UNIVERSITÄT
KOBLENZ · LANDAU
Fachbereich 4: Informatik

Realistische Darstellung unterschiedlicher Farbschattierungen von Haaren

Studienarbeit

im Studiengang Computervisualistik

vorgelegt von

Katrin Frank

Betreuer: Dipl.-Inform Matthias Biedermann
(Institut für Computervisualistik, AG Computergraphik)

Koblenz, 15. Mai 2006

Erklärung

	Ja	Nein
Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden.	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☐	☐

Ort, Datum Unterschrift

Inhaltsverzeichnis

Motivation

Faszinierendes und gleichzeitig herausforderndes Computergrafik-Thema, ist die realistische Darstellung von Haaren. Dies ist nicht alleine faszinierend für Fachleute, sondern auch Laien verfolgen die Entwicklung der letzten Jahre gespannt. So ist allerseits ersichtlich, eine naturgetreue Nachbildung von Menschen und Tieren im Rechner ohne Haare ist unabdingbar, und je weiter diese Technologie fortschreitet, desto ähnlicher wird sie unserer realen Welt. Aus diesem Grund geriet in den letzten Jahren das Thema der Haarsimulation immer entscheidender in den Blickpunkt der Computergraphik. Nicht zuletzt durch die erhöhte Leistungsfähigkeit von Grafik-Hardware, sind hier große Forschritte zu verzeichnen.

Die Abbildung von Haar in Naturhaarfarben, das durch Wechselwirkung mit z.B. der Sonne im Laufe der Zeit signifikante Farbschattierungen erhält, wird bisher nur dürftig berücksichtigt. Oft handelt es sich um die Darstellung einheitlicher Haarfarben, allenfalls mit einigen, farblich abgesetzten Strähnen. Auf eine realistische Wiedergabe unterschiedlicher Farbschattierungen von Haaren soll in der Studienarbeit näher eingegangen werden.

Problemstellung

Haare stellen eine große Herausforderung an die Computergrafik und insbesondere an die Computerhardware dar. Durch die diffizile und winzige Geometrie eines einzelnen Haares, kommt es bei einer realistischen Computersimulation sehr schnell zu einer großen Anzahl, die bei menschlichen Haaren etwa von rothaarig 90.000 bis zu blond 150.000 variiert und bei Fell sogar noch weitaus größer. Bisher übliche Geometrien in der Computergrafik sind für eine solche feine Darstellung nicht gut geeignet. Oftmals beträgt der Durchmesser eines einzelnen Haares weniger als einen Bildschirmpixel. Erste Lösungsansätze hierzu sind unter anderem die Verwendung von Transparenzen.

Beobachtet man die Haare genauer, fallen weitere Besonderheiten ins Auge. Das Haar für sich ist nicht vollständig opaque, aber auch nicht völlig transparent. Ähnlich wie bei der menschlichen Haut, dringt auch bei Haaren ein Teil des Lichtes in das Haar ein, ein Teil wird gespiegelt und ein weiterer Bereich ist vollkommen deckend.

Auch die Beleuchtung dieser Materie ist keine Kleinigkeit. Abhängig von der Lage und Ausrichtung der Schuppen, findet keine einfache Reflektion des Lichts statt. Es handelt sich bei Haaren um so genanntes anisotropes Material, das heißt, das Licht ist abhängig von der Struktur oder dem Verlauf der Oberfläche.

Darüber hinaus müssen einige weitere Variablen berücksichtigt werden, die großen Einfluss nehmen und zu einer realistischen Darstellung beitragen. Hiermit wird sich das Kapitel „Eigene Beobachtungen“ näher beschäftigen.

Aufgabenstellung

Beginnend mit einer übergreifenden, theoretischen Analyse verschiedener Ansätze und Beobachtungen unterschiedlich wichtiger Eigenschaften des Erscheinungsbildes von Haaren, wird diese Studienarbeit einen elementaren Ansatz näher verfolgen und diesen realisieren. Die vorliegende Arbeit wird diese erarbeiteten Eigenschaften von vorn herein berücksichtigen und geht hauptsächlich auf die Farbgebung von menschlichen Haaren ein. Die dazu gemachten Betrachtungen werden an hand eines statischen Models umgesetzt, das sich an bereits existierenden Ideen orientiert. Auf eine Bewegungssimulation der Haare, auf Bedienelemente einer Oberfläche und auf eine optimale Ausleuchtung wird in dieser Arbeit nicht eingegangen. Die Ausarbeitung beschäftigt sich weiter mit verwendeten Komponenten und Bestandteile der Umsetzung. Zu Ende wird sie weitere offene Arbeit konkretisieren und einen Ausblick in die Zukunft geben.

Analyse existierender Ansätze

Die Literatur und Forschung lieferte in den letzten Jahren zum Thema „Haare“ viele verschiedene Ansätze, Theorien und Umsetzungen. Physikalische Eigenschaften und die Abhängigkeit des Erscheinungsbildes von äußeren Umständen zu berücksichtigen ist keine geringe Herausforderung. Schon der mathematische Ansatz der Implementierung einzelner Haare bietet ein breites Arbeitsfeld, hinzu kommt das der Texturierung, Beleuchtung und korrekten Bewegung. Folgende Forschungsbereiche werden unterschieden:

- Haardynamik: Bewegung von Haar, Kollision mit dem Kopf und miteinander
- Rendering: optisches Erscheinungsbild bestimmt durch Haarfarbe, Schattierung, Reflexion und Transparenz
- Visualisierung einzelner Haarsträhnen und -büschel oder einzelner Haare

Man unterscheidet weiter mehrere mathematische Modelle zur Visualisierung von Haaren:

- Explizite Haarmodelle,
- Haare als fließende Strömung,
- Cluster- und Wispmodelle,
- Volumetrische Texturen,
- Partikelsysteme.

Sie alle sind sehr aufwendig und problematisch in Echtzeit zu realisieren. Wie der Ansatz von T. Scheuermann [12] zeigt, sind auch Varianten denkbar, die wesentlich geringere geometrische Komplexität aufweisen, hierdurch einfacher per Tiefenwert zu sortieren und schon mit Low-end-Hardware zu verwenden sind. Um Alpha-Transparenzen korrekt anzuzeigen, wird bei Scheuermann nicht von hinten nach vorne sortiert, sondern von Kopfmitte nach außen. Der Nachteil ist die durch das Modell festgelegte und nicht variable Frisur.

