Technische Universität Berlin **SoSe 2021**
Fakultät II, Institut für Mathematik
Sekretariat MA 5–2, Dorothea Kiefer-Hoeft
Prof. Dr. Martin Skutella
Sven Jäger, Dr. Frank Lutz, Manuel Radons

Programmierprojekt Computerorientierte Mathematik II

Vorstellung des Projekts bei Ihrer Tutorin bzw. Ihrem Tutor bis: 29.06.2021

1. Thema

Ziel des diesjährigen Programmierprojekts ist es, Ihnen einen Einstieg in grundlegende Konzepte der Computergrafik zu vermitteln. Hierzu sollen Sie eine Textur in Form eines PNG-Bildes auf eine Kugel im Raum projizieren und von dieser Kugel mit Textur „ein Foto schießen“, welches wiederum als PNG ausgegeben werden soll. Das Einlesen soll dabei mittels `Python` durchgeführt werden, während die Bildverarbeitung in `Julia` stattfindet. Das Programmierprojekt wird nicht über den CoMA-Judge abgegeben, sondern soll bis zum 29.6.2021 bei Ihrer Tutorin bzw. Ihrem Tutor vorgestellt werden.

2. Einlesen der Textur in Python

Die `Pillow`-Bibliothek in `Python` stellt Methoden zum Einlesen und Abspeichern von Bildern im PNG-Format bereit. In der Gestaltung des `Python`-Teils des Programmierprojekts sind Sie relativ frei. Die einzigen Mindestanforderungen sind die folgenden: Ihr `Python`-Programm soll eine `Julia`-Funktion `snapshot_sphere(b,h,daten,m,r,dichte)` aufrufen.

- Die Parameter `b,h,daten` sind dabei Breite und Höhe eines eingelesenen PNG-Bildes, sowie eine Liste der Länge `b` $\times$ `h` von 4-Tupeln der RGBA-Werte der Pixel des besagten PNG-Bildes.
- Der Parameter `m` ist ein 3-Tupel das den Mittelpunkt einer 2-Sphäre im $\mathbb{R}^3$ darstellt. Der Parameter `r` ist der Radius besagter Sphäre.
- Der Parameter `dichte` ist ein Qualitätsparameter, dessen Bedeutung weiter unten erörtert wird.

Rückgabewert von `snapshot_sphere` soll eine Liste von 250.000 4-Tupeln von RGBA-Werten sein, die mittels der Methoden der Pillow-Bibliothek zur Generierung eines PNG-Bildes im Format 500×500 Pixel genutzt werden soll.

3. Bildverarbeitung in Julia

In diesem Abschnitt legen wir das notwendige theoretische Fundament zur Bewältigung des Projekts und machen einen Vorschlag für eine sinnvolle Strukturierung des Unterfangens in Teilfunktionen. An einigen Stellen lassen wir Ihnen die Wahl, sich für eine einfachere oder kompliziertere Vorgehensweise zu entscheiden. Falls Sie sich sicher genug fühlen, ist es Ihnen auch überlassen, die Untergliederung in Teilfunktionen komplett umzustrukturieren, sofern am Ende die gleiche – oder sogar eine erweiterte – Funktionalität steht. Von unserer Seite sind lediglich Mindestanforderungen gegeben, sowie ein maximal vereinfachter Bauplan um diese zu erfüllen. Fühlen Sie sich frei, hierüber hinauszugehen. Anregungen hierzu finden sich, zum Beispiel, in dem hervorragenden DIY Buch `Ray Tracing in One Weekend` von Peter Shirley, welches frei erhältlich ist unter

`https://www.realtimerendering.com/raytracing/Ray%20Tracing%20in%20a%20Weekend.pdf`

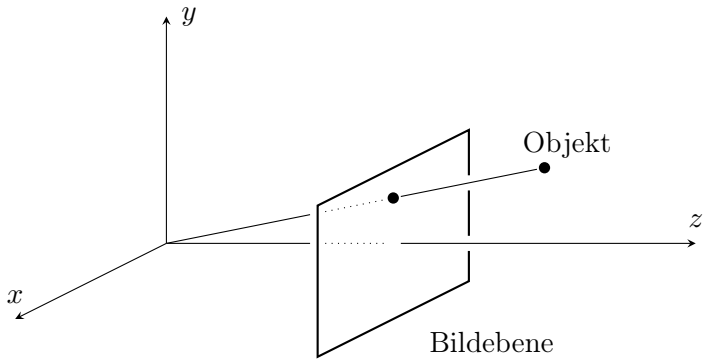

Abbildung 1: Vereinfachtes Kameramodell

Kamera

Ein vereinfachtes Modell einer Kamera besteht aus einer Bildebene und einem Fokuspunkt, siehe Abbildung 1. Wir machen die folgenden Annahmen:

