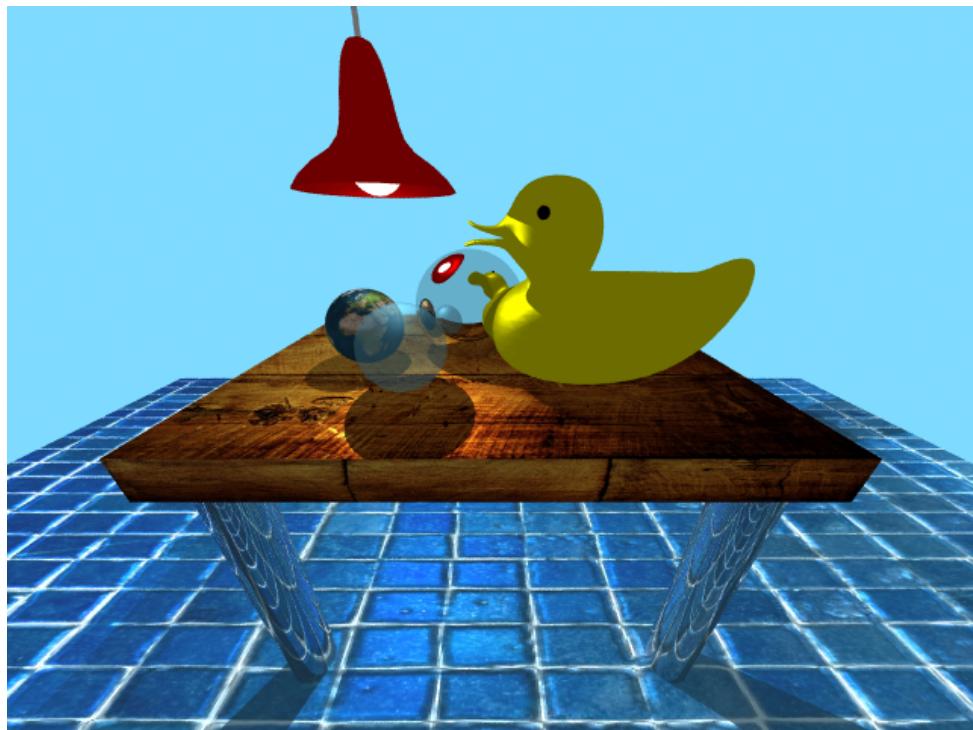


Implementierung eines einfachen Raytracers



Praktikum WS 2011/12

Philipp Ruchti

25. Januar 2012

Inhaltsverzeichnis

1 Einleitung	3
2 Raytracing	3
3 Transformationen	3
3.1 Homogene Notation	3
3.2 Platzierung von Objekten und der Kamera	3
4 Strahlen - Objekt Schnittpunkt	4
4.1 Implizite Oberflächen: hier Kugeln und Zylinder	4
4.2 Parametrische Oberflächen: hier Drei- und Vierecke	4
4.3 AABBs	4
5 Phong Shading und Phong Beleuchtungsmodell	4
5.1 Phong	4
5.2 Lichtquellen und Schatten	5
5.3 Texturen	5
6 Sampling	5
6.1 Random Sampling	6
6.2 Stratified Sampling	6
6.3 Poisson Sampling	6
6.4 Halton Sampling	6
6.5 Ergebnisse der verschiedenen Sampling Methoden	6
7 Rekonstruktion	7
7.1 Box Rekonstruktion	7
7.2 Mitchell Rekonstruktion	7
8 Vergleich verschiedener Sampling- und Rekonstruktionstechniken	7
9 Beschleunigung der Schnittpunktberechnung	9
9.1 kd-Baum-Hierarchie	9
10 Zusätzliche Funktionalität	10
10.1 Reflektionen und einfache Transparenz	10
10.2 Bump-Mapping	11
11 Ergebnisse	11
12 Quellen	13
12.1 C++Mathe-Bibliothek	13
12.2 Weiterer Code	13
12.3 Texturen und Modelle	13

1 Einleitung

Im Verlaufe dieses Praktikums wurde ein einfacher Raytracer implementiert, welcher die grundlegenden Funktionalitäten bietet eine 3D-Szene abzubilden. Im folgenden werden die dafür verwendeten Techniken kurz erläutert und einige Ergebnisbilder gezeigt. Es wird hierbei ein besonder Augenmerk auf die in der Implementierung umgesetzten Methoden und Verfahren gelegt.

Im zweiten Kapitel wird einleitend diskutiert was einen Raytracer ausmacht. Danach werden in Kapitel 3 benötigte Transformationen und eine geeignete Notation eingeführt. Mit Hilfe dieser Transformationen lassen sich Objekte in einem einheitlichem Koordinatensystem plazieren. In Kapitel 4 werden die mathematischen Vorgehen besprochen, welche benötigt werden um Strahlen mit Objekten zu schneiden. Im folgenden fünften Kapitel wird die Lichtberechnung beschrieben, welche an den ermittelten Strahl-Objekt Schnittpunkten, zur Farbbestimmung, ausgewertet wird. In den Kapiteln 6 und 7 werden Sampling und Rekonstruktion besprochen um auch feine Strukturen darstellen zu können. Diese Methoden werden anschließend in Kapitel 8 evaluiert. In Kapitel 9 wird eine Datenstruktur zur Beschleunigung der Schnittpunktberechnung besprochen. In Kapitel 10 werden Reflektionen, einfache Transparenz und Bump-Mapping vorgestellt, Methoden welche ein Bild realistischer erscheinen lassen. Im letzten Kapitel sollen einige Ergebnisbilder gezeigt und beschrieben werden.

2 Raytracing

Bei der Erstellung eines Bildes mit Hilfe von Rasterisierung, wie man es beispielsweise von OpenGL kennt werden die Vertices einer 3D-Szene mit Hilfe einer Projektionsmatrix auf eine Bildfläche projiziert und mit Hilfe dieser die Farbe eines Bildpunktes bestimmt. Beim Raytracing werden im Gegensatz hierzu Strahlen von der Kamera durch die Bildfläche geschickt und diese mit der 3D-Szene geschnitten. Diese Technik bietet die Möglichkeit auch Reflektionen, Schatten und viele andere Lichteffekte darzustellen. Im Gegensatz zur ersten Technik bietet Raytracing die Möglichkeit realistischere Bilder einer 3D-Szene zu berechnen ist jedoch rechenaufwendiger.

