DEATH TO MOBY DICK

Dokumentation zur Bachelorarbeit

von
Christoph Bülter
(Matr.-Nr.: 15197035)

Betreuung:
Prof. Dipl.-Anim. Peter Kaboth
Dr. Frank Lechtenberg

Hochschule Ostwestfalen-Lippe
Fachbereich Medienproduktion
Wintersemester 2010 / 2011

edienproduktion

Hochschule Ostwestfalen-Lippe
University of Applied Sciences

Inhaltsverzeichnis

1 Vorwort

Im Verlauf meines Studiums konnte ich mich vor allem für eine Fachrichtung begeistern: 3D-Grafik und -animation. 3D ist vielfältig und abwechslungsreich, technologisch faszinierend und sowohl technisch als auch kreativ anspruchsvoll. Schon im dritten Semester stand mein Entschluss daher fest, einen animierten Film als Bachelorarbeit anzufertigen. Da ich in diesem Bereich im Studium fast ausschließlich bei Gruppenprojekten beteiligt war, wollte ich meine Abschlussarbeit alleine bewältigen. Hierdurch erhoffte ich mir wertvolle Erfahrungen auch in Arbeitsbereichen, die ich zuvor kaum oder noch gar nicht erkundet hatte. Um innerhalb eines Semesters die Menge an Aufgaben bewältigen zu können, war eine Reduktion in der Zielsetzung unumgänglich. Nicht mehr als zwei animierte Charaktere und idealerweise nur ein Schauplatz waren die Bedingungen für meinen Film, der eine Gesamtlänge von ein bis zwei Minuten nicht überschreiten sollte. Die Themensuche erwies sich als längerer Prozess, bis ich ein passendes Szenario fand, das mich mit seiner vielschichtigen Thematik für sich begeistern konnte: „Moby Dick, or The White Whale“ von Herman Melville, veröffentlicht 1851 in den USA. In der konkreten Zielsetzung für den Film habe ich mich nach praktischen Gesichtspunkten orientiert und mich auf eine Momentaufnahme aus der Vorlage konzentriert. Die vorliegende Ausarbeitung dokumentiert die Produktion eines 3D-animierten Trailers für einen fiktiven Kinofilm. Über den genauen Ablauf des Projekts, die einzelnen Arbeitsschritte, aufgetretene Probleme und gefundene Lösungen, sowie meine persönlichen Erfahrungen möchte ich in dieser Arbeit Aufschluss geben. Ich habe, meinem persönlichen Interesse entsprechend, dem technischen Teil der Produktion eine stärkere Gewichtung zuteilwerden lassen. Die Kapitel Stoffsimulation, Charaktersetup und Skriptprogrammierung möchte ich daher ausführlicher behandeln. Ein persönliches Fazit schließt die Dokumentation ab.

2 Zielsetzung und Themenfindung

Da für die Erstellung meiner Bachelorarbeit keine einschränkenden Vorgaben existierten, hatte ich die Möglichkeit die Art und Thematik meiner Abschlussarbeit frei zu wählen. Hierfür war es hilfreich, vorher festzulegen was ich mit der Arbeit erreichen wollte. Einerseits wollte ich meine bisherigen 3D-Kenntnisse festigen und erweitern. Beispielsweise konnte ich in vorherigen Projekten gute Erfahrungen in der Erstellung animierter Charaktere machen, die ich gerne nutzen und erweitern wollte. Das macht nur Sinn in bewegten Bildern, sprich einem animierten Film. Andererseits möchte ich nach dem Studium beruflich mit 3D-Grafik und -Animation meinen Lebensunterhalt bestreiten können. Um eine entsprechende Anstellung zu finden, ist der übliche Weg eine Bewerbung mit einem Portfolio, das meine besten Arbeiten in Bildern und einem kurzen Video (dem “Demoreel”) präsentiert. Meinen Film auf ca. eine Minute zu reduzieren, fiel mir daher nicht schwer, da ich für ein Demoreel hiervon ohnehin nur die besten Szenen verwerten würde. Um eine zusätzliche berufliche Qualifikation zu erwerben, habe ich einen Schwerpunkt auf die Programmierung von Skriptdateien zur Unterstützung verschiedenster Arbeitsabläufe innerhalb des Projekts gelegt.
Als mögliche Formate standen für mich Werbespot, Kurzfilm und Trailer zur Auswahl. Alle drei haben ihren Reiz und ich sammelte verschiedene Ideen für jedes Format, indem ich Bücher, Filme und Bilder als Inspirationsquelle heranzog. Ich war schon immer an historischen Stoffen interessiert und meine Themensuche konzentrierte sich daher auf diesen Bereich. Der Konflikt zwischen verfeindeten Parteien in Form eines Zweikampfs taucht oftmals in historischen Erzählungen auf: Ob der Holmgang der Germanen, die mittelalterliche Fehde zwischen Rittern oder das in der napoleonischen Ära verbreitete Duell unter Ehrenmännern: Diese Form der Konfliktlösung birgt eine morbide Spannung, die mich als Leser und Zuschauer stets fesseln konnte. Es entstand meine Absicht, in meinem Film ein Duell in Szene zu setzen.
In dieser kurzen Zeit eine interessante Geschichte zu erzählen ist problematisch: Selbst für das Kurzfilmformat ist eine Minute sehr wenig Zeit um eine Ausgangssituation vorzustellen, Charaktere zu etablieren, Konflikte heraufzubeschwören und die Handlung in einem finalen Höhepunkt aufzulösen. Um den grundlegenden Konflikt der Buchvorlage dennoch innerhalb dieser Beschränkung darstellen zu können, habe ich mich für das Format des Filmtrailers entschieden. Ein Filmtrailer umreißt hochverdichtet die Handlung eines Films, hat aber selten die Absicht, die Geschichte vollständig aufzuklären – schließlich soll beim Zuschauer Interesse für den Film geweckt werden. Es können also Fragen offen bleiben und Szenen nacheinander gezeigt werden, die nicht zwingend aufeinander folgen oder kausalen Zusammenhang haben. Zusätzlich zum Bild ist ein erzählender Sprecher im Filmtrailer üblich, um die Handlung des Films zu erläutern. Im Vergleich zum konventionellen Kurzfilm ergibt sich dadurch eine größere Freiheit in der Bildauswahl, da diese nicht vorrangig die Handlung vermitteln müssen. Ausgehend hiervon habe ich mich dafür entschieden, in meiner Bachelorarbeit einen vollständig 3D-animierten Trailer für einen fiktiven Kinofilm auf Basis von „Moby Dick“ zu produzieren. Im Rahmen der von mir selbst aufgestellten Beschränkungen ergaben sich folgende Produktionsdaten:

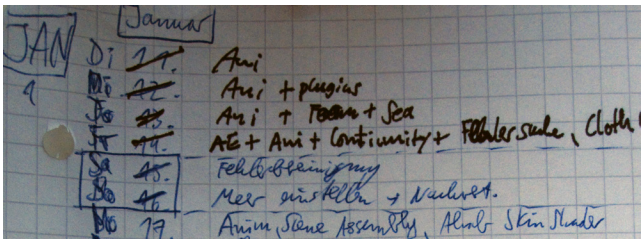
- Dauer: ca. eine Minute
- Charaktere: Ahab und der weiße Wal
- Schauplatz: Das Schiff auf dem offenen Meer

Zusätzlich zur Erstellung des 3D-Filmmaterials sollten professionelle Sprecheraufnahmen, Musik und animierte Texteinblendungen im endgültigen Trailer eingebunden werden. In der Produktion eines 3D-Films von der Idee bis zum fertigen Produkt erhoffte ich mir lehrreiche Erfahrungen und neue Kompetenzen in den verschiedenen Arbeitsbereichen allgemein, und speziell bei der Programmierung in MAXScript.

Die Erstellung eines Films lässt sich klassisch in drei Phasen gliedern: Am Anfang steht die Vorproduktion, in welcher der Film geplant und entworfen wird. Darauf folgt die Produktion, die Dreh- bzw. Animationsarbeiten beinhaltet. Und zum Schluss die Postproduktion, in der das Endprodukt entsteht und seinen finalen Look erhält. Heutzutage verschwimmen die Grenzen zwischen diesen Phasen zunehmend. Die Digitalisierung der Filmproduktion und die technischen Möglichkeiten erlauben es, die meisten Aspekte eines Films noch in der Postproduktion ändern zu können. Im 3D-Animationsfilm ist der früher zwingend notwendige chronologische Ablauf von Arbeitsschritten ebenfalls unnötig geworden – eine entsprechende Organisation der Daten vorausgesetzt. Die Animation von Charakteren beispielsweise kann bereits weitgehend finalisiert werden, wenn noch nicht einmal der Schauplatz, geschweige denn die Figuren modelliert wurden. Animationsdaten lassen sich von Platzhalterfiguren im Nachhinein auf die finalen Charaktere übertragen. Viele Arbeitsprozesse laufen somit heutzutage parallel ab, und nicht mehr nacheinander. Um der Dokumentation eine übersichtliche Struktur zu geben, habe ich dennoch eine klassische, chronologische Reihenfolge der verschiedenen Arbeitsschritte gewählt.

3 Organisation

Die Erstellung meiner Bachelorarbeit war ein komplexes und langwieriges Projekt. Um hierbei den Überblick zu behalten und das angestrebte Ziel innerhalb eines Semesters zu verwirklichen, war ein Zeitplan vonnöten. Obwohl heutzutage Software wie z.B. Excel Möglichkeiten zur Erstellung von Stäbchenplänen oder Ähnlichem bietet, habe ich in diesem Zusammenhang der analogen Variante den Vorzug gegeben. Meine Zeitpläne wurden per Hand auf DIN-A4 Seiten geschrieben, die ich jederzeit auch losgelöst vom Computer ergänzen und überarbeiten konnte.



Trotz fragwürdiger Optik erwiesen sich die handgeschriebenen Zeitpläne als äusserst nützlich.

Es existierten zeitgleich immer ein grober Plan für den Gesamtverlauf des Projekts, sowie jeweils ein Plan für die aktuelle Woche bzw. den Monat, in dem ich für jeden Tag zu erfüllende bzw. bereits erledigte Aufgaben eintragen konnte. Dieses Mittel zur Selbstkontrolle war hilfreich um den Fortschritt des Projekts überprüfen zu können. Der Zeitplan selbst wurde wöchentlich angepasst, und hatte schlussendlich nur noch wenig mit seiner ersten Fassung gemein. Vor allem technische Probleme und deren zeitaufwendige Beseitigung wurden zu einer lehrreichen Erfahrung, die ich in zukünftigen Projekten verstärkt beachten werde.
Neben der zeitlichen Organisation entstanden auch Tabellen und Listen, um den Arbeitsaufwand für den Trailer einschätzen zu können. Die Komplexität der Szenen unterschied sich teilweise sehr, sodass z.B. in Excel eine Tabelle angelegt wurde, die die einzelnen Kameraeinstellungen einander gegenüberstellt. Hierfür bot sich das aufgeräumte, übersichtliche Format einer Exceldatei gegenüber einem handgeschriebenen Zettel an, da es auf einen Blick vergleichende Informationen liefert.

Shot:	Chars:		Szene:			FX:				Besonderheiten:
	Ahab	Moby	Meer	Horizont	Schiff	Partikel	PhysX	Rauch	Cloth	
E01			x	x	x	x			x	
E02			x		x	x			x	
E03			x	x	x	x			x	
E04	x				x				x	
E05	x				x				x	
E06		x	x	x	x					Unterwasser
E07		x	x		x				x	
E08	x	x			x	x	x		x	Slow Motion
E09	x		x		x				x	
E10	x	x			x		x			
E11					x		x		x	
E12	x				x		x		x	
E13					x		x		x	
E14			x		x					
E15			x		x				x	
E16		x								
E17	x				x				x	
E18					x	x		x		Rauch
E19	x		x		x	x		x	x	Rauch
E20		x	x	x		x		x		Rauch
E21	x	x	x	x	x	x			x	Slow Motion
E22	x	x			x				x	Slow Motion
E23	x				x				x	
Schriftzug										

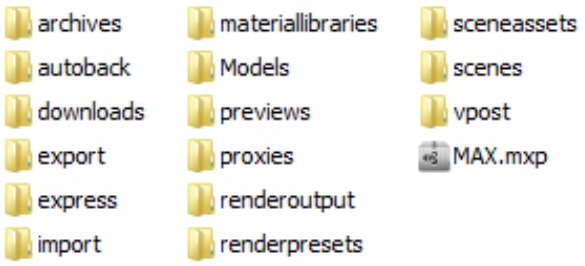
Exceltabellen waren aufgrund ihrer Übersichtlichkeit für die Auflistung des Animationsaufwands der Szenen besonders geeignet.

Die Einteilung nach Kameraeinstellungen (insgesamt 23 im Trailer) spiegelt sich auch in der Dateiverwaltung und Ordnerstruktur wider. Für die 3D-Dateien der Animation, die ausgelagerten Simulationsdaten der Stoffsimulation und der animierten Meeresoberfläche, sowie die gerenderten Bilder gab es jeweils einen Ordner pro Einstellung. Somit war stets klar, wo man benötigte Daten finden kann. Das Projekt insgesamt befand sich in einem einzigen Ordner, der mithilfe der Projektordnerstruktur von 3ds Max angelegt wurde. Diese besitzt den Vorteil von relativen Pfaden der Unterordner, d.h. dass man den gesamten Projektordner auf einen beliebigen Computer oder ein Laufwerk migrieren kann, ohne die Verlinkung der dazugehörigen Bild- und Animationsdaten später manuell erneuern zu müssen. Diese Problematik kann sich als sehr zeitaufwendig erweisen, wenn beispielsweise zum Rendern das Projekt auf die Renderfarm der Hochschule transferiert werden muss. Die „Ein Ordner für Alles“-Strategie erleichtert diesen Transfer enorm. Zusätzlich beschle-

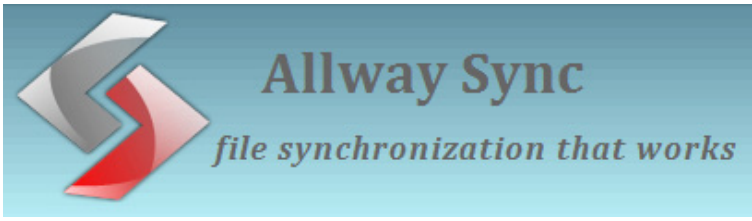
unigt sie die Navigation während der Arbeit in 3ds Max. So wird etwa beim Import oder Export von Dateien automatisch der entsprechende Unterordner aufgerufen. Dieser subtile Bonus ist nicht zu unterschätzen, spart er doch nicht nur Zeit, sondern vor allem Nerven, da man sich nicht wiederholt durch dutzende Unterordner klicken muss.

Bei einem Projekt dieser Art wäre der Verlust aller Daten z.B. durch einen Festplattendefekt eine Katastrophe und würde vermutlich das Ende des Projekts bedeuten. Eine mehrfache Sicherung der Daten auf externen Datenträgern ist daher obligatorisch, um das Risiko eines Datenverlusts zu minimieren. Da das Kopieren des kompletten Projektordners vor allem im späteren Verlauf sehr zeitaufwendig geworden wäre, habe ich eine spezielle Backup-Software benutzt, nämlich Allway Sync von der Firma Botkind. Diese bietet den Vorteil eines intelligenten Backups: Man wählt zwei Ordner oder Laufwerke aus, die synchronisiert werden wollen. In einer schnellen Analyse ermittelt das Programm die Unterschiede zwischen beiden Ordnern, die dann in einer Gesamtübersicht angezeigt werden.

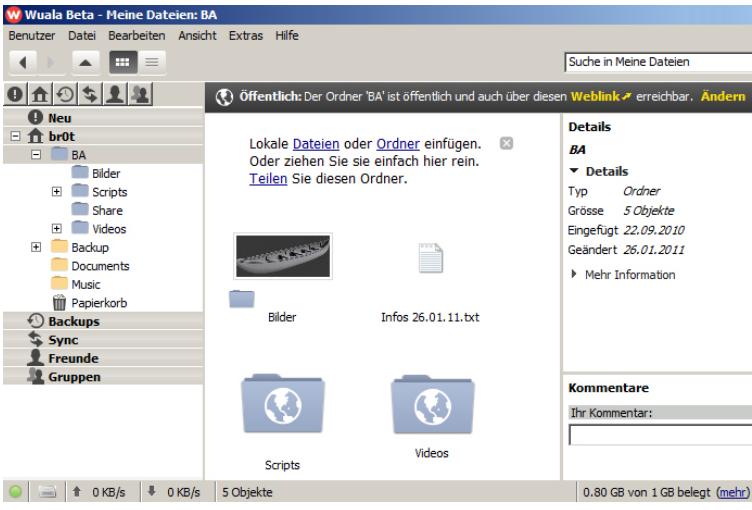
Die Übersicht bietet die meiner Meinung nach wichtige Möglichkeit, zu überprüfen welche Daten kopiert, gelöscht oder überschrieben werden sollen. Bei der Synchronisation werden anschließend beide Ordner auf den aktuellsten Stand gebracht, sofern nicht ein einseitiger Abgleich gewünscht ist. Dieser Abgleich ist inkrementell, d.h. wenn in dem 30 GB großen Projektordner nur 200 MB geändert wurden, werden auch nur diese kopiert, und nicht der ganze Ordner. So ist stets ein schnelles vollständiges Backup möglich, ohne dass man sich manuell um einzelne Dateien kümmern muss. Sicherungen wurden täglich angelegt, und zudem wöchentlich auf einen dritten Datenträger überspielt. Ein möglicher Datenverlust hätte somit im Normalfall maximal die Arbeit eines Tages zerstört.
Um jederzeit von jedem Ort Zugriff auf Bilder, Tabellen und Daten des Projekts Zugriff zu haben, wurde ein Online-Laufwerk mit der Software Wuala eingerichtet. Dieses bietet im Vergleich zu gewöhnlichem Webspace die Möglichkeit, eine ganze Ordnerstruktur online zugänglich zu machen. Damit eignet es sich gut dazu, den Projektfortschritt in Bildern, Text und Video zu veranschaulichen. Aktuelle Änderungen wurden in einer Textdatei auf dem Online-Laufwerk vermerkt, um die Kommunikation mit meinem Betreuer zu vereinfachen.



Die Ordnerstruktur des Projekts. Über ein zugehöriges Skript ("MAX.mxp") lässt sich die 3ds Max Installation mit dem Projektorder verknüpfen.



Die Backup-Software Allway Sync ist zur nichtkommerziellen Nutzung frei verfügbar.



Wuala bietet ein Online-Laufwerk, dass auch ohne Software über den Browser verwaltet werden kann. Da alte Daten erhalten bleiben kann der Verlauf des Projekts in Bildern und Videos dokumentiert werden.

4 Vorproduktion

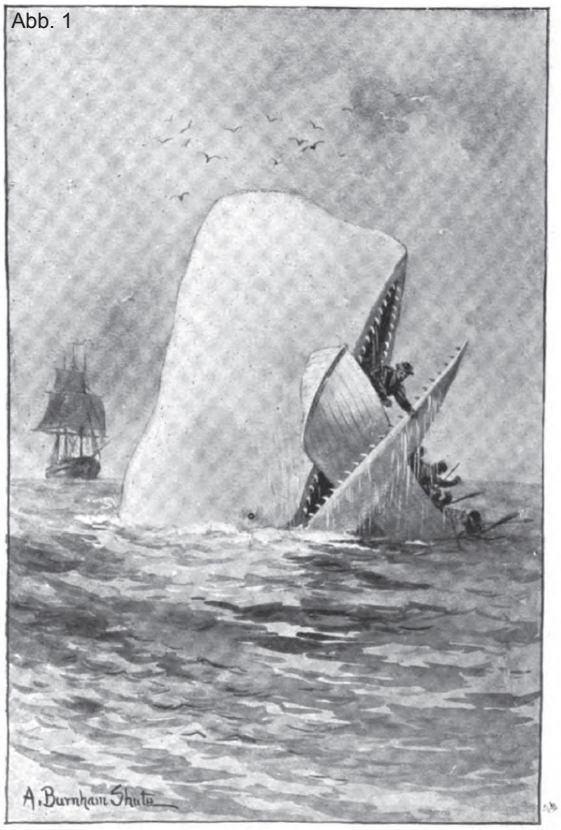
4.1 Recherche

„Strömt herbei aus fernster Ferne, all ihr Wellen meines nun verflossenen Lebens, und schwellt mir die eine himmelhohe Woge, die mich fällen soll. Dir, Wal, treibt sie mich zu, du magst mich vernichten, wie du alles zerstörst – überwinden wirst du mich nie! Bis ans Ende ring ich mit dir, aus der tiefsten Hölle noch stoß ich nach dir, mit dem letzten Atemzug spei ich dir meinen Hass ins Gesicht."

Ahab

Der vorangehende Satz gehört zu den bekanntesten Zitaten aus dem Roman „Moby Dick" und charakterisiert in wenigen Worten die Figur des zutiefst vom Wunsch nach Rache getriebenen Kapitän Ahab. Auch lässt sich bereits das hohe sprachliche Niveau erkennen, in dem das Buch verfasst wurde. Da ich die Welt und die Figuren des Romans als Vorlage für meinen Trailer gewählt hatte, konnten das Buch und seine bisherigen Verfilmungen als Inspirationsquelle für die grafische Umsetzung dienen. In einer umfangreichen Recherche wurden alle verfügbaren Text, Bild und Videomaterialien die ich im Zusammenhang zu „Moby Dick" und anderen Schlagwörtern finden konnte von mir gesichtet und teilweise archiviert. Das Internet hat sich hierbei als äußerst schnelle und ergiebige Informationsquelle erwiesen.

Abb. 1



Oben: Zeichnung eines möglichen Angriffs durch Moby Dock.
Unten: Holzschnitt der Jagd auf einen Grönlandwal.

Abb. 3

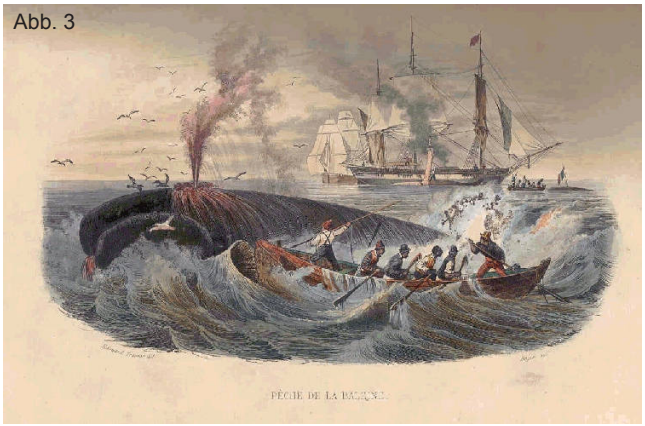
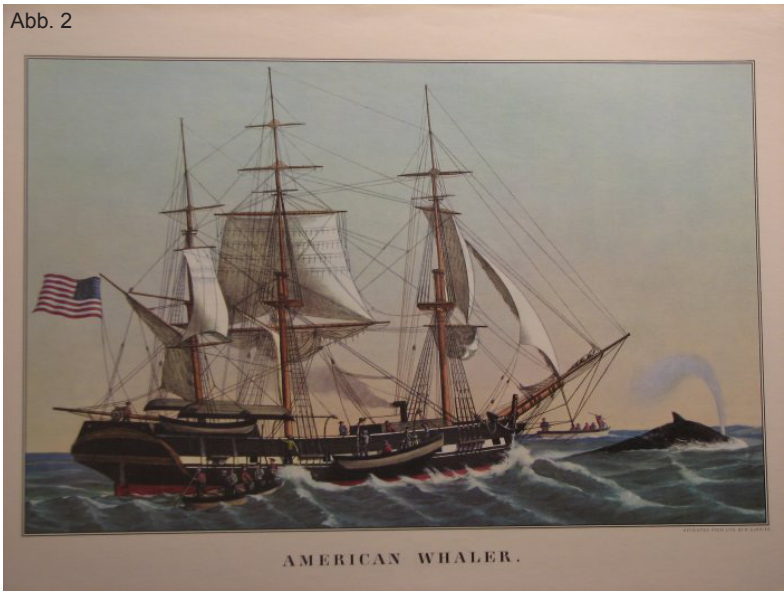
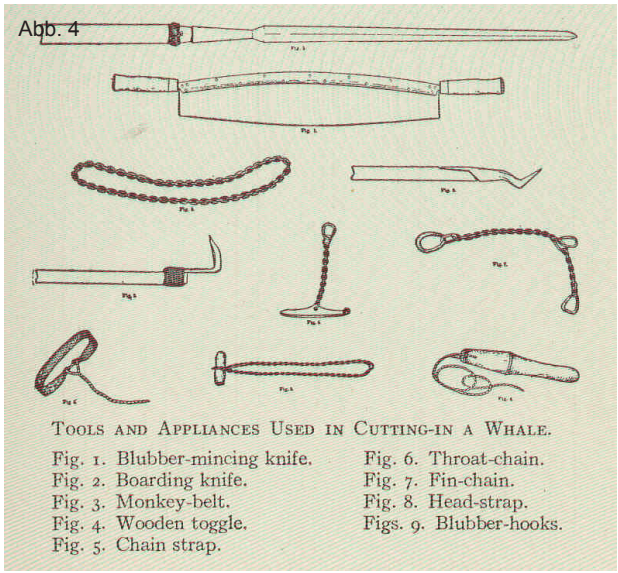


Abb. 2



Oben: Zeichnung eines amerik. Walfangschiffs im 19. Jhdt.
Unten: Typische Utensilien zum Zerteilen eines getöteten Wals.