Merkmale und Erscheinungsbild von Haaren

Kopfhaare eines Menschen lassen sich klassifizieren. Die menschliche Haarfarbe unterteilt man in zwei Pigmenttypen, in das dunkle Eumelanin und das hellere Phäomelanin. Diese sind in jeder Haarfarbe in unterschiedlichen Anteilen vorhanden und erzeugen elf bis zwölf Tontiefen von schwarz bis helllichtblond. Die Anzahl der Haare liegt für rot bei 90.000, für schwarz bei 100.000, für braun bei 110.000 und für blond bei 150.000.

Aber nicht nur unterschiedliche „Typen“ können unterschieden werden, auch die Aufteilung in Regionen an einem einzigen Haarschopf. Schon ein einzelnes Haar kann an verschiedenen Positionen ganz andere Farben aufweisen. Dies hängt besonders mit dem Einfluss äußerer Umstände zusammen, wie etwa der Sonne, der Temperatur oder der Berührung mit Salzwasser, um nur einige Beispiele zu nennen. Nicht alle Haare reagieren gleich. So bleicht braunes und blondes Haar wesentlich schneller in der Sonne aus, als sehr dunkles bis schwarzes Haar.

Hinzu kommt eine unterschiedliche Spiegelung der Haaroberfläche, die wiederum vom Gesundheitszustand abhängen kann. Lange Haare sind Witterungen länger ausgesetzt, so dass hier eine wesentliche Beobachtung zu machen ist. Sie bleichen an den Spitzen durch langes Strapazieren extrem aus.

Abb. 1 u. 2: Haarfarbe auf der Höhe des Hinterkopfes und den helleren Haarspitzen

Ebenfalls unterscheiden muss man in vielen Fällen die Färbung des Unterhaars und Deckhaars. Auch heraushängende Strähnen an den Seiten werden stärker

beansprucht und sind deshalb ebenfalls oft heller in ihrer Färbung. Der Ansatz an der Kopfhaut weist dagegen die dunkelste Farbe auf. Das Haar ist dort gesund und der Umwelt noch nicht lange ausgesetzt.

Eine weitere Besonderheit unserer Haare ist ihr anisotropes Verhalten, was Richtungsabhängigkeit einer Eigenschaft bedeutet, in diesem Fall der uns erscheinenden Farbe. Erzeugt wird diese Eigenheit durch die Struktur des Haares. „Das Haar ist grob in drei Schichten aufgebaut. Die erste Schicht, Schuppenschicht oder Cuticula genannt, besteht aus flachen, übereinander greifenden Zellen, vergleichbar mit einem Tannenzapfen. Sie besteht aus sechs bis zehn solcher Zelllagen. Die Schuppenschicht ist insofern wichtig, als diese den Gesundheitszustand des Haares am offensichtlichsten zeigt. Beim gesunden Haar liegt die Schuppenschicht flach an und ergibt so eine flache Oberfläche. Das Licht wird optimal reflektiert und ergibt so den gesunden Glanz des Haares.“[1] Auf der Struktur dieser oberen Schicht ist die Anisotropie begründet.

Wie bereits im Marschner Modell [12] beobachtet, weisen Haare einen weißen Lichtschein (Highlight) Richtung Haarspitzen und einen benachbarten farbigen Richtung Haarwurzel auf. Begründet ist dies durch die unterschiedlich reflektierten Lichtstrahlen. Das erste, hellere Highlight wird nach Marschner direkt an der Haaroberfläche reflektiert. Das zweite, farbige Highlight ist die Verfolgung des ersten Lichtstrahls bis in die teilweise transparente Struktur des Haares hinein. Im Inneren des Haares wird sie dann ebenfalls zurückgeworfen und an der Oberfläche durch die Haarschuppen noch einmal gebrochen reflektiert. Diese Beobachtungen wollte ich überprüfen und machte im Laufe der Studienarbeit eigene Beschauungen, die im Ergebnis etwas spezieller waren. Als Resultat dieser Beobachtung ist festzuhalten, dass die als Marschner-Modell bekannte Reflektion nicht allgemeingültig ist. Das beschriebene Phänomen tritt insbesondere bei Fotografien mit Blitzlicht auf, das heißt enormer Lichtbestrahlung und damit verbundener Reflektion, bei Fotografien ohne Blitz jedoch kaum bis gar nicht und auch in Natura konnte ich diese Beobachtung nicht teilen.

[1] Wikipedia Stichwort „Haar“ (13.05.2006): http://de.wikipedia.org/wiki/haar.


Abb. 3 u. 4: Foto mit Reflektionsschein, ausgelöst durch Blitz und Foto ohne Blitz.

Anforderungsanalyse

Um eine realitätsnahe Simulation der optischen Eigenschaften von Haaren umzusetzen, sind zunächst die Erfordernisse zusammenzustellen.

Aus den zu Beginn beschriebenen Eigenschaften bezüglich der Haarmenge ergibt sich, dass zunächst eine Entscheidung über die gewählte Datenstruktur getroffen werden muss. Einerseits gibt es die Möglichkeit einer mathematischen Generierung von Haaren oder Haarsträhnen zur Laufzeit, andererseits kann die Haargeometrie auch vorab mittels einer Digital Content Creation Software (DCC) wie bspw. Alias Maya oder 3D Studio Max erzeugt werden.

In dieser Arbeit wurde entschieden, die Geometrie vorab zu erstellen, da dies in der Visualisierung zu erheblich größerer Performanz führt.

Zusätzlich sollte in die Echtzeitfähigkeit durch Verwendung eines Shading-Programmes gesichert werden. Um die Vorzüge der bereits vorhandenen Grafik-

Hardware der Firma Nvidia ausnutzen zu können, wurde hierzu die High-Level Shading-Language Cg (C for Graphics) der allgemeineren OpenGL Shading Language vorgezogen. Das Shading-Programm wird aus einem mit OpenGL realisierten Rahmenprogramm aufgerufen.

