

186.140 Real Time Rendering, Hall of Fame - Abgabe 2

Bernhard Rainer 0828592

Andreas Gogel 0801243

26. Januar 2016

Übersicht

Man befindet sich in einem großen Halle, die dem Inneren einer Kirche ähnelt. In der Szene befinden sich mehrere Feuertöpfe, die die große Halle spärlich beleuchten. Durch die farbige Fenster fällt Sonnenlicht, welches den Raum in unterschiedlichen Farben zusätzlich erleuchtet und einen Kontrast zu dem sonst dunklen und düsteren Raum bietet. Die Feuertöpfe dienen sowohl als zusätzliche stimmungsvolle Objekte in der Szene, als auch als praktische Lichtquellen.

System

Die Applikation wurde vorwiegend auf AMD-Grafikkarten entwickelt. Das Programm wurde auf dem Abgaberechner auf Nvidia getestet und läuft auf diesem fehlerfrei.

Szene

Die Szene beruht auf dem Modell der Sibenik-Kathedrale [1]. Das Modell verfügt bereits über Farbtexture, sowie Normal-Maps. Zusätzlich wurde eine weitere BodenTexture, sowie mosaik Fenstertexturen eingefügt und die Fenster-Objekte vom Grundmodell getrennt. Das Model weist leider eine Löcher auf, die sich durch kleine weiße Flecken bemerkbar machen. Ein weiteres verwendetes Modell ein Blumentopf [2], der als Basis für die Feuertöpfe dient. Da dieses Modell über keine Normal-Map verfügt wurde aus der bestehenden Farbtextur eine Normal-Map generiert und hinzugefügt.

Deferred Rendering

Da sich mehrere Lichtquellen in der Szene befinden, empfiehlt es sich auf Deferred Rendering zurückzugreifen, um die Performance zu verbessern. Hierbei wird in einem ersten Rendering-Pass die Geometrie gerendert und Die Information über die Position, Normalen und Farbe auf Texturen gespeichert. In einem zweiten Rendering-Pass wird diese Information verwendet, um die Beleuchtung der

Szene zu berechnen. Somit muss nur für jedes Fragment einmal Beleuchtungsinformation berechnet werden. Die Applikation verwendet eine Kombination aus Deferred Rendering für solide Geometrie-Objekte und normale Forward Rendering für transparente Objekte.

Effekte

Bump Mapping

Sämtliche Modelle in der Szene verfügen über Normal-Maps, die die Oberflächennormalen in Tangent Space beschreiben. Diese Information wird verwendet, um vermeintlich glatten Oberflächen durch Modulation der Normalen eine rauhere Oberflächenstruktur zu simulieren. Hierzu wird zuerst aus der Normalen (in World Space), der Tangente und der Bitangente eine Transformationsmatrix aufgebaut, die die Normale aus der Texture in den World Space transformiert. Diese transformierte Normale wird dann zur Beleuchtungsberechnung verwendet.

Bloom

Ein Bloom-Effekt beschreibt eine Verstärkung von hohen Farbfrequenzen, sodass diese auf benachbarte Pixel überstrahlen. Hierbei wird mit einem Highpass-Filter die hohen Frequenzen der Farbe herausgefiltert und anschließend mit einem Gauss-Filter „geblurt“. Das Ergebnis dieser Operation wird in einem Postprocessing-Schritt auf das Bild aufaddiert. Als Vorlage für diesen Effekt dienen die Folien der Real Time Rendering-Vorlesung [3].

Screen Space Ambient Occlusion

Screen Space Ambient Occlusion ist ein Postprocess-Effekt, der mit Hilfe von räumlicher Verdeckung eine Verschattung der Szene approximiert. Hierbei wird für um jeden Pixel des Bildes eine Anzahl an Samples aus dem Tiefenbuffer ausgewählt. Sind diese Pixel nahe beisammen und der Winkel entsprechend klein, wird das Pixel schattiert. Somit wird eine Verdunkelung in Ecken simuliert. Der verwendete Ansatz beruht auf dem Tutorial von John Chapman [4]. Hierzu wird für jedes Pixel der Tiefenwert mit Pixeln aus der Nachbarschaft verglichen. Die Nachbarschaft baut sich aus einer Halbkugel auf, die in Richtung der Normalen rotiert wird. Somit werden nur Werte verglichen, die sich „vor“ den Pixel befinden. Aus dieser Nachbarschaft werden zufällig Tiefenwerte entnommen. Ist dieser Tiefenwert geringer, trägt er zur Verdeckung des Pixels bei.

Animierte Texturen

Animierte Texturen befinden sich in der Szene in der Form von Feuer. Aus einer GIF-Datei werden die einzelnen Bilder als Texturen gespeichert. Anhand der vergangenen Zeit, Anzahl an Bildern und der Abspieldauer der Datei kann der aktuelle Frame berechnet werden. Dieser wird als normale Farbtextur verwendet. Da es sich in unserer Implementierung um einen Feuereffekt handelt, muss aus dem schwarzen Hintergrund des Videos ein Transparenzwert errechnet werden.

