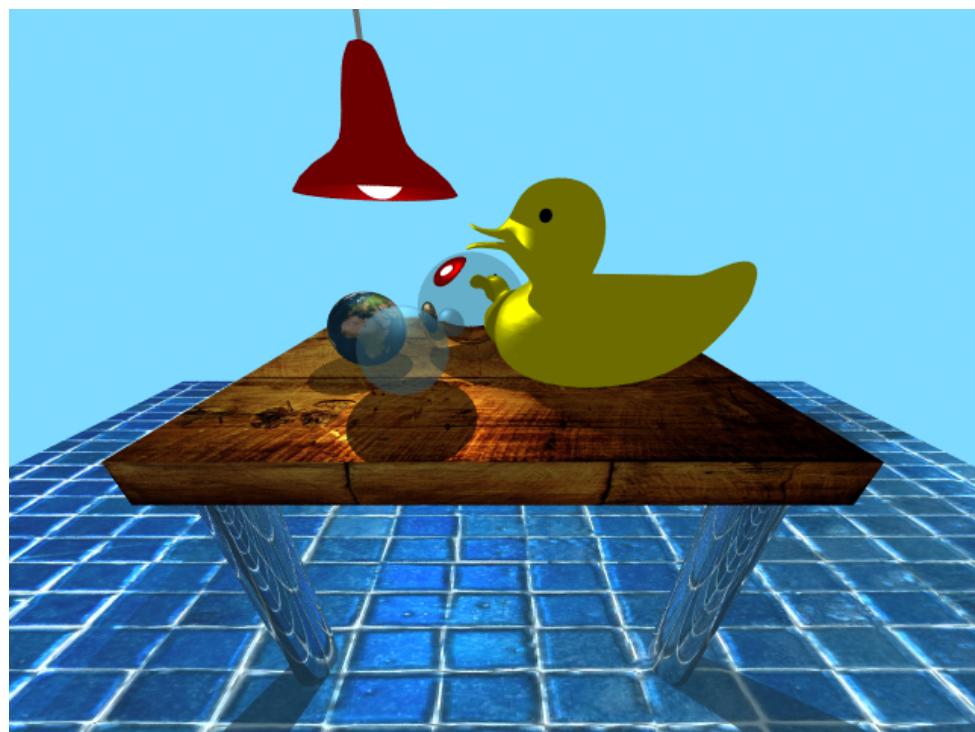


# Implementierung eines Raytracers



Praktikum WS 2011/12  
Philipp Ruchti

# Inhaltsverzeichnis

<b>0 Einleitung</b>	<b>1</b>
<b>1 Raytracer</b>	<b>1</b>
<b>2 Transformationen</b>	<b>1</b>
2.1 Homogene Notation . . . . .	1
2.2 Platzierung von Objekten und der Kamera . . . . .	1
<b>3 Strahlen-Objekt Schnittpunkte</b>	<b>2</b>
3.1 Implizite Oberflächen: Hier Kugeln und Zylinder . . . . .	2
3.2 Parametrische Oberflächen: Hier Drei- und Vierecke . . . . .	2
3.3 AABBs . . . . .	2
<b>4 Phong Shading und Phong Beleuchtungsmodell</b>	<b>2</b>
4.1 Phong . . . . .	2
4.2 Lichtquellen und Schatten . . . . .	3
4.3 Texturen . . . . .	3
<b>5 Sampling</b>	<b>3</b>
5.1 Random Sampling . . . . .	4
5.2 Stratified Sampling . . . . .	4
5.3 Poisson Sampling . . . . .	4
5.4 Halton Sampling . . . . .	4
5.5 Ergebnisse der verschiedenen Sampling Methoden . . . . .	5
<b>6 Rekonstruktion</b>	<b>5</b>
6.1 Box Rekonstruktion . . . . .	5
6.2 Mitchell Rekonstruktion . . . . .	5
<b>7 Vergleich von Sampling- und Rekonstruktionstechniken</b>	<b>6</b>
<b>8 Beschleunigung der Schnittpunktberechnung</b>	<b>7</b>
8.1 K-d-Baum-Hierarchie . . . . .	7
<b>9 Zusätzliche Funktionalität</b>	<b>8</b>
9.1 Reflektionen und einfache Transparenz . . . . .	8
9.2 Bump-Mapping . . . . .	9
<b>10 Ergebnisse</b>	<b>10</b>
<b>11 Quellen</b>	<b>11</b>
11.1 C++Mathe-Bibliothek . . . . .	11
11.2 Weiterer Code . . . . .	11
11.3 Texturen und Modelle . . . . .	11

# 0 Einleitung

Im Verlaufe dieses Praktikums wurde ein 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 besonderes Augenmerk auf die in der Implementierung umgesetzten Methoden und Verfahren gelegt.

Im ersten Kapitel werden einleitend die Besonderheiten eines Raytracers diskutiert. Danach werden in Kapitel 2 benötigte Transformationen und eine geeignete Notation eingeführt. Mit Hilfe dieser Transformationen lassen sich Objekte in einem einheitlichen Koordinatensystem platzieren. In Kapitel 3 werden die mathematischen Vorgehen besprochen, welche benötigt werden um Strahlen (engl. *rays*) mit Objekten zu schneiden. Im folgenden vierten Kapitel wird die Lichtberechnung beschrieben, welche an den ermittelten Strahl-Objekt Schnittpunkten zur Farbbestimmung ausgewertet wird. In den Kapiteln 5 und 6 werden Sampling und Rekonstruktion besprochen um auch feine Strukturen darstellen zu können. Diese Methoden werden anschließend in Kapitel 7 evaluiert. In Kapitel 8 wird eine Datenstruktur zur Beschleunigung der Schnittpunktberechnung besprochen. In Kapitel 9 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.

