Aufgabenstellung zum Praktikum Computergraphik

Universität Ulm
Abteilung Medieninformatik
Prof. Dr. M. Weber

Sommersemester 2008

Inhaltsverzeichnis

1 Inhalt des Praktikums

Das Praktikum befaßt sich mit grundlegenden Algorithmen und Problemen der Bildsynthese. Dazu werden zuerst das Abtasten und die durch Abtasten entstehenden Artefakte untersucht. Danach wird ein einfacher Strahlverfolgungsalgorithmus entwickelt und daran die Probleme der Approximation von Geometrie und numerischer Rechenungenauigkeiten erörtert. Mit der Beschleunigung des Strahlschnitts durch Raumaufteilung wird es schließlich möglich, komplexe Schattierungseffekte in Betracht zu ziehen. Auch hier müssen die allgegenwärtigen Unzulänglichkeiten der Fließkommaeinheiten berücksichtigt werden. Die letzte Aufgabe des Praktikums bietet Raum zur selbständigen Gestaltung und wird aus vorgeschlagenen Projekten ausgewählt.

1.1 Literatur

Literatur zum Praktikum:

- Peter Shirley, R. Keith Morley: Realistic Ray Tracing, 2nd Ed., AK Peters.
- Markus Geimer, Stefan Müller: A Cross-Platform Framework for Interactive Ray Tracing, 2003
- Ingo Wald, Solomon Boulos, and Peter Shirley: Ray Tracing Deformable Scenes using Dynamic Bounding Volume Hierarchies, 2006
- Peter Shirley: Fundamentals of Computer Graphics, AK Peters.
  Kapitel Ray Tracing online verfügbar

Literatur zur Programmiersprache C++ :

- Alexander Keller: C++ Crash Kurs
- Kyle Loudon: C++ - kurz & gut, O'Reilly , ISBN 3-89721-262-5
- Steve Oualline: Praktische C++-Programmierung, O'Reilly ISBN 3-89721-358-3

2 Abtasten und Rekonstruktion ($\frac{3}{2}$ Wochen Bearbeitungszeit)

Es ist meist unmöglich die der Bildsynthese zugrunde liegenden Integrale analytisch auszuwerten und daher muß man sich mit Approximationen durch numerische Algorithmen begnügen. In der Computergraphik bedeutet das, die Integrale durch gewichtete Mittel aus Stichproben des Integranden zu bestimmen. Man rekonstruiert also das eigentliche Bild durch Abtasten des Integranden an ausgewählten Punkten. Die Probleme solcher Approximationen werden schon an einfachen Testfunktionen deutlich.

Die Funktion

$$\begin{aligned} Z_1 : [0,1)^2 &\rightarrow [0,1] \\ (x,y) &\mapsto \frac{1}{2}\left(1+\sin 2\pi(4x)^2\right) \end{aligned}$$

ist eigentlich nur eindimensional und stellt eine Sinusfunktion dar, deren Frequenz mit der positiven x-Achse zunimmt.

Ein Testmuster (*Campbell-Robson contrast sensitivity chart*), mit dem man normalerweise das Abtastverhalten des Auges überprüft, ist

$$\begin{aligned} Z_2 : [0,1)^2 &\rightarrow [0,1] \\ (x,y) &\mapsto \frac{1}{2}\left(1+(1-y)^3 \cdot \sin\left[2\pi x e^{5x}\right]\right), \end{aligned}$$

wobei nun zur zunehmenen Frequenz längs der positives x-Achse entlang der y-Achse die Amplitude moduliert wird. Wird dieses Muster in einer ausreichenden Auflösung hergestellt, kann man *sehen*, wie weit das Auge in der Lage ist, die einzelnen Frequenzen noch wahrzunehmen.

$$\begin{aligned} Z_3 : [0,1)^2 &\rightarrow [0,1] \\ (x,y) &\mapsto \frac{1}{2}\left(1+\sin\left(1600\cdot\left(x^2+y^2\right)\right)\right) \end{aligned}$$

ist eine typische Testfunktion um Aliase zu verdeutlichen. Wird diese Funktion nicht richtig abgetastet, entstehen Moirée-Muster. Das sind im Prinzip dieselben Muster, die entstehen, wenn man mit einer Digitalkamera karierte Sakkos photographiert. Deshalb tragen Fernsehmoderatoren auch nie kleinkarierte Sakkos.

