Karlsruher Institut für Technologie
Lehrstuhl für Computergrafik

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

3. Übungsblatt zur Vorlesung Interaktive Computergrafik im SS 2014

Besprechung am Mittwoch, 21.05.2014

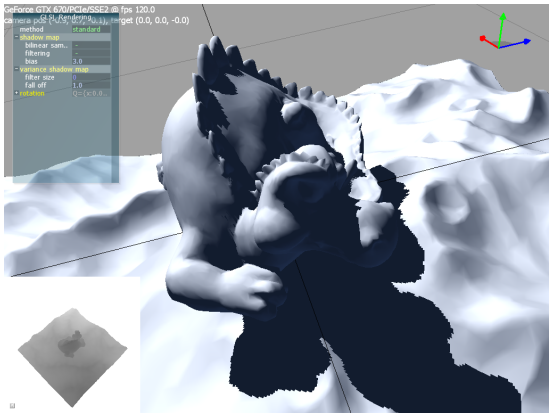

Abbildung 1: Normale Schadow-Maps

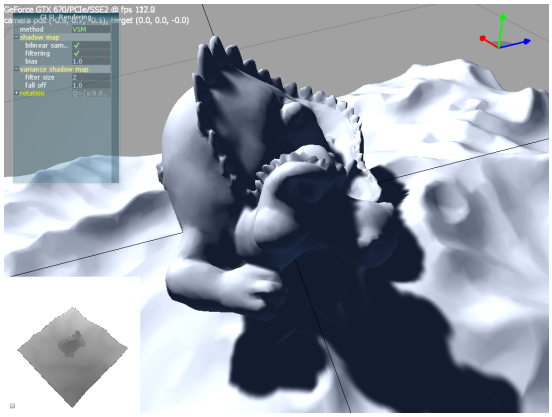

Abbildung 2: Variance Shadow-Maps

Diese Aufgabenblatt behandelt Shadow-Mapping, das gängigste Echtzeitverfahren zur Schattenberechnung. Das entsprechende Programmskelett finden Sie erneut auf der Vorlesungswebsite. Am besten kompilieren Sie zunächst die Anwendung und machen sich mit dem Programm vertraut.
Die Anwendung ist so programmiert, dass Sie den Inhalt der Shadow-Map durch die Visualisierung in der linken unteren Ecke direkt überprüfen können, während Sie die Shader implementieren.

Aufgabe 1 *Erzeugung der Shadow-Map*

Beginnen Sie mit der Implementierung des Shadow-Mapping-Algorithmus, indem Sie zunächst im Vertex-Shader `createSM.vp.glsl` die Tiefe berechnen und in der `out`-Variablen `depth` speichern. Dabei sollen Sie nicht die z-Tiefe verwenden, sondern explizit die Vertex-Position von Modell- in Weltkoordinaten transformieren und den Abstand zur Lichtquelle ermitteln. Im Fragment-Shader `createSM.fp.glsl` geben Sie danach die interpolierte Tiefe aus. Verwenden Sie dazu den Rotkanal von `out_color`.
Sie sollten in der linken unteren Ecke des Bildschirms die Shadow-Map als Tiefenkarte sehen können. Desto weiter ein Punkt von der Lichtquelle entfernt ist, desto heller ist er. *Hinweis:* Sie können die Lichtquelle mittels AntTweakBar-GUI drehen. In dieser Aufgabe wird die Shadow-Map auch auf dem Modell in der Mitte dargestellt.

Aufgabe 2 *Auswertung der Shadow-Map*

Ändern Sie das Shader-Program `perpixel_lighting.(v|f)p.glsl` so ab, dass der Algorithmus vervollständigt wird.
Transformieren Sie dazu zunächst im Vertex-Shader den Vertex in das Koordinatensystem der Lichtquelle und geben die Texturkoordinaten für die Shadow-Map in der `out`-Variablen `texCoordSM` aus. Beachten Sie, dass nach der Projektion (durch die virtuelle Kamera der Lichtquelle) die xy-Koordinaten im Intervall $[-1; 1]^2$ liegen, Texturkoordianten aber im Intervall $[0; 1]^2$ liegen müssen.
Führen sie schließlich im Fragment-Shader den Tiefentest durch, indem Sie die Tiefe aus der Shadow-Map mit der tatsächlichen Fragment-Tiefe vergleichen. Setzen Sie entsprechend des Ergebnisses die Variable `lit` auf 0 oder 1. Nun sollten Sie die Szene mit korrekten Schatten wie in Abbildung 1 sehen können. *Hinweis:* Vergessen Sie nicht, die Darstellung der Shadow-Map am Ende des Fragment-Shaders zu entfernen!

