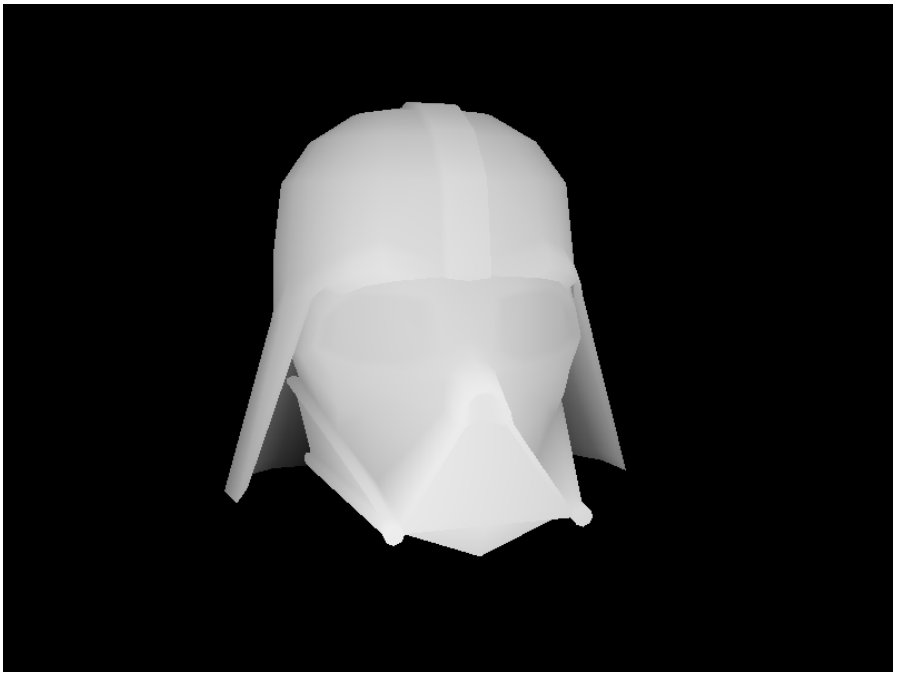

Abbildung 1: Durch Raycasting erzeugtes Bild mit distanzbasiertem Shading

CG/task1a — Einfaches Raycasting

Raycasting ist ein Raytracingverfahren, mit dem die Sichtbarkeit von Objekten von einem Punkt aus in einer gegebenen Szene bestimmt werden kann. Ausgehend von einem Startpunkt wird ein Strahl konstruiert und der erste (d.h. der dem Startpunkt näheste) Schnittpunkt mit einem Objekt der Szene gesucht. Basierend auf Raycasting kann ein Bild einer virtuellen Szene gerendert werden, indem vom Augpunkt einer virtuellen Kamera aus Sichtstrahlen durch alle Pixel der Bildebene erzeugt und deren erster Schnittpunkt mit einem Objekt der Szene gesucht wird. In einem *Shading* genannten Prozess wird für jeden so gefundenen Schnittpunkt die Farbe des Objektes an der getroffenen Stelle bestimmt, welche wiederum die Farbe des Lichts, das vom Objekt auf den entsprechenden Pixel auf der Bildebene fällt, bestimmt.

