Web-Visualisierung komplexer 3D-Stadtmodelle mit pyRT

MARTIN CHRISTEN[1] & MARKUS FEHR[1]

Zusammenfassung: Das interaktive Rendern von komplexen 3D-Stadtmodellen mit hochaufgelösten Texturen und einer hohen Anzahl an Geometrien ist im Webbrowser ist eine Herausforderung. In diesem Beitrag wird gezeigt, wie sehr grosse texturierte CityGML Datensätze unter Verwendung von GPUs in der cloud gerechnet und im Webbrowser effizient dargestellt werden können. Dabei wird das neu entwickelte Open-Source Rendering-Tool «pyRT» eingesetzt, welches unter Verwendung der Programmiersprache Python 3D-Szenen rendern kann. Im Beitrag wird auch gezeigt wie mit pyRT globale Beleuchtungsmodelle approximiert werden, um 3D-Stadtmodelle in hoher Qualität darzustellen.
Die physikalisch plausiblen berechneten 3D Ansichten werden in der Cloud berechnet und auf dem Webclient darstellt. Mittels hochperformanten GPU-unterstützten Ray-Tracing können somit auch sehr komplexe 3D-Stadtmodelle mit einer sehr grossen Anzahl von Geometrien effizient dargestellt werden.

1 Einleitung

Die Visualisierung von 3D-Stadtmodellen im Webbrowser und auf mobilen Geräten hat in den letzten Jahren an Bedeutung gewonnen. Dank „out of core-Rendering"-Ansätzen und Level of Detail lassen sich sehr grosse 3D-Stadtmodelle mit weit über 100 Millionen Dreiecken und 12 GB Speicherplatz für die Geometrie darstellen (JIANG et al. 2014). Durch die Verwendung von Level of Detail und dem meist langsamen Aufbau der 3D-Modelle müssen jedoch Kompromisse eingegangen werden, was immer auch reduzierte Qualität bedeutet. Zudem werden bei der Darstellung nach wie vor sehr einfache Beleuchtungsmodelle eingesetzt.
Um qualitativ hochwertige 3D-Stadtmodelle im Webbrowser interaktiv zu rendern müssen neue Ansätze gefunden werden. Eine Möglichkeit dazu ist die Verwendung von interaktiven orthogonaler 3D-Karten, welche in der Cloud gerechnet werden und als G-Buffer (SAITO 1990). im Client dargestellt werden (CHRISTEN 2016). In diesem Beitrag wird dieses Verfahren erweitert, um qualitativ hochwertige 3D-Ansichten von untexturierten 3D-Stadtmodellen in Echtzeit zu erstellen, ohne dabei die Originalgeometrie zu reduzieren. Dabei wird ein neu entwickelter Ray-Tracer vorgestellt, welcher GPU und CPU optimiert neben dem G-Buffer auch eine Ambient Occlusion Map berechnen kann. Dieser ist optimiert auf Berechnungen von Schrägansichten aufgeteilt im Quadtree und berechnet in der Cloud (CHRISTEN 2016).

2 Stand der Technik

2.1 GPU unterstütztes Ray-Tracing

Der erste Versuch von Raytracing auf einer GPU zu beschleunigen wurde im Jahr 2002 mit der Ray Engine von CARR et al. (2002) gemacht. Die Ray Engine konnte nur die Strahl-Dreieck

[1] Fachhochschule Nordwestschweiz, Institut Vermessung und Geoinformation, Gründenstrasse 40, CH-4132 Muttenz, E-Mail: [martin.christen, markus.fehr]@fhnw.ch

