Komorebi - Ein Waldweg sonnenbeschienen

Simon Köhler, Johannes Wambach

Hochschule Darmstadt

Abstract

Landschaften, die mithilfe von Computer-Graphik gerendert werden, wirken trotz guter Schattierungen und sorgfältig hergestellten Texturen oft nicht räumlich und nicht sehr atmosphärisch. Für die Anwendung in Computer-Spielen, in Architekturmodellen oder anderen Simulationen wäre es gewinnbringend, dem Betrachter ein möglichst atemberaubendes Bild zu präsentieren. Eine geeignete Beleuchtung der Szenerie ist somit unerlässlich. Ein Mittel um atemberaubende Landschaftsbilder zu generieren, sind Lichtschafte - also Lichtstrahlen, die in einer natürlichen Umwelt durch andere Objekte gefiltert werden. In dieser Arbeit wird ein Konzept, die Realisierung und ein Ergebnis zur Umsetzung von Lichtschaften mit Standardtechnologien und Basisalgorithmen dargelegt.

I. Einleitung

Komorebi ist ein einzigartiger Begriff. Der Effekt, dass Sonnenschein in einer bewaldeten Gegend durch die Blätter scheint und Strahlen hinterlässt, hat nur in der japanischen Sprache einen so kurzen wohlklingenden Namen. Kürzer lässt sich demnach die im Rahmen des Praktikums gestellte Aufgabe nicht beschreiben. Das Ziel dieses Projekts ist, eine Simulation für das Durchscheinen von Licht durch Bäume oder Blätter zu bauen (s. Figure 1 [2]). Diese Arbeit beschreibt die Grundlagen, Realisierungsschritte und Ergebnisse in den folgenden Kapiteln:

I. Einleitung
II. Grundlagen - beleuchtet die Verfahren Billboard Volumetrics, Post-Processing und Volumetric Lighting
III. Konzept - beschreibt den vollständigen Rendering-Prozess
IV. Realisierung - erläutert die verwendeten Technologien und Realisierungsschritte
V. Ergebnisse - beschreibt die Resultate des Projekts mit Bildern
VI. Zusammenfassung und Ausblick - skizziert mögliche weitere Entwicklungsschritte und angrenzende Fachgebiete und Forschungsergebnisse

Die projektinitiierende Planung beinhaltete somit die Umsetzung einer kleinen Allee-Szene, die mit einem einfachen Beleuchtungsmodell beleuchtet wird und in der Lichtschafte mittels des Post-Processing Verfahrens (s. Grundlagen) gerendert werden. Zusätzlich bestand das optionale Ziel, Nebeleffekte zu implementieren und weitere Verfahren umzusetzen und somit visuell vergleichen zu können.

II. Grundlagen

Die umzusetzenden Lichtstrahlen werden in der meist englischsprachigen Literatur mit vielen Begriffen beschrieben. Vorkommende Begriffe sind Lichtschafte (light-shafts), god-rays, Strahlenbüschel, Lichtbüschel, Wolkenstrahlen oder (Gegen-)Dämmerungsstrahlen. Physikalisch lässt sich das Phänomen mit dem Tyndall-Effekt erklären, der Erscheinungen beschreibt, die

> *auf der Lichtstreuung in trüben Medien, vor allem in kolloidalen Lösungen, beruhen, wie Polarisation, Spektrum (Farbe) und Intensität des Streulichts; im engeren Sinn die Erscheinung, dass ein Lichtstrahl beim Durchgang durch ein trübes Medium infolge Streuung von der Seite sichtbar ist. [wissen.de, 2016]*

Im Verlauf der Projektarbeit wurde bei der Implementierung der Verfahren auch ein Spezialfall dieses Effekts umgesetzt, der den Tyndall-Effekt beziehungsweise eine leichte Reflexion im Auge simuliert (s. Figure 1 [2]). Dieser zweite Effekt wird vom Menschen weniger räumlich wahrgenommen und kann als Schimmern oder Flimmern beschrieben werden. Dagegen wird der Effekt der Lichtschafte eher räumlich wahrgenommen. Es scheint, als wären die Strahlen im Raum.

Die beschriebenen Effekte werden im Rahmen dieser Arbeit mit zwei unterschiedlichen Algorithmen

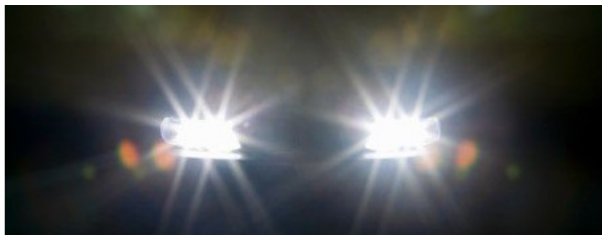

Figure 1: *Die Vision [1] - Scheinwerfer [2]*

umgesetzt. Zusätzlich wird ein weiteres einfaches Verfahren erläutert, dass jedoch nicht praktisch umgesetzt wurde.

