entwickler.press
shortcuts

# Erfolgreiche Spieleentwicklung

## Moderne Licht- und Schattenspiele mit OpenGL

Alexander Rudolph

*Alexander Rudolph*

# Erfolgreiche Spieleentwicklung

## Moderne Licht- und Schattenspiele mit OpenGL

ISBN: 978-3-86802-513-2

Ein Imprint der Software & Support Media GmbH

# 1 Post Processing – Revolution in der Spielewelt

Alle in diesem Shortcut behandelten Programmbeispiele finden Sie hier zum Download:

*http://www.graphics-and-physics-framework.spieleprogrammierung.net/*

*http://www.space-combat-and-strategy.spieleprogrammierung.net/*

In den letzten Jahren hat sich im Bereich der Echtzeit-3-D-Grafik eine regelrechte Revolution vollzogen, denn die Art und Weise, wie eine 3-D-Szene aufgebaut und beleuchtet wird, hat sich grundlegend verändert. In der klassischen Szenendarstellung, dem so genannten „Forward Rendering“, erfolgte die Beleuchtung der einzelnen 3-D-Objekte stets zeitgleich mit ihrer Positionierung in der Spielewelt (Transform and Lighting). Auch wenn sich diese Vorgehensweise bereits in unzähligen Spielen bewährt hat, sind die Grenzen dieser Technik längst erreicht. Zum einen ist die Implementierung von realistisch wirkenden Beleuchtungsmodellen (globale Beleuchtung; global Illumination) nicht möglich, und zum anderen werden viel zu viele unnötige Beleuchtungs- und Schattenberechnungen durchgeführt. Denn nicht wenige der zuvor beleuchteten Oberflächen sind am Ende der Szenendarstellung vollständig oder zumindest teilweise verdeckt. Diese Verschwendung von Rechenleistung führt ihrerseits dazu, dass nur eine sehr begrenzte Anzahl von Lichtquellen in einer 3-D-Szene platziert werden können.

In modernen 3-D-Anwendungen und -Spielen werden die Beleuchtungsberechnungen zunehmend getrennt von der Geometriedarstellung während der Post-Processing-Phase durchgeführt, wodurch sich die zuvor beschriebenen Probleme des Forward Renderings auf elegante Art und Weise beheben

lassen. Mithilfe der Geometrie- und Farbinformationen einer 3-D-Szene, die man vor der eigentlichen Umsetzung der gewünschten Post-Processing-Effekte zunächst in diversen Render Targets (Render-Zielen, Texturen) zwischenspeichert, lassen sich Beleuchtungsberechnungen für nicht sichtbare Szenenpixel gezielt vermeiden. Die hierdurch frei werdenden Ressourcen können wir nutzen, um die für die globale Beleuchtung verantwortlichen Wechselwirkungen zwischen dem Licht und den Szenenpixeln unter Berücksichtigung der zuvor gespeicherten Geometrieinformationen zu simulieren, und um die Spielewelt mit einer deutlich größeren Anzahl von Lichtquellen auszuleuchten. Nachfolgend sind einige der Post-Processing-Effekte aufgelistet, die in modernen Computerspielen standardmäßig zum Einsatz kommen:

- Screen Space Ambient Occlusion (SSAO: Simulation der indirekten Beleuchtung, ambientes Licht)
- Schattenberechnungen
- Deferred Lighting: Simulation der direkten Beleuchtung (direktionale Lichtquellen sowie Punktlichtquellen)
- Nebelberechnungen
- Depth-Of-Field-Effekt (Schärfentiefe)

Schritt für Schritt wird das zunächst noch unbeleuchtete Szenenbild mithilfe dieser Effekte nachbearbeitet. Zum Speichern des modifizierten Szenenbilds nutzen wir eine Textur (ein zusätzliches Render Target), die ihrerseits auf ein bildschirmfüllendes Vertex Quad (Screen Quad) gemappt wird. Bildlich gesprochen entspricht diese Textur einer bemalten Leinwand, die wir über einen Holzrahmen (unser Vertex Quad) spannen.