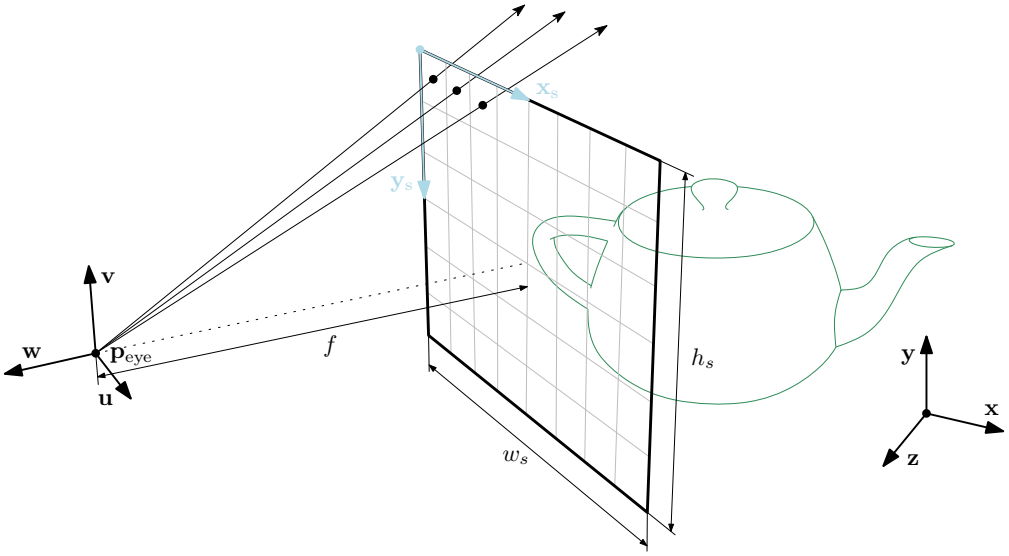

Abbildung 2: Sichtstrahlen werden von der Kameraposition aus durch die Bildebene geschossen. Orientierung und Position von Kamera und Bildebene sind durch das durch $\mathbf{u}$, $\mathbf{v}$, $\mathbf{w}$ und $\mathbf{p}_{\text{eye}}$ gegebene, lokale Kamerakoordinatensystem sowie den Abstand f der Bildebene bestimmt.

1 Raycasting Grundlagen

Ein Strahl ist mathematisch beschrieben durch

$$\mathbf{r}(t) = \mathbf{p} + t \cdot \mathbf{d} \tag{1}$$

wobei $\mathbf{p}$ die Koordinaten des Startpunkes und $\mathbf{d}$ die Richtung des Strahls angeben, der Strahlparameter $t > 0$ entspricht der Distanz des Punktes $\mathbf{r}(t)$ auf dem Strahl vom Startpunkt $\mathbf{p}$ als Vielfaches der Länge der Strahlrichtung $\|\mathbf{d}\|$.

Herzstück eines Raycasters ist die in Abbildung 2 illustrierte Generierung der Sichtstrahlen – auch *Primärstrahlen* genannt – durch die Bildebene. Die Basisvektoren $\mathbf{u}$, $\mathbf{v}$, $\mathbf{w}$ sowie die Position des Augpunktes $\mathbf{p}_{\text{eye}}$ definieren die Position und Orientierung der Kamera in der Szene. Die Bildebene liegt zentriert im Abstand f entlang der Blickrichtung $-\mathbf{w}$, ihre Abmessungen sind durch die Breite w_s und Höhe h_s gegeben.

Um ein Rasterbild der Szene mit Auflösung $r_x \times r_y$ zu erzeugen, wird die Szene durch die Bildebene abgetastet. Jedem Pixel des Ausgabebildes entspricht ein Punkt in der Bildebene, durch den ein Strahl geschossen wird; das erste Objekt, auf das der Strahl trifft, bestimmt die Farbe des Pixels. Zu beachten ist dabei, dass die Pixel eines Bildes in der Regel zeilenweise, beginnend von links oben indiziert werden, der Pixel $(0,0)$ liegt also links oben im Bild, der Pixel $(r_x - 1, r_y - 1)$ rechts unten. Die Samplepositionen ergeben sich aus der Unterteilung der Bildebene in ein regelmäßiges Gitter mit $r_x \times r_y$ Zellen, die Strahlen verlaufen dann durch die Mittelpunkte der Gitterzellen.