I. Billboard Volumetrics

Die Visualisierung von Lichtschaften beziehungsweise direktional einfallenden Lichtstrahlen durch eine Tür, eine Luke oder andere Spalte mithilfe von Billboards ist ein sehr einfaches Konzept. Das Billbaord-Verfahren ist in der Computer-Grafik und in Echtzeit-Anwendungen weit verbreitet und basiert auf zweidimensionalen Rechtecken. Diese Rechtecke (Billboards) werden texturiert, in x-y-Ebene platziert und rotieren abhängig von der Kamera auf der y-Achse. Dabei steht das Billboard immer orthogonal zum Betrachter und wirkt dementsprechend bei passender Texturierung dreidimensional. Bei einer gut modellierten Textur und mehrfachen Schichtung von transparenten Billboards kann somit auch der Eindruck von Lichtstrahlen oder Nebellichtern erzeugt werden. (s. Figure 2 [1])

II. Radial Blur [Mitchell 07]

Der Radial-Blur-Effekt bezeichnet eine zweimensionale Weichzeichen-Methode, die ausgehend von einem definierten Mittelpunkt direktional weichzeichnet (s. Figure 2 [2]). Bei regulären Farbigen Bildern entsteht ein Zoom-Effekt. Bei markant hellen Mittelpunkten entsteht ein strahlender Effekt.

III. Volumetric Lighting [Tóth 09]

Klassische Beleuchtungs- und Reflexionsmodelle werden auf Oberflächen von Objekten angewendet und können auch nur Aussagen über diese treffen. Der Tyndall-Effekt hingegen bezieht sich bei Sonnenstrahlen auf kleinste Objekte, wie zum Beispiel Wassertropfen, in der Luft. Beim Volumetric Lighting wird zusätzlich zu den Objekten im Raum auch die Streuung des Lichts von kleinsten Teilchen im Raum berücksichtigt. Wenn Sonnenstrahlen auf kleinste Teilchen treffen wird ein kleiner Teil des Lichts in verschiedene Richtungen gestreut. Die Streuung des Lichts in Richtung der Kamera wird *in-scattering* genannt.

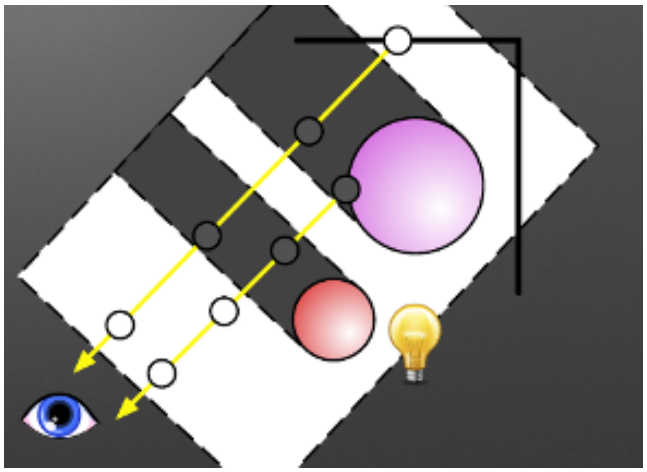

Figure 3: *Grundlegender Ray-Marching Algorithmus*

Für die Berechnung des Volumetric Lighting kann das in-scattering mithilfe von *Volume Ray-Marching*, oder kurz Ray-Marching, ermittelt werden. Dabei werden Sehstrahlen für jeden Pixel von der Kamera in die Szene geschickt und in gleichen Abständen *Samples*, Abtast-Punkte, betrachtet. An jedem Punkt wird nun die Beleuchtung (in-scattering) berechnet. Dabei sollte unter anderem berücksichtigt werden, ob sich der Abtast-Punkt im Schatten befindet. Anschließend werden die Werte der Samples eines Strahls addiert und gemittelt, um die endgültige Farbe des Pixels zu ermitteln. (s. Figure 3)

Figure 2: *Billboard Volumetrics [1], Radial Blur [2]: Originalbild und blurred bezüglich eines Mittelpunkts (rot)*

IV. Gaußscher Blur

Unterstützend für mehrere umgesetzte Verfahren wurde zudem der Gaußsche Weichzeichner (engl. Gaussian Blur) verwendet. Dieser einfache Weichzeichner mischt die Farben nebeneinander liegender Punkte beispielsweise mit einer 5x5-Matrix A_G. Für ein Standardweichzeichner gilt

$$A_G = \frac{1}{273} \times \begin{pmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{pmatrix}$$

Das verwendete Bild wirkt nach der Anwendung des gaußschen Weichzeichner verwaschen und ist unscharf. Der Weichzeichner kann jedoch in Bildern mit scharfen Kanten, die durch lineare Algorithmen entstanden sind, verwendet werden, um ein realistischeren Eindruck zu erzeugen. Verwendet wurde der Gaußsche Weichzeichner in dieser Arbeit beim Verfahren Radial Blur und in erweiterter Form beim Verfahren Volumetric Lighting.