3 Transformationen

In diesem Kapitel wird zuerst die homogene Notation beschrieben. Anschließend werden die für den Raytracer verwendeten Transformationen erklärt welche benötigt werden um Objekte und Kamera in die korrekte Position zu bringen.

3.1 Homogene Notation

Um 3D-Positionen und Verschiebungen zu repräsentieren eignet sich die sogenannte homogene Notation. Hierzu werden 3D-Vektoren um eine vierte Komponente w erweitert. Ein homogener Punkt (x, y, z, w) repräsentiert hierbei den euklidischen Punkt $(\frac{x}{w}, \frac{y}{w}, \frac{z}{w})$. Diese vierte Komponente w beschreibt eine Skalierung des Vektors. Punkte haben als 4. Komponente einen Wert $w \neq 0$. Vektoren welche eine 4. Komponente $w = 0$ aufweisen sind Richtungen, welchen eine unendliche Länge aufweisen und sich nicht verschieben lassen. Mit dieser Notation ist es möglich mit Hilfe von 4x4-Transformatrizen Punkte und Vektoren zu rotieren sowie Punkte zu verschieben. Eine Matrix \mathbf{M} die einen Punkt mit einer Rotationsmatrix \mathbf{R} rotiert und um (x, y, z) verschiebt sieht homogen wie folgt aus:

$$\mathbf{M} = \begin{pmatrix} & & x \\ & \mathbf{R} & y \\ 0 & 0 & z \\ 0 & 0 & 1 \end{pmatrix}$$

3.2 Platzierung von Objekten und der Kamera

Um Objekte auf ein Bild abzubilden müssen sich alle Objekte und die Kamera in einem einheitlichem Koordinatensystem befinden. Hierzu werden zwei Arten von Transformationen benötigt, die erste ist die Model-Transformation, die zweite die View-Transformation. Die Model-Transformation transformiert die Modelle oder Objekte vom Objekt eigenem ins globale Koordinatensystem, sie verschiebt die Objekte an ihre Stelle in der Welt. Die zweite Transformation bringt alle Objekte in das Koordinatensystem der Kamera, so dass diese an der richtigen Stelle gesehen werden. Um Objekte und Kamera platzieren und

verschieben zu können verwendet man homogene Matrizen. Hierbei erhält man das gleiche Ergebnis wenn man im zweiten Schritt die Kamera in einer Richtung bewegt, oder alle Objekte in die entgegengesetzte Richtung bewegt, da für die Sicht in die Szene lediglich die relative Position von Kamera und Objekten eine Rolle spielt. Um die Verwendung von AABBs zu vereinfachen wurde in dieser Implementierung die Kamera verschoben, beziehungsweise wurden die Strahlen nach Berechnung aus dem Ursprung mit Hilfe einer Transformationsmatrix in kanonische Richtung verschoben und rotiert.

4 Strahlen - Objekt Schnittpunkt

Um Objekte in einem Raytracer anzeigen zu können müssen Schnittpunkte von Strahlen (engl. *rays*) mit den unterschiedlichen Objekten berechnet werden. Hierbei ist in der Regel lediglich der erste Schnittpunkt auf einem Strahl von Interesse. Hierbei sind die Verfahren um diese Schnittpunkte zu ermitteln je nach Objekt unterschiedlich.

4.1 Implizite Oberflächen: hier Kugeln und Zylinder

Implizite Oberflächen werden durch mathematische Gleichungen $f(x, y, z)$ beschrieben, welche für alle Punkt die auf der Oberfläche liegen $f(x, y, z) = 0$ ergeben. Für Kugeln benötigt man hierzu lediglich den Mittelpunkt der Kugel, sowie einen Radius. Die Punkte $\{x, y, z\}$ welche die Gleichung

$$f(x, y, z) = (x - u)^2 + (y - v)^2 + (z - w)^2 - r^2 = 0$$

erfüllen bilden die Oberfläche einer Kugel mit Radius r und Mittelpunkt $\{u, v, w\}$. In dieser Arbeit wurden des weiteren noch Zylinder implementiert, bei welchen die Oberfläche ebenfalls auf eine solche Weise beschrieben werden kann.

4.2 Parametrische Oberflächen: hier Drei- und Vierecke

Parametrische Oberflächen sind Oberflächen für welche zur Berechnung der Oberflächenpunkte ein lineares Gleichungssystem gelöst werden muss. Für Dreiecke liegen die Punkte eines Strahles $o + t \cdot d$ welche die Gleichung

$$o + t \cdot d = (1 - b_1 - b_2)p_0 + b_1 * p_1 + b_2 * p_2$$

unter der Bedingung $b_1 \geq 0$ und $b_2 \geq 0$ und $b_1 + b_2 \leq 1$ erfüllen auf der Oberfläche dieses. Hierbei ist o der Startpunkt und d die Richtung des Strahls. Für Vierecke ändert sich lediglich der letzte Teil der Bedingung zu $b_1 < 0$ und $b_2 < 0$.

4.3 AABBs

Zur Beschleunigung der Schnittpunktberechnung kann um komplexere Objekte ein Boundingvolume gelegt werden. Die Kollisionsberechnung mit diesem ist hierbei einfach und schnell zu berechnen und so müssen Strahlen, die das Boundingvolume nicht treffen nicht mit jedem Objekt des Inhaltes getestet werden. In der hier vorliegenden Implementierung wurden Axis-Aligned-Boundingboxes implementiert. Zur Kollisionsprüfung werden so genannte *slabs* berechnet, Intervalle entlang einer Achse. Mit Hilfe dieser *slabs* lässt sich feststellen ob die Linie das Boundingvolume schneidet.

