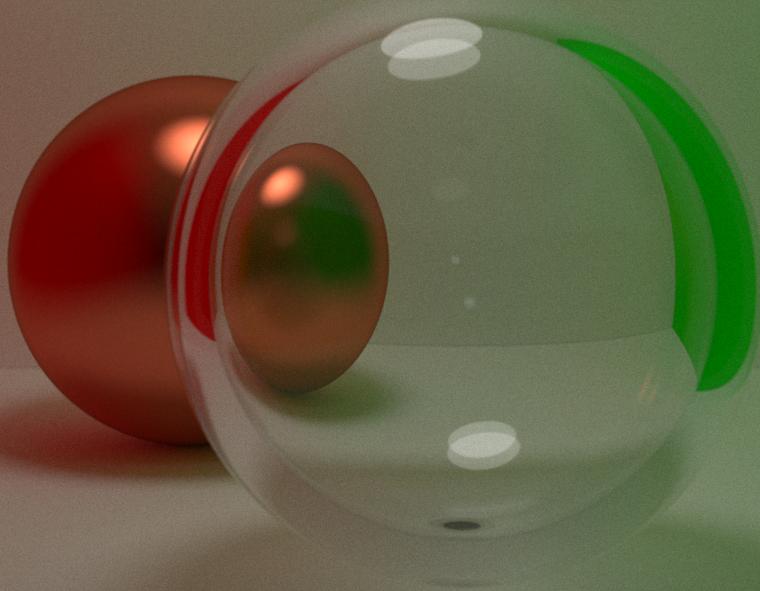


CT Projekt

---

# RAYTRACING

---



Christian Korn

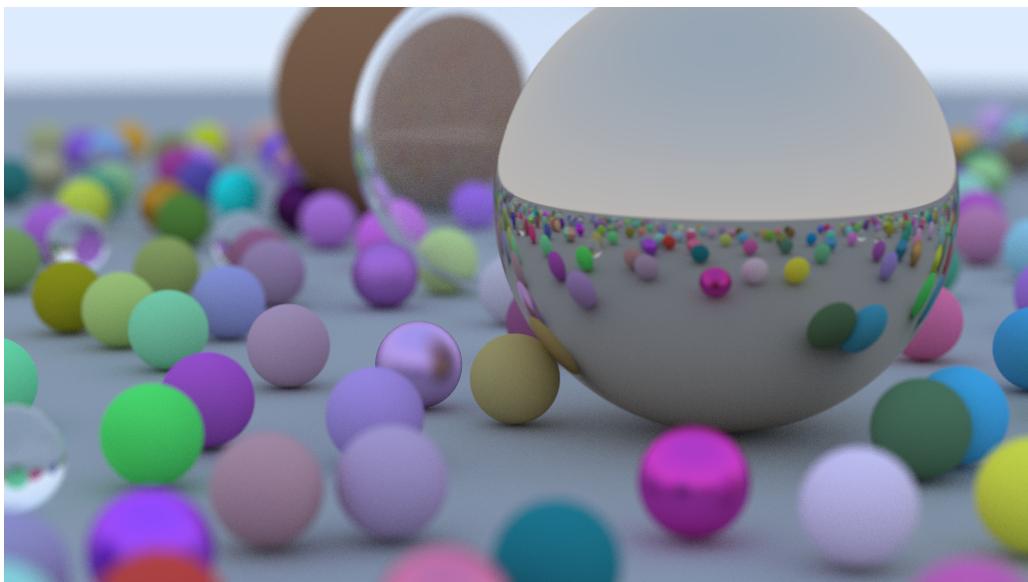
# RAYTRACING

---

Christian Korn, TGI 13/4

SJ2022-2023 H2

Der Quellcode kann unter <https://github.com/MacAphon/RT> gefunden werden.



## 1 Projektidee

Raytracing ist eine Technik zur Erstellung von Bildern, die auf der Simulation von Licht basiert. Dabei wird für jeden Pixel des Bildes ein (oder mehrere) Strahlen in die Szene geschickt und die Farbe des Pixels anhand der Objekte, die der Strahl trifft, berechnet. Durch die Simulation von Licht können realistische Bilder erstellt werden, die sich von Bildern, die mit Rasterisierung erstellt wurden, unterscheiden.

Dieses Projekt orientiert sich an Peter Shirley's Buch „Ray Tracing in One Weekend“ [1]. Teile des Codes orientieren sich außerdem an den Büchern „Ray Tracing: The Next Week“ [2] und „Ray Tracing: The Rest of Your Life“ [3].

## 2 Ziele

### 2.1 Verbindlich

- Erstellen von Bildern durch Raytracing
- Mehre darstellbare Objekte
- Verschiedene Materialien

Alle verbindlichen Ziele wurden erreicht.