Omnidirectional Shadow Mapping

Um die Schatten der Punktlichter realistisch darzustellen wird Omnidirectional Shadow Mapping verwendet. Die Implementierung beruht auf dem, in der Real Time Rendering-Vorlesung vorgestellten Ansatz [5]. Pro Lichtquelle wird in entlang der Koordinatenachsen in positive und negative Richtung die Szene gerendert und die Distanzen zur Lichtquelle in einer Cubemap auf der entsprechenden Texture gespeichert. Im Beleuchtungs-Pass des Deferred Shadings werden dann die Distanz des Fragments zur Lichtquelle und der in der Cubemap gespeicherten Distanz verglichen und das Fragment dementsprechend beleuchtet.

Stochastic Shadow Mapping (Fehlerhaft)

Dieser Effekt wurde mittels einer stochastic shadow map [6] realisiert. Hierfür wurde für eine sich außerhalb der Kirche befindliche direktionale Lichtquelle, welche die Sonne darstellt, eine farbige Shadowmap erstellt. In diese wurden zuerst die Tiefenwerte aller Objekte, welche nicht transparent sind, in einer normalen Shadowmap abgelegt. Diese wird dann in eine Farbtextur gerendert um die Tiefenwerte auf allen Kanälen zur Verfügung zu haben. Als nächstes werden mit einer geeigneten Blendfunktion (`gl_one, gl_one`) die lichtdurchlässigen Objekte der Szene in diese Farbtextur gerendert. Hierbei ist zu beachten, dass die Farben als Materialmodell von 1 abgezogen werden. Dies beschreibt dann die Durchlässigkeit des Materials für die jeweiligen Farbkkanäle (Bsp: Rot = $(1,0,0)$ = i $(0,1,1)$, der Rotkanal ist voll durchlässig, die anderen beiden werden geblockt). Es wird nun verglichen, ob dieser kleiner als ein Zufallsvektor ist (Grund wird später behandelt). Das Resultat ergibt ein Boolvektor. Um diese Information in die Shadowmap zu speichern, muss dieser zuerst mit dem aktuellen Tiefenwert des Pixels verglichen werden da es sonst zu Artefakten im Ergebnisbild kommen würde. Die Farbe des Pixels ergibt sich also mit: $\max(\text{depth}, \text{boolvektor})$. Um jetzt zu verhindern, dass manche Farben gar nicht in die Textur geschrieben werden können, da diese nur kleine Werte beinhalten (z.B. Durchlässigkeit = $(0.2,0.1,0.2)$), kommt der Vergleich mit dem Zufallsvektor ins Spiel um dennoch Farbwerte für diese Pixel zu bekommen. Mit der resultierenden Shadowmap kann nun, wie bei einer normalen Shadowmap, bestimmt werden welche Pixel im Schatten sind und welche nicht, nur wird, statt den Schattenpixel einfach abzdunkeln, die Farbe aus der Map verwendet. (Es ist leider noch ein Fehler mit den Koordinaten enthalten. Wird aber bis zur Präsentation ausgebessert)

Steuerung

Sämtliche Effekte über die F-Tasten der Tastatur ein- und ausgeschalten werden.

- F1: Beleuchtungsmodell ändern
- F2: Bump Mapping
- F3: Bloom
- F4: SSAO
- F5: Transparente Objekte

- F6: Omnidirectional Shadow Mapping
- F7: Stochastic Shadow Mapping (standardmäßig ausgeschalten)

Die Applikation verfügt über 1st-Person Kamera, mit der sich der User frei in der Szene bewegen kann. Die WASD-Tasten bewegen die Kamera relativ zur Blickrichtung. Is die rechte Maustaste gedrückt, kann die Kamera rotiert werden. Alternativ können auch die Pfeiltasten verwendet werden. Mit der ESC-Taste wird die Applikation geschlossen

Literatur

- [1] Sibenik cathedral. <http://hdri.cgtechniques.com/~sibenik2/download/>.
- [2] Jardiniere. <http://www.turbosquid.com/3d-models/free-jardiniere-3d-model/845875/>.
- [3] Screen space effects, pages 7-11. http://www.cg.tuwien.ac.at/courses/Realtime/slides/VU.WS.2014/12Screenspace_effects.pdf.
- [4] Ssao tutorial. <http://john-chapman-graphics.blogspot.co.at/2013/01/ssao-tutorial.html>.
- [5] Omni-directional shadows. <https://www.cg.tuwien.ac.at/~husky/RTR/OmnidirShadows-whyCaps.pdf/>.
- [6] Morgan McGuire and Eric Enderton. Colored stochastic shadow maps. February 2011.