Intersektion auf der GPU berechnen. Das Übertragen der Geometrie auf die GPU war zu dieser Zeit die grösste Schwierigkeit. PURCELL et al. (2002) und PURCELL (2004) konnten alle Berechnungen auf die GPU auslagern, wie zum Beispiel die Primärstrahlen, die Beschleunigungsstrukturen, Schattierung, Strahl-Dreieck Intersektion. Dieses Verfahren war die Basis einiger anderer Implementationen, wie von KARLSSON (2004), ERNST et al. (2004) und CHRISTEN (2005). Das Problem dieser Verfahren war, dass sie mangels einer geeigneten Hochsprache nur über Umwege durch Verwendung von Vertex- und Fragmentshadern implementiert werden konnten.
Eine weitere Schwierigkeit war das traversieren von hierarchischen Beschleunigungsstrukturen auf der GPU. FOLEY & SUGERMAN (2005) implementierten zwei Verfahren des kd-Baums auf der GPU. Weiter entwickelten HORN et al. (2007) und POPOV et al. (2007) je einen Raytracer basierend auf der kd-Tree Beschleunigungsstruktur. Auch BVH Beschleunigungsstrukturen wurden entwickelt, so zum Beispiel von THRANE & SIMONSEN (2005), CARR et al. (2006) oder von GÜNTHER et al. (2007).
Kürzlich AMD hat das Projekt GPUOpen veröffentlicht - ein Open Source Produkt, welches verschiedene Libraries, Tools und SDKs beinhaltet, um bessere Kontrolle über die GPU erhalten (THIBIEROZ 2016).
Eine dieser SDKs ist die Radeon Rays SDK (ehemals FireRays). Radeon Rays ist eine high-performance Raytracing Library, optimiert auf GPU beschleunigte Intersektionsberechnung. Sie beinhaltet eine C++ API und basiert auf OpenCL. Das heisst sie unterstützt alle Plattformen basierend auf den OpenCL 1.2 Standard (AMD, 2016a). Die Radeon Rays SDK ist nicht limitiert auf AMD Hardware, sondern ist Hardware unabhängig. Lizensiert ist Radeon Rays unter der MIT Lizenz (AMD, 2016a).
Auch CPU-optimierte Ray-Tracer haben in den letzten Jahren an grosser Bedeutung gewonnen, so hat Intel mit Embree einen Raytracing-Kernel entwickelt, welcher sich für fotorealistisches rendern auf den neusten Intel Prozessoren eignet (INTEL CORPORATION 2016b). Embree ist ebenfalls ein Open Source Produkt, veröffentlicht unter der Apache 2.0 Lizenz. Ray Tracing Kernel verfügen normalerweise nur über basis-Funktionalitäten wie Intersektionstest

2.2 Ambient Occlusion

Ambient Occlusion ist insbesondere in der Filmbranche ein Standardverfahren geworden für qualitativ hochwertiges Rendering (LAINE & KARRAS 2010). Erstmals vorgestellt wurde es von ZHUKOV et al. (1998) und danach in der Rendering Community aufgenommen (LANDIS 2002).
Ambient Occlusion wurde von ZHUKOV et al. (1998) folgendermassen definiert:

$$W(p,n) = \frac{1}{\pi}\int_{\Omega} \rho\big(D(p,\omega)\big)(\omega \cdot n)d\omega$$

W enthält die gewichtete Anzahl Verdeckungen; p ist der Punkt der Oberfläche, welcher verdeckt wird und n dessen Normale. Das Integral läuft dabei über die Hemisphäre, welche mittels n orientiert wird.
$D(p,\omega)$ misst die Distanz zur am nächsten liegenden Verdeckung in Richtung ω und mit ρ wird ein Faktor bestimmt, welcher die Distanz zur Verdeckung in einen Wert zwischen 0 und 1 konvertiert und deshalb auch eine maximale Distanz benötigt. Üblicherweise wird das Integral mittels Monte Carlo-Methoden approximiert, so dass nur eine vordefinierte Anzahl zufälliger

Strahlen in der Hemisphäre definiert werden. Typischerweise sind dies im Bereich N = 200 bis N = 1000 Strahlen pro Punkt. Ambient Occlusion kann somit folgendermassen vereinfacht werden:

$$W(p,n) \approx \frac{1}{N}\sum_{i=1}^{N} \rho\big(D(p,\omega_i)\big)$$

Später wurde Ambient Occlusion auch für Real-Time Applikationen erweitert wie beispielsweise mittels Screen Space Ambient Occlusion (BAVOIL 2008) oder Screen Space Far-Ambient Obscurance (TIMONEN 2014), Abbildung 1. Qualitativ ist dies jedoch weniger gut als die Originallösung.

