wenn der Mauszeiger sich über dem Button befand. Damit der Anwender erkennen konnte, dass es sich bei diesem Button um ein aktives Steuerelement handelt. Um dies zu bewerkstelligen, wurde die Hintergrundtextur geändert, sobald sich der Mauszeiger über dem Button befand und wieder zurück geändert, sobald der Mauszeiger sich nicht mehr über dem Button befand. Das wurde mit Hilfe von zwei Methoden erreicht, die jeweils beim Auslösen des *MouseEnter*- und *MouseLeave*-Ereignisses ausgeführt wurden.

Beim Bewegen des Fensters über die Bildschirmoberfläche kam es zum Flackern der Textur. Die Vermutung lag nahe, dass die Bildwiederholungsfrequenz nicht synchronisiert mit dem Aufruf der Zeichnen-Methode des Buttons war. Um dies zu erzwingen, wurde die Eigenschaft *DoubleBuffered* auf *true* gesetzt. Dabei werden die Grafikdaten zuerst in einen Puffer und dann schnell in den Speicher der angezeigten Oberfläche geschrieben. Dies brachte jedoch keine Verbesserung und die Textur flackerte immer noch. Um das Problem zu lösen, mussten noch *ControlStyle*-Flags gesetzt werden. Dies waren:

- OptimizedDoubleBuffer
- AllPaintingInWmPaint
- UserPaint

Diese *Controlstyle*-Flags ließen sich jedoch nicht mehr setzen, nachdem das Button-Objekt initialisiert wurde.

5.2.2.2 Erstellung der MyButton-Klasse

Es wurde deutlich, dass zum korrekten Anzeigen eines einzigen Buttons schon sehr viele Einstellungen vorgenommen werden mussten. So musste dem Button eine Hintergrundtextur zugewiesen werden, das Layout der Textur musste angepasst werden, die Größe des Buttons musste angeglichen sowie das vordefinierte Aussehen musste geändert werden. Seine Verhaltensweisen mussten angepasst werden und für jeden Button mussten zwei Methoden implementiert werden, um ein visuelles Feedback für den Anwender zu ermöglichen. Weil schon die erste Ebene acht Buttons beinhalten sollte und die Einstellungen für jeden einzelnen Button getroffen werden mussten, war es auf lange Sicht keine akzeptable Lösung. Daher wurde beschlossen, eine eigene MyButton-Klasse zu implementieren, welche von der Button-Klasse erbt.

Als erstes wurden zwei Felder, *stand_Image* und *highlight_Image* vom Typ *Image*, definiert. Das *stand_Image*-Feld sollte die Textur enthalten, die standartmäßig angezeigt werden sollte und das *highlight_Image*-Feld sollte die Textur enthalten, die angezeigt werden sollte, wenn sich der Mauszeiger über dem Steuerelement befindet. Als nächstes wurden ein Standardkonstruktor und ein allgemeiner Konstruktor definiert. Im Standardkonstruktor wurden die beiden Image-Felder zu *null* gesetzt, da der Standardkonstruktor ein MyButton-Objekt ohne eine Textur erzeugen sollte. Des Weiteren wurden die Eigenschaften für *FlatStyle*, *FlatAppearance* sowie die *DoubleBuffer*-Flags gesetzt. Im allgemeinen Konstruktor wurden ebenfalls die Eigenschaften und Flags gesetzt. Da dieser Konstruktor dafür gedacht war, ein MyButton-Objekt mit Texturen zu erzeugen, wurde ihm auch noch ein Übergabeparameter vom Typ *string* gegeben. Der Übergabeparameter enthielt den Namen der beiden Texturen. Dieser Name wurde im allgemeinen Konstruktor in einen vordefinierten Pfad eingesetzt, von wo aus die beiden Texturen aus zwei Projektordnern eingelesen und den *Image*-Feldern übergeben wurden. Zum Schluss wurde die Größe (Höhe und Breite) der Texturen ermittelt und das Steuerelement an diese Größe angepasst. Dies hatte den Vorteil, dass die Textur vollständig und verzerrungsfrei dargestellt wurde, wodurch eine nachträgliche Anpassung entfiel.

Manche der Steuerelemente sollten eventuell eine Tooltip-Anzeige erhalten, die beim Verweilen des Mauszeigers über dem Steuerelement angezeigt werden sollte. Um dies zu erreichen, wurde der MyButton-Klasse ein weiteres Feld *tip* vom Typ *ToolTip* hinzugefügt. Um diesem *tip*-Feld eine Beschreibung zu übergeben, wurde ein weiterer Konstruktor, diesmal mit zwei Übergabeparametern, implementiert, wobei der zweite Übergabeparameter die Beschreibung enthielt und vom Typ *string* war. Dieser Konstruktor war identisch mit dem allgemeinen Konstruktor und wurde dahingehend erweitert, um die Beschreibung an das *tip*-Feld zu übergeben und es anzuzeigen.

Um den Texturwechsel, beim Eintreten des Mauszeigers in das Steuerelement, zu ermöglichen, wurden die geerbten Methoden *OnPaint*, *OnMouseEnter* und *OnMouseLeave*, in der MyButton-Klasse überschrieben. Zusätzlich wurde der Klasse ein weiteres Feld *mouse_enter* vom Typ *bool* hinzugefügt. Die *OnMouseEnter*-Methode wurde aufgerufen, wenn das *MouseEnter*-Ereignis eintrat, also, wenn der Mauszeiger in das Steuerelement hinein bewegt wurde.

Entsprechend wurde die *OnMouseLeave*-Methode aufgerufen, wenn der Mauszeiger aus dem Steuerelement heraus bewegt wurde. In der *OnMouseEnter*-Methode, wurde zuerst die *OnMouseEnter*-Methode der Basisklasse aufgerufen. Danach wurde das Mauszeiger-Symbol in eine Hand geändert, um ein zusätzlichen visuelles Feedback für den Anwender bereitzustellen. Zum Schluss wurde das *mouse_enter*-Feld auf *true* gesetzt und die *Invalidate*-Methode ausgerufen, welche das Neuzeichnen des Steuerelementes veranlasste. Die *OnMouseLeave*-Methode war identisch aufgebaut, nur wurde entsprechend der Funktionsweise das Mauszeiger-Symbol zurück in einen Pfeil geändert und das *mouse_enter*-Feld auf *false* gesetzt. In der *OnPaint*-Methode wurde zuerst die *OnPaint*-Methode der Basisklasse aufgerufen. Danach fanden mehrere Fallunterscheidungen statt. Zuerst wurde abgefragt, ob überhaupt Texturen zum Zeichnen vorlagen. Wenn diese Bedingung erfüllt war, wurde abgefragt, ob das *mouse_enter*-Feld den Wert *true* oder *false* hatte. Je nachdem wurde dann entweder die *stand_Image*-Textur oder die *highlight_Image*-Textur gezeichnet.

Diese Klasse funktionierte zufriedenstellend. Beim Erstellen eines neuen MyButton-Objektes, musste diesem lediglich der Name der Textur und eventuell ein Beschreibung, die als Tooltip angezeigt werden sollte, übergeben werden. Alle anderen Einstellungen wurden von den Konstruktoren gesetzt und die Verhaltensweisen von den implementierten Methoden beschrieben.

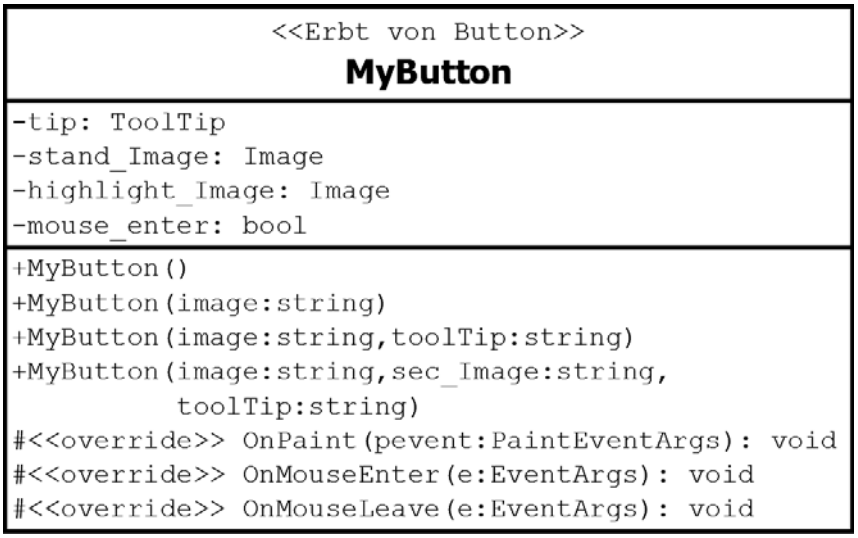


Abbildung 5.40: UML-Diagramm der Klasse MyButton

5.2.3 Erweiterung des Programms

Nachdem das Grundgerüst des Programms, mit acht MyButton-Elementen, erstellt worden ist, wurde es Schrittweise um weitere Steuer- und Anzeigeelemente erweitert.

5.2.3.1 Erweiterung der TPHC-Klasse

Als erstes wurde die TPHC-Klasse erweitert. Zuerst wurde dem Fenster ein weiterer MyButton hinzugefügt, welcher für den Aufruf eines Informationsfensters zuständig war (vgl. Kapitel

5.2.3.2). Da das Fenster in der Größe verändert werden konnte, musste die Position der Steuer- und Anzeigeelemente neu berechnet werden, damit diese nicht außerhalb des Fensterrahmens landeten. Hierzu wurde die Methode ***Resize_Handler*** implementiert, welche beim Auslösen des Resize-Ereignisses, also beim Verändern der Fenstergröße, ausgeführt wurde. In der ***Resize_Handler***-Methode wurde die ***Locate_Elements***-Methode aufgerufen. Hier wurde die Position, gegebenenfalls auch die Größe, aller Steuer- und Anzeigeelemente, in Abhängigkeit von der Fenstergröße, berechnet. Dies wurde aus Übersichtlichkeitsgründen so gemacht, da in der ***Resize_Handler***-Methode eventuell später noch weitere Methodenaufrufe stattfinden könnten.

Als nächstes wurde die Methode ***ToSecond_Handler*** implementiert. Diese wurde aufgerufen, sobald einer der acht MyButtons, auf der ersten Ebene, geklickt wurde. In der Methode wurde durch eine Fallunterscheidung ermittelt, welches der Steuerelemente geklickt wurde. Daraufhin wurden ein entsprechendes Bild für die Großansicht, ein Infotext und ein Video aus den Projektordnern geladen.

Zum Schluss musste eine Möglichkeit zum Wechseln auf eine andere Ebene implementiert werden. Der erste Ansatz war es, alle Elemente der ersten Ebene auf ein Panel zu setzen, welches den gesamten Fensterbereich ausfüllte. So konnte durch Festlegung der Sichtbarkeit des Panels auch die Sichtbarkeit der sich auf ihn befindenden Elemente festgelegt werden. Entsprechend für jede Ebene, würde es drei Panels mit den jeweiligen Elementen geben, deren Sichtbarkeit durch Aufrufen von den Methoden ***Window_1_Visible***, ***Window_2_Visible*** und ***Window_3_Visible*** geregelt werden würde. Dies hat jedoch nicht so gut funktioniert wie erwartet. Auf den Texturen der Steuerelemente befanden sich transparente Bereiche. In den transparenten Bereichen wurde nicht das dahinter liegende Panel angezeigt, sondern die Hintergrundfarbe des Fensters selbst, was zu unschönen Anzeige-Artefakten führte.

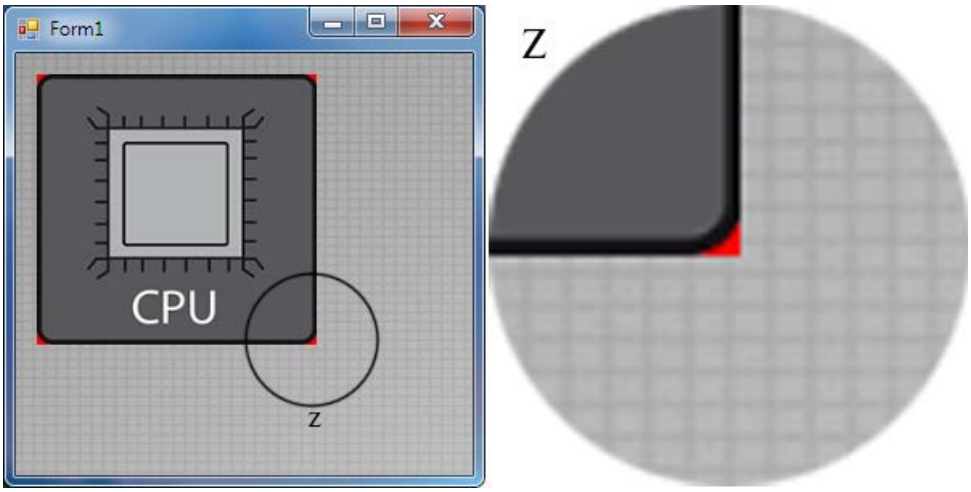


Abbildung 5.41: Durchscheinen der Hintergrundfarbe in transparenten Bereichen

Um dieses Problem zu umgehen, wurden die Elemente auf das Fenster selbst gesetzt und die Sichtbarkeit von jedem Element einzelnen, in den Methoden ***Window_1_Visible***, ***Window_2_Visible*** und ***Window_3_Visible***, geregelt.

5.2.3.2 Erstellung der InfoForm-Klasse

Zum Anzeigen allgemeiner Informationen über das Programm, wurde ein neues Fenster benötigt, welches beim Klicken auf den Informations-Button aufgerufen wurde.