III. Konzept

Wir haben Billboard Volumetrics nicht umgesetzt, da sie nicht mehr state-of-the-art sind. Außerdem sind sie eher für einzelne große Lichtschafte und weniger für viele kleine Lichtstrahlen durch Bäume und Blätter geeignet.

I. Radial Blur

Für die Erzeugung von Lichtschaften mithilfe des Radial Blur-Verfahrens sind zwei Operationen auf der Basis von zwei gerenderten Bildern nötig (s. Figure 4 [1] und [2]). Für das erste Eingangsbild wird die Szene ohne Phong-Shader und mit schwarzen Texturen gerendert. Zusätzlich ist der Hintergrund mit der Standardfarbe belegt und wird bei heller Farbgebung zum Horizont. Die Sonne wird als hellweiße Kugel gerendert. Diese Zusammenstellung wird festgehalten und als Basis für das Radial Blur-Verfahren genutzt (s. II - Radial Blur). Das Ergebnis dieser Operation ist ein dunkel schummriges Bild, das Verzerrungen in Richtung der Lichtquelle aufweist (s. Figure 4 [3]).

Als zweites Basisbild wird eine mit Phong und Texturen regulär gerenderte Szene verwendet. Das Ergebnis des verzerrten Bild wird additiv mit dem zweiten Basisbild verbunden (s. Figure 4 [4]). Auf diese Weise werden die Strahleneffekte in die farbige Szene eingebettet.

II. Volumetric Lighting

Im Gegensatz zum Radial Blur Verfahren wird beim Volumetric Lighting die Beleuchtung im 3D-Raum berechnet. In einem ersten Schritt wird die Szene mit Phong-Shading gerendert und dabei eine *depth-map* und eine *shadow-map* als Texturen erzeugt (s. Figure 5). Im zweiten Schritt wird nur mithilfe dieser zwei Texturen das Volumetric Lighting berechnet und in eine weitere Textur gerendert. Abschließend werden die beiden Schritte mithilfe von additivem Blending

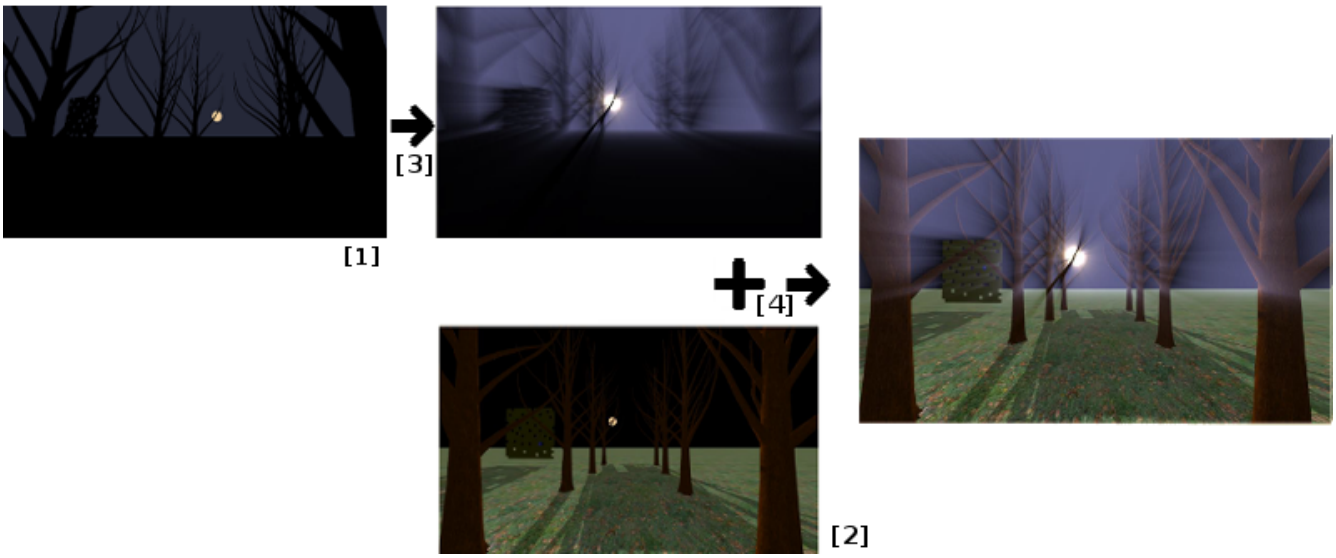

Figure 4: *Radial Blur-Verfahren*

zusammengeführt.