Abb. 1: Screen-Space Far-Field Ambient Obscurance (TIMONEN 2014)

2.3 Serverseitiges Vorrendern von 3D-Geodaten

In letzter Zeit wurden Methoden vorgestellt, um Geodaten durch vorgänges rendern effizient darzustellen. HILDEBRANT et al. (2011) haben zum Beispiel eine Architektur zum serverseitigen Rendering von 3D-Panoramen mittels Cube-Maps vorgestellt. Bei dieser Lösung werden aus dem 3D-Modell verschiedene Panoramen vorgerechnet und mit einem Panorama-Viewer dargestellt. Eine andere Lösung, bei der komplexe 3D-Modelle dargestellt wurden, ist das Zusammensetzen von Schrägansichten aus Kacheln, welche es ermöglicht diese im Webbrowser und auf mobilen Geräten darzustellen (KLIMKE et al. 2014). Auch erwähnenswert ist das HuMoRS system (Huge models Mobile Rendering System), bei dem die 3D-Modelle interaktiv über das Netzwerk gestreamt werden, und es ermöglicht wird, sich auf einem mobilen Gerät interaktiv durch eine Szene zu bewegen (RODRIGEZ 2014). Das Open Geospatial Consortium (OGC) hat sich in letzter

Zeit auch mit der Interoperabilität von 3D-Geodaten für die Visualisierung auseinandergesetzt, insbesondere auch für den Austausch von komplexeren Szenen (OGC 2015).
In der Arbeit von CHRISTEN & NEBIKER (2015) wurden 3D-Geodaten mittels Ray-Tracing gerechnet und der resultierende G-Buffer in einer Quadtree-Struktur gespeichert um 3D Karten ähnlich wie 2D-Karten zu behandel, wie in Abbildung 2 zu sehen ist.

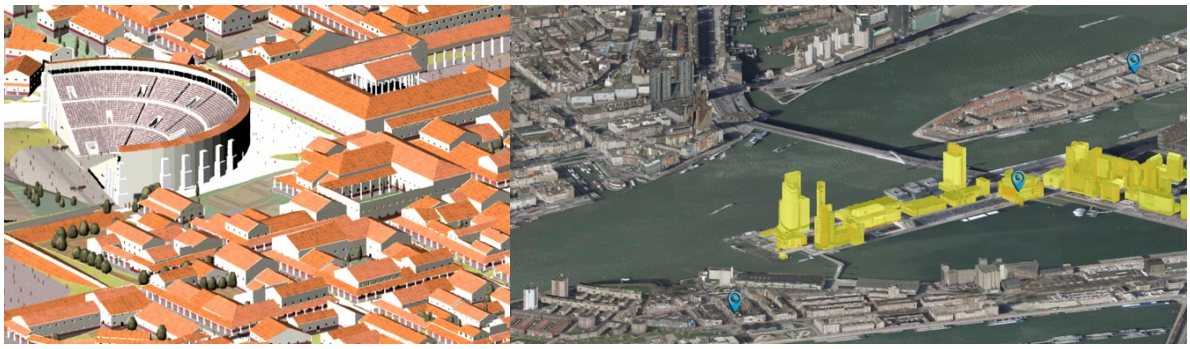

Abb. 2: Beispiele für Cloudgerenderte, gekachelte Schrägansichten (CHRISTEN & NEBIKER 2015)