Dazu wurde die InfoForm-Klasse, welche von der Form-Klasse erbt, erstellt. Damit der gestalterische Gesamteindruck nicht durch den Fensterrahmen gestört würde, wurde dieser deaktiviert. Dies geschah im Standardkonstruktor, indem die Eigenschaft *FormBorderStyle* auf *None* gesetzt wurde.

Da nun kein Fensterrahmen mehr zur Verfügung stand, musste ein Button hinzugefügt werden, der für das Schließen der InfoForm verantwortlich war. Dieser wurde in der rechten oberen Ecke der InfoForm positioniert und als Text wurde ihn lediglich ein 'X' zugewiesen. Beim Klicken auf den Button wurde die Methode *Close_Handler* aufgerufen. In dieser wurde wiederrum die geerbte Methode *Close* aufgerufen, welche die InfoForm beendete und den Speicherplatzt freigab.

Dadurch, dass der Fensterrahmen entfernt wurde, war der optische Übergang von der InfoForm zum Hauptfenster abrupt. Um eine visuelle Grenze bei dem Übergang zu erzeugen, wurden die Außenkanten der InfoForm mit einer sich von der Hintergrundfarbe abhebenden, Farbe bemalt. Dazu wurde die *OnPaint*-Methode überschrieben. In dieser wurde zuerst die *OnPaint*-Methode der Basisklasse aufgerufen. Danach wurde mit Hilfe der *DrawRectangle*-Methode der Graphics-Klasse ein Rechteck in der Größe der InfoForm gezeichnet.

Damit die Farbe des Rahmens verändert werden konnte, wurde die InfoForm um das Feld *borderColor* vom Typ *Color* erweitert. Außerdem wurde die Eigenschaft *BorderColor* hinzugefügt. Diese besaß einen get und set accessor. Dadurch konnte dem *borderColor*-Feld zur Laufzeit eine neue Farbe zugewiesen oder abgefragt werden.

Um einen Informationstext auf der InfoForm ausgeben zu können, wurde die InfoForm um das Feld *text* vom Typ *string* erweitert. Der anzuzeigende Text, wurde im Standardkonstruktor aus einer Textdatei, welche sich in einen Projektordner befand, eingelesen und dem *text*-Feld übergeben. Mit Hilfe der Methode *DrawString* der Graphics-Klasse wurde der eingelesene Text, in der *OnPaint*-Methode, auf die InfoForm gezeichnet.

Beim Klicken des Info-Buttons im Hauptfenster, wurde die Methode *InfoForm_Handler* aufgerufen. In dieser wurde ein neues InfoForm-Objekt erstellt und mit der Methode *ShowDialog* gestartet. Dies hatte den Vorteil, dass die InfoForm automatisch das Hauptfenster als sein Elternelement übernommen hatte und dadurch immer in der Mitte des Hauptfensters gestartet wurde, egal wo es sich auf dem Bildschirm befand.

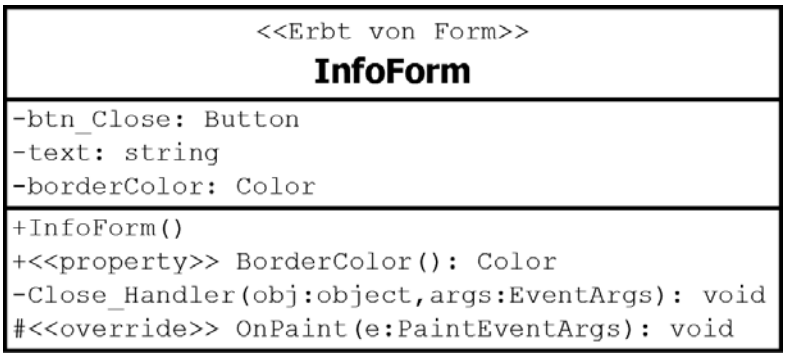


Abbildung 5.42: UML-Diagramm der Klasse InfoForm

5.2.3.3 Erstellung der DoubleBufferPanel-Klasse

Als Anzeigeelement für Bilder und Texte wurde die, im .Net Framework definierte, Panel-Klasse gewählt. Diese hatte jedoch dieselben Probleme, im Hinblick auf das Flackern, wie die im Kapitel 5.2.2.1 beschriebene Button-Klasse. Um das Flackern in den Griff zu bekommen, mussten die ***DoubleBuffer***-Flags der Klasse gesetzt werden. Da dies jedoch nach der Initialisierung des Panel-Objektes nicht mehr möglich war, wurde eine eigene DoubleBufferPanel-Klasse, die von der Panel-Klasse erbt, implementiert.

Die DoubleBufferPanel-Klasse enthielt lediglich einen Standardkonstruktor, indem die Flags gesetzt wurden. Implementierungen von weiteren Methoden waren nicht nötig, da der Funktionsumfang der Basisklasse bereits ausreichend war.

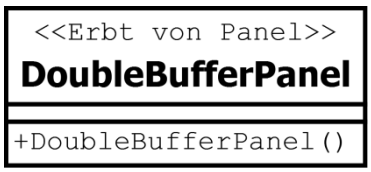


Abbildung 5.43: UML-Diagramm der Klasse DoubleBufferPanel

5.2.3.4 Erstellung der InfoPanel-Klasse

Zum Ausgeben der bauteilspezifischen Informationen wurde ein Anzeigeelement benötigt, welches Texte und gegebenenfalls auch Bilder ausgeben konnte. Falls die Textlänge die Höhe des Anzeigeelementes übersteigen würde, sollten Scroll-Balken eingeblendet werden.

Die Wahl fiel auf das DoubleBufferPanel, da es von der Panel-Klasse erbte und dadurch bereits die Eigenschaft ***AutoScroll*** mit sich brachte. Um die Funktionsweise zu testen, wurde die Klasse um das Array ***text*** vom Typ ***string*** erweitert. Im Konstruktor wurde ein Text mit Hilfe der Methode ***ReadAllLines*** der File-Klasse, zeilenweise in das Array eingelesen. Um den eingelesenen Text ausgeben zu können, wurde die ***OnPaint***-Methode überschrieben. In dieser wurde das Array, mit Hilfe der Methode ***DrawString*** der Graphics-Klasse, auf das DoubleBufferPanel gezeichnet. Da das Array in jeder Speicherzelle nur eine Zeile des Textes enthielt und die ***DrawString***-Methode immer nur jeweils eine Zeile zeichnen konnte, musste beim Zeichnen durch das Array iteriert werden. Um die Zeilen untereinander auszugeben, wurden in der ***OnPaint***-Methode zwei lokale Variablen erstellt. Dies war zum einen ***startPosition*** vom Typ

Point und zum anderen *lineHeight* vom Typ *int*. Der Variablen *lineHight* wurde die maximale Höhe des zurzeit verwendeten Fonts zugewiesen. Der Y-Wert der Variablen *startPosition* wurde bei jedem Iterationsdurchgang um den Wert der *lineHeight*-Variablen erhöht und als neue Startposition der nächsten Zeile an die *DrawString*-Methode übergeben. Beim Ausführen des Programms stellte sich heraus, dass der Text richtig ausgegeben wurde, jedoch wurden keine Scroll-Balken angezeigt. Das führte dazu, dass es nicht möglich war, den Text, der über die Ränder des DoubleBufferPanels ging, zu lesen.

Damit die Scroll-Balken angezeigt werden konnten, war es nötig, das Innere des DoubleBufferPanels mit einem anderen Element zu füllen. Daher wurde beschlossen, die InfoPanel-Klasse zu implementieren. Wie im vorherigen Testversuch besaß die Klasse ein Array vom Typ *string*. Die Methode *ChildPanelPaint*, die auf das *Paint*-Ereignis von *childPanel* reagiert, wurde implementiert. Zusätzlich besaß die Klasse das Feld *childPanel* vom Typ *DoubleBufferPanel*, welches das InfoPanel als sein Elternelement übernahm. Damit die Scroll-Balken angezeigt wurden, musste das *childPanel*, in der Breite oder Höhe, größer sein als sein Elternelement. Die Größe des *childPanels* wiederum sollte sich nach der Länge des eingelesenen Textes richten. Zu diesem Zweck wurden zwei weitere Methoden, *GetText* und *GetLongestLine*, implementiert. Die *GetText*-Methode bekam den Pfad zur Textdatei als Übergabeparameter. Der Text wurde in das Array eingelesen. Als nächstes wurde die *GetLongestLine*-Methode aufgerufen und ihr das Array übergeben. Die *GetLongestLine*-Methode ging jede Zeile im Array durch, verglich dabei die Länge der einzelnen Zeilen und gab die längste vorkommende Zeile als *string* zurück. Die zurückgegebene Zeile wurde dann mit Hilfe der Methode *MeasureString* der Graphics-Klasse und des derzeitigen Fonts vermessen und als Breite des *childPanel* festgelegt. Um die Höhe des *childPanel* anhand des Textes festzulegen, wurde die Anzahl der Zeilen, also die Anzahl der Array-Elemente, mit der Höhe des derzeitigen Fonts multipliziert.

Um Bilder, welche zum Text gehörten, ausgeben zu können, musste zuerst festgelegt werden, wie die Bilder im Text referenziert werden sollten und die *ChildPanelPaint*-Methode musste erweitert werden. Es wurde festgelegt, dass der Verweis auf ein Bild in einer neuen Zeile, mit dem Zeichen ‘@‘ gefolgt von der Pfadangabe des Bildes, welches sich im selben Ordner befinden musste wie die Textdatei, beginnen muss. In der Iterationsschleife der *ChildPanelPaint*-Methode wurde eine Fallunterscheidung hinzugefügt. Diese kontrollierte, ob das erste Zeichen in der nächst zu zeichnenden Zeile ein ‘@‘ ist. Wenn das der Fall war, wurde diese Zeile als Pfadangabe behandelt und an die *DrawImage*-Methode der Graphics-Klasse übergeben. Diese passte die Größe des Bildes an die Breite des *childPanel* an, wodurch die Bilder vorher nicht pixelgenau zugeschnitten werden mussten.

Um die Höhe des *childPanel* samt den Bildern zu berechnen, wurde die *GetText*-Methode um eine ähnliche Iterationsschleife erweitert. Diese addierte die Höhe aller im Text vorkommender Bilder zu der Höhe des *childPanels*.

Die *GetText*-Methode wird in der *ToSecond_Handler*-Methode der TPHC-Klasse aufgerufen, also nachdem im Hauptfenster ein Bauteil ausgewählt wurde.

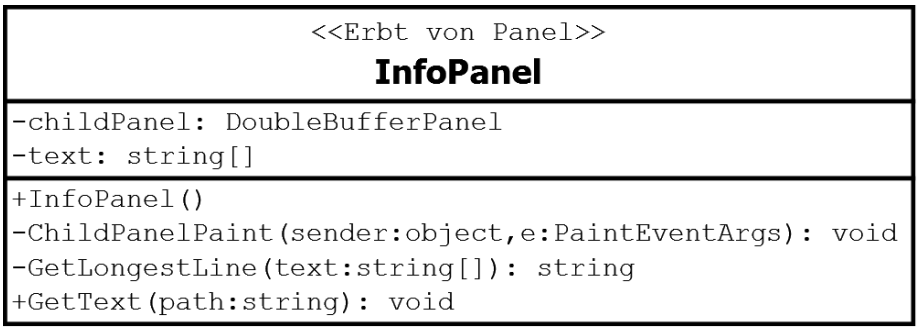


Abbildung 5.44: UML-Diagramm der Klasse InfoPanel

5.2.3.5 Einbindung von VideoLan DotNet-Bibliotheken

Bevor die *VLC*-Bibliotheken und die VideoLan DotNet-Bibliotheken in das Hauptprogramm implementiert wurden, wurden zuerst ihre Funktionsweisen in einem separaten Testprogramm analysiert. Dazu wurde in *Visual Studio* ein neues Projekt zur Erstellung einer Fensteranwendung generiert. Um an die *VLC*-Bibliotheken heranzukommen, wurde der VLC-Player heruntergeladen [17] und installiert. Danach wurden die Dateien liblvc.dll und libvlccore.dll sowie der Plug-ins Ordner aus dem Installationsverzeichnis in den Projektordner kopiert. Als nächstes wurden die VideoLan DotNet-Bibliotheken heruntergeladen [18]. Die Dateien Vlc.DotNet.Core.dll, Vlc.DotNet.Forms.dll und Vlc.DoNet.Interops.dll wurden als Verweise dem Projekt hinzugefügt sowie die entsprechenden ***using***-Direktiven angegeben.

Die Analyse der Funktionsweise gestaltete sich als schwierig, da es keine Dokumentation der VideoLan DotNet-Klassen gab. Auf der Hersteller-Seite gab es lediglich eine Anleitung, wie der Player vor dem Starten der Fensteranwendung initialisiert werden muss. So müssen der statischen VlcContext-Klasse zuallererst die Pfadangaben über den Aufenthaltsort der *VLC*-Bibliotheken und des Plug-in Ordners übergeben werden. Nachdem die Fensteranwendung beendet wurde, mussten alle vom Player benutzten Ressourcen durch den Aufruf der ***CloseAll***-Methode wieder freigegeben werden.

In die Form-Klasse wurde das Feld ***myPlayer*** vom Typ ***VlcControl*** hinzugefügt. Dies war das Anzeigeelement des Players. Es konnte in der Fensteranwendung positioniert werden, bot eine Playlist, in die die abzuspielenden Videos geladen werden konnten und es bot Methoden zum Widergeben, Pausieren und Stoppen des Videos. Damit ein Video in die Playlist geladen werden konnte, musste zuerst ein Objekt vom Typ ***PathMedia*** erzeugt werden. Dieses benötigte bei der Initialisierung eine Pfadangabe zum Video. Danach wurde das ***PathMedia***-Objekt der Playlist vom ***myPlayer*** hinzugefügt.

