REAL TIME HIGH DYNAMIC RANGE IMAGE BASED LIGHTING MIT DEFERRED SHADING

Michael Horsch
Hochschule Darmstadt
Fachbereich Informatik

1 Abstract

Diese Arbeit beschreibt Real Time High Dynamic Range Image Based Lighting mit Deferred Shading, ein Verfahren zum Zeichnen von Objekten unter der Beleuchtung einer Environment Map. Im Gegensatz zu vorherigen Verfahren wird die Environment Map durch eine hohe Zahl von Lichtquellen angenähert um *dynamische Objekte in Echtzeit bei interaktiven Frameraten* beleuchten zu können. Durch den Einsatz von Deferred Shading eignet sich das Verfahren besonders gut für die Implementierung auf programmierbarer Grafik-Hardware.

2 Einleitung

In vielen Gebieten wird eine realitätsnahe Beleuchtung von Computer-generierten Objekten benötigt. Vor allem bei Produktvisualisierungen und Filmen ist es zudem nötig die Beleuchtungsverhältnisse einer echten Umgebung nachzuempfinden, z.B. um Produkt-Designs zu testen oder um am Computer erstellte Schauspieler in eine echte Szene einzufügen. Hierfür ist es notwendig zwei Probleme zu lösen, nämlich die Repräsentation der Umgebung und die Rekonstruktion der Beleuchtung durch die Umgebung. Während die Umgebung relativ einfach mit Bildern repräsentiert werden kann, ist die Lösung zur Rekonstruktion der Beleuchtung mit einem sehr hohen Rechenaufwand verbunden. Die Berechnungen müssen daher für Echtzeit-Anwendungen vereinfacht werden.
Diese Arbeit beschreibt wie die Beleuchtungsberechnungen mit Deferred Shading optimiert werden können um die Beleuchtung einer Umgebung mit möglichst vielen Lichtquellen annähern zu können. Dies dient wiederum dazu ein möglichst realistisches Bild zu erzeugen.

Abbildung 1. Ein mit Real Time High Dynamic Range Image Based Lighting mit Deferred Shading berechnetes Bild

3 Stand der Technik

Beleuchtungsverfahren haben eine lange Geschichte in der Computer Graphik. Eine Vielzahl an Arbeiten wurden bereits zu diesem Thema verfasst. Schon früh wurde erkannt, dass man Bilder zur Beleuchtung benutzen kann um künstliche Objekte in echte Szenen einzufügen. Besonders die Filmindustrie zeigte Interesse an solchen Techniken und nutzte sie für revolutionäre Effekte. Mittlerweile gehört die Bild-basierte Beleuchtung zum Standard bei Filmeffekten und durch die immer größer werdende Rechenleistung moderner Computer wird das Konzept zunehmend auch in interaktiven Anwendungen benutzt. Der folgende Abschnitt stellt eine kleine Auswahl der Arbeiten zu diesem Thema vor.

3.1 Repräsentation der Umgebung

Für die Repräsentation einer Umgebung werden sog. Environment-Maps benutzt[1]. Eine Environment-

Map ist einfach ein Bild der gesamten Umgebung eines bestimmten Punktes. D.h. jeder Pixel repräsentiert das Licht das aus einer bestimmten Richtung auf diesen Punkt trifft. Hierfür ist es nötig, dass in der Environment-Map die realen Intensitäten der Lichter gespeichert werden um auch die realen Beleuchtungsverhältnisse rekonstruieren zu können. Normale Bildformate mit 8 Bit pro Farbkanal sind nicht Präzise genug um die hohen Dynamikumfänge echter Umgebungen zu speichern. Daher wurden spezielle Dateiformate entwickelt, die dieser Anforderung genügen. Die populärsten sind das .HDR-Format von Greg Wards Radiance Rendering System[2] und das .EXR-Format von Industrial Light and Magic[3]. Das .HDR-Format kodiert die Farben mit 32-Bit pro Pixel, wobei die Farbe mit 24-Bit und ein Exponent zur Skalierung der Helligkeit mit 8-Bit gespeichert wird. Durch dieses Format können jedoch nicht alle vom Menschen wahrnehmbaren Farben dargestellt werden. Das .EXR-Format speichert die Farben mit 16 Bit Fließkommazahlen pro Farbkanal. Um den benötigten Speicherplatz zu reduzieren werden die Daten daher komprimiert.