Abb. 4



Die bisherigen Verfilmungen der Buchvorlage wurden studiert und als Inspirationsquelle für die visuellen Elemente im Trailer benutzt. Die ersten drei Bilder entstammen dem Kinofilm “Moby Dick” von John Huston aus dem Jahr 1952. Die folgenden Bilder entstammen der Fernsehproduktion “Moby Dick”, verfilmt von Franc Roddam 1998.

Abb. 5

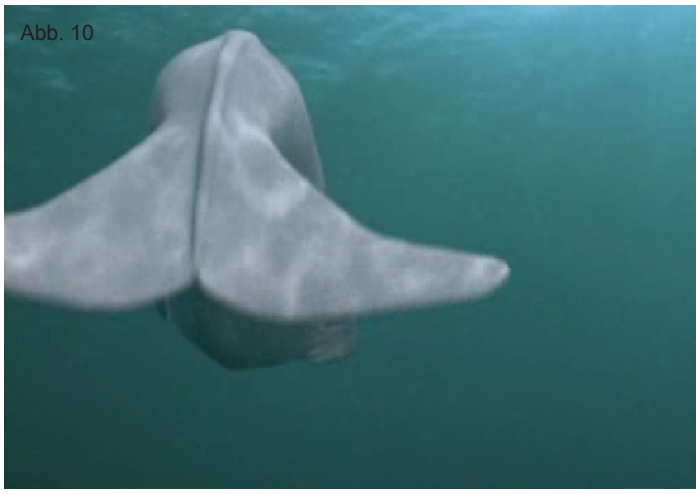
Abb. 6

Abb. 7

Abb. 8

Abb. 9

Abb. 10



Durch das gesammelte Referenzmaterial konnten in der Produktion viele Fragen zu Form und Aufbau des Schiffes und seiner Ausrüstung beantwortet werden. Speziell bei der Modellierung des Rumpfs und der Takelage des Schiffs waren originale Konstruktionszeichnungen von Walfangschiffen der damaligen Zeit hilfreich. Anatomische Unklarheiten bei der Modellierung des weißen Wals konnten durch die Beobachtung entsprechender Fotografieren von heute lebenden Pottwalen geklärt werden. So besitzt beispielsweise der Pottwal als einziger

Abb. 11

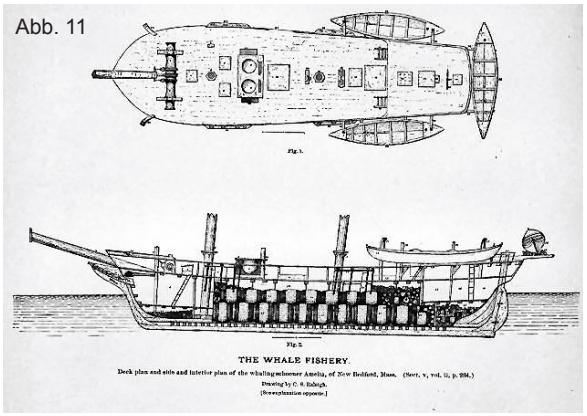
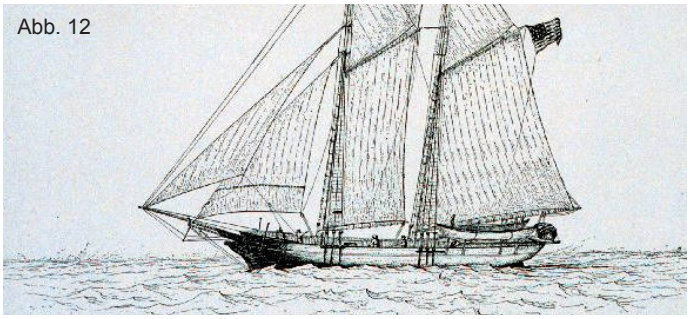
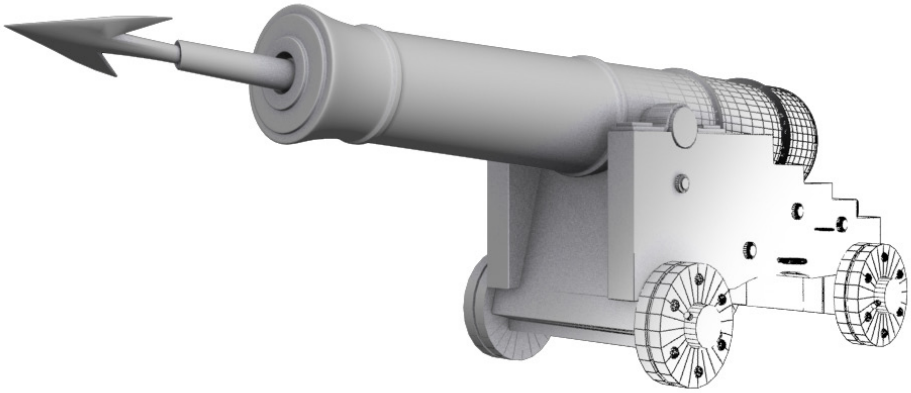


Abb. 12



Für die Konstruktion des Schiffes wurden mehrere Originalpläne von Walfangschiffen zu einem individuellen Schiffstyp frei kombiniert.

Wal seiner Größe Zähne, allerdings nur im Unterkiefer, mit entsprechenden Vertiefungen im Oberkiefer. Bilder vom Skelett eines Pottwals waren hilfreich bei der Konstruktion des Animationsskeletts während des Riggings, um Gelenke korrekt platzieren zu können. Auch für einzelne Ausrüstungsgegenstände wie Harpunen, Walfangboote oder Ahabs Kleidung wurden Referenzbilder genutzt, um reine Fantasieprodukte zu verhindern. Jedoch hatte ich nicht den Anspruch, jedes Detail historisch korrekt darzulegen und habe mir die Freiheit genommen, bei Bedarf die Vorlage abzuändern. So wurden beispielsweise bei der Konstruktion des Schiffs mehrere Baupläne für verschiedene Schiffstypen gemischt, und auch die Besegelung frei miteinander kombiniert, um aus jedem Blickwinkel ein angemessenes Profil bieten zu können. Das Ergebnis ist eine Mischung aus einem Schooner und einer Bark, die zusätzlich ein aufgebautes Achterdeck mit darunterliegender Kajüte besitzt, das an eine Galleone erinnert. Obwohl Anfang des 19. Jahrhunderts noch vielfach rechteckige Rahsegel zum Einsatz kamen, habe ich mich zugunsten der Ästhetik für die moderneren, schräg stehenden Schratsegel entschieden.
Ein Schiff mit Rahsegeln wird vom Wind geschoben, kann also nur in Windrichtung Fahrt aufnehmen. Modernere Schratsegel können durch ihre Schrägstellung gegen den Wind fahren und sind somit flexibler. Ein weiterer Grund hierfür war der Umstand, dass ich außer Kapitän Ahab keine Besatzung zeigen würde: Dies war leichter zu verschmerzen wenn ich Rahsegel vermied, da diese eine Vielzahl an Männern benötigen um das Schiff segeln zu könne. Die Schratsegel hingegen brauchen nur eine sehr kleine Besatzung, wodurch deren Abwesenheit im Film nicht völlig absurd wirkt.
Die Recherche und das Sammeln von Referenzmaterialien fanden nicht nur zu Beginn des Projekts, sondern auch parallel zu Modeling, Texturing und der Animation statt. Oftmals gaben die gefundenen Bilder auch Anregung für die Objekte im Film: Zur Zeit von „Moby Dick“ wurden Wale noch umständlich mit zu Wasser gelassenen Booten gejagt, die von Ruderern angetrieben wurden. Die Harpunen wurden mit Muskelkraft auf den Wal geschleudert. Diese Art der Jagd kam für meinen Trailer jedoch nicht in Frage, da Ahab alleine in einem Boot wenig Eindruck macht. Bei der Recherche stieß ich jedoch auf Bilder modernerer Walfangschiffe, die mit Harpunenkanonen ausgestattet sind, welche mit Luft- oder Explosionsdruck abgefeuert werden. Diese Idee habe ich aufgegriffen und Ahabs Schiff mit einem Kanonendeck ausgestattet, wobei die Kanonen nicht mit Kugeln, sondern mit Harpunen geladen sind. Obwohl unrealistisch, bietet diese Variante sowohl einen Überraschungseffekt (da die Kanonen vorher nicht sichtbar sind) und eine adäquate und visuell interessante Waffe für Ahab gegen seinen mächtigen Feind.

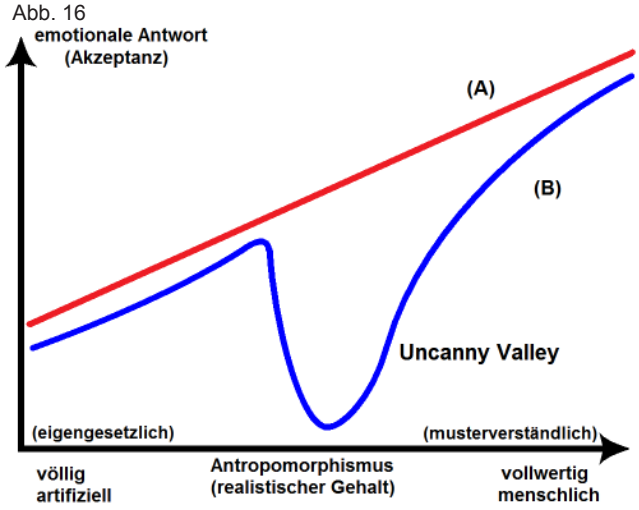


Das Modell der Harpunenkanonen auf Ahabs Schiff.

4.2 Produktions- und Charakterdesign

Um festzulegen, welchen visuellen Stil der Trailer haben sollte, war es zunächst hilfreich nach existierenden Vorlagen im Internet zu suchen. Hierbei fanden sich viele teils professionell, teils amateurhaft produzierte Trailer, Musikvideos und Animationen. Konkrete 3D-animierte Filme waren hingegen kaum darunter, von einem Demoreel abgesehen. Meine Absicht war es, die Optik des Trailers weitgehend realistisch zuhalten, jedoch speziell bei den Charakteren zu abstrahieren. In der Vermeidung einer allzu realistischen Menschendarstellung lässt sich ein im Animationsfilm verbreitetes Problem umgehen oder zumindest abschwächen, nämlich das des sog. „Uncanny Valley“. Dieser Begriff bezeichnet die abnehmende Akzeptanz des Zuschauers beim Betrachten von sehr menschenähnlichen Charakteren. Allgemein steigt die Anziehungskraft von künstlichen Figuren auf den Zuschauer an, umso realistischer diese gezeichnet sind. Empirische Untersuchungen haben jedoch gezeigt, dass ab einem bestimmten Wert diese Akzeptanz dramatisch abnimmt, und erst ab einer beinahe fotorealistischen Ähnlichkeit zum Menschen wieder steigt

Abb. 16



Die Grafik veranschaulicht das Problem des Uncanny Valley. Bei steigender Menschenähnlichkeit bricht die emotionale Akzeptanz des Zuschauers rapide ein.

Eine mögliche Erklärung dieses Effekts ist, dass realistische künstliche Figuren vom Betrachter als eine Art Mensch gesehen werden, wodurch aber alle noch vorhandenen Differenzen zum wirklichen Menschen stärker und negativer auffallen als zuvor. Die kleinsten Unstimmigkeiten in Mimik, Körperbewegung, Proportionen und Aussehen werden vom Gehirn registriert und als Makel erkannt – die Akzeptanz sinkt. Das „Uncanny Valley“ wird als ein möglicher Grund für den ausbleibenden Erfolg mancher modernen Animationsfilme gesehen. Mögliche Beispiele sind „Beowulf“ oder „Der Polarexpress“, in denen die Darbietung realer Schauspieler auf hochrealistische künstliche 3D-Charaktere übertragen wurde. Ich persönlich verbinde ein gewisses Unwohlsein damit, dass computeranimierte Charaktere die klassische Schauspielerei ersetzen sollen.


Abb. 13


Abb. 14

Ray Winstone war in “Beowulf” und Tom Hanks in “Der Polarexpress” die Vorlage zur Erstellung der digitalen Schauspieler.

Für einen abstrahierten Kapitän Ahab fanden sich viele Inspirationen und Bilder in Filmen, Comics und im Internet. Mithilfe eines Moodboards wurden die Vorlagen verglichen und die besten ausgewählt. Nach meiner Vorstellung sollte Ahabs Besessenheit und Skrupellosigkeit ihm ins Gesicht geschrieben sein: Runde, weiche Formen wollte ich daher vermeiden und stattdessen harte, kantige Linien nutzen, die sein Gesicht wie gemeißelt aussehen lassen.


Abb. 15

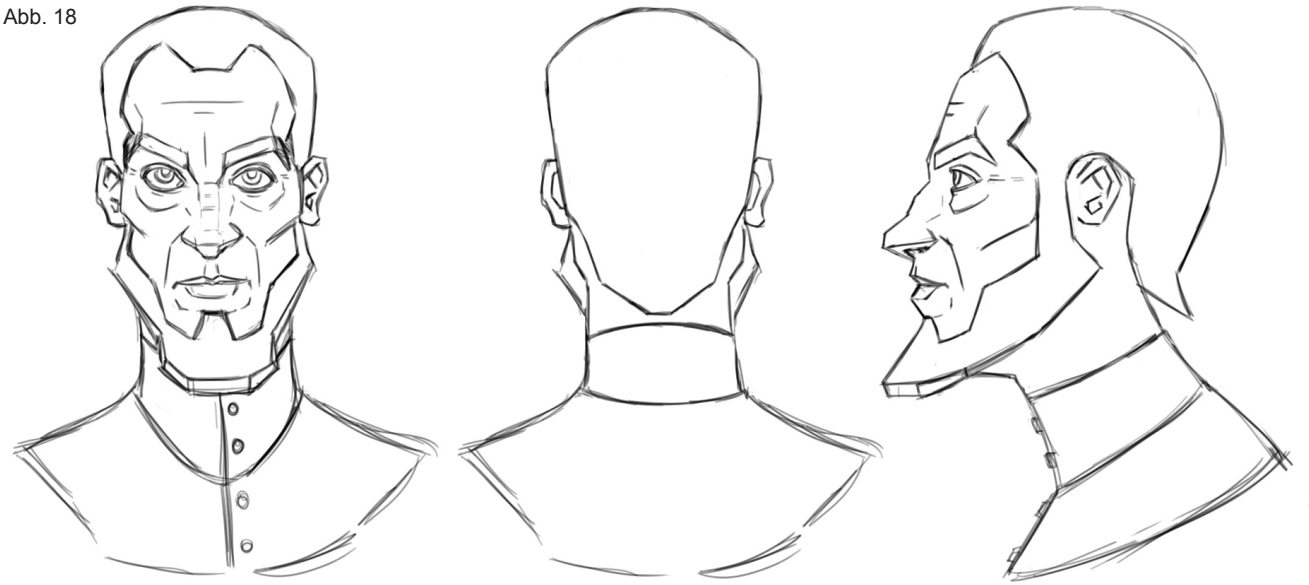
Ein Standbild aus Star Wars: The Clone Wars. Die harten Gesichtslinien des Charakters Count Dooku standen Vorlage für das spätere 3D-Modell von Ahab.


Abb. 17

In Internetgallerien fanden sich diese Zeichnungen der Figur des Ahab, die anschaulich darlegen, dass sein Charakter sehr unterschiedlich visualisiert werden kann.

Da meine eigenen zeichnerischen Fähigkeiten und Erfahrungen im Charakterdesign begrenzt waren, habe ich Konstantin Krug um Hilfe gebeten. Er erstellte mir einen sog. „Model Sheet“ von Ahabs Gesicht, also eine Zeichnung aus der Front- und Seitenansicht, welche auf einem ausgerichtet ist:

Abb. 18



Der Modelsheet von Ahab, gezeichnet von Konstantin Krug.

Der Model Sheet ist eine ideale Vorlage für die Modellierung in einem 3D-Programm und war die Basis für das Modell von Ahabs Kopf.
Für das Modell des Moby Dick habe ich weitgehend Fotografien von realen Pottwalen zu Rate gezogen. Die meisten künstlerischen Vorlagen unterschieden sich teilweise dramatisch in ihrer anatomischen Genauigkeit, daher waren die Fotografien in dieser Hinsicht eine bessere Grundlage.

Um dem weißen Wal dennoch eine individuelles Aussehen zu geben, habe ich seine Proportionen verändert. Ein generell massigeres Erscheinungsbild, die bullige Augenpartie, der kantige und verbeulte Kopf, sowie ein verlängerter Unterkiefer lassen ihn bedrohlicher erscheinen. Aus der Buchvorlage wurde zudem die Idee übernommen, seinen Rücken mit alten Harpunen und Lanzen zu spicken, als Andenken an vergangene Kämpfe gegen Walfänger.
Für die allgemeine Optik des Films und der Kameraeinstellungen habe ich mich an Filmen wie „Fluch der Karibik“, „Hornblower“ und „Master and Commander“ orientiert. Speziell letzterer zeigt vielfältige Möglichkeiten zur filmischen Auflösung eines Schiffs auf dem offenen Meer.

Abb. 19

Abb. 20

Stilistische Vorlagen aus dem Kinofilm von 1952 (oben) und dem Moby Dick Comicheft von Marvel (unten) für das 3D-Modell von Moby Dick.

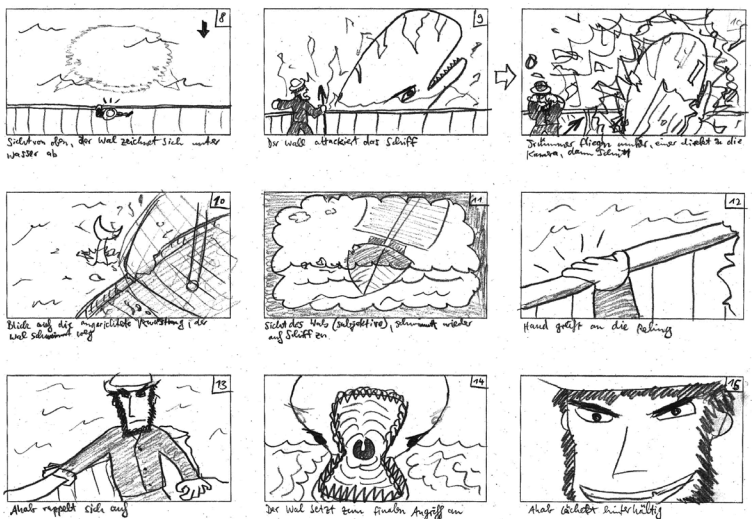
Abb. 21

Abb. 22

Zwei Standbilder aus “Master and Commander” von Peter Weir, 2003.

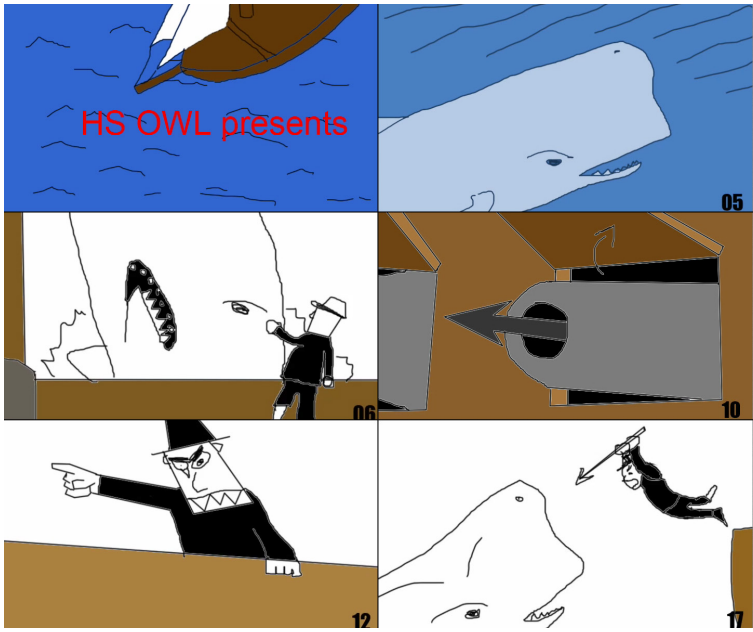
4.3 Storyboard, Animatic und Blocking

Bei der Produktion eines Kurzfilms wird vor dem Erstellen eines Storyboards üblicherweise ein Drehbuch angefertigt. Bei der zeitlichen Begrenzung meines Projekts war dieses aber von nicht nennenswertem Aussagewert, sodass ich den Ablauf der Geschichte in der ersten Phase handschriftlich festhielt und dann direkt zum Storyboard überging. Dieses dient zu Beginn vorrangig der Visualisierung der Geschichte. Die Ausrichtung der Kamera und verschiedene Einstellungsgrößen lassen sich auf einfache Art und Weise testen. Im Gegensatz zum Film, der in finaler Form erst im Schnittraum entsteht und für den das Drehbuch eine vage Basis bildet, entsteht ein Animationsfilm in filmischer Hinsicht bereits zum größten Teil in der Vorproduktion. Das erste Storyboard muss kein zeichnerisches Meisterwerk sein und kann dennoch bereits einen wertvollen Ausblick auf den geplanten Film liefern. Das am Rand gezeigte Storyboard basierte noch auf einer Idee für einen Kurzfilm, und wurde nach umfangreicher Überarbeitung zum finalen Storyboard für den Trailer. Die Idee ist in grobe Bilder umgesetzt worden, die im folgenden Schritt, dem Animatic, weiter geprüft und verbessert werden.



Dieses frühe Storyboard konzipierte noch einen Kurzfilm.

Das Animatic ist ein animiertes Storyboard und bietet somit die Möglichkeit, mit einfachen Mitteln das Timing der Einstellungen und mögliche Kamerabewegungen zu testen. Auch erste, sehr simple Animationen können im Animatic auftauchen, im gezeichneten Animationsfilm sogar bis zu einem Grad, der schon sehr nahe an der finalen Animation ist. In meinem Fall wurde das Storyboard eingescannt und in der Software After Effects in eine animierte Diashow verwandelt. In späteren Versionen wurden mithilfe von Photoshop Bilder abgeändert oder hinzugefügt, wobei Papier und Bleistift zunehmend durch das Grafiktablett ersetzt wurden, da dieses das umständliche Einscannen obsolet machte. Wo im Storyboard vor allem die Frage „Was zeige ich, und auf welche Art?“ geklärt wird, gibt das Animatic Antwort auf weitere Fragen: „Was zeige ich, wie und wann zeige ich es und für wie lange?“ Die Planung des Trailers ließ sich mithilfe des Animatics bereits sehr genau ausführen. Im nächsten Schritt, dem Blocking, wurde das Animatic in die dritte Dimension transferiert.

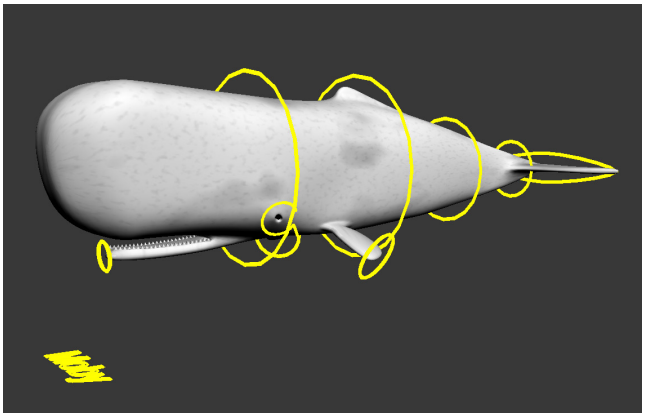


Trotz der groben Optik waren Timing und Ablauf des Trailers in diesem Animatic bereits weitgehend final.