Zum Testen des Players wurden einige Button zu der Fensteranwendung hinzugefügt. Ein Button öffnete einen ***OpenFile***-Dialog, in welchem eine Videodatei ausgewählt werden konnte und der Pfad der Datei an ein ***PathMedia***-Objekt übergeben wurde. Weitere Buttons waren jeweils dafür zuständig, die ***Play***-Methode, die ***Pause***-Methode und die ***Stop***-Methode vom ***myPlayer***-Feld aufzurufen. Dazu wurden die Methoden ***Play_Handler*** und ***Stop_Handler*** implementiert. Diese wurden jeweils beim Klicken des Stop- bzw. des Play-Buttons aufgerufen.

In diesen fanden Fallunterscheidungen statt, die dafür sorgten, dass das Video pausiert, gestoppt und wiedergegeben werden konnte.

Nachdem die Anwendung gestartet, ein Video ausgewählt und der Play-Button gedrückt wurde, passierte nichts. Das Video wurde nicht abgespielt. Es war zuerst unklar ob die Implementierung falsch war. Nach einiger Recherche auf der VideoLan DotNet-Homepage kam jedoch heraus, dass die zum herunterladen bereitgestellten Bibliotheks-Dateien nicht der aktuellen Version entsprachen und einen Fehler enthielten, welcher verhinderte, dass die ***Play***-Methode nicht richtig funktionierte. Um das Problem zu lösen, musste der aktuellere Quellcode heruntergeladen und neu kompiliert werden. Der Quellcodedateien lagen bereit als ein *Visual Studio* Projekt vor und mussten daher nur noch kompiliert werden. Nach dem Kompilieren, wurden die neuen VideoLan DoNet-Bibliotheken erneut in das Projekt eingebunden und die Anwendung ausgeführt. Nun wurde das Video korrekt abgespielt und auch die anderen Funktionen, zum pausieren, stoppen und wiedergeben, funktionierten einwandfrei.

Nachdem ein Video gestoppt und ein anderes geöffnet werden sollte, wurde nicht das neue Video abgespielt, sondern das Video, welches als allererstes in der Playlist hinzugefügt worden ist. Um das Problem zu lösen, wurde der Inhalt der der Playlist, beim öffnen eines neuen Video, gelöscht.

5.2.3.6 Erstellung der Volume-Klasse

Um die Lautstärke des Videos zu regeln, wurden weitere Steuerelemente benötigt. Diese wurden in das Testprogramm, welches im Kapitel 5.2.3.5 beschrieben wurde, eingebunden und getestet.

Das Stummschalten des Videos wurde durch einen MyButton übernommen. Beim Klicken auf diesen wurde die Methode ***Mute_Handler*** aufgerufen. In dieser fand eine Fallunterscheidung statt. Wenn das Video nicht stummgeschaltet war, wurde es stummgeschaltet und umgekehrt. Dabei wurde die Eigenschaft ***IsMute*** des ***myPlayer*** Feldes abgefragt bzw. auf ***true*** oder ***false*** gesetzt.

Damit der Lautstärke-Wert des Videos zwischen 0 bis 100% geändert werden konnte, wurde ein anderes Steuerelement benötigt, da ein Button lediglich auf einen Klick reagieren kann. Dazu wurden die Steuerelemente „Schieberegler“ und „Ladebalken“, welche im .Net Framework definiert sind, getestet. Diese erfüllten die Anforderungen nur zum Teil. Das Hauptproblem war, dass das Aussehen der Steuerelemente sich je nach verwendeter Windows-Version änderte und somit keine einheitliche Gestaltung des Userinterfaces möglich war. Daher wurde beschlossen, ein eigenes Steuerelement zu implementieren, welches die funktionalen und gestalterischen Anforderungen erfüllen würde.

Ziel der Volume-Klasse war es, eine Kombination aus Anzeige- und Steuerelement zu erstellen. Diese sollte den aktuellen Lautstärke-Wert anzeigen oder diesen beim Klicken auf das Element, in Abhängigkeit des Klick-Ortes auf dem Element, verändern. Um einige der Funktionalitäten wie bei anderen Steuer- und Anzeigeelementen zu erhalten, erbte die Klasse von der UserControl-Klasse, welche im .Net Framework definiert ist. Die Breite des Elements wurde im

Konstruktor auf 100 Pixel gesetzt, so stand ein Pixel für einen Prozentwert der Lautstärke. Des Weiteren wurde die Hintergrundfarbe auf Transparent gesetzt und *DoubleBuffer*-Flags ebenso, um das Flackern beim Anzeigen zu minimieren. Es wurden die Felder *fillBarRectangle* vom Typ *Rectangle*, *fillBarColor* und *borderColor* vom Typ *Color* hinzugefügt. Die *OnPaint*-Methode wurde überschrieben. In ihr wurden mit Hilfe der *DrawRectangle*-Methode der Graphics-Klasse und des *borderColor*-Feldes, die Ränder des Elements gezeichnet. Mit Hilfe der *FillRectangle*-Methode, der beiden *fillBarRectangle* und *fillBarColor*, wurde das Innere des Elements ausgefüllt. Dieser Füllbalken repräsentierte dabei den aktuellen Lautstärke-Wert. Damit der Füllbalken bis zum Ort des Klicks gefüllt werden konnte, wurden die Methoden *OnMouseClick* und *OnMouseMove* überschrieben. In diesen wird die Breite des *fillBarRectangle*-Feldes an den Klick-Ort des Elements angepasst und ein Neuzeichnen veranlasst. In der TPHC-Klasse konnte so auf die *MouseClick*- und *MouseMove*-Ereignisse reagiert und die Klick-Position als neuer Lautstärke-Wert an den Player übergeben werden. Damit der Füllbalken geleert werden konnte, sobald das Video stummgeschaltet wurde, wurde die Eigenschaft *Mute* implementiert. Diese wies dem Füllbalken den übergebenen Wert zu und veranlasste das Neuzeichnen des Elements. Beim Klicken auf das Volume-Element wurde die Methode *Volume_Handler* aufgerufen. Hier fand eine Fallunterscheidung statt, welche die Position des Mauszeigers beim Klick auf dem Volume-Element bewertete. War die Position kleiner als 0, wurde Lautstärke auf 0% gesetzt. War die Position größer als 100, wurde die Laustärke auf 100% gesetzt. War die Position zwischen 0 und 100, wurde die Lautstärke auf diesen Wert gesetzt. Zum Schluss wurden der Volume-Klasse die Eigenschaften *FillBarColor*, *BorderColor* und *ControlColor* hinzugefügt. Durch diese konnte die Farbe des Füllbalkens bzw. des Rahmens gesetzt werden.

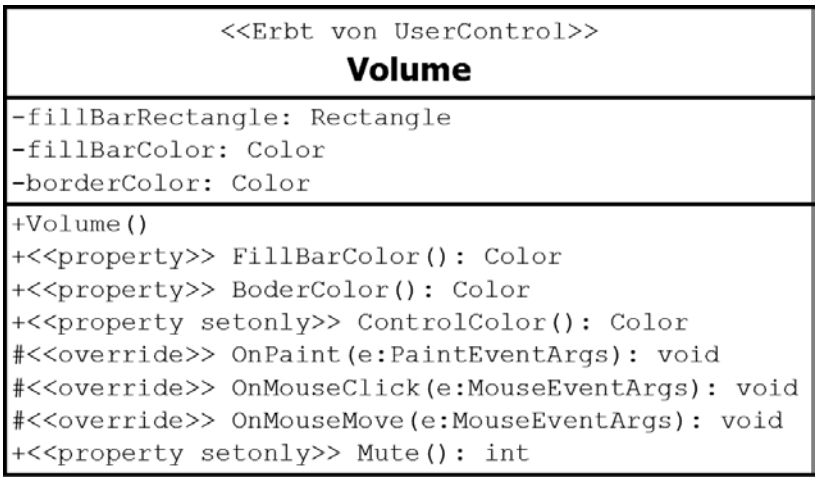


Abbildung 5.45: UML-Diagramm der Klasse Volume

5.2.3.7 Erstellung der Timeline-Klasse

Um die Wiedergabeposition des Videos beim Abspielen zu verändern, wurde ein weiteres Steuerelement benötigt. Dieses wurde in das Testprogramm, welches im Kapitel 5.2.3.5 beschrieben, eingebunden und getestet.

Aufgrund der gesammelten Erfahrungen beim Erstellen der Volume-Klasse, im Kapitel 5.2.3.6, wurde beschlossen, eine Timeline-Klasse zu erstellen, welche nach ähnlichen Prinzipien

funktionieren sollte. Ziel der Timeline-Klasse war es, eine Kombination aus Anzeige- und Steuerelement zu erstellen. Diese sollte die aktuelle Position des Videos anzeigen oder diese beim Klicken auf das Element, in Abhängigkeit des Klick-Ortes verändern. Um einige der Funktionalitäten wie bei anderen Steuer- und Anzeigeelementen zu erhalten, erbte die Klasse von der UserControl-Klasse, welche im .Net Framework definiert ist.

Es wurden, wie bereits in der Volume-Klasse, die Felder *fillBarRectangle* vom Typ *Rectangle*, *fillBarColor* und *borderColor* vom Typ *Color* hinzugefügt. Es wurden ebenfalls die *OnPaint*- und *OnMouseClick*-Methoden überschrieben. In der *OnPaint*-Methode werden der Rahmen und der Füllbalken des Elements gezeichnet. In der *OnMouseClick*-Methode wird die Breite des Füllbalkens an die Klick-Position angepasst. Im Konstruktor wurden die *DoubleBuffer*-Flags gesetzt, jedoch keine feste Größe für das Element angegeben, da diese beim Verändern der Fenstergröße ebenfalls geändert würde.

Das *myPlayer*-Feld bot zum Setzten oder Abfragen der Wiedergabeposition die Eigenschaft *Position* an. Zusätzlich konnte das *PositionChanged*-Ereignis dazu benutzt werden, die Breite des Füllbalkens kontinuierlich steigen zu lassen. Um die Breite des Füllbalkens eines Timeline-Objekts setzten zu können, wurde die Eigenschaft *Progress* implementiert. In dieser wurde der Füllbalken an den übergebenen Wert angepasst. Beim Eintreten des *PositionChanged*-Ereignisses des Players wurde die Methode *TimelineFill_Handler* aufgerufen. In dieser wurde der Wert von *Position* an *Progress* übergeben. Beim Eintreten des *MouseClick*-Ereignisses des Timeline-Objekts wurde die Methode *Position_Handler* aufgerufen. In dieser wurde das Verhältnis der Breite des Füllbalkens zur Gesamtbreite des Timeline-Objekts an *Position* übergeben. Damit der Füllbalken geleert werden konnte, sobald ein neues Video gestartet wurde, wurde die Methode *Empty* implementiert. Zum Schluss wurden der Timeline-Klasse die Eigenschaften *FillBarColor*, *BorderColor* und *ControlColor* hinzugefügt. Durch diese konnte die Farbe des Füllbalkens bzw. des Rahmens gesetzt werden.

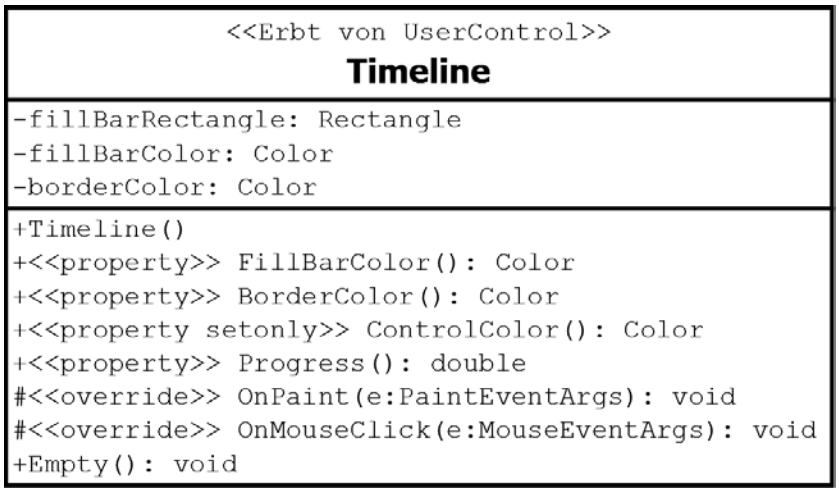


Abbildung 5.46: UML-Diagramm der Klasse Timeline

5.2.3.8 Implementierung des Players

Da sich nun mit der Funktionsweise der VideoLan DotNet-Bibliotheken in einem separaten Testprogramm vertraut gemacht wurde, mussten der erstellte Player und die erstellten Anzeige- bzw. Steuerelemente in die TPHC-Klasse eingebunden werden.

Die TPHC-Klasse wurde dabei auf zwei Quellcode-Dateien, 'TPHC Form.cs' und 'TPHC Player Form.cs', aufgeteilt. In der ersten Datei wurden die Anzeige- und Steuerelemente, der ersten und zweiten Ebene beschrieben. In der zweiten Datei wurden die Elemente des Players beschrieben. Dazu wurde das Schlüsselwort ***partial*** vor die Klassendefinition gesetzt. Dadurch konnte der Linker erkennen, dass es sich bei diesen Dateien um eine Klasse handelt. Das Aufteilen der Klassendefinition auf mehrere Dateien diente dazu, die Übersicht zu bewahren und die Player-Definition von den anderen Ebenen zu trennen.

Die im Testprogramm erstellten Methoden und Klassen-Definitionen konnten ohne zusätzliche Anpassungen in das Hauptprogramm übernommen werden.