Es sind vier verschiedene Materialien implementiert:

- Lambertian (Matte Oberfläche)
- Metallisch (mit anpassbarer Rauheit)
- Dielektrikum (Glas, mit anpassbarem Brechungsindex)
- Emission

Bei allen Materialien kann die Farbe angepasst werden.

## 2.2 Optional

- Transparenz mit Lichtbrechung
- Multithreading
- Verschiedene Ausgabeformate
- Lichtquellen
- Verschiedene Objekttypen
- Lesen der Szene aus Dateien
- Optimierungen

### Transparenz mit Lichtbrechung

Die Transparenz mit Lichtbrechung wurde implementiert. Zusätzlich wurde eine gute Approximation von Spiegelung bei spitzen Winkeln implementiert.

### Multithreading

Das Programm nutzt einen Threadpool, um die Berechnung der Pixel zu parallelisieren. Die Anzahl der Threads kann über die Kommandozeile angegeben werden.

### Verschiedene Ausgabeformate

Das Programm kann Bilder in den meisten gängigen Bildformaten ausgeben.

### Lichtquellen

Das Programm unterstützt Lichtquellen. Bei jedem Auftreffen eines Strahls auf ein Objekt wird ein Zusätzlicher Strahl in Richtung der nächsten Lichtquelle geschickt.

### Verschiedene Objekttypen

Das Ziel wurde nicht erreicht, da nur Kugeln implementiert wurden. Die Architektur des Programmes erlaubt aber die Implementierung weiterer Objekttypen ohne großen Aufwand.

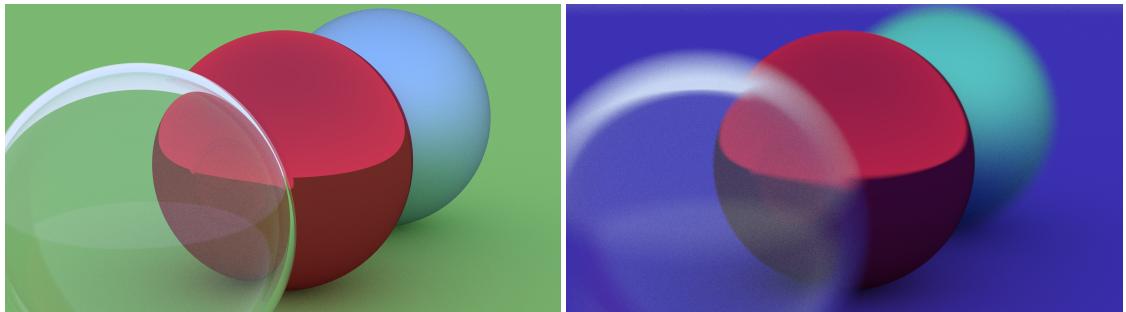
### Lesen der Szene aus Dateien

Das Ziel wurde nicht erreicht, die Szene muss im Code erstellt werden.

### Optimierungen

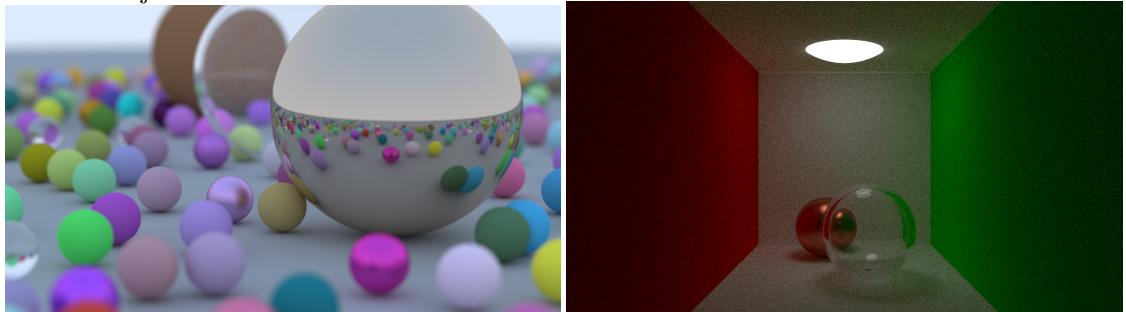
Das Programm ist nicht optimiert. Es gibt aber einige Optimierungsmöglichkeiten, die in Zukunft implementiert werden könnten, wie z.B. Axis-Aligned Bounding Boxes.