3 Rendern von Schrägansichten mit pyRT

Das Projekt pyRT wurde gestartet um einen neuen Raytracer zu erstellen, welcher die Resultate in einem einfach erweiterbaren G-Buffer liefert. So ist es beispielsweise möglich Tiefenkarten mit mehr als 64-bit pro Pixel zu generieren, oder Objekt-ID maps mit direkter Referenz auf das Objekt. Mit herkömmlichen Open-Source Ray-Tracern wie POV-Ray (POV-RAY 2017) oder RENDERMAN (2017) ist dies nur über ineffiziente Umwege möglich (LINDENBERGER 2016).
In diesem Beitrag wird jedoch nicht auf diese eingegangen, sondern auf die Ambient Occlusion map, welche zusätzlich generiert werden kann, um die Qualität zu verbessern.
pyRT ist in Python geschrieben, benutzt aber Bindings zu Embree und AMD FireRays API, um auch moderne CPUs und GPUs direkt zu nutzen. Es werden nun verschiedene Szenen angesehen und verglichen. Texturen interessieren bei der Ambient Occlusion nicht, da nur die Ambiente Beleuchtung gerechnet wird. In einem nächsten Prozess kann die Textur mit der Ambient Occlusion kombiniert werden. Die Berechnug des G-Buffers erfolgt dabei im selben Rendering Schritt, es müssen dafür keine doppelten oder dreifachen Berechnungen durchgeführt werden wie z.B. bei der Verwendung von Renderman.

Tab. 1: Verwendete 3D-Modelle für die Tests

Modell	Anzahl Dreiecke	Dateigrösse (Wavefront OBJ Format)
Sponza	66’447	5.53 MB
Basel	128’993	11.90 MB
Drache	871’308	62.59 MB

3.1 Sponza Modell

Die Ambient Occlusion Map der Sponza Szene wurde als erstes gerendert. In der Abbildung 3 sind drei Ansichten (S1 bis S3) mit verschiedenen Maximaldistanzen dargestellt (FEHR 2017). Wird die Distanz zu kurz gewählt, wird das Bild heller und hat teilweise unschöne Übergänge. Wird die Distanz zu gross gewählt verdunkelt sich das Bild zunehmend.

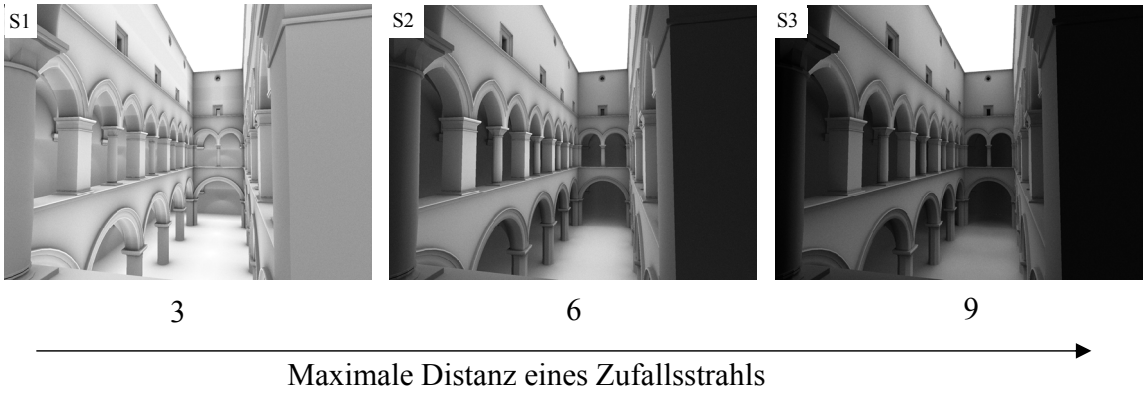

Abb. 3: Einfluss der Maximalen Distanz auf die Ambient Occlusion (FEHR 2017)

Auch die Anzahl Strahlen hat einen Einfluss auf die Qualität. Werden zu wenig Strahlen gewählt (S4, S5), so wirkt das Resultat eher pixelig, wie in Abbildung 4 zu sehen ist.