5 Phong Shading und Phong Beleuchtungsmodell

text hier

5.1 Phong

Trifft ein Strahl ein Objekt so wird an diesem zur Berechnung der „gesehenen“ Objektfarbe sowohl die Farbe des Objektes beachtet als auch die Beleuchtung mit Hilfe der verschiedenen Lichtquellen und deren Farben berechnet, dies übernimmt das Beleuchtungsmodell. Hierbei entsteht das Licht aus drei verschiedenen Komponenten. Die erste Komponente ist hierbei das ambiente Umgebungslicht. Die Zweite ist das diffuse Richtungslicht. Die Dritte Komponente ist das spekulare Glanzlicht. Für die Berechnung des hier verwendeten Beleuchtungsmodells wird neben dem Kollisionspunkt auch die Normale N des Punktes benötigt,

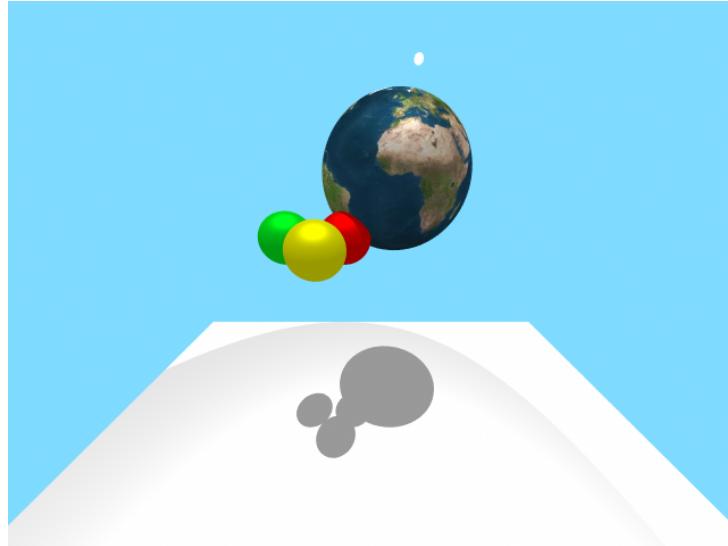
dieses wird beim Phong Beleuchtungsmodell über das Objekt interpoliert. Zur Berechnung der Farbe werden sowohl die entsprechenden Komponenten der Farbe des Objektes $\text{Farbe}_{\{\text{ambient}, \text{diffuse}, \text{specular}\}}$ verwendet als auch die entsprechenden Farbkomponenten $E_{\{\text{ambient}, \text{diffuse}, \text{specular}\}}$ der Lichtquelle. Wobei die [Wort für tiefgestelltes Zeugs] die jeweiligen Farbkomponenten des Lichtes beschreiben. Die Beleuchtung an einer Stelle wird nach folgenden Formel berechnet:

$$\text{Farbe} = \text{Farbe}_{\text{ambient}} \otimes E_{\text{ambient}} + \text{Farbe}_{\text{diffuse}} \otimes E_{\text{diffuse}} \cdot (N \cdot L) + \text{Farbe}_{\text{specular}} \otimes E_{\text{specular}} \cdot (R \cdot V)^m$$

Hierbei ist N die Normale am Objekt, L die Richtung in welcher die Lichtquelle liegt R die Reflektionstrichtung des Strahles sowie V die Sichtrichtung auf das Objekt, also der Richtungsvektor des Strahles. \otimes ist die komponentenweise Multiplikation der einzelnen Vektorwerte der Farben. m beeinflusst die Größe der Glanzlichtanteile. Hat die Szene mehr als eine Lichtquelle so entsteht das resultierende Licht als Durchschnitt aller Farbbeiträge der einzelnen Lichtquellen.

5.2 Lichtquellen und Schatten

Erweitert man obiges Modell um eine Prüfung ob die entsprechenden Lichtquellen gesehen werden, so kann man leicht Schatten erzeugen. Hierzu wird bei einer Punktlichtquelle ein Strahl vom Punkt des Objektes, dessen Farbe soeben bestimmt wird, zur Lichtquelle geschickt und überprüft ob dieser auf dem Weg zum Licht ein Objekt trifft, ist dies der Fall so wird das Licht nicht gesehen. Handelt es sich um eine gerichtete Beleuchtung so darf kein Objekt in diese Richtung liegen, egal in welchem positivem Abstand.



Im obigen Bild sind neben einer texturierten Kugel insbesondere Schatten zu sehen. Diese entstehen dadurch dass die Kugeln das Licht der weißen Lichtquelle (über der Erdkugel) sowie das eines gerichteten Lichtes von schräg oben rechts nicht auf die weiße Fläche durchlassen. So entstehen durch das Ausbleiben von Licht zwei unterschiedlich dunkle Schattenregionen.

5.3 Texturen

Statt an einem Kollisionspunkt eines Strahles mit einem Objekt eine Objekt globale Farbe zurückzugeben kann diese auch aus einer Textur gelesen werden. Bei Dreiecken ist dies am einfachsten, da man hier die Baryzentrische Koordinaten bereits bei der Kollisionsberechnung ermittelt und diese direkt zum Zugriff auf ein Bild verwenden kann. Bei Kugeln bedarf es einer aufwendigeren Berechnung des Punktes auf der Kugel in Koordinaten des Bildes.

6 Sampling

Durch wenige oder gar nur einen Strahl welche durch die Bildpixel gesendet werden kann es bei kleinen Objekten oder feinen Strukturen dazu kommen, dass diese nicht oder Falsch dargestellt werden. Diesen Effekt kann man dadurch umgehen, dass man mehr Strahlen durch ein Bildpixel sendet und aus diesen danach einen Farbwert rekonstruiert. Um den Effekt des Aliasing, bei welchem bei zu geringer Abtastrate,

Zahl der Strahlen, statt einer feinen Struktur Muster entstehen welche nicht vorhanden sind, zu reduzieren ist es zudem ratsam die Strahlen nicht geordnet anzulegen. Werden jedoch viele Farbwert für ein Pixel berechnet und lediglich gemittelt, so wird das Bild unscharf. Für diesen hier erstellten Raytracer wurden verschiedene Samplingmethoden zum Vergleich implementiert.

6.1 Random Sampling

Beim Random Sampling werden durch ein Pixel mehrere Strahlen gesendet, die Position dieser ist hierbei zufällig und kann so dazu führen, dass eine unregelmäßige Abdeckung verschiedener Bereiche des Pixels erfolgt.