1. Die Bildebene steht orthogonal auf der z-Achse und hat Koordinaten in

$$[-250, 250) \times [-250, 250) \times \{250\} \subset \mathbb{R}^3 .$$

2. Der Fokuspunkt ist der Koordinatenursprung $\mathbf{0} := (0, 0, 0)^T \in \mathbb{R}^3$.

Es sei $p \in \mathbb{R}^3$ gegeben. Falls der Streckenzug $[\mathbf{0}, p]$ die Bildebene schneidet, so nennen wir den Schnittpunkt $\bar{p}$ das Abbild von p. Ist letzteres nicht der Fall, so sagen wir, dass p nicht im Bild der Kamera liegt.
Wir bezeichnen nun mit `p` das Tupel (x, y, z), welches die Koordinaten des Punktes $p = (x, y, z)^T\mathbb{R}^3$ enthält. Die erste Teilaufgabe (im Sinne eines Vorschlages zur Strukturierung) ist: Schreiben Sie eine `Julia`-Funktion `abbild(p)`, welche Folgendes zurückgibt:

1. Falls p im Bild der Kamera liegt, so gibt die Funktion ein Integer-Tupel (x', y') wieder, wobei x' und y' die abgerundeten ganzzahligen $x-$ und $y-$Koordinaten von $\bar{p}$ sind.

2. Falls p nicht im Bild der Kamera liegt, so gibt die Funktion `nothing` zurück.

Sichtbarkeit eines Punktes auf der Sphäre

Eine Gerade g durch den Koordinatenursprung des $\mathbb{R}^3$ läßt sich folgendermaßen parametrisieren:

$$g : \mathbb{R} \to \mathbb{R}^3, \quad t \mapsto t \cdot \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} ,$$

wobei $\alpha, \beta, \gamma \in \mathbb{R}$. Die Schnittpunkte der Geraden g mit einer 2-Sphäre S vom Radius r mit Mittelpunkt $m := (x_0, y_0, z_0)^T \in \mathbb{R}^3$ sind gegeben durch die quadratische Gleichung

$$\begin{aligned} (t\alpha - x_0)^2 + (t\beta - y_0)^2 + (t\gamma - z_0)^2 &= r^2 \\ \iff (\alpha^2 + \beta^2 + \gamma^2) \cdot t^2 - 2(\alpha x_0 + \beta y_0 + \gamma z_0) \cdot t + (x_0^2 + y_0^2 + z_0^2 - r^2) &= 0 \end{aligned}$$

Es seien nun

$$a := (\alpha^2 + \beta^2 + \gamma^2)\,, \quad b := -2(\alpha x_0 + \beta y_0 + \gamma z_0)\,, \quad c := x_0^2 + y_0^2 + z_0^2 - r^2\,.$$

Dann lassen sich die Schnittpunkte $s_{1,2}$ der Geraden g mit S mittels der p/q-Formel berechnen:

$$s_{1,2} \;=\; \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}\,.$$

Dabei gibt es keinen Schnittpunkt, falls $\sqrt{b^2 - 4ac} < 0$, einen Schnittpunkt, falls $\sqrt{b^2 - 4ac} = 0$ und zwei Schnittpunkte, falls $\sqrt{b^2 - 4ac} > 0$.
Die Zweite Teilaufgabe, bzw. der zweite Vorschlag für eine Teilfunktion, ist nun: Es sei eine Sphäre S gegeben durch einen Radius `r` und einen Mittelpunkt $m := (x_0, y_0, z_0)^T \in \mathbb{R}^3$. Des Weiteren sei $p := (x, y, z)^T \in S$. Wie im vorigen Aufgabenteil etabliert, schreiben wir ebenfalls `p` und `m` für die Tupel, welche die Koordinaten der gleichnamigen Punkte des $\mathbb{R}^3$ bzw. der Sphäre S enthalten.
Schreiben sie nun eine `Julia`-Funktion `is_visible(p,m,r)`, welche Folgendes leistet:

1. Die Funktion gibt `true` zurück, falls p im Bild der Kamera liegt *und* das Liniensegment $[\bar{p}, p]$ die Sphäre in keinem zweiten Punkt schneidet.

2. Die Funktion gibt `false` zurück, falls p nicht im Bild der Kamera liegt *oder* falls p im Bild der Kamera liegt und das Liniensegment $[\bar{p}, p]$ die Sphäre in einem zweiten Punkt schneidet.