Die TPHC-Klasse wurde als nächstes um die Methode ***ToThird_Handler*** erweitert. Diese wurde aufgerufen, sobald der Video-Starten-Button in der zweiten Ebene geklickt wurde. In ihr wurde die Methode ***Window_3_Visible*** aufgerufen, welche die Ansicht auf die dritte Ebene schaltet.

In der Methode ***To_Second_Handler*** wurden die Pfadangaben für die jeweiligen Videos hinzugefügt.

Damit das Video durch Klicken auf die Player-Anzeige pausiert bzw. wiedergegeben werden konnte, wurde vor die Player-Anzeige ein transparentes Panel gesetzt. Beim Eintreten des ***Click***-Ereignisses des Panels wurde die ***Play_Handler***-Methode aufgerufen.

Um die Übersichtlichkeit des Quellcodes zu bewahren, wurde die Initialisierung der Anzeige- und Steuerelemente der Ebenen in die Methoden ***CreateWindow_1***, ***CreateWindow_2*** und ***CreateWindow_3*** ausgelagert. Diese werden einmalig im Konstruktor aufgerufen.

5.2.3.9 Erweiterung der MyButton-Klasse

Der Play- und der Volume-Button benötigten noch weitere Texturen, da diese noch zusätzliches visuelles Feedback geben mussten. So sollte der Play-Button, nachdem er einmal geklickt wurde, die Funktion eines Pause-Buttons und die entsprechenden Texturen übernehmen. Der Volume-Button sollte, nach einmaligem Klicken, andere Texturen erhalten, welche symbolisieren würden, dass der Ton ausgeschaltet wurde.

Um das zu erreichen, wurde die MyButton-Klasse um weitere Felder erweitert. Dies waren zum einem ***sec_Stand_Image*** und ***sec_Highlight_Image*** vom Typ ***Image***. Zum zweiten war es das Feld ***secundary_Funktion*** vom Typ ***bool***.

Zusätzlich wurde ein neuer Konstruktor implementiert. Dieser nahm drei Übergabeparameter vom Typ ***string*** entgegen. Es waren zum einem die bereits aus den anderen Konstruktoren bekannten Namen der Texturen und die Beschreibung für den Tooltip, zum anderen war es ein

Name für die sekundären Texturen. Wie bereits bei den anderen Konstruktoren wurden die Texturen für die sekundäre Funktion des Buttons aus Projektordnern eingelesen und den beiden neuen Feldern *sec_Stand_Image* und *sec_Highlight_Image* übergeben.

Damit die Funktion des Buttons zur Laufzeit geändert werden konnte, wurde die Eigenschaft *SecundaryFunktion* implementiert. Diese Eigenschaft veränderte das *secundary_Funktion*-Feld oder gab seinen Zustand zurück. Im set-accessor wurde zusätzlich die *Invalidate*-Methode aufgerufen, wodurch die Oberfläche des Buttons für ungültig erklärte und das Neuzeichnen des Buttons veranlasste wurde.

Damit die richtigen Texturen dargestellt würden, wurde die *OnPaint*-Methode um eine weitere Fallunterscheidung erweitert. Es wurde abgefragt, ob das Feld *secundary_Funktion* auf *true* oder *false* gesetzt war. Je nach dem wurden entweder die Texturen für die primäre Funktion (Play-Button, Volume-Button) oder die Texturen für die sekundäre Funktion (Pause-Button, VolumeOff-Button) gezeichnet.

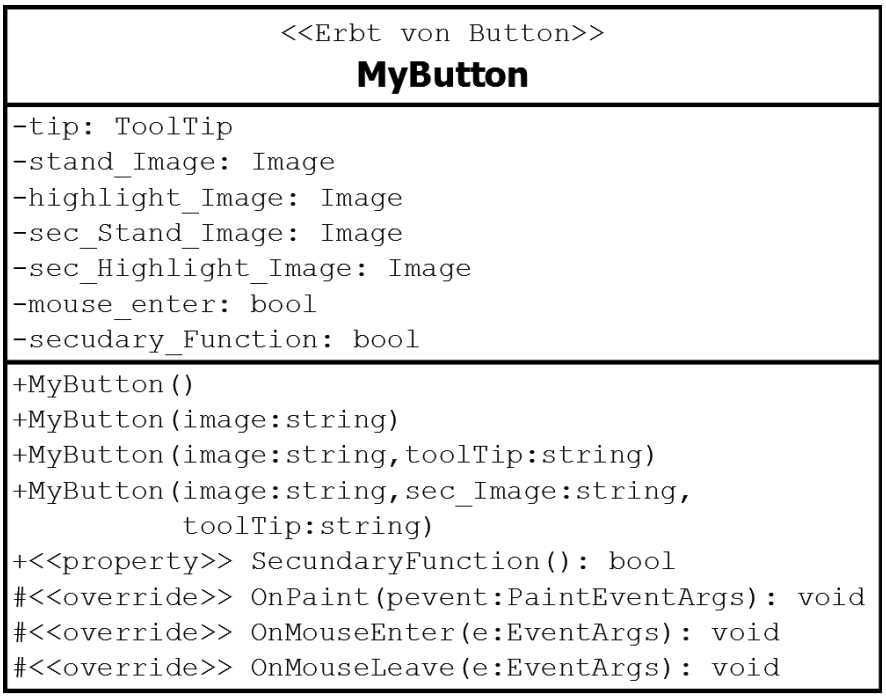


Abbildung 5.47: UML-Diagramm der Klasse MyButton, nach Erweiterung

5.2.3.10 Tastatureingaben

Die Steuerung des Players sollte nicht nur durch Bedienen der Steuerelemente, sondern auch durch Tastatureingaben erfolgen können.

Um auf Tastatureingaben in der Anwendung reagieren zu können, wurde die *KeyPreview*-Eigenschaft der TPHC-Klasse auf *true* gesetzt und die Methode *KeyDown_Handler* implementiert. Diese wurde aufgerufen, sobald das *KeyDown*-Ereignis ausgelöst wurde. In dieser fanden Fallunterscheidungen statt. In Abhängigkeit der gedrückten Taste wurden entsprechende Verarbeitungsketten in Gang gesetzt. Beim Drücken der Leertaste oder Pfeiltaste-Oben wurde

die Methode *Play_Handler* aufgerufen. Beim Drücken der S-Taste oder Pfeiltaste-Unten wurde die Methode *Stop_Handler* aufgerufen. Die Links- und Rechts-Pfeiltasten setzten die *Position*-Eigenschaft des Players 5 Sekunden vor bzw. 5 Sekunden zurück. Beim Drücken der X-Taste wurde die Anwendung beendet.

Beim Ausführen der Anwendung wurde jedoch festgestellt, dass die Pfeiltasten das *KeyDown*-Ereignis nicht auslösten. Die Eingaben der Pfeiltasten wurden vom Betriebssystem abgefangen, bevor diese bei der Anwendung ankamen. Um das zu umgehen, musste die Methode *ProcessDialogKey* überschrieben werden. In dieser wurde nun lediglich der boolesche Wert *false* zurückgegeben, wodurch die Pfeiltasten nun das *KeyDown*-Ereignis korrekt auslösten und die Eingaben verarbeitet werden konnten.

Zum Schluss wurde eine Erläuterung der möglichen Tastatureingaben der InfoForm hinzugefügt.

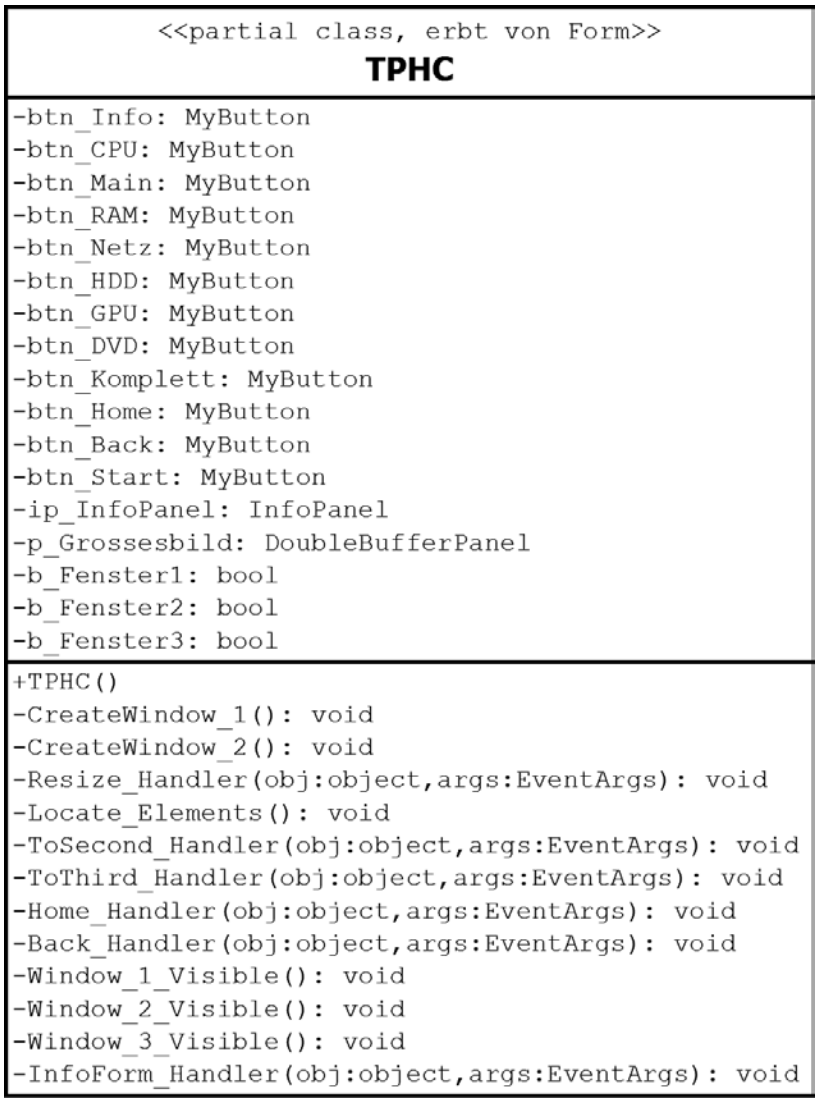


Abbildung 5.48: UML-Diagramm der TPHC-Datei

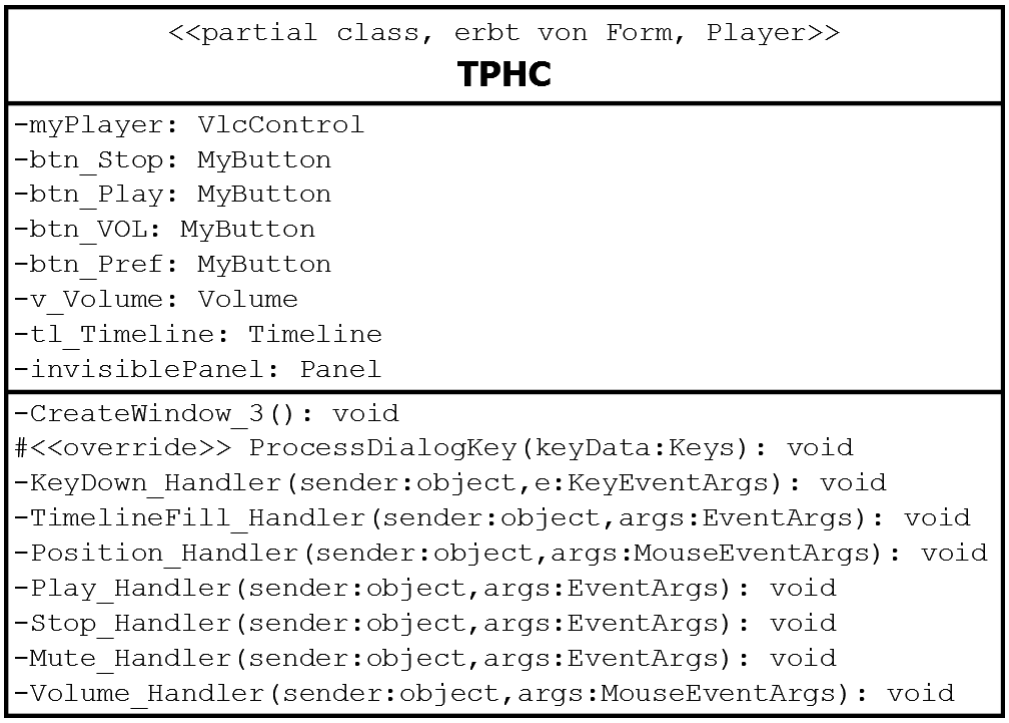


Abbildung 5.49: UML-Diagramm der TPHC Player-Datei

5.2.4 Userinterface-Design

Im Kapitel 4.2.2 wurde bereits die Anordnung der Anzeige- und Steuerelemente, an denen sich orientiert werden sollte, besprochen. Einige der folgenden Design-Entwürfe stammten vor der Konkretisierung des Interface-Aufbaus, wodurch die Anzahl oder die Anordnung nicht übereinstimmten. Das war jedoch nicht weiter wichtig, da es beim Design vor allem um den Eindruck der farblichen Gestaltung ging.

Die Entwürfe wurden mit *Adobe Illustrator CS4* erstellt und als Bilddateien exportiert, welche später als Texturen in die TPHC-Anwendung eingefügt wurden.

Nach der Erstellung eines Entwurfs wurde jeweils Kritik zu diesem eingeholt. Dies geschah in der Vorlesung Userinterface-Design.