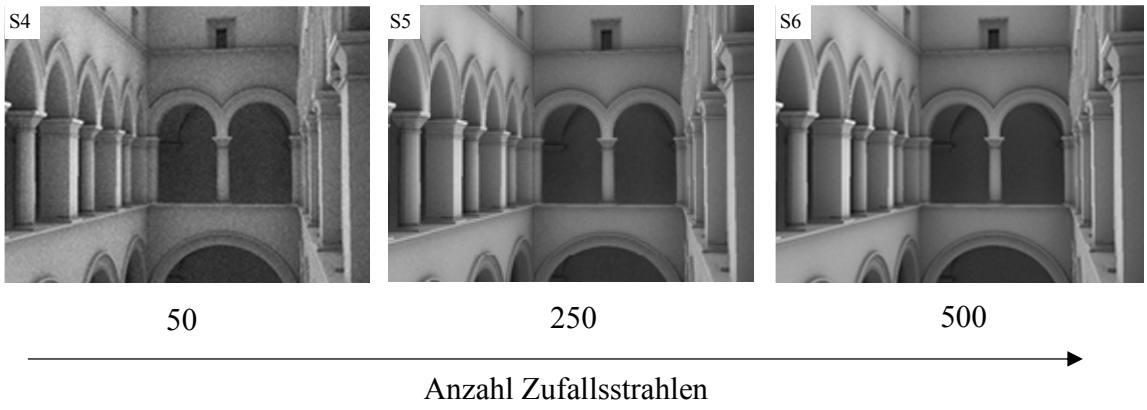

Abb. 4: Einfluss der Anzahl Strahlen auf die Szene (FEHR 2017).

Noch als Vergleich soll die Szene mit dem Blinn-Phong Beleuchtungsmodel (PHONG 1975) berechnet werden. Dabei wird es einmal mit Schatten (S8) und ohne Schatten (S7) gerendert. Intuitiv kann festgestellt werden, dass die visuelle Qualität signifikant schlechter ist, wie in Abbildung 5 zu sehen ist.

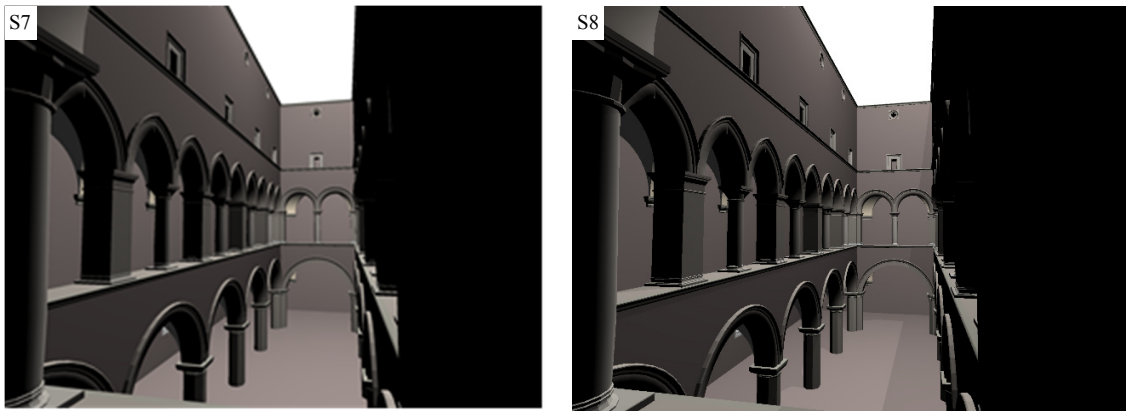

Abb. 5: Die Szene mit dem Blinn-Phong Modell gerechnet, mit und ohne Schatten (FEHR 2017)

3.2 Modell von Basel

Das 3D-Stadtmodell Basel vom Grundbuch und Vermessungsamt Basel Stadt wurde verwendet um die Ambient Occlusion zu rechnen. Dieses wurde wiederum mit Blinn Phong (BS1) und Ambient Occlusion berechnet und ist in Abbildung 6 zu sehen.