## 1 Raytracer

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 erzeugen, ist jedoch rechenaufwendiger.

## 2 Transformationen

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

### 2.1 Homogene Notation

Um 3D-Positionen und Richtungsvektoren zu repräsentieren und zu unterscheiden eignet sich die sogenannte homogene Notation. Hierzu werden 3D-Vektoren um eine vierte Komponente  $w$  erweitert. Richtungsvektoren erhalten hierbei eine 4. Komponente  $w = 0$ , Positionen erhalten als 4. Komponente  $w \neq 0$ . Diese vierte Komponente  $w$  beschreibt eine Skalierung des Vektors. Ein homogener Punkt  $(x, y, z, w)$  repräsentiert hierbei den euklidischen Punkt  $(\frac{x}{w}, \frac{y}{w}, \frac{z}{w})$ . Ist  $w = 0$  so handelt es sich um einen Richtungsvektor, welcher eine unendliche Länge aufweist und sich nicht verschieben lässt. Mit Hilfe dieser Notation ist es möglich mit 4x4-Transformatrizien Positionen und Richtungsvektoren zu rotieren sowie Positionen 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:

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

### 2.2 Platzierung von Objekten und der Kamera

Um Objekte auf ein Bild abzubilden müssen sich alle Objekte und die Kamera in einem einheitlichen 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 Objekte vom objekteigenen ins globale Koordinatensystem, sie verschiebt diese 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. Man erhält das selbe Ergebnis, wenn man im zweiten Schritt die Kamera in einer Richtung 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 (siehe weiter unten) zu vereinfachen, wurde in dieser Implementierung die Kamera verschoben. Hierzu wurden die Strahlen nach Berechnung aus dem Ursprung mit Hilfe einer Transformationsmatrix in kanonische Richtung verschoben und rotiert.

## 3 Strahlen-Objekt Schnittpunkte

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

### 3.1 Implizite Oberflächen: Hier Kugeln und Zylinder

Implizite Oberflächen werden durch mathematische Funktionen  $f(x, y, z)$  beschrieben, welche für alle Punkte, die auf der Oberfläche liegen,  $f(x, y, z) = 0$  ergeben. Für Kugeln benötigt man hierzu lediglich den Mittelpunkt der Kugel, sowie den 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 Zylinder als implizite Oberflächen implementiert. Auch hier kann die Oberfläche auf eine solche Weise beschrieben werden.

### 3.2 Parametrische Oberflächen: Hier Drei- und Vierecke

Zur Berechnung der Punkte von parametrischen Oberflächen muss ein lineares Gleichungssystem gelöst werden. 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 den Bedingungen  $b_1 \geq 0$ ,  $b_2 \geq 0$  und  $b_1 + b_2 \leq 1$  erfüllen, auf der Oberfläche dieser. Hierbei ist  $o$  der Startpunkt und  $d$  die Richtung des Strahls. Für Vierecke ändert sich lediglich die letzte Bedingung zu  $b_1 < 1$  und  $b_2 < 1$ .

### 3.3 AABBs

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

## 4 Phong Shading und Phong Beleuchtungsmodell

In diesem Kapitel wird erklärt, wie an einem ermittelten Strahl-Objekt Schnittpunkt die Farbe bestimmt werden kann. Dazu wird zuerst das Phong Beleuchtungsmodell erklärt. Anschließend wird beschrieben wie Schatten entstehen und wie die Farbe eines Objektes anhand einer Textur bestimmt werden kann.

### 4.1 Phong

Trifft ein Strahl ein Objekt, so wird an diesem Punkt zur Berechnung der „gesehenen“ Farbe sowohl die Objektfarbe beachtet, als auch die Beleuchtung mit Hilfe der verschiedenen Lichtquellen und deren Farben berechnet. Diese Berechnung 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 ist das spiegelnde Glanzlicht. Für die Berechnung des hier verwendeten Beleuchtungsmodells wird neben dem Kollisionspunkt auch die Normale  $N$  des Punktes benötigt. Zur Berechnung der Farbe werden sowohl die entsprechenden Komponenten der Farbe des Objektes Farbe<sub>Lichtart</sub>, als auch die entsprechenden Farbkomponenten  $E_{\text{Lichtart}}$  der Lichtquelle verwendet. Die Beleuchtung an einem Punkt wird nach folgender Formel berechnet:

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

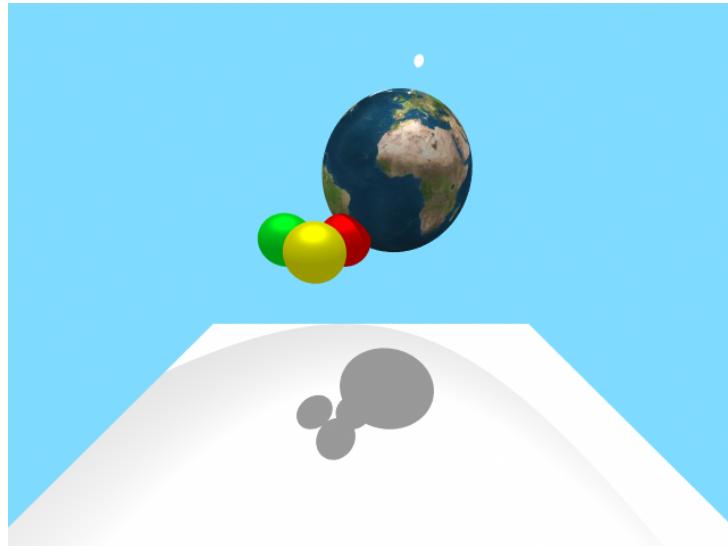
Hierbei ist  $N$  die Normale am Objekt,  $L$  die Richtung, in welcher die Lichtquelle liegt,  $R$  die Reflektionrichtung des Strahles sowie  $V$  die Sichtrichtung auf das Objekt.  $\otimes$  ist in diesem Fall die komponentenweise Multiplikation der einzelnen Vektorwerte.  $m$  beeinflusst die Größe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