Komplementär zu den modulierten konzentrischen Wellen um den Ursprung ist

$$\begin{aligned} Z_4 : [0,1)^2 &\rightarrow [0,1] \\ (x,y) &\mapsto \frac{1}{2}\left(1+\sin\left(60\cdot 4\pi\cdot\arctan\frac{y}{x}\right)\right) \end{aligned}$$

der entsprechende Fächer um den Ursprung.

Die Funktionen Z_1 bis Z_4 sind unendlich oft stetig differenzierbar und somit sind $Z_1, \ldots, Z_4 \in C^\infty([0,1)^2)$. Das ist interessant für Fourieranalyse und erlaubt auch teilweise analytische Lösungen. In der Computergraphik jedoch kommen die häufigsten Probleme durch Unstetigkeiten zustande. Indem Sie die Funktion

$$\text{positive}(x) := \begin{cases} 1 & x > 0 \\ -1 & \text{sonst} \end{cases}$$

vor jedem Sinus in Z_1 bis Z_4 einfügen, werden die Graurampen durch Kanten ersetzt. Die entstehenden Funktionen werden oft in sehr hoher Auflösung auf Film[1] belichtet, um optische Apparaturen auf ihre Eigenschaften zu prüfen.

Aufgabe 1 (Abtastartefakte - Aliasing)**.** Implementieren Sie die Funktionen Z_1 bis Z_4 mit einem Flag, das es erlaubt zwischen den stetigen und unstetigen Versionen umzuschalten. Berechnen Sie von den Funktionen quadratische Bilder frei wählbarer Auflösung, indem Sie die Funktion jeweils in der Mitte eines Pixels abfragen. Speichern Sie die Bilder systematisch (z.B. indem Sie automatisch .png Bilder in pdfLaTeX einbinden und kommentieren). Begründen Sie die entstehenden Artefakte und deren Natur.

2.1 Reduktion von Abtastartefakten (Anti-Aliasing)

Schaut man sich die Bilder aus Aufgabe 1 an, so stellt man fest, dass diese nur wenig an die abzutastenden Funktionen erinnern, sondern vielmehr an andere. Dies nennt man *Aliasing*. Der Begriff *Anti-Aliasing* umfaßt die Methoden zur Minderung dieser Artefakte.

Neben den offensichtlichen Methoden wie Erhöhen der Abtastrate und/oder Filtern der Funktion, hängt der Erfolg der angewendeten Verfahren auch wesentlich davon ab, was man über den Integranden weiß.

2.1.1 Reduktion durch Erhöhen der Abtastrate

Bisher wurde eine Stichprobe genau im Pixel-Mittelpunkt ausgewertet. Um ein besseres Bild zu erzielen, muß allerdings das Mittel der Funktion über die Fläche des Pixels ausgerechnet werden. Wenn analytische Lösungen nur schwierig oder gar nicht zu berechnen sind, muß man hier auf numerische Approximationen ausweichen. In der folgenden Aufgaben werden verschiedene Quadraturformeln zur Approximation der Pixelintegrale miteinander verglichen.

[1] Diese Filme werden dann *Zoneplates* genannt.

Aufgabe 2. Anstatt nur eine Stichprobe im Pixelmittelpunkt zu nehmen, werden nun mehrere Stichproben gemittelt, indem ihre Summe durch die Anzahl geteilt wird. Je nach dem, wo die Stichproben genommen werden, ergibt sich ein anderes Verhalten bezüglich des Anti-Aliasing.

In dieser Aufgabe entstehen viele Daten. Organisieren Sie Ihre Ergebnisse so, dass sie nachvollziehbar und vergleichbar bleiben.

1. Die üblicherweise bereitgestellten Pseudo-Zufallszahlengeneratoren liefern oft sehr schlechte Simulationen von Zufallszahlen. Benutzen Sie daher den Generator Mersenne Twister, um zufällige Punkte im Pixel zu generieren. Untersuchen Sie systematisch die Funktionen $Z_1, \ldots, Z_4$, auch in den unstetigen Versionen, für $n = 1, \ldots, 100$ Stichproben.

2. Verwenden Sie reguläre Gitter zur Rekonstruktion der Funktionen. Teilen Sie dazu die Pixel in $m \times m$ Unterpixel, wobei $m = 1, \ldots, 10$. Wie entwickelt sich die visuelle Qualität im Vergleich zu den zufälligen Stichproben?

3. Variieren Sie die regulären Gitter des vorhergehenden Aufgabenteils, indem Sie statt der Mittelpunkte in den Unterpixeln zufällige Punkte benutzen. Was beobachten Sie?