Planung der Studienarbeit

Vorab ist es sinnvoll zur genaueren Planung der Implementation der Studienarbeit Gedanken schriftlich festzuhalten. Diese werden nachfolgend wiedergegeben.

Berücksichtigung von Vorraussetzungen

Die erste Frage, die gestellt werden sollte, ist die Frage nach Vorkenntnissen und für welche Dinge eine Einarbeitungszeit berechnet werden muss. Für die Studienarbeit liegen zum Planungszeitpunkt weder Kenntnisse in der 3D-Modellierung (hier Alias Maya) und keine Kenntnisse in Cg (C for Graphics) vor. Aus Hardwaresicht war ein ausreichender Arbeitsspeicher und eine Cg fähige Grafikkarte notwendig. An die gesamte Applikation sind folgende Anforderungen zu stellen:

Geplante Funktionen des zu erstellenden Produkts:

- Laden von Modellen
- Laden von Texturen
- Bewegung des Objekts im OpenGL[2] Fenster
- Funktionsfähiges Kontextmenü
- Ein und Ausschalten der Shader

[2] OpenGL (Open Graphics Library) ist eine Spezifikation für ein plattform- und programmiersprachenunabhängiges API (Application Programming Interface) zur Entwicklung von 3D-Computergrafik.

Musskriterien

- Laden und Anzeigen von Kopfmodell
- Laden und Anzeigen von Haarmodelle
- Zusammensetzen von Modellen
- Laden der Texturen auf Modelle
- Modellunterscheidung, Unterschiedliche Texturierung der Haarmodelle
- Anwendung von Shader auf Haarmodell

Wunschkriterien

- Ein und Ausschalten der Shader
- Selbstverschattung von Strähnen und Haare
- Weiteres alternatives 3-D Modell

Abgrenzungskriterien

- Bedienungsumgebung mit Einstellmöglichkeiten (z. B. durch Regler)
- Bewegung der Haare

Vorgehen

Die Simulation von deformierbaren Objekten ist sehr zeitaufwändig und umfangreich, insbesondere die Dynamiksimulation und die damit verbundene Kollisionsbehandlung. Eine Vereinfachung der verwendeten Geometrie führt zwar zu einer Abnahme der visuellen Qualität, die Performanz steigt jedoch erheblich. Als Haarmodelle wurden deshalb mehrere 3D-Haarmodelle mit Maya erzeugt. Diese werden von Maya im Dateiformat Vrml 2.0 exportiert und mit einem, unter anderem mit Texturkoordinaten erweiterten, Vrml-Loader in OpenGL geladen und angezeigt.

Die Texturen wurden zuvor mit einem Grafikprogramm erzeugt, in Maya auf die Modelle gelegt, um korrekte Texturkoordinaten zu erhalten, eine Selbstverschattung angefertigt und in Open GL mit der Bibliothek DevIL geladen. Zur Laufzeit wird ein Shader-Programm auf das Modell angewendet.

3-D-Modelle

Erstellung der Modelle mit Alias Maya 6.0

Zu Begin wurden drei 3-D Modelle in unterschiedlicher Form entworfen, mit Maya Paint Effects, durch breite Strähnen (Rasterlook) und durch ein breites großes Basismodell. Vollständig verwirklicht wurde Letzteres.

Die Darstellung gliedert sich in Kopf, Basishaare, Strähnen und lose Haarsträhnen. Um das Haar weicher wirken zu lassen und um grobe Kanten zu vermeiden, werden die Haargrundmodelle mit Nurbs-Kurven gezeichnet, die geschwungene Kurven in jede Richtung ermöglichen, später zu Flächen verbunden und in Polygone umgewandelt. Beachtet werden muss ausdrücklich die richtige Ausrichtung der Normalen, die eine große Angriffsfläche für sich einschleichende Fehler bilden. Jede Strähne wird unabhängig von der anderen erstellt und abgeändert. Strähnen überkreuzen sich und sind unterschiedlich lang. Es soll kein symmetrisches Bild entstehen.

Zum Schluss werden alle Strähnen einer Ebene gruppiert, um sie als ein Modell zu exportieren und wieder in OpenGL zu laden.

Wichtig ist bereits in der Modellierung zu beachten, dass bei zu geringem Abstand der Modelle von einander ein Flackern in OpenGL entsteht.

Verwendete Modelle

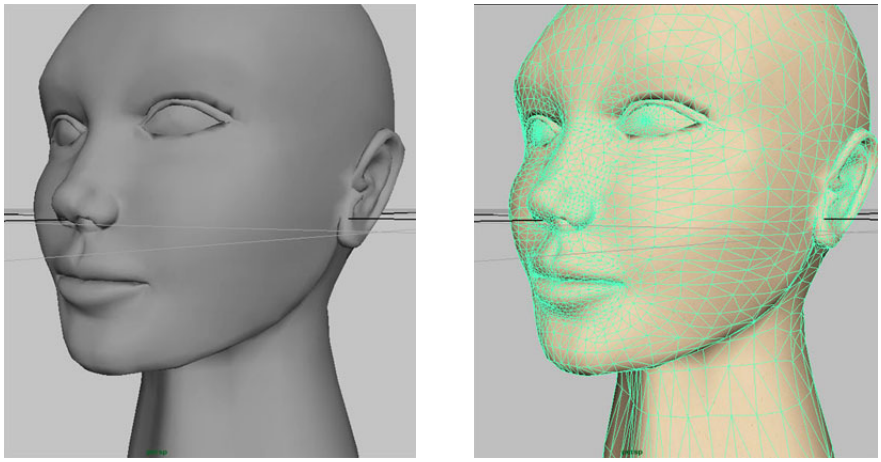

Abb. 5 u. 6: Kopf ohne und mit Textur.