## 4.2 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 dieser Richtung liegen.

## 4.3 Texturen

Statt an einem Kollisionspunkt eines Strahles mit einem Objekt eine objektglobale Farbe zurückzugeben, kann diese auch aus einer Textur gelesen werden. Bei Dreiecken ist dies am einfachsten, da man hier die baryzentrischen Koordinaten bereits bei der Kollisionsberechnung ermittelt und diese direkt zum Zugriff auf ein Bild verwenden kann. Bei Kugeln bedarf es einer aufwendigeren Umrechnung des Punktes auf der Kugel in Koordinaten des Bildes. Hierbei werden Längen- und Breitengrad des Kollisionspunktes ermittelt und mit Hilfe dieser auf die Textur zugegriffen<sup>1</sup>.



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 Licht einer gerichteten Lichtquelle von schräg oben rechts verdecken. So entstehen durch das Ausbleiben von Licht zwei unterschiedlich dunkle Schattenregionen auf der weißen Ebene.

## 5 Sampling

Wird durch die Region eines Pixels nur ein Strahl gesendet, so kann es dazu kommen, dass kleine Objekte oder feine Strukturen falsch oder nicht dargestellt werden. Diesen Effekt kann man dadurch umgehen,

---

<sup>1</sup>Siehe: <http://www.cs.unc.edu/~rademach/xroads-RT/RTarticle.html>

dass man mehrere Strahlen durch die Region eines Pixels sendet und aus diesen einen Farbwert für dieses Pixel rekonstruiert. Um den Effekt des Aliasings zu reduzieren, bei welchem bei zu geringer Abtastrate bzw. Zahl der Strahlen pro Pixelregion statt einer feinen Struktur unerwünschte Muster entstehen, ist es zudem ratsam die Strahlen nicht geordnet anzulegen. Für diesen hier implementierten Raytracer wurden verschiedene Samplingmethoden implementiert.

## 5.1 Random Sampling

Die Position der Strahlen beim Random Sampling wird in der Pixelregion zufällig gewählt. Dies kann dazu führen, dass eine unregelmäßige Abdeckung verschiedener Bereiche des Pixels erfolgt.

## 5.2 Stratified Sampling

Beim Stratified Sampling wird die Pixelregion 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 der Pixelregion, da so keines der *Strata* ohne Sample bleiben kann.

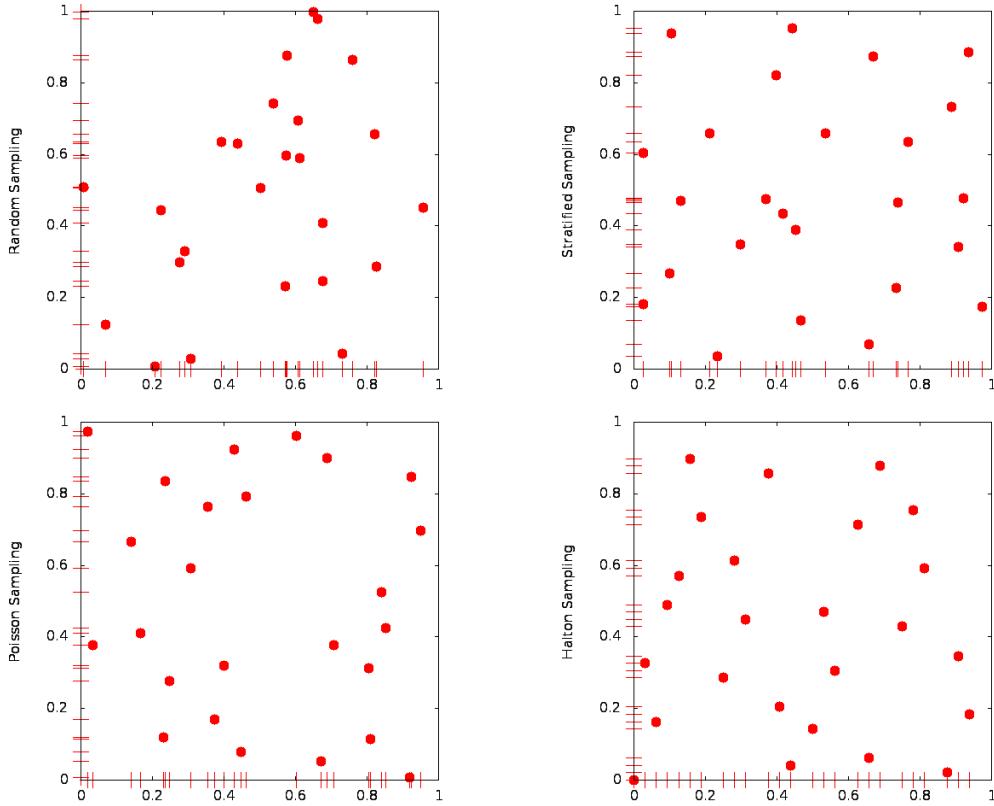
## 5.3 Poisson Sampling

Beim Poisson Sampling werden zufällig über die Pixelregion verteilt Orte gewählt, durch welche Strahlen gesandt werden sollen. Es werden jedoch nur solche Positionen verwendet, die zu ihren Nachbarn einen gewissen Mindestabstand aufweisen. So wird die Häufung der Strahlen an einer Stelle reduziert.