6.2 Stratified Sampling

Beim Stratified Sampling wird das Pixel in mehrere regelmäßige *Strata* eingeteilt, in jedem dieser Subregionen wird nun zufällig eine Position für einen Strahl ermittelt. Im Gegensatz zum Random Sampling verbessert sich die Abdeckung des Pixels, da so keines der *Strata* nicht beachtet wird.

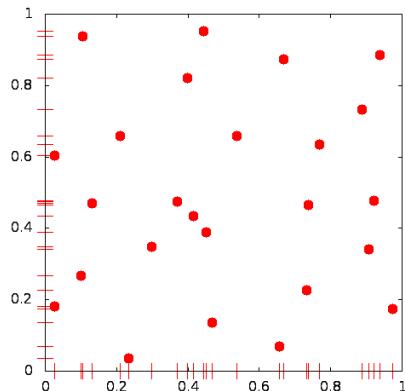
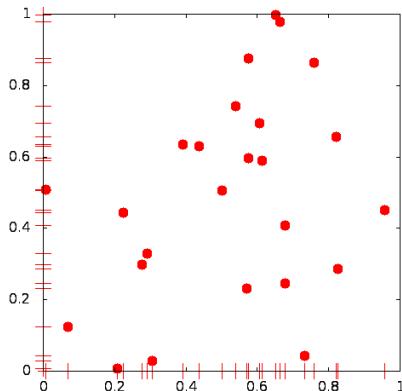
6.3 Poisson Sampling

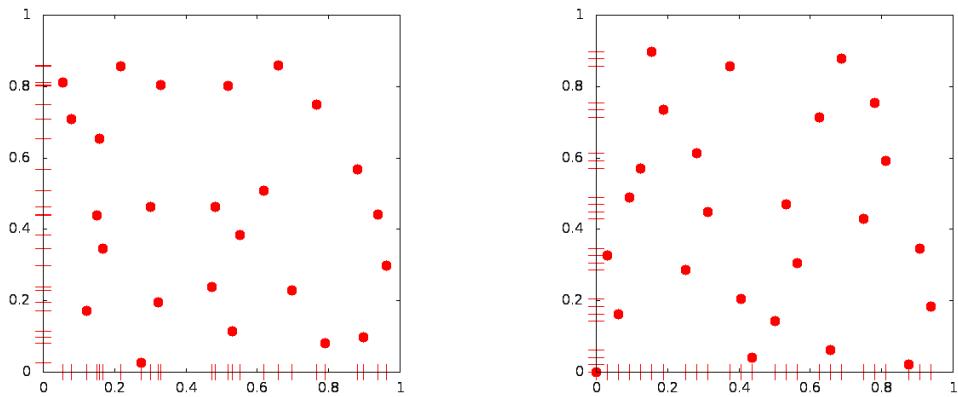
Für den hier implementierten Raytracer wurde das Verfahren des Poisson Sampling gewählt. Hierbei werden zufällig über das Pixel verteilt Orte gewählt, durch welche Strahlen gesandt werden sollen. Jedoch werden nur solche Positionen verwendet, die zu ihren Nachbarn einen gewissen Mindestabstand aufweisen. So wird die Häufung der Strahlen an einer Stelle vermieden.

6.4 Halton Sampling

Das ausgeklügelteste Sampling-Verfahren welches hier implementiert wurde ist das sogenannte Halton Sampling. Bei diesem Verfahren wird die Missverteilung des Positionen für die Strahlen minimiert. Hierzu werden sowohl die X- als auch die Y-Komponenten der Position des Strahles anhand einer Hammerley Sequenz gewählt. Eine Hammerley Sequenz liefert Werte welche zwischen 0 und 1 liegen und diesen Bereich zu jedem Zeitpunkt möglichst gleichverteilt abgedecken. Mit diesem Verfahren wird gewährleistet das der Bereich des Pixels möglichst gleichmäßig abgedeckt wird.

6.5 Ergebnisse der verschiedenen Sampling Methoden





Die von den verschiedenen Verfahren berechneten Positionen für Strahlen innerhalb eines Pixels. Von oben links nach unten rechts: Random Sampling, Stratified Sampling, Poisson Sampling, Halton Sampling (mit $p_1 = 2, p_2 = 7$). Zusätzlich wurden die Positionen auch auf die X- und Y-Achse projiziert, so das einfacher ersichtlich ist ob das Pixel so in beiden Dimensionen ausreichend und gleichverteilt abgedeckt wird.

7 Rekonstruktion

Um aus den Farbwerten, welche die verschiedenen Strahlen die für den Bereich eines Pixels berechnen wurden einen konsistenten Farbwert zu wählen bedarf es einer Rekonstruktionstechnik. Diese gewichtet die Farbwerte anhand eines Kernels $f(d)$ und normalisiert am Schluss das Ergebnis.

7.1 Box Rekonstruktion

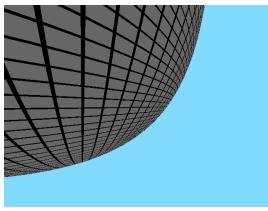
Ein einfaches Rekonstruktionsverfahren ist die Box-Rekonstruktion. Hierbei werden alle Farbwerte gleich gewichtet ($f(d) = 1$) und durch die Anzahl der Farbwerte geteilt, es wird also ein Mittelwert über alle vorhandenen Farbwerte der Strahlen gebildet.

7.2 Mitchell Rekonstruktion

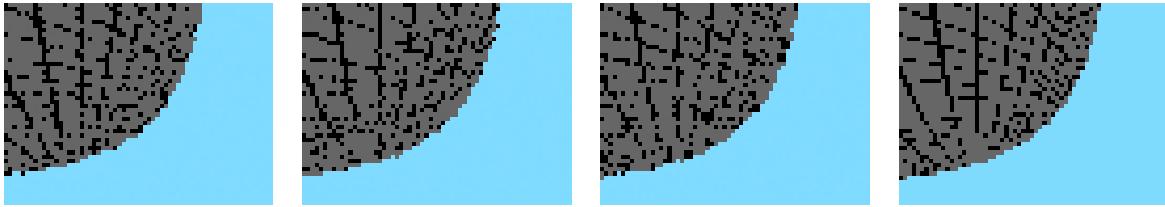
Die Mitchell Rekonstruktion gewichtet die Farbwerte mit einer Standart-Normalverteilung ihres Abstandes zum Pixelzentrum. Somit werden Farbwerte die nah an der Pixelmitte liegen stärker beachtet.