4. Ein Problem der Gitter ist das schnelle Wachsen der Anzahl von Zellen. Im *Latin-Hypercube-Sampling* (auch n-*Rooks-Sampling*) teilt man jede Achse des Integrationsgebietes (hier das Pixel) in n Intervalle und füllt dann die Zellen auf der Diagonalen mit je einem zufälligen Punkt. Danach werden die Intervalle entlang jeder Achse zufällig permutiert (also die Spalten und Zeilen). Testen Sie das Verfahren und vergleichen Sie es für $n = 1, \ldots, 100$ Stichproben mit den anderen. Implementieren sie hierfür eine eigenständige Routine zur Erzeugung zufälliger Permutationen.

5. Implementieren Sie Rang-1 Gitter der Form

$$p_i = \frac{i}{n}(1, a) \bmod 1 \qquad i = 0 \cdots n - 1.$$

   Wählen Sie den Parameter a mit Hilfe des Spektraltestes (D. Knuth, The Art of Computer Programming, volume 2, Seminumerical Algorithms, Addison Wesley) so, dass der minimale Abstand zwischen zwei Punkten maximiert wird. Testen Sie auch hier das Verfahren für verschiedene n. Was beobachten Sie?

   Erweitern Sie das Verfahren indem Sie zusätzlich Cranley Patterson Rotation verwenden.

Konvergieren die Verfahren wenn Sie die Abtastrate erhöhen? Begründen Sie Ihre Beobachtung.

2.1.2 Reduktion durch bessere Basisfunktionen

Das Abtasttheorem von Shannon besagt, dass man bandbegrenzte Funktionen mittels der sinc-Funktion exakt durch Stichproben rekonstruieren kann. Bisher haben wir als Basisfunktion nur die charakteristische Funktion eines Pixels verwendet. Im Gegensatz zur sinc-Funktion hat diese grobe Annäherung den Vorteil, dass ihr Träger (also der Bereich, wo die Funktion verschieden von Null ist) genau die Pixelfläche und somit endlich ist.

Aufgabe 3. Auch wenn in der Computergraphik die Vorraussetzungen für Shannon's Theorem so gut wie nie erfüllt sind, soll in dieser Aufgabe die Rekonstruktion verbessert werden, indem die sinc-Funktion besser approximiert wird.

1. Implementieren Sie das Box-Filter für beliebige Trägergrößen. Wie entwickeln sich die Aliase? Berücksichtigen Sie, dass Sie nun mehr als nur das sichtbare Bild abtasten müssen, da die Träger größer als ein Pixel sein können.

2. Implementieren Sie nun Filter, die mit ungleichen Gewichten arbeiten: Dreiecksfilter, Gauss-Filter und Mitchell's bikubisches Filter. Wie kann man die Filteralgorithmen effizient implementieren? Nehmen Sie dazu an, dass die Berechnung eines Samples teuer ist.
   Beachten Sie: Die Bilder sollen nicht nachträglich gefiltert werden. Das Filter soll direckt auf die Samples (bzw. deren Gewichtung) angewendet werden!

2.1.3 Tone Mapping

Die Erhöhnung der Abtastrate hilft nicht immer schnell genug, vor allem, wenn die Funktion nicht auf $[0, 1)$ beschränkt sind. Dies ist jedoch häufig der Fall in der Computergraphik, da zum einen eine physikalisch korrekte Lichtsimulation viel hellere Farben erzeugt, als der Bildschirm darstellen kann, und immer mehr so genannte *High-Dynamic-Range-Images* als Lichtquellen eingesetzt werden, die den ganzen Dynamikumfang des wahrnehmbaren Lichtes erfassen.

Aufgabe 4. Um die Auswirkungen hoher Dynamik auf die Konvergenz unserer Integrationsmethoden zu verstehen, verwenden wir ein einfaches Beispiel: Ein Bild mit hoher Dynamik und hoher Auflösung soll verkleinert werden.

1. Schreiben Sie sich eine Funktion

$$Z_{\text{HDR}}(x, y),$$

die das vorgegebene Bild mit großem Dynamikumfang einliest und für die ganzzahligen Koordinaten (x, y) die entsprechenden Werte für rot, grün und blau zurückgibt. Berechnen Sie eine Verkleinerung des Bildes auf ein Viertel Größe, indem Sie je vier benachbarte Pixel mitteln. Was ist festzustellen? Die entsprechenden Bilder und notwendigen Programmfragmente finden Sie auf der Web-Seite des Praktikums.