3.2 Rekonstruktion der Beleuchtung

Die erste bekannte Verwendung von Environment-Maps zur Beleuchtung von Objekten geht auf Jim Blinn zurück[1]. Er zeigte, wie spiegelnde Reflexionen mit Environment-Maps erzeugt werden können.
Die Beleuchtung von Objekten mit HDR-Environment-Maps wurde maßgeblich von Paul Debevec mitenwickelt[4a und 4b]. Von ihm stammen viele Arbeiten über Bild-basierte Beleuchtung. Seit er das Konzept vorgestellt hat entstanden verschiedene Techniken, von denen sich jedoch die meisten nicht für Echtzeit-Anwendungen eignen.
Eine Möglichkeit ist, den Lichttransport vorzuberechnen[5]. Dieser Weg hat jedoch viele Nachteile. Die vorberechneten Daten benötigen z.B. sehr viel Speicherplatz und müssen daher komprimiert werden. Außerdem müssen alle Objekte statisch sein. Es können zwar dynamische Objekte beleuchtet werden, aber selber beeinflussen sie die Beleuchtung nicht. Der Vorteil solcher Techniken ist die sehr hohe Qualität der Beleuchtung bei geringen Rechenzeiten.
Es gibt aber auch eine relativ einfache Methode die selbst auf älterer Grafik-Hardware mit interaktiven Frameraten läuft. Dabei werden für die Environment-Map zwei weitere Environment-Maps vorberechnet, die jeweils nur die diffuse bzw. spekulare Beleuchtung enthalten. Diese neuen Environment-Maps werden dann als Look-Up-Table für die Beleuchtung aus einer bestimmten Richtung benutzt[6]. Der klare Nachteil ist, dass Schatten für die einzelnen Lichtquellen mit dieser Methode nicht berechnet werden können.
Um den Rechenaufwand möglichst gering zu halten wird auch versucht, nur die hellsten Lichtquellen aus der Environment-Map zu extrahieren[7]. Dabei kann es aber vorkommen, dass kleine Lichtquellen nicht gefunden werden. Die indirekte Beleuchtung der Umgebung wird hierbei nicht berücksichtigt.

3.3 Deferred Shading

Der Rechenaufwand kann weiterhin mit Deferred Shading reduziert werden. Beim Deferred Shading werden vom Betrachter aus die für die Beleuchtungsberechnung benötigten Daten in einem sogenannten G-Buffer gespeichert. Der G-Buffer besteht aus mehreren Texturen, in denen jeweils die einzelnen Daten gespeichert sind. So könnte der G-Buffer beispielsweise aus einer Textur für die Normalen, einer Textur für die Positionen und einer Textur für die Farben der Oberflächen zusammengesetzt sein. Um die Beleuchtung zu berechnen muss nur noch ein über den Bildschirm gestrecktes Rechteck gezeichnet werden. Die Beleuchtungsberechnungen können dann mit Hilfe der im G-Buffer enthaltenen Texturen in einem Shader ausgeführt werden.
Diese Vorgehensweise hat verschiedene Vorteile: Die Beleuchtungsberechnungen sind im Gegensatz zum normalen Forward Shading nicht mehr von der Komplexität der Szene abhängig und die Beleuchtungsberechnungen werden in konstanter Zeit durchgeführt. Die Beleuchtung kann für alle Pixel parallel berechnet werden, dadurch eignet sich die Technik perfekt für den Einsatz auf programmierbarer Grafik-Hardware. Da die Beleuchtungsberechnungen mit Deferred Shading wesentlich beschleunigt werden, können viel mehr Lichtquelle simuliert werden als ohne Deferred Shading.
Die Technik hat jedoch auch Nachteile. So können aufgrund der Arbeitsweise keine durchsichtigen Materialien simuliert werden. Objekte die solche Materialien benutzen müssen auf herkömmliche Weise gezeichnet und zum Schluß mit den restli-