## 5.4 Halton Sampling

Das ausgeklügeltste Samplingverfahren, welches in dieser Arbeit implementiert wurde, ist das sogenannte Halton Sampling. Bei diesem Verfahren wird die Fehlverteilung der Positionen für die Strahlen minimiert. Hierzu werden sowohl die x- als auch die y-Komponenten der Position des Strahles anhand einer Hammersley Sequenz gewählt. Eine Hammersley Sequenz liefert Werte, welche das Intervall zwischen 0 und 1 zu jedem Zeitpunkt möglichst gleichverteilt abdecken. Mit diesem Verfahren wird gewährleistet, dass der Bereich des Pixels möglichst gleichmäßig abgedeckt wird.

## 5.5 Ergebnisse der verschiedenen Sampling Methoden



In den obigen Bildern sind die von den verschiedenen Verfahren berechneten Positionen für Strahlen innerhalb eines Pixels dargestellt. 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 dass einfacher ersichtlich ist, ob die Region des Pixels in beiden Dimensionen ausreichend und gleichverteilt abgedeckt wird. Es ist hierbei sofort ersichtlich, dass beim Random Sampling die Region nicht gleichmäßig abgedeckt wird. Bei Stratified und Poisson Sampling wird die Region sehr viel gleichmäßiger abgedeckt. Im letzten Bild ist auf den Achsen die Hammersley Sequenz gut zu erkennen.

## 6 Rekonstruktion

Um aus den Farbwerten, welche die verschiedenen Strahlen im Bereich um die Pixelmitte berechnet haben, einen konsistenten Farbwert zu wählen, bedarf es einer Rekonstruktionstechnik. Diese Rekonstruktionstechnik gewichtet die Farbwerte anhand eines Kernels  $f(d)$  und summiert diese auf. Hierbei ist  $d$  der Abstand der Sampleposition zur Pixelmitte. Nachfolgend werden zwei verschiedene Samplingverfahren vorgestellt.

### 6.1 Box Rekonstruktion

Ein einfaches Rekonstruktionsverfahren ist die Box Rekonstruktion. Diese gewichtet alle Farbwerte gleich:  $f(d) = \frac{1}{N}$ . Hierbei ist  $N$  die Anzahl der Samples, welche für dieses Pixel beachtet werden. Es wird also ein Mittelwert über alle vorhandenen Farbwerte gebildet.

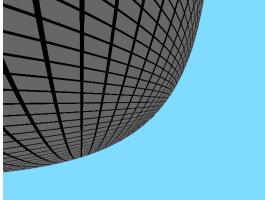
### 6.2 Mitchell Rekonstruktion

Die Mitchell Rekonstruktion gewichtet die Farbwerte mit einer Standard-Normalverteilung ihres Abstandes zum Pixelzentrum:  $f(d) = \frac{1}{N} \cdot \frac{1}{\sqrt{2\pi}} e^{-\frac{d^2}{2}}$ . Hierbei ist  $N$  wieder die Anzahl der Samples, welche beachtet werden und  $d$  der Abstand zur Pixelmitte. Durch dieses Verfahren werden Farbwerte, die nah an der Pixelmitte liegen, stärker gewichtet.

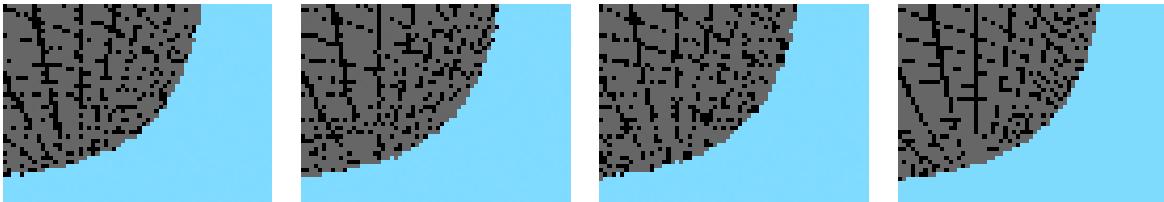
## 7 Vergleich von Sampling- und Rekonstruktionstechniken

In diesem Kapitel sollen die oben erwähnten Sampling- und Rekonstruktionsverfahren evaluiert werden. Hierzu wurde ein Bild mit normaler Auflösung erstellt. Dieses wird nun mit Bildern der selben Szene verglichen, welche mit einer sehr viel geringeren Auflösung erstellt wurden. Es wird hierbei geprüft wie gut diese Bilder das Vergleichsbild rekonstruieren.

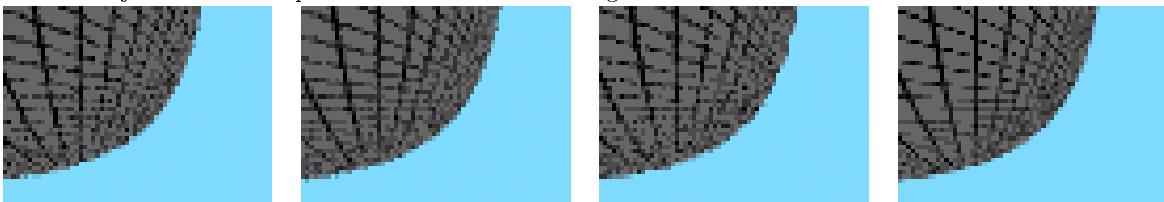
Bild mit normaler Auflösung ( $640 \times 480$  Pixel):



Bilder mit je einem Strahl pro Pixel und einer Auflösung von  $64 \times 48$  Pixeln:

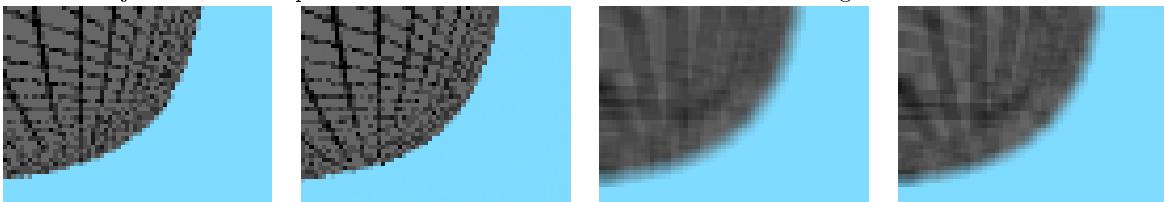


Bilder mit je vier Strahlen pro Pixel und einer Auflösung von  $64 \times 48$  Pixeln:

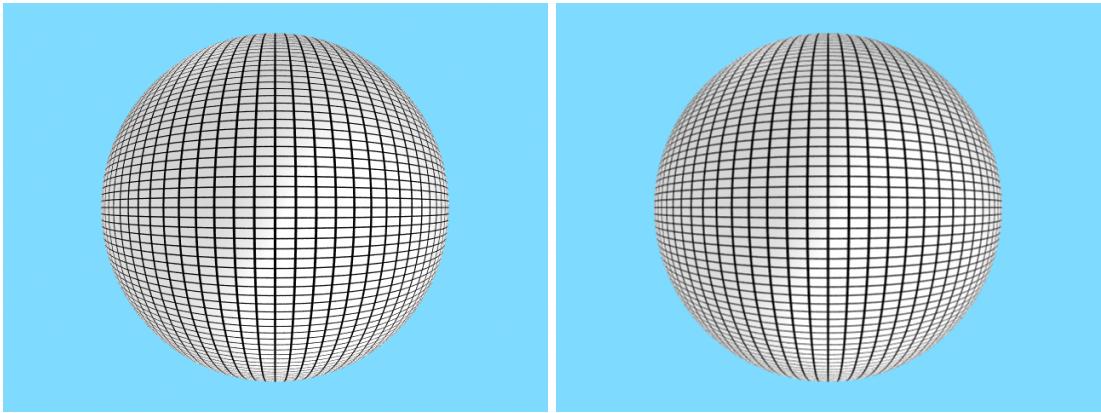


Zu sehen sind hier von links nach rechts: Random, Stratified, Poisson und Halton Sampling jeweils mit Mitchell Rekonstruktion über die Region des entsprechenden Pixels. In der ersten Reihe ist gut zu erkennen, dass das Auftreten und Ausbleiben der Linien beim Random Sampling mit einem Strahl pro Pixel sehr zufällig ist. Die Linien treten beim Halton Sampling bereits konsistenter auf. Mit mehreren Strahlen pro Pixel (untere Reihe) gehen bei allen Verfahren weniger der Linien verloren. Die Linien verschwimmen jedoch auch je kleiner die Struktur wird. Mit nur einem Strahl pro Pixel erzeugen das Poisson sowie das Halton Sampling noch die meisten der horizontalen Linien. Mit vier Strahlen können das Stratified und das Halton Sampling am besten mit diesen Linien umgehen.

Bilder mit je vier Strahlen pro Pixel und unterschiedlichen Rekonstruktionsregionen:



Mit den obigen Bildern soll der Unterschied zwischen der Rekonstruktion über verschiedenen großen Regionen pro Pixel aufgezeigt werden. Auch soll der Unterschied zwischen Box und Mitchell Rekonstruktion dargestellt werden. Die Bilder sind alle mit vier Strahlen pro Pixel und Random Sampling erstellt worden. Es sind von links nach rechts zu sehen: Box Rekonstruktion über ein Pixel rekonstruiert, selbiges mit Mitchell Rekonstruktion, Box Rekonstruktion über vier Pixel und selbiges mit Mitchell Rekonstruktion. Man erkennt gut, dass die Rekonstruktion über eine größere Region das Bild stärker verschwimmen lässt. Auch ist gut erkennbar, dass die Mitchell Rekonstruktion die Strukturen besser erhält, als dies mit der Box Rekonstruktion der Fall ist.



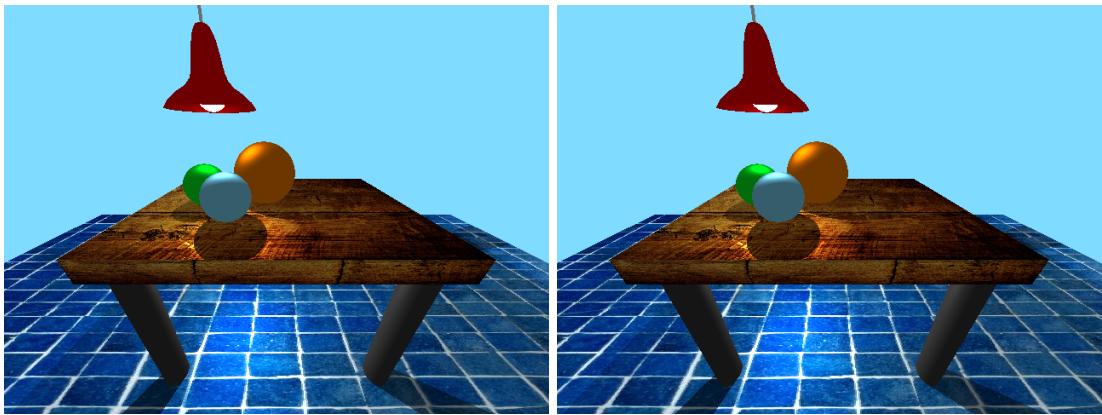
In obigen Bildern soll der Effekt des Aliasing und der Vorteil der Integration der Farbwerte angrenzender Pixel aufgezeigt werden. So entstehen im linken Bild, bei welchem nur Farbwerten der Region des eigenen Pixels verwendet wurden, an den Polen und Seiten der Kugel Strukturen, welche nicht erwünscht sind. Im rechten Bild wurden zusätzlich berechnete Farbwerte aus Regionen der angrenzenden acht Pixel bei der Berechnung der Farbe beachtet. So verschwimmen die Strukturen etwas und der Effekt des Aliasing wird reduziert.