2. Was passiert, wenn Sie vor dem Mitteln die Luminanz L auf 1 begrenzen? Dazu müssen die Farbwerte in einen Farbraum mit Luminanz abgebildet werden.

3. Das unstetige Begrenzen der Luminanz entspricht nicht dem Verhalten eines Films, das eher der Abbildung

$$L_m = \frac{L}{L+1}$$

genügt. Was sind Vor- und Nachteile dieser Abbildung?

2.1.4 TFT-Bildschirme

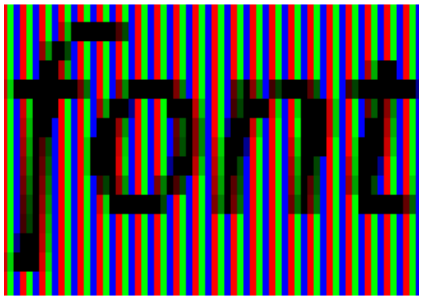


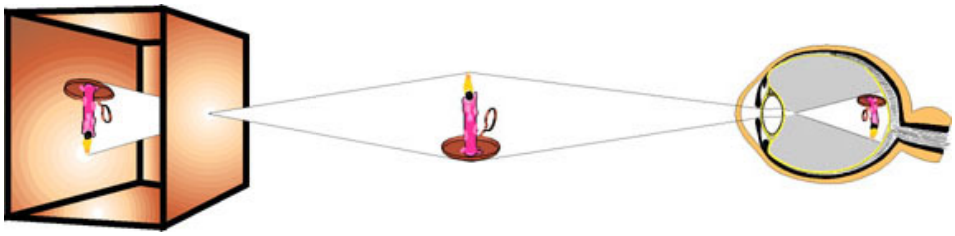
Abbildung 1: Anti-Aliasing von Buchstaben auf einem TFT-Bildschirm mittels der ClearType-Technologie.

Ein einzelnes Pixel eines TFT-Bildschirms besteht aus drei "'Subpixeln"': Ein rotes, ein grünes und ein blaues. Das menschliche Auge nimmt dieses Triplet als ein Pixel wahr. Die ClearType-Technologie verwendet diese direkt adressierbaren Subpixel, um die effektive horizontale Auflösung des Bildschirms zu verdreifachen. Während bei der "'traditionellen"' Bildsynthese angenommen wird, dass die drei Farben Rot, Grün und Blau übereinanderliegen, soll die andere Beschaffenheit von TFT-Bildschirmen nun beim Anti-Aliasing berücksichtigt werden (siehe auch Bild 1).

Aufgabe 5. Informieren Sie sich zuerst über die ClearType-Technologie, indem Sie mit Hilfe des Internets die entsprechenden Artikel suchen. Finden Sie dann heraus, wie die Farbtripel auf Ihrem Bildschirm angeordnet sind. Erweitern Sie schließlich Ihre Anti-Aliasing Algorithmen um einen Modus für TFT-Bildschirme, wobei die Anordnung der Farbtripel wählbar sein soll. Weisen Sie subjektiv die Verbesserung am Bildschirm nach.

3 Einfache Strahlverfolgung (2 Wochen Bearbeitungszeit)

Um von mathematischen Beschreibungen von Objekten in Szenen Bilder zu berechnen, muß das Verdeckungsproblem gelöst werden. Das Auge funktioniert ähnlich einer Lochkamera (lat. *Camera Obscura*). Zur Bildsynthese werden Strahlen von der Bildebene durch den Punkt, der als Objektivöffnung dient, verfolgt. Die Farbe des ersten Treffers wird dann dem Strahl als Ergebnis zugeordnet.



3.1 Datensätze

Die Szenen werden im Blender-Format zur Verfügung gestellt. Ein Skript zum extrahieren der Geometridaten in das `*.ra2` Format ist in den `*.blend` Dateien enthalten. Code zum laden von `*.ra2` Dateien findet sich ebenfalls auf der Web-Seite des Praktikums. Zum extrahieren der Daten muss lediglich der Dateipfad angepasst werden und das Skript muss mit Alt+p ausgeführt werden. Blender selbst finden Sie unter www.blender.org für jedes gängige Betriebssystem.

3.2 Schnitt eines Strahls mit einer Liste von Dreiecken

In professionellen Rendering-Programmen wird die Geometrie meist einheitlich durch Dreiecke oder Vierecke approximiert. Wir wählen die etwas einfacher zu handhabenden Dreiecke.