5.2.4.1 Version 1

Beim ersten Entwurf bestand der Hintergrund aus einer planen hellgrauen Fläche, damit er für den Betrachter nicht zu sehr auffällt. Die Buttons bestanden aus Rechtecken mit abgerundeten Ecken. Der Rahmen der Buttons war Schwarz und die Fläche dunkelgrau, um sich vom hellgrauen Hintergrund abzuheben. Die Buttons wurden mit Symbolen versehen, die sich an den charakteristischen Merkmalen des jeweiligen Bauteils orientierten. Zusätzlich wurde der Name des jeweiligen Bauteils unter die Symbole gesetzt. Beim Bewegen des Mauszeigers über einen Button sollte sich der schwarze Rahmen in Weiß verändern.

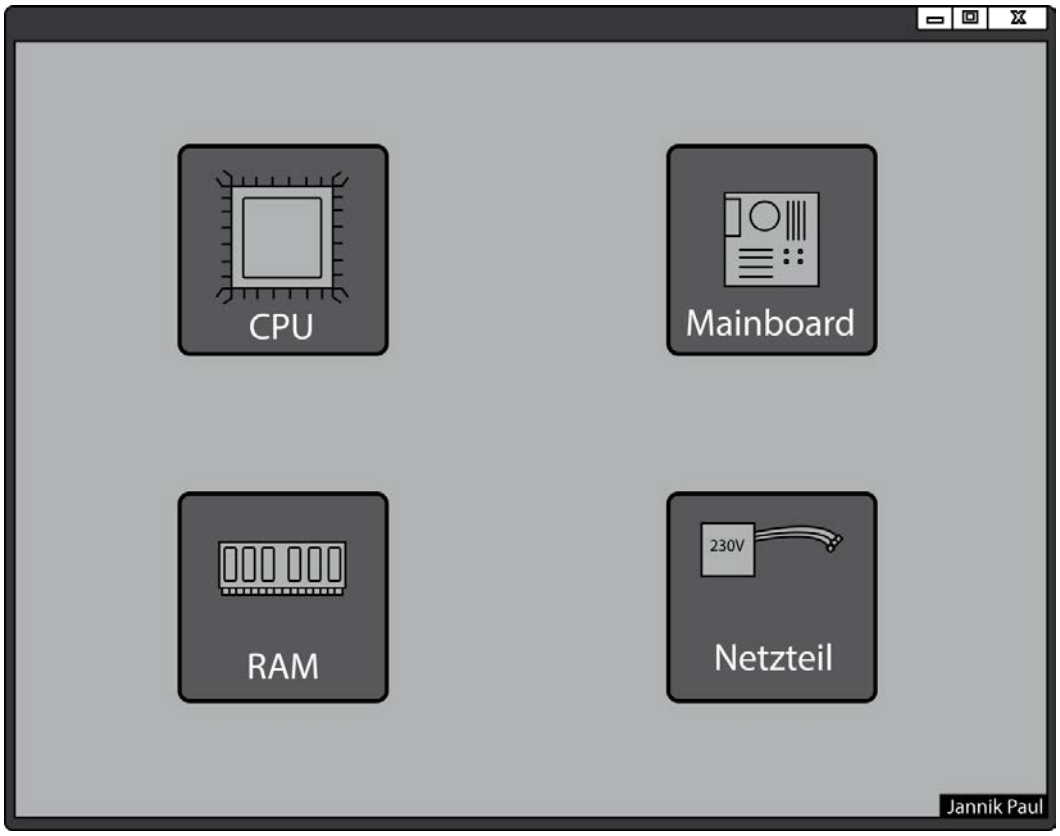


Abbildung 5.50: Erster Entwurf des Auswahlfensters

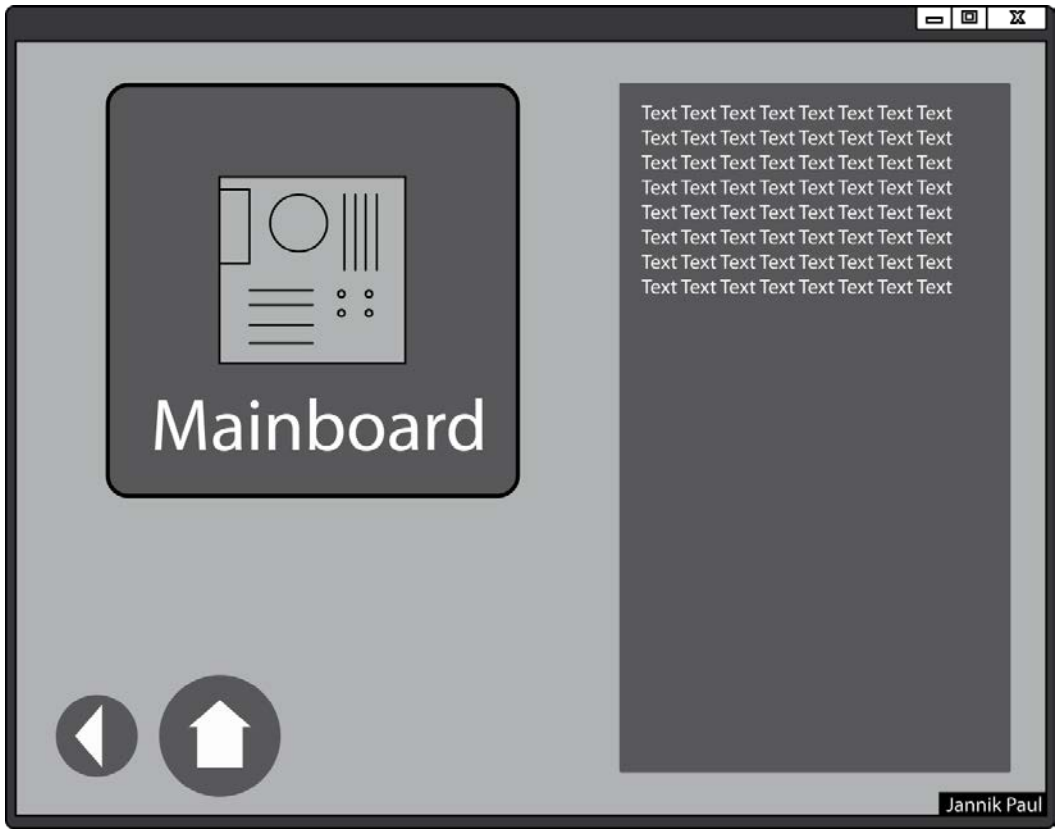


Abbildung 5.51: Erster Entwurf des Infofensters

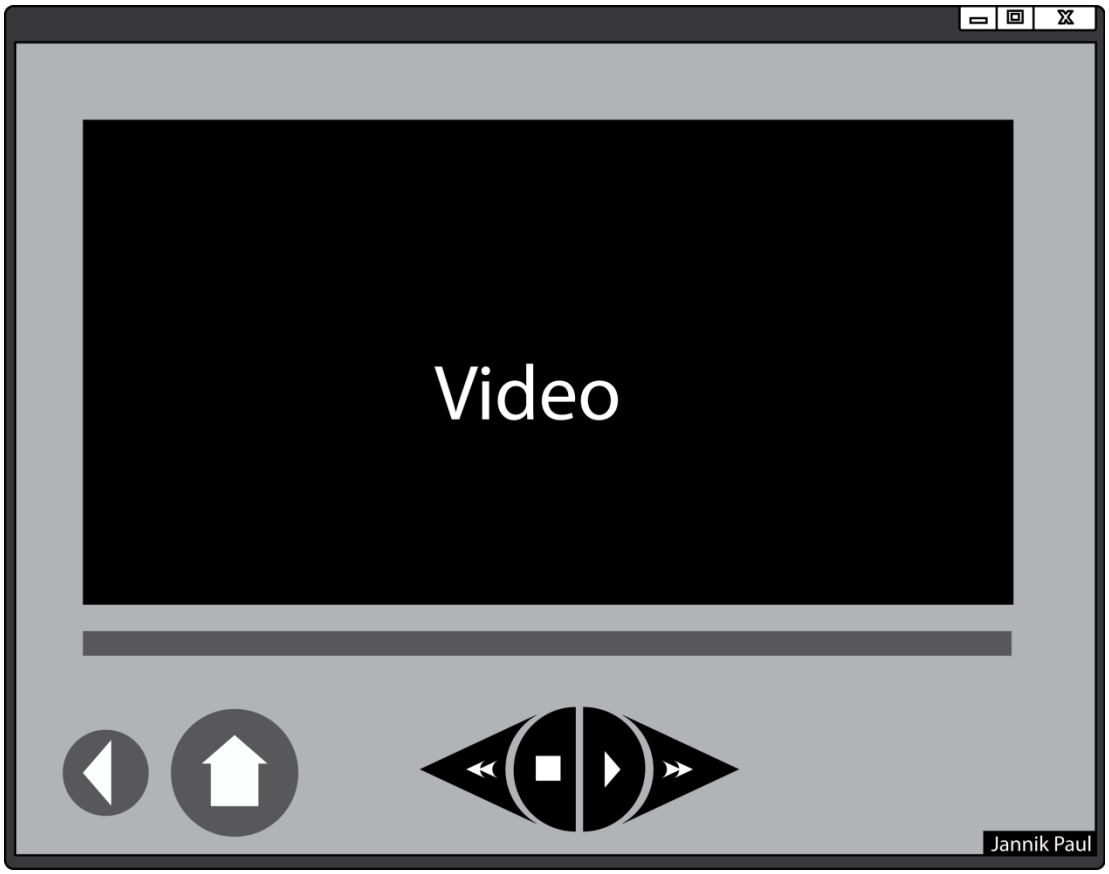


Abbildung 5.52: Erster Entwurf des Videofensters

Kritisiert wurden an diesem ersten Entwurf vor allem die Größe und Form der Steuerelemente des Players und die des Home- und Zurück-Buttons. Außerdem wirkte der Entwurf trist und eintönig, da die Buttons und der Hintergrund lediglich aus verschiedenen Graustufen bestanden.

5.2.4.2 Version 2

Beim zweiten Entwurf sollte vor allem auf den Kritikpunkt der Größe und Form der Player-Steuerelemente eingegangen werden. So wurden diese verkleinert und die Form in Kreise verschiedener Größe, geändert. Die Symbole wurden mit mehr Details versehen.

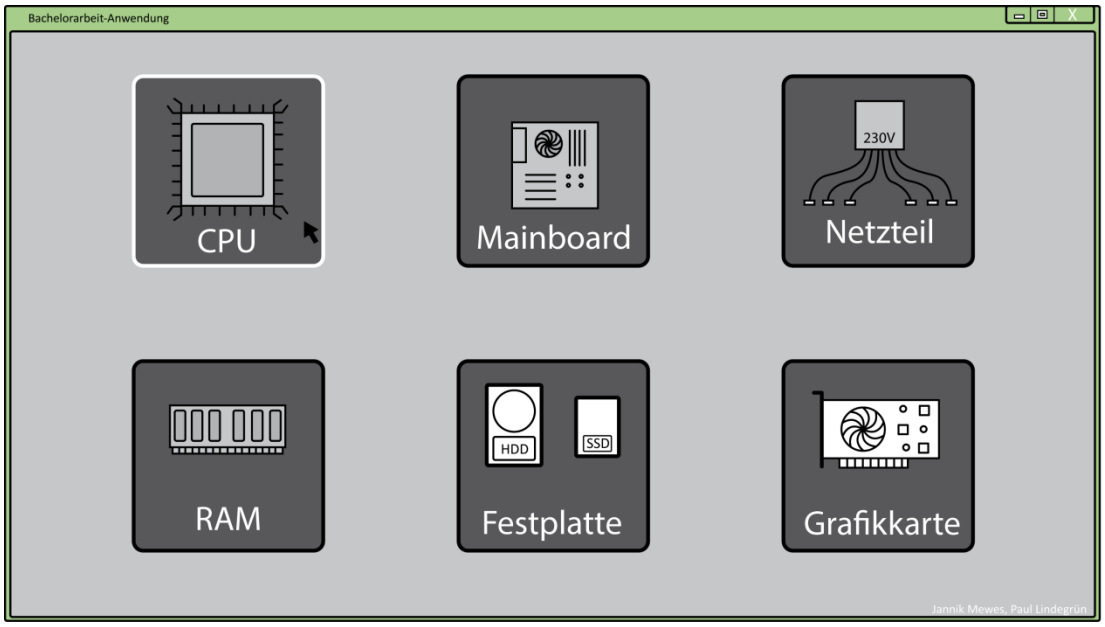


Abbildung 5.53: Zweiter Entwurf des Auswahlfensters

Abbildung 5.54: Zweiter Entwurf des Infofensters

Abbildung 5.55: Zweiter Entwurf des Videofensters

Weiterhin wurde an diesem Entwurf die Farbgebung kritisiert, da sich diese zum vorherigen Entwurf nicht geändert hatte.

5.2.4.3 Version 3

Beim dritten Entwurf wurde vor allem die Farbgebung verändert. So wurde die Hintergrundfarbe zu einem blauen Verlauf geändert. Die Buttons selber wurden jetzt zu Rechtecken ohne Rundungen oder Rahmen geformt und mit einer grasgrünen Farbe ausgefüllt. Weiterhin sollte ein weißer Rahmen eingeblendet werden, sobald sich der Mauszeiger über dem Button befand. Die Symbole wurden weiter ausgearbeitet und herausstechende Merkmale mit weißer Farbe gekennzeichnet.