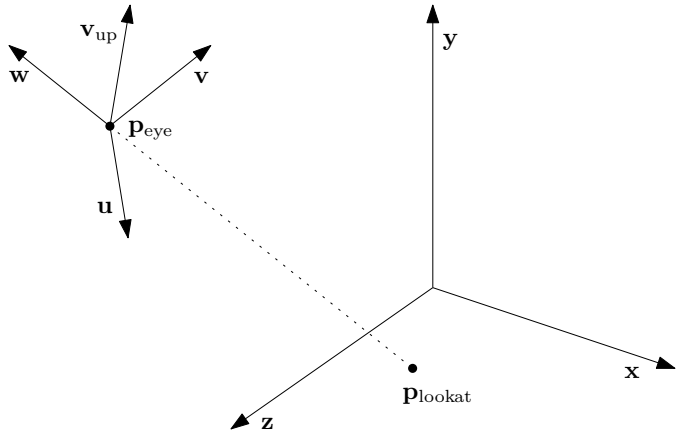

Abbildung 3: Referenzdarstellung eines Kamerakoordinatensystems mit Basisvektoren $\mathbf{u}$, $\mathbf{v}$ und $\mathbf{w}$, sowie den vorgegebenen Konfigurationsparametern $\mathbf{p}_{\text{eye}}$, $\mathbf{p}_{\text{lookat}}$ und $\mathbf{v}_{\text{up}}$.

1.1 Kameraparameter

Da die direkte Angabe der Basisvektoren $\mathbf{u}$, $\mathbf{v}$ und $\mathbf{w}$ relativ kompliziert und fehleranfällig wäre (in der Regel wird eine orthonormale Basis gefordert), wird die Lage der Kamera üblicherweise durch Angabe der Position des Augpunktes $\mathbf{p}_{\text{eye}}$, des *Lookat-Punktes* $\mathbf{p}_{\text{lookat}}$ und eines *Up-Vektors* $\mathbf{v}_{\text{up}}$ fixiert. Der Lookat-Punkt entspricht dem Punkt, auf den die Kamera gerichtet ist, der Up-Vektor legt die Rotation der Kamera um die Blickrichtung fest (er gibt an, welche Richtung für die Kamera „nach oben“ ist). Wie in Abbildung 3 dargestellt, können die Basisvektoren des Kamerakoordinatensystems dann wie folgt berechnet werden:

$$\mathbf{w} = \frac{\mathbf{p}_{\text{eye}} - \mathbf{p}_{\text{lookat}}}{\left\|\mathbf{p}_{\text{eye}} - \mathbf{p}_{\text{lookat}}\right\|} \qquad \mathbf{u} = \frac{\mathbf{v}_{\text{up}} \times \mathbf{w}}{\left\|\mathbf{v}_{\text{up}} \times \mathbf{w}\right\|} \qquad \mathbf{v} = \mathbf{w} \times \mathbf{u}. \tag{2}$$

Der Abstand der Bildebene f entspricht der Brennweite der virtuellen Kamera. Die Abmessungen der Bildebene werden üblicherweise durch Angabe des vertikalen[1] *Field-of-View* β (vertikales Sichtfeld; entspricht dem „Öffnungswinkel“ der Kamera) und der *Aspect-Ratio* $a = \frac{w_s}{h_s}$ (Seitenverhältnis) des Ausgabebildes festgelegt:

$$h_s = 2 \cdot f \cdot \tan\left(\frac{\beta}{2}\right) \qquad w_s = a \cdot h_s. \tag{3}$$

Da eine uniforme Abtastung der Bildebene gewünscht ist, gilt

$$a = \frac{w_s}{h_s} = \frac{r_x}{r_y} \tag{4}$$

[1] Dieses Vorgehen hat den Vorteil, dass sich bei wechselnder Bildschirmbreite das horizontale Sichtfeld automatisch anpasst.

2 Aufgabe

Ziel dieser Übung ist es, den Kern eines einfachen Raycasters zu implementieren. Das zur Verfügung gestellte Framework lädt bereits eine Szene sowie Kameraparameter aus einer JSON-basierten Konfigurationsdatei, stellt Funktionen zur Schnittpunktsberechnung mit der Szenengeometrie bereit und kümmert sich um die Ausgabe des Ergebnisbildes in eine PNG-Datei. Der Pfad zur Konfigurationsdatei wird dem resultierenden Programm als einziges Argument übergeben. Erfolgreiche Ausführung erzeugt eine Datei «`<taskname>.png`» in «`output/<taskname>/`».

