

VU Echtzeitgraphik

Dokumentation

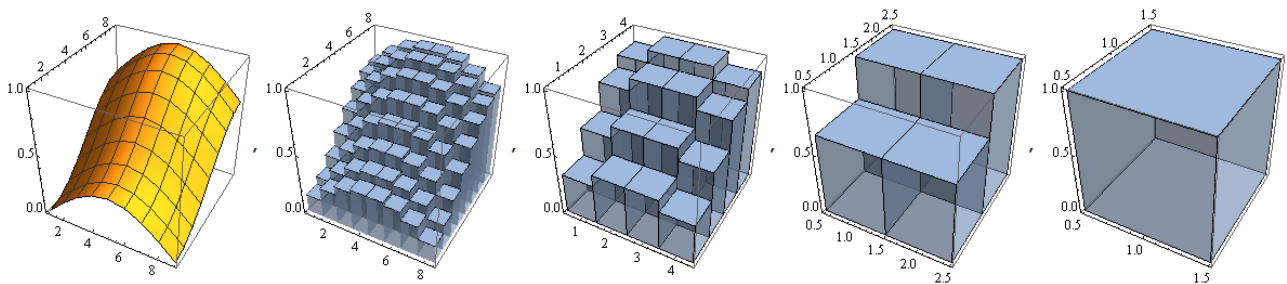
Gruppe Relief

Andreas Reichinger, 0427552, 086 881 059

Sebastian Metzler, 0927550, 066 932

Einführung

In unserem Projekt wird eine Oberfläche mithilfe von Tiefenvideos strukturiert, sodass eine 3-dimensionale Struktur dargestellt wird. Realisiert wurde dies mithilfe von Max Mipmapping [1]. Dabei werden die Levels der Mipmaps mit dem jeweils größten Wert des vorhergehenden Levels befüllt (siehe Grafik 1). Die Erstellung der Mipmaps erfolgt in Echtzeit. Für jeden neuen Frame des Tiefenvideos wird eine Max Mipmap erzeugt, welche dann vom Raytracer gerendert wird. Dieser stellt die Heightmap korrekt dar, berechnet Phong-Shading texturiert mit der Height-Map und fügt außerdem Reflexionen der Umgebung hinzu. Weiters enthält das Projekt Shader für Film Grain und Vignetting, welche über das gerenderte Heightfield gelegt werden.

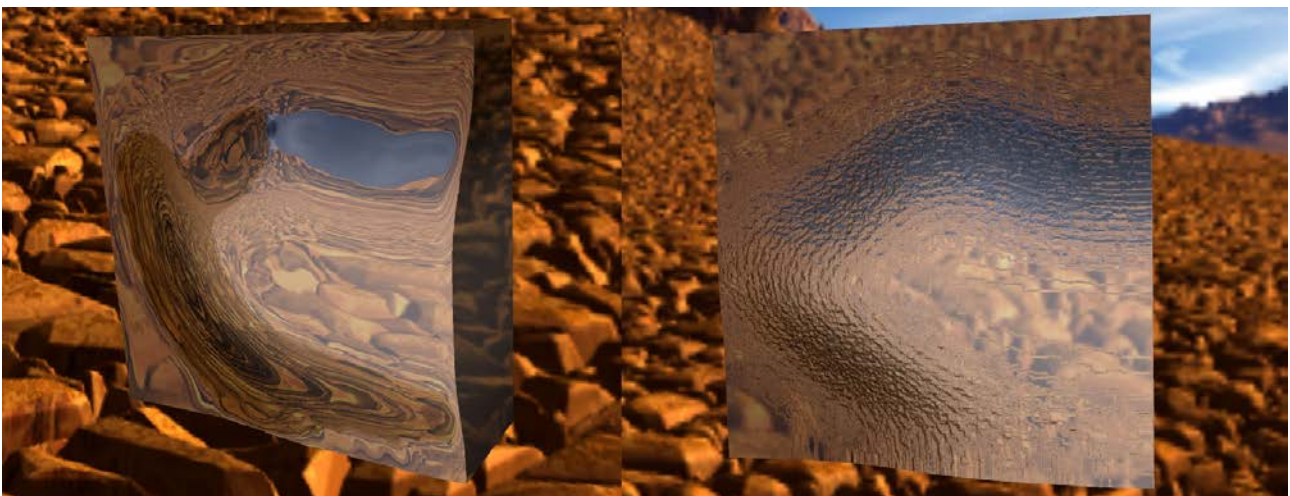


Grafik 1: Visualisierung einzelner Levels der Max MipMaps einer zugehörigen Oberfläche (links)