Eine depth-map ist eine Textur, die die Tiefeninformationen von jedem Pixel speichert. Die Tiefe ist hierbei die Distanz zwischen dem Pixel der Kamera und dem ersten Objekt in Kamerarichtung. Diese Textur wird beim Rendern eines Objektes mitberechnet und besitzt die gleiche Auflösung wie das Rendering-Fenster. Eine shadow-map ist eine depth-map aus der Sicht der Lichtquelle. Dazu werden die Objekte, anstatt mit der Kamera-Matrix, mit der Lichtquellen-Matrix transformiert. Als Lichtquelle dient die Sonne und damit ein Richtungslicht, das am Besten durch eine orthogonale Projektion beschrieben werden kann.

Figure 5: *Visualisierung Depth(oben) und Shadow-Map(unten)*

Diese beiden Texturen werden dem Volumetric Lighting Shader übergeben.

Für das Ray-Marching werden noch die Start- und Endpunkte der Strahlen benötigt. Diese werden mithilfe der depth-map und folgenden Gleichungen ermittelt:

$$WorldPos = CamPos + Dir * linearDepth \quad (1)$$

$$ViewDir = invViewProjMatrix * uv \quad (2)$$

$$linearDepth = \frac{(-f * n)/(f - n)}{depth - f/(f - n)} \quad (3)$$

Es werden also Strahlen von der Kamera aus in die Szene geschickt, in die Richtung in die die Kamera sieht. Diese Richtung *Dir* wird berechnet, indem die Texturkoordinaten *uv* mit der inversen Kamera-Projektionsmatrix multipliziert wird. Dabei ist uv ein 3D-Vektor bei dem die Z-Koordinate auf eins gesetzt wird. Die Länge des Strahls, *linearDepth*, in die Szene wird aus der depth-map entnommen. Dabei ist zu beachten, dass die in der depth-map gespeicherte Tiefe nicht linear ist. Die Genauigkeit der depth-map nimmt mit der Tiefe zu und daher resultiert eine einfache Visualisierung der Tiefenwerte auch in einer weißen Textur. Es wird also die linearisierte Tiefe der depth-map Werte benötigt, um die reale Position im 3D-Raum, *WorldPos*, zu rekonstruieren. Dabei werden die Tiefenwerte, *depth*, mit den Nahen- und Fernen Clip Plane, n und f, der Kameramatrix verrechnet.

Bei einem Richtungslicht ist der Startpunkt des Strahls gleich der Kameraposition, also der Tiefe null. Mit dem Start- und Endpunkt des Strahls wird nun der Ray-Marching Algorithmus angewendet, indem der Strahl in gleichmäßigen Abständen abgetastet wird. Bei jedem dieser Abtast-Punkte wird überprüft, ob er im Schatten liegt und entsprechend der in-scattering Term berechnet.

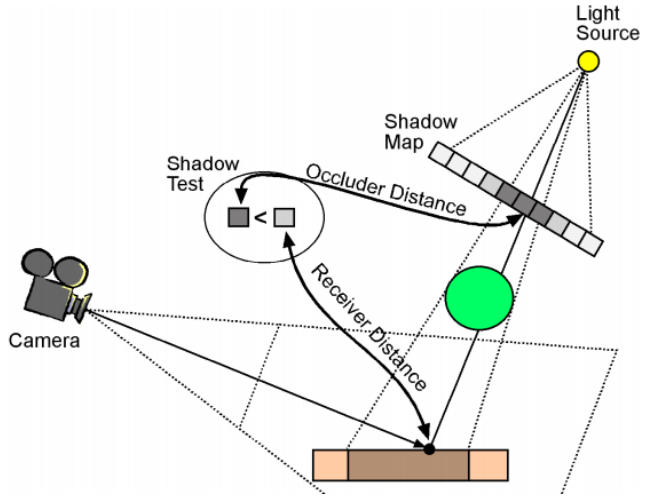

Figure 6: *Shadow-Mapping*

Der Schattentest wird mithilfe von Shadow-Mapping durchgeführt. Für den Test werden zwei Tiefenwerte aus Sicht der Lichtquelle benötigt. Der aktuelle Abtast-Punkt und seine 3D-Koordinaten werden also mit der Lichtquellenmatrix multipliziert. Der erste Wert für den Test entspricht nun der Z-Koordinate und entspricht der Distanz zwischen der Lichtquelle und dem aktuellen Abtast-Punkt. Der zweite Wert wird mithilfe der XY-Koordinaten und der Shadow-Map ermittelt. Durch Vergleichen der beiden Werte lässt sich nun ermitteln, ob ein Abtast-Punkt im Schatten liegt. (s. Figure 6)