Aufgabe 6. Entwerfen Sie eine Strahlschnittroutine, die den nächsten Schnittpunkt eines Strahls mit einem Feld von Dreiecken bestimmt. Das Gültigkeitsintervall des Strahls fängt immer bei Null an und reicht bis Unendlich. Versuchen Sie die Routine so effizient wie möglich zu gestalten. Implementieren Sie die beiden Strahl-Dreieck Schnitttests Möller-Trumbore (Google) und Badouel (Google + MonteCarlo Skript) und vergleichen diese auf Geschwindigkeit und ”Gleichheit” der Ergebnisse (*float*).

Um das Programm zu testen, verwenden Sie den Datensatz `Test.blend`. Berechnen Sie davon ein Bild, indem Sie vom Ursprung Strahlen durch die Bilde-

bene, die in $z = -128$ senkrecht auf der z-Achse steht, schießen. Der Bildbereich in der xy-Ebene ist $[-128, 128]^2$.

Schattieren Sie einen Treffer entweder mit der Nummer des Dreiecks als Grauwert oder durch die Entfernung vom Auge zum Trefferpunkt.

3.3 Achsparallele Begrenzungsquader

Ein fundamentales Konzept der Computergraphik ist der achsparallele Begrenzungsquader. Seine Erstellung ist ganz einfach: Von einem zu begrenzenden Objekt sucht man die jeweils minimalen und maximalen Koordinaten. Die resultierenden zwei Punkte beschreiben dann eindeutig den einschließenden Quader.

Aufgabe 7. Verbessern Sie Ihr Programm dadurch, dass Sie achsparallele Begrenzungsquader benutzen. Dazu müssen Sie zuerst den Quader bestimmen. Die Effizienz läßt sich nun verbessern, indem Sie die zu verfolgenden Strahlen erst mit dem Quader verschneiden, also Ein- und Austrittspunkt bestimmen. Gibt es keinen Schnitt, so muß der Strahl nicht verfolgt werden. Die Routinen sollen derart gestaltet werden, dass sie später gut verwendet werden können. Besonderes Augenmerk ist auf die numerische Stabilität zu legen.

3.4 Eine Anwendung zur Betrachtung von Szenen

Die Rechner sind heutzutage so schnell, das interaktives Ray Tracing möglich ist. Für einfache Szenen ist das sogar mit der linearen Suche des nächsten Schnittpunktes zu erreichen.

Aufgabe 8. Verwenden Sie Ihre bisher entworfenen Programmteile, um eine einfache Anwendung zu implementieren, die es erlaubt Szenen zu laden und interaktiv zu betrachten. Es soll eine Schnittstelle für den Benutzer geschaffen werden, damit sich dieser Interaktiv durch die Szene bewegen kann (z.B. *Quakestyle, Flugzeug oder Rolling-Ball-Metapher*). Auch ist es sinnvoll Bilder zu speichern und deren Spezifikation wie Szene und Kameraeinstellungen zu sichern. Beim Programmstart soll eine sinnvolle Außenansicht des Modells automatisch gewählt werden.

Im Gegensatz zu Anwendungen auf Graphikkarten wird beim Ray Tracing die Kamera modifiziert und nicht jeweils die Szene transformiert. Ein geeignetes Kameramodell ist gegeben durch einen *Up*-Vektor, der sozusagen die Aufrichtung des Betrachters definiert, einen Augpunkt und eine Blickrichtung, auf der die Bildebene senkrecht steht. Die Blickrichtung kann natürlich auch durch einen Punkt, auf den geschaut wird, angegeben werden. Schließlich gibt die Brennweite an, wie weit die Bildebene vom Augpunkt entfernt ist.

3.5 Numerische Stabilität

Auf dem Computer wird nicht mit reellen Zahlen, sondern mit Fließkommaarithmetik gerechnet. Der darstellbare Zahlenumfang ist nur eine kleine Teilmenge der rationalen Zahlen. Die unvermeidbaren Rundungsfehler der Berechnungen wirken sich daher auf die Präzision aus.

Aufgabe 9. Untersuchen und erklären Sie die Effekte, die sichbar werden, wenn Sie die Testszenen `Degenerate.blend`, `AxisAligned.blend` und `SamePlane.blend` betrachten.

3.6 Approximation von Oberflächen

Außer in der Architektur wird hauptsächlich mit NURBS oder Unterteilungsflächen modelliert. Beim Rendering dagegen ist es meist effizienter die Freiformflächen zu tesselieren, also in ebene Polygone zu zerlegen und dann die Strahlschnitte auf dieser Approximation durchzuführen.

