Institut für Visualisierung
und Interaktive Systeme

Prof. Dieter Schmalstieg Dr. Guido Reina

3. Übungsblatt zur Vorlesung Computergraphik im WS 2025/26

Besprechung am Montag, Dienstag & Mittowch 24.11., 25.11 & 26.11.2025

Aufgabe 1 *Analytische Geometrie [4 Votierpunkte]*

Um ein Bild einer Szene mittels Raytracing zu erzeugen, benötigt man sogenannte Sichtstrahlen. Gegeben sei ein Sichtstrahl $\vec{r}(t) = \vec{p} + t\vec{d}$, mit dem Ortsvektor des Ausgangspunktes $\vec{p} = (-1, -1, 1)^\top$ und dem Richtungsvektor $\vec{d} = (1, 1, 0)^\top$.

1. Berechnen Sie die Schnittpunkte des Sichtstrahls mit einer Kugel mit Radius $r = 3$ und Mittelpunkt $M = (5, 5, 2)^\top$. Welcher der beiden Schnittpunkte liegt in Richtung des Sichtstrahls vorne und ist also für den Betrachter sichtbar?

2. Berechnen Sie den Schnittpunkt des Sichtstrahles mit der Ebene mit Abstand $d = 3$ vom Ursprung und der Normalen $\vec{n}$ mit Richtung $(1, 1, 0)^\top$. Gehen Sie davon aus, dass die Ebene vor der Kamera liegt.

3. Berechnen Sie den Schnittpunkt des Sichtstrahls mit dem Dreieck A, B, C mit $A = (6, 0, 0)^\top$, $B = (0, 6, 0)^\top$ und $C = (0, 0, 6)^\top$.

4. Begründen Sie, warum der Sichtstrahl das Dreieck schneidet – oder warum nicht.

Geben Sie vollständige Rechenwege mit Zwischenschritten an.

Aufgabe 2 *Raytracing [4 + 2 Votierpunkte]*

In dieser Aufgabe sollen Sie einen einfachen CPU-Raytracer implementieren. Dafür finden Sie in ILIAS ein Programmgerüst, das wie folgt aufgebaut ist:

main Hier wird die Szene erstellt und das Raytracing durchgeführt.

sceneobject Enthält eine abstrakte Klasse `SceneObject`, die ein Objekt in der Szene repräsentiert. Außerdem die konkreten Objekttypen `Plane` und `Sphere`, die von `SceneObject` erben und zwei Methoden implementieren: `intersect(...)`, in der ein Schnitt des Objektes mit einem Strahl berechnet bzw. überprüft wird, sowie `getSurfaceColor(...)`, welche die Oberflächenfarbe des Objektes zurückgibt.

util / vec3 Enthält eine 3D-Vektorklasse, eine Ray-Klasse, die einen Strahl repräsentiert, sowie Utility-Funktionen zur Erzeugung und Vergleich von PPM-Bildern und Zufallszahlengenerierung.

Das Programm erzeugt nach erfolgreichem Ausführen ein Ergebnisbild `result.ppm`. Dieses sollte im Auslieferungszustand des Programmgerüstes komplett schwarz sein.

Aufgabe 2.1 Strahlerzeugung und -verfolgung

Implementieren Sie zunächst die Strahlerzeugung und -verfolgung in `main.cpp`. Gehen Sie dabei wie folgt vor (die entsprechenden Stellen im Code sind mit `TODO` 2.1.n markiert):

1. Erzeugen Sie in der Methode `render(...)` für jeden Pixel auf der Bildebene einen Sichtstrahl, der die Kameraposition als Ursprung hat und durch den Mittelpunkt des Pixels geht. Benutzen Sie die vorgegebenen Parameter der Kamera und Bildebene. Rufen Sie mit jedem so erzeugten Strahl die Methode `castRay(...)` auf.
   Hinweis: Es handelt sich um ein rechtshändiges Koordinatensystem. Die Kamera blickt in negative z-Richtung, der up-Vektor zeigt in positive y-Richtung. Die Bildkoordinaten laufen von links oben nach rechts unten.
   Ergebnisbild: Die `castRay(...)` Methode gibt bisher lediglich eine dunkelblaue Hintergrundfarbe zurück. Wenn Sie das Programm nun erneut ausführen, sollte das Ergebnisbild vollständig dunkelblau sein.

