Ein rudimentärer Raytracer soll implementiert werden.

1 Vorbereitung

- Erstellen Sie ein neues Verzeichnis für diese Aufgabe.
- Kopieren die Quelltexte zu `geometry` und `math` des zweiten Übungsblatts in das Verzeichnis.
- Kopieren Sie auch das cmake-File des zweiten Übungsblatts in das Verzeichnis.
- Laden Sie aus Ilias das Programmfragment `raytracer.zip` herunter und entpacken Sie es in dem Verzeichnis. Es ist lediglich eine Datei `raytracer.cc` mit Kommentaren, Erklärungen und der `main`-Methode.
- Ergänzen Sie das cmake-File um ein neues Executable für `raytracer.cc`.

2 Raytracer

Implementieren Sie einen Raytracer mit den folgenden Features, die Sie auf jeden Fall alle erfüllen müssen:

- Beachten Sie die Quelltextkommentare in `raytracer.cc`.
- Verwenden Sie immer, wenn möglich, die Datenstrukturen und Funktionalitäten aus `math` und `geometry` wieder.
- Als grafische Primitive müssen entweder Kugeln (am einfachsten) oder Dreiecke (schwieriger) verwendet werden. Ein entsprechender Schnittpunkt-Algorithmus befindet sich jeweils in `geometry`.
- Es muss mindestens eine punktförmige Lichtquelle implementiert werden.
- Als Shading muss Lambertian-Shading implementiert werden. Bei Dreiecken muss die Oberflächenormale des Schnittpunkts mit Hilfe der uv-Parameter aus den Normalen der Eckpunkte interpoliert werden.
- Reflexion muss implementiert werden.
- Schatten müssen implementiert werden.
- Als Testszene müssen Sie eine Variante der Cornelbox rendern. Die zugehörige Szene kann im Quelltext fest definiert sein.
- Zur Ausgabe des Bildes können Sie entweder eine PPM-Datei erzeugen, welche mit einem externen Bildanzeigeprogramm dargestellt wird, oder mit SDL2. In jedem Fall sollte dies in einer eigenen Klasse verborgen bleiben.
- Es dürfen keine Speicherlecks auftreten. Benutzen Sie möglichst kein new/delete. Höchstens Smart-Pointer. Es reicht für die Cornelbox aus, die Objekte im statische Speicherbereich oder auf dem Stack zu erzeugen.
- Verwenden Sie keine Programmiersprachemkonstrukte von C, die durch neue bei C++ ersetzt werden, z.B. kein printf, alloc, free, C-Castoperator, ...

Noch einige Hinweise:

- Gehen Sie schrittweise vor: zuerst nur Raycasting implementieren. Dann ein Feature nach dem anderen.
- Sie können noch mehr Funktionalitäten einbauen, z.B. Transmission. Anti-Alias. Texturen. Alles auf eigene Gefahr. Verzetteln Sie sich nicht.
- Beachten Sie die Quelltextkommentare in `raytracer.cc`. Sie geben Hinweise auf Datenstrukturen und zugehörige Funktionen.
- Sie können alles in der Datei `raytracer.cc` implementieren. Falls diese zu komplex wird, können Sie Teile in eigenen Dateien auslagern (cmake-File entsprechend anpassen).
- Versuchen Sie nicht Ihre Implementierung auf Geschwindigkeit zu optimieren. Es lohnt sich nicht.