Aufgabe 10. Auf der Web-Seite des Praktikums finden Sie die Originaldaten des Utah-Teapots als Bezier-Patches. Machen Sie sich mit der Datenstruktur vertraut und schreiben Sie eine Routine, die durch Unterteilung die Bezier-Patches mit Dreiecken approximiert. Die Anzahl der Unterteilungen, also die Genauigkeit, soll einstellbar sein. Stellen Sie die entstehende Szene mit Ihrer Applikation zum Betrachten dar. Berechnen Sie die Oberflächennormalen zunächst aus den Dreiecken.

1. Verbessern Sie die Berechnung der Normalen und benutzen Sie diese zum Schattieren der Treffer, indem Sie den Cosinus zwischen Blickrichtung und Normalen berechnen.

2. Welche Möglichkeiten der Unterteilung gibt es? Wie entwickeln sich Qualität und Leistung des Systems mit wachsender Unterteilung?

3. Welche Probleme ergeben sich durch solche Tesselierungen?

W

4 Beschleunigte Strahlverfolgung (2 Wochen Bearbeitungszeit)

Die lineare Suche des ersten Schnittpunkts eines Strahls mit den Dreiecken ist langsam. Viel effizienter ist es, die Szene in kleinere Volumenelemente (engl. Voxels) zu unterteilen und nur die Dreiecke zu testen, die in den vom Strahl geschnittenen Voxeln liegen. Eine entsprechende Datenstruktur muss in einem Vorverarbeitungsschritt aufgebaut werden.

In den folgenden Aufgaben geht es darum, den Aufbau dieser Datenstruktur und den Strahlschnitt so schnell wie nur irgendmöglich durchzuführen, damit Ihre Applikation zur Betrachtung von Szenen interaktiv zu bedienen ist. Benutzen Sie alle Literatur, die Sie finden können. Um Ihre Algorithmen zu testen, sollen Sie den Utah Teapot und die schon vorhandenen Szenen in Ihrem interaktiven Betrachter verwenden. Eine einfache Schattierung kann durch den Cosinus zwischen Betrachtungsrichtung und Oberflächennormale erfolgen.

4.1 Bounding Volume Hierarchy (BVH)

Zusätzlich zur reinen Oberflächenbeschreibung wird ein Baum erstellt, dessen Knoten Bregrenzungsvolumen sind. Der Baum ist so definiert, dass die Begrenzungsvolumen aller Söhne im Begrenzungsvolumen des Vaters enthalten sind. In den Blättern befinden sich dann Listen von oder einzelne Objekte. Ein Strahl wird zuerst mit dem Begrenzungsvolumen der Wurzel verschnitten. Gibt es eine nicht leere Schnittmenge, so werden die Söhne rekursiv getestet. Auf diese Art und Weise wird ein Großteil der Objekte gar nicht erst mit dem Strahl verschnitten und der Strahlschnitt erheblich beschleunigt.

Aufgabe 11. Da allgemeine Begrenzungsvolumen nicht sehr effizient handhabbar sind, beschränken wir uns bei der Implementierung auf achsparallele Begrenzungsquader. Es sind prinzipiell zwei Routinen zu erstellen: Die eine baut den Baum auf und die andere traversiert ihn, um einen Strahl mit der Szene effizient zu verschneiden. Den Aufbau kann man *top-down* oder *bottom-up* gestalten. Bei einem Aufbau von oben wird die im aktuellen Knoten enthaltene Menge von Elementen partitioniert und die entstehenden disjunkten Teilmengen als Söhne an den aktuellen Knoten angehängt. Das Verfahren setzt man rekursiv fort, bis eine Unterteilung nicht mehr möglich ist oder nicht mehr lohnt.

Es bietet sich an, einen Binärbaum aufzubauen. Experimentieren mit Unterteilungskriterien, indem Sie versuchen den aktuellen Begrenzungsquader jeweils in der Mitte der längsten Kante zu teilen oder dafür zu sorgen, dass in beiden Söhnen etwa gleichviele Elemente verbleiben.

Machen Sie sich auch gedanken über die Reihenfolge der Traversierung. Also, ob zuerst der linke oder der rechte Teilbaum weiter traversiert werden soll. Kriterien dafür wäre z.B. die Strahlrichtung oder welche Boundingbox zuerst getroffen wird.