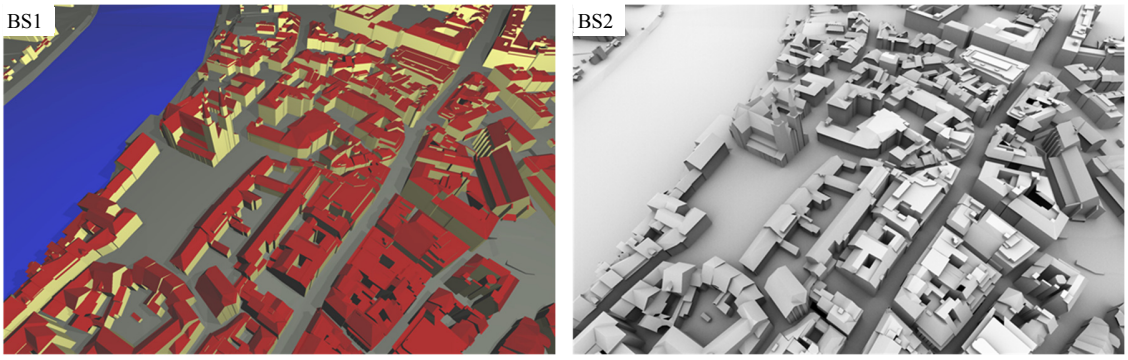

Abb. 6: Die Basel Szene mit dem Blinn-Phong (links) und Ambient Occlusion (rechts) (FEHR 2017)

3.3 Drachenmodell

Um auch mit einem Modell mit lokal höherer Anzahl Dreiecke zu testen, wurde das bekannte Drachenmodell mit Phong und mit Ambient Occlusion gerendert, wie in Abbildung 7 zu sehen ist.

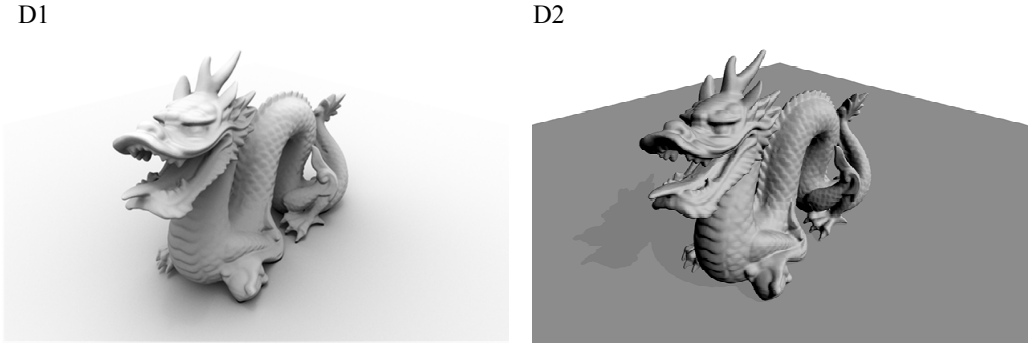

Abb. 7: Drachenszene mit Ambient Occlusion (links) und Phong (rechts)

4 Fazit und Ausblick

Es konnte gezeigt werden, dass mit pyRT ein Ray Tracer entwickelt wurde, welcher in einem einzigen Rendering-Schritt einen G-Buffer inklusive optionaler Ambient Occlusion Map generieren kann. Dabei können zur Beschleunigung Raytracing Kernel von AMD und Intel verwendet werden. Der Renderer eignet sich dazu, serverseitig 3D-Ansichten vorzurechnen, um diese dann interaktiv in einem (Web-)Client darzustellen. pyRT unterstützt im Moment allerdings noch keine Texturen, deren Support soll dann in einer späteren Version auch noch implementiert werden.

5 Literaturverzeichnis

ÁFRA A.T., WALD I., BENTHIN C. & WOOP S., 2016: Embree ray tracing kernels: overview and new features. ACM SIGGRAPH 2016 Talks (SIGGRAPH '16), ACM, New York, USA.

AMD, 2016a: Introducing the Radeon Rays SDK.