chen Objekten kombiniert werden. Unterschiedliche Beleuchtungsmodelle sind auch problematisch, da ja für alle Pixel die gleichen Berechnungen durchgeführt werden. Hierfür sind Verzweigungen im Shader-Programm nötig, welche aber nicht von jeder Hardware unterstützt werden. Als Nachteil wird auch oft genannt, dass unterschiedliche Materialien ebenso problematisch wie unterschiedliche Beleuchtungsmodelle seien. In den meisten Fällen unterscheiden sich die Materialien jedoch nur durch Farbe, Oberflächenstruktur und den Parametern für die Beleuchtungsberechnung. D.h. die meisten Materialien können parameterisiert werden und mit den Inhalten des G-Buffers beschrieben werden. Dadurch sind im Shader-Code keine Verzweigungen nötig und es können viele unterschiedliche Materialien benutzt werden.
Der letzte und größte Nachteil ist, dass der Speicherverbrauch durch die Texturen größer ist als beim Forward Shading. Bei komplexen Geometrien kann es daher vorkommen, dass Daten auf dem Hauptspeicher ausgelagert werden müssen, was zu erheblichen Leistungsverlusten führt.

4 Real Time High Dynamic Range Image Based Lighting mit Deferred Shading

Der folgende Abschnitt beschreibt einen Ansatz, wie das Zeichnen von Objekten unter der Beleuchtung einer Enivronment-Map effizient auf programmierbarer Grafik-Hardware implementiert werden kann. Erst durch die Leistung und den neuen Fähigkeiten aktueller Grafik-Hardware ist es möglich, realistische Ergebnisse bei interaktiven Frameraten zu erzielen.

4.1 Konzept

Die Beleuchtung durch die Umgebung soll mit möglichst vielen Lichtquellen angenähert werden. Im besten Fall würde man für jeden Pixel in der Environment-Map eine Lichtquelle erzeugen. Selbst bei niedrigen Auflösungen der Environment-Maps würden jedoch so viele Lichtquellen erzeugt werden, dass eine Berechnung der Beleuchtung bei interaktiven Frameraten nicht mehr möglich wäre. Stattdessen wird eine konstante Anzahl an Lichtquellen erzeugt und gleichmäßg auf der Environment-Map verteilt. Die Lichtquellen werden außerdem nur auf der oberen Hemisphäre der Environment-Map verteilt, da Objekte meistens auf einer Ebene stehen, die die untere Hemispäre verdeckt.
Für jede Lichtquelle werden auch die Schatten berechnet. Da die Berechnung von Schatten ein sehr umfassendes Thema mit seinen eigenen Problemen ist, wird hier nicht weiter darauf eingegangen.

4.2 Ableitungen von der Rendering-Gleichung

Die Rekonstruktion der Beleuchtung lässt sich auf die Lösung der Rendering-Gleichung zurückführen. Die Rendering-Gleichung berechnet die Strahlungsdichte *L*, die von einem Punkt x in eine Richtung $\vec{\omega}$ abgestrahlt wird.

$$L(x,\vec{\omega}) = L_e(x,\vec{\omega}) + \int_\Omega f_r(x,\vec{\omega_e},\vec{\omega}) L(x,\vec{\omega_e})(\vec{\omega_e}\cdot\vec{n}) d\vec{\omega_e} \quad (1)$$

Mit:

- $L_e(x,\vec{\omega})$ = Emissionsterm, die eigene Leuchtkraft des Punktes
- $f_r(x,\vec{\omega_e},\vec{\omega})$ = Bidirektionale Reflektanzverteilungsfunktion (BRDF)
- $L(x,\vec{\omega_e})$ = Strahlungsdichte aus der Richtung $\vec{\omega_e}$
- $(\vec{\omega_e}\cdot\vec{n})$ = Cosinus-Gewichtung
- $\vec{\omega_e}$ = Einfalls-Vektor (Richtung von der Lichtquelle zum Punkt *x*)
- $\vec{n}$ = Normale des Punktes
- Ω = Gesamtheit aller Winkel der Hemisphäre über der Oberfläche

Damit diese Berechnungen bei interaktiven Frameraten ausgeführt werden können, müssen sie erst vereinfacht werden. Der Emissionsterm wird zuerst ignoriert, da die meisten Materialien nicht selber leuchten. Die Berechnung des Integrals kann auf digitaler Hardware nicht ausgeführt werden. Deshalb ist die wichtigste Vereinfachung die Umwandlung des Integrals in eine diskrete Summierung. Dazu wird nur noch die Beleuchtung der zuvor verteilten Lichtquellen aufsummiert, und nicht mehr die Beleuchtung aus der

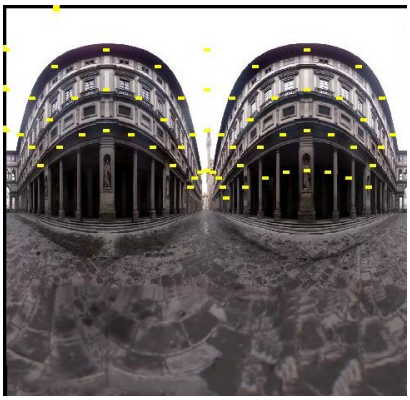

Abbildung 2. Visualisierung der Verteilung der Lichtquellen

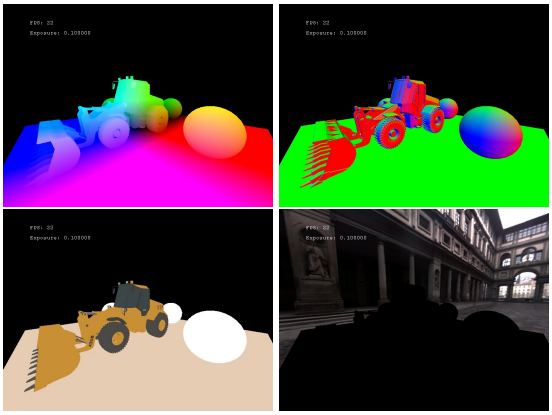

Abbildung 3. Die im G-Buffer gespeicherten Texturen: Positionen, Normalen, Farben und Umgebung

gesamten Hemisphäre. Die vereinfachte Rendering-Gleichung lautet dann:

$$L(x,\vec{\omega}) = \sum_{i=0}^{N} f_r(x,\vec{\omega_e[i]},\vec{\omega}) L(x,\vec{\omega_e[i]})(\vec{\omega_e[i]} \cdot \vec{n}) \tag{2}$$

Wobei N die Anzahl der Lichtquellen ist. Um die Rendering-Gleichung möglichst genau anzunähern, muss N also möglichst groß gewählt werden. Der Index i steht für die i-te Lichtquelle.

4.3 Verteilung der Lichtquellen

Die Lichtquellen werden gleichmäßig auf der oberen Hemisphäre der Environment-Map verteilt. Die Intensität einer Lichtquellen ergibt sich aus der Farbe des Pixels an der Stelle der Lichtquelle im Koordinatensystem der Environment-Map. Um gleichmäßig verteilte Richtungen zu erzeugen werden Kugelkoordinaten benutzt.