4.2 Binary Space Partition (BSP)

Die Binary Space Partition teilt den Raum rekursiv durch Ebenen. Jeder innere Knoten des Baums repräsentiert so eine Teilungsebene und in den Blättern findet man alle Objekte, die das durch die Ebenen bestimmte Voxel schneiden. Im Gegensatz zur Bounding Volume Hierarchy überschneiden sich die Voxel in einer Binary Space Partition nicht. Die Traversierung des Binärbaums besteht nun daraus zu entscheiden, welcher Halbraum längs des Strahls zuerst und welcher überhaupt traversiert werden muss. In den Blättern müssen dann alle Objekte gegen den Strahl getestet werden. Befindet sich der nächste gefundene Treffer in dem Voxel des Blattes, so hat man den nächsten Treffer gefunden.

Aufgabe 12. Es sind die zwei Routinen zum Aufbau und der Traversierung des Baumes zu entwerfen. Um den Algorithmus möglichst effizient zu gestalten, werden wieder nur Ebenen senkrecht zu den Koordinatenachsen verwendet. Ausgehend von dem achsparallelen Begrenzungsquader wird dieser entlang seiner längsten Seite in der Mitte geteilt und die Objekte demensprechend in die beiden Hälften einsortiert. Diese Prozedur wird rekursiv wiederholt, bis eine Teilung aufgrund der Tiefe des Baums oder der Anzahl der verbleibenden Elemente keinen Sinn mehr macht.

Die Traversierung ist rekursiv recht einfach erklärt: In jedem Knoten entscheidet man, welcher Sohn entlang des Strahls zuerst traversiert werden muss und ob der andere danach noch traversiert werden muss. Eine effiziente Implementierung arbeitet aber iterativ. In den Blättern werden alle Dreiecke dann auf den nächsten Schnitt getestet. Nehmen Sie Literatur und das Internet zu Hilfe, um entsprechende Hinweise und Tricks für eine effiziente Implementierung zu finden.

4.3 Quellen

Hier ein paar Quellen zu BVH und BSP (kd). Achtung(!): Zum Teil geht das, was in den gegebenen Quellen beschrieben wird, weit über das hinaus, was in den Aufgaben oben gefordert wird! Die erste Quelle deckt eigentlich schon alles ab, was man wissen muss um eine BVH und eine BSP zu implementieren.

- Online Kapitel - Peter Shirley: Fundamentals of Computer Graphics, AK Peters.
  http://www.cs.utah.edu/ shirley/books/fcg2/rt.pdf

- Ingo Wald Doktorarbeit
  http://www.sci.utah.edu/ wald/Publications/2004///WaldPhD/download//phd.pdf
- Ingo Wald, Solomon Boulos, and Peter Shirley: Ray Tracing Deformable Scenes using Dynamic Bounding Volume Hierarchies, 2006
  http://www.sci.utah.edu/ wald/Publications/2007///BVH/download//togbvh.pdf
- Carsten Wächter, Alexander Keller: Instant Ray Tracing: The Bounding Interval Hierarchy, 2006
  http://graphics.uni-ulm.de/BIH.pdf

4.4 Analyse und Vergleich

Es gibt viele Ansätze den Strahlschnitt zu beschleunigen. Welcher Ansatz besser ist, hängt oft von den zu bearbeitenden Szenen ab. Es gibt aber prinzipielle Unterschiede zwischen den Algorithmen und wie immer können durch die Fließkommaarithmetik numerische Probleme auftreten.

Aufgabe 13. Erörtern Sie:

1. Die Algorithmen sollen numerisch stabil sein. Wie können Ihre Algorithmen mit den numerischen Testfällen aus dem vorherigen Aufgabenteil umgehen?
2. Wie unterscheiden sich Regular Grids, Bounding Volume Hierarchy und Binary Space Partition prinzipiell und bezüglich der Komplexität aller Phasen des Algorithmus und des Speicherverbrauchs?

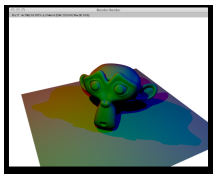
5 Schattieren Teil 1 (2 Wochen Bearbeitungszeit)

Bisher haben wir uns nur mit Geometrie und der effizienten Lösung des Sichtbarkeitsproblems befaßt. Jetzt soll die Geometrie auch schattiert werden. Das geschieht durch den Aufruf einer sogenannten *Shader*-Routine, die für jeden verfolgten Strahl eine Farbe als Ergebnis einer Stichprobe berechnet. Welcher *Shader* aufgerufen wird, ist meistens durch eine Referenz am gerade getroffenen Objekt vereinbart. Versuchen Sie Ihren Code so einfach und klar wie möglich zu halten.