2. Rufen Sie in der Methode `castRay(...)` die Methode `trace(...)` auf. Sollte der Strahl ein Szenenobjekt treffen, berechnen Sie außerdem den Schnittpunkt und geben die Farbe des Objektes am getroffenen Punkt zurück.
   Ergebnisbild: Keine Änderung zu 1.

3. Iterieren Sie in der Methode `trace` über alle Szenenobjekte. Sollte mindestens eines der Objekte getroffen werden, übergeben sie die Referenz auf das getroffene Object, welches sich am nächsten zur Kamera befindet, in `hitObject`, sowie den Abstand zu diesem Schnittpunkt in `t_near`.
   Ergebnisbild: Wie in Abbildung 1 (links).

4. Welche negativen Effekte können Sie in den Ergebnisbildern erkennen? Geben Sie zwei Beispiele im Bild an und machen Sie Lösungsvorschläge, um diese zu verhindern. Sie können diese Aufgabe mithilfe der Bilder im Aufgabenblatt lösen.

So lange lediglich die `intersect(...)`-Methode für Ebenen implementiert ist, sind auch nur diese Szenenobjekte im Ausgabebild (vgl. Abbildung 1 links) zu sehen.

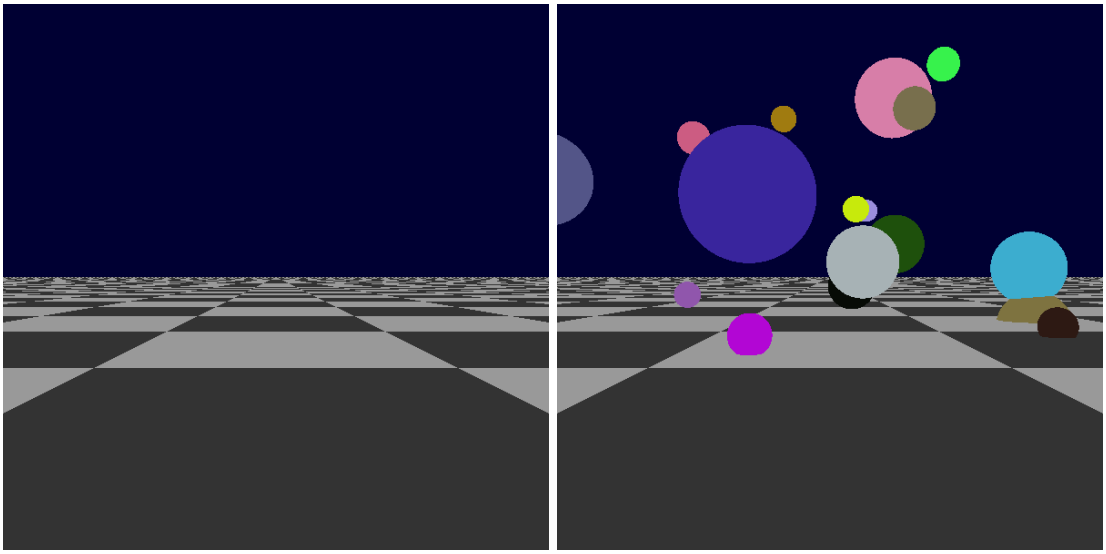

Abbildung 1: Referenzausgaben für Aufgabe 2.1 (links) und Aufgabe 2.2 (rechts).

Aufgabe 2.2 Strahl-Kugel-Schnitt

Implementieren Sie die `intersect(...)`-Methode für Kugel-Objekte in der `Sphere`-Klasse (in `sceneobject.cpp`), indem Sie den Schnittpunkt für Strahlen mit Kugeln berechnen.
Ergebnisbild: Wie in Abbildung 1 (rechts).

Hinweis: Sie können Ihre Implementierung mit Hilfe der Tests `TEST_RAY_GENERATION` bzw. `TEST_SPHERE_INTERSECT` überprüfen, welche das Ausgabebild auf Pixelebene gegen das entsprechende Referenzbild (Abbildung 1) vergleicht.