Der letzte Schritt beim Ray-Marching ist nun noch den in-scattering Term zu berechnen und so die Beleuchtung im Raum zu ermitteln:

$$L_{in}(s,\theta) = \frac{\beta_R(\theta)+\beta_M(\theta)}{\beta_R+\beta_M} E_{sun}(1-F_{ex}) \tag{4}$$

$$\beta_M(\theta) = \frac{1}{4\pi}\beta_M \frac{(1-g)^2}{(1+g^2-2gcos(\theta))^{3/2}} \tag{5}$$

$$\beta_M = 0.434c\pi(\frac{2\pi}{\lambda})^{v-2}K \tag{6}$$

$$\beta_R(\theta) = \frac{3}{16\pi}\beta_R(1+cos^2(\theta)) \tag{7}$$

$$\beta_R = \frac{8\pi^3(n^2-1)^2}{3N\lambda^4}(\frac{6+3p_n}{6-7p_n}) \tag{8}$$

$$F_{ex}(s) = e^{-(\beta_R+\beta_M)s} \tag{9}$$

Die Formeln werden in [Preetham, 1999] [HP-Paper 02] beschrieben. L_{in} ist der zu berechnende in-scattering Term. E_{sun} ist die Intensität der Lichtquelle und wird entweder auf einen konstanten Wert gesetzt oder je nach Sonnenstellung variiert bzw. berechnet. F_{ex} beschreibt die Abschwächung des Lichts, also die Streuung die nicht beim Betrachter ankommt. ß$_M$ und ß$_R$ stehen jeweils für Mie- und Rayleigh-Streuung [RL 1871]. Die Mie-Streuung wird hierbei aus Performancegründen durch die Henyey-Greenstein Phasenfunktion [HG 41] angenähert. Während die Mie-Streuung den Fokus sehr stark in Vorwärtsrichtung hat, gibt es bei der Rayleigh-Streuung keinen großen Schwerpunkt. Der Hauptunterschied zwischen den beiden Streuungen ist, dass Rayleigh für kleinste Teilchen, z.b. Luftmoleküle, stattfindet, während Mie-Streuung für größere Teilchen anzuwenden ist (s. Figure 7).

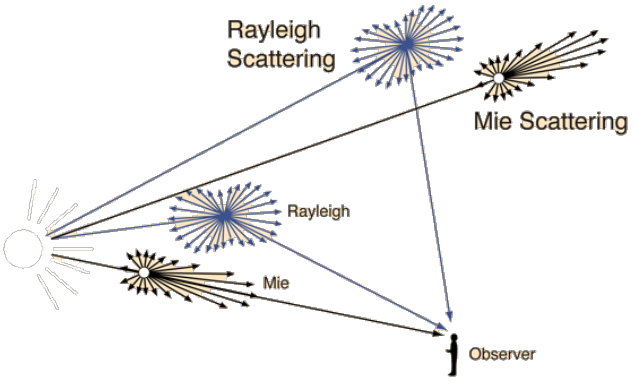

Figure 7: *Rayleigh- und Mie-Scattering*

IV. Realisierung

Das beschriebene Konzept wurde, um eine annehmbar gute Performanz zu ermöglichen, mithilfe von Standard-Technologien umgesetzt. Das Programm wurde mit der auf OpenGL aufsetzenden Bibliothek OpenTK implementiert. Diese Bibliothek ermöglicht die Verwendung von OpenGL-Methoden und -Berechnungen in der Programmiersprache C#. Die OpenGL-Bibliothek führt Graphik-Berechnungen

effizient auf dem dafür optimierten Graphikprozessor aus und wird in einer Vielzahl von Graphikanwendungen verwendet. Zusätzlich wurde OpenGL-Programmiersprache OpenGL Shading Language (GLSL) verwendet um performance-kritische Routinen auf dem Grafikprozessor auszuführen. Diese GLSL-Abschnitte sind zudem übertragbar und integrierbar in andere OpenGL-Grafikanwendung.

I. Radial Blur

Ein großes Problem des Radial Blur Verfahrens entsteht wenn die Kamerarichtung senkrecht zur Lichtquelle ist. Die Distanz zwischen den Samples wird sehr groß und somit werden die einzelnen Samples visuell unterscheidbar. Dies kann nur behoben werden, indem der Radial Blur Effekt immer schwächer wird desto größer der Winkel zwischen Kamerarichtung und Lichtquellenrichtung wird. Außerdem funktioniert das Verfahren nur korrekt solange die Lichtquelle für die Kamera sichtbar ist.

Besonders um das erste Problem weiter zu minimieren, wenden wir nach dem Radial Blur (Figure 4 [3]) noch einen Gaußschen Blur an. Das erzeugt zusätzlich auch weiche Lichtschafte, was für ein realistischeres Aussehen sorgt.