5.1 Normalen und Material Export

Das Exportskript soll so erweitert werden, dass auch Normalen und Materialdaten exportiert werden. Hilfe zu Python in Blender finden Sie hier:
http://www.blender.org/documentation/244PythonDoc/index.html

5.2 Schatten



Neben der Anfrage den nächsten Treffer entlang eines Strahls zu finden, ist die Frage, ob irgendein Objekt den Strahl auf einem festgelegten Intervall schneidet, die wichtigste. Sie läßt sich effizienter implementieren und erlaubt es festzustellen, ob ein Punkt von einer Punktlichtquelle beleuchtet wird oder nicht.

Aufgabe 14. Fügen Sie Ihren Routinen zum Strahlverfolgen die Sichtbarkeitsfunktion V hinzu. Diese soll entscheiden, ob ein Strahl auf dem Intervall $(0, f)$ ein Objekt schneidet oder nicht.

1. Entwickeln Sie dann einen *Shader*, der eine Liste von Punktlichtquellen einliest und diese benutzt, um einen Trefferpunkt zu schattieren. Die Farbe

$$C(x) = \sum_{i=0}^{N-1} V(x, y_i) \frac{\cos^+ \theta_x \cos^+ \theta_{y_i}}{|x - y_i|^2} L_i$$

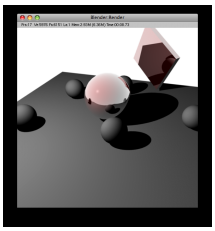
   ist die Summe der Beiträge aller N Punktlichtquellen, wobei y_i die Position und L_i die Radianz der i-ten Punktlichtquelle ist. $\cos^+$ bedeutet, dass der Cosinus nur genommen wird, wenn er positiv ist, ansonsten ist $\cos^+ = 0$. Die Winkel sind jeweils zwischen den entsprechenden Oberflächenormalen und der Strahlrichtung zu nehmen. Um den Algorithmus zu testen, kann es sinnvoll sein nur einzelne Lichtquellen anzuschalten. Die Lichtquellen sind in den neuen Testszenen Enthalten und müssen mit einem angepassten Pythonskript ebenfalls exportiert werden.

2. Welche numerischen Probleme gehen mit Sekundärstrahlen einher? Welche Artefakte kann man in keinem Fall verhindern und welche kann man wie abschwächen? Benutzen Sie die Testszene `3colorlight.blend` mit den entsprechenden Lichtquellen.

3. Verbessern Sie die Qualität der Oberflächennormalen, indem Sie diese aus den Normalen in den Eckpunkten der Dreiecke interpolieren. Zum eigentlichen Schnitttest muss natürlich immer noch die original geometrische Normale verwendet werden! Tipp: Verwenden Sie Blender zum Berechnen der Normalen und expoertieren Sie diese.

4. Was ist Ambient Occlusion? Entwickeln Sie einen Shader dafür.

5.3 Einfacher Pathtracer



Viele Dreiecke werden mit der gleichen Methode schattiert, aber eben nicht alle. Daher referenziert man pro Dreieck einen Datensatz mit der Schattierungsinformation. Dieser Datensatz kann von allen auf die gleiche Art schattierten Dreiecken geteilt werden.

Aufgabe 15. Lesen Sie An Improved Illumination Model for Shaded Display. Verwenden Sie die Testszene `whitted_scene.blend` und schattieren Sie die Trefferpunkte mit der im Paper vorgestellten Methode. Tipp: Die Szene enthällt nur spiegelnde und diffuse Oberflächen.

5.4 Verbesserte Materialien

Oberlächen sind nicht nur spiegelnd oder diffus. Um reale Oberflächeneigenschaften zu simulieren wird ein Beleuchtungsmodell verwendet.

Aufgabe 16. Entwickeln Sie eine Routine `Shade`, die einen gefundenen Trefferpunkt schattiert. Dazu sollen die im Dreieck dereferenzierten Daten mit dem Beleuchtungsmodell von Ashikhmin und Shirley beleuchtet werden. Wie in der vorherigen Aufgabe sollen Strahlen rekursiv weiterverfolgt werden. Verwenden Sie die Testszenen der vorhergehenden Aufgaben und die `conference.blend` Testszene.