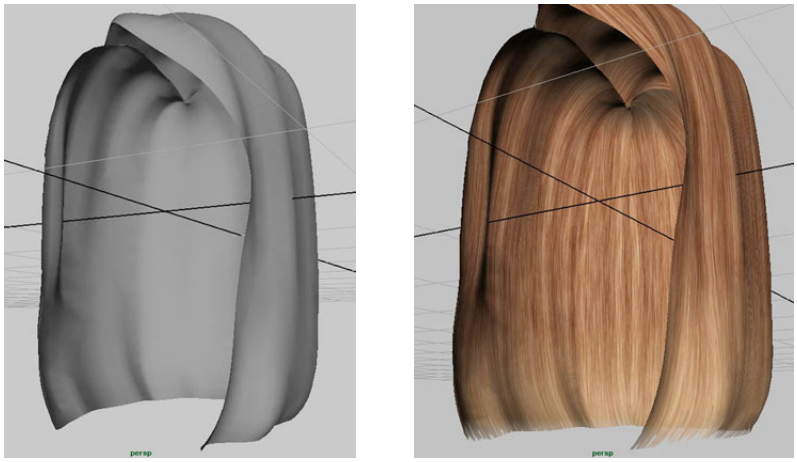

Abb. 7 u. 8: Haarbasismodell ohne und mit Textur.

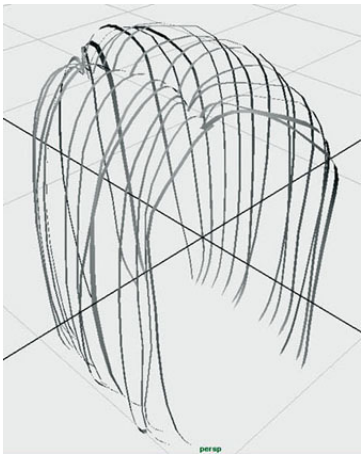
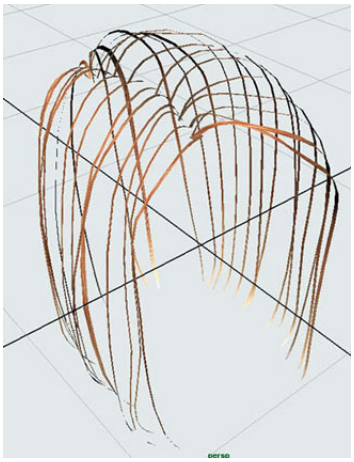

Abb. 9 u. 10: Strähnen ohne und mit Textur.

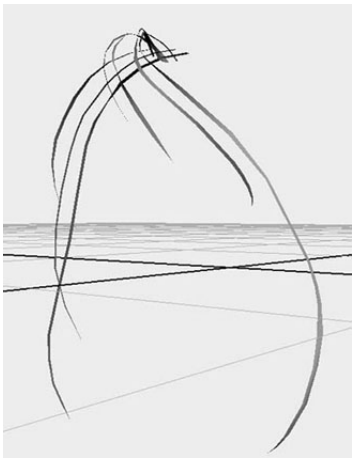
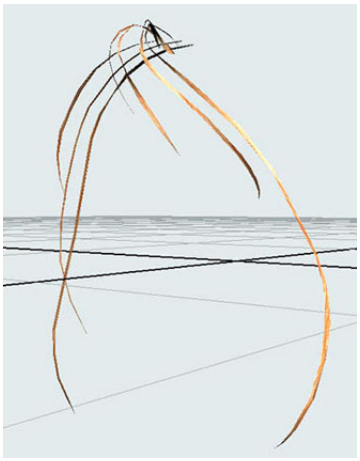

Abb. 11 u. 12: Lose Strähnen vorne ohne und mit Textur.

Abb. 13: Geladenes komplettes Modell mit Gold-Reflektion in OpenGL.

Vrml-Dateiformat

Mittlerweile zum internationalen Standard aufgestiegen, ist die Virtual Reality Modeling Language (Vrml) eine animationsfähige Grafik-Beschreibungssprache, die vorzugsweise für dreidimensionale Modelle verwendet wird. Ursprünglich für das Internet entwickelt, wird dieses Dateiformat mit der Endung „*.wrl“ heute vor allem als Austauschformat zwischen 3D-Animation- und Modellierungs-Programme genutzt. Vrml enthält eine Reihe von in einer Knotenstruktur angeordneten Geometriedaten.

Vrml 2.0

Aus Moving Worlds ging 1996 eine Weiterentwicklung Vrml 2.0 hervor. Dieses Format wird standardmäßig von den neuen Versionen des Modellierungs-Programms Alias Maya (im vorliegenden Fall Maya 6.0) per PlugIn exportiert.

Vrml 97

Vrml 97(-ISO 14772) ist ein festgelegter Standard. Er ist die Weiterentwicklung des Vrml 2.0 und weist nur geringe Änderungen zu seinem Vorgänger auf. Genau diese kleinen Änderungen erlaubten jedoch im vorliegenden Fall nur das Auslesen des Vrml 97 Formats, nicht das jedoch beider Formate. Die Gründe werden im hierauf folgenden Kapitel geklärt. Vrml 97 wird standardmäßig von neueren Versionen des Modellierungs-Programms 3-D-Studio-Max und von alten Alias Maya-Versionen, exportiert.

Verwendeter Vrml-Loader

Verwendet wird ein Vrml 97-Loader des Projektpraktikums ARLight WS 05/06 der Universität Koblenz. Aufbauend auf dieser Implementation füge ich durch Änderungen und Ergänzungen die weitere Funktionalität zum Laden von Textur-Koordinaten und dem Format Vrml 2.0 hinzu.

Die Handhabung von 3D-StudioMax-Modellen ist in diesem Fall unproblematisch. Diese werden von 3D-Studio-Max in ein beliebiges Vrml-Format übertragen. Maya bietet in diesem Fall einen nicht ganz so großen Komfort, es exportiert per PlugIn seine 3D-Objekt-Dateien nur in Vrml 2.0.

Der Versuch Modelle von Alias Maya in 3-D-Max zu importieren, zum Beispiel über die Formate WRL oder OBJ war zunächst erfolgreich. Problematischer erwies sich die Tatsache, dass nach einem Export über Vrml 2.0 von 3-D-Max in Maya keinerlei Eckpunktnormalen mehr, sondern nur noch Flächennormalen vorhanden waren. Nur mit Flächennormalen ausgestattete Modelle erscheinen sehr grob und kantig. Bei Ex- und Importen über das OBJ-Format (*.obj) kam es kontinuierlich zu mehreren fehlerhaften Ausrichtungen von Normalen insbesondere an komplizierten Bereichen des Modells. Beachtet werden sollte beim gegenseitigen Konvertieren, dass diese zwei Programme ihre Objekte standardmäßig an unterschiedlich ausgerichteten Koordinatensystem-Achsen positionieren.