## HDR-Rendering, Tone Mapping, Helligkeitsadaption und Blend-Effekte

Computerspielen wie Far Cry und Half Life 2 haben wir es zu verdanken, dass Begriffe wie HDR-Rendering (HDR: High Dynamic Range) und Post Processing mittlerweile zum festen Wortschatz in der Gamer-Community gehören. Um nun die Bedeutung dieser neuen Technologien verstehen zu können, müssen wir uns mit der folgenden Frage auseinandersetzen:

**Wie viele Farben (Farbwerte) benötigt man, um eine 3-D-Szene möglichst wirklichkeitsgetreu auszuleuchten?**

Vor der Einführung des HDR-Renderings kamen bei der Bildsynthese so genannte „True-Color-Farbwerte“ zum Einsatz. Auf den ersten Blick würde man meinen, dass der True-Color-Farbraum mit seinen 16,78 Millionen Farben und seiner 24-Bit-Genauigkeit mehr als genug Farbwerte für eine realistische Beleuchtung beinhaltet, doch schlüsselt man die einzelnen Farben nach ihren Rot-, Grün- und Blau-Anteilen auf, so stehen für jeden Farbkanal lediglich 256 Farbabstufungen (8 Bit) zur Verfügung. Der größtmögliche Helligkeitswert, den ein Fragment-Shader-Programm in ein True Color Render Target (bzw. in eine RGBA8-Integer-Textur) schreiben kann, liegt bei 1.0, und der kleinstmögliche Helligkeitswert beträgt 0.0. Größere oder kleinere Farbwerte werden beim Schreibvorgang rigoros auf 1.0 bzw. 0.0 beschnitten (Clamping). Im Unterschied zum HDR-Rendering spricht man in diesem Zusammenhang vom LDR-Rendering (Low Dynamic Range), da hierbei lediglich ein Helligkeitsbereich von 0 bis 1 abgedeckt wird. Insbesondere, wenn sich die Sonne oder irgendeine andere intensive Lichtquelle im Blickfeld der Kamera befindet, erstrecken sich natürliche Helligkeitsschwankungen über ein Vielfaches des LDR-Wertebereichs. **Abbildung 1.1** zeigt HDR-Rendering am Beispiel der Atmosphärendarstellung.