Shader

Heightfield Raytracer

Der Heightfield Raytracer besteht aus drei Komponenten. Im ersten Schritt wird ein Frame des Tiefenvideos eingelesen und die Max Mipmaps erzeugt. Dann werden die Normalen der erzeugten Patches berechnet. Da die 8-bit Farbtiefe herkömmlicher Videocodes für Tiefendarstellungen nicht ausreichend ist, entstehen in der Darstellungen sichtbare Abstufungen zwischen einzelnen Patches. Als korrigierende Maßnahme werden die berechneten Normalen nach der Erzeugung mithilfe eines Gauß Filters geglättet (siehe Grafik 2), und eine Abschätzung der Divergenz der Normalen berechnet. Schlussendlich wird das Heightfield durch effizientes Raytracing mithilfe der Max Mipmaps gerendert.



Grafik 2: Artefakte aufgrund der 8-bit Videoauflösung entstehen bei der Reflexion der Umgebung an der Oberfläche (rechts). Abhilfe schafft das Glätten der Normalen mit einem Gaußfilter (links).

Zum Raytracing wird die Bounding-Box des Height-Fields gerendert, mit Front-Face-Culling, da selbst bei Augpunkt innerhalb der Bounding-Box dieser sehrwohl außerhalb des Height-Fields sein kann. No Culling würde doppelten Aufwand bedeuten. Selbst wenn der Augpunkt außerhalb ist, eliminiert Front-Face-Culling Probleme mit Near-Clipping.

Der Algorithmus beginnt bei der höchsten Mipmap-Stufe und tested auf Strahl-Schnitt mit der bounding Box der aktuellen Zelle (Pixel der Mipmap). Wird diese nicht getroffen, geht der Strahl zum Ende der Zelle weiter, und geht dort in die maximale Mipmap-Stufe hinauf, die dort eine Grenze hat ([1] hat eine andere Strategie verfolgt, da es der Befehl „findLSB“ noch nicht gab, hat die Möglichkeit aber angedeutet). Wird aber die BBox der Zelle getroffen, geht der Strahl bis zum Schnittpunkt weiter, eine Mipmap-Stufe tiefer, und in die Zelle des Schnittpunktes. Ist er in der niedrigsten Stufe, wird mit dem bi-linearen Patch der Zelle geschnitten. Dafür ist die Lösung einer quadratischen Gleichung nötig, die 0, 1 oder 2 Lösungen gibt. Es wird der nächste Schnitt genommen, der in der Zelle ist. Gibt es keinen Schnittpunkt innerhalb der Zelle, geht es wie oben zum Ende der Zelle. Ansonsten ist die Suche zu Ende und das Fragment wird geshadet.

Wir verwenden ein einfaches Phong-Shading mit beliebig vielen Punkt-Lichtquelle (im Demo ein weißes als Sonne und am Ende noch ein rotierendes rotes Licht), und Environment Mapping. Dazu wird der Strahl an der Oberfläche anhand der Normalen reflektiert und durch einen Texturelookup im gewünschten Mipmap-Level der Environment Map die Reflexion berechnet. Die Mipmap Stufe wird entsprechend der vorher berechneten Divergenz-Abschätzung gewählt, da konkave oder konvexe Bereiche einen größeren Bereich der Umgebung pro Pixel reflektiert, und es sonst zu aliasing führt. Die Abschätzung wird als der größte Winkel (α) zwischen je zwei der vier Normalen der angrenzenden Flächen je Vertex berechnet. Die Mipmap-Stufe wird proportional zu $\log_2(\tan(\alpha))$ abgeschätzt. Der \log_2 wird aber erst nach dem Gauss-Filtern der Normal-Map berechnet, da der Logarithmus nicht linear ist, und somit nicht sinnvoll gemittelt werden kann.