8 Vergleich verschiedener Sampling- und Rekonstruktionstechniken

Bild mit 640×480 Pixeln:



Bilder mit je einem Strahl pro Pixel und einer Auflösung von 64×48 :

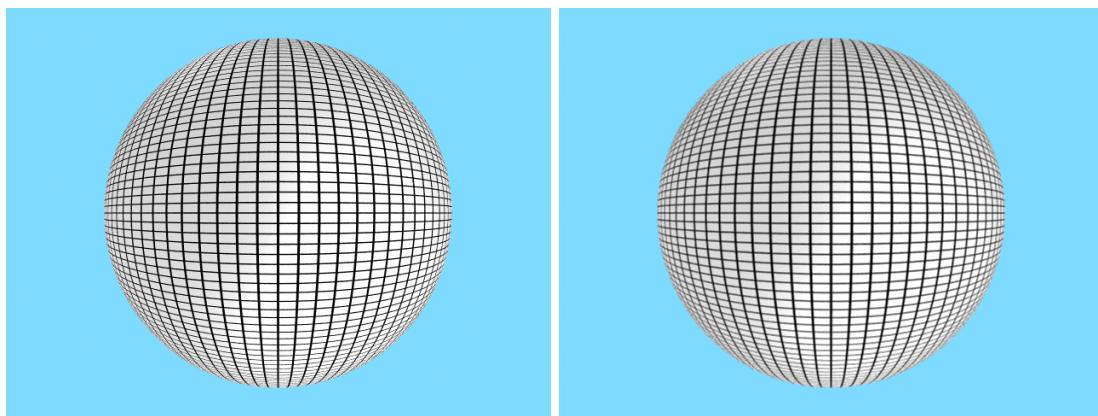
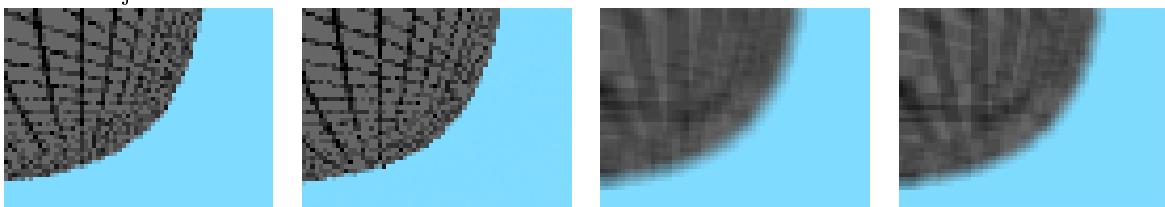


Bilder mit je 4 Strahlen pro Pixel und einer Auflösung von 64×48 :



Zu sehen sind hier von links nach rechts: Random, Stratified, Poisson und Halton Sampling jeweils mit Mitchell Rekonstruktion. Es ist gut zu erkennen das das Auftreten und Ausbleiben der Linien beim Random Sampling mit einem Strahl pro Pixel sehr zufällig ist und diese beim Halton Sampling bereits konsistenter Auftreten oder Ausbleiben. Mit mehreren Strahlen pro Pixel gehen bei allen Verfahren weniger der Linien verloren, diese verschwimmen jedoch auch je kleiner die Struktur wird. Mit nur einem Strahl pro Pixel erzeugen das Random sowie das Poisson Sampling noch die meisten der fließenden Linien. Mit 4 Strahlen können das Stratified und das Halton Sampling am besten mit diesen Linien umgehen.

Bilder mit je v.....:



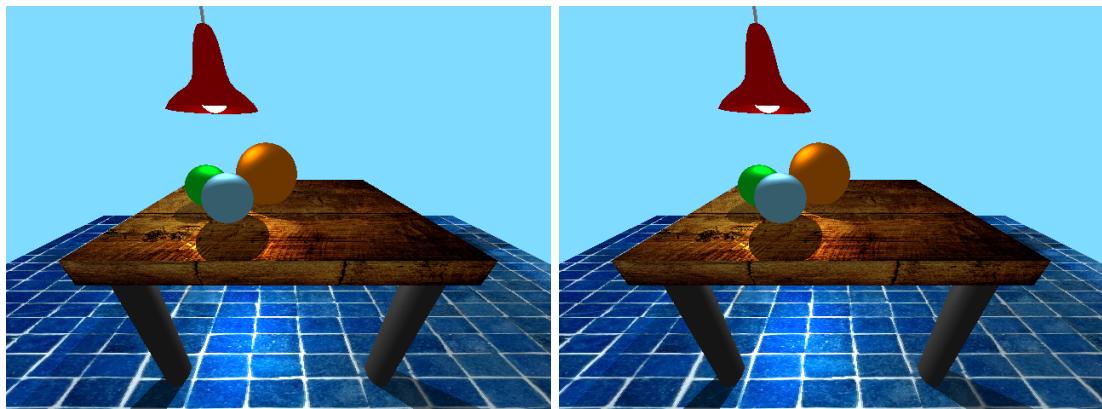
In diesen Zwei Bildern soll der Effekt des Aliasing aufgezeigt werden. So entstehen im linken Bild an den Polen und Seiten der Kugel Strukturen welche nicht erwünscht sind. Im rechten Bild wurden zusätzlich berechnete Farbwerte von Positionen der angrenzender acht Pixel bei der Berechnung der Farbe des zentralen Pixels beachtet. So verschwimmen die Strukturen leicht und der Effekt des Aliasing wird reduziert.