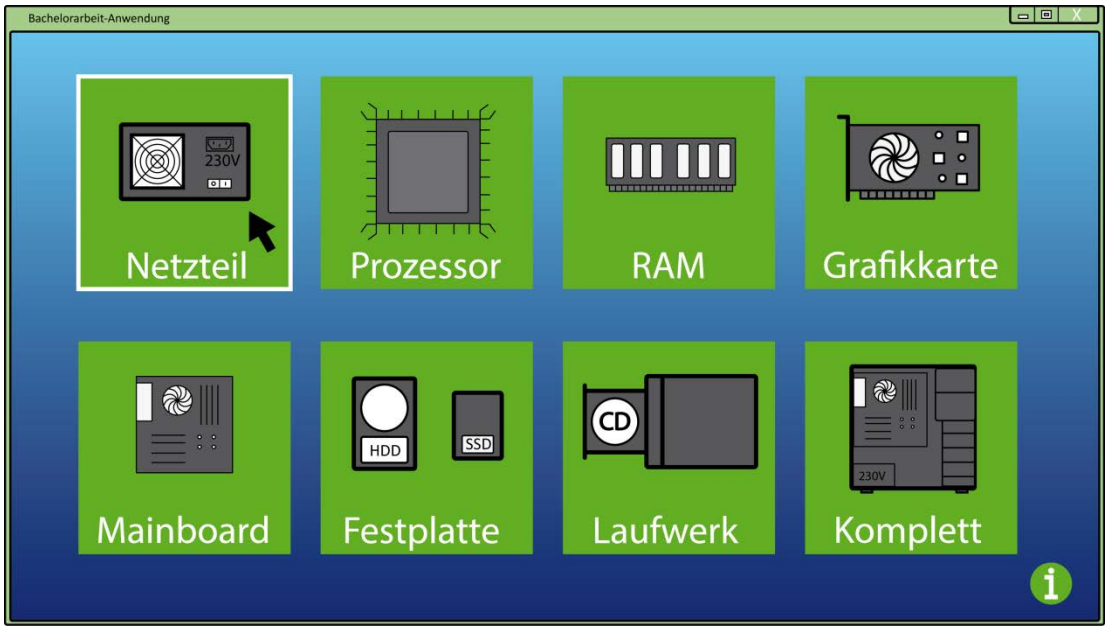


Abbildung 5.56: Dritter Entwurf des Auswahlfensters

Abbildung 5.57: Dritter Entwurf des Infofensters

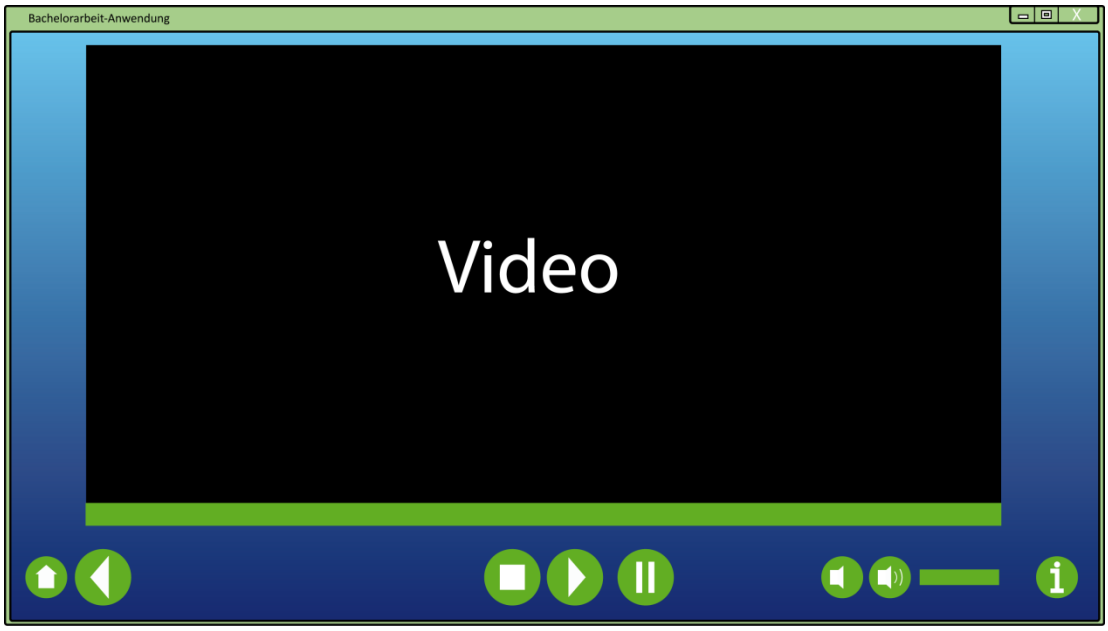


Abbildung 5.58: Dritter Entwurf des Videofensters

Kritisiert wurde bei diesem Entwurf die Wahl der Hintergrundfarbe. Der Kontrast zwischen Buttons und Hintergrund war vor allem im oberen Teil des Fensters, wo die Farbe heller war, nicht mehr gut zu erkennen. Positive Kritik gab es zu den Symbolen auf den Buttons. Diese waren gut zu erkennen und deutlich zu identifizieren.

5.2.5 Zusammenfassung des Kapitels

In diesem Kapitel wurden die implementierten Klassen des Programms sowie deren Funktionsweisen beschrieben. Außerdem wurden die Entwürfe für das Userinterface vorgestellt und evaluiert.

Das nächste Kapitel befasst sich mit dem Testen des Programms.

5.3 Implementierung der Einzelergebnisse

Wie bereits erläutert, sollen die von den beiden Studenten erstellten Interface-Designs zu einem Design vereint werden.

Als erstes wurden die Vor- und Nachteile der beiden fertigen Designs gegenübergestellt. Die folgende Tabelle zeigt Vor- und Nachteile auf:

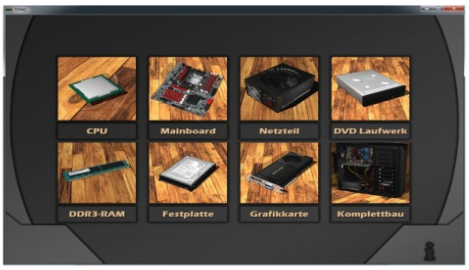
	J. Mewes	P. Lindegrün
Vorteile	- Buttons klar erkennbar (Rahmen) - Keine Farbablenkungen vom Video (Hintergrund) - 3D-Modelle im Menü sichtbar	- Symbole (Icons) im Auswahlbildschirm - Farbreicher (fröhlicherer Eindruck)

Nachteile:	- Grau in Grau - Schatten zu stark - Buttons zu groß	- Zu wenig Kontrast zwischen Buttons und Hintergrund - Farben schmerzten im Auge - Flat-Design wirkt nicht hochwertig

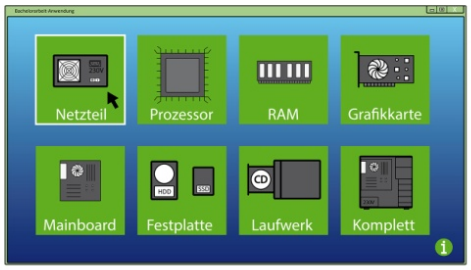
Nach der Kombination aus den Vorteilen der beiden Designs entstand das folgende Userinterface-Design:

Im Ergebnis war der Hintergrund dunkelgrau mit einem sehr leichten Verlauf, auf dem sich ein dünnes schwarzes Karo-Muster erstreckt. Die Buttons wurden allgemein mit einem hellgrünen Rahmen versehen und mit schwarz ausgefüllt.

Finaler Entwurf (Jannik Mewes)



Finaler Entwurf (Paul Lindegrün)



Kombination aus Einzelergebnissen

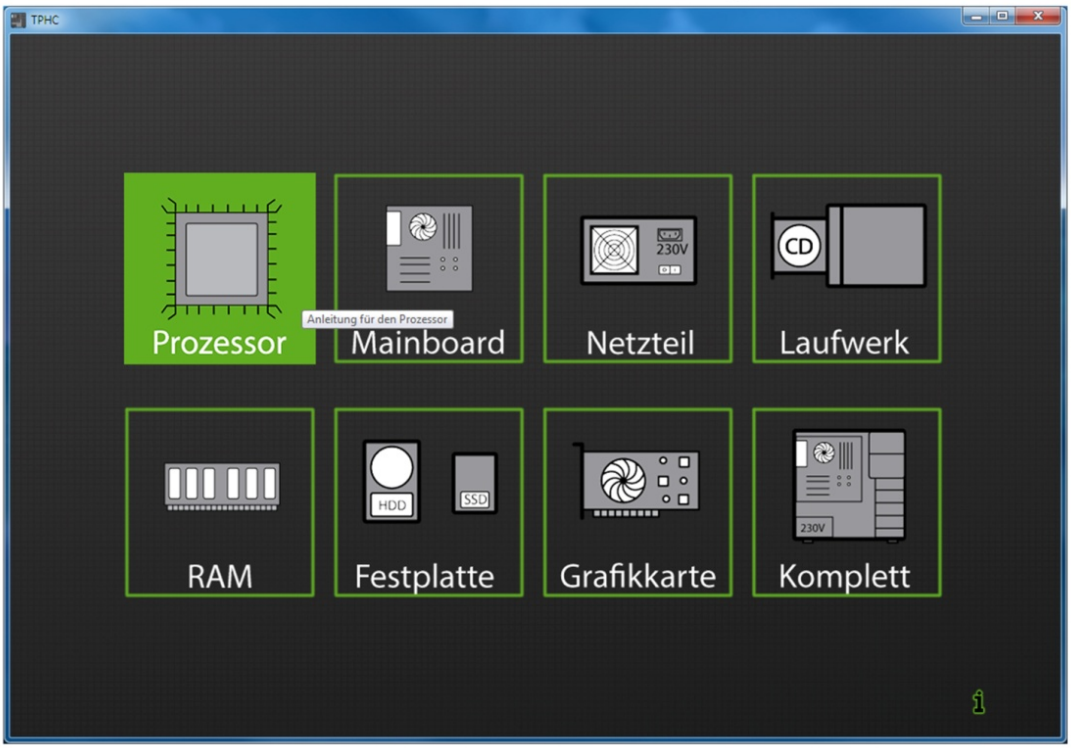


Abbildung 5.59: Finaler Entwurf des Auswahlfensters

Im 1. Fenster wurden die Icons übernommen, die einen durchsichtigen Hintergrund mit einem weichen Rahmen in einer hellgrünen Farbe bekamen. Dadurch sind die Button visuell auffällig, jedoch nicht zu grell oder bunt. Die Icons zeigen vereinfachte Darstellungen der einzelnen Komponenten. Sie sind für den Anwender auf den ersten Blick einfacher zu erkennen als die detaillierten Darstellungen der 3D-Modelle. Beim Bewegen des Mauszeigers über einen Button wird dieser mit der hellgrünen Farbe ausgefüllt. Somit werden sie untermalt und dadurch kann der Anwender erkennen, dass es sich hierbei um ein aktives Steuerelement handelt.

Abbildung 5.60: Finaler Entwurf des Infofensters

Im 2. Fenster wurde eine Großansicht der detaillierten 3D-Modelle gewählt, womit der Anwender schrittweise an das Design der Animation herangeführt wird. So erkennt er schneller im Video, welches 3D-Modell welches Bauelement darstellt. Neben dem Bild wurden die Informationen für das Bauteil in beige und dem Font "Times New Roman" eingefügt. Die Farbe Beige hebt sich gut vom Hintergrund ab, ohne den User dabei zu blenden.

Abbildung 5.61: Finaler Entwurf des Videofensters

Im 3. Fenster wurde das Video über einer grünen Timeline positioniert. Die Timeline wurde durchsichtig mit einem hellgrünen Rahmen und hellgrünen Füllbalken gestaltet. Dadurch erkennt man leichter, dass es sich um eine Timeline handelt. Auch stören die Steuerelemente nicht durch ihre Farbgebung beim Betrachten des Videos.

6 Testphase

6.1 Jannik Mewes

In diesem Kapitel sollen die Videos sowie das gemeinsam mit P. Lindegrün erstelle Userinterface mit einem Probandentest auf Funktionalität, Eindruck und Verständnis überprüft werden.

6.1.1 Probandentest

Im Probandentest wurde ein Übungs-PC auf einen Tisch sowie ein Notebook daneben gestellt. Auf dem Notebook wurde das Programm kopiert und eine Verknüpfung auf dem Desktop erstellt. Die Probanden wurden einzeln nacheinander getestet, ohne dass sie vorher Einblick hatten, was der Proband zuvor getan hatte.

Dem Probanden wurde ein Bauteil (z.B. eine Grafikkarte) sowie ein passender Schraubendreher ausgehändigt. Die Aufgabe war folgende:

"Stellen Sie sich vor, Sie hätten eine Grafikkarte im Internet gekauft und hätten dann diese Platine zugeschickt bekommen. Nun wollen Sie sie einbauen, ohne dafür in einem Fachgeschäft bezahlen zu müssen. Sie haben durch eine Internetrecherche dieses Programm gefunden und kostenlos heruntergeladen."

Nachdem der Proband die Aufgabenstellung verstanden hatte, wurde Ihm noch die Verknüpfung auf dem Desktop gezeigt. Dann gab es von Seiten der Prüfer keine Hilfestellungen mehr. Der Prüfer schaute dem Probanden aufmerksam zu, ob er sich im Userinterface zurecht fand und den Hardwareumbau vollziehen konnte.

Nach dem Test bekam der Proband einen Fragebogen, den er/sie ausfüllen musste. Nach dem Ausfüllen wurde der Proband entlassen und der Prüfer notierte, was ihm aufgefallen war.