Die Implementierung des Raytracing war nicht trivial. Es gibt diverse Sonderfälle wie das Rendern der Box, die das Height-Field umschließt, unterschiedliche Anfangsbedingungen je nach Position des Aug-Punktes (außerhalb / innerhalb der Bounding Box, im inneren des Height-Fields,...), und Unterschiede im Algorithmus, ob der Strahl in $+z$ oder $-z$ geht. Auch gibt es numerische Instabilitäten, wenn der Strahl parallel zu einer der Achsen verläuft, welche einer gesonderten Behandlung bedurften.

Es war interessant diesen Shader auszuprobieren. Die Sonderfälle machen den Shader aber sehr Rechenintensiv, und es ist zu überlegen, ob klassischer Triangulierung, oder die Verwendung von Geometry-Shadern und Rasterung nicht effizienter sind.

Die einzelnen MipMap Stufen können übrigens mit den + und - Tasten dargestellt werden.

Vignetting

Der Vignetting Shader erhält eine Textur mit dem gerenderten Bild des Raytracers. Die Fragment Koordinaten werden normalisiert, skaliert und dann als Parameter für eine 2-dimensionale Gaußfunktion verwendet. Anschließend wird der erhaltene Wert nochmals skaliert und sodann mit dem ursprünglichen Farbwert multipliziert um eine Maskenfunktion zu erfüllen.

Filmgrain

Der Output des Vignetting Shaders dient als Input für den Filmgrain Shader. In bestimmten zeitlichen Abständen werden randomisierte Lookups einer Textur, welche Film Grain enthält, durchgeführt. Die zeitliche Frequenz ist dabei absichtlich niedriger als die

Framerate, da sonst bei hoher Framerate das Auge den meisten Noise wieder wegfiltert (zeitliches Mittel). Die erhaltenen Farbwerte werden sodann dem ursprünglichen Farbwert additiv beigemischt.

Außerdem beinhaltet der Shader ein einfaches Color-Grading, mit den beiden Parametern Saturierung und Helligkeit.

Animation

Das Demo verwendet Key-Frame Animation passend zur Musik. Musik und Depth-Map ist in einem Video abgespeichert. Die Musik liefert die Synchronisationszeit. Die einzelnen Frames werden gegrabbed, und sofern diese noch aktuell sind in die Height-Field Textur kopiert.

Wir haben keine Library gefunden, die gleichzeitig Audio abspielt und die Video-Frames als Bilder übergeben kann. Deshalb haben wir das Projekt AForge.Video.DirectShow [4] für unsere Zwecke erweitert.

Am Ende wechselt das Demo in einen Interaktiven Modus bei dem eine Halb-Kugel als Relief von allen Seiten betrachtet werden kann.

Steuerung

Leertaste: Pause bzw. Play

Links / Rechts: Skip 5 sec. rückwärts/vorwärts, mit Ctrl 30 sec.

Alt-Enter: toggle Fullscreen

Escape, Alt-F4: Beenden

Viewport:

Maus links dragging: Blick-Richtung (während Demo), Orbit (im interaktiven Modus)

Maus rechts dragging bzw. Scroll-Wheel: Camera-Distanz (im interaktiven Modus)

Max-Mipmap-Ansicht:

+ / -: Auswählen der Minimalen Mipmap Stufe des Raytracers

B: Toggle Darstellung der Zellen-Übergänge, Mipmap-Stufe codiert in rot – grün.

Test

Das Programm wurde auf dem Rechner HAL mit NVidia-Karte getestet.

References:

[1] Art Tevs, Ivo Ihrke, and Hans-Peter Seidel. 2008. Maximum mipmaps for fast, accurate, and scalable dynamic height field rendering. In Proceedings of the 2008 symposium on Interactive 3D graphics and games (I3D '08). ACM, New York, NY, USA, 183-190. <www.tevs.eu/files/i3d08.pdf>

[2] <http://sourceforge.net/projects/ogl4net/>

[3] <http://herakles.zcu.cz/~pvanecek/opengl4net/nopper.php>

[4] <http://www.aforgenet.com/>