Übungsblatt 2

Willkommen zur zweiten Übung der Veranstaltung *Generative Computergrafik* im Sommersemester 2023. Ziel dieses Übungsblatts ist, dass Sie sich mit den geometrischen Grundlagen von *Raytracings* vertraut machen.

Aufgabe 1. Geben Sie das Kamerakoordinatensystem $\mathbf{f}, \mathbf{s}, \mathbf{u}$ an, welches zu

$$\mathbf{e} = (0, 3, 4)^\top, \mathbf{c} = (0, 0, 0)^\top \text{ und } \mathbf{up} = (0, 2, 0)^\top$$

korrespondiert.

Aufgabe 2. Bestimmen Sie (falls vorhanden) den *ersten* Schnittpunkt des Sichtstrahls $\mathbf{r}(t) = (0, 1, 1)^\top + t \cdot (0, 0, -1)^\top$ mit

1. der Kugel mit Mittelpunkt $\mathbf{c} = (0, 0, -4)^\top$ und Radius $r = 2$,
2. der Ebene durch den Punkt $\mathbf{e} = (0, 0, -4)^\top$ mit Normale $\mathbf{n} = 1/\sqrt{3}(1, 1, 1)^\top$
3. dem Dreieck mit den Ecken $\mathbf{a} = (-1, 0, -2)^\top, \mathbf{b} = (1, 0, -4)^\top, \mathbf{c} = (0, -2, -3)^\top$

Welches der drei Objekte wird auf das Pixel zu dem Sichtstrahl abgebildet?

Aufgabe 3. Implementieren Sie einen einfachen Raytracer. Gehen Sie hierbei vom Ray Tracing Pseudo Code auf Folie 18 der dritten Vorlesung aus. **Verwenden Sie die Python-Bibliothek `Numpy` für alle Berechnung von Schnitten zwischen Strahlen und den unterschiedlichen Objekten einer Szene**. Ihr Raytracer soll dabei für jedes Pixel des erzeugten Bildes lediglich den Abstand des Objekts vom Kamerazentrum (in Richtung des entsprechenden Strahls) darstellen. Abbildung 1 zeigt ein Bild der Größe 400×400 Pixel, aufgenommen mit einer Kamera zu $\mathbf{e} = (0, 1.8, 10)^\top, \mathbf{c} = (0, 3, 0)^\top, \mathbf{up} = (0, 1, 0)^\top$ und $FOV = \pi/4$. Dargestellt werden drei gleich große Kugeln, ein Dreicke, dessen Eckpunkte den Mittelpunkten der Kugeln entsprechen und eine Ebene, die den Boden repräsentiert. Ein Beispiel zur Erzeugung eines Bildes mit `Numpy` und `PIL` finden Sie auf der nächsten Seite.

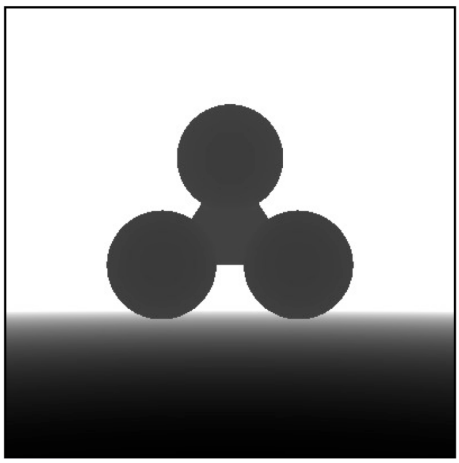

Abbildung 1: Raytracing einer Szene, welche aus drei Kugeln, einem Dreieck, dessen Eckpunkten die Kugelmittelpunkte sind (in der Mitte) und einer Ebene (unten) besteht. Die dargestellten Grauwerte repräsentieren die Entfernung eines 3D Objektpunkts vom Kamerazentrum in Richtung des Sehstrahls. Je hellen ein Pixel dabei ist, desto weiter entfernt ist dabei der korrespondierende 3D Punkt vom Kamerazentrum. Weiß entspricht dabei 3D Punkten, die "unendlich" weit entfernt sind.

Listing 1: Python-Beispiel zur Erzeugung eines Farbbildes mit Hilfe von `Numpy` und `PIL`.

```
from PIL import Image
import numpy as np

# image width, height
w, h = 600, 400

# generate a gray scale (one channel) noise image
gs_img1 = np.random.rand(w, h)
gs_img2 = np.random.rand(w, h)
gs_img3 = np.random.rand(w, h)

# scale image values to [0,255]
gs_img1 = 255*gs_img1
gs_img2 = 255*gs_img2
gs_img3 = 255*gs_img3

# generate 8 Bit PIL image. These are transposed compared
# to numpy arrays.
r = Image.fromarray(gs_img1.astype(np.uint8).T)
g = Image.fromarray(gs_img2.astype(np.uint8).T)
b = Image.fromarray(gs_img3.astype(np.uint8).T)

# make RGB image (here all three channels are the same)
rgb = [r, g, b]

# merge channels and sage image
Image.merge("RGB", rgb).save("noise_image.png")
```

Abbildung 2: Beispiel eines Bildes, das mit dem Listing 1 erzeugt wurde.