Shadow Mapping

Im Tarball `cg-2021-a05.tar.bz2` finden Sie eine erweiterte Variante des Codes der letzten OpenGL-Aufgabe der Sie in dieser Übung Schatten mittels Shadow Mapping hinzufügen sollen.

Überblick.
Im Vergleich zu den letzten Übungen hat sich der Code des Renderloops deutlich vergrößert. Wie schon in der Vorlesung beschrieben können Sie einen kleinen Modellhubschrauber in der Szene fliegen lassen und durch das daran montierte Spotlight die Szene (anstelle des über UI-Regler positionierten Punktlichts) individuell beleuchten.

Achtung, alte Shader wurden nicht 'umgerüstet'. Auf neue Code-Passagen wird in den folgenden Aufgaben eingegangen.

Aufgabe 1

Machen Sie sich mit der Steuerung des Hubschraubers vertraut (im Abschnitt eingefasst von `update_timer` sowie der Deklaration der `heli_cam` und `heli_mat`). Welche Tasten haben welche Funktion? (Für Vim-Nutzer sollte das etwas naheliegender sein). Im Sinne der Vorlesung zu Transformationen, versuchen Sie nachzuvollziehen wie die inkrementelle Modifikation der `heli_mat` funktioniert.

Überlegen Sie ebenfalls welche impliziten Annahmen in der ursprünglichen View-Matrix sowie der Animation der Rotoren bezüglich des Modells stecken.

Als Übung zu Transformationen können Sie gerne weitere Bewegungsoptionen hinzufügen (auch wenn das mit Shadow Mapping erstmal nichts zu tun hat).

Aufgabe 2

Wie Sie aus der Vorlesung wissen muss die Shadowmap vor der Verwendung in jedem Frame neu gerendert werden. Das findet in der Vorlage in dem Abschnitt statt, der über den Timer `render_sm` eingefasst ist. Hier ist zunächst nichts zu tun.

Im Render-Teil der Ihnen schon aus den vorigen Aufgaben bekannt ist werden nun zwei weitere Texturen als Uniform übergeben. Die angegebenen Indizes bezeichnen die Textureinheiten (das wurde in der Vorlesung nicht behandelt, sie können im der entsprechenden `Shader::uniform` Methode nachschauen wie das funktioniert, wenn Sie möchten).

Fügen Sie hier die benötigten Matrizen ein um für beide Shadow Maps die korrekte Parameterisierung für den Shadow Lookup zu berechnen.

Damit ist auf C++-Seite alles nötige erledigt um die Shadow Maps und Matritzen im Shader zu verwenden. In `shaders/shadows.frag` muss nur noch der Aufruf der `shadow_coeff` Funktion korrekt aufgesetzt werden. Die Funktion erwaret als Parameter die Matrix die eine Fragmentposition in Weltkoordinaten in die NDC der entsprechenden Shadow Map übersetzt. Weiterhin muss eben diese Funktion, `shadow_coeff`, implementiert werden. Bestimmen Sie dazu die korrekte NDC-Koordinate aus der entsprechenden Fragmentshader-Inputvariablen, transformieren Sie das Ergebnis in den Textur-Raum und führen Sie den Schatten-Test mit der passenden `texture` Funktion von GLSL aus.

Aufgabe 3

Wenn Sie die vorige Aufgabe korrekt implementiert haben werden die verschatteten Teile der Szene als klar im Schatten zu erkennen sein, auf den eigentlich beleuchteten Oberflächen werden Sie aber Shadow-Acne wahrnehmen. Wenden Sie die in der Vorlesung vorgestellten Lösungsstrategien an um die Probleme (großteils) zu lösen.

Aufgabe 4

Nun sollte klar erkennbar sein, welche Oberflächen im Schatten bzw. beleuchtet sind, die Auflösung der Schadow Map (insb. der des direktionalen Lichts) sollte aber klar an den Schattenkanten sichtbar werden. Ebenso sollte es Ihnen einfach möglich sein das Problem der gegengleichen Fragmentgrößen nachzustellen (Vorlesung 'Resolution mismatch'). Prüfen Sie das und aktivieren Sie dann bilineare Filterung des Schatten-Tests via Texture-Parameter (hier muss nur der Parameter im schon vorhandenen Aufruf angepasst werden).

Können Sie das in der Vorlesung beschriebene Problem immernoch reproduzieren?

Happy Hacking :)