2.1 Strahlgenerierung (4 Punkte)

Implementieren Sie in «`task1a.cpp`» die Funktion

```
void render(const Scene1a& scene, surface<float>& output_image)
```

die in `output_image` ein Tiefenbild der in `scene` übergebenen Szene schreiben soll. Berechnen Sie – wie in Abschnitt 1 beschrieben – für jeden Pixel des Ausgabebildes einen Sichtstrahl, bestimmen Sie die Distanz vom Startpunkt des Strahls (Augpunkt der Kamera) bis zum ersten Schnittpunkt mit einem Objekt der Szene und schreiben Sie diese, falls ein solcher Schnittpunkt existiert, in das Ausgabebild. Die zu verwendenden Kameraparameter finden Sie in der über `scene.getCamera()` erreichbaren Struktur; der erste Schnittpunkt eines Strahls mit der Szenengeometrie kann über die Methode `scene.intersectWithRay()` berechnet werden.

2.2 Worte der Weisheit

- Aufgepasst bei der Berechung der Aspect-Ratio: Die Auflösung des Output-Bildes ist in Werten vom Typ `int` angegeben, der Operator `/` auf zwei `int`-Werten führt in C++ eine Ganzzahldivision durch.
- Die Sichtstrahlen werden durch die „Pixelmittelpunkte“ geschossen, d.h. die Pixel des Ausgabebildes liegen nicht an ganzzahligen, sondern an halbzahligen Koordinaten in der Bildebene.

3 Abgabe

Die Aufgaben bestehen jeweils aus mehreren Schritten, die zum Teil aufeinander aufbauen, jedoch unabhängig voneinander beurteilt werden. Dadurch ist einerseits eine objektive

Beurteilung sichergestellt und andererseits gewährleistet, dass auch bei unvollständiger Lösung der Aufgaben Punkte erzielt werden können.

Wir weisen ausdrücklich darauf hin, dass die Übungsaufgaben von jedem Teilnehmer *eigenständig* gelöst werden müssen. Wenn Quellcode anderen Teilnehmern zugänglich gemacht wird (bewusst oder durch Vernachlässigung eines gewissen Mindestmaßes an Datensicherheit), wird das betreffende Beispiel bei allen Beteiligten mit 0 Punkten bewertet, unabhängig davon, wer den Code ursprünglich erstellt hat. Ebenso ist es nicht zulässig, Code aus dem Internet, aus Büchern oder aus anderen Quellen zu verwenden. Es erfolgt sowohl eine automatische als auch eine manuelle Überprüfung auf Plagiate.

Da die abgegebenen Programme halbautomatisch getestet werden, muss die Übergabe der Parameter mit Hilfe von entsprechenden Konfigurationsdateien genauso erfolgen wie bei den einzelnen Beispielen spezifiziert. Insbesondere ist eine interaktive Eingabe von Parametern nicht zulässig. Sollte aufgrund von Änderungen am Konfigurationssystem die Ausführung der abgegebenen Dateien mit den Testdaten fehlschlagen, wird das Beispiel mit 0 Punkten bewertet. Die Konfigurationsdateien liegen im JSON-Format vor, zu deren Auswertung steht Ihnen die Klasse `Config` zur Verfügung. Die Verwendung der Klasse ist aus dem Programmgerüst ersichtlich.

Jede Konfigurationsdatei enthält zumindest einen Testfall und dessen Konfiguration. Es ist auch möglich, dass eine Konfigurationsdatei mehrere Testfälle enthält, um gemeinsame Parameter nicht mehrfach in verschiedenen Dateien spezifizieren zu müssen. In manchen Konfigurationsdateien finden sich auch einstellbare Parameter, die in Form eines `select` Feldes vorliegen. Diese sollen die Handhabung der Konfigurationsdateien erleichtern und ein einfaches Umschalten der Modi gewährleisten.

Es steht Ihnen frei, z.B. zu Testzwecken eigene Erweiterungen zu implementieren. Stellen Sie jedoch sicher, dass solche Erweiterungen in Ihrem abgegebenen Code deaktiviert sind, damit ein Vergleich der abgegebenen Arbeiten mit unserer Referenzimplementierung möglich ist.

Die Programmgerüste, die zur Verfügung gestellt werden, sind unmittelbar aus unserer Referenzimplementierung abgeleitet, indem nur jene Teile entfernt wurden, die dem Inhalt der Übung entsprechen. Die Verwendung dieser Gerüste ist nicht zwingend, aber Sie ersparen sich sehr viel Arbeit, wenn Sie davon Gebrauch machen.