9 Beschleunigung der Schnittpunktberechnung

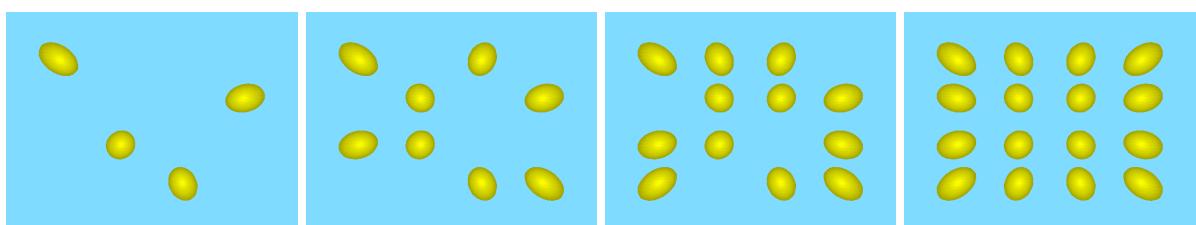
Zur Beschleunigung der Schnittpunktberechnung mit komplexeren Objekten eignen sich neben den oben genannten Boundingvolumes hierarchische Zerlegungen der zu zeichnenden Objekte oder des Zeichenraumes. In dieser Implementierung wurden die Objekte zerlegt. Hierzu wurde eine kd-Baum ähnliche Boundingvolumen-Hierarchie implementiert. Diese zerteilt rekursiv das zu zeichnende Objekt in je zwei Teil-Hierarchien.

9.1 kd-Baum-Hierarchie

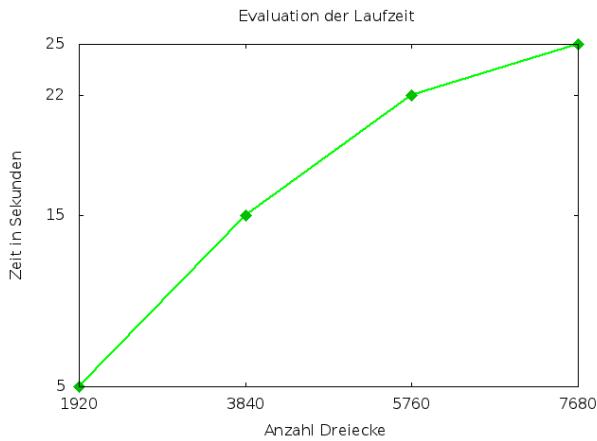
Ein kd-Baum zerlegt eine das komplette Objekt umschließende AABB mit Hilfe verschiedener Ebenen rekursiv in zwei Teil kd-Bäume. Schneidet der Strahl einen solchen Teilbaum nicht, so kann dieser von der Schnittpunktberechnung komplett ausgeschlossen werden und so die Berechnung erheblich beschleunigt werden. Der kd-Baum trennt hierbei in wechselnder Reihenfolge das (Teil-)Objekt in Teilräume bezüglich einer zu einer der Achsen orthogonalen Ebene. Zur einfacheren Traversierung der Datenstruktur wurden statt die Teilbäume mit Hilfe einer Ebene zu trennen AABBs für die zwei Teil-Hierarchien berechnet und verwendet. So teilt auch die hier implementierte kd-Baum-Hierarchie das Objekt in oben genannter Weise in je zwei Teil-Hierarchien. Für beide Teileobjekte werden nun AABBs berechnet die alle Primitive umschließen. Durch diesen Trick kommt es dazu dass sich die AABBs zum Teil überlagern, jedes Primitiv aber nur in einem der zwei Teil-Hierarchien gespeichert werden muss. Wenn ein Strahl nun mit der Hierarchie geschnitten werden soll, so wird zuerst geprüft ob der Strahl die alles umschließende AABB schneidet, Wenn ja wird rekursiv jeweils geprüft mit welcher der zwei Teil-Hierarchien der Strahl schneidet. So können schnell große Teile des Objektes verworfen werden. Danach werden nur die Um die Korrektheit und Effizienz zu überprüfen wurden folgende zwei Bilder gerendert. Die Lampe besteht hierbei aus [x] Dreiecken. Das linke Bild wurde hierbei ohne die Datenstruktur erstellt und benötigte 212 Sekunden zur Erstellung, das rechte Bild benötigte mit Datenstruktur lediglich 10 Sekunden.



Des Weiteren wurde getestet wie die Laufzeit bei Verwendung von zusätzlichen Primitiven skaliert. So wurden verschiedene Instanzen des selben Objektes in ein und die selbe Datenstruktur integriert.



Hierbei benötigten die Bilder in der Erstellung von links nach rechts: 6, 53 bzw 419 Sekunden.