4.4 Deferred Shading

Für eine effiziente Berechnung der Beleuchtung wird Deferred Shading benutzt. Alle Berechnungen werden mit 16-Bit Fließkomma-Zahlen durchgeführt. Der G-Buffer besteht aus 4 RGBA-Texturen. Die ersten drei Texturen speichern die Positionen, die Normalen und die Farbe der sichtbaren Punkte. Im Alpha-Kanal der Farb-Textur wird die Reflektivität gespeichert und im Alpha-Kanal der Normalen-Textur die Glossiness, ein Maß für die Schärfe der Reflexionen. So gibt es bereits 3 Grundmaterialeigenschaften die frei miteinander kombiniert werden können um viele unterschiedliche Materialien zu erzeugen. In der vierten Textur wird die Umgebung gespeichert. Da die Umgebung selber nicht nochmal beleuchtet werden soll wird sie nach den Beleuchtungsberechnungen zum Rest der Szene hinzugefügt.
Die Beleuchtung kann jetzt mit Hilfe des G-Buffers auf einem über den Bildschirm gestreckten Quader in einem Shader berechnet werden. Theoretisch könnten alle Lichtquellen in einem Durchgang berechnet werden. Weil aber jede Lichtquelle eine Shadow Map benötigt und die Anzahl der aktiven Texturen von der Grafik-Hardware begrenzt wird, werden nur 4 Lichtquellen bei jedem Deferred Shading Durchgang berechnet. Die Ergebnisse der einzelnen Durchgänge werden durch additives Blending kombiniert.

4.5 High Dynamic Range und Tone Mapping

In der Natur kommen viele unterschiedliche Typen von Lichtquellen vor, von der Sonne bis zu einem kleinen LED-Lämpchen. Die hohen Kontrastunterschiede können mit einem niedrigen Dynamikbereich nicht nachempfunden werden. Alle Beleuchtungsberechnungen werden daher mit einem hohen Dynamikbereich ausgeführt. Nur so ist es möglich die Beleuchtungsverhältnisse realistisch zu rekonstruieren. Weil die Environment-Maps und der G-Buffer mit 64-Bit gespeichert wird, werden die Berechnungen ebenfalls mit 64-Bit durchgeführt.
Moderne PC-Bildschirme können nur Bilder mit niedrigem Dynamikbereich anzeigen. Um das mit hohem Dynamikbereich berechnete Bild korrekt anzeigen zu können muss es deshalb erst in ein Bild mit niedrigem Dynamikbereich umgewandelt werden. Den Vorgang der Umwandlung bezeichnet man als Tone Mapping.

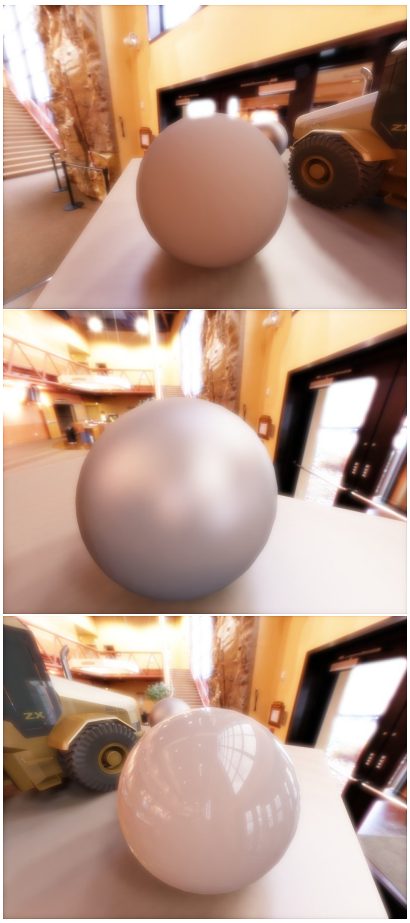

Abbildung 4. Materialien mit den drei Grundeigenschaften. Von Oben nach Unten: Diffuse Farbe, Glossiness, Reflektivität