Eine wichtige Eigenschaft des ursprünglichen Parsers[3] war, die Vrml-Befehle der Datei in vermuteter Reihenfolge abzuarbeiten. Vrml 2.0 und 97 wiesen dabei die folgende unterschiedliche Befehlstruktur auf:

```
Vrml 2.0

1.    coord DEF … Coordinate{ point [-0.315 0.312 -0.543, ... 0.057 -
      0.606 0.339] }
2.    coordIndex [1629 3031 44 -1, ... 2819 884 4090 -1]
3.    normal Normal { vector [-0.101 0.557 0.825, ... -0.534 -0.635] }
4.    normalIndex [0 1 2 -1, ... 2515 3681 4113 -1]
5.    texCoord DEF … TextureCoordinate{ point [0.443992 0.976652 ...
      0.047846 0.331228] }
6.    texCoordIndex [ 4 5 1 -1, ... 2805 4181 4624 -1 ]
```

[3] Parser (engl.: to parse „analysieren“ bzw. von lateinisch pars „Teil“; weshalb Parser im Deutschen ggtl. auch als Zerteiler bezeichnet werden.) Siehe http://de.wikipedia.org.

Vrml 97

```
1.    coord DEF … Coordinate{ point [-0.049 0.887 -0.521, ... 0.233 -
      1.149 0.228] }
3.    normal Normal{ vector [0.9218 -0.1557 0.355, ... 0.1669 0.8596] }
5.    texCoord DEF … TextureCoordinate{ point [0.444 0.9767 ... 0.3567
      3.022] }
2.    coordIndex [0, 1, 2, -1, ... 555, 24556, 24557, -1]
6.    texCoordIndex [0, 1, 2, -1, 3, 4, 5, -1 ..., 24523, 24524, -1]
4.    normalIndex [1615, 1615, 1615, -1, ..., 3008, 3008, -1]
```

Dies musste im ursprünglichen Vrml-Loader entsprechend berücksichtigt und ergänzt werden. Der Loader basiert auf dem Prinzip eines einfachen Parsers, der nach speziellen Befehlen im Vrml-Code sucht und dann nach diesen die darauf folgenden Daten einliest. Außerdem wurde das Auslesen von Texturkoordinaten hinzugefügt.

Textur

Eine sehr wichtige Rolle unabhängig von der Beleuchtung spielt die Textur. Wichtige Eigenschaften von Haaren sind schon mit ihrer Hilfe zu ermöglichen. Des Weiteren kann auch bereits mit der Textur räumliche Tiefe erzeugt werden.

Genau dies soll in der vorliegenden Arbeit und schon beim Erstellen der Texturen berücksichtigt werden. Schon mit einfachen Mitteln kann man einen gewissen Grad an räumlicher Tiefe in der Textur erzeugen. Dies passiert, angelehnt an die Malerei, mit dunklen Tönen.

Um zu Vermeiden, dass ein Modell wie eine große geometrische Form aussieht, gibt es zwei Möglichkeiten. Zum einen, die direkte Modellierung einzelner Haare, wie etwa durch Paint Effects von Maya, diese sind jedoch schwierig auszurichten und sehr rechenintensiv. Sind die Haare unterschiedlich lang, erhält man an den Haarspitzen einen Fransenlook. Bei einer geometrischen Form ist das nicht so einfach, dafür steigert sie die Performance erheblich. Texturiert man die Form, bleibt der Geometrie-Eindruck erhalten. Um den Geometrie-Eindruck zu vermeiden, wird ein Übergang in eine Transparenz kreiert, und um

den Fransenlook noch weiter zu verstärken, erzeugt man zackige Enden der Haare [12]. Der Effekt entsteht so auf der Textur und nicht durch das 3-D-Modell.

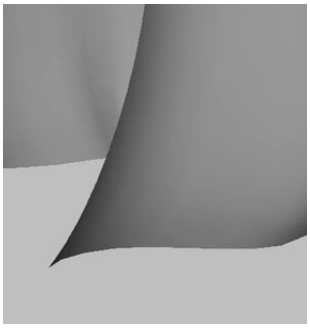


Abb. 14 u. 15: Unteres Ende des Basishaarmodells ohne und mit Textur.

Die Textur wurde mit Hilfe eines Bildbearbeitungsprogramms erstellt. Sie basiert auf einem eingefärbten Rauschen mit Verrechnung eines „Bewegungsunschärfe-Filters“. Hierdurch entsteht eine haarähnliche Linienbildung.

Abb. 16: Haartextur mit eingefärbtem Rauschen.

Um nun Tiefe zu erzeugen, sind weitere Farbabstufungen nötig. Für diesen Effekt werden in unregelmäßigen Abständen parallele Linien aufgehellt und abgedunkelt.

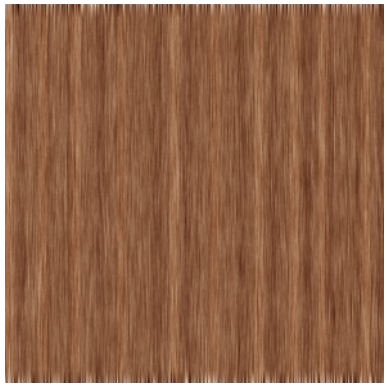

Abb. 17: Haartextur mit farblich abgesetzten Strähnen.

Im nächsten Schritt entstehen ein dunkler Haaransatz und helle Haarspitzen. Dies kann mit jeweils zwei halbtransparenten Farbverläufen nachgestellt werden.

Abb. 18: Haartextur mit dunklem Ansatz und hellen Spitzen mit voller Deckkraft.

Um nun das untere Ende in völlige Transparenz übergehen zu lassen, wird eine Maske erzeugt und für Fransen noch etwas mit Löschwerkzeugen nachgeholfen. Die fertige Textur sieht wie folgt aus:

Abb. 19: Im Modell verwendete fertige Textur mit Transparenz.

Das Haarmodell besteht aus drei unterschiedlichen Texturen. Sie unterscheiden sich insbesondere durch Färbung, aber auch durch unterschiedlich starke Transparenz.

