# Algorithm Engineering

## Projekt: Interaktiver Raytracer

Christoph Tank, Christian Schubert

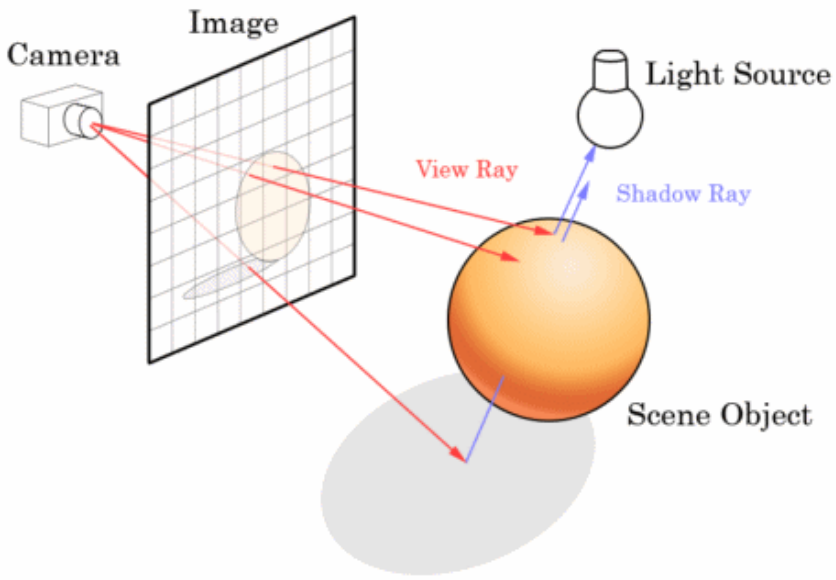
8. Februar 2016

# Inhaltsverzeichnis

# 1 Einleitung - Raytracing

- Raytracing - „Strahlenverfolgung“
- Algorithmus zur Bildsynthese (Erzeugung eines Bildes aus einer 3D-Szene)
- durch Zufallszahlen werden „Lichtstrahlen“ im Bild simuliert

# 1 Einleitung - Raytracing



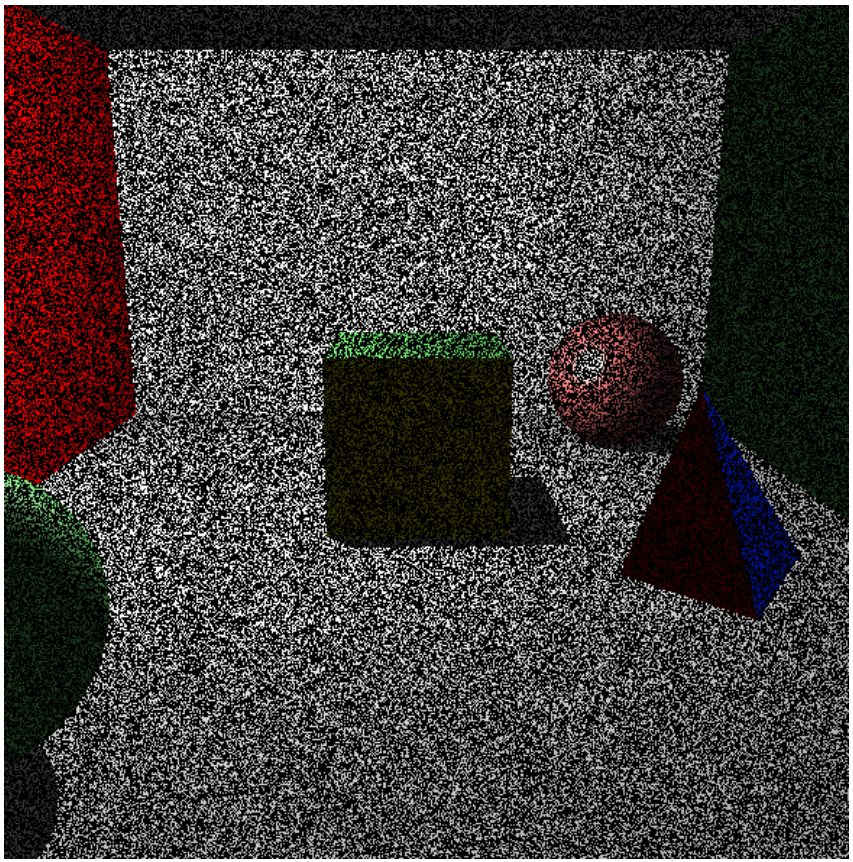



Abbildung 1: Aufbau Raytracer Quelle: blog.codinghorror.com
Abbildung 2: Interaktiver Raytracer