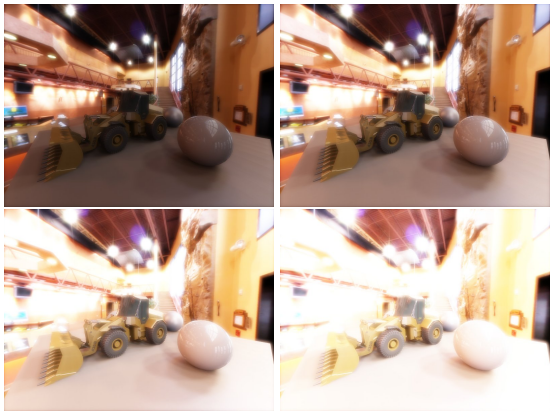

Abbildung 5. Simulation unterschiedlicher Belichtungszeiten

Die Rechenvorschrift zur Umwandlung ist der Tone Mapping Operator.
Als Tone Mapping Operator wurde die Exposure-Funktion von Hugo Elias benutzt[8]. Diese Funktion nähert die physikalischen Vorgänge bei der Fotografie mit einer einfachen Funktion an, die einfach und schnell zu berechnen ist:

$$C_t = 1.0 - e^{-E*C_o} \tag{3}$$

C_t ist dabei die Ergebnis-Farbe nach dem Tone Mapping, C_o ist die Original-Farbe mit hohem Dynamikbereich, e entspricht der Eulerschen Konstante. E ist ein Faktor für die Belichtungszeit, je höher der Wert, desto heller wird das Bild. Um Details in dunklen Bereichen sichtbar zu machen wird also ein hoher Wert benutzt, für Details in hellen Bereichen ein niedriger Wert.

4.6 Post Processing

Um die Objekte besser in die Szene einzufügen wird ein sogenannter Glow-Effekt in einem Post Processing Schritt hinzugefügt. Der Glow-Effekt sorgt dafür, dass helle Lichtquellen die Kanten von Objekten überstrahlen. Dieser Effekt tritt auch bei der menschlichen Warhnehmung auf, ist in der Fotografie aber eigentlich ein Artefakt der Technik der Kameras. Da dieses Artefakt jedoch bei jeder normalen Kamera auftritt (und die Environment-Maps mit Kameras aufgenommen werden), versucht man es auch in der Computer Grafik zu simulieren. Für einen Glow-Effekt wird eine Kopie des Bildes weichgezeichnet und wieder zum Original dazuaddiert. Durch die häufige Verwendung

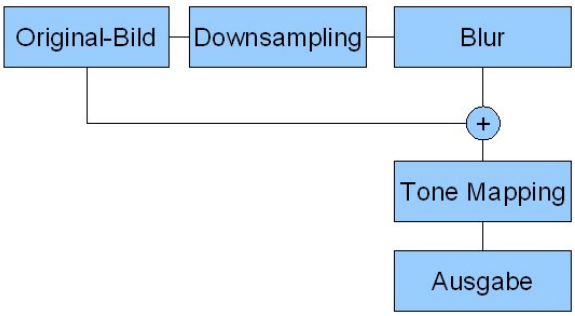

Abbildung 6. Die Post Processing Pipeline

wird der Glow-Effekt fälschlicherweise von Laien oft mit HDR-Rendering gleichgesetzt.
Die Implementierung des Effekts fängt mit dem Bild der fertig beleuchteten Szene an. Da die Bilder sehr hohe Auflösungen besitzen, wäre das Weichzeichnen bei diesen Auflösungen zu rechenaufwändig. Daher wird das Bild zuerst in mehreren Schritten herunterskaliert. Erst danach wird das Bild weichgezeichnet. Dies könnte mit einer großen ($\geq$5x5) Filtermatrix geschehen, kostet aber zu viel Leistung. Deshalb wird der Filter auf zwei Durchgänge aufgeteilt. Im ersten Durchgang wird die Weichzeichnung nur Vertikal ausgeführt, im zweiten Durchgang dann Horizontal auf das Bild aus dem ersten Durchgang. Mit Hilfe dieser Aufteilung können große Filtermatrizen effizient berechnet werden. Für den Glow-Effekt wird das weichgezeichnete Bild vor dem Tone Mapping zum Ausgangsbild dazuaddiert.