AMD, 2016b: Radeon™ R7 Series Graphics Cards. *AMD*. http://www.amd.com/en-us/products/graphics/notebook/r7-m200#, letzter Zugriff am 22.12.2016.

BARRINGER, R. & AKENINE-MÖLLER, T., 2014:,Dynamic ray stream traversal. ACM Transactions on Graphics (TOG) **33**(4), article 151, 9 p.

CARR, N. A.; HALL, J. D. & HART, J. C., 2002: The Ray Engine. Proceedings of Graphics Hardware **2**, 37-46.

CARR, N. A., HOBEROCK, J. & CRANE, K., 2006: Fast GPU Ray Tracing of Dynamic Meshes using Geometry Images. Proceedings of Graphics Interface 2006, Canadian Information Processing Society, 203-209.

CHRISTEN, M., 2005: Implementing Ray Tracing on the GPU. ShaderX4 - Advanced Rendering Techniques, W. Engel (ed.)Charles River Media, 413-424.

CHRISTEN, M., NEBIKER, S. & LOESCH, B., 2012: Web-Based Large-Scale 3D-Geovisualisation Using WebGL. International Journal of 3-D Information Modeling, **1**(3), 16-25.

CHRISTEN, M., HÜRBI, K. & NEBIKER, S., 2014: OpenWebGlobe: 3D-Visualisierung und Caching von globalen Stadtmodellen aus OpenStreetMap mittels Cloud-basiertem Framework. Publikationen der Deutschen Gesellschaft für Photogrammetrie, Fernerkundung und Geoinformation eV, Band 23, Potsdam, Seyfert, E., Gülch, E., Heipke, C., Schiewe, J., Sester, M. (Hrsg.), Beitrag 250.

CHRISTEN, M. & NEBIKER S., 2015: Visualisation of Complex 3D City Models on Mobile Webbrowsers using Cloud-based Image Provisioning, ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume II-3/W5.

CHRISTEN, M., 2016: Visualisierung von 3D-Geodaten im (mobilen) Webbrowser. Geoinformationssysteme 2016, Kolbe, T. H.; Bill, R.; Donaubauer, A. (Hrsg.), Wichmann, 2016, ISBN 978-3-87907-610-9

EVANS, A., ROMEO, M., BAHREHMAND, A., AGENJO, J., & BLAT, J., 2014: 3D graphics on the web: A survey. Computers & Graphics **41**(0), 43-61.

ERNST, M., 2009: Ray Tracing Techniques for Hybrid and Photorealistic Rendering. Dissertation, Friedrich Alexander Universität, Erlangen-Nürnberg.

FEHR, M., 2017: Beleuchtungsmodelle für 3D-Stadtmodelle mit GPU-optimiertem Rendering in der Cloud. Master-Thesis, Institut Vermessung und Geoinformation, Fachhochschule Nordwestschweiz

FOLEY, T. & SUGERMAN, J., 2005: KD-tree Acceleration Structures for a GPU Raytracer. ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware, 15-22.

GÜNTHER, J., POPOV, S. & SEIDEL, H.-P., 2007: Realtime Ray Tracing on GPU with BVH-based Packet Traversal. IEEE Symposium on Interactive Ray Tracing, Ulm, 113-118.

HILDEBRANDT, D., KLIMKE, J., HAGEDORN, B. & DÖLLNER, J., 2011. Service-oriented interactive 3D visualization of massive 3D city models on thin clients. Proceedings of the 2nd International Conference on Computing for Geospatial Research & Applications (COM.Geo '11). ACM, New York, NY, USA

INTEL CORPORATION, 2016a: A Ray Tracing Based Rendering Engine for High-Fidelity Visualization. OSPRay. http://www.ospray.org/, letzter Zugriff am 26.09.2016.

INTEL CORPORATION, 2016b: Embree Overview. Embree - High Performance Ray Tracing Kernels. https://embree.github.io/, letzter Zugriff am 26.09.2016.

JIANG, S., SAJADI, B., IHLER, A. & GOPI, M., 2014: Optimizing redundant-data clustering for interactive walkthrough applications. The Visual Computer, **30**(6-8), 637-647.