Abbildung 6.1: Fragebogen

Insgesamt wurde der Test mit 19 Probanden durchgeführt. Die Probanden hatten unterschiedliche Vorkenntnisse und kamen aus verschiedenen Berufsgruppen. Die Durchschnittsergebnisse werden im Folgenden aufgelistet (Ergebnisse gerundet auf eine Kommastelle):

Frage	Probanden	Prüfer
Ich habe mich im Menü zurecht gefunden	Ø 4,5	Ø 4,1
Mir gefällt das Design	Ø 4,5	Ø --
Ich konnte den Hardware-Umbau bewerkstelligen	Ø 4,2	Ø 4,6
Ich hatte zuvor schon einmal PC Hardware ausgetauscht	Ø 1,3	Ø --
Ich habe Angst davor, elektronische Bauteile anzufassen	Ø 2,4	Ø 2,0
Ich benutze vorwiegend einen PC (keinen Mac oder DELL)	Ø 4,8	Ø --
Ich habe die Videos gut verstanden	Ø 4,3	Ø 4,6
Ich musste das Video öfter ansehen	Ø 2,4	Ø 3,2
Ich fand die Bedienoberfläche ansprechend	Ø 4,5	Ø --
Ich habe mir die Informationen im 2. Fenster durchgelesen	Ø 2,5	Ø 1,5
Ich fand die Informationen im 2. Fenster hilfreich beim Umbau	Ø 3,3	Ø --
Ich konnte etwas durch das Programm lernen	Ø 4,7	Ø --
Ich werde den Umbau in Zukunft auch ohne das Programm hinbekommen	Ø 4,1	Ø 3,8
Das Programm lief flüssig und ohne Abstürze	Ø 4,9	Ø 4,9
Ich würde das Programm weiterempfehlen	Ø 4,4	Ø --
Ich besitze Kenntnisse über PC-Hardware	Ø 1,6	Ø 2,0
Wie oft haben Sie das Anleitungsvideo angeschaut?	Ø 3,2	Ø 3,2

6.1.1.1 Resultat Videoverständnis

In manchen Videos fielen kleinere Schwachstellen auf. So gab es Probleme, den Grafikkartenhebel zu finden, da dieser beim Beispiel-PC etwas anders aussah. Deshalb wurde ein Vermerk im Infotext hinzugefügt, dass sich die Hebel im Aussehen unterscheiden können. Zudem wurde im Video ein besonderer Stromanschluss eingebaut, der nicht bei allen Grafikkarten vorhanden ist. Zwar war im Text diese Information vorhanden, wurde aber oft von den Probanden übergangen. Um dem entgegenzuwirken, wurde diese Information im Text deutlicher hervorgehoben.

Im Mainbordvideo trat ein ähnliches Problem bei den Stromanschlüssen, wie bei dem Grafikkartenhebel auf. Das leicht unterschiedliche Design verwirrte. Zudem wurde das Mainboard von zwei Probanden so stark festgeschraubt, dass es Beschädigungen an der Platine gab. Auch hierfür wurde nachträglich eine Zusatzinformation im Infotext geschrieben.

Bei allen Videos, in denen die Gehäuse Klappe entfernt wird, sollte zusätzlich ein Pfeil eingeblendet werden, der die Animation unterstützt. Der Pfeil würde deutlich anzeigen, dass die Klappe zuerst nach hinten und erst dann zur Seite geschoben wird.

Alles im allen wurden die Videos sehr gut verstanden und es gab sehr wenig Verständnisprobleme. Die Testpersonen hatten im Allgemeinen keine Erfahrungen im Hardwareumbau und waren zufrieden und überrascht, dass sie mit Hilfe des Videos den Hardwareumtausch leicht bewerkstelligen konnten. Die meisten Probanden fanden die visuelle Anleitung sehr viel angenehmer als textgestützte Unterweisungen und meinten, den Umbau in Zukunft auch selbstständig meistern zu können.

6.1.1.2 Resultat Userinterface

Die Tabelle der Resultate zeigt kleinere Schwächen beim Userinterface auf. Zwar sind die Probanden gut durch das Menü gekommen und fanden schnell die benötigten Informationen, jedoch hakelte es manchmal an der Bedienung. Vor allem wurde oft der *Zurück*- Button gedrückt, der dazu dient zum vorherigen Fenster zurückzukehren, um das Video ein Stück zurückzuspulen. Um dem Problem entgegenzuwirken, wurde die Form des Pfeiles für ein besseres Verständnis geändert.

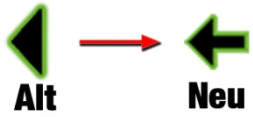


Abbildung 6.2: Veränderung des Zurück-Pfeiles

Auch zeigte sich, dass der Informationstext für die Probanden eher abschreckend war und deshalb gar nicht erst gelesen wurde. Nach Rücksprache stellte sich heraus, dass dies vor allem an der Länge und Fachsprache des Textes lag. Daraufhin wurden die Texte stark reduziert und mehr Veranschaulichungsbilder eingefügt. Des Weiteren könnte man die Informationen noch einmal an den passenden Stellen in den Videos einblenden.

Ein Wunsch der meisten Probanden war es ebenfalls, dass das Video im 3. Fenster bereits startet, wenn das 3. Fenster geöffnet wird, ohne vorher den *Play*-Button gedrückt zu haben. Auch dieses wurde korrigiert.

Das Design des Userinterfaces wurde allgemein als angenehm empfunden.

6.1.2 Zusammenfassung des Kapitels

In diesem Kapitel wurde aufgezeigt, wie die Videos und das Userinterface-Design anhand von Probandentests überprüft wurden.

6.2 Paul Lindegrün

Im vorherigen Kapitel wurde beschrieben, wie das Programm implementiert wurde.

In diesem Kapitel wird das Programm auf seine Stabilität hin getestet. Außerdem wird beschrieben, ob und wie aufgetretene Probleme behoben wurden.

6.2.1 CPU-Auslastung

Die CPU-Auslastung durch die Anwendung wurde auf Systemen mit unterschiedlichen CPUs durchgeführt. Die Auslastung wurde mit Hilfe des Task-Managers von *Windows* gemessen.

Beim ersten System war der Prozessor Core 2 Quad Q6600, mit einer Taktrate von 2.4 GHz, von *Intel* verbaut. Beim Bedienen der Anwendung lag die Auslastung im Durchschnitt bei 6-8%. Während des Abspielens eines Videos stieg diese auf 20-30% an.

Beim zweiten System handelte es sich um ein Notebook. Darin war der Pentium Dual CPU T2390, mit einer Taktrate von 1,86 GHz von *Intel*, verbaut. Beim Bedienen der Anwendung lag die Auslastung im Durchschnitt bei 10-13%. Während des Abspielens eines Videos stieg diese auf 30-45% an.

6.2.2 Speicher-Auslastung

Die Messung des Speicherverbrauchs durch die Anwendung wurde mit Hilfe des Task-Managers von *Windows* durchgeführt.

Bei der Messung stellte sich heraus, dass beim Bedienen der Anwendung der Speicherverbrauch kontinuierlich anstieg. So erhöhte sich der Verbrauch beim Bewegen des Mauszeigers über einen Button um die Dateigröße der Textur. Daher wurde vermutet, dass beim Zeichnen der Texturen in der MyButton-Klasse Speicherplatzt beim Aufruf der Methoden reserviert, jedoch nach dem Beenden nicht mehr freigegeben wurde. Um das Freigeben sicherzustellen, wurden am Ende jedes Anweisungsblocks die benutzten Objekte durch Aufrufen der ***Dispose***-Methode wieder freigeben. Das brachte jedoch keine Verbesserung. Erst durch manuelles Aufrufen des Garbage Collectors blieb der Speicherverbrauch konstant.

Beim Bedienen der Anwendung lag die Speicherauslastung zwischen 40 und 48 MB. Beim Abspielen eines Videos stieg diese teilweise auf über 100 MB (abhängig vom Video).

6.2.3 Stabilität des Programmes

Um die Stabilität des Programms zu testen, wurden immer wieder die Steuerelemente und die Videos der Reihe nach bedient bzw. abgespielt. Dabei stellte sich heraus, dass das Programm beim Stoppen eines Videos gelegentlich abstürzte. Es wurde vermutet, dass der Player die Videos nicht schnell genug freigeben konnte, bevor die Verarbeitung im Programm fortgesetzt wurde. Um dies zu überprüfen, wurde nach dem Aufruf der ***Stop***-Methode des Players die Weiterverarbeitung der Anwendung für 100 ms pausiert und wieder getestet. Das brachte jedoch keine Verbesserung. Auch eine Erhöhung der Zeit blieb wirkungslos. Dieses Problem konnte bis zum Schluss nicht gelöst werden.

Während des Probandentests stürzte das Programm einmal ab.

6.2.4 Starten unter verschiedenen Windows Versionen

Um die Kompatibilität des Programms mit verschiedenen *Windows*-Versionen zu überprüfen, wurde es unter *Windows XP*, *Windows Vista* und *Windows 7* gestartet und getestet. Wenn die Fensteranwendung beim Abspielen eines Videos unter *Windows XP* minimiert und maximiert wurde, wurde das Anzeigeelement des Players verzerrt, wodurch nur noch ein Teil des Videos zu sehen war. Es ergaben sich keine weiteren Probleme oder Einschränkungen durch die verschiedenen *Windows*-Versionen.

6.2.5 Helligkeits-Kontrast-Sättigungs-Einstellungen

Beim Testen des Programms auf verschiedenen Systemen wurde festgestellt, dass die Bildschirme nicht gleich kalibriert waren, wodurch das angezeigte Video zu dunkel oder zu hell erschien. Daher wurde beschlossen, eine Form für Helligkeits-, Kontrast- und Sättigungs-Einstellungen zu implementieren.

Dazu wurde die Klasse PreferencesForm, welche von der Form-Klasse erbt, erstellt. Diese besaß 7 Label-Elemente, 3 Volume-Elemente und 2 MyButton-Elemente. Die Volume-Elemente dienten zum Einstellen der Helligkeit, des Kontrastes und der Sättigung. Drei Labels dienten zur Beschriftung der Volume-Elemente und drei dienten zum Anzeigen der eingestellten Werte. Ein Label diente zum Anzeigen einer Aufforderung, das Programm neu zu starten, da Änderungen nur bei der Initialisierung des Players übernommen wurden. Der erste Button diente zur Bestätigung der Änderungen, wobei diese in eine Textdatei geschrieben wurden. Der zweite Button setzte die Einstellungen auf Standardwerte. Um Änderungen der Volume-Elemente auslesen zu können, wurden die Methoden *Volume1Handler*, *Volume2Handler* und *Volume3Handler* implementiert. Um die Änderungen anzeigen zu können wurde die Methode *UpdateInterface* implementiert.

Auf der Player-Ebene des Programms wurde ein weiterer Button hinzugefügt, der das Anzeigen des PreferencesForm-Objekts veranlasst.

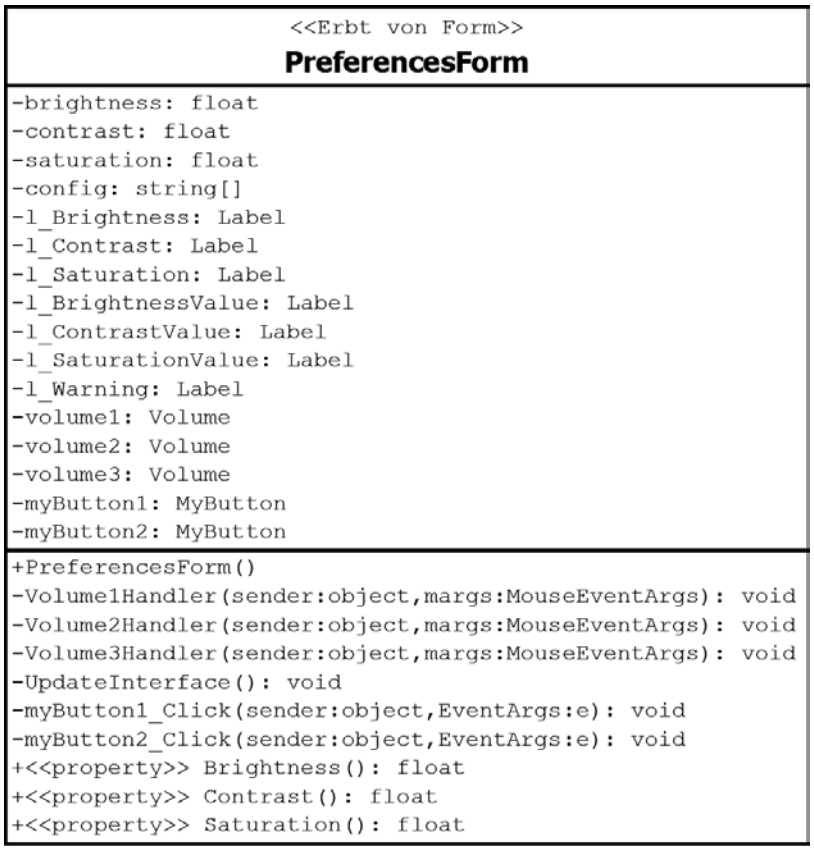


Abbildung 6.3: UML-Diagramm der Klasse PreferencesForm

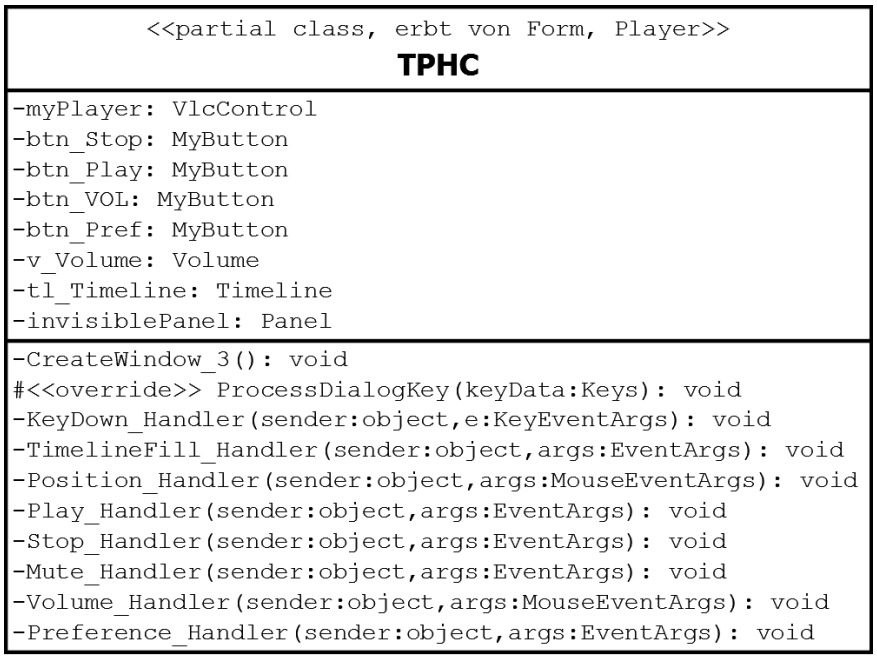


Abbildung 6.4: UML-Diagramm der TPHC Player-Datei, nach Erweiterung

6.2.6 Zusammenfassung des Kapitels

In diesem Kapitel wurde die CPU- und Arbeitsspeicher-Auslastung festgehalten sowie die Stabilität des Programms getestet. Des Weiteren wurde beschrieben, wie aufgetretene Probleme gelöst wurden.