Aufgabe 3 *Variance-Shadow-Mapping*

In dieser Aufgabe sollen Sie Variance-Shadow-Mapping (VSM), eine Erweiterung des Standardverfahrens, implementieren.
Dazu müssen Sie zunächst im Fragment-Shader `createSM.fp.glsl` neben der Tiefe auch das Quadrat der Tiefe in der Shadow-Map speichern. Es bietet sich an, dafür den Grünkanal der Ausgabevariablen `out_color` zu verwenden, während die Tiefe (weiterhin) im Rotkanal gespeichert wird. *Hinweis:* Sie können optional die Visualisierung in `viewSM.fp.glsl` so abändern, dass Sie beide Komponenten der Shadow Map sehen.
Werten Sie als nächstes im Fragment-Shader `perpixel_lighting.fp.glsl` die Chebychev-Ungleichung aus. Die Ungleichung dient als Schätzung des verschatteten Anteils der Oberfläche und ist gegeben durch:

$$P \leq \frac{\sigma^2}{\sigma^2 + (t - \mu)^2},$$

wobei μ und σ^2 die rekonstruierte Teifenverteilung repräsentieren und t die tatsächliche Fragment-Tiefe ist. Die Tiefenverteilung berechnen Sie mit:

$$\begin{aligned} \mu &= E(x), \\ \sigma^2 &= E(x^2) - E(x)^2, \end{aligned}$$

wobei $E(x)$ und $E(x^2)$ die Werte aus der Variance-Shadow-Map sind. Beachten Sie, dass diese Ungleichung nur für $t > \mu$ gilt, d.h. wenn der Tiefentest fehlschlägt. Weisen Sie in diesem Fall der Variablen `lit` den Wert P zu. Für $t <= \mu$ liegt das Fragment, wie beim ursprünglichen Shadow-Mapping, nicht im Schatten und Sie können den Wert von `lit` unverändert lassen.

Aufgabe 4 *Filterung der Shadow-Map*

Ein großer Vorteil des VSM-Verfahrens ist, dass die Shadow-Map gefiltert werden darf. Zum einen kann man dass ausnutzen, indem man Hardware-unterstützt mit trilinearer Filterung darauf zugreift. Zum anderen kann man zur Erzeugung weicher Schatten auch die Shadow-Map von Hand filtern.
In dieser Aufgabe sollen Sie einen Gauß-Filter implementieren. Der Einfachheit halber soll der Filter direkt bei der Auswertung der Shadow-Map in `perpixel_lighting.fp.glsl` berechnet werden.

Iterieren Sie hierzu zwei Variablen `i`, `j` über den Filterkern (im Intervall $[-r, r]^2$, wobei $r =$ `filterRadius` der Radius des Filterkerns ist), indem Sie zwei verschachtelte `for`-Schleifen schreiben. Im inneren Schleifenrumpf müssen Sie zunächst den Filterwert

$$w(i,j) = \exp(-\frac{i^2 + j^2}{r^2} \cdot \texttt{fallOff})$$

berechnen, um den zu lesenden Texturwert zu gewichten. Die passenden Texturkoordinaten zum Lesen der Textur erhalten Sie, indem Sie das Offset `vec2( float(i), float(j) ) / 1024` (1024 ist die Größe der Shadow-Map) verwenden. Akkumulieren Sie die gewichteten Texturwerte in einer Variablen, die Sie am Ende der Schleife(n) noch normalisieren müssen (*Hinweis:* Zum Normalisieren müssen Sie durch die Summe aller Gewichte teilen). Den (gefilterten) Wert dieser Variablen verwenden Sie dann wie bisher bei der Auswertung der Shadow-Map.
Wenn Sie alles richtig gemacht haben, sollten Sie nun eine weichere Schattenkante (ohne die üblichen Treppenstufenartefakte), wie in Abbildung 2 sehen können. Weiterhin sollten Sie an Stellen mit hoher Tiefenkomplexität *Light-Bleeding* ausmachen können: dieses Artefakt ist typisch für Variance-Shadow-Mapping und nur mit fortgeschrittenen Techniken in den Griff zu bekommen.