## 8 Beschleunigung der Schnittpunktberechnung

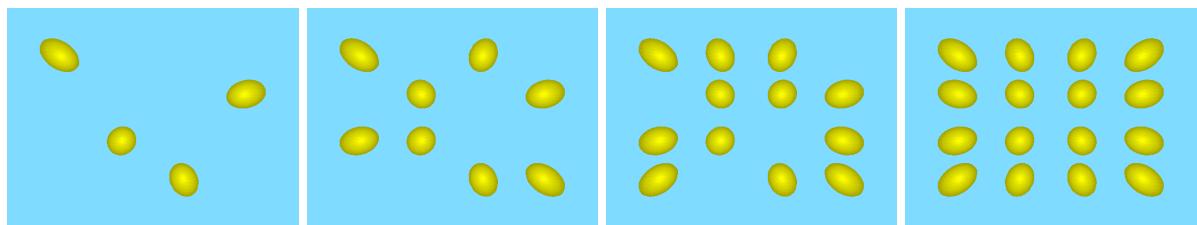
Zur Beschleunigung der Schnittpunktberechnung mit komplexeren Objekten eignen sich neben den oben genannten Boundingvolumen hierarchische Zerlegungen der zu zeichnenden Objekte oder des Zeichenraumes. Hierzu wurde eine k-d-Baum ähnliche Boundingvolumen-Hierarchie implementiert. Diese zerteilt rekursiv das zu zeichnende Objekt in je zwei Teilhierarchien.

### 8.1 K-d-Baum-Hierarchie

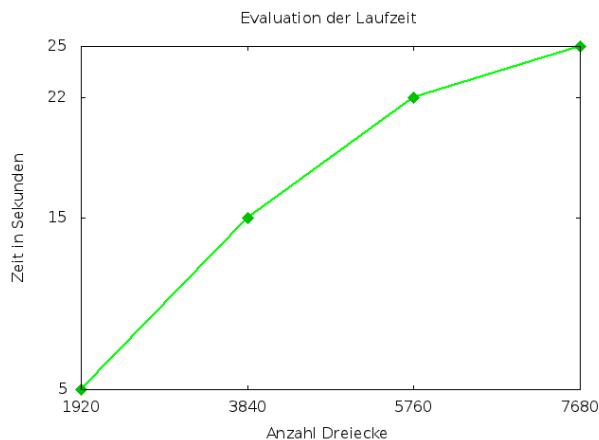
Ein k-d-Baum zerlegt eine, das komplette Objekt umschließende AABB, mit Hilfe verschiedener Ebenen rekursiv in zwei Teil-k-d-Bäume. Schneidet der Strahl einen solchen Teilbaum nicht, so kann dieser von der Schnittpunktberechnung komplett ausgeschlossen und so die Berechnung erheblich beschleunigt werden. Der k-d-Baum trennt hierbei in wechselnder Reihenfolge das (Teil-)Objekt in Teilräume bezüglich Ebenen, welche jeweils von zwei der Achsen aufgespannt werden. Zur einfacheren Traversierung der Datenstruktur wurden statt die Teilbäume mit Hilfe einer Ebene zu trennen AABBs für die zwei Teilhierarchien berechnet und verwendet. So teilt auch die hier implementierte k-d-Baum-Hierarchie das Objekt in oben genannter Weise in je zwei Teilhierarchien. Für beide Teileobjekte werden nun AABBs berechnet, die alle Objekte der Teilhierarchie umschließen. Durch diesen Trick kommt es dazu, dass sich die AABBs zum Teil überlappen, jedes Primitiv aber nur in einer der zwei Teilhierarchien 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 jeweils rekursiv geprüft welche der zwei Teilhierarchien der Strahl schneidet. So können schnell große Teile der Primitive ausgeschlossen werden. In den Blättern der Struktur werden schlussendlich die dort gespeicherten Primitive mit dem Strahl auf Kollision geprüft. Um die Korrektheit der hier implementierten Datenstruktur zu überprüfen, wurden folgende zwei Bilder erzeugt. Die Lampe besteht hierbei aus 840 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. Beide Bilder weisen optisch aber keinerlei Unterschiede auf.



Des Weiteren wurde getestet wie sich die Laufzeit bei Verwendung von zusätzlichen Primitiven verhält. So wurden verschieden viele Instanzen des selben Objektes (je 480 Dreiecke pro Objekt) in ein und die selbe Datenstruktur integriert.



Hierbei benötigten die Bilder in der Erstellung von links nach rechts: 5, 15, 22 bzw. 25 Sekunden. Beim Aufbau der Szenen wurde versucht die Objekte möglichst gleichmäßig über das Bild zu verteilen.



Trägt man diese Zeiten in ein Diagramm ein, so kann ein annähernd logarithmisches Verhalten in der Anzahl der Primitiven vermutet werden. Diese Beobachtung deckt sich mit theoretischen Überlegungen, da es sich bei der Datenstruktur insbesondere um einen Binärbaum handelt. Da die Datenstruktur die Objekte immer in der Mitte der Anzahl der Primitive teilt, handelt es sich hierbei um einen best möglic balancierten Baum.