4.7 Anti-Aliasing

Die Bildqualität wird mit Anti-Aliasing verbessert. Dies ist nötig, da sich sonst die virtuellen Objekte durch die Aliasing-Artefakte von der Umgebung abheben würden.
Multi-Sampling-Anti-Alisasing (MSAA) wird zwar von jeder aktuellen Grafik-Hardware unterstützt, kann aber nicht in Kombination mit Deferred Shading benutzt werden. Der Grund hierfür ist, dass durch die Filterung die im G-Buffer enthaltenen Daten verfälscht werden würden.
Als Alternative zu MSAA wird deshalb Super-Sampling-Anti-Aliasing (SSAA) verwendet. Hierbei werden die Berechnung mit einer höheren Auflösung als der des Ergebnis-Bildes durchgeführt. Das hochaufgelöste Bild wird dann auf die Größe des Ergebnis-Bildes herunterskaliert. Durch die Skalierung werden die Kanten gefiltert und die Aliasing-Artefakte können reduziert werden. So wird z.B. das Bild bei 2-fachem SSAA in der doppelten Auflösung berechnet.
Das SSAA kostet allerdings Leistung und Speicher. Auch wird die Anti-Aliasing-Qualität durch die Grafik-Hardware eingeschränkt, da es für die maximale Auflösung von Texturen Begrenzungen gibt. Diese Einschränkung kann man aber umgehen, indem das Bild aufgeteilt wird. Die Bildqualität kann aber auch schon mit 2-fachem SSAA erheblich verbessert werden.

Abbildung 7. Oben: Ohne Anti-Aliasing. Unten: Mit 2-fachem SSAA.

5 Ergebnisse

Die vorgestellte Technik wurde in C++ mit der OpenGL API implementiert. Als Testobjekte wurden

ein Bagger mit ca. 65.000 Polygonen, ein Alfa mit ca. 700.000 Polygonen und ein Audi TT mit fast 1.1 Millionen Polygonen benutzt. Die Leistungstest wurden auf einer nVidia GeForce 8800 GTS mit 512 MB VRAM durchgeführt. Die folgende Tabelle zeigt die Leistungs-Ergebnisse bei 64 Lichtquellen:

Objekt	Polygone	Auflösung	SSAA	FPS
Bagger	65.000	800x600	Aus	22
Alfa	700.000	800x600	Aus	5
Audi TT	1.100.000	800x600	Aus	3
Bagger	65.000	1280x1024	Aus	13
Alfa	700.000	1280x1024	Aus	5
Audi TT	1.100.000	1280x1024	Aus	3
Bagger	65.000	800x600	An	10
Alfa	700.000	800x600	An	4
Audi TT	1.100.000	800x600	An	4
Bagger	65.000	1280x1024	An	4
Alfa	700.000	1280x1024	An	2
Audi TT	1.100.000	1280x1024	An	2

6 Zusammenfassung und Ausblick

In dieser Arbeit wurde gezeigt wie Real Time High Dynamic Range Image Based Lighting mit Deferred Shading realisiert werden kann. Diese Technik erlaubt die Rekonstruktion der Beleuchtung durch eine Umgebung mit einer hohen Zahl von Lichtquellen. So können dynamische Objekte in Echtzeit bei interaktiven Frameraten in reale Umgebungen eingefügt.
Um die Lichtquellen besser zu verteilen oder um die Anzahl der Lichtquellen zu reduzieren könnte die in [7] vorgestellte Technik benutzt werden. Zur Zeit kann es aufgrund der gleichmäßigen Verteilung der Lichtquellen passieren, dass kleine helle Lichtquellen nicht erfasst werden. Auch Lichtquellen mit einer großen Oberfläche machen noch Probleme, da sie ja über Punktlichtquellen angenähert werden müssen.
Der Code kann an vielen Stellen noch optimiert werden. So ist es z.B. nicht nötig den gesamten G-Buffer mit 16-Bit zu speichern, diese hohe Präzision wird eigentlich nur bei der Hintergrund-Umgebung benötigt. Anstatt der Positionen würde es auch genügen, nur die Tiefenwerte zu speichern.