Der wichtigste Parameter ist die Anzahl der Samples mit dem die Qualität des Effekts kontrolliert werden kann. Die Quantität lässt sich mit *Exposure*, Intesität bzw. Helligkeit insgesamt, *Weight*, Intensität der einzelnen Samples, *Decay*, Abschwächung des Strahls, und *Density*, Länge und abhängig davon Helligkeit des Strahls, regulieren.

II. Volumetric Lighting

Da wir beim Shadow Mapping das Artefakt *Shadow Acne* nicht zufriedenstellend beseitigen konnten, mussten wir zusätzlich zu den klassischen Verfahren *(slope-scaled) depth-bias* noch einen *Normal Offset* [DH 11] hinzufügen.

Eine Performanceverbesserung bei Ray-Marching ist *Noise* und anschließenden Blur hinzuzufügen. Dadurch kann die Anzahl der Samples reduziert und dennoch eine gute Qualität erreicht werden. Der Noise wird als 2D-Textur vorberechnet und dann auf die Startpositionen der Strahlen angewendet. Nun haben wir die sichtbaren Samples gegen Noise ausgetauscht und um auch noch den Noise zu eliminieren benutzen wir einen bilateralen gaußschen blur filter [TM 98]. Der normale Gauß-Filter lässt das Bild an den Kanten von Objekten, bei großen Unterschieden der Tiefenwerte, verschwimmen. Mit dem bilateralen Blur bleiben die Objektkanten scharf und der Blur wird trotzdem auf die Lichtstrahlen angewendet. Beim bilateralen Blur wird die Tiefe (depth-map) des aktuellen Kernelpixels mit der Tiefe des zentralen Pixels verglichen und entsprechend gewichtet. (s. Figure 9)

Um die Lichtstrahlen hervorzuheben wird bei der Berechnung, während des Ray-Marching, beim Übergang zwischen Licht und Schatten dem Schatten ein höherer Einfluss als normal zugeordnet.

Abschließend haben wir noch die Berechnung des in-scattering so erweitert, dass wir atmospheric scattering nach [HP 02] [Preetham, 1999] berücksichtigen und somit auch die Beleuchtung des Himmels berechnen können.

Die Qualität bzw. Performance des Effekts lässt sich, wie schon beim Radial Blur Verfahren, mit der Anzahl der Samples regulieren. Weitere Parameter leiten sich aus den Formeln zur Berechnung des inscattering ab. Beispiele sind die Mie- und Reayleigh Koeffizienten und die Lichtquellenintensität E_{sun}.

V. Ergebnisse

In den Figures 8-10 präsentieren wir unsere Ergebnisse und geben Vergleichbilder der Techniken.

VI. Zusammenfassung und Ausblick

Komorebi wurde in dieser Arbeit in Form einer kleinen Baum-Szene mit Erfolg umgesetzt. Der Tyndall-Effekt und die Grundlagen zu den Implementierungsvarianten wurden beleuchtet und in ein durchgängiges Konzept integriert. Das Konzept wurde wie beschrieben und mit weiteren Optimierungen und Anpassungen mit C#, OpenTK und GLSL umgesetzt.

Das Radial Blur Verfahren ist in der Spielebranche, seit Crytek es im Spiel Crysis integriert hat [TS 08], das Standardverfahren für Lichtschafte. Doch mit Lords of the Fallen 2014 und Fallout4 2015 wird es wahrscheinlich in Zukunft von dem überlegenerem Verfahren des Volumetric Lighting ersetzt werden.

Während sich das Radial Blur Verfahren kaum erweitern bzw. (performance-)verbessern lässt, gibt es für Volumetric Lighting verschiedenste Ansätze der