Für das Basishaarmodell ist die dunkelste Haarfärbung verwendet worden. Die oben aufliegenden Strähnen erhielten eine etwas hellere Färbung, genau wie die losen Strähnen am Gesicht, die die hellste Farbe aufweisen.

Allen drei Texturen gemeinsam ist der Verlauf in die Transparenz an den Haarspitzen. Einige losen Strähnen am Gesicht des Modells sind darüber hinaus durchgängig transparent.

Die Textur wird bereits im Programm Maya richtig ausgerichtet, das sich durch die Form der Modelle als etwas problematisch gestaltet. Hierdurch sind die Texturkoordinaten später in OpenGL an ihrer gewünschten Position verfügbar.

Selbstverschattung

Wie uns schon das Beispiel von Scheuermann [12] zeigt, ist die Selbstverschattung maßgeblich an der Tiefenwirkung der Haaroberfläche beteiligt. Dies soll für das Haarmodell übernommen werden.

Schatten kann bekanntlich nur geworfen werden, wenn eine Lichtquelle existiert. Um dann die Umgebung mit all ihren Schatten zu erfassen bzw. zu

berechnen, wäre ein so genanntes Raytracing-Verfahren[4] notwendig, das vergleichsweise aufwändig zu implementieren ist.

Selbstverschattung mit Maya

Alternativen bildet das so genannte „Baken" von Licht und Schattenverhältnissen. Durch ein nicht ganz triviales Verfahren werden Licht und Schattenwerte auf die Textur addiert und dort festgeschrieben. Ziel dieses Vorgangs ist, dass die Textur den Schatten schon in ihre Farbwerte integriert. Diese Technik ist nicht neu und wird bereits erfolgreich in Spielen eingesetzt, um die Performance erheblich zu steigern. So soll es auch beim vorliegenden Haarbasismodell angewendet werden.

Die Idee für diese Studienarbeit ist mindestens ein zweites Modell zu erzeugen, das auf dem ersten aufliegt und auf diese Weise Schatten auf das untere werfen kann. Durch das Strähnenmodell wird auf den Grundhaaren dieser Effekt entstehen. Gleichzeitig wirken sie durch ein großes Grundmodell immer noch einheitlich und gekämmt.

Für das Baken werden Licht und Schattenparameter in Maya nach Wunsch gesetzt und auf eine geeignete Verschattung hin ausgerichtet. Zu beachten ist, dass hierfür nur Schatteneinstellungen des Depth-Map-Shadows möglich sind, nicht in den Renderzeit-intensiveren Ray-Trace-Shadows. Das gerenderte Bild mit seinen Farbwerten entspricht dann dem späteren Ergebnis.

Um Beleuchtung zu baken wird außer der Textur zuerst eine Ramp[5] erzeugt, die als Farbe ebenfalls die Texturdatei zugewiesen bekommt und auf der der Schatten als Farbe später integriert wird. UV-Koordinaten sind nur für die erste Textur nötig. Die UV-Koordinaten der Ramp können gelöscht werden. Beide Texturen ggf. Textur-Dateien und sämtliche Lichtquellen werden nun im Graph-Network in Maya zusammengebaut.

[4] Raytracing (Strahlverfolgung), Bezeichnung für Aussendung von Strahlen basierenden auf Algorithmus zur Verdeckungsberechnung; simulieren die Ausbreitung von Lichtstrahlen.
[5] Ramp, Textur-Einstellung im Programm Maya für einen Farbverlauf.

Die Lichtquellen werden durch einen Plus-Minus-Knoten addiert, in Maya 6.0 ist das mit dem folgenden Maya-Konsolenbefehl möglich:

```
connectAttr ObjectName . ObjectAttribute ObjectName .
ObjectAttribute;
```
[6]

Sowohl der Plus-Minus-Knoten als Textur und Ramp werden mit einem BlendColors-Knoten verbunden. Das Baken funktioniert über den im Hypershade von Maya aufrufbaren Menüpunkt „Convert to file“[8]. In den erweiterten Einstellungen dieses Befehls können verschiedene Bildformate ausgewählt werden, in diese dann exportiert werden soll. Das Ganze wird daraufhin fest in die Textur geschrieben und ist als separate Datei in Maya-Ordnern wieder zu finden.

Abb. 20: Selbstverschattung ohne Textur.

[6] Objectname muss durch Lichtname, ObjectAttribute durch Verbindungname ersetzt werden.

Abb. 21: Selbstverschattung mit Textur.

Problematik der Selbstverschattung

Die gebakte Textur kann von DevIL in OpenGL geladen werden. Der wie oben beschrieben erstellte Effekt erzeugt eine gewisse Tiefe durch Selbstverschattung der Strähnen auf dem Basishaarmodell.

Problematisch ist jedoch, dass Maya für bestimmte Bereiche, genauer Polygone, des Basishaarmodells auf dieselben Texturkoordinaten zugreift. Das verursacht eine mehrfache Überlagerung von Schattierung oder Beleuchtung an derselben Stelle der Textur und führt zu Artefakten, in diesem Fall auf dem Haar-Basismodell. Um dieses Problem zu vermeiden, müsste jedem Polygon des Haarmodells einmalige Texturkoordinaten zugewiesen werden, was wiederum ein enorm hoher Aufwand oder unter Umständen sehr viele Texturen bedeuten würde.

Abb. 22: Überlagerung der Polygone auf der Textur.

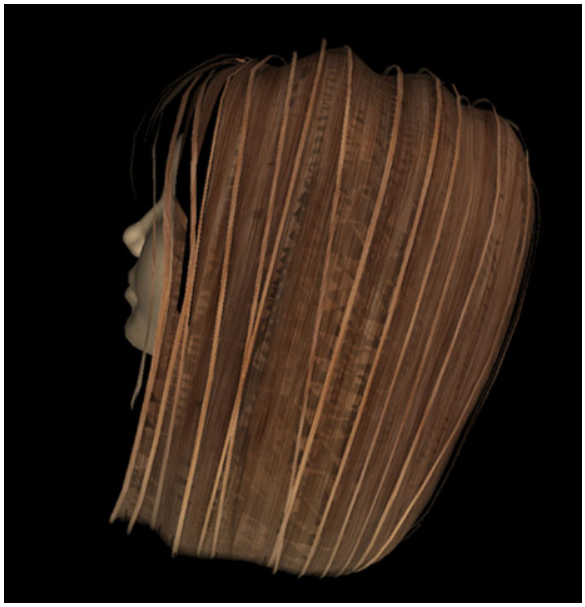

Abb. 23: Überlagerte, gebakte Textur auf Modell.