KAJALIN, V., 2009: Screen-Space Ambient Occlusion. ShaderX7 - Advanced Rendering Techniques, W. Engel (ed.), Charles River Media, 413-424.

KARLSSON, F., 2004: Ray tracing fully implemented on programmable graphics hardware. Master Thesis, Chalmers University of Technology, Göteborg.

KLIMKE, J., HAGEDORN, B. & DÖLLNER, J., 2014: Scalable Multi-Platform Distribution of Spatial 3D Contents. International Journal of 3-D Information Modeling (IJ3DIM) **3**(3), 35-49.

LAINE, S. & KARRAS, T., 2010: Two methods for fast ray-cast ambient occlusion. Computer Graphics Forum **29**(4), Blackwell Publishing Ltd., 1325-1333.

LANDIS, H., 2002: RenderMan in Production. ACM SIGGRAPH 2002, Course 16.

LINDENBERGER, C., 2015: Rendertechniken für komplexe Cloud-Basierte 3D-Stadtmodelle. Master-Thesis, Institut Vermessung und Geoinformation, Fachhochschule Nordwestschweiz.

MONGODB 2015: The MongoDB 3.0 Manual. http://docs.mongodb.org/manual/, last access November 20, 2015.

NEBIKER, S., BLEISCH, S. & CHRISTEN, M., 2010: Rich point clouds in virtual globes – A new paradigm in city modeling? Computers, Environment and Urban Systems **34**(6), 508-517.

OPEN GEOSPATIAL CONSORTIUM (OGC), 2015: OGC seeks public comment on candidate 3D Portrayal Service Standard. OGC Press Release, last access November 19, 2015 http://www.opengeospatial.org/pressroom/pressreleases/2165

PHONG, B.T., 1975: Illumination for Computer Generated Pictures. Communications of the ACM **18**(6), 311-317.

POVRAY, 2017: Persistence of Vision Renderer „POV-Ray“. http://povray.org, last access January 5, 2017.

PURCELL, T. J., 2004: Ray Tracing on a Stream Processor. PhD thesis, Stanford University.

PURCELL, T. J., BUCK, I. & MARK, W. R., 2002: Ray Tracing on Programmable Graphics Hardware. Proceedings of ACM SIGGRAPH, ACM (3), New York, 703-712.

RENDERMAN, 2017: Pixar Renderman Pro Server. http://renderman.pixar.com, last access January 10, 2017.

RODRÍGUEZ, M. B., AGUS, M., MARTON, F. & GOBBETTI, E., 2014: HuMoRS: huge models mobile rendering system. Proceedings of the Nineteenth International ACM Conference on 3D Web Technologies (Web3D '14), ACM, New York, USA.

SAITO, T. & TAKAHASHI, T., 1990: Comprehensible rendering of 3-D shapes. In Proceedings of the 17th annual conference on Computer graphics and interactive techniques (SIGGRAPH '90). ACM, New York, NY, USA, 197-206.

SCHAUB, M., 2014: Das Bronzemodell von Augusta Raurica: vom Stadtplan zum Stadtmodell. Augusta Raurica Magazin, **2014**(1), 7-9.

THIBIEROZ, N., 2016: It's Time to Open up the GPU. GPUOpen. http://gpuopen.com/welcometogpuopen/, letzter Zugriff am 26.09.2016.

THRANE, N. & SIMONSEN, L. O., 2005: A Comparison of Acceleration Structures for GPU Assisted Ray Tracing. Master Thesis, University of Aarhus.

TIMONEN, V., 2013: Screen-space far-field ambient obscurance. In Proceedings of the 5th High-Performance Graphics Conference (HPG '13). ACM, New York, NY, USA, 33-43.

ZHUKOV, S., INOES, A. & KRONIN, G. 1998: An Ambient Light Illumination Model. Rendering Techniques '98, Springer-Verlag Wien New York, G. Drettakis & N. Max (Eds.), Eurographics, 45-56.