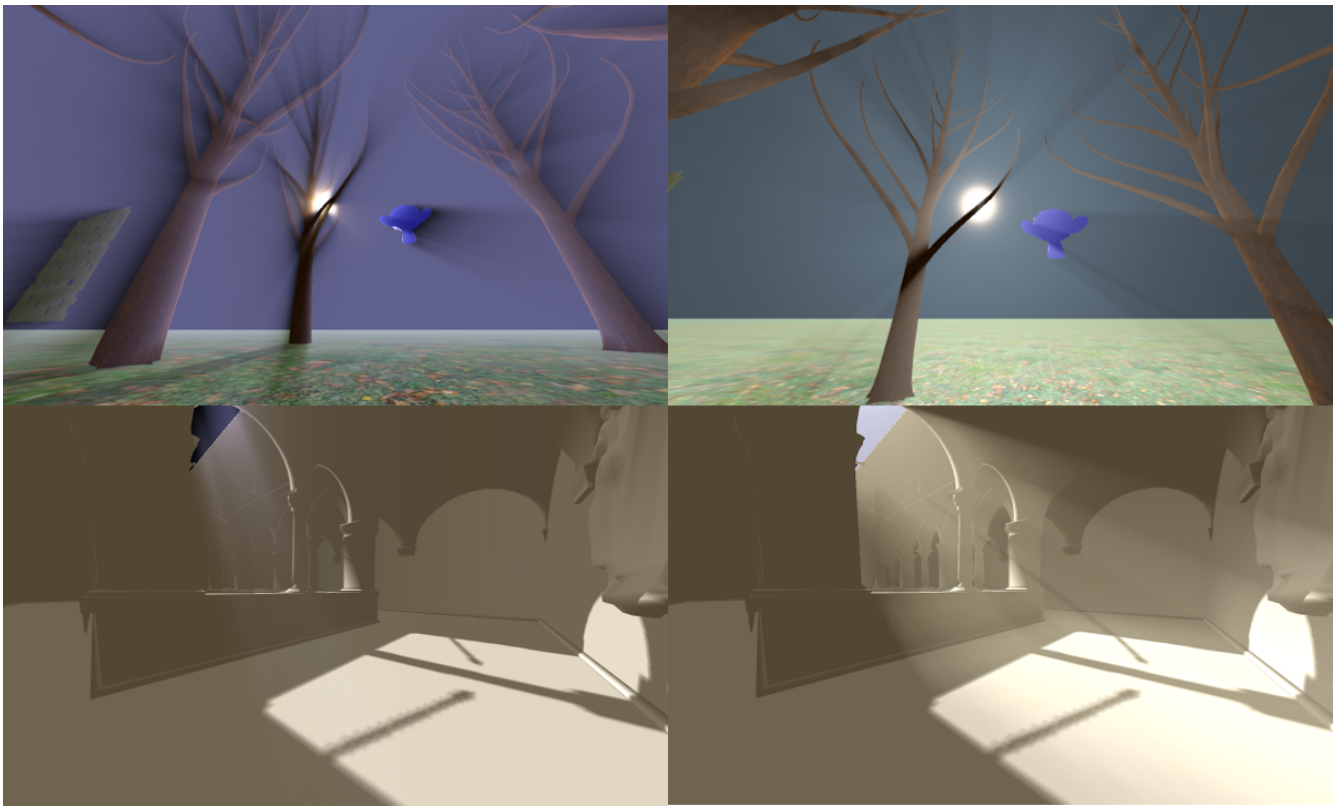

Figure 8: *Radial Blur (links) und Volumetric Lighting (rechts)*

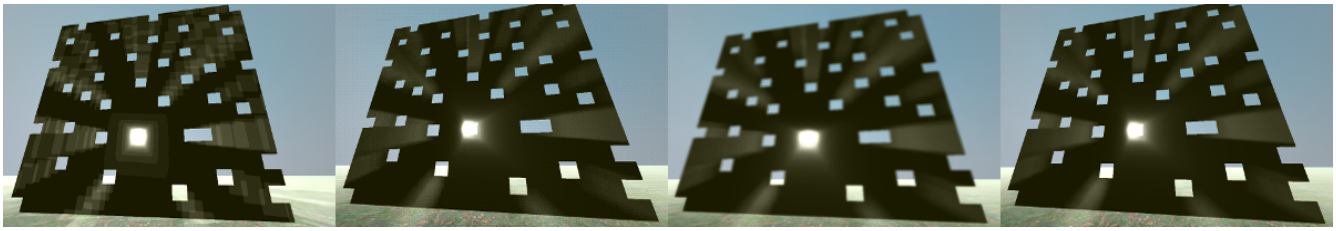

Figure 9: *Volumetric Lighting 10 Samples(links), Noise, Gauß-Blur und Bilateraler-Gauß-Blur*

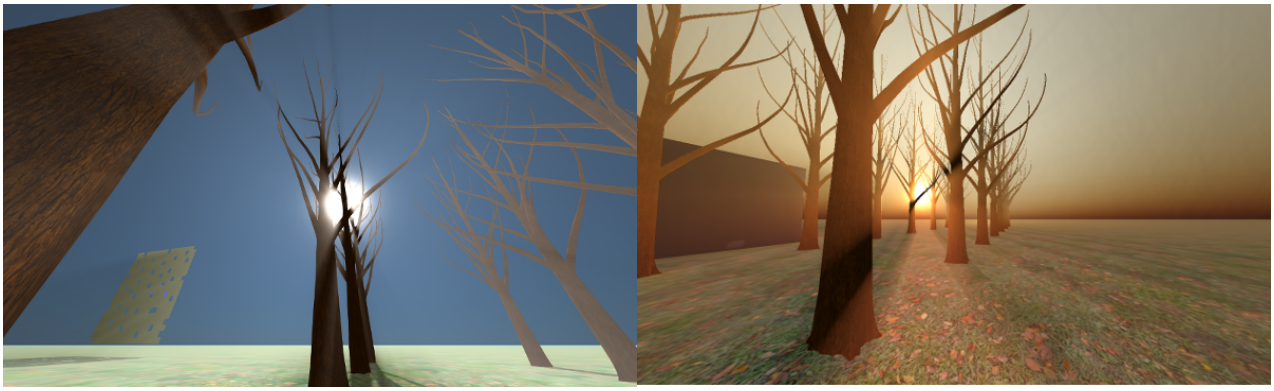

Figure 10: *Volumetric Lighting mit Atmospheric Scattering am Tag und bei Sonnenuntergang*