DevIL Bibliothek

Bei DevIL handelt es sich um eine freie plattformunabhängige Bild-Bibliothek, die das Laden der Dateiformate BMP, CUT, DCX, DDS, ICO, GIF, JPG, LBM, LIF, MDL, PCD, PCX, PIC, PNG, PNM, PSD, PSP, RAW, SGI, TGA, TIF, WAL, ACT, PAL, HDR und das Speichern von BMP, DDS, JPG, PCX, PNG, PNM, RAW, SGI, TGA, TIF, PAL und HDR unterstützt.

Für die Umsetzung der Haartextureigenschaften ist es notwendig transparente

Texturen von Dateien zu laden, in Frage kommen hierfür die Formate TIF, GIF und PNG[7]. Mit OpenGL und mit der Bibliothek Gaux ist das Laden des für diese Arbeit verwendeten PNG-Formats nicht möglich. Die Entscheidung fiel deshalb auf die frei zur Verfügung stehenden Bibliotheken von DevIL.

Man sollte jedoch beachten, dass DevIL die Formate teilweise verdreht wiedergibt. Der Ursprung eines Bild-Koordinatensystems, z. B. einer Bitmap-Datei liegt oben links. Das bedeutet, das Bild wird nicht wie von mathematischer Seite und OpenGL bekannt von unten links ausgelesen, sondern von oben links. Die Tatsache, dass BMP-Dateien richtig herum geladen werden, könnte ein automatisches Drehen von DevIL sein, etwa durch ein rückwärtiges Auslesen bedingt. Das Ergebnis wäre ein wieder korrekt ausgerichtetes Bild. Bei anderen Formaten wie z. B. dem PNG oder JPG-Format liegt die Textur als Ergebnis falsch herum, genauer vertikal gespiegelt auf dem Modell. Durch das von DevIL berücksichtigte Drehen bzw. rückwärtige Auslesen erscheint das Bild somit falsch herum!

Um das Problem zu umgehen, kann die Textur entweder von OpenGL neu berechnet oder in einem Grafikprogramm vertikal gespiegelt (nicht 180° gedreht) und wieder gespeichert werden.

Cg Shader

Cg (C for Graphics) ist eine auf C++ aufbauende GPU[8]-Programmiersprache. Diese von NVIDIA entwickelte Umgebung wird vor allem für schnelle und realistische Darstellung von erweiterten, visuellen Effekten genutzt.[9]

Für die verbesserte Beleuchtung der Haare ist ein Phong-Shading implementiert, das mit der gewöhnlichen OpenGL Pipeline nicht erreicht werden kann, hier ist nur Gauroud-Shading möglich. Phong interpoliert, im Gegensatz zu Gauroud, die Normalen der Eckpunkte, errechnet die Farbe und weist sie dann den Pixeln zu.

---

[7] PNG (Portable Network Graphic; gespr. Ping): wurde kreiert, um Lizenzproblemen, die beim GIF-Format bestehen, zu meiden; außerdem sind noch weitere Funktionen hinzu gekommen; unterstützt eine Farbtiefe von 24 Bit (16 Millionen Farben); bietet mehrere Transparenzfarben und ist für Animationen geeignet. [11]

[8] GPU (Graphics Processing Unit) Grafikprozessor; entweder auf Grafikkarte oder auf Hauptplatine.

[9] Weitere Informationen unter: http://developer.nvidia.com/page/home.html

Abb. 24: Haarmodell mit angewandtem Shader

Aufbau des Programms

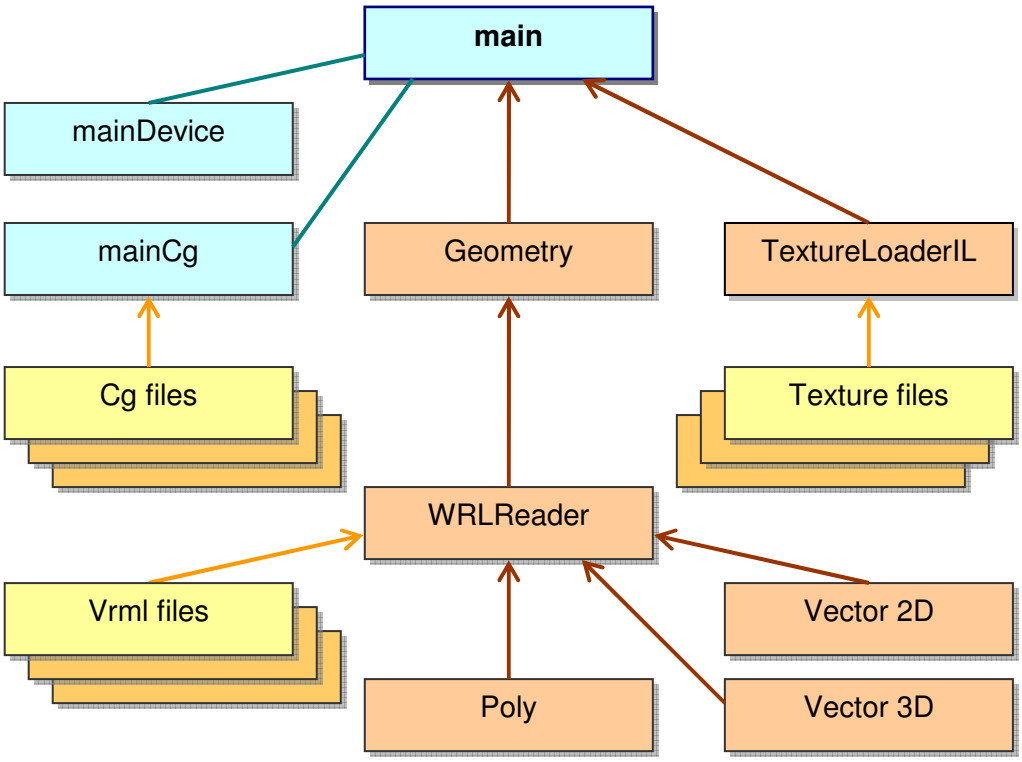

Abb. 25: Aufbau der Implementierung

Einige Bereiche des Main-Programms sind in die Dateien, „mainCg“ und „mainDevice“ ausgelagert. Wie der Name vermuten lässt, handhabt „mainDevice“ Eingabe und Navigation und „mainCg“ die Vertex- und Fragmentshader. Die Klasse „Geometry“ führt das Laden, Auslesen und Zeichnen der 3-D Modelle aus. Die Klasse „TextureLoaderIL“ lädt Grafikdateien und speichert diese als OpenGL-Textur ab.