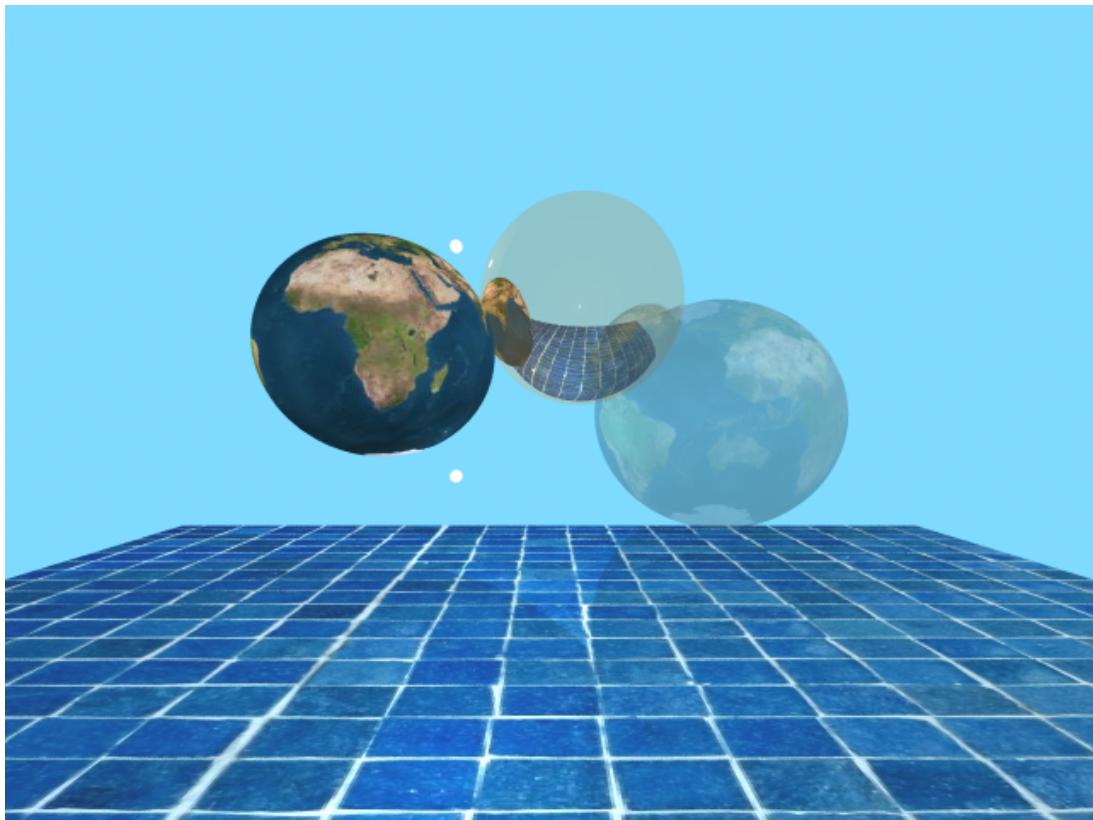


Trägt man diese Zeiten in ein Diagramm ein so muss ein anähernd quadratisches Verhalten in der Anzahl der Primitiven vermutet werden. Dies ist der Fall da TEXT scheiß Implementation.

10 Zusätzliche Funktionalität

10.1 Reflektionen und einfache Transparenz

Versendet man an einem Kollisionspunkt eines Strahls mit der Szene rekursiv zwei weitere Strahlen so kann man einfache Transparenz und Reflektionen erzeugen. Hierzu wird ein Strahl in Reflektionsrichtung und ein Strahl durch das Objekt gesendet. Je nach Gewichtung der drei entstehenden Farben entsteht der Anschein von Reflektionen oder Transparenz.

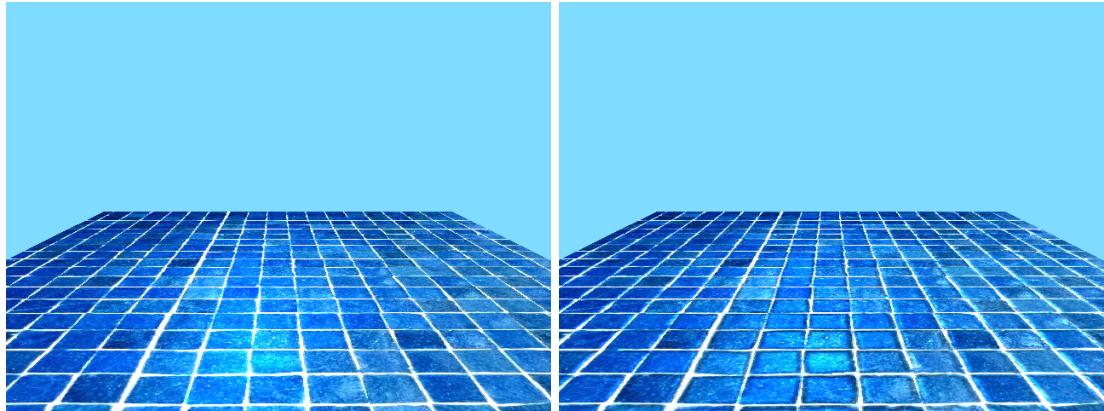


Dieses Bild soll den Effekt von Reflektionen und Transparenz aufzeigen. So ist die hintere Kugel eigentlich orange, da sie jedoch zu 80% reflektiert, bildet sie die Umgebung ab. Der Boden reflektiert zu ebenfalls

leicht und lässt so die Kugeln erahnen. In der reflektierenden Kugel ist auch zu erkennen, dass die zwei Erdkugeln auf Grund ihrer vorhanden oder nicht vorhandenen Transparenz unterschiedlich stark zu erkennen sind.

10.2 Bump-Mapping

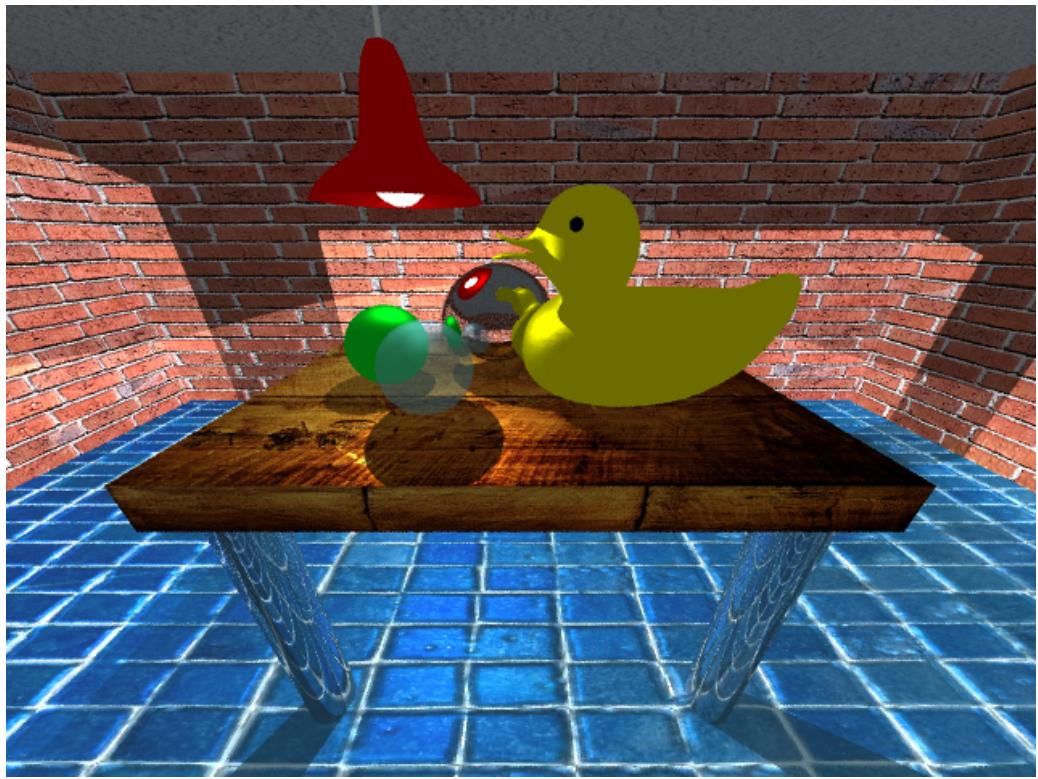
Ähnlich wie bei der Verwendung von Texturen zur Bestimmung der Farbe können Grauwertbilder auch zur Modifikation der Objektnormale verwendet werden. Durch die neuen Normale werden Licht- und Schatteneffekte erzeugt die den Eindruck von Struktur auf einer glatten Fläche vermitteln. Links ein Bild ohne eine Bump-Map, rechts mit:



11 Ergebnisse

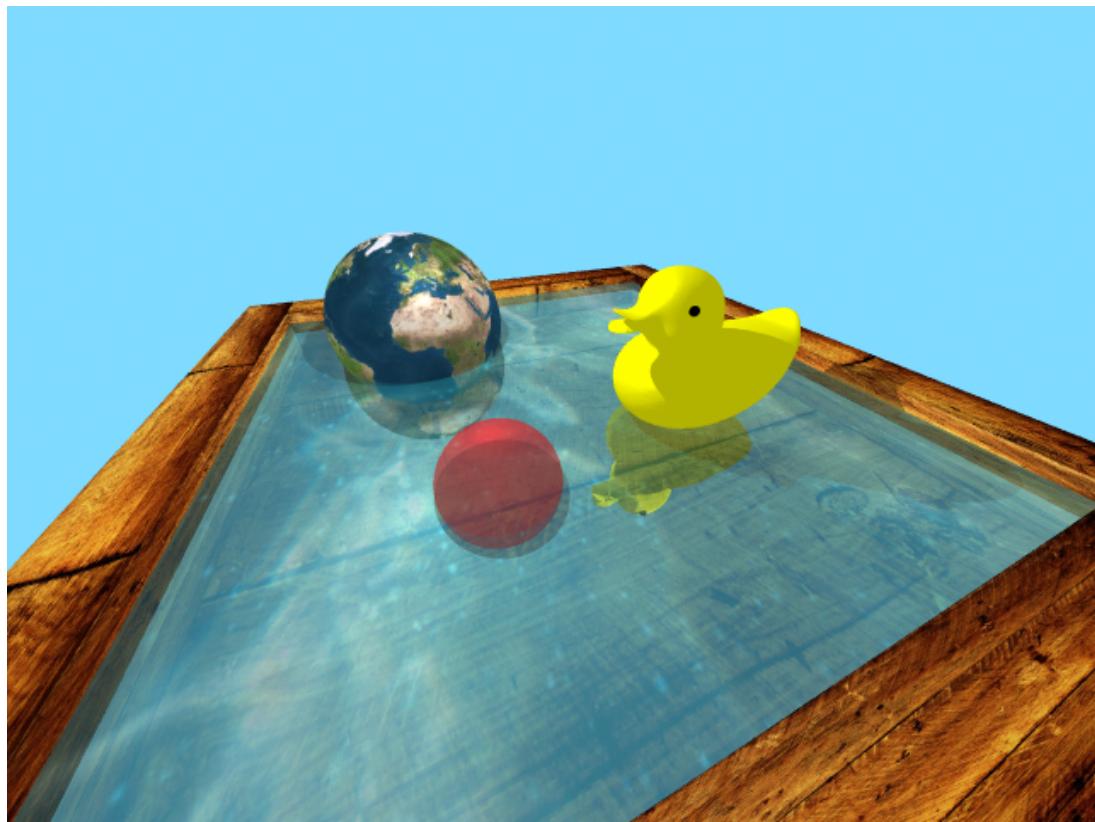
Ein Eindruck des implementierten Programmes:





Die obere Szene besteht aus zwei triangulierten Dreiecken, der Ente mit 10432 und der Lampe mit 840 Dreiecken, je einer reflektierenden, transparenten sowie einer normalen Kugel, texturierten Flächen sowie reflektierenden Zylindern. In der Szene sind vor allem an den Wänden Schatten von zwei Lichtquellen, einer in der Lampe sowie einer unter dem Tisch zu erkennen. Der Boden erhält seine Struktur mit Hilfe von Bump-Mapping.

Im unteren Bild sind ähnliche Objekte zu erkennen, wobei das Wasser unter Anwendung einer transparenten, reflektierenden, texturierten Fläche entstanden ist.



12 Quellen

12.1 C++Mathe-Bibliothek

Für die hier verwendete Mathematik wurde die in dieser Implementierung die Mathebibliothek des Lehrstuhles für Mustererkennung und Bildverarbeitung von Prof. Brox verwendet.

Copyright: Prof. Brox (<http://lmb.informatik.uni-freiburg.de>)

12.2 Weiterer Code

Jeglicher weiterer Code wurde persönlich angefertigt. Qt und C++ wurden verwendet.

12.3 Texturen und Modelle

Modelle:

<http://www.oyonale.com/modeles.php?lang=en&page=53>

Texturen:

http://fc03.deviantart.com/fs26/i/2008/042/e/d/Local_Texture__Three_by_One_by_Beyond_Oddities.jpg
http://www.seos-project.eu/modules/landuse/images/2_earth_2400_960.jpg
http://free-images-etc.rb-d.com/wp-content/uploads/IMG_9203.jpg
http://upload.wikimedia.org/wikipedia/commons/8/8e/Solna_Brick_wall_Stretcher_bond_variation1.jpg
http://4.bp.blogspot.com/-TNdnCVil1zE/TdbRmB02L4I/AAAAAAAAGE/ZGBehf57dQ0/s1600/Water_Texture_by_Wisdoms_Pearl07.jpg
<http://www.malertv.de/wp-content/uploads/2009/12/DSCF9223.jpg>