Optimierung. Neben *Interleaved-Sampling* [Glatzel 14] [Tóth 09] ist das Epipolar-Sampling mit 1D min/max mipmaps [ED 10] [EY 12] populär. Wir haben mit unserer Implementierung des grundelegenden Algorithmus eine Basis, um diese mit den verschiedensten Optimierungen leicht erweitern zu können.

Eine Erweiterungsmöglichkeit ist Atmospheric Scattering beim Volumetric Lighting mitzuberechnen. Dies haben wir nur grundlegend implementiert und benötigt noch Optimierungen. Eine Implementierung hierzu ist in [EY 13], als Erweiterung zu [EY 12], beschrieben.

References

[Tóth 09] Tóth, B., And Umenhoffer, T. (2009). Real-Time Volumetric Lighting in Participating Media. EUROGRAPHICS 2009. http://sirkan.iit.bme.hu/ szirmay/lightshaft.pdf

[Glatzel 14] Benjamin Glatzel (2014). Volumetric Lighting for Many Lights in Lords of the Fallen. *bglatzel.movingblocks.net* http://bglatzel.movingblocks.net/wp-content/uploads/2014/05/Volumetric-Lighting-for-Many-Lights-in-Lords-of-the-Fallen.pdf. Digital Dragons Conference 2014

[ED 10] Engelhardt, T., and Dachsbacher, C. (2010). Epipolar sampling for shadows and crepuscular rays in participating media with single scattering. In Proc. 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games, ACM, 119–125. http://www.vis.uni-stuttgart.de/ engelhts/paper/espmss10.pdf

[Mitchell 07] Mitchell, Kenny (2007). Volumetric Light Scattering as a Post-Process. *GPU Gems 3*, Chapter 13 http://http.developer.nvidia.com/GPU Gems3/gpugems3_ch13.html

[Preetham, 1999] Preetham A.J., Shirley P., Smits B.E. (1999). A practical analytic model for daylight. In Computer Graphics Proceedings, Annual Conference Series (Proc.SIGGRAPH '99), pp 91–100. http://www.cs.duke.edu/courses/cps124 /spring08/assign/07_papers/p91-preetham.pdf

[HP 02] Hoffman N., Preetham A. J.: Rendering outdoor light scattering in real time. Proceedings of Game Developer Conference (2002). http://developer.amd.com/wordpress /media/2012/10/GDC_02_Hoffman Preetham.pdf

[HP-Paper 02] Hoffman N., Preetham A. J.: Rendering outdoor light scattering in real time. http://amd-dev.wpengine.netdna-cdn.com/wordpress/media/2012/10/ATI-LightScattering.pdf

[DH 11] Daniel Holbert: Saying "Goodbye" to Shadow Acne. GDC 2011. http://www.dissidentlogic.com/old/ images/NormalOffsetShadows/ GDC_Poster_NormalOffset.png

[TS 08] Tiago Sousa: Crysis Next Gen Effects. GDC 2008, pp 62-65. http://crytek.com/download/ GDC08_SousaT_CrysisEffects.ppt

[EY 12] Egor Yusov, Intel: IVB Atmospheric Light Scattering. https://software.intel.com/en-us/articles/ivb-atmospheric-light-scattering

[EY 13] Egor Yusov, Intel: Outdoor Light Scattering Sample. https://software.intel.com/en-us/blogs/2013/06/26/outdoor-light-scattering-sample

[TM 98] Tomasi C., Manduchi R.: Bilateral Filtering for Gray and Color Images. Proceedings of the 1998 IEEE International. Conference on Computer Vision, Bombay, India. https://users.soe.ucsc.edu/ manduchi/Papers/ICCV98.pdf

[RL 1871] Rayleigh, L., On the scattering of light by small particles, Philosophical Magazine 41, S. 447-454, 1871

[HG 41] Henyey, C., and Greenstein, J. 1941, Diffuse Radiation in the Galaxy, Astro. J., 93, 70-83

[wissen.de, 2016] wissen.de Lexikon (2016). Tyndall-Effekt *www.wissen.de* http://www.wissen.de/lexikon/tyndall-effekt.