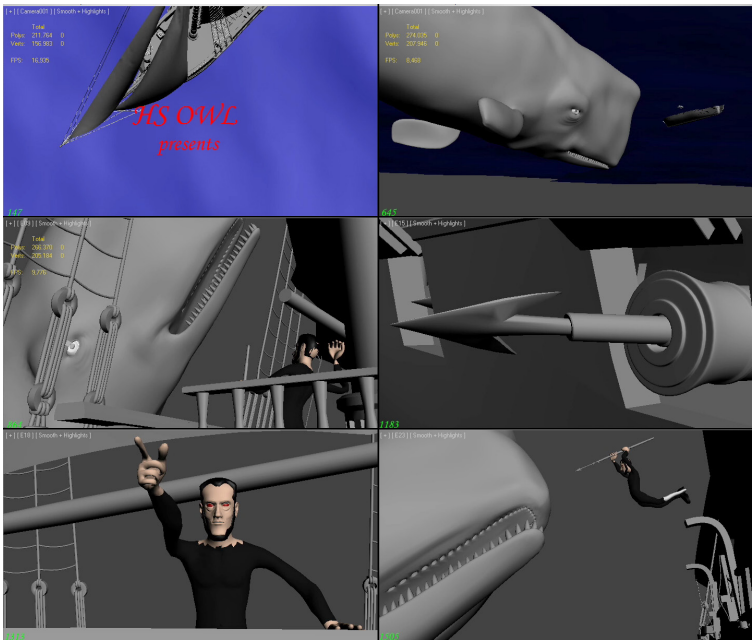
Das Blocking ist ein Animatic im dreidimensionalen Raum, der in meinem Projekt mit 3ds Max erstellt wurde. Hier lassen sich mithilfe von einfachen Charakteren und Objekten, die als Platzhalter für die späteren Modelle dienen, die im Storyboard und Animatic entwickelten Kameraeinstellungen nachbauen. Indem man die Möglichkeiten der Software nutzt, können hier Kamerafahrten, Objektive mit verschiedenen Brennweiten und einfache Animationen benutzt werden, um

Stück für Stück den Film immer genauer in bewegte Bilder aufzulösen. Eine Zeichnung kann, vor allem wenn sie die Szenerie nur grob skizziert, Probleme z.B. bei der Kamerapositionierung verschleiern. Wenn man etwa im Blocking versucht, die Kamera exakt so zu positionieren wie es in der Zeichnung skizziert wurde, kann plötzlich ein Objekt den Blick versperren, da die Zeichnung perspektivische Ungenauigkeiten beinhaltet. Solche Probleme lassen sich im Blocking erkennen und lösen. Da alle Objekte grobe Platzhalter sind, sind die Handhabung der Szenerie und das Umsetzen von Änderungen im Film zu diesem Zeitpunkt noch schnell und einfach. Es lohnt sich, Zeit in diesen Prozess zu investieren, um in der Produktion Fehler zu vermeiden. Früherkennung und Beseitigung von Problemen in der Vorproduktion können eine ganze Fehlerkette, die womöglich erst in der Postproduktion bemerkt wird, verhindern und so viel Zeit und Arbeit sparen. In meinem Fall habe ich auch vom Blocking mehrere Versionen angefertigt und immer wieder angepasst, sodass zum Schluss das Timing der Szenen und die Kameraanimation bereits annähernd final waren und in der Produktion übernommen werden konnten. Die Entwicklung des Trailers mithilfe von Storyboard, Animatic und Blocking verlief parallel zu diversen anderen Arbeitsschritten, die in den nächsten Kapiteln erläutert werden.

Für die dreidimensionale Prävisualisierung wurden einfache Platzhaltercharaktere erstellt, die rudimentär animiert werden konnten. Ahab basiert hier auf dem "MAX"-Rig von Peter Starostin.



Das Previz-Modell und Rig des Wals wurde innerhalb von zwei Stunden angefertigt. Es diente zudem als Basis für die Modellierung des finalen 3D-Modells von Moby Dick.



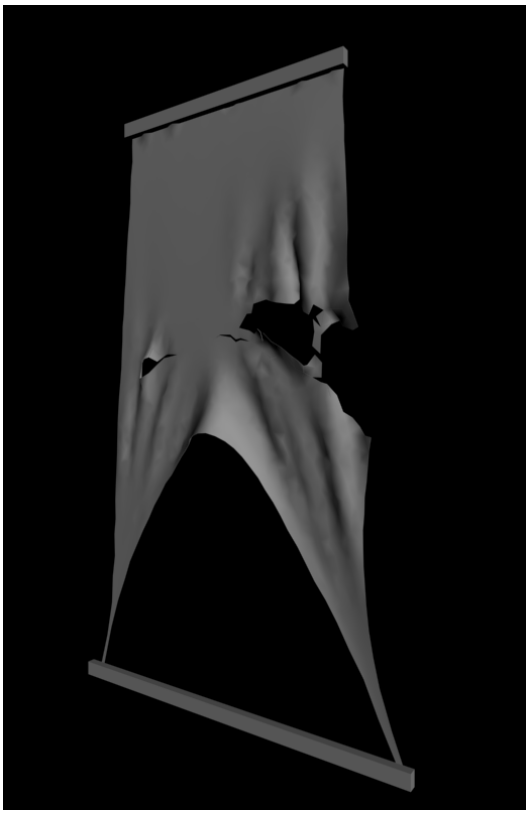
Das Blocking baut auf dem Animatic auf, nutzt jedoch vorhandene Ressourcen, Modelle und Animationen, um den Ablauf des Trailers weiter zu finalisieren. Diese Bilder zeigen für einen besseren Vergleich die gleichen Einstellungen wie das Animatic auf der vorherigen Seite.

4.4 R&D

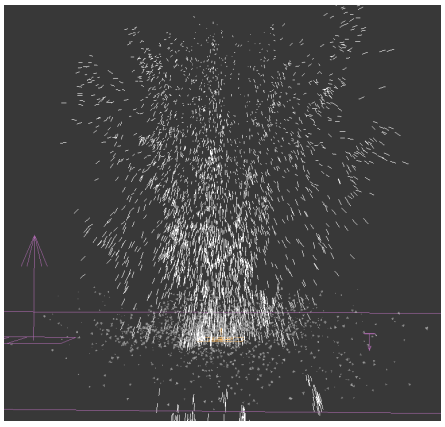
R&D ist die Abkürzung für den englischen Begriff „Research & Development“, also Forschung und Entwicklung. Im 3D-Kontext wird damit oft eine Reihe von Experimenten bezeichnet, in denen neue Wege zur Lösung von bisher nicht aufgetretenen Problemen und Herausforderungen gesucht werden. Trotzdem ich grundlegende Erfahrungen in allen 3D-Arbeitsbereichen besaß, gab es eine Vielzahl von speziellen Aufgaben, mit denen ich mich bis dahin noch nicht konfrontiert sah. Einige Beispiele sind:

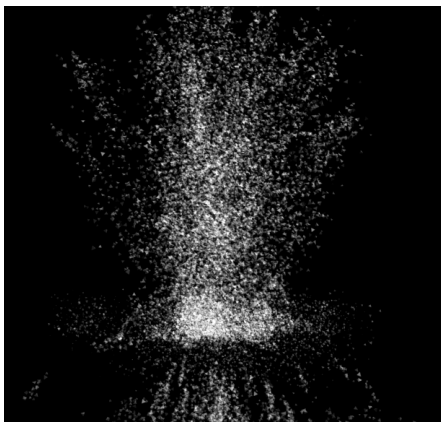
- Animation einer bewegten Meeresoberfläche
- Effiziente Stoffsimulation von sich überlagernden Kleidungsstücken
- Stoffsimulation von Segeln, die an beweglichen Tauen befestigt sind
- Automatisierte Änderungen einer Vielzahl von Dateien per Skript (sog. „Batch processing“)

Für diese Arbeit gab es keinen konkreten Zeitraum, stattdessen wurden zu verschiedensten Gelegenheiten Lösungen ausprobiert und Experimente durchgeführt. Für gewöhnlich geschieht dies mithilfe von groben Objekten und einfachen Animationen, um beispielsweise den Zeitaufwand bei der Stoffsimulation so gering wie möglich zu halten. Auf diese Art ließen sich schnell wertvolle Erkenntnisse gewinnen, die im Verlauf des Projekts auf die finalen Modelle und Animationen übertragen werden konnten. Auch habe ich Testläufe späterer Arbeitsschritte wie dem Compositing einer gerenderten Kameraeinstellung oder der Nachvertonung des Videos durchgeführt, um ein Gefühl für den nötigen Zeitaufwand zu bekommen. Diese Tests haben bei der regelmäßigen Anpassung des Zeitplans geholfen.



Die Abbildung zeigt einen Testlauf zum Zerreißen von Segeln mittels einer Stoffsimulation. Dieser Effekt wurde im Nachhinein nicht genutzt, war aber Teil des R&D.





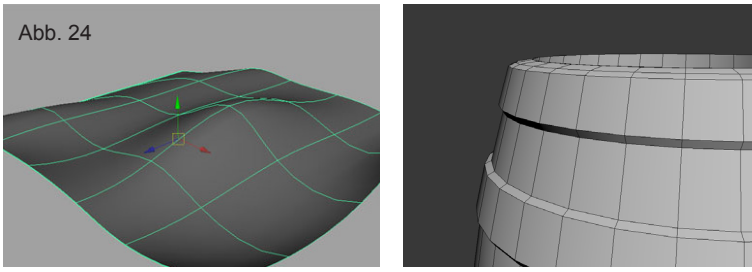
Die aufbrausende Gischt wurde in einer separaten Datei mithilfe eines Partikelsystems erstellt. Oben die Ansicht in 3ds Max, unten das Renderbild. Dieses wurde über einen Alphakanal maskiert und im Compositing in das Video integriert.

5 Produktion

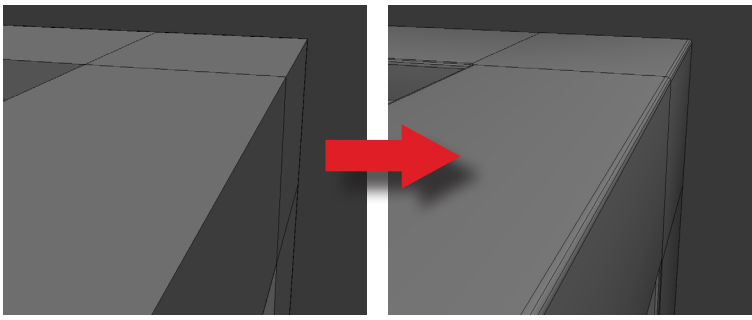
5.1 Modeling

Moderne 3D-Software bietet verschiedene Ansätze zur Modellierung dreidimensionaler Objekte. Modelle können aus Kurven bestehen, die auf mathematischen Funktionen basieren und somit auflösungsunabhängig sind, d.h. aus jeder Entfernung einen hohen Detailgrad bieten. Diese in der Frühzeit der 3D-Grafik entstandene Methode wird NURBS („Non-Rational Uniform B-Splines“) genannt und ist nach wie vor gebräuchlich, speziell bei geschwungenen Formen (z.B. in der Automobilvisualisierung). Die am meisten verbreitete Form der Modellierung ist aber das sog. „Polymodeling“, also die Konstruktion von Objekten mithilfe polygonaler Flächen. Die Polygone bzw. Vielecke bestehen aus Eckpunkten und Kanten, die ein dreidimensionales Gitter bilden. Durch Verschieben der Punkte im Raum, sowie durch Teilen der Kanten und das Hinzufügen neuer Flächen entsteht im Verlauf der Zeit die gewünschte Form. Polygonale Modellierung ist sehr flexibel und geeignet für jede Form von Objekt, ob mechanisch oder organisch. Mögliche Nachteile sind die feste Auflösung und ein oft ungewollt kantiges Erscheinungsbild.

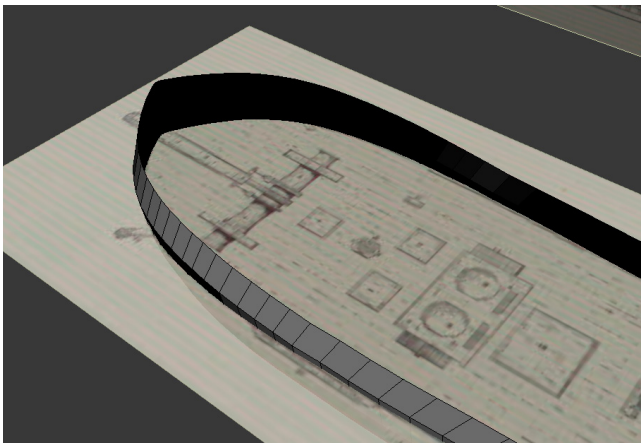
Grundsätzlich wurden für die Modellierung aller Objekte Referenzbilder genutzt, in Form von Fotografien oder Zeichnungen aus dem Internet. Alle Modelle wurden in einem realistischen Größenverhältnis angefertigt. Dies ist wichtig, da im 3D-Raum der Software zwar augenscheinlich die Größe keine Rolle spielt, in späteren Arbeitsschritten eine falsch gewählte Größeneinheit aber zu Problemen führen kann. Die Ausrüstungsgegenstände (Harpunen, Fässer, Kisten, Seile, Boote etc.) wurden in separaten Dateien konstruiert und im Nachhinein in die Hauptszene importiert. Wenn die Größeneinheit bei den Objekten nicht übereinstimmt, kann z.B. das Schiff statt 30 m plötzlich nur noch 3 mm lang sein. Zwar lassen sich Modelle auch nach dem Import skalieren, die Skalierung auf Objektebene beeinflusst aber die Transformationsmatrix, was zu einer falschen Interpretation von Position, Rotation und Größe des Objekts während der Animation und Simulation führen kann. Derartige Schwierigkeiten konnten durch eine konsequente Größeneinheit (1 Einheit = 1 cm) vermieden werden. Bei der Konstruktion polygonaler Modelle entstehen oft unnatürlich harte Kanten, die den Realitätseindruck vermindern. An vielen Objekten im Film konnte dieser Effekt abgemildert werden, indem die Modelle abgekantet, also durch zusätzliche Unterteilung abgerundet wurden.



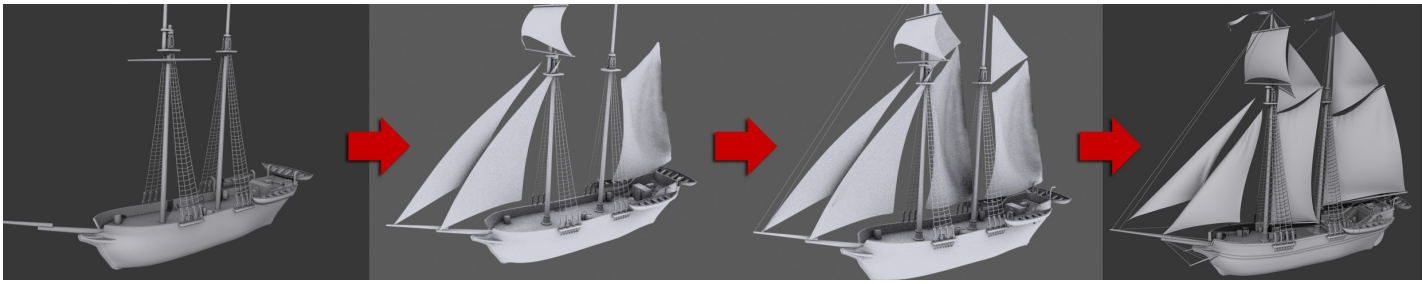
Ein NURBS-Modell (links) ist durch mathematische Gleichungen definiert. Polygonale Modelle, wie das Fass im rechten Bild, bieten mehr Kontrolle über die genaue Form, neigen jedoch zu unrealistisch harten Kanten.



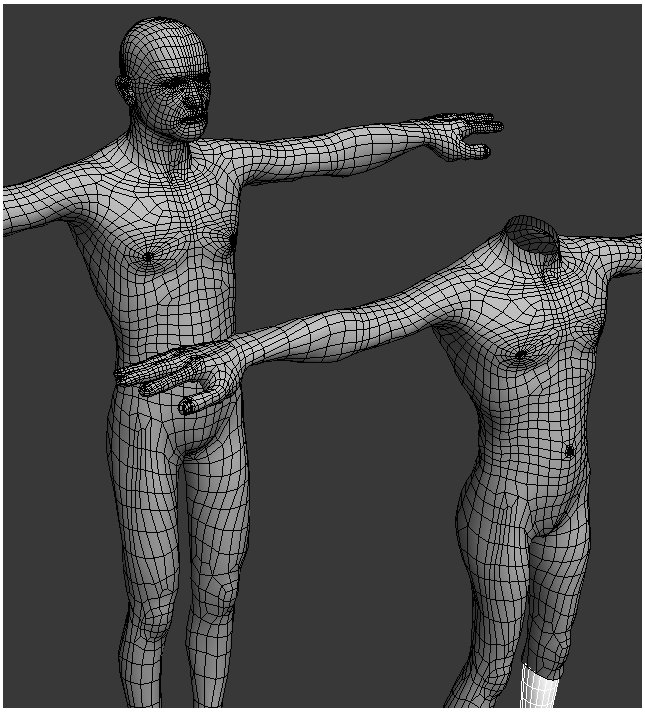
Diese Kanten lassen sich durch Unterteilung abrunden, und sehen somit realistische aus. Perfekt harte Kanten existieren in der Realität nicht.



Die Konstruktion des Schiffsrumpfs anhand von Referenzebenen.



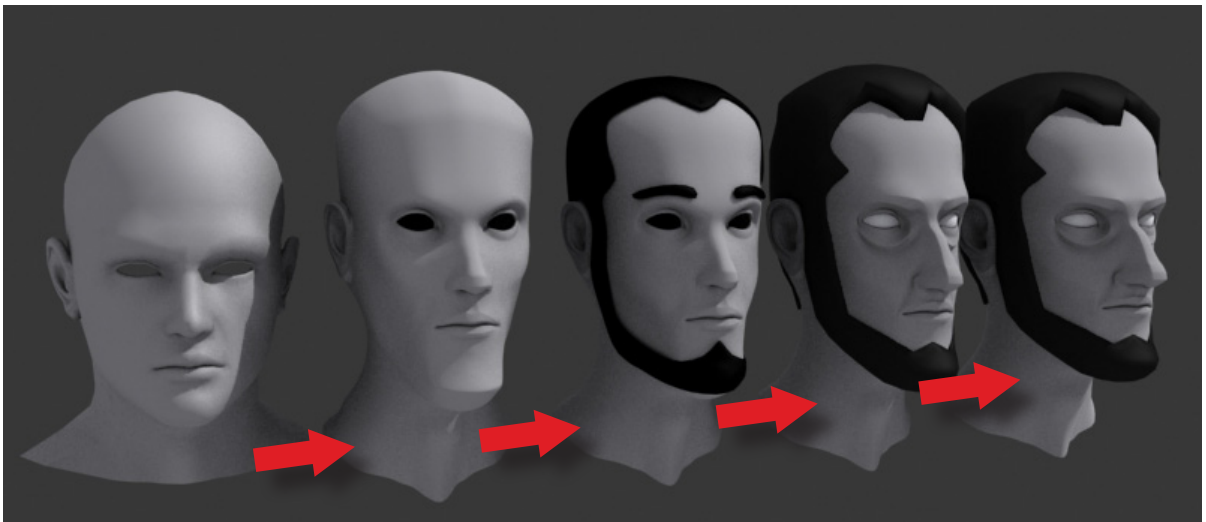
Die Entwicklung des Schiffsmodells vollzog sich über einen Zeitraum von ca. drei Wochen.



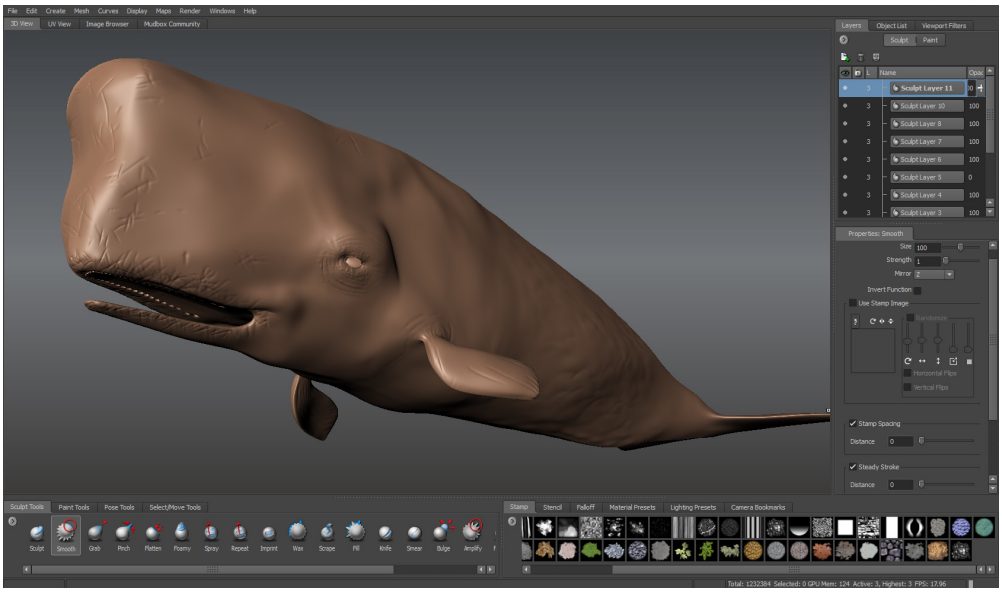
Links das aus makeHuman exportierte Modell, daneben die veränderte Kopie für Ahabs Körper.

Da die 3D-Figuren von Kapitän Ahab und Moby Dick animiert werden sollten, war der Aufbau der Modelle von besonderer Wichtigkeit. Ein Modell, das für die Animation gebaut wird, muss einen für die spätere Verformung geeigneten Verlauf der Gitterlinien besitzen. Dieser wird „Flow“ genannt, und richtet sich bei humanoiden Charakteren nach der menschlichen Anatomie. Wichtig ist dies vor allem in den Gelenkbereichen wie Schulter, Nacken, Kniekehle und Hüfte, in denen viel Bewegung stattfindet. Ein korrektes Körpermodell mit gutem Flow zu erstellen, erfordert anatomische Kenntnisse und viel Erfahrung in organischer Modellierung. Ich habe mich daher entschlossen, für den Körper von Ahab auf eine existierende Basis aufzubauen. Diese Entscheidung wurde durch den Umstand begünstigt, dass der größte Teil des Körpers ohnehin dauerhaft von Kleidung bedeckt sein würde. Das Open Source Programm „makeHuman“ ermöglicht die Manipulation eines 3D-Modells eines menschlichen Körpers mithilfe einfacher Schieberegler. Das Modell besitzt einen sehr guten Flow für die Animation, und ist trotz einer vergleichsweise geringen Polygonanzahl anatomisch gut definiert. Es lassen sich Faktoren wie Gewicht, Größe, Muskulosität und sogar das Geschlecht stufenlos regulieren. Ich habe einen kräftigen, aber ausgezehrten alten Mann erstellt und in eine Datei exportiert. Nach dem Import in 3ds Max wurden die Proportionen der Figur verändert (breiteres Kreuz, längere Arme und Beine, größere Hände und Füße), um durch leichte Abstraktion eine zu großes Maß an menschlicher Realität und damit das Problem des „Uncanny Valley“ zu vermeiden. Der Kopf des Modells wurde entfernt, um ihn durch einen später separat erstelltes Kopfmodell zu ersetzen.Für den Kopf von Ahab wurde ebenfalls ein frei verfügbares Modell als Ausgangsbasis genutzt, um den vorhandenen Flow zu übernehmen. Als Vorlage für die weitere Manipulation war die Zeichnung von Konstantin Krug von großem Wert. Der ursprüngliche Kopf wurde in zahlreichen Überarbeitungen individuell ausgestaltet, wie die untenstehende Grafik zeigt.Harte Gesichtslinien und asketische Proportionen sollen Ahabs Besessenheit verdeutlichen und gleichzeitig allzu großer Menschenähnlichkeit vorbeugen.

Die Entwicklung von Ahabs Kopf zeigt stark variierende Stadien des Modells.