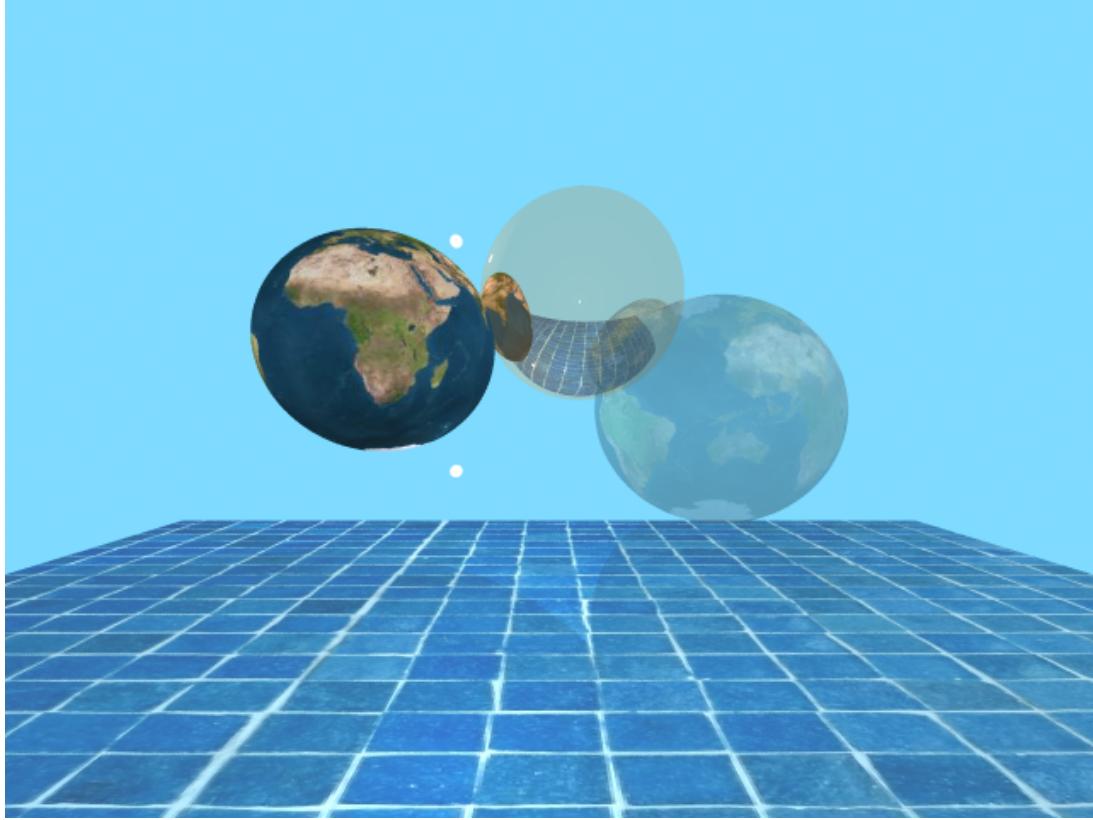
## 9 Zusätzliche Funktionalität

In diesem Kapitel werden drei einfache Methoden beschrieben um das optische Resultat der Bilder zu verbessern.

### 9.1 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

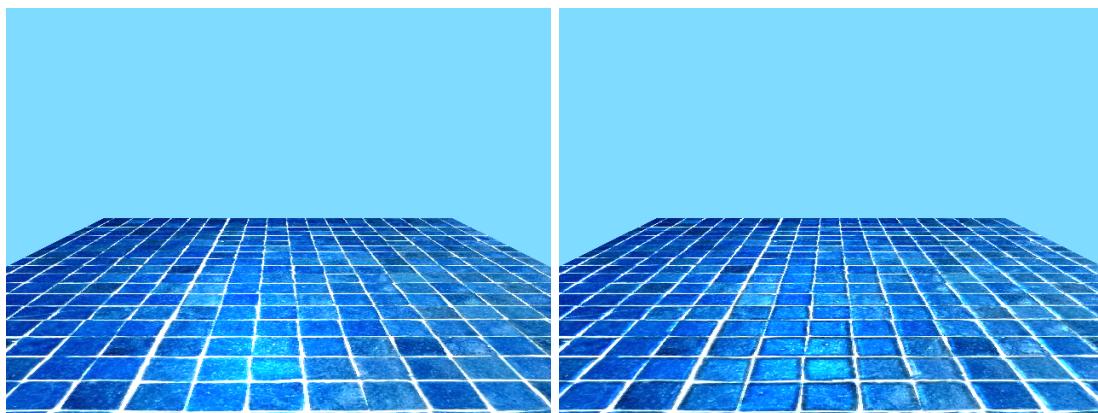
und ein Strahl durch das Objekt gesendet. Je nach Gewichtung der drei entstehenden Farben entsteht der Anschein von Reflexionen oder Transparenz.



Dieses Bild soll den Effekt von Reflexionen und Transparenz aufzeigen. So ist die mittlere Kugel eigentlich orange, da sie jedoch zu 80% reflektiert, bildet sie statt dessen die Umgebung ab. Der Boden reflektiert 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 abgebildet werden.