Für jedes Modell existiert im Hauptprogramm eine eigene Methode, die seine spezifische Transformation, die Aufrufe eines TextureloaderIL-Objekt und eines Geometry-Objekt lenkt. Durch ein Kontextmenu ist es möglich zwischen Shadern zu wechseln oder sich das Ergebnis der gebakten Textur ausgeben zu lassen. Außerdem ist eine Navigation des Kopfmodells mit Maussteuerung und den Pfeiltasten in Echtzeit möglich.

Weiterführende Aufgaben

Auf dieser Arbeit aufbauend könnte man sich intensiver mit weiteren Aufgabenfeldern des Themas „Darstellung von Haaren“ beschäftigen. Insbesondere die Beleuchtung und ihre unterschiedlichen Verfahren sind ein interessantes, für realistische Färbung der Haare nötiges Arbeitsfeld. Leider konnte dieser Bereich in der vorliegenden Arbeit zwar analysiert, aber aufgrund von zeitlichen Umständen nur kurz angerissen werden.

Umfangreiche, zu erarbeitende Aufgaben sind nach wie vor die unterschiedlichen Verhaltensweisen der anisotropen Eigenschaften von Haaren. Damit in Verbindung steht das Erscheinungsbild der Farbe bei diversen Einstrahlungen von Licht, gemeint ist hier die Haarfarbe mit all ihren facettenreichen Spiegelungen. Hier wären nähere Untersuchungen und Implementationen verschiedener Beleuchtungen, eventuell durch Shader realisiert, nötig und wünschenswert.

Unabhängig davon gibt es sicherlich weitere Ansätze von statischen 3-D-Modellen.

Weiter könnte man sich des Themas des beweglichen Haarmodells annehmen. Hierzu kämen zwei Möglichkeiten in Betracht, eine modellierte Animation oder das Implementieren der Haare mit unterschiedlich existierenden, mathematischen Verfahren. Beides könnte man dann wiederum mit Beleuchtung verbinden.

Ausblick

Aktuelle Filme wie „Ice Age 2“ zeigen große Fortschritte der Technik in der realistischen Darstellung von Haaren, insbesondre bei der von Fell. Dieser Forschritt ist auch unter Laien populär und offensichtlich. Fell ist nicht mehr gleich Fell. Man ist schon so weit unterschiedliche Fell-Typen, ihre Eigenschaften und Verhaltensweisen sehr getreu zu simulieren.

Abb. 26 u. 27: Unterschiedliche Felltypen des vorzeitlichen Streifenhörnchen „Scrat“, des Säbelzahntigers „Diego“ und des Opossums aus dem Animationsfilm Ice Age 2.

Die Rechenleistung ist mittlerweile so weit fortgeschritten, dass eine große Anzahl von Haaren keine unüberwindbaren Probleme mehr schafft. Demzufolge wird aller Wahrscheinlichkeit nach die Forschung nun an einen Punkt angelangen, an dem es intensiverer Anstrengung bedarf, sehr genaue physikalische Eigenschaften von Haar und Fell zu ergründen.

Nach eigenen Beobachtungen gibt es noch andere wichtige, zu berücksichtigende Punkte um sich einer naturgetreuen Nachbildung von Haaren und Fellen anzunähern. Ein Beispiel ist die Tatsache, dass Haare und Fell im Unterschied zur Natur in vielen bekannten Darstellungen perfekt sauber, geordnet und geglättet sind. Dies entspricht nicht immer der Realität.

Abb. 28 u. 29: Verfärbtes und verfilztes Fell

Dort sind Felle oft teilweise verfärbt, verkrustet oder verfilzt, mit anderen Worten es fehlen die Unreinheiten in der Darstellung. Vorausblickend darf man behaupten, dass wohl immer noch mancherlei Arbeit und Beobachtung erforderlich sein wird, um der Realität zu entsprechen.

Literaturverzeichnis

[1] *Vrml 97 (12.05.2006):* http://www.sdsc.edu/~moreland/courses/Siggraph98/vrml97/vrml97.htm.

[2] *DevIL Bibliothek (12.05.2006):* http://openil.sourceforge.net.

[3] *OpenGL GLUT (12.05.2006):* http://pyopengl.sourceforge.net/documentation/ref/glut.html.

[4] *OpenGL Quick Reference Guide (12.05.2006):* http://www.cs.umd.edu/users/mount/427/OpenGL/ogl_ref/ogl_ref.html,

[5] *Shreiner, Woo, Neider, Davis (Juli 2005)*: "OpenGL Programming Guide 5th Edition: The Official Guide to Learning OpenGL", Version2 by Addison Wesley,.

[6] *MSDN Library (12.05.2006)*: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/opengl/apxb4_82lh.asp

[7] *OpenGL Wiki (12.05.2006):* http://wiki.delphigl.com/index.php/OpenGL.

[8] *Tutorial Baking in Maya (12.05.2006):* http://www.highend3d.com/maya/tutorials/rendering_lighting/general/145.html.

[9] *Fernando, R. und Kilgard M.*: "Cg The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics", NVIDIA; Addison Wesley.

[10] *NVIDIA:* "Cg Toolkit: User's Manual, A Developer's Guide to Programmable Graphics".

[11] *Markt und Technik (2004):* "Photoshop CS magnum".

[12] *T. Scheuermann, 3D Application Research Group ATI Research, Inc.:* "Hair Rendering and Shading".

[13] *T.-Y. Kim, U. Neumann:* "A Thin Shell Volume for Modeling Human Hair", University of Southern California.