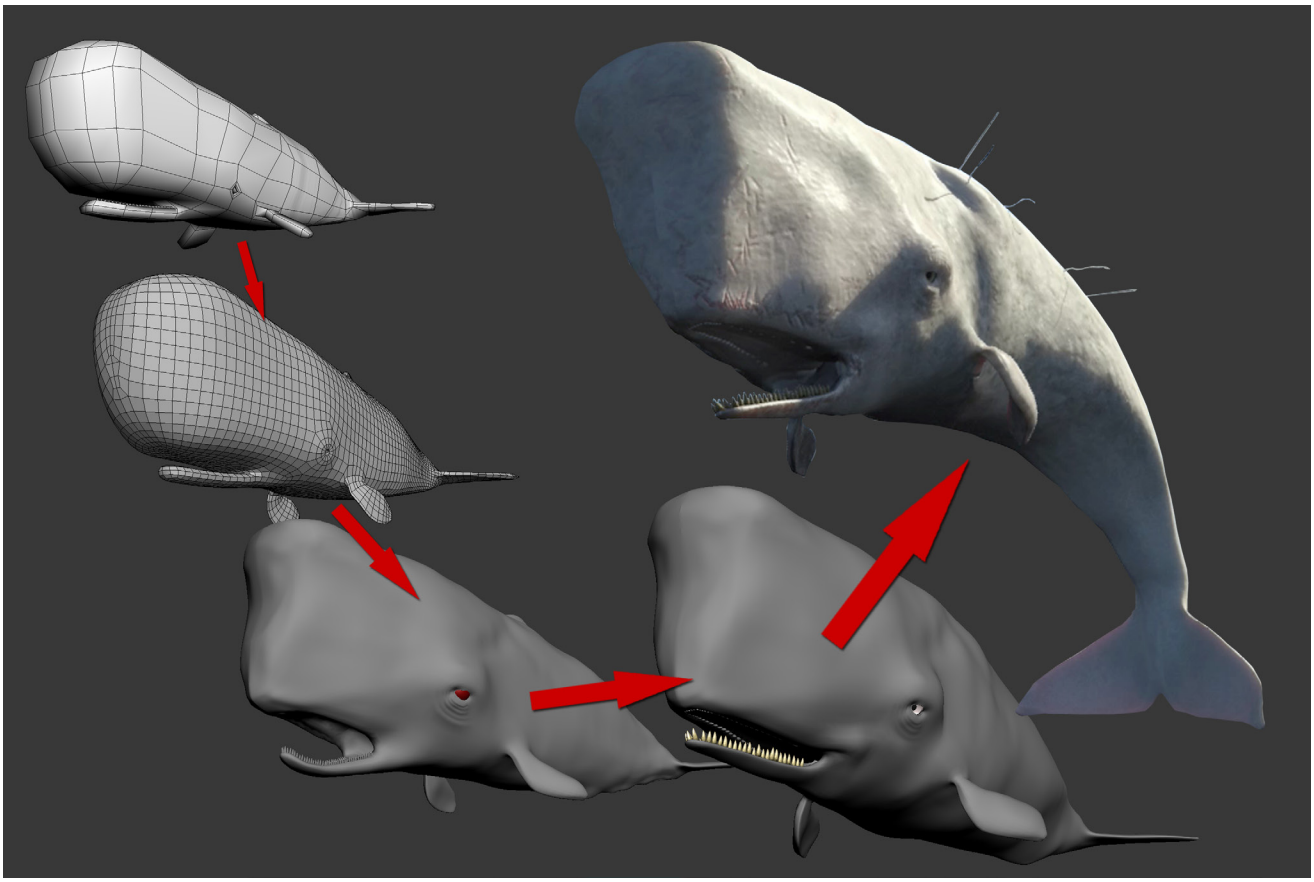


Auch das Modell des weißen Wals hat eine Entwicklung mit sehr unterschiedlichen Stadien durchlaufen. Für das Blocking wurde eine grobe Version von Moby Dick angefertigt, die vorerst nur zur Visualisierung der Geschichte gedacht war. Nach einigen erfolglosen Versuchen, das finale Walmodell von null an zu beginnen, stellte sich das Blocking-Modell als geeignete Basis heraus. Der Wal ist nicht nur in 3ds Max, sondern auch in der zusätzlichen Software Mudbox erstellt worden. Mudbox erlaubt durch eine entsprechend optimierte Programmierung das Arbeiten mit sehr hohen Polygonanzahlen, die in anderer Software durch starken Leistungsabfall nicht möglich wäre. Hierdurch ist eine sehr intuitive Art der organischen Modellierung möglich, die am ehesten mit dem Arbeiten mit Ton oder Modellierungsmasse vergleichbar ist.



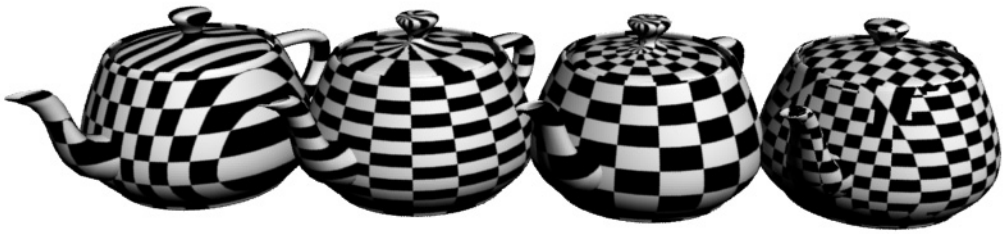
Ansicht der Arbeitsoberfläche in Mudbox. Das Walmodell besteht zu diesem Zeitpunkt aus über zwei Millionen Polygonen. Durch die hohe Auflösung lassen sich feine Details wie Narben oder Falten in das Modell einarbeiten. Um diese Änderungen auf das normal aufgelöste Ursprungsmodell zu übertragen werden Texturen exportiert. Diese sog. “Normal Maps” speichern Tiefeninformation und gaukeln später einen höheren Detailgrad vor.

Die Entwicklungsphasen von Moby Dick:



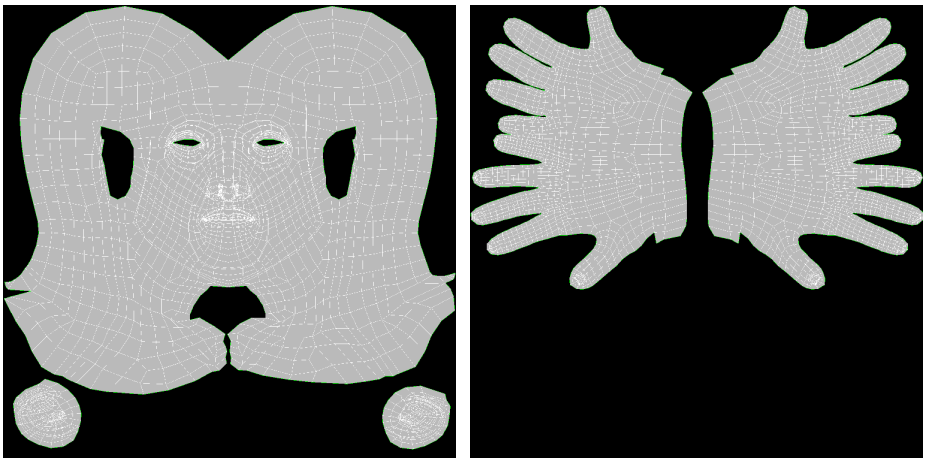
5.2 Texturing

Die 3D-Objekte sind nach der Modellierung noch grau und unrealistisch. Um ihnen Farbe und Struktur zu verleihen, müssen Texturen auf die Modelle aufgebracht werden. Eine Textur ist eine zweidimensionale Bilddatei, die Farbinformationen enthält und bei meinem Projekt meist auf Fotobasis erstellt wurde. Um eine zweidimensionale Textur korrekt auf ein dreidimensionales Objekt übertragen zu können, müssen Texturkoordinaten erstellt werden. Dies ist möglich durch eine simple Projektion oder durch manuelle Manipulation der Texturkoordinaten im sog. „Unwrapping“. Die Projektion ist vergleichbar mit einem Diaprojektor, der ein Bild in eine Richtung auf das 3D-Objekt wirft. 3ds Max bietet verschiedene Arten der Projektionen, z.B. planar, quaderförmig, zylindrisch und kugelförmig. Hiermit werden die meisten Grundformen von Objekten abgedeckt. Der Vorteil einer Projektion ist der geringe Zeitaufwand: Viele Objekte auf dem Schiff wurden daher mit dieser Methode texturiert. Ein Nachteil ist die geringe Präzision und Flexibilität, die sichtbare Kanten und verzerrte Texturen erzeugen kann.



Verschiede Projektionsarten am Modell einer Teekanne. Von links nach rechts: Planar, zylindrisch. kugelfürmig, quaderförmig.

Mehr Kontrolle bietet das Unwrapping, also das manuelle Abwickeln von Texturkoordinaten. Es bietet sich vor allem bei organischen und komplexen Modellen an, besonders wenn diese oft und aus der Nähe zu sehen sind und keine Fehler aufweisen dürfen. Das zugrundeliegende Prinzip lässt sich mit der Kürschnerei vergleichen, also dem Abziehen einer Tierhaut. Das 3D-Modell ist sozusagen in eine Haut aus Polygonen gehüllt, die beim Unwrapping abgezogen und flach ausgelegt werden muss, um eine zweidimensionale Textur korrekt übernehmen zu können. Eine Methode des Unwrappings wird „Pelt Mapping“ genannt, bezieht sich also direkt auf diesen Vergleich. Beim Pelt Mapping werden manuell Kanten im Modell ausgewählt, die als Schnittkanten dienen. Im nächsten Schritt wird die „Polygonhaut“ an ihren Enden auseinandergezogen, wobei sie sich an den Schnittkanten auseinanderfaltet. Eine abschließende Manipulation und Relaxation der Scheitelpunkte kann mögliche Überlappungen und Verzerrungen korrigieren. Da das Unwrapping deutlich zeitaufwendiger ist als eine einfache Projektion, wurde es nur bei entsprechend komplexen Objekten angewandt. Dazu zählen Ahab und der Wal, die Walfangboote, das Schiff und einige Ausrüstungsgegenstände. Nach dem Unwrapping wurde je eine Ansicht des abgewickelten Modells als Bild gespeichert und als Vorlage für die Texturierung in Photoshop genutzt-

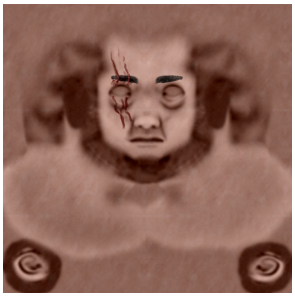


Das Unwrapping von Ahabs Kopf und den Händen kann als Bild herausgerendert und in einer Bildbearbeitungssoftware zur Vorlage für die Texturierung benutzt werden.

Die Texturen entstanden größtenteils zweidimensional in Photoshop, wobei in einigen Fällen auch Mudbox eingesetzt wurde. Mudbox erlaubt das Aufmalen und Projizieren von Texturen direkt auf das 3D-Modell, was bei Moby Dick hilfreich war. Fotos von Holz, Metall, Segeltuch und ähnlichen Materialien wurden zu Texturen kombiniert und an die Projektion bzw. das Unwrapping des jeweiligen Modells angepasst. Hierbei konnten mehrere Texturen mithilfe von Masken übereinandergelegt werden, um einen allzu einheitlichen Look zu vermeiden und Texturbilder wiederverwendbar zu machen.

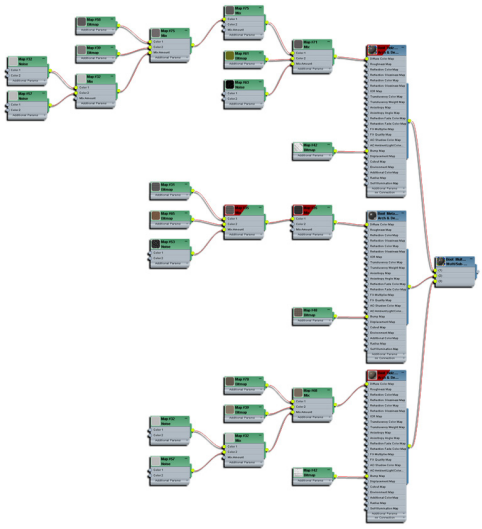
Texturen liefern grundsätzlich nur die Farbinformation, die eigentlichen Materialeigenschaften wie Struktur, Reflektion, Refraktion, Glanz oder Transluzenz werden durch den sog. „Shader“ berechnet. Ein Shader ist ein kleines Programm, das verschiedene Bilddateien, sog. „Maps“ kombiniert, um das Aussehen des Objekts festzulegen. Zu den Maps gehören nicht nur Texturen für die Farbe, sondern z.B. auch Masken für das Mischen mit anderen Shadern oder Schwarz-Weiß-Bilder als Tiefeninformation für die Berechnung der Oberflächenstruktur, die sog. „Bump Map“. Diese erzeugt z.B. bei Holz rillenförmige Vertiefungen, die das Material deutlich glaubwürdiger machen.

3ds Max bietet im Zusammenhang mit mental ray eine Vielzahl von spezialisierten Shadern. Ein Beispiel hierfür ist der „SSS Fast Skin Shader“, der bei Ahab und Moby Dick eingesetzt wurde. „SSS“ steht hier für „Sub Surface Scattering“ und beschreibt den Effekt der Lichtstreuung innerhalb eines Objektes. Der Effekt ist typisch für Materialien wie Wachs, Milch, Jade und eben auch die menschliche Haut. Der SSS Fast Skin Shader simuliert die Streuung des Lichts durch mehrere Hautschichten: Dies äußert sich in sanften Farbverläufen, rötlich durchschimmerndem Licht und weichen Schatten. Der Shader wurde nur subtil eingesetzt, da beim Rendern der Animation starkes Flackern auftreten kann. Wie untenstehende Abbildung zeigt, ergibt sich dennoch ein lebendigerer Gesamteindruck:



Die Textur von Ahabs Kopf beinhaltet Hautfarbe, Narben und Augenbrauen. Die Oberflächenstruktur wird mit einer modifizierten Schwarz-weißvariante dieser Textur erzeugt.

Obwohl Moby Dick der “weiße” Wal ist, wurden rote und grüne Farbtöne in seine Textur eingearbeitet, um eine subtile Variation zu erhalten.



Ein Shader kann simpel oder komplex sein. Die obere Grafik zeigt den Aufbau des Materials des Walfangbootes, das aus drei Untermaterialien (Holz, Metall, Algenbewuchs) mit jeweils mehreren Maps zusammengesetzt ist.

Der Vergleich zeigt links Ahabs Kopf ohne SSS Shader und rechts mit. Der linke Kopf wirkt blutleer und leblos. Durch Farbverlauf, Glanzlichter und weichen Schattenwurf verbessert der SSS Shader den visuellen Gesamteindruck, allerdings auf Kosten der Renderzeit.

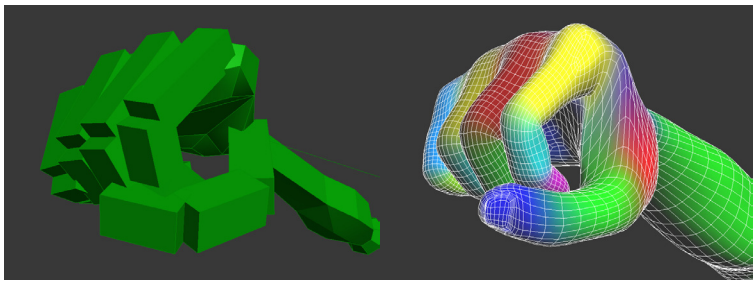
5.3 Lighting

Lichtstimmung und Intensität verändern sich durch ein Tageslichtsystem mit der Uhrzeit, da diese den Sonnenstand beeinflusst.

Da beinahe alle Szenen des Trailers am Tag und auf offener See spielen, war die Wahl einer Lichtquelle einfach. Bei der Beleuchtung der Szenerie im 3D-Programm habe ich als einzige Lichtquelle ein Tageslichtsystem benutzt. Dieses kombiniert zwei Lichtquellen, ein Sonnenlicht und ein Himmelslicht. Das Sonnenlicht ist eine direkte Lichtquelle, die durch ihre parallelen Lichtstrahlen die Sonne simuliert. Es ist durch seine starke Intensität und den markanten Schattenwurf maßgeblich für die Lichtstimmung im Bild verantwortlich. Das Himmelslicht ist eine indirekte Lichtquelle, die bläuliches, diffuses Licht mit geringer Intensität aus allen Richtungen wirft. Es dient zur Simulation des Streulichts, das in der Realität durch abprallende Lichtstrahlen zwischen Himmel und Erde entsteht. Das Tageslichtsystem bietet Möglichkeiten zur Veränderung von Sonnenstand, Intensität und Farbe des Lichts anhand von Positionsdaten oder einem konkreten Zeitpunkt. So lässt sich etwa, falls gewünscht, der Sonnenuntergang am 13. Mai 2007 um 19:27 Uhr in London simulieren. Derartig korrekte Daten waren in meinem Projekt nicht nötig, und ich habe stattdessen die Position der Sonne am Himmel nach ästhetischen Kriterien in jeder Kameraeinstellung manuell gesetzt. Da sie in das Tageslichtsystem eingebunden ist, wirkt der Sonnenstand sich auf die Färbung des Himmels und der Wolken, die generelle Helligkeit in der Szenerie und somit auf die vermittelte Stimmung aus. Das System generiert einen farbigen Hintergrundverlauf, der einen wolkenfreien Himmel simuliert. Um die Leere des Himmels mit Wolken zu füllen, habe ich eine zusätzliche Bilddatei benutzt, die in den Himmelsverlauf eingefügt wurde.
Das Tageslichtsystem ist darauf ausgelegt, in Verbindung mit globaler Illumination zu arbeiten. Globale Illumination beschreibt eine Methode zur Simulation von Licht im dreidimensionalen Raum, die an physikalische Gesetze angelehnt ist. In 3ds Max und dem von mir genutzten Renderer „mental ray“ funktioniert dies so, dass von jeder Lichtquelle aus kleine Partikel, genannt Photonen, abgefeuert werden. Diese Methode wird „Photon Tracing“ genannt, da die Flugbahn der Photonen verfolgt wird. Jedes Photon besitzt eine gegebene Lichtintensität und Farbe, abhängig von ihrer ursprünglichen Lichtquelle. Beim Aufprall auf Flächen oder Objekte im Raum verliert das Photon an Intensität und nimmt Farbinformationen des Kollisionsobjekts auf, wobei es (abhängig von den Rendereinstellungen) mehrfach abprallen und seinen Flug fortsetzen kann. Jeder Aufprall wird vom Renderer registriert und auf einer sog. „Lightmap“ gespeichert. Die Lightmap ist die Grundlage der Berechnung von Helligkeit und Farbe des Bildes. Sie wird zusätzlich verfeinert durch das sog. „Final Gathering“, eine Methode die im Kapitel Rendering näher beschrieben wird. Im Gegensatz zu einfacher, direkter Beleuchtung, simuliert Global Illumination indirektes Licht, das in der Realität überall existiert. Der Effekt äußert sich in mehrfacher Hinsicht: Unnatürlich dunkle Schattenbereiche werden aufgehellt und eingefärbt, da indirektes Licht auch an Stellen gelangt, die von der Sonne nicht beschienen werden. Die Farbe von Objekten beeinflusst das abprallende Licht und die Färbung der Umgebung (sog. „Color bleeding“).

Der zusätzliche Realismus erfordert eine deutlich erhöhte Berechnungszeit, wobei im Rendering ein idealer Kompromiss zwischen Renderzeit und Bildqualität gefunden werden muss. Hierbei war mir Nico Bauerschäfer behilflich, der diese Optimierung freundlicherweise für mich vorgenommen hat. Die Kombination von Tageslichtsystem und globaler Illumination ist eine simple und effektive Methode zur Beleuchtung von Außenszenen. In den einzelnen Einstellungen brauchte ich nur noch die Ausrichtung der Sonne zu verändern, um die Lichtstimmung festzulegen.

5.4 Charakter Setup

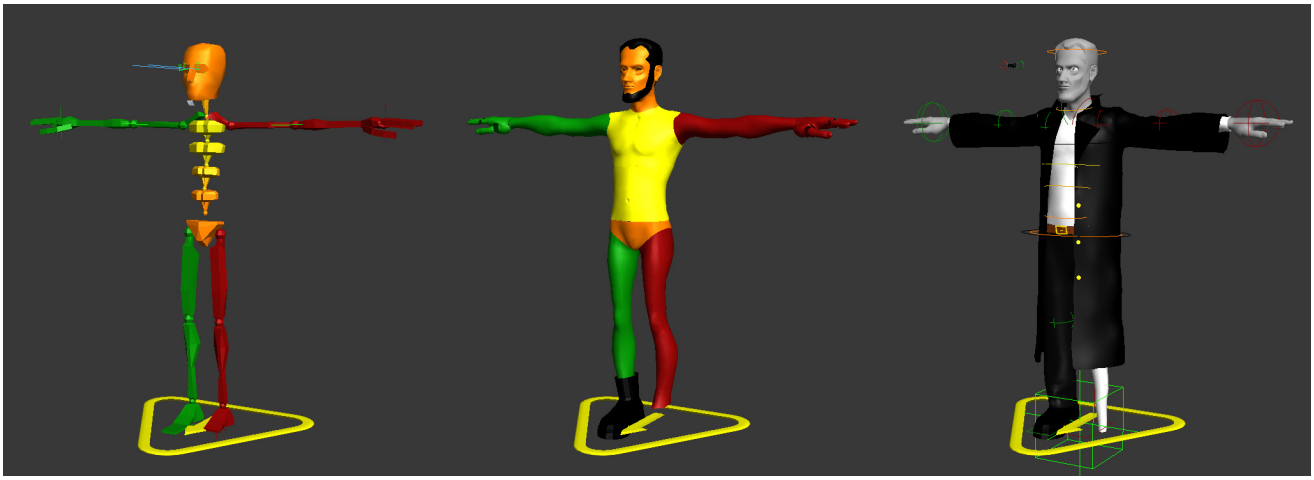


Die grünen Bones der Hand sind ein System aus einfachen Gelenken, die animiert werden. Das Modell wird im Skinning an die Bones gebunden. Das rechte Bild zeigt in farbigen Verläufen die Gewichtung der Fingerteile an den jeweiligen Bone.

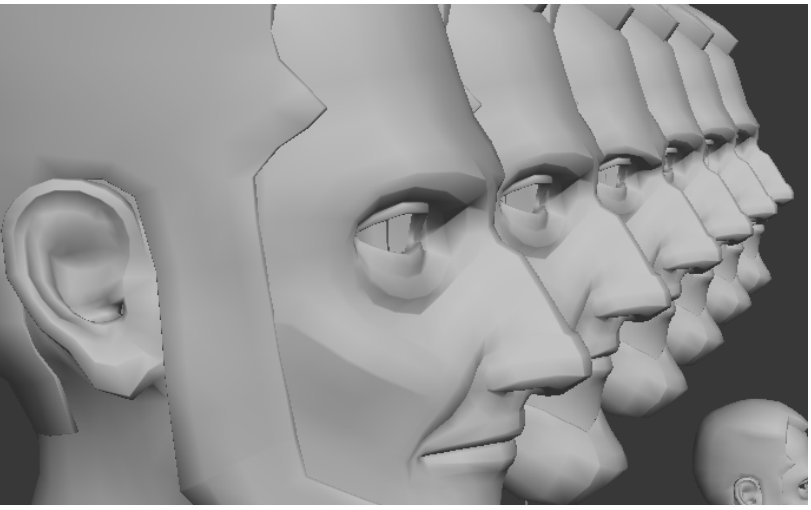
Das Charakter-Setup, auch „Rigging“ oder „Puppeting“ genannt, ist ein zwingender Schritt vor der Animation einer Figur. Allgemein gesprochen beinhaltet das Charakter-Setup das Erstellen von Animations- und Deformationssystemen für die Figur. Es ist vergleichbar mit der Konstruktion eines mechanischen Skeletts in der traditionellen Puppentrickanimation. Auch in der 3D-Software wird aus Hilfsobjekten, sog. „Bones“, ein Skelett erstellt, an dessen Gelenken sich der Charakter verformen soll. Die Platzierung der Gelenke ist hier von entscheidender Wichtigkeit und erfordert teilweise anatomische Kenntnisse. Ist das Skelett fertig, wird das Charaktermodell im Prozess des sog. „Skinnings“ mit dem Skelett verbunden. Das Skinning legt fest, welche Gitterpunkte des Modells an welchen Bone des Skeletts gebunden sind, und mit welcher Gewichtung er diesem folgt. Da mehrere Bones einen Gitterpunkt beeinflussen können, sind weiche Deformationen möglich und auch beabsichtigt, um die Bewegung von Haut und Muskulatur zu imitieren.