Das nächste Kapitel fasst die Ergebnisse dieser Arbeit zusammen und gibt Ausblicke für mögliche zukünftige Erweiterungen des Programms.

7 Zusammenfassung

7.1 Jannik Mewes

7.1.1 Ergebnis

Das Ziel der Bachelorarbeit und des Praxisprojekts war es, einfach zu verstehende Videoanleitungen für einen PC-Umbau mit 3D-Animationen zu erstellen. Dafür sollte ein Computer, der der HAWK-Göttingen zur Verfügung steht, möglichst detailgetreu in 3D modelliert und anschließend animiert werden. Diese Animationen sollten nach dem Rendern mit einer Videobearbeitung verfeinert und anschließend an das entwickelte Programm von P. Lindegrün angefügt werden. Parallel sollte ein ansprechendes Userinterface in Zusammenarbeit entstehen. Dieses Ziel wurde vollständig erfüllt und sogar noch um einige Aspekte erweitert.

An kleineren Stellen sind allerdings noch Verbesserungen möglich.

So ist z. B. die Videodateigröße noch zu optimieren. Des Weiteren könnten noch weitere Videoeffekte in der Anleitung selbst erscheinen, um den Vorgang noch mehr zu verdeutlichen.

Ebenfalls könnten die Render-Einstellungen optimiert werden, um visuell bessere Schatten- und Lichtverläufe zu generieren. Diese weisen bei den jetzigen Videos teilweise noch Komprimierungsartefakte auf.

Auch würden die Videos einen besseren Eindruck beim User erzielen, wenn noch Soundeffekte sowie ein Hintergrund-Rhythmus dazu gemischt werden, da die Videos tonlos etwas "kahl" wirken.

Trotz dieser kleineren Verbesserungsmöglichkeiten erfüllt das Projekt die gestellten Anforderungen vollständig und die Funktionalität erwies sich durch die Probandentests, die sehr erfolgreich waren.

7.1.2 Ausblick

Bereits vor Beginn des Projekts wurde eingeplant, dass das fertige Produkt erweiterbar ist.

So können z. B. noch weitere Bauteile modelliert und in die Animationen eingebunden werden. Auf diesem Weg würden die im Probandentest festgestellten kleineren Probleme ausgeräumt werden können. Wenn man z. B. von jedem Mainboard-Typ ein entsprechendes modelliert, könnte sich der User im Menü das passende raussuchen und es können keine Verwirrungen mehr über unterschiedlich aussehende Grafikkarten-Hebel oder Stromversorgungsstecker entstehen. Das gleiche gilt auch für alle anderen Einzelmodule.

Der Infotext im 2. Fenster könnte ebenfalls um detaillierte Informationen über die Bauteile erweitert werden. Hardwareproduktionsfirmen könnten ihre neuen Produkte als 3D-Modell einbinden und die entsprechenden Produktinformationen zur Verfügung stellen. Dadurch würde das Projekt um eine Art Kaufberatung erweitert. Der User könnte sich in dem Programm selbst einen eigenen PC zusammenstellen und nachsehen, ob die Bauteile zueinander passen (z. B. ob die erwünschte Grafikkarte in das ausgesuchte Gehäuse passt).

Grade für Gehäusehersteller könnte eine solche Erweiterung recht attraktiv sein, da sie auf diesem Weg Besonderheiten (z. B. Vorderklappe, Festplattenhalterung, Kabelleisten usw.) gut hervorheben können.

Wie auch schon im Kapitel 7.1.1 erwähnt, können ebenfalls Soundeffekte hinzugefügt werden.

Das Programm erklärt bisher nur den Ein- und Ausbau von PC-Bestandteilen. Andere Hersteller, wie z.B. Apple oder Dell, ließen sich ebenfalls in das Programm einbinden, um auch Usern dieser Produkte Anleitungen zur Verfügung zu stellen. Des Weiteren ist es für viele User ebenfalls interessant, wie man einen fertig gebauten PC richtig anschließt (Monitor, Maus, Tastatur, Strom etc.). Auch hierfür würden sich Videos mit 3D-Animationen erstellen lassen.

7.2 Paul Lindegrün

7.2.1 Ergebnis

Ziel dieser Bachelorarbeit war es, ein eigenständig lauffähiges Programm zu erstellen, welches den Anwender beim Ein- und Ausbau von PC-Komponenten unterstützt. Dies sollte durch ein einfaches und für den Anwender übersichtliches Interface und das Abspielen der vorgerenderten Videos erreicht werden.

Das Programm ist voll funktionsfähig. Die im Kapitel 2.2.2 besprochen Anforderungen, konnten erfüllt werden. Jedoch gibt es eine Reihe von Punkten, die verbessert werden könnten bzw. sollten. So werden die Anzeige- und Steuerelemente, beim Wechsel der Ebenen, teilweise sichtbar nacheinender eingeblendet. Hier sollte eine andere Lösung des Ebenenwechsels gefunden werden.

Ein weiteres Problem ist die Speicherverwaltung. Die Speicherbereinigung findet an einigen Stellen im Quellcode durch Aufruf des Garbage Collectors statt. Dies ist keine effiziente Lösung, da der Garbage Collector beim Arbeiten selbst einen nicht unerheblichen Teil der Systemressourcen für sich beansprucht. Hier sollte die Speicherverwaltung optimiert werden, sodass nicht mehr gebrauchte Objekte sofort freigegeben werden und der Aufruf des Garbage Collectors überflüssig wird.

Die Stabilität des Programms sollte verbessert werden. Dieses stürzt gelegentlich beim Aufruf der *Stop*-Methode des Players ab. Hierzu ist eine gründlichere Analyse der VideoLan DotNet-Bibliotheken notwendig.

Die Strukturierung des Quellcodes sollte weiter optimiert werden. So könnte der Player aus der TPHC-Klasse in eine eigene Klasse aufgelagert und vollständig gekapselt werden. Das Anzeigen der Infotexte in der InfoForm-Klasse kann weiter optimiert werden, da beim Neuzeichnen des Elements das Array jedes Mal nach Bildern durchsucht wird.

Die Anordnung der Anzeige- und Steuerelemente im Interface kann weiter verbessert werden. Vor allem in der zweiten Ebene gibt es Freiräume, die auf den Anwender störend wirken könnten.

Die Einstellungsmöglichkeiten für Helligkeit, Kontrast und Sättigung, könnten so implementiert werden, dass ein Programmneustart entfällt.

7.2.2 Ausblick

Das Programm kann in der Zukunft noch um weitere Funktionalitäten erweitert werden.

Die modellierten Bauteile und Animationen ließen sich unter Zuhilfenahme von OpenGL oder DirectX in Echtzeit auf der Grafikkarte berechnen. Dadurch würde das Rendern der Videos entfallen und verschiedene Bauteile flexibler in die Datenbank einbezogen werden.

Das Interface kann um Animationen erweitert werden, um einen flüssigeren Übergang zwischen den Ebenen zu ermöglichen und die Benutzerfreundlichkeit zu steigern.

Die Anwendung könnte mit Hilfe des “Mono-Project“ auch auf anderen Betriebssystemen lauffähig gemacht werden.

8 Literaturverzeichnis

8.1 Jannik Mewes

[1] Heinrich Hübscher, Hans-Joachim Petersen, Carsten Rathgeber, Klaus Richter und Dr. Dirk Scharf; IT-Handbuch; Westermann Verlag; 7.Auflage 2011; Seite 127-204; ISBN: 978-3-14-22 5024-6

[2] Scott Onstott; AutoCAD 2013 und AutoCAD LT 2013; Wiley-VCH Verlag GmbH & Co. KGaA; 1. Auflage; 11. Juli 2012; ISBN-10: 3527760288

[3] Carsten Wartmann; Das Blender-Buch: 3D-Grafik und Animation mit Blender 2.5; Dpunkt Verlag; Auflage: 4., akt. und erw. Neuaufl.; 29. August 2011; ISBN-10: 3898646106

[4] Randi L. Derakhshani; Autodesk 3ds-Max 2013 Autodesk Official Training Guide; Wiley VCH Verlag GmbH; 2012; ISBN-10: 3527760296

[5] Alexander „Nico“ Ostermann; Autodesk Maya 2013: 3D-Animation vom Concept zum Final; mitp Verlag; Auflage: 2012; ISBN-10: 382669208X

[6] Adobe Creative Team of designers; Adobe Premiere Pro CS6 Classroom in a Book; Adobe Press; Juli 2012; ISBN-10: 0321822471

[7] Dipl.-Math. Wilhelm Noack; Photoshop CS5 Grundlagen; 1. Auflage August 2010; Dieses Handbuch ist ein unveränderter Nachdruck einer Seminarunterlage des Herdt-Verlages

[8] Monika Gause; Adobe Illustrator CS6: Das umfassende Handbuch; Galileo Design; Auflage: 1 28. September 2012;ISBN-10: 3836218860

[9] Autodesk; Systemvoraussetzungen; 27.04.2013; URL: http://www.autodesk.de/adsk/servlet/pc/index?siteID=403786&id=14677668

[10] appleradar; Apple aktualisiert seine Computer – Neuer iMac, neuer Mac Pro; 03.05.2013; URL: http://www.appleradar.de/imac/apple-aktualisiert-seine-computer-%E2%80%93-neuer-imac-neuer-mac-pro/

[11] Claudioverfuerth; Wikipedia; Auge; Gesichtsfeld; 05.05.2013; URL: https://de.wikipedia.org/wiki/Auge#cite_note-AP-2

[12] Marco Chiappetta; PC clever aufrüsten - so machen Sie es richtig; PC-WELT; 14.04.2013; URL: http://www.pcwelt.de/ratgeber/Ratgeber-Computer-Tipps-fuer-Ihren-PC-Zusammenbau-6125735.html

[13] Beispiel einer perspektivischen Entzerrung; URL: http://www.archi-plan.ch/images/entzerr.jpg; 01.08.2013

8.2 Paul Lindegrün

[1] Bernhard Lahres, Gregor Rayman; Objektorientierte Programmierung; Galileo Computing Verlag; 2. Auflage 2009; ISBN-13: 978-3836214018

[2] Heiko Kalista; C++ für Spieleprogrammierer; Carl Hanser Verlag GmbH & Co. KG; 2. Auflage 2005; ISBN-13: 978-3446403321

[3] Christian Ullenboom; Java ist auch eine Insel; Galileo Computing Verlag; 10. Auflage 2011; ISBN-13: 978-3836218023

[4] Mono-Project; Compatibility; 01.05.2013; URL: http://www.mono-project.com/Compatibility

[5] Andreas Kuehnel; Visual C# 2008; Galileo Computing Verlag; 4. Auflage 2008; ISBN-13: 978-3836211727

[6] Andreas Kuehnel; Visual C# 2010; Galileo Computing Verlag; 5. Auflage 2010; ISBN-13: 978-3836215527

[7] NetBeans; 01.05.2013; URL: https://netbeans.org/

[8] Eclipse; 01.05.2013; URL: http://www.eclipse.org/

[9] MonoDevelop; 01.05.2013; URL: http://monodevelop.com/

[10] Eric Freeman, Elisabeth Freeman, Kathy Sierra und Bert Bates; Entwurfsmuster von Kopf bis Fuß; O'Reilly Verlag; 1. Auflage 2005; ISBN-13: 978-3897214217

[11] Das Encodingwissen; Die Videocodecs; 01.05.2013; URL: http://encodingwissen.de/formatedschungel/videocodecs

[12] Martin Fiedler; Videokompressionsverfahren von MPEG-1 bis H.264; Dream Chip Technologies GmbH; 01.05.2013; URL: http://keyj.s2000.ws/files/projects/videocomp.pdf

[13] Das Encodingwissen; Die Containerformate; 01.05.2013; URL: http://encodingwissen.de/formatedschungel/container

[14] Flash Video; 01.05.2013; URL: http://de.wikipedia.org/wiki/Flash_Video

[15] Using the Windows Media Player Control with Microsoft Visual Studio; 01.05.2013; URL: http://msdn.microsoft.com/en-us/library/windows/desktop/dd564585%28v=vs.85%29.aspx

[16] Media Player Classic – Home Cinema; 01.05.2013; URL: http://mpc-hc.org/

[17] VideoLan; 01.05.2013; URL: http://www.videolan.org/vlc/

[18] VideoLan DotNet for WinForm, WPF & Silverlight 5; 01.05.2013; URL: http://vlcdotnet.codeplex.com/

[19] Microsoft Developer Network, C# Programmierhandbuch; 01.05.2013; URL: http://msdn.microsoft.com/de-de/library/vstudio/67ef8sbd.aspx