Nutzen Sie hierbei die Funktion `abbild` aus dem vorigen Paragraphen.

Parametrisierung der Sphäre

Eine Parametrisierung der 2-Sphäre mit Mittelpunkt $m := (x_0, y_0, z_0)^T \in \mathbb{R}^3$ und Radius r ist durch folgende Funktion gegeben:

$$f : [0, \pi] \;\times\; [0, 2\pi] \;\to\; \mathbb{R}^3\,, \quad (\theta, \varphi) \;\mapsto\; \begin{pmatrix} x_0 + r \cdot \sin(\theta) \cdot \cos(\varphi) \\ y_0 + r \cdot \sin(\theta) \cdot \sin(\varphi) \\ z_0 + r \cdot \cos(\theta) \end{pmatrix}$$

Ein PNG-Bild mit $b \times h$ Pixeln kann nun mittels verschiedener Strategien auf die Sphäre abgebildet werden. Der einfachste Weg ist, den Pixel (x, y) abzubilden auf $f(x \cdot \pi/h, y \cdot 2 \cdot \pi/b)$. Je nach Auflösung des Bildes können sich hierbei unschöne Rauscheffekte ergeben. Abhilfe kann diesem Problem dadurch geschaffen werden, dass man Pixel mehrfach sampelt, also den Wert des Pixels auf mehrere Punkte in der Sphäre abbildet. Ein Weg, dies zu erreichen, ist, eine gleichverteilt zufällige Liste von Werten $(\delta_x^1, \delta_y^1), \ldots, (\delta_x^k, \delta_y^k) \in [0, 1] \times [0, 1]$ zu erstellen und anschließend die Werte $f((x + \delta_x^i) \cdot \pi/h, (y + \delta_y^i) \cdot 2 \cdot \pi/b)$, wobei $i \in \{1, \ldots, k\}$, zu berechnen. Aber auch äquidistante Rasterungen des Einheitsquadrats sind denkbar.
Es seien nun Bildabmessungen `b,h` für Pixelbreite und -Höhe eines Bildes gegeben, sowie Pixelkoordinaten `x,y`, wobei $0 \leq$ `x` $<$ `b` gilt, sowie $0 \leq$ `y` $<$`h`. Der nächste Strukturierungs-Vorschlag ist: Schreiben sie eine `Julia`-Funktion `samples(x,y,b,h,m,r,dichte)`, die eine Liste von von Koordinatentupeln von Punkten auf der Sphäre S, die durch `m` und `r` definiert ist, zurückgibt. Hierbei soll `dichte` ein Parameter sein, der die Zahl der Samples pro Pixel variiert. Es ist dabei Ihnen überlassen, nach welcher Methode sie Koordinaten und Zahl der Samples bestimmen.

Gesamtprogramm Julia-Teil

Die Funktion `snapshot_sphere(b,h,daten,m,r,dichte)` sollte nun Folgendes leisten:

1. Lege eine leere Liste `Bildebene` der Länge 500×500 an.
2. Setze $l := b \times h$. Für alle $i \in \{1, \ldots, l\}$ führe Folgendes durch:
   - Ermittle die entsprechenden Pixel-Koordinaten (x, y) im übergebenen Textur-PNG.
   - Erstelle mittels `samples(x,y,b,h,dichte)` eine Liste `punkte` von Punkten auf der Sphäre S, die durch `m` und `r` gegeben ist.
   - Für jeden Punkt `p` in `punkte` ermittle mittels `is_visible(p,m,r)` ob er im Bild der Kamera liegt und sichtbar ist. Falls nein, ignoriere ihn. Falls ja, schreibe den RGBA-Wert aus `daten[i]` in ein geeignetes Feld von `Bildebene`.
     Anmerkung: Es kann passieren, dass mehrere RGBA-Werte in ein Feld von `Bildebene` geschrieben werden. Man kann auf verschiedene Weisen mit dieser Eventualität umgehen. Eine Möglichkeit ist, den bereits vorhandenen Wert schlicht zu überschreiben. Alternativ kann man alle Werte in eine Liste schreiben und anschließend den Mittelwert bilden. Hier haben Sie die Freiheit, die von Ihnen bevorzugte Herangehensweise zu wählen.
3. Gib die Liste `Bildebene` zurück in die Python-Umgebung.

Auch diese Liste ist wiederum als Strukturierungs-Vorschlag zu verstehen. Falls Sie die geschilderte Funktionalität anders umsetzen – oder gar erweitern – können, fühlen Sie sich frei dazu.