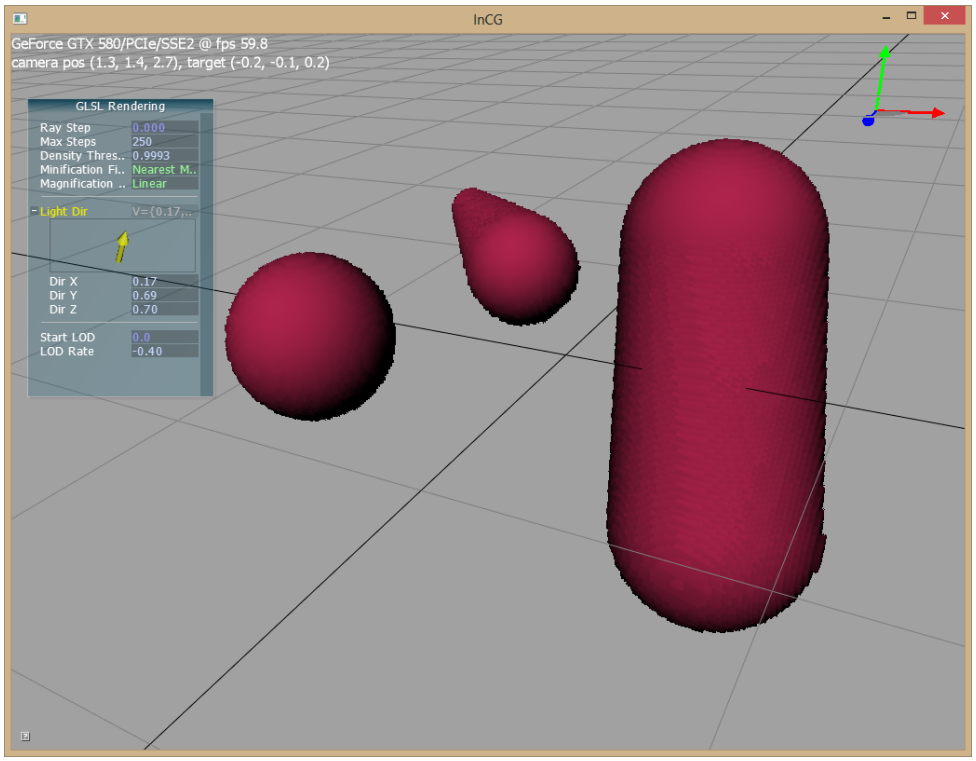
Karlsruher Institut für Technologie
Lehrstuhl für Computergrafik

Prof. Dr.-Ing. Carsten Dachsbacher Dipl.-Inf. Florian Simon

4. Übungsblatt zur Vorlesung Interaktive Computergrafik im SS 2014

Besprechung am Mittwoch, 28.05.2014



Diese Aufgabenblatt behandelt Voxelisierungen als Datenstruktur für approximatives Ray-Tracing. Das entsprechende Programmskelett finden Sie erneut auf der Vorlesungswebsite. Am besten kompilieren Sie zunächst die Anwendung und machen sich mit dem Programm vertraut. Im Auslieferungszustand erzeugt das Programm einen Volumendatensatz und stellt dessen Hüllquader dar. Der Datensatz enthält für jeden Voxel eine Normale in den `xyz`-Komponenten sowie einen Dichtewert in der `w`-Komponente. Ein Voxel soll als opak gelten, wenn der Dichtewert den globalen Schwellwert `g_threshold` überschreitet.

Aufgabe 1 *Ray-Marching im Fragment-Shader*

Implementieren Sie im Fragment-Shader `RayCast.fp.glsl` Ray-Marching! Laufen Sie dazu in einer Schleife mit `maxSteps` Iterationen entlang des Strahls, der durch die Variablen `ray_pos` und `ray_dir` gegeben ist. Die Schrittweite entlang des Strahls soll `dt` sein. Die Schleife soll abgebrochen werden, sobald sich die aktuelle Position außerhalb des Einheitswürfels befindet, d.h. wenn $x \notin [0,1]^3$. Greifen Sie in jeder Iteration auf die Textur `tVolume` zu und prüfen Sie, ob der aktuelle Voxel opak ist. Ist dies der Fall, können Sie die Schleife vorzeitig abbrechen, ansonsten bewegen Sie sich weiter entlang des Strahls. Geben Sie am Ende der Schleife aus, ob sie einen Treffer gefunden haben, indem Sie für die beiden Fälle unterschiedliche Werte in `gl_FragColor` schreiben.

Aufgabe 2 *Shading*

Implementieren Sie Lambertsches Shading für die gefundenen Schnittpunkte. Verwenden Sie dazu die Richtung zur (direktionalen) Lichtquelle `lightDir`. Beachten Sie, dass Sie die Komponenten der Normale aus der Volumentextur (wie auch bei Normal-Maps üblich) aus dem Bereich $[0, 1]$ in den Bereich $[-1, 1]$ transformieren müssen. Nachdem Sie die schattierte Farbe in `gl_FragColor` geschrieben haben, verwerfen Sie schließlich noch Fragmente, für die kein Schnittpunkt gefunden wurde (Schlüsselwort: `discard`).
Bonusaufgabe: Überlegen Sie sich, wie Sie neben der Fragmentfarbe auch die korrekte Tiefe ausgeben können.
Bonusaufgabe: Verschießen Sie einen Schattenstrahl in Richtung der Lichtquelle und berechnen Shading nur dort, wo die Lichtquelle sichtbar ist.

Aufgabe 3 *Mip-Mapping und Level of Detail*

In der letzten Aufgabe soll ein einfaches Level-Of-Detail-Schema implementiert werden, indem entlang des Strahls auf verschiedene Mip-Map-Stufen zugegriffen wird. Verändern Sie den Fragment-Shader so, dass am Eintrittspunkt in das Volumen auf das Mip-Map-Level `lod_start` zugegriffen wird und dieses entlang des Strahls mit `lod_rate` linear verändert wird. Experimentieren Sie mit verschiedenen Einstellungen für diese Parameter; Sie können auch verschiedene Texturfilterungsarten wählen.