Aus früheren Projekten war mir der hohe Zeitaufwand für die vollständige Konstruktion eines Rigs bereits bekannt, daher habe ich mich eines Hilfsmittels bedient. Eine der Stärken von 3ds Max ist das umfangreiche Angebot von Plugins und Erweiterungen, die im Verlauf der Entwicklung auch teilweise aufgekauft und in die Software integriert werden. Seit der Version 2011 beinhaltet 3ds Max das frühere Plugin „Character Animation Tools“ oder kurz: CAT. Diese Werkzeugsammlung bietet vorgefertige Rigs, die vergleichsweise einfach an die Proportionen eines Charakters angepasst werden können. Das Körper-Rig von Ahab basiert auf einem vorgefertigten CAT-Rig, wurde jedoch zusätzlich erweitert. Hinzu kamen ein von Grund auf neu erstelltes System für die Gesichtsanimation und ein geskriptetes Interface zur vereinfachten Bedienung des Rigs während der Animation.

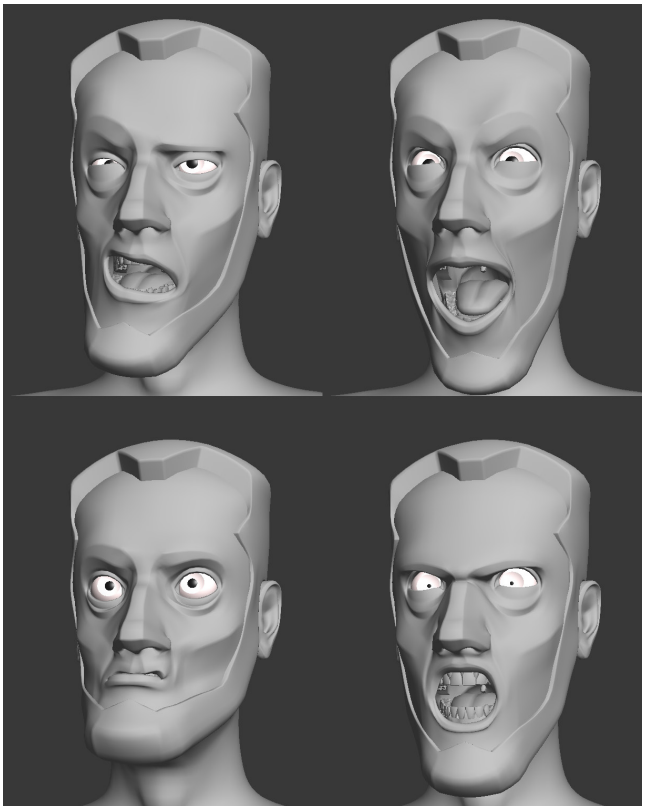
Das Körper-Rig von Ahab hat dank seiner CAT-Basis die Fähigkeit, Posen vom gesamten Körper oder einzelnen Körperteilen zu speichern und während der Animation wiederzuverwenden. Dies war vor allem bei der Animation der Hände und Finger hilfreich, und um die Figur in ihre Ausgangspose zu versetzen. Das CAT-Rig unterliegt zudem anatomisch korrekten Beschränkungen im Spielraum der Gelenke. Wird z.B. das Handgelenk in der Animation über einen bestimmten Winkel (ca. 90°) nach unten gedreht, rotiert der Arm automatisch mit, um jederzeit eine anatomisch halbwegs korrekte Pose zu gewährleisten. Diese Beschränkung ist nicht immer gewollt und lässt sich deaktivieren, ich empfand sie jedoch als hilfreich.



Die einzelnen Ebenen des Rigs von Ahab. Ganz links das Animationsskelett, das in der Mitte als sog. “Puppet” zu sehen ist. Eine Puppet ermöglicht eine gute Vorschau des Charakters während der Animation. Da sie jedoch wie eine Gliederpuppe an den Gelenken auseinandergeschnitten ist, muss keine rechenintensive Verformung wie beim Skinning erfolgen. Hierdurch ist die Leistung im Ansichtsfenster der Software verbessert und das Animieren liefert schnellere Rückmeldung. Rechts das vollständige Rig von Ahab mit Kleidung und Kontrollen.



Das Gesichtsrig von Ahab basiert auf ca. 40 Morph Targets.



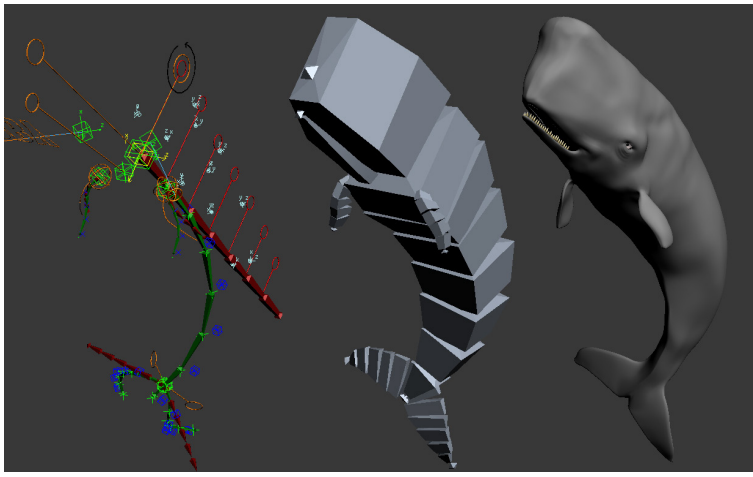
Durch Manipulation des Interfaces im Programm werden die Morph Targets gemischt und können so unterschiedlichste Gesichtsausdrücke entstehen lassen.

Das Gesicht von Ahab wird durch ein sog. „Blended Morph System“ gesteuert. Die genutzte Technik des Morphens unterscheidet sich hierbei vom Animationsskelett des Körpers. Beim Morphen wird ein Modell, in diesem Fall der Kopf von Ahab, durch Interpolation mit einem sog. „Morph Target“ verformt. Das Morph Target ist eine Kopie des ursprünglichen Modells, das nur verformt aber nicht mehr in der Anzahl oder Nummerierung seiner Gitterpunkte verändert werden darf. Morph Targets sind für gewöhnlich verformte Kopien eines Kopfes die bestimmte Gesichtsausdrücke, z.B. Wut, Angst, Freude oder Überraschung darstellen. Mithilfe des Morpher-Modifiers in 3ds Max lassen sich diese animieren und bewegen so durch Interpolation der einzelnen Ausdrücke das Gesicht. In dem bei Ahab benutzten System stellen die Morph Targets isolierte Bewegungsstadien einzelner Gesichtspartien dar, die für sich gesehen noch keinen Gesichtsausdruck liefern. Da insgesamt fast 40 Morph Targets stufenlos miteinander gemischt werden können, ergibt sich dennoch eine umfangreiche Variation in der Anzahl der möglichen Gesichtsausdrücke. Über eine grafische Benutzeroberfläche wird die Gesichtsanimation gesteuert, wobei auch hier das Speichern und Laden von Animationsposen für das Gesicht bzw. dessen Einzelteile (Augen, Mund) möglich ist.

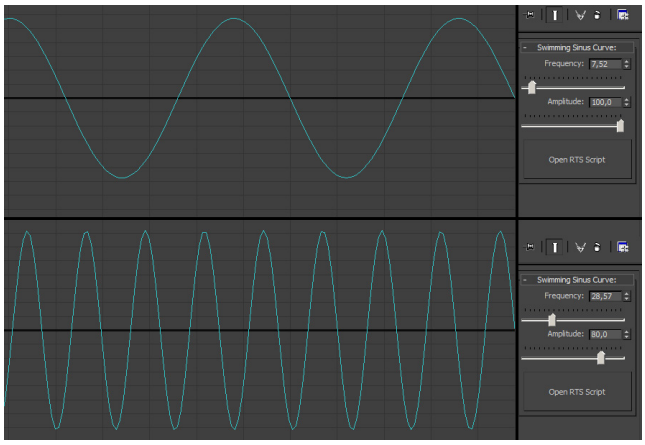
Die verschiedenen Kontrollebenen und Funktionen des Rigs wurden in einem geskripteten, frei verschiebbaren Interface zusammengefasst. Hierdurch wurde vor allem das Ein- und Ausblenden von Teilen des Rigs vereinfacht, sowie ein Verändern der Auflösung des Modells und das Setzen von Keyframes für das gesamte Rig ermöglicht.

Das Rig wird durch viele Kontrollobjekte gesteuert und animiert. Es besteht zudem aus mehreren Ebenen (Bones, Puppet, Kontrollen, 3D-Modell, Kleidung) die wechselhaft ein- oder ausgeblendet werden müssen, um übersichtlich mit dem Rig arbeiten zu können. Um diese Arbeit einfacher zu gestalten, wurde das in der nebenstehenden Abbildung zu sehende Interface konstruiert. Es ist schlank genug, um dauerhaft im Ansichtsfenster angezeigt zu werden. Die Ebenen des Rigs lassen sich an- und ausschalten oder durchsichtig anzeigen (“X-Ray-Modus”). Weitere Buttons steuern den Detailgrad des Modells, selektieren alle Animationskontrollen bzw. platzieren einen Keyframe für diese. Zudem kann von hier ein weiteres Skript, die “Ahab Cloth Tools” gestartet werden. Dieses wird in folgenden Kapiteln ausführlicher erläutert.

Das Rig des Wals bietet für die Animation sowohl die Möglichkeit der manuellen Manipulation, als auch eine prozedurale Steuerung. Für das Vorbeischwimmen des Wals an der Kamera bietet sich das prozedurale System an, wohingegen etwa für den Angriff mit der Schwanzflosse auf den Schiffsmast das manuelle System vonnöten war. Das Kernstück des Rigs ist die Wirbelsäule, die aus einer Kette von Bones besteht. An ihr hängen die Bones für die Seitenflossen und die Fluke (Schwanzflosse), sowie der Unterkiefer zum Öffnen des Mauls. Dieser grundlegende Knochenaufbau existiert dreimal: Das erste Skelett dient der manuellen Animation, in dem die Bones jeweils rotiert werden um die Pose des Wals festzulegen. Das zweite Skelett nutzt ein Skript von Harrison Yu namens „Realtime Spring“. Durch Rotation oder Translation des ersten Bones in der Wirbelsäule bewegt sich der gesamte Körper in einer verzögerten, flüssigen Bewegung mit. Dieses System ermöglicht in Verbindung mit einem Sinusgenerator die automatisierte Schwimmbewegung des Wals. Das dritte Skelett wird nicht selbst animiert und ist trotzdem entscheidend: An dieses ist das Modell des Wals geskinnt, es folgt also den Bewegungen des Skeletts. Dieses dritte Skelett ist mithilfe von sog. „Constraints“, also Rotations- und Positionsbeschränkungen, an die beiden anderen Skelette gebunden. Über ein geskriptetes Interface lässt sich die Gewichtung dieser Bindung beeinflussen, sodass der Wal entweder der manuellen Animation oder der Automatisierung folgt. Diese Gewichtung ist stufenlos möglich, sodass sich auch manuelle und prozedurale Animation mischen lassen. Zudem ist sie für die Wirbelsäule und jede Flosse einzeln einstellbar. Es mag also ausreichen, die Posen der Wirbelsäule festzulegen, und die Flossen und Fluke folgen automatisch in flüssiger Bewegung nach.



Die “Innereien” des Rigs von Moby Dick bestehen aus drei Knochensystemen. Links sind das rote, manuelle und das grüne, skriptgesteuerte Skelett zu sehen. Diese beeinflussen wahlweise das mittlere, blaue Skelett, welches das eigentliche 3D-Modell bewegt. In diesem Fall wird die Pose durch das grüne Skelett kontrolliert.



Je nach gewählter Amplitude und Frequenz wird eine unterschiedliche Sinusrotation erzeugt, die den Wal in eine schwimmende Bewegung versetzt.

Das prozedurale Skelett lässt sich an ein Hilfsobjekt binden, den Sinusgenerator. Dieses Objekt rotiert auf und ab, wobei der Rotationswert r durch eine simple programmierte Funktion generiert wird:

r = Sinus ((Konstante * Frequenz) / Amplitude)

Die Funktion resultiert in einer Bewegung, deren Entfernung vom Ausgangspunkt durch die Amplitude und deren Geschwindigkeit durch die Frequenz bestimmt wird. Frequenz und Amplitude wurden jeweils mit einem Interface verknüpft, um sie für die Animation zugänglich zu machen. Folgt das automatisierte Skelett dieser Sinusrotation mit passenden Werten, entsteht eine automatisierte, organische Schwimmbewegung. Diese imitiert die Schwimmweise des Pottwals, der sich durch Krümmung seines muskulösen Körpers durchs Wasser bewegt. Die Animation der Augenpartie wurde über ein simples Morphsystem ohne spezielles Interface ermöglicht.

5.5 Animation

Nachdem die Charaktere im Rigging beweglich gemacht wurden, konnten sie anschließend animiert werden. Die Animation in der Computergrafik hat viele Parallelen zur klassischen Zeichentrickanimation. Über den Verlauf der Zeit können für jeden Frame Posen der Figur als Keyframe gespeichert werden, vergleichbar mit einzelnen Zeichnungen im Zeichentrick. Liegen zwischen zwei Posen leere Frames, werden für diese vom Computer automatisch Zwischenbilder interpoliert, sodass eine Bewegung entsteht. Die Interpolation der Keyframes kann im Kurveneditor geändert werden. Die Interpolation zwischen zwei Werten, z.B. der Position des Arms, werden als Kurven angezeigt und sind mittels Tangentenanfasser manipulierbar. Damit lassen sich Beschleunigung oder Abbremsen einer Bewegung umsetzen, z.B. beim Aufprall der Schwanzflosse von Moby Dick auf den Mast des Schiffes. Neben der Charakteranimation mussten auch Objekte und Kamera bewegt werden. Die Kamera wurde ebenfalls mit Keyframes animiert, jedoch zusätzlich bei Bedarf durch prozedurale Animation unterstützt. Hilfreich war dies etwa bei Szenen in denen ein starkes Kamerawackeln erwünscht war. Die Position der Kamera ergibt sich dabei nicht ausschließlich aus dem Wert des Keyframes bzw. den interpolierten Werten, sondern zusätzlich durch einen prozedural generierten Wert, der durch einen sog. „Noise-Controller“ entsteht. Der Noise-Controller produziert für jeden Frame einen Zufallswert zwischen einem Minimal- und Maximalwert für die Positionsabweichung der Kamera. Die Frequenz dieser Werte lässt sich ebenfalls angeben, wodurch schnelle bzw. langsame Zufallsbewegungen entstehen, die wie reales Wackeln einer Handkamera wirken. Dieser Effekt wird u.a. auch bei der Kollision von Wal und Schiff angewendet. Um die Beschädigung des Schiffes zu visualisieren, wurden Teile der Bordwand und des Masts in hunderte Holzsplitter fragmentiert. Diese einzeln mit Keyframes zu animieren, wäre zeitlich nicht möglich und ineffektiv gewesen. Stattdessen wurde „PhysX“, ein System zur Physiksimulation, benutzt. PhysX eignet sich für sog. „Rigid-Body-Simulation“, also die Simulation von harten, nicht-verformbaren Objekten. Das Gegenstück hierzu ist die sog. „Soft-Body-Simulation“, die bei Simulation von z.B. Stoff und Kleidung eingesetzt wird und im folgenden Kapitel erläutert wird. PhysX ermöglicht die das Wirken von Kräften (Gravitation, Wind uvm.) und die Kollision zwischen Objekten mit verschiedenen physikalischen Eigenschaften (Metall, Holz, Gummi etc.). Der Simulationsaufwand richtet sich nach der Anzahl der Kollisionsobjekte und der zu simulierenden Zeitspanne. Da eine Simulation durchaus viele Stunden dauern kann, wurden die PhysX-Parameter in einer simplen Szene vorher getestet, um bei der finalen Szene die gleichen Einstellungen verwenden zu können und nicht mehrfach simulieren zu müssen. Bei der Simulation der Holzsplitter traten teilweise enorme Geschwindigkeiten auf, die bei der zugehörigen Kollision unrealistisch wirkten. Um die Bewegung aller Splitter gleichzeitig und gleichmäßig strecken zu können, habe ich auf das „Point-Cache-Format“ zurückgegriffen. Ein Point Cache erlaubt das Speichern von Positionsdaten von Gitterpunkten in einem 3D-Modell über den Verlauf einer Zeitspanne. Für jeden Gitterpunkt wird also pro Frame der Animationszeit eine Position in eine externe Datei gespeichert. Damit entfallen die durch die Simulation erzeugten Keyframes – die Abspielgeschwindigkeit des Caches lässt sich über einen einfachen Prozentwert regeln. Mithilfe eines eigenen Skripts konnte ich die Geschwindigkeit aller Holzsplitter gleichzeitig verändern und so einen idealen Wert finden.

Simulationstests des zersplitternden Masts. Da der Effekt auch ohne das Modell des Schiffs ausprobiert werden kann, ist die Simulationsgeschwindigkeit deutlich beschleunigt.

5.6 Skriptprogrammierung

5.6.1 Einleitung

3ds Max besitzt die Möglichkeit (wie viele Programme mit einer vergleichbaren Komplexität) mithilfe einer Skriptsprache gesteuert zu werden. Diese unterscheidet sich von einer klassischen Programmiersprache vor allem dadurch, dass sie in ein Programm eingebunden ist und zur alternativen Bedienung des Programms dient. Im Vergleich zu Software wie Adobe Photoshop oder After Effects, die auf JavaScript und entsprechende Erweiterungen zurückgreifen, wurde für 3ds Max eine eigene Skriptsprache namens “MAXScript” entwickelt. Sie wurde 1996 von John Wainwright im Auftrag des damaligen Entwicklers discreet entworfen und im Verlauf der Jahre stetig weiterentwickelt. MAXScript ist objektorieniert, d.h. Daten und Funktionen werden zu Objekten zusammengefasst und ein Programm ist definiert als die Menge seiner interagierenden Objekte. Zudem ist MAXScript interpretativ, der Quellcode wird also direkt Zeile für Zeile ausgeführt. Dies ist ein wichtiges Merkmal von Skriptsprachen und unterscheidet sie von den klassischen Programmiersprachen. Ein in C++ entwickeltes Plugin muss kompiliert werden um in 3ds Max getestet zu werden. Der zeitliche Aufwand und die reine Schreibarbeit sind zudem in C++ deutlich aufwendiger, da nicht wie in MAXScript die Basisapplikation bereits vorhanden ist. Zwar ist auf Prozessorebene die zeilenweise Ausführung einer Skriptdatei grundsätzlich langsamer als das Ausführen eines kompilierten Plugins - bei vielen Anwendungen wirkt sich dieser Unterschied aber kaum aus, zudem steht er der kurzen Entwicklungszeit gegenüber. MAXScript ist somit am besten geeignet für schnelle, maßgefertigte Lösungen innerhalb von 3ds Max. Grundsätzlich lässt sich mit MAXScript fast jede Aufgabe in 3ds Max, die manuell erledigt werden kann, auch mit einem oder mehreren entsprechenden Befehlen erledigen. Zusätzlich ist die Sprache mächtig genug, um Plugins (z.B. eigene Geometrieobjekte) zu programmieren und rudimentäre Interaktion mit Drittprogrammen (z.B. Photoshop, Excel) oder dem Internet zu ermöglichen. Da sie speziell für 3D-Grafik und Animation entworfen wurde, lassen sich sogar kleine Echtzeitanwendungen mit Kollisionsberechnung und Tastatursteuerung schreiben. So existieren Varianten bekannter Computerspiele wie Mine Sweeper, Bomberman oder Space Invaders als Skript für 3ds Max, die auf ungewöhnliche Art die Möglichkeiten der Sprache demonstrieren.

Abb. 25

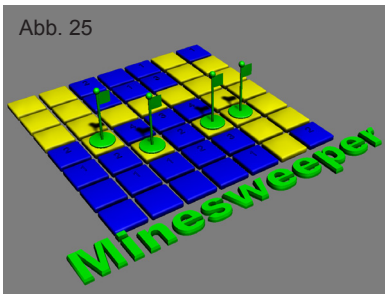


Abb. 26

MAXScript ist in der Lage, Echtzeitanwendungen zu konstruieren. Dazu gehören beispielsweise “MineSweeper von Håvard Schei, oder “MaxJack”, eine Blackjack Variante für 3ds Max von Jeff Hanna.

5.6.2 Syntax

Die Syntax von MAXScript lehnt sich an die Programmiersprachen C und C++ an, ist aber benutzerfreundlicher gestaltet und mit weniger syntaktischen Einschränkungen versehen. So spielt etwa die Groß- und Kleinschreibung im Quellcode keine Rolle bei der Interpretation, empfiehlt sich aber dennoch aufgrund gesteigerter Lesbarkeit. Dank eindeutiger Namensgebung der Befehle liest sich der Quellcode beinahe wie geschriebenes Englisch. Kommentare werden durch „--“ gekennzeichnet und während der Laufzeit vom Interpreter ignoriert. Variablen sind typenfrei, müssen somit nicht zu Programmbeginn explizit deklariert werden und können im Verlauf des Skripts wechselnde Werte verschiedenen Typs annehmen (z.B. Zahlenwerte, Text, Objektreferenzen). Ebenso ist eine simple Typenkonvertierung möglich, z.B.:

1. Der Variable „theString“ wird der Wert 123 zugewiesen. Die Anführungszeichen vor und nach der Zahl weisen diese als String, also Text aus. Die Addition von Text und Zahl ist nicht möglich:

```
theString = “123”
theString + 15
--Ausgabe: Unable to convert: 15 to type: String
```

2. Eine Typenkonvertierung ermöglicht die Addition beider Zahlen:

```
theString = “123”
(theString as Integer) + 15
--Ausgabe: 138
```

Ob Variablen lokal, also nur in einem durch Klammern abgetrennten Codeblock, oder global auch außerhalb des Quellcodes existent sind, ergibt sich automatisch durch den Ort ihrer ersten Deklaration. Folgendes Beispiel illustriert die Thematik:

1. Die Variable “theString” wird ausserhalb eines Codeblocks, also global deklariert.
Somit kann sie von überall in 3ds Max angesteuert und mit dem Befehl “print” ausgegeben werden.

```
theString = “Hallo Welt!”
print theString
--Ausgabe: “Hallo Welt!”
```

2. Die Variable “theString” wird innerhalb eines Codeblocks, also lokal deklariert.
Somit ist sie ausserhalb des Codeblocks unbekannt und der “print”-Befehl scheitert mit einer Fehlermeldung.

```
(
        theString = “Hallo Welt!”
)
print theString
--Ausgabe: undefined
```

Die lokale Deklaration innerhalb eines Codeblocks kann mit dem Zusatzwort „global“ überschrieben werden, wodurch die Variable auch außerhalb ausgegeben werden kann:

```
(
        global theString = “Nun aber doch!”
)
print theString
--Ausgabe: “Nun aber doch!”
```

Um die sog. Sphäre, also den Wirkungsgrad einer Variablen auf den ersten Blick sichtbar zu machen, hat es sich als gute Praxis erwiesen, den Zusatz „lokal“ oder „global“ für alle Variablen zu verwenden. MAXScript verfügt über ein dynamisches Speichermanagement, das im Hintergrund abläuft und per sog. „Garbage Collector“ unbenötigte Variablen und Speicherressourcen automatisch freigibt. Lokale Variablen und ihr Inhalt werden gelöscht, sobald der umgebende Codeblock oder das Skript beendet werden. Globale Variablen bleiben für den Verlauf einer Sitzung, also bis zum Beenden von 3ds Max erhalten. Durch diese Mechanismen ist die Speicherverwaltung von Skripten selten ein Problem und muss nicht manuell gesteuert werden.Viele Aufgaben können über wenige Befehle in einer Zeile Code ausgeführt werden. Schleifen und Bedingungsabfragen können intelligente Auswahlen treffen und eine Vielzahl an Objekten bearbeiten. Ein Beispiel zur Selektion aller Objekte in der Szene liest sich wie eine englische Anweisung:

```
select objects
```

Um nur polygonale Modelle innerhalb der Objekte zu selektieren, lässt sich der Befehl konkretisieren:

```
for obj in objects where classof obj == Editable_Poly
        do selectMore obj
```

Mittels einer for-Schleife wird hier die Klasse jedes Objekts überprüft. Eine Klasse bestimmt den Typ eines Objekts, z.B. Geometrie, Licht, Kamera oder Hilfsobjekt. Stimmen Objektklasse und gesuchte Klasse überein, wird das Objekt der Selektion hinzugefügt. Ein weiteres Beispiel:

```
for obj in selection do obj.material = undefined
```

Jedes selektiert Objekt bekommt als Material den Wert „undefined“ zugewiesen. Dieser ist gleichbedeutend mit „NULL“ oder „void“ in anderen Programmiersprachen und dient als leerer Platzhalter. Effektiv entfernt damit die Skriptzeile alle Materialien von den selektierten Objekten.