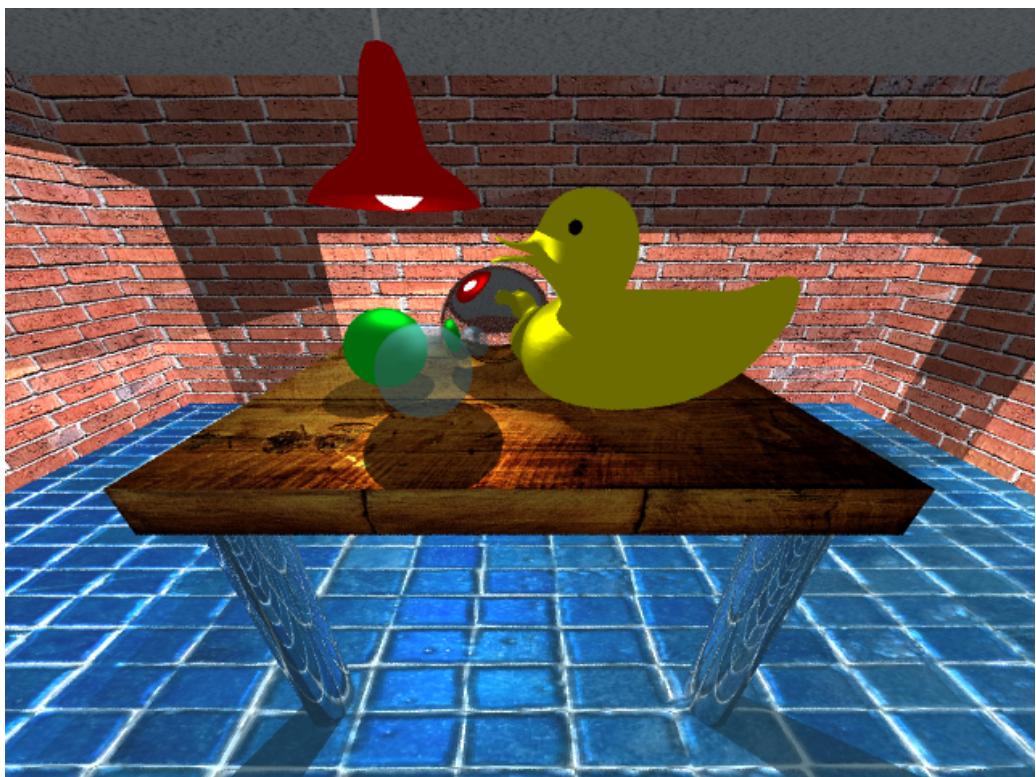
## 9.2 Bump-Mapping

Ähnlich der Verwendung von Texturen zur Bestimmung der Farbe von Objekten können Grauwertbilder 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 ist ein Bild einer planaren, texturierten Fläche ohne eine Bump-Map zu sehen, rechts das selbe Bild unter Verwendung einer Bump-Map. Man erkennt deutlich die Fugen des Bodens.



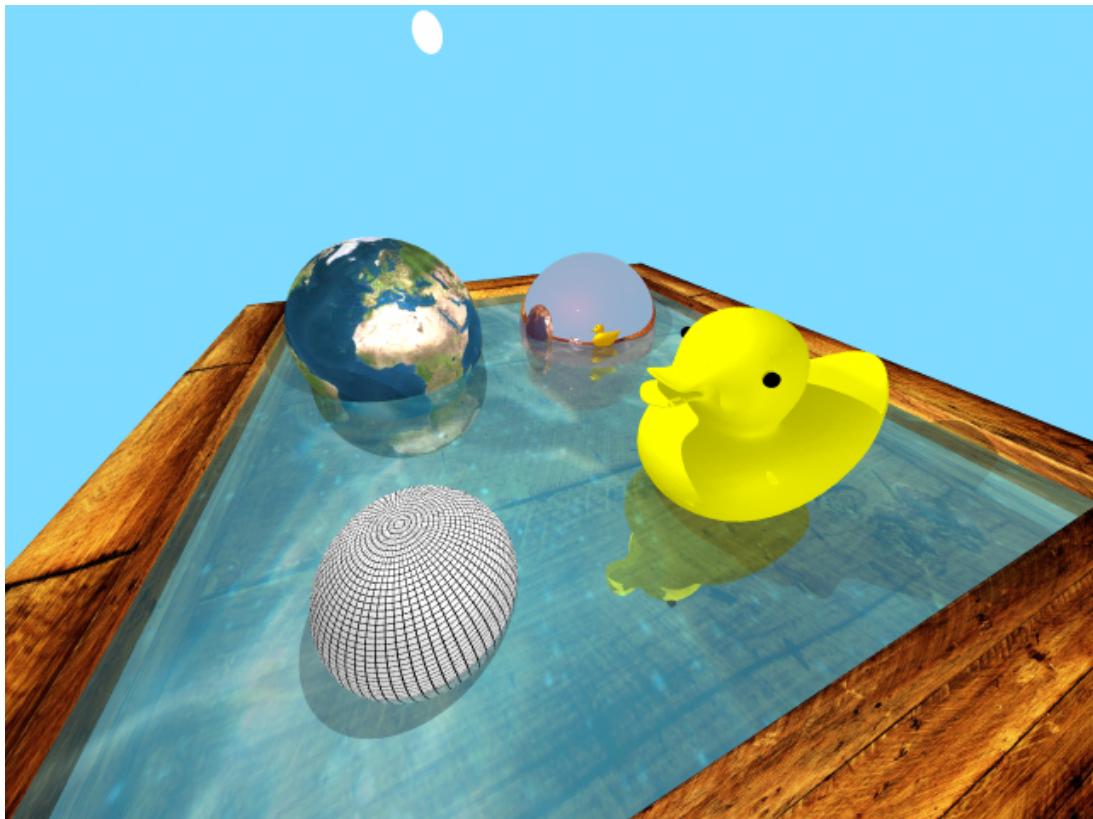
## 10 Ergebnisse

In diesem Kapitel sollen einige Ergebnisbilder vorgestellt und auf Besonderheiten hingewiesen werden.  
Ein Eindruck des implementierten Programmes:



Die obere Szene besteht aus zwei triangulierten Objekten, 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 einer Bump-Map.

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



## 11 Quellen

### 11.1 C++Mathe-Bibliothek

Für die hier verwendete Mathematik wurde in dieser Implementierung die Mathe-Bibliothek des Lehrstuhls für Mustererkennung und Bildverarbeitung von Prof. Brox verwendet.

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

### 11.2 Weiterer Code

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

### 11.3 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\\_0dities.jpg](http://fc03.deviantart.com/fs26/i/2008/042/e/d/Local_Texture__Three_by_One_by_Beyond_0dities.jpg)

[http://www.seos-project.eu/modules/landuse/images/2\\_earth\\_2400\\_960.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://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://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://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>