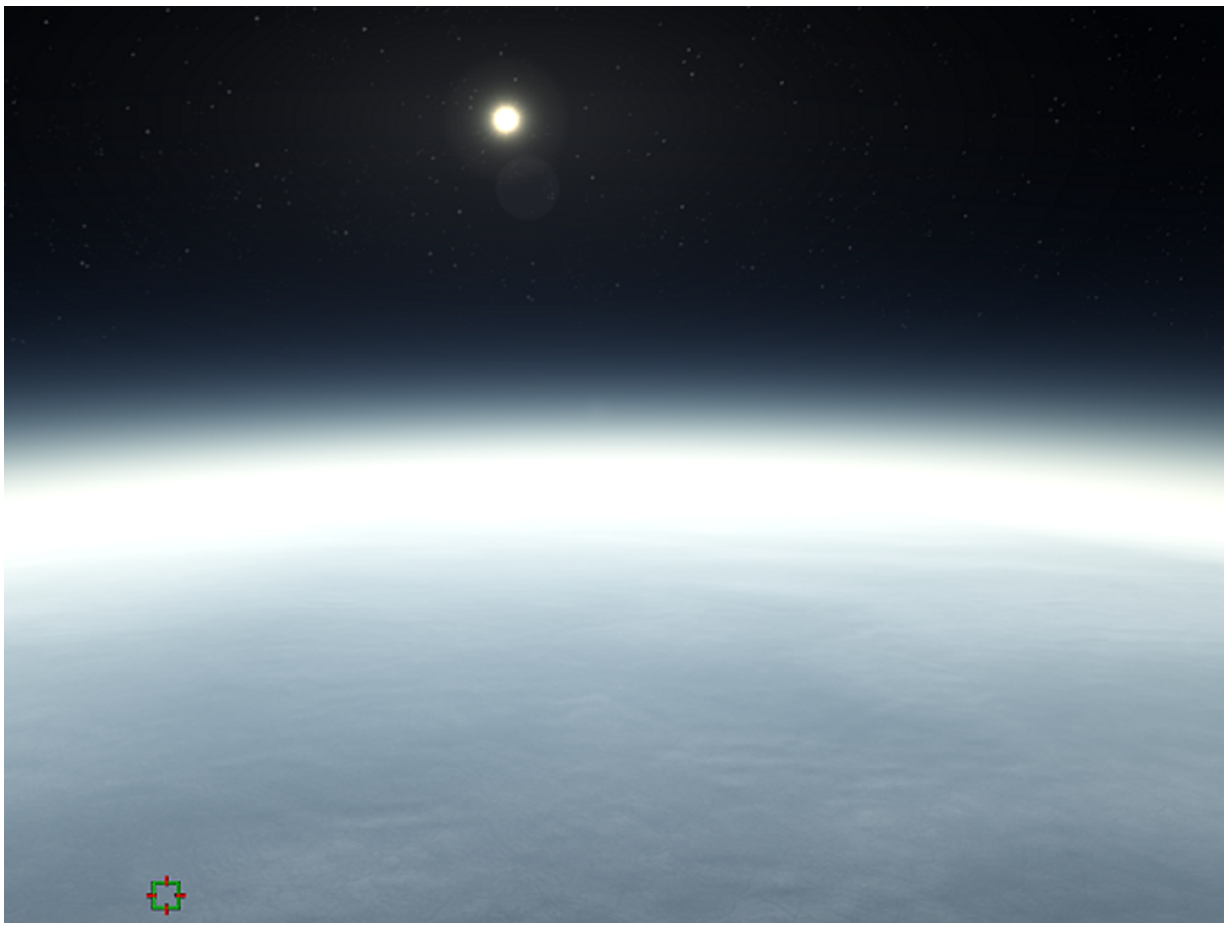

**Abb. 1.1:** HDR-Rendering am Beispiel der Atmosphärendarstellung

Um diese Helligkeitswerte speichern zu können, bedarf es der Verwendung von Floating Point Render Targets (bzw. Texturformaten), die pro Farbkanal 16-, 32- oder mehr Bit an Informationen und Farbabstufungen unbeschnitten (sowohl negative wie auch positive Werte > 1) speichern können:

- 16-Bit-RGBA-Floating-Point-Texturformat:

```
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA16F, width, height, 0,
GL_BGRA, GL_HALF_FLOAT, 0);
```

- 32-Bit-RGBA-Floating-Point-Texturformat:

```
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA32F, width, height, 0,
GL_BGRA, GL_FLOAT, 0);
```

- 16-Bit-RGB-Floating-Point-Texturformat:

```
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB16F, width, height, 0,
GL_BGR, GL_HALF_FLOAT, 0);
```

- 32-Bit-RGB-Floating-Point-Texturformat:

```
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB32F, width, height, 0,
GL_BGR, GL_FLOAT, 0);
```

Nun wird es etwas paradox, denn auf der einen Seite lassen sich realistisch wirkende Beleuchtungsberechnungen nur in Rahmen des HDR-Wertebereichs durchführen, auf der anderen Seite ist jedoch weder das menschliche Auge noch ein Computerbildschirm technisch bedingt dazu in der Lage, den zugehörigen Kontrastumfang (Dynamikbereich) auch komplett zu erfassen bzw. darzustellen. Die Umrechnung eines HDR-Szenenbilds nach Abschluss aller durchzuführenden Beleuchtungsberechnungen in ein LDR-Monitor-Bild wird als „Tone Mapping“ bezeichnet. Da bei dieser Umrechnung ein Großteil der HDR-Farbwerte verloren geht, muss man im Vorfeld zunächst anhand der Szenenhelligkeit ermitteln, auf welche Farbinformationen man verzichten kann und auf welche nicht. Mit dem Ziel, die Helligkeitsadaptionsfähigkeit des menschlichen Auges so gut wie möglich zu simulieren, führt das Tone Mapping bezogen auf die nachfolgenden Szenenbeispiele zu folgenden Ergebnissen:

- **Beispiel 1:** In einer sehr dunklen 3-D-Szene kann das menschliche Auge die Helligkeitsunterschiede von sehr hellen Objekten nur eingeschränkt wahrnehmen. Infolge dieser Einschränkung wirken helle Objekte nahezu gleich hell. Nach der Durchführung des Tone Mappings werden dunkle Farben mit großem Kontrast dargestellt, helle Farbnuancen lassen sich hingegen kaum bis gar nicht voneinander unterscheiden.