Die farbliche Kodierung des Quellcodes trägt zur Übersicht und Lesbarkeit bei. An der Farbe lässt sich die Bedeutung eines Wortes erkennen:

```
-- Reservierte Schlüsselwörter:
if, then, else, do, for, where

--Objektklassen:
Sphere, Box, Point, Editable_Poly

--Selektionssets:
Geometry, Objects, Lights, Selection

--Strings:
“Call me Ishmael”

--Numerische Werte:
0, 1, 3, 99.06, 10517

--Referenzen zu Szenenobjekten:
$Teapot001, $AHAB_Hemd, $Bone_Skin_001
```

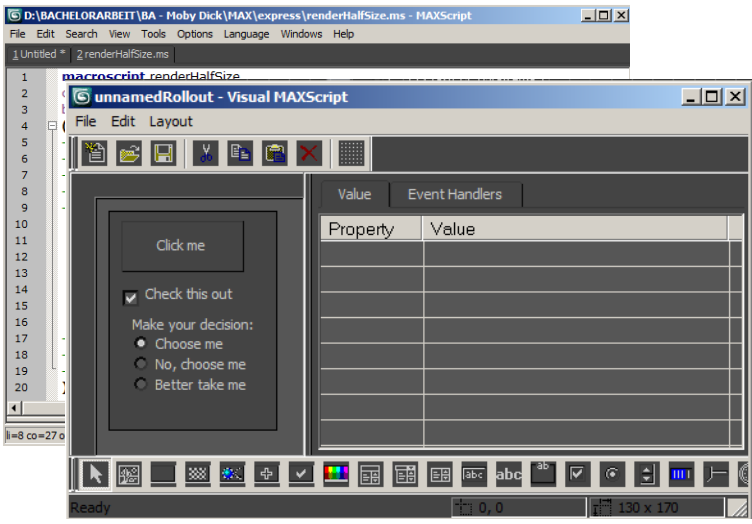
5.6.3 Entwicklungsumgebung

3ds Max wird mitsamt einer simplen, aber genügenden Entwicklungsumgebung ausgeliefert, die im Wesentlichen aus drei Teilen besteht:

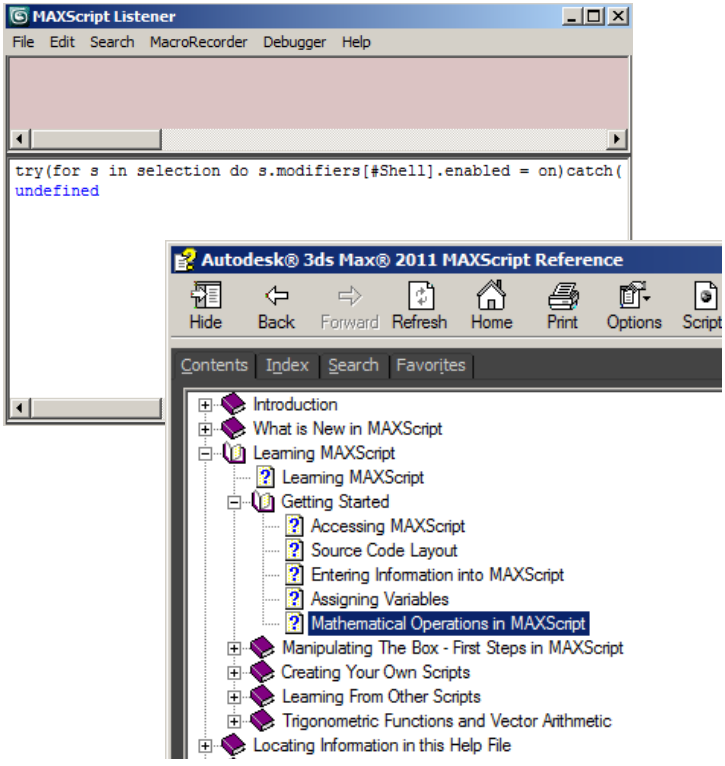
1. MAXScript Editor
Hier entsteht das eigentliche Skript. Hauptfunktion ist vor allem die Farbkodierung der einzelnen Befehle, sowie verschiedene Werkzeuge zur Formatierung, z.B. um Teile des Quellcodes schnell auszukommentieren. Zudem gibt es einen "Visual MAXScript Editor", also die Möglichkeit, die Benutzeroberfläche des jeweiligen Skripts (Knöpfe, Regler, Eingabefelder etc.) grafisch darzustellen und zu manipulieren. Dies trägt wesentlich zur schnellen Programmierung der Skripte bei.

2. MAXScript Listener
Der sog. „Listener" ist im Wesentlichen zur Ausgabe von Informationen zuständig. Er liefert direkte Rückmeldung beim Ablauf von Skriptdateien, seien es Variablenwerte, selbstgeschriebene Rückmeldungen oder Fehlermeldungen, die zur späteren Korrektur von hohem Nutzen sind. Eine weitere Funktion ist der "Macro Recorder". Sobald diese eingeschaltet ist, versucht der Listener, jede Aktion, die in 3ds Max per Hand ausgeführt wird, in eine Skriptzeile auszugeben, die diese Aktion reproduziert. Somit kann man einfache Aufgaben einmalig manuell ausführen, den ausgegebenen Code kopieren und mittels einer entsprechenden Schleife z.B. an einer Vielzahl von Objekten durchführen - der einfachste Weg um eine Automatisierung zu schreiben. Der Listener kann zudem als Eingabefenster genutzt werden, d.h. man kann kurze Codeschnipsel eingeben und ausführen lassen. Für gewöhnlich werden Editor und Listener zusammen genutzt, um schnell Skriptzeilen zu generieren und zu testen, Rückmeldung zu erhalten und dann im Editor die Einzelteile zusammenzubauen.

3. MAXScript Reference
Die MAXScript-Hilfedatei ist eine hervorragende Informationsquelle. Sie beinhaltet alle Informationen und Referenzen, die zum Schreiben von Skripten von variierender Komplexität benötigt werden, darunter eine umfangreiche Liste aller möglichen Befehle, sowie Lerneinheiten und Beispielskripte. Durch ihre klare Struktur, den großen Umfang und die intelligenten Querverweise trägt sie maßgeblich bei zu der kurzen Zeitspanne, in der Skriptdateien geschrieben werden können.
Mithilfe dieser Werkzeuge lassen sich innerhalb von 3ds Max sehr schnell Skripte entwickeln, teilweise in wenigen Minuten.



Im Editor entstehen mithilfe des Visual MAXScript Editors schnell angepasste Benutzeroberflächen für die Bedienung von Skripten.



Der Listener liefert Fehlermeldungen und RÜckgabewerte bei der Laufzeit von Skripten. Wenn Probleme auftreten, lässt sich in den allermeisten Fällen eine Lösung innerhalb der Hilfedatei finden.

Ein Skript lässt sich auf mehrere Arten benutzen. Der Quellcode lässt sich zeilenweise oder komplett direkt aus dem Editor, Listener oder über das MAXScript Menü ausführen. Speichert man die Skriptdatei in den Startup Ordner der Installation von 3ds Max, wird das Skript bei jedem Programmstart ausgeführt. Für wiederholte und schnellere Nutzung empfiehlt sich die Installation als Makroskript. Makroskripte lassen sich (ähnlich wie Plugins) in Form von Buttons, Mausmenüs oder Tastenkürzeln an vielen Stellen in die Benutzeroberfläche von 3ds Max integrieren. Sie sind die am meisten verbreitete Skriptform, auch weil es nur einen minimalen Zusatzaufwand bedeutet, ein normales Skript in ein Makroskript zu verwandeln:

```
macroscript myMacro
category:"myCategory"
(
	-- Das normale Skript befindet sich in diesem Codeblock
)
```

5.6.4 Quelltextanalyse

Anhand eines einfachen Beispiels möchte ich im Folgenden ein typisches MAXScript im Detail erläutern. Das Simulationssystem PhysX, dass zur Animation der Holzsplitter benutzt wurde, war auf den Rechnern der Renderfarm nicht installiert. Da PhysX Keyframeanimation generiert, war es für das Rendering glücklicherweise nicht vonnöten. Um Renderunterbrechungen mit Backburner aufgrund von Fehlermeldungen zu vermeiden, musste ich die verbleibenden PhysX-Modifer in den Szenen löschen. Bei hunderten Holzsplittern und somit hunderten Modifiern war dies effektiv mit einem Skript zu lösen:

```
------------------------------------------------------------
--find and delete all PhysX modifiers in the scene:
fn delete_all_missing_modifiers=
(
	local counter = 0
	for o in objects do
	(
		if o.modifiers.count > 0 do
		(
			for m=o.modifiers.count to 1 by -1  do
			(
				if classof o.modifiers[m] == Missing_OSM do
				(
					deleteModifier o m
					counter += 1
				)--end  if
			)--end  for
		)--end  if
	)--end for
	messagebox (counter as String + " missing modifiers have been deleted!") title:"Result:"
)--end fn
delete_all_missing_modifiers()
------------------------------------------------------------
```

Das Skript wird im Folgenden zeilenweise analysiert:

fn delete_all_missing_modifiers=
Deklaration einer Funktion mit Namen “delete_all_missing_modifiers”. Mittels einer Funktion kann ein Codeblock oder Skript in einen Wert gespeichert werden.D ie Funktion kann von anderen Skripten aufgerufen werden und eignet sich somit zur Strukturierung des Quellcodes, indem wiederholt benutzte Programmteile nicht ständig neu ausgeschrieben werden müssen. Stattdessen wird nur die Funktion aufgerufen, wodurch der enthaltene Code ausgeführt wird.

```
local counter = 0
```
Eine lokale Zählvariable wird deklariert und auf Null gesetzt. Diese dient der Rückmeldung über die Anzahl der gelöschten Modifier zu Ende des Skripts.

```
for o in objects do
```
Die erste for-Schleife iteriert nacheinander über alle Objekte in der Szene.

```
if o.modifiers.count > 0 do
```
Wenn das aktuelle Objekt einen oder mehr Modifier besitzt, fahre fort. Wenn nicht, stoppe das Skript und gehe zum nächsten Objekt. Diese Abfrage dient der Stabilität des Skripts. Hat das Objekt keine Modifier und versucht das Skript im Anschluss dennoch auf diese nicht vorhandenen Modifier zuzugreifen, resultiert dieser Versuch im Absturz des Skripts.

```
for m=o.modifiers.count to 1 by -1  do
```
Die zweite for-Schleife iteriert über die Modifier des Objekts und wird begrenzt auf deren Anzahl. Dies geschieht wieder aus Gründen der Stabilität, um einen Skriptabsturz zu vermeiden. Hat das Objekt fünf Modifier, wird die Schleife fünf mal ausgeführt. Der Zusatz “by -1” erzwingt eine umgekehrte Reihenfolge. Dies liegt an der Nummerierung der Modifier eines Objekts, die bei sich bei Löschen eines Modifiers verändert. Die umgekehrte Reihenfolge stellt sicher, dass kein Modifier übersehen wird.

```
if classof o.modifiers[m] == Missing_OSM do
```
Der durch den Wert von “m” indizierte Modifier wird auf seine Klasse untersucht. Ein PhysX Modifier wird bei fehlender PhysX-Installation als “Missing_OSM” deklariert. Der Abgleich stellt sicher, dass nur solch fehlende Modifier gelöscht werden.

```
deleteModifier o m
```
Der aktuelle Modifier des aktuellen Objekts wird gelöscht. Die Indizes “o” und “m” entstammen der ersten und zweiten for-Schleife.

```
counter += 1
```
Der Wert der Zählvariable wird um eins erhöht. Diese Addition findet nur jeweils nach Löschen eines Modifiers statt. Die benutzte Schreibweise ist eine Kurzform für: “counter = counter + 1”

```
messagebox (counter as String + " missing modifiers have been deleted!") title:"Result:"
```
Nach Beendigung des Skripts wird ein Fenster mit der Überschrift “Result:” geöffnet, um Rückmeldung über die Menge der gelöschten Modifier zu geben. Die Zählvariable kann durch eine Typenkonvertierung als Text ausgegeben werden.

```
fn delete_all_missing_modifiers=
(
        ...
)--end fn
```
Die Funktion wird deklariert und ist global in 3ds Max verfügbar. Hierdurch findet noch keine Aktion statt.

```
delete_all_missing_modifiers()
```
Die Funktion wird ausgeführt. Erst jetzt werden die Modifier gesucht und gelöscht.

5.6.5 Skriptkategorien und Beispiele

Im Folgenden möchte ich drei Kategorien von Skripten vorstellen und diese jeweils mit einigen während des Projekts entstandenen Beispielen illustrieren:

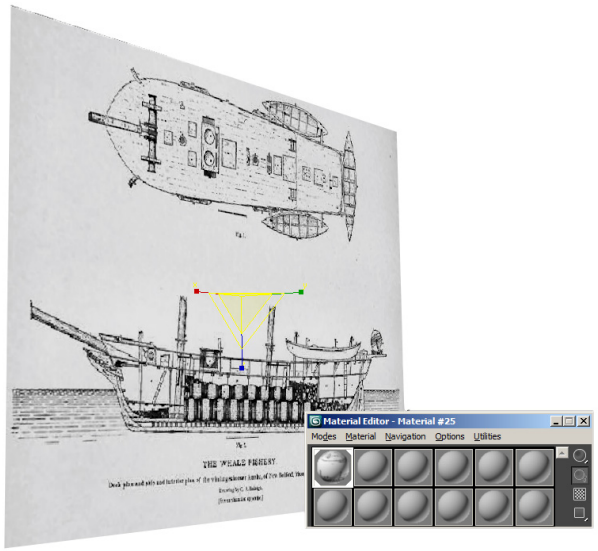
1. Shortcuts

Ein „Shortcut“ stellt eine Abkürzung für Arbeitsschritte in 3ds Max dar, die über eine angepasste Benutzeroberfläche oder das Zusammenfassen von mehreren Schritten in einen einzigen Befehl umgesetzt wird. Auch die manuelle Eingabe einer kurzen Codezeile kann unter diesen Begriff fallen. Beispiele:

Name: timelineToolbar
Funktion: Schnelle Änderung der Zeitleiste
Erläuterung: Da die Kameraeinstellungen während des Trailers unterschiedlich lang sind, musste ich während der Animation wiederholt die Länge und Position der Zeitleiste in 3ds Max ändern. Dies ist mithilfe von Tastenkürzeln und Mausbewegung möglich (ungenau), oder über ein Untermenü (unübersichtlich, umständlich). Das timelineToolbar-Skript ist ein Beispiel für ein „Custom Interface“, also eine geskriptete Bedienoberfläche. Um sich in das bestehende Interface zu integrieren, wurden alle unnötigen Menüelemente entfernt. Mittels dreier Schieberegler können Start- und Endpunkt der Zeitleiste, sowie die Position des Abspielkopfs schnell geändert werden. Eine präzise Eingabe der Positionen wird durch zusätzliche Eingabefelder ermöglicht. Durch seine minimalen Ausmaße kann das Skript dauerhaft geöffnet bleiben.

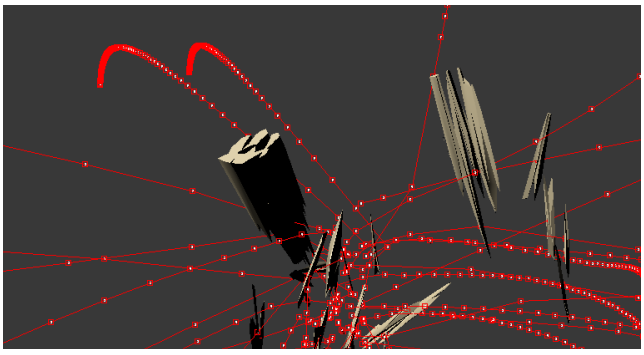
Die Timeline Toolbar integriert sich durch ihr minimales Design sehr gut in die bestehende Benutzeroberfläche von 3ds Max.

Name: createReferencePlane
Funktion: Erstellen von Referenzebenen für die Modellierung
Erläuterung: Die Modellierung der Gegenstände erforderte den Import von Referenzbildern in 3ds Max. Der Arbeitsablauf beinhaltet mehrere Schritte, die immer gleich ablaufen: 1. Navigation zur Bilddatei, 2. Laden der Datei in ein neues Material, 3. Hinzufügen des Materials auf eine neu erstellte Ebene, 4. Genaue Skalierung der Ebene an die Größe der Bilddatei. Die Skalierung erwies sich als umständlich und der gesamte Prozess war sehr repetitiv. Das Skript ermöglicht, die Bilddatei mit der Maus in 3ds Max zu ziehen, und anschließend automatisch eine Ebene mit passender Skalierung und bereits zugeteiltem Material zu erstellen. Die Konstruktion von Referenzebenen nahm somit nur noch wenige Sekunden in Anspruch.



createReferencePlane erstellt in sekundenschnelle Vorlagen für die Modellierung.

Name: switchTrajectories
Funktion: Anzeige von Trajektorien
Erläuterung: 3ds Max bietet nativ keinen Weg, die Anzeige von Trajektorien (Bewegungslinien eines animierten Objekts) für mehr als ein Objekt zu aktivieren bzw. deaktivieren. Das Skript erstellt zwei Buttons, um dieses zu ermöglichen. Es kam vor allem bei der



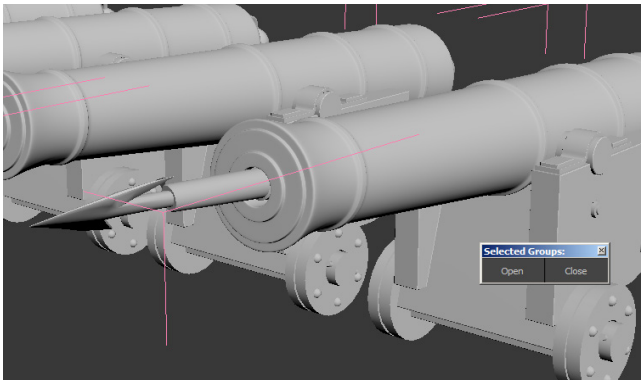
Die roten Linien stellen die Trajektorien, also den Bewegungspfad der Holzsplitter dar.

Korrektur der animierten Holzsplitter zum Einsatz, um deren Flugbahn und fälschliche Durchdringung von Kollisionsobjekten verfolgen zu können.

2. Automatisierungen

Automatisierungen dienen zur maschinellen Abwicklung repetitiver Prozesse. Wenn ein Arbeitsschritt üblicherweise an einer Mehrzahl von Objekten manuell ausgeführt wird, kann dieser eventuell automatisiert werden. Hunderte oder tausende von Prozessen können so in kurzer Zeit abgearbeitet werden. Beispiele:

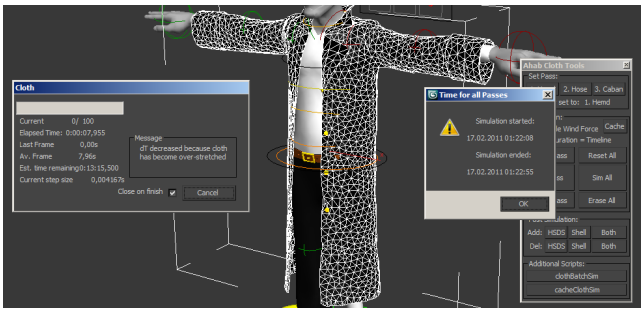
Name: openGroups
Funktion: Öffnen bzw. Schließen von beliebig vielen Gruppen
Erläuterung: Zur Organisation zusammengehöriger Objekte können diese innerhalb von 3ds Max in eine Gruppe zusammengefasst werden. Die Gruppe verhält sich wie ein einzelnes Objekt, das nur alleine angefasst und manipuliert werden kann. Um dennoch die inneren Einzelteile der Gruppe zu ändern, muss die Gruppe vorher über ein Untermenü geöffnet werden. Nativ ist das in 3ds Max nur für je eine Gruppe möglich. Die Kanonen des Schiffes bestehen aus zahlreichen Einzelteilen. Sie wurden zur einfacheren Positionierung je als Gruppe zusammengefasst, mussten später jedoch wieder geöffnet werden, um sie zu animieren. Das Skript ermöglichte das gleichzeitige Öffnen bzw. Schließen aller Gruppen mit je nur einem Tastendruck.



openGroups hilft beim Umgang mit gruppierten Objekten.

Name: pointcachePlaybackType
Funktion: Ändern der Abspielgeschwindigkeit von ausgelagerten Animationsdaten
Erläuterung: Die Animation der Segel und Taue des Schiffes wurde in eine externe Datei gespeichert. Da diese in mehreren Szenen wiederverwendet wurde, u.a. auch in Slow-Motion-Einstellungen, musste die Geschwindigkeit der Animation geändert werden. 3ds Max bietet hierfür eine Möglichkeit innerhalb des Point-Cache-Modifier, jedoch zeitgleich nur für ein Objekt. Um für die sieben Segel und ca. zehn Seile nicht jeweils die Einstellung vornehmen zu müssen, wurde das Skript geschrieben. Es ermöglicht das gleichzeitige Strecken oder Stauchen der Animationsgeschwindigkeit der Besegelung.

Name: ahabClothTools
Funktion: Automatisierung einer Multilayer-Cloth-Simulation
Erläuterung: Die Kleidungsstücke von Ahab (Hemd, Hose, Caban) wurden separat simuliert. Dazu wurde jeweils die unten liegende Kleidungsschicht simuliert, dann in ein Kollisionsobjekt umgewandelt und anschließend mit der Simulation der darüber liegenden Schicht fortgefahren. Um das manuelle Umstellen der Simulation und die damit mehrfach notwendige Anwesenheit und Aufmerksamkeit des Nutzers zu vereinfachen, wurden alle nötigen Arbeitsschritte in ein



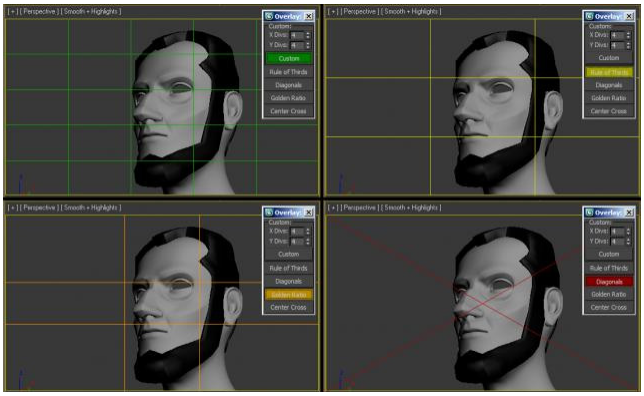
Die ahabClothTools waren ein wichtiger Bestandteil der Stoffsimulation und eine sehr lehrreiche Skriptübung.

Skript zusammengefasst. Zusammen mit Werkzeugen und Shortcuts zur Nachbearbeitung der Simulation und dem geskripteten Interface bilden die ahabCloth-Tools mit über 500 Zeilen Programmcode das umfangreichste Skript des Projekts.

3. Tools

Ein Tool ist ein Hilfsprogramm, das vorhandene Funktionen von 3ds Max zu etwas Neuem verbindet oder selbst neue Funktionen bereitstellt. Sie werden hauptsächlich zur Unterstützung bei speziellen Aufgaben geschrieben, die mit den vorhandenen Mitteln der Software allein nur umständlich oder zeitaufwendig zu lösen sind. Beispiele:

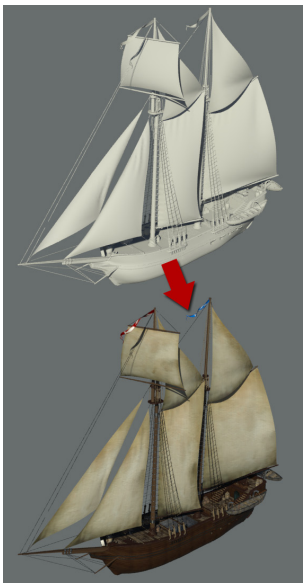
Name: imageCompHelper
Funktion: Visuelle Hilfestellung bei der Bildkomposition
Erläuterung: Während des Blockings war die Positionierung und Ausrichtung der Kamera von großer Bedeutung. Hierfür habe ich teilweise Hilfestellungen wie die „Drittel-Regel“ herangezogen. Die aus der Fotografie stammende Regel teilt das Bild in neun gleich große Abschnitte und empfiehlt ein Ausrichten des Motivs an den entstehenden Trennlinien. Indem somit das Motiv nicht zentriert ist, wird eine allzu statische und langweilige Bildwirkung vermieden. Es gibt weitere Möglichkeiten für Hilfslinien in der Bildkomposition, etwa angelehnt an die Proportionen des goldenen Schnitts. Das Skript generiert diese und weitere Linien als einen temporären Rendereffekt im Ansichtsfenster von 3ds Max. Die visuelle Hilfestellung hat somit keinen Einfluss auf die Szene und erfordert keine zusätzliche Kamera. Durch ständig im Hintergrund stattfindende geometrische Berechnungen passen sich die Proportionen des Rendereffekts automatisch der Skalierung des Ansichtsfensters und dem Kameraausschnitt an.



imageCompHelper zeichnet Hilfslinien zur Unterstützung bei der Bildkomposition in das Ansichtsfenster von 3ds Max.