# 2.1 OpenMP - Shared-Memory

```
omp_set_num_threads(NUM_THREADS);
#pragma omp parallel for
for (int i = 0; i < num_samples; i++) {
        /*work*/
        //get random coordinates
        //ray_origin+ray_direction
        //-> intersection of objects
        //Object color in buffer
}
```

- Optimale Anzahl an Threads durch Vergleich von Rechenzeiten eines Durchlaufes mit fester Anzahl von Ray-samples (um u.a. unnötigen Overhead zu vermeiden)

# 2.2 MPI - kommunizierende Prozesse

- Ein Prozessor vom Localhost ist Master (kein Xeon Phi)

```
if(taskid==0) {
//open Window
//communicate Camera position via MPI to workers
/*work:trace random samples*/
//receive Pixelbuffer from workers:
        for(int i=1; i<numtasks; i++){
                MPI_Recv(&buffer[...],...,i,...);
        }
}
```

# 2.2 MPI - kommunizierende Prozesse

- Localhost und Xeon Phi sind „Workers“

```
if(taskid!=0)    {
//receive Camera position via MPI from Master
/*work:trace random samples*/
//send Pixelbuffer to master:
        MPI_Send(&buffer[...],...,taskid,...);
}
```

# 3 Vektorisierung

- Vektor mit Zufallszahlen vor Beginn der Iteration
- Verwendung von „aligned“ Vektoren

```
#include <boost/align/aligned_allocator.hpp>

template <typename T>
using aligned_allo =
boost::alignment::aligned_allocator<T, 64>;
template <typename T>
using aligned_vector =
std::vector<T, aligned_allo<T>>;

aligned_vector<Object*> scene_objects;
aligned_vector<...>...;
```

# 3 Vektorisierung

Vektorisierung für den Fall nicht umsetzbar, aufgrund:

- speziellem Aufbau des interaktiven Raytracers z.b. ineinandergreifen von selbstdefinierter Vektorenklassen
- Anzahl der Iterationen vor Schleifenausführung unbekannt
- bedingter Terminierung ( breaks; Schleifen-Bedingungen)

D.h. um Vektorisierung umzusetzen ist eine teilweise Umstrukturierung des Programmes notwendig.

# 4 Speedup

**OpenMP**

**10.000 Iterationen** in [sec]:

| Threads | **Xeon E5** | **Xeon Phi** |
|---|---|---|
| 1 | 0.05981 | 0.39654 |
| 4 | 0.02151 | 0.16023 |
| 8 | 0.01260 | 0.10021 |
| 60 | | 0.04558 |
| 120 | | 0.03282 |
| 180 | | 0.02793 |
| 240 | | 0.02313 |

**Speedup Xeon E5**:

$$\frac{1ThreadXeonE5}{8ThreadsXeonE5} = \frac{0.05981}{0.01260} = 4.74683$$

**Speedup Xeon Phi**:

$$\frac{1ThreadXeonPhi}{240ThreadsXeonPhi} = \frac{0.39654}{0.02313} = 17.14397$$

# 4 Speedup

**OpenMP+MPI**
**10.000 Iterationen** in [sec]:

| Threads | **Xeon E5 oder Xeon Phi** |
|---|---|
| 1 | 0.42147 |
| 4 | 0.17441 |
| 8 | 0.10418 |
| 60 | 0.05236 |
| 120 | 0.03223 |
| 180 | 0.02571 |
| 240 | 0.02423 |

**Speedup**:

$$\frac{1ThreadXeonE5}{\frac{1}{6}(MPI)*240ThreadsXeon} = \frac{0.05981}{\frac{1}{6}*0.02423} = 14.81125$$

# 5 Herausforderungen/erlernte Fähigkeiten

- Ausbau von Programmierfähigkeiten :P
- Kontinuierlicher Aufbau eines Programms im Team
- Überführung der Struktur eines seriellen Programms in ein paralleles Programm
- verschiedene Anwendungsmöglichkeiten von MPI und OpenMP
- Lösung von Problemen durch verschiedene Ansätze

# Vielen Dank für die Aufmerksamkeit!