7 Literaturverzeichnis

1 Jim Blinn : „Texture and reflection in computer generated images“, Communications of the ACM Vol. 19, No. 10 (October 1976), 542-547.

2 Greg Ward : Radiance .HDR File Format, Radiance Synthetic Imaging System, online erhältlich auf: „http://radsite.lbl.gov/radiance/“

3 Industrial Light and Magic : OpenEXR File Format, http://www.openexr.com/

4a Paul Debevec: „http://www.debevec.org“

4b Paul Debevec: „Rendering Sythetic Objects into Real Scenes“, SIGGRAPH 98, online erhältlich auf: „http://debevec.org/Research/IBL/“, Juli 1998

5 Ravi Ramamoorthi et.al.: „All-Frequency Shadows Using Non-linear Wavelet Lighting Approximation“, online erhältlich auf: „http://graphics.stanford.edu/papers/allfreq/“

6 Masaki Kawase : „rthdribl“, Real-Time High Dynamic Range Image Based Lighting Demo, erhältlich auf http://www.daionet.gr.jp/masa/

7 Paul Debevec: „A Median Cut Algorithm for Light Probe Sampling“, Abstract online erhältlich auf: „http://gl.ict.usc.edu/Research/MedianCut/“

8 Hugo Elias: „The Exposure-Function“, online erhältlich auf: „http://freespace.virgin.net/hugo.elias/ graphics/x_posure.htm“

8 Bilder

Abbildung 8. Unterschiedliche Beleuchtungsverhältnisse durch unterschiedliche Umbegungen.

Abbildung 9. Das Alfa-Modell in unterschiedlichen Umgebungen.

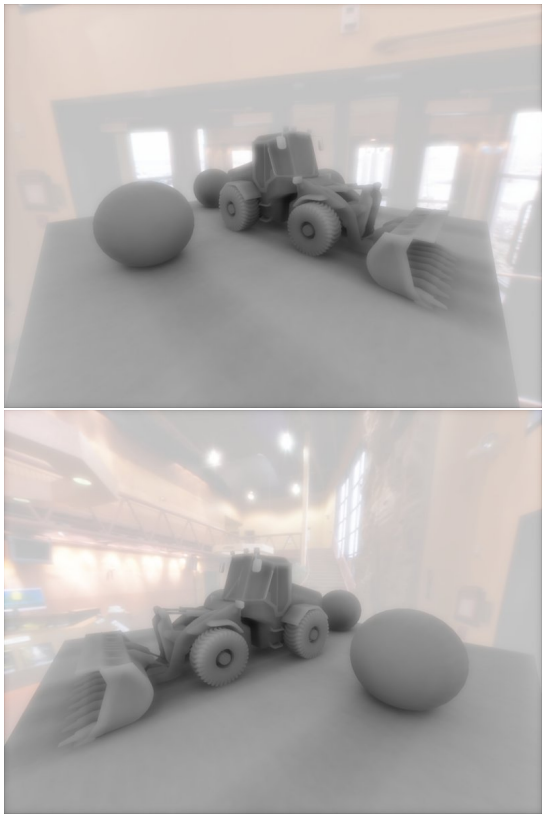

Abbildung 10. Ambient Occlusion.

Abbildung 11. Das Audi-TT-Modell.

Abbildung 12. Oben: Ohne SSAA. Unten: Mit SSAA.