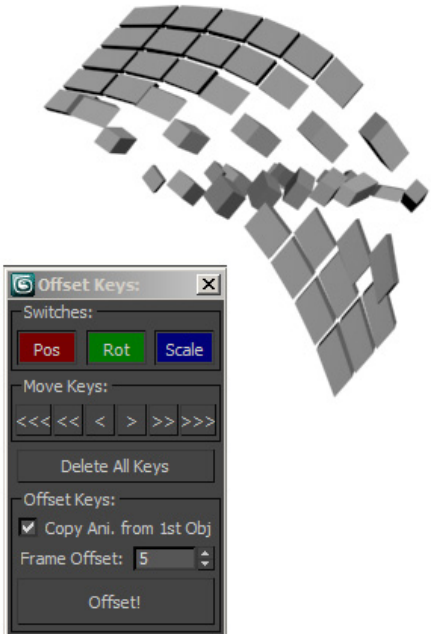
Name: sceneMatsIO
Funktion: Automatischer Import und Export von Materialien
Erläuterung: Die kryptische Abkürzung steht für „Scene Materials In Out“, also das Speichern und Laden von Materialien aus externen Dateien. Das Skript wurde speziell für ein großes Problem während des Projekts geschrieben: Die Kameraeinstellungen des Trailers wurden für die Animation jeweils in eine separate Datei gespeichert. Dadurch entstanden rund zwanzig Dateien mit Kopien des Schiffs, Ahabs und Moby Dicks, die zu diesem Zeitpunkt zwar schon ein Unwrapping, aber noch keine Texturen oder Shader besaßen. Nachdem die Shader und Texturen in einer anderen Datei finalisiert wurden, mussten diese auf die Szenen verteilt werden. Auf gewöhnlichem Weg hätte dies bedeutet, die Materialien händisch auf rund 350 Objekte pro Szene zu verteilen. Das ergibt 350 x 20 = 7000 Arbeitsschritte. Zwar beinhalten die Szenen jeweils nicht alle Modelle, aber selbst bei halbem Ar-

beitsaufwand wäre dieser manuell kaum zu bewältigen gewesen und sehr anfällig für Fehler. Das Skript löst dieses Problem in zwei Schritten: In einer Masterdatei werden alle Modelle mitsamt Materialien gespeichert. Das Skript erkennt im ersten Schritt alle Materialien in der Szene (ca. 60 Stück), und speichert diese in eine externe Materialbibliothek im Projektordner. Die Materialien in dieser Bibliothek können über einen Index identifiziert werden. An den gleichen Ort wird eine Liste aller Szenenobjekte als Textdatei gespeichert, in folgender Form: Objektname, Materialindex. Der zweite Schritt erfolgt jeweils in den noch untexturierten Szenen der Kameraeinstellungen. Das Skript lädt die externe Objektliste und die Materialbibliothek in einen Zwischenspeicher, und vergleicht die Objektnamen in der Szene mit den Namen in der Liste. Wird eine Übereinstimmung gefunden, wird dem Szenenobjekt das durch den zugehörigen Index ermittelte Material zugewiesen. Der ganze Prozess erfolgt in wenigen Sekunden. Die Problematik der Materialverteilung konnte somit fast vollständig automatisiert und erheblich beschleunigt werden. Die Entwicklungszeit wird nicht nur durch die Zeitersparnis im Projekt, sondern zusätzlich durch die Tatsache aufgewogen, dass das Tool ab nun in zukünftigen Projekten zur Verfügung steht. Einzige Voraussetzung ist eine eindeutige Namensgebung der Objekte in Ausgangs- und Zieldatei.



sceneMatsIO lädt Texturen und Shader aus einer externen Datei und verteilt diese intelligent and die Objekte in der Szene. Das Übertragen von Shadern von einer Szene auf die andere ist damit vollständig automatisiert.

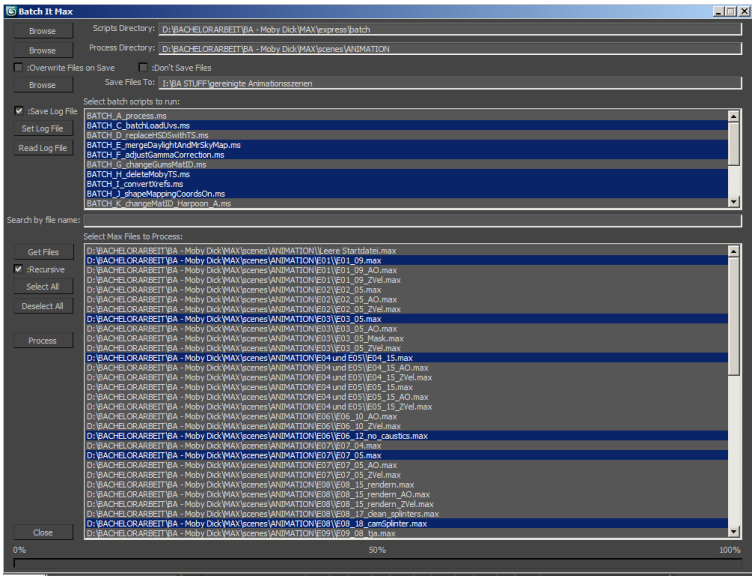
Name: offsetKeys
Funktion: Kopieren und sukzessives Verschieben von Objektanimation
Erläuterung: Dieses Tool unterstützt die Animation der aufgehenden Bordklappen und feuernden Kanonen im Trailer. Hierbei entsteht jeweils eine schnelle Aneinanderreihung von gleichen Bewegungen, die zeitlich leicht versetzt ist: Beispielsweise feuert zuerst Kanone eins, vier Frames später feuert Kanone zwei, vier Frames später feuert Kanone drei usw.
Auf diese Art muss jede Kanone einzeln animiert werden – falls das Timing nicht passt, muss jede Animation angepasst oder neu erstellt werden. Um Timing und Geschwindigkeit der Animation zu testen, wurde das Skript genutzt, das bereits lange vor dem Projekt zur Übung geschrieben wurde. Nach der Animation eines einzelnen Objekts, in diesem Fall der ersten Kanone, kann deren Animation auf alle anderen selektierten Objekte übertragen werden. Hierbei wird jede Animation um einen angegebene Anzahl Frames versetzt, abhängig von der Selektionsreihenfolge. Da Selektionen gespeichert werden können, waren das Abändern der Animation und die Übertragung auf die anderen Kanonen sehr schnell möglich. Zusätzliche Funktionen zum Verschieben von Keyframes halfen bei der Variation der Animation.



Mit offsetKeys können visuelle interessante Animationen erstellt werden.

Batching

Da die Kameraeinstellungen des Trailers über ca. 20 Dateien verteilt waren, mussten Änderungen z.B. am Modell von Ahab oder dem Schiff auf die restlichen Dateien übertragen werden. Zu Ende des Projekts galt es eine Reihe von trivialen Änderungen zu verteilen. Dazu zählen beispielsweise das Ändern von Texturkoordinaten, Materialkanälen und das zeitaufwendige Verteilen von Materialien auf die Objekte in der Szene (s. „sceneMatsIO“) . Derartige Arbeitsschritte lassen sich einfach über ein Skript erledigen. Um diese automatisch in allen Dateien auszuführen, habe ich ein Skript von Paul Neale namens „Batch It Max“ benutzt. Dieses erlaubt eine Stapelverarbeitung von Skriptdateien über eine Sammlung von Dateien. Das Skript öffnet eine Datei nach der anderen, führt alle angegebenen Skripte aus und speichert die Datei wieder. Hiermit konnte ich finale Änderungen kurz vor Beginn des Renderings in einer einzigen, großen Stapelverarbeitung umsetzen, und mir mehrere Stunden Arbeit sparen.



Die Benutzeroberfläche von “Batch It Max” von Paul Neale.

5.6.6 Fazit zu MAXScript

Zusätzlich zu den vorgestellten Beispielen entstanden während des Projekts noch ca. 40 weitere Skripte mit variierender Komplexität und Nützlichkeit. In einigen Fällen hat deren Entwicklung mehr Zeit gekostet, als ihre Funktion eingespart hat. Im Hinblick auf die Verbesserung meiner MAXScript-Kenntnisse waren sie dennoch von großem Nutzen. Ich habe mich zudem bemüht, die Programmierung allgemeingültig zu halten, sodass sie in zukünftigen Projekten Verwendung finden kann. Meiner persönlichen Erfahrung nach hilft mir MAXScript nicht nur dabei, die internen Vorgänge von 3ds besser zu verstehen, sondern bietet vor allem ein mächtiges Werkzeug zur Problemlösung und Zeiteinsparung. Eines meiner Ziele während der Bachelorarbeit war das Verbessern meiner MAXScript-Kenntnisse, auch als zusätzliche Qualifikation für die Arbeitssuche nach dem Studium. Da man am besten Skripten lernt, indem man Skripte schreibt, habe ich während meines Projekts jede Gelegenheit hierfür genutzt. Hierdurch konnte ich die Stärken und Schwächen von MAXScript besser kennenlernen:

Stärken:

- Einfache Syntax und Bedienung, auch für Nichtprogrammierer
- Niedrige Lernkurve durch gute Hilfedatei und Websupport (www.scriptspot.com)
- Zeitersparnis durch Automatisierungen und Optimierung von Arbeitsabläufen
- Besserer Verständnis der internen Programmvorgänge und technischen Limitationen
- Kurze Entwicklungszeit durch integrierte Entwicklungsumgebung

Schwächen:

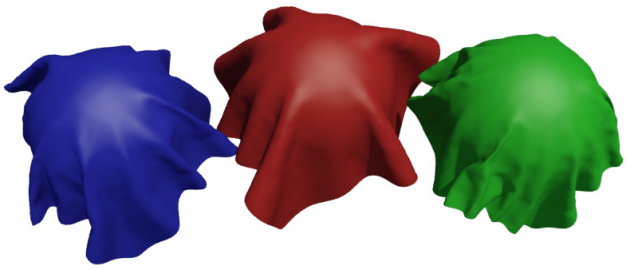
- Nur rudimentäre Kommunikation mit Drittprogrammen, beschränkt auf 3ds Max, bisher keine verlässliche Pipeline-Konstruktion z.B. über Python möglich
- Stellenweise langsame Geschwindigkeit bei der Ausführung
- Falsche Programmierung kann das Programm zum Absturz bringen und im schlimmsten Fall die Datei beschädigen

Man sollte es vermeiden, die Skriptsprache als Komplettlösung für eine Aufgabe zu sehen. Den größten Nutzen hat MAXScript, indem man es unterstützend zur manuellen Arbeit einsetzt. Wenn repetitive, nervenzehrende Prozesse automatisiert und beschleunigt oder tägliche Arbeitsabläufe durch spezielle Werkzeuge und angepasste Benutzeroberflächen optimiert werden können, bleibt mehr Zeit für kreative und gestalterische Arbeit. Auch lohnen sich komplexe Skripte am ehesten bei täglicher mehrfacher Benutzung, idealerweise durch mehrere Mitarbeiter einer Firma. Für eine Einzelperson ist der Entwicklungsaufwand trotz allem meist zu hoch, um mehr als kleine Programme schreiben zu können. Als unterstützende Fähigkeit zur 3D-Arbeit ist die Kenntnis von MAXScript wertvoll und hilfreich, auch da sich das Gelernte einfach auf andere Programme mit ihren Skriptsprachen übertragen lässt (z.B. Flash mit ActionScript, Dreamweaver mit JavaScript, Director mit Lingo).

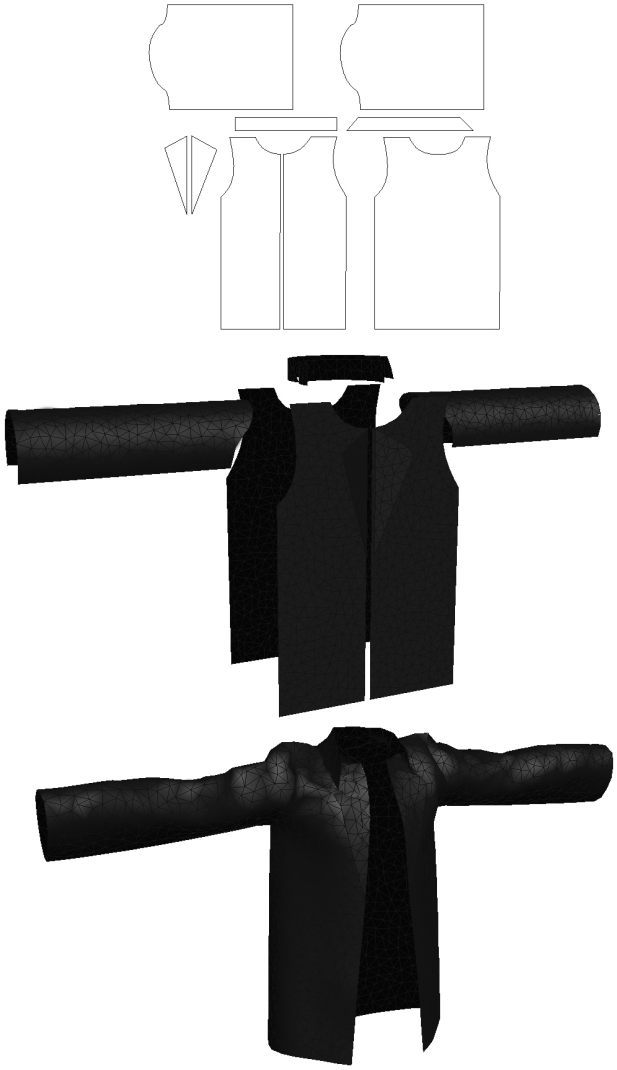
5.7 Stoffsimulation

Ich habe mich bereits zu Beginn des Projekts entschlossen, die Kleidung für die Figur des Ahab nicht über das Skinning am Charakter zu befestigen, sondern die Bewegung des Stoffes zu simulieren. Gerade lange oder weite Kleidung sieht sehr unnatürlich aus, wenn sie starr dem Skinning des Charakters folgt. Simulierte Kleidung hingegen schafft während der Animation eine detaillierte Sekundärbewegung, die die Bewegungen der Figur insgesamt glaubhafter und interessanter macht. 3ds Max bietet zu diesem Zweck ein System namens „Cloth“, das früher als externes Plugin erhältlich war und seit einigen Jahren in das Programm integriert ist. Cloth ist dabei nicht ausschließlich zur Simulation von Stoff geeignet. Jede Soft-Body-Simulation lässt sich mit dem System durchführen. Soft-Body steht hierbei für die Fähigkeit von Objekten, sich selbst und andere Objekte zu verformen. Sich verbiegendes Metall, elastische Gummibälle, im Wind wehende Haare oder organische Substanzen können genauso simuliert werden wie Stoff, da die Berechnung der Verformung auf den gleichen Algorithmen beruht. Kollision mit anderen Objekten und der Einfluss physikalischer Kräfte wie Gravitation und Wind können ebenfalls simuliert werden.
Ahabs Kleidung besteht aus drei Kleidungsstücken: Dem Hemd, der Hose und der Seemannsjacke (Caban). Zusätzlich gehören der Gürtel, die Gürtelschnalle und die Jackenknöpfe dazu, die jedoch nicht selbst Teil der Stoffsimulation waren. Diese wurden über einen Modifier mit den Bewegungen der Stoffobjekte verbunden, sodass sie diesen folgen. Die Steuerung der Stoffeigenschaften wird geregelt über eine Vielzahl an Parametern, die z.B. Dehnungsvermögen, Masse, Faltenwurf, Luftwiderstand, Reibung oder Kollisionsverhalten beeinflussen. Hierdurch lassen sich unterschiedliche Stoffe simulieren, wie Leinen, Wolle, Seide, Polyester oder eben auch härteres Material wie Metall.

Die Erstellung von Kleidung in 3ds Max ist auf zwei Arten möglich: Grundsätzlich kann durch gewöhnliche polygonale Modellierung jedes Modell als Kleidungsstück simuliert werden, jedoch ist diese zeitaufwendig und führt selten zu einem realistischen Flow der Kleidung. Über den sog. „Garment Maker“, also Gewandmacher, kann die Konstruktion der Kleidung ähnlich erfolgen wie in der Kleidungsindustrie oder Schneiderei. Es werden zuerst die Umrisse eines Schnittmusters aufgezeichnet, das danach in ein flaches, zweidimensionales Stoffmodell umgewandelt wird. Diese Stücke werden grob an der Figur oder einem Platzhalterkörper positioniert. Der Garment Maker bietet nun die Möglichkeit, Nahtkanten am 3D-Modell miteinander zu verknüpfen. Über sukzessives Simulieren und Anpassen der Stoffstücke entsteht in Echtzeit



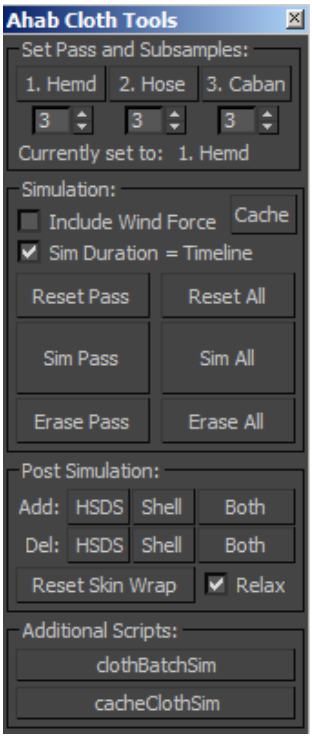
Faltenwurf und Stoffverhalten variieren je nach simuliertem Material. Von links nach rechts: Baumwolle, Seide, Leder.



Anhand eines zweidimensionalen Schnittmusters werden Kleidungsteile wie Ärmel, Kragen und Torso der Jacke einzeln arrangiert, un dann in der Simulation an den Körper der Figur passgenau angelegt.

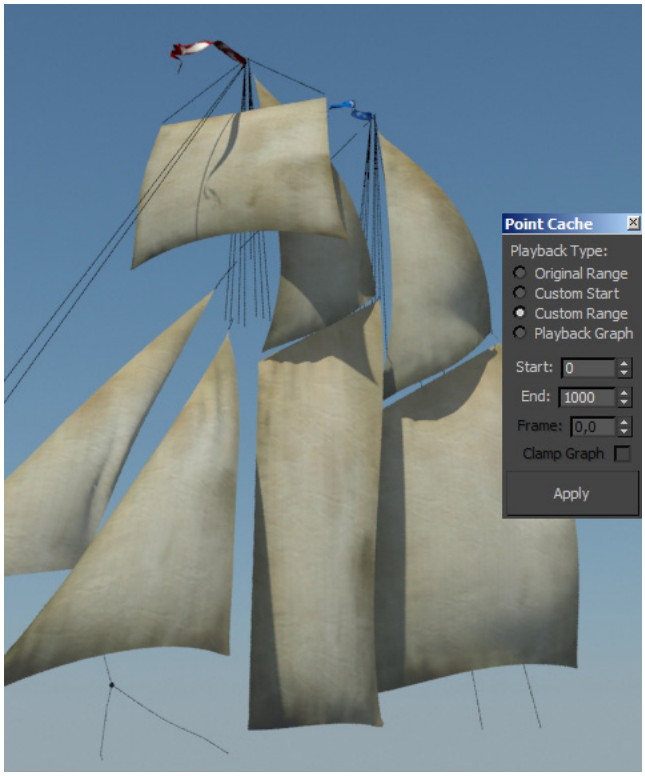
das Kleidungsstück, das zudem direkt an den Charakter maßgeschneidert wird. Die Basis für Ahabs Kleidung wurde auf diese Art konstruiert, was vor allem bei der Seemannsjacke mit ihrem Kragen, den umgekrempelten Ärmeln und Frontklappen effektiv war.

Die überlappenden Kleidungsschichten von Ahab waren aufgrund der gegenseitigen Interaktion der Kleidungsstücke in der Simulation ein Problem. Durch ausführliche Tests ergab sich eine Fehlerhäufung und erhöhte Simulationszeit bei der zeitgleichen Simulation aller Kleidungsstücke. Eine Erklärung hierfür ist die ständige Kollision des einen Stoffobjekts mit dem anderen Stoffobjekt, die sich hochschaukeln und zu ruckartigen Bewegungen oder Durchdringungen des Stoffes führen kann. Das Problem lässt sich umgehen, indem eine Stoffschicht nach der anderen separat simuliert wird. Da in diesem Fall nur das Stoffobjekt auf die Kollisionsobjekte reagieren muss, ist die Fehlerhäufigkeit und der Bedarf an Nachbearbeitung deutlich geringer. Ein weiterer Bonus dieses Verfahrens ist die kürzere Simulationszeit. Das Einrichten dieser Methode und die Nachbearbeitung der Stoffsimulation erwiesen sich dennoch als zeitaufwendig und anfällig für Flüchtigkeitsfehler. Der Ablauf war wie folgt:
Einrichten der Simulation, Simulation des Hemds (ca. 10 min Wartezeit), Änderung der Simulationseinstellungen, Simulation der Hose (ca. 15 min Wartezeit), Änderung der Simulationseinstellungen, Simulation des Cabans (ca. 30 min Wartezeit), Nachbearbeitung der Simulationen. Diesen Ablauf für alle Szenen im Trailer zu wiederholen, schien wenig verlockend, daher wurde ein eigenes Interface für die Simulation programmiert.



Ziel war es, den Ablauf der Stoffsimulation für jede Szene auf einen einzigen Knopfdruck zu reduzieren. Sobald die Animation einer Szene final war, konnte die Stoffsimulation nach Angabe weniger Parameter automatisiert werden. Das manuelle Umstellen der Simulation für die einzelnen Kleidungsstücke entfällt hierdurch, alle Phasen der Simulation werden nacheinander abgearbeitet. Die Zwischenzeit konnte so effektiver für andere Aufgaben genutzt werden, da sie unterbrechungsfrei und auch insgesamt kürzer war. Die Qualität der Simulation (Anzahl der Samples pro Frame), und damit die benötigte Berechnungszeit lassen sich separat pro Kleidungsstück einstellen. In einer Szene, in der z.B. die Hose von Ahab kaum zu sehen ist, kann somit unnötige Simulationszeit gespart werden. Das Interface wurde so gestaltet, dass es von oben nach unten Hilfestellungen zur Simulation und Nachbearbeitung der Kleidung bietet.

Alle Segel und beweglichen Taue des Schiffs wurden ebenfalls mit Cloth simuliert. Die beweglichen Teile mussten dazu untereinander und mit dem Schiff verbunden werden, um sich gegenseitig zu beeinflussen. Ihre Bewegung im Wind sollten dem Schiff Leben einhauchen. Zu diesem Zweck wurden sog. „Spacewarps“, also Hilfsobjekte zur Simulation von Wind eingesetzt und der Stoffsimulation der Segel hinzugefügt. Da ein Windobjekt allein die Bewegung der Segel sehr gleichförmig aussehen ließ, wurden verschiedene Variationen erstellt. Diese sind teils statisch, teils dynamisch in ihrer Position, Rotation, Windstärke und Turbulenz. Die Dynamik wurde durch das Generieren von Zufallswerten mittels Noise-Controller erzeugt. Als die Bewegung der Seile und Segel realistisch wirkte, wurde ein Zeitraum von 500 Frames simuliert und die Bewegungsdaten als Point-Cache in eine externe Datei gespeichert. Diese Datei wurde in den meisten Szenen wiederverwendet, wobei Start- und Endzeitpunkt sowie die Geschwindigkeit der Bewegung variiert wurden, um eine Wiederholung zu vermeiden.



Segel und Seile wurden über externe Simulationsdaten animiert. Dle Geschwindigkeit wurde mit einem Skript global verändert .