### 3 Beispielbilder



Mehrere Objekte mit verschiedenen Materialien

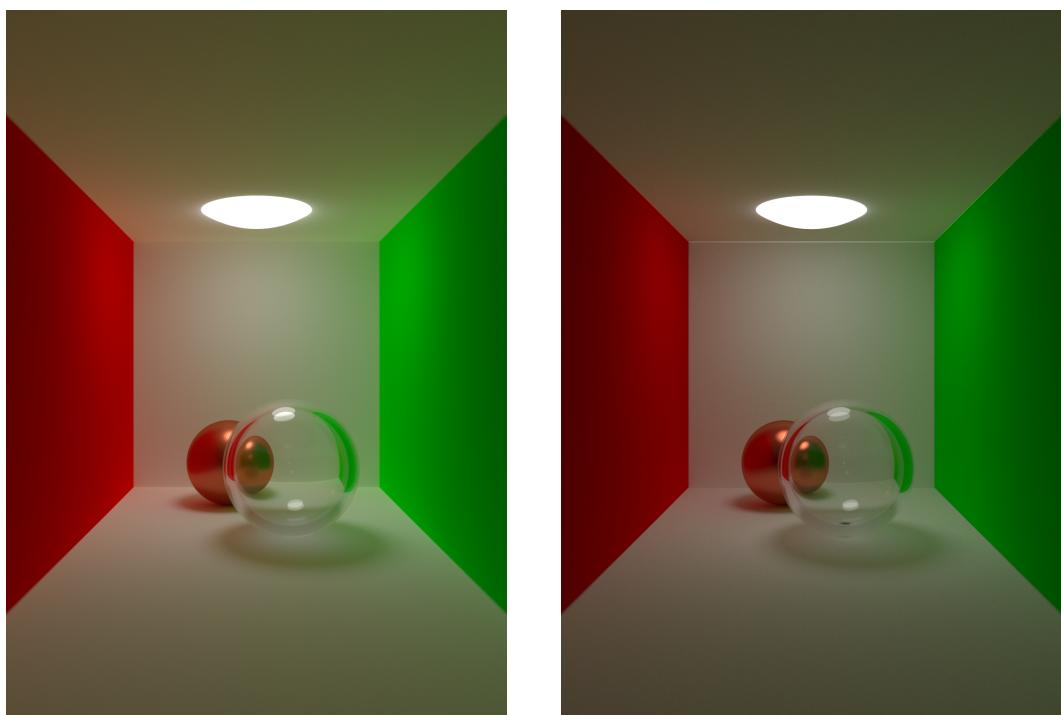
Tiefenschärfe



Große Szene mit vielen Objekten

Für den Effekt einer hohlen Glaskugel wird in die Glaskugel eine zweite, kleinere Kugel gelegt. Die kleinere Kugel hat den gleichen Brechungsindex wie die größere, hat aber einen negativen Radius. Alternativ kann auch der inverse Brechungsindex der äußeren Kugel verwendet werden.

### 4 Vergleich mit Blender



Um zu überprüfen, ob das Programm korrekt funktioniert, wurde ein Vergleich mit Blender

durchgeführt. Dazu wurde eine Szene im eigenen Programm erstellt und in Blender nachgebaut. Die Szene ist eine Cornell-Box mit einer metallisch glänzenden Kugel und einer hohlen Glaskugel. Die Lichtquelle ist eine Kugel in der Decke der Box. Beide Bilder wurden mit einer Auflösung von 4094x5793 Pixeln, 1024 Samples pro Pixel und 24 Bounces pro Sample gerendert. Die Renderzeit betrug ca. 4 Stunden für das eigene Programm und 23 Minuten für Blender. Beide Bilder sind ähnlich verrauscht.

Es können einige Unterschiede zwischen den Bildern festgestellt werden: In Blender können bei der beleuchtenden Kugel die Flächen Kanten der Polygone gesehen werden, während im eigenen Programm die Kugel glatt erscheint. Am Berührungsrand der Glaskugel mit dem Boden ist im eigenen Programm ein schwarzer Punkt zu sehen, der in Blender nicht vorhanden ist. Im eigenen Programm sind außerdem die oberen Kanten der Box unnatürlich hell. Insgesamt ist das Blender Bild etwas heller als das eigene, beide sind aber ähnlich genug, um zu sagen, dass das Programm funktioniert.

Beim Erstellen der Szene in Blender ist aufgefallen, dass das eigene Programm Bilder spiegelverkehrt rendernt. Um die Bilder zu vergleichen, wurde die Szene in Blender spiegelverkehrt aufgebaut.

## Quellen

- [1] Peter Shirley. *Ray Tracing in One Weekend*. Dez. 2020. URL: <https://raytracing.github.io/books/RayTracingInOneWeekend.html>.
- [2] Peter Shirley. *Ray Tracing: The Next Week*. Dez. 2020. URL: <https://raytracing.github.io/books/RayTracingTheNextWeek.html>.
- [3] Peter Shirley. *Ray Tracing: The Rest of Your Life*. Dez. 2020. URL: <https://raytracing.github.io/books/RayTracingTheRestOfYourLife.html>.

Die Ausarbeitung wurde mit L<sup>A</sup>T<sub>E</sub>X erstellt, unter Verwendung von Github Copilot.