6 Postproduktion

6.1 Rendering

Für die Simulation von globaler Beleuchtung in mental ray wurde eine Kombination aus zwei verschiedenen Methoden benutzt: Photon Tracing und Final Gathering. Photon Tracing basiert, wie bereits im vorigen Kapitel erläutert, auf der Kollision von Photonen, die von Lichtquellen ausgehend in die Szene fliegen. Der Zeitaufwand der Methode ist unabhängig von der Kameraposition, aber abhängig von der Anzahl der Lichter und der insgesamt benutzten Photonen. Final Gathering funktioniert auf ähnliche, aber umgekehrte Weise. In diesem Verfahren werden ausgehend von der Kamera Strahlen in die Szene geschickt, die mit Objekten kollidieren und mehrfach abprallen können. Auch hier werden die Kollisionen registriert und auf einer Map gespeichert. Beide Methoden sind für sich gesehen ausreichend um indirektes Licht und globale Beleuchtung zu simulieren, und können unabhängig voneinander dafür genutzt werden. Mental ray bietet aber die Möglichkeit, beide miteinander zu kombinieren, wodurch Qualitäts- und Geschwindigkeitsvorteile erzielt werden: Photon Tracing kann fleckige Artefakte im Bild entstehen lassen, die durch Final Gathering geglättet und beseitigt werden. Die Photonenlösung stellt also eine schnelle Grundlage für die globale Illumination dar, die durch das Final Gathering verfeinert wird – in der Kombination ist das Verfahren spürbar schneller als reines Final Gathering.
Beide Methoden sind adaptiv, d.h. die Anzahl der verwendeten Photonen / Strahlen und die Genauigkeit der Berechnung variiert je nach Bildinhalt. Beim Rendern einer Animation mit bewegter Kamera und bewegten Objekte entstehen somit leichte Unterschiede in der Helligkeit der Lichtlösung. Diese können sich als Flackern bemerkbar machen, wenn der Film in Echtzeit abgespielt wird. Mental ray bietet die Option, die Final Gather Strahlen für jeden Frame von mehreren Positionen entlang des Pfades der animierten Kamera aus abzufeuern. Hierdurch werden zusätzliche Daten, sog. „Samples“ gesammelt, die dann interpoliert werden und eine gemittelte Helligkeit ergeben. Dieses Vorgehen verringert das Auftreten von Flackern während des Trailers spürbar.
Während der Produktion eines 3D-animierten Films ist lediglich eine Vorschau der 3D-Daten möglich, da die Berechnung der finalen Bilder aufgrund des hohen Rechenaufwands nicht in Echtzeit erfolgen kann. Im Rendering werden aus den Rohdaten konkrete zweidimensionale Bilder erzeugt, die alle Oberflächeneigenschaften sowie Licht- und Schattenwurf enthalten. Für die Berechnung dieser Bilder wird vorrangig Prozessorleistung benötigt, sodass mehr Rechenkerne eine deutliche Verkürzung der Renderzeit bedeuten können. Obwohl der Rechenaufwand je nach Komplexität und Beschaffenheit einer Szenerie stark schwanken kann, hat sich für meinen Trailer eine durchschnittliche Berechnungszeit von ca. 15 min pro Frame ergeben. Für meinen Trailer konnte ich die Renderfarm des Fachbereichs Medienproduktion nutzen, und statt nur meinem eigenen Computer mit sechs Maschinen rendern. Der nötige Zeitaufwand wurde dadurch von mehreren Wochen auf wenige Tage reduziert:

3D-Anteil des Trailers: ca. 55 s

Renderzeit =	15 min pro Bild *
	25 Frames pro Sekunde *
	55 Sekunden Film
=	20.625 min
=	343,75 h
=	14,32 d

Statt 14,32 Tagen an einem Rechner, sollte der Film theoretisch also an sechs Rechnern vergleichbarer Geschwindigkeit nur 2,39 Tage rendern. Da die Arbeit beim Rendern aber nicht nur in der reinen Renderzeit zu messen ist, sondern auch in Vorbereitung, Kontrolle und Fehlerbereinigung, belief sich die Gesamtrenderzeit auf ca. sieben Tage.
Die Koordination der einzelnen Rechner geschah mittels der Software „Backburner“, die zusammen mit 3ds Max ausgeliefert wird. Backburner stellt ein mächtiges Hilfsmittel zum Rendern vieler Dateien dar: Alle Einstellungen werden zentral an einem einzigen Rechner getätigt, der die Administration der übrigen Rendermaschinen übernimmt. Renderaufträge können in eine Warteschlange eingereiht und nacheinander abgearbeitet werden, das Programm verteilt dabei dynamisch verbleibende Frames gleichzeitig an alle Rechner. Jeder Renderauftrag liefert umfangreiche Statistiken zur benötigten Zeit und aufgetretenen Fehlern, welche die Fehlersuche und das Einschätzen des restlichen Zeitaufwandes erleichtern. Manuelles Starten und Beenden von Renderjobs entfällt durch Backburner, die Renderfarm wird zu jeder Zeit voll ausgelastet.
Zum Rendering gehört eine abschließende Überprüfung jeder Szene.

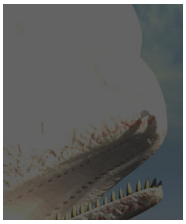
E08_15_rendern_ZVel2	12	50	Complete	(100%) 0126/0126	christoph
E08_17 fr98-126 no splinters	13	50	Complete	(100%) 0028/0028	christoph
E08_17 68-97 hoffentlich heller	14	50	Complete	(100%) 0030/0030	christoph
E08_18_camSplinter fr98-125	15	50	Complete	(100%) 0028/0028	christoph
E08_16_reduced fg fr18-67 neu	16	55	Complete	(100%) 0049/0049	christoph

Backburner reiht Renderjobs in eine Warteschlange ein und arbeitet sie sukzessiv ab, wodurch Leerlauf entfällt.

6.2 Compositing

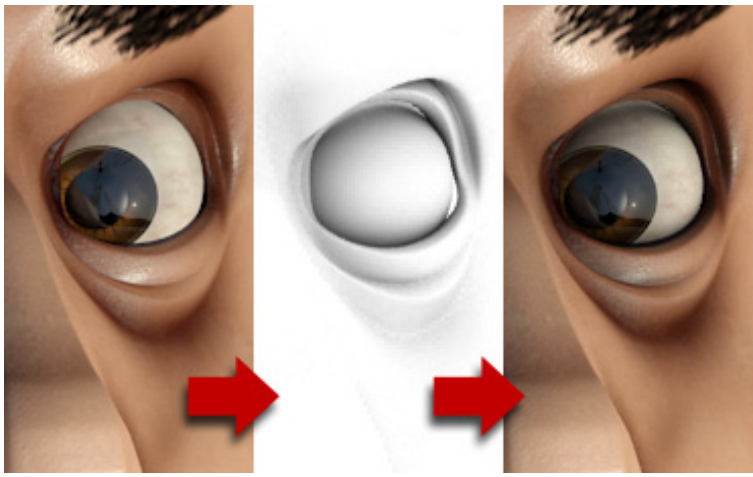
Während des Renderings muss nicht zwangsweise nur ein Bild pro Frame entstehen. Jedes Bild lässt sich zahlreiche Unterbilder, sog. „Passes“ aufteilen und einzeln ausgeben. Beispielsweise lassen sich alle Schatten im Bild als separates Bild rendern, ebenso alle Lichter oder Reflektionen. Die Passes lassen sich in einer entsprechenden Software, in meinem Fall After Effects, wieder miteinander zu einem finalen Bild kombinieren. Hierbei müssen Nutzen und Mehraufwand abgewogen werden: Wenn alle Bestandteile des Bildes getrennt voneinander als Bild vorhanden sind, lassen sie sich auch einzeln korrigieren und verändern. Daraus resultiert eine große Kontrolle und Flexibilität für das Endprodukt, da sich z.B. Schattenintensität, Lichtfarbe von Sonnen- und Himmelslicht oder die Stärke der Reflektionen im Bild im Nachhinein stufenlos verändern lassen, ohne das für diese Änderungen das Bild neu gerendert werden muss. Andererseits erfordert die Kombination der Passes für jeden Shot einen nicht unerheblichen Arbeitsaufwand. Ich habe mich daher für einen Kompromiss entschieden, und das Renderbild mit Schatten, Reflektionen und Lichtern in ein einziges Bild gerendert. Hierbei wurde das OpenEXR-Format benutzt, das 32 Bit RGB Farbtiefe besitzt. Zwar wird am Ende des Projekts ein Trailer mit normaler 8 Bit Farbtiefe entstehen – dennoch bietet die höhere Farbtiefe während des Compositings Vorteile. Die Farbe und Helligkeit eines Pixels wird im OpenEXR-Format als Gleitkommazahl gespeichert und besitzt so eine deutlich höhere Auflösung als beispielsweise das JPEG-Format. Nachträgliche Helligkeitskorrekturen von über- oder unterbelichteten Renderbildern sind daher deutlich verlustfreier, da augenscheinlich weiße oder schwarze Bildbereiche immer noch genügend Bildinformationen enthalten.

Das linke Bild wurde beim Rendern überbelichtet ausgegeben, jeweils in 8 und 32 bit.



Versucht man in dem 8 bit Bild, durch eine Veränderung der Bildhelligkeit oder des Kontrastes die hellen, ausgebrannten Bildbereiche zu korrigieren, ist in diesen keinerlei Zeichnung mehr zu erkennen. Eine Bildkorrektur an dem 32 bit Bild enthüllt hingegen verlorene Details des Wals.

Zusätzlich zum normalen Bild wurden noch drei weitere Passes gerendert, die zusätzliche Informationen beinhalten. Der sog. „Ambient Occlusion Pass“ simuliert Umgebungsabdunklung und resultiert in einem Graustufenbild, das naheliegende Objekte und Strukturen dunkler zeichnet. Durch eine Multiplikation mit den Tonwerten des normalen Bildes werden Kontaktschatten betont und abgedunkelt: Dies führt zu einer erhöhten Tiefenwirkung im Bild.



Ein Beispiel anhand der Augenregion von Ahabs Gesicht. Das Renderbild wird mit dem Ambient Occlusion Pass kombiniert und wirkt so plastischer.

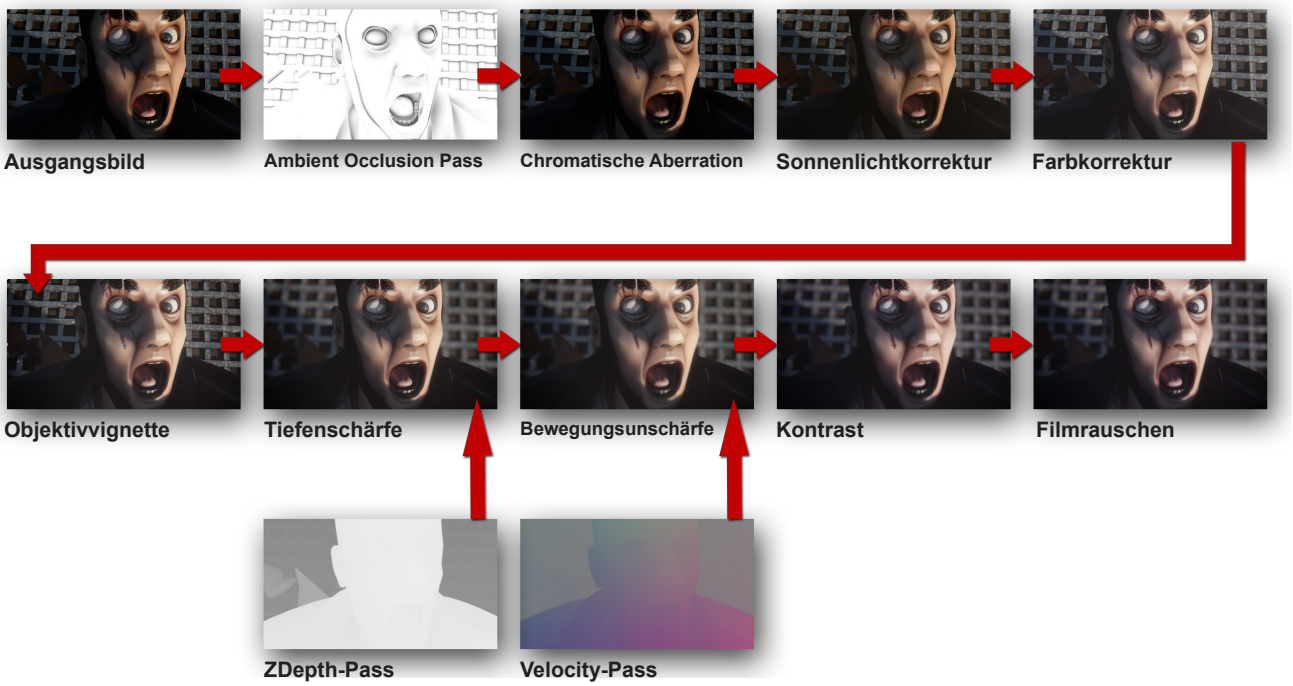
Mental ray ist in der Lage, Tiefenschärfe und Bewegungsunschärfe einer Kamera während des Renderns zu berechnen und in das Endbild einzufügen. Die Qualität dieser Unschärfe ist gut, jedoch steigt die Renderzeit sehr stark an, bei mittlerer Qualität ca. um den Faktor vier. Für eine Animation ist die vierfache Renderzeit nicht akzeptabel, doch glücklicherweise lassen sich mittels „Z-Depth“- und „Velocity“-Pass beide Unschärfen einzeln ausgeben. Der vergleichsweise geringe Einrichtungsaufwand der Szene wird durch die schnelle Renderzeit mehr als aufgewogen. In meinem Projekt habe ich die Einstellung dieser Passes mit MAXScript unterstützt, um den Prozess zu beschleunigen und Flüchtigkeitsfehler zu vermeiden. Das Ergebnis des Z-Depth-Pass ist ein Graustufenbild, in dem nahe Objekte hell und entfernte Objekte dunkel gezeichnet werden. Eine Compositing-Software kann diese Information nutzen, um Tiefenunschärfe über das Bild zu legen. Die Qualität der Unschärfe ist weniger gut als bei der direkten Berechnung im Rendering, bietet aber den Vorteil, dass sie im Compositing stufenlos verschoben und verändert werden kann.

Verschiebung der Fokusebene mithilfe des ZDepth-Passes. Links ist Ahab im Fokus, rechts ist es die Takelage hinter ihm.

Der Velocity-Pass generiert ein Bild, in dem die Farbigkeit von Objekten und ihre Sättigung die Richtung und Geschwindigkeit ihrer Bewegung kodiert. Im Compositing wird diese Information interpretiert um Bewegungsunschärfe zu erzeugen.

Wird der Velocity Pass in einer Compositing-Software mit dem Renderbild kombiniert, entsteht justierbare Bewegungsunschärfe.

Die folgende Abbildung zeigt den Ablauf der Kombination aller Passes zum finalen Bild, wie er für die meisten Kameraeinstellungen genutzt wurde.



Vor dem Compositing:

Nach dem Compositing:

6.3 Musik und Ton

Der Trailer sollte in Hinsicht auf Ton und Musik hauptsächlich von der Melodie und einer Sprecherstimme getragen werden. Die Nachvertonung mit Geräuschen sollte hierbei unterstützend wirken. Für ein kleines Budget konnte Andrew Montesi für die Sprachaufnahmen gewonnen werden, der seit 16 Jahren als professioneller Sprecher arbeitet und sich u.a. auf Filmtrailer spezialisiert hat (Chronicles of Riddick, Star Wars Episode 3, Aeon Flux). Der Text der Sprachaufnahmen umschreibt in kurzen, prägnanten Sätzen die Geschichte des Films:

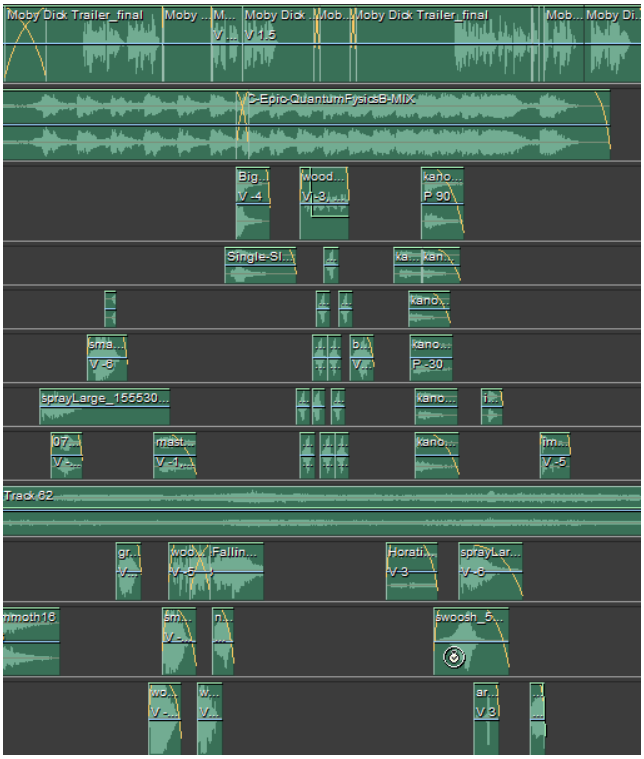
“On the calm, endless sea... one man is seeking revenge. His name is Ahab. Crippled by his nemesis... propelled by infinite hate and lust for vengeance. This is the story of his legendary hunt for the ferocious white whale: Moby Dick. When the ocean is the battlefield... there is no place to run. In this epic duel, only one can prevail. The everlasting struggle of man and beast concludes... in: ‘Death to Moby Dick’ Coming soon to a theater near you.”

Die Kernaussagen wurden sprecherisch zusätzlich betont und mit Schriftanimation unterlegt, um in wenigen Worten den Film zu erläutern:

AHAB
VENGEANCE
MOBY DICK
ONLY ONCE CAN PREVAIL

Ahab will Rache an Moby Dick – Nur einer kann überleben.

Für die Musik und Soundeffekte habe ich auf eine käufliche DVD mit Melodien und Tracks speziell für die Produktion von Trailern, Promovideos und Teasern zurückgegriffen. Die Melodie wurde auf den Trailer angepasst, sodass sie durch Steigerung in Geschwindigkeit und Lautstärke die filmische Dynamik bis zum Finale unterstützt.
Die Geräusche zur Nachvertonung entstammen der Soundbibliothek des Fachbereichs Medienproduktion. Sie beinhalten u.a. Meeresrauschen, Wind, splitterndes Holz und Explosionen für das Abfeuern der Kanonen. Das Anlegen der Geräusche an das Trailervideo fand parallel zum Compositing statt, und wurde mehrfach an die geänderte Länge des Videos angepasst. Im nächsten Schritt wurden die Amplituden der Geräusche, Sprecherstimme und Musik aufeinander angepasst. Hierbei bilden die Geräusche eine Hintergrundkulisse. Im sog. „Panning“ wird die grobe Richtung der Geräusche festgelegt. Ein Geräusch das durch ein Objekt am rechten Rand des Videos erzeugt wird, ist somit bei Stereowiedergabe auch hauptsächlich durch den rechten Lautsprecher zu hören, um eine grobe Ortung zu ermöglichen. Dieser Effekt ist z.B. bei der Kanonensalve zu hören, da die Schüsse im linken Hörbereich beginnen und dann nach rechts wandern.



Die Multitrackansicht in Adobe Audition. Hier wurden Sprachaufnahmen, Musik und Geräusche in Spuren übereinander gelegt und gemischt. Dle finale Tonspur wurde als .wav exportiert und in After Effects dem Video hinzugefügt.

7 Schlusswort und Danksagung

Die Umsetzung meiner Bachelorarbeit war in jeder Hinsicht ein lehrreiches Unterfangen. Ich habe mir Zeit nehmen können, um neue Fähigkeiten zu erlernen und bestehende Kenntnisse zu vertiefen. Eine Filmproduktion in Gänze allein abzuschließen, hat mir einen guten Einblick in die Anforderungen und den nötigen Zeitaufwand unterschiedlicher Arbeitsbereiche gegeben. Dass ein 1,5 minütiges Animationsprojekt mehrere Monate Arbeit mit sich bringen kann, die ich in jeder Hinsicht selber koordinieren und verantworten musste, war für mich eine nachhaltige Erfahrung. In meiner Berufswahl fühle ich mich durch den erfolgreichen Abschluss des Projekts bestärkt und konnte zudem konkrete Erfahrungen in der Projekt- und Zeitplanung und verbesserte Programmierkenntnisse erwerben. Der zeitlichen Beanspruchung durch technische und organisatorische Probleme werde ich in zukünftigen Projekten mehr Aufmerksamkeit widmen. Mit der Qualität meiner Arbeit und ersten Resonanzen bin ich insgesamt zufrieden, obwohl ich nicht alle meine Vorstellungen umsetzen konnte und Raum für Verbesserungen sehe. Zu Ende meines Bachelorstudiums stelle ich erfreut fest, wie sehr sich meine theoretischen und praktischen Fähigkeiten und mein Umgang mit medialen Problemen und Herausforderungen verbessert und gefestigt haben.

Für die Betreuung meiner Bachelorarbeit und zahlreiche Hilfestellungen möchte ich mich sehr herzlich bei Prof. Peter Kaboth bedanken. Mein Dank gilt auch meinem Zweitprüfer Dr. Frank Lechtenberg für seine spontane Zusage. Dankbar bin ich auch allen Helfern und Helferinnen, die mich auf unterschiedlichste Art und Weise unterstützt haben: Nico Bauerschäfer, Bernd Buttala, Christoph Bührig, Nadja Giesbrecht, Andrew Montesi, Fabian Neuhaus, Konstantin Krug, Matthias Lappe, Daniel Ruthmann und Mark Spindler. Abschließend gilt mein Dank meiner Familie und meiner Freundin für die Unterstützung während meines ganzen Studiums.

Quellenverzeichnis

Abbildungen:

Abbildung 1-4 entstammen dem Bilderarchiv von www.coolantarctica.com.

Abbildung 5-7 wurden dem Kinofilm “Moby Dick” von John Huston aus dem Jahr 1952 entnommen.

Abbildung 8-10 wurden dem Fernsehfilm “Moby Dick” von Franc Roddam aus dem Jahr 1998 entnommen.

Abbildung 11 und 12 entstammen dem Bilderarchiv von www.coolantarctica.com.

Abbildung 13 entstammt http://images.allmoviephoto.com/2007_Beowulf/big/2007_beowulf_017.jpg.

Abbildung 14 entstammt http://500.the400club.org/wp-content/uploads/2010/12/tomhankspolarexpress.jpg.

Abbildung 15 entstammt http://www.wired.com/images_blogs/underwire/images/2008/08/14/count_dooku.jpg.

Abbildung 16 enstammt dem Wikipedia Eintrag zum Uncanny Valley.

Abbildung 17 entstammt verschiedenen Künstlergallerien auf www.deviantart.com.

Abbildung 18 wurde von Konstantin Krug für mich gezeichnet.

Abbildung 19 wurde dem Kinofilm “Moby Dick” von John Huston aus dem Jahr 1952 entnommen.

Abbildung 20 entstammt dem Marvel Comicbuch “Moby Dick” aus dem Jahr 2007.

Abbildung 21 und 22 entstammen dem Kinofilm “Master and Commander” von Peter Weir aus dem Jahr 2003.

Abbildung 24 entstammt http://lab.visual-logic.com/wp-content/uploads/2010/02/nurbs_surface_0002_Background-copy.jpg.

Abbildung 25 entstammt http://folk.ntnu.no/havardsc/site/wordpress/wp-content/uploads/2010/02/mine2.png.

Abbildung 26 entstammt http:\\www.jeff-hanna.com.

Sofern nicht hier angegeben, sind alle übrigen Abbildungen von mir selbst erstellt worden.

Weitere Quellen:

Die Sprachaufnahmen wurden von Andrew Montesi gemacht: www.andrewvoice.com

PEN Attribute Holder 2 und Batch It Max stammen von Paul Neale: http://www.paulneale.com

Cache Bar stammt von Joe Gunn: http://www.joegunn3.com

Realtime Spring stammt von Harrison Yu: http://www.seageo.com/Harrison.html

Die Erläuterungen über MAXScript resultieren aus meinen persönlichen Skripterfahrungen und dem Studium der MAXScript Hilfedatei (geschrieben von John Wainwright, Boris Petrov, Larry Minton u.v.a.), sowie Hilfestellungen des Onlineforums von http:\\www.scriptspot.com. Hierbei sei dankend Panavgat “Anubis” Karabakalov erwähnt, der auf meine Fragen stets eine Antwort wusste.

Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Dokumentation zu meiner Bachelorarbeit selbstständig und ohne unzulässige Hilfe Dritter angefertigt habe. Desweiteren versichere ich, dass jede Stelle, die aus anderen Quellen übernommen wurde, als solche kenntlich gemacht ist und ihre Herkunft im Quellennachweis dargelegt wird.

Gütersloh, den 16. Februar 2011