## Echzeitgraphik VU - WS2007

# Demo

**Gruppe „THE DUDEMEISTERS“**

Dude0: **Ferdinand Pilz** #0727396/66932

Dude1: **Harald Reingruber** #0726257/66932

**The Dudemeisters – On our way to mars**

**Implementierte Effekte**

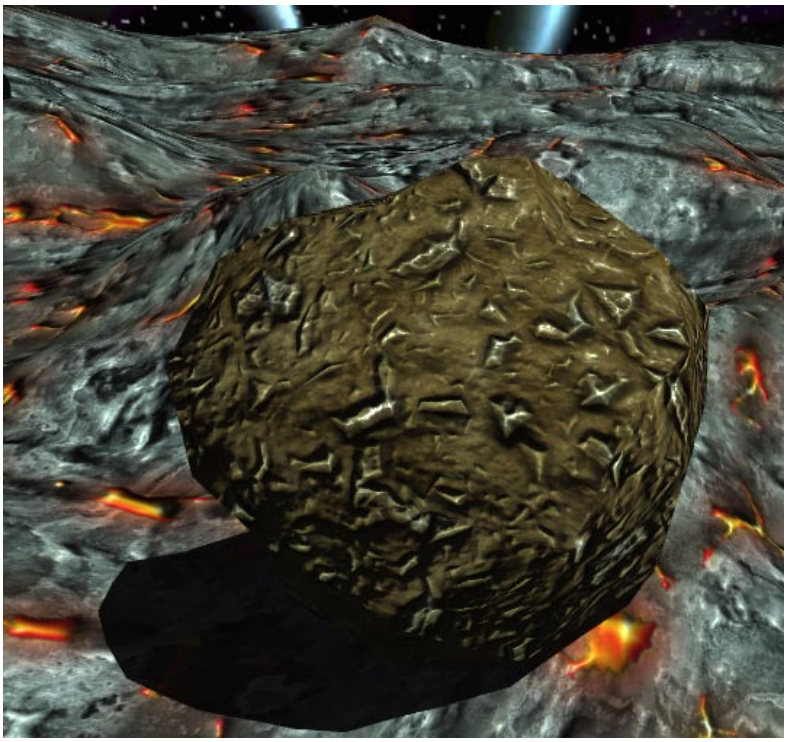
*Shadow Volumes*

Verwendete Quellen:

Zur Implementierung wurden die Paper „Fast, Practical and Robust Shadows“ (http://developer.nvidia.com/object/fast_shadow_volumes.html) mit dazugehörigem Programmbeispiel und „Practical and Robust Shadow Volumes“ (http://developer.nvidia.com/object/robust_shadow_volumes.html) herangezogen. Weiters wurde das Buch „Mathematics for 3D Game Development and Computer Graphics“ (2nd Edition) von Eric Lengyel verwendet.

Kurz zur Umsetzung:

Es kommt der Z-Pass Algorithmus zum Einsatz.
Für das Erzeugen der Schattengeometrie wird eine einzelne direktionale Lichtquelle verwendet, die die Sonne repräsentiert.
Bei Initialisierung der Applikation werden alle Edges eines 3D-Modells in eine Datenstruktur abgespeichert. Diese enthält die Indizes der beiden Vertices, sowie die Indices der beiden angrenzenden Faces. Dabei wird auch überprüft, ob jedes Edge an zwei Faces angrenzt.
Zur Laufzeit werden zuerst die Back- bzw. Front-Faces identifiziert, und danach die Silhouette-Edges gefunden, indem überprüft wird, ob ein Face-Index in der Datenstruktur auf ein Back- und einer auf ein Front-Face zeigt.
Die Silhouette-Edges werden dann als Quads extrudiert und laufen, wegen der direktionalen Lichtquelle, in einem Punkt zusammen.
Beim Schreiben in den Stencil-Buffer werden die Extensions `GL_STENCIL_TEST_TWO_SIDE_EXT`, `GL_DECR_WRAP_EXT` und `GL_INCR_WRAP_EXT` verwendet. Durch erstere muss die Geometrie nur einmal gerendert werden.



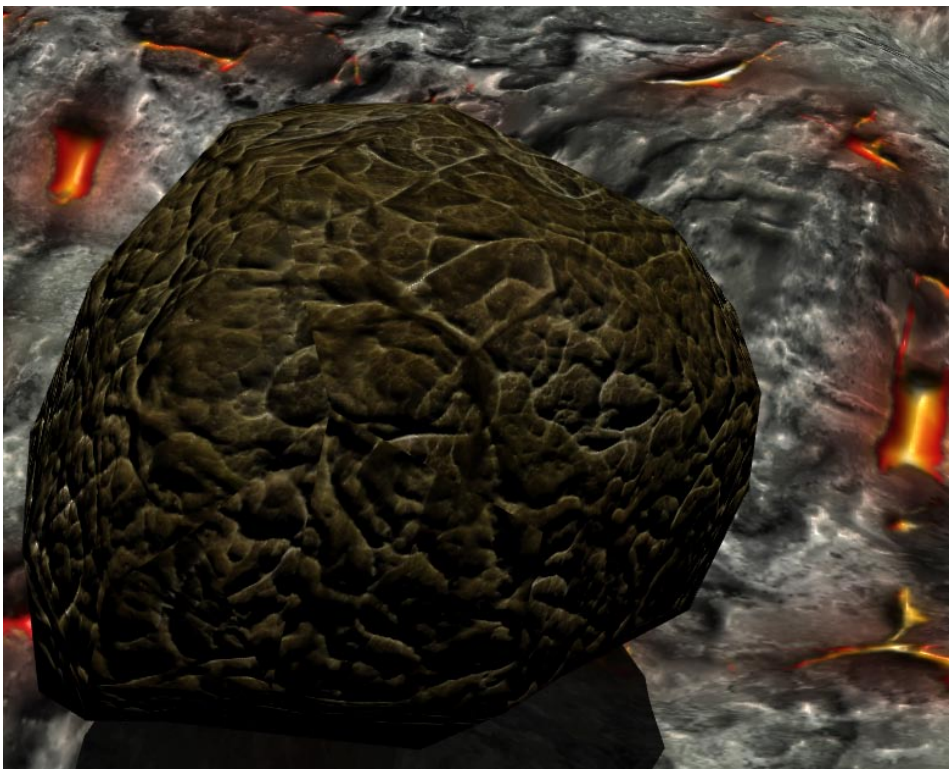
Zuerst wird die Geometrie mit ambienter Beleuchtung ferendert, danach die Shadow Volumes in den Stencil Buffer und anschließend kommt der zweite Rendering Pass mit diffus und specular Anteil.
Da nur der Z-Pass Algorithmus zum verwendet wird, werden die Werte im Stencil Buffer invertiert, wenn sich die Kamera in ein Shadow Volume begibt.

**Wird angewendet auf:** Raumschiff, Asteroiden, Säule

*Bump Mapping*

Verwendete Quellen:

“The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics”,
sowie “http://www.blacksmith-studios.dk/?p=3” und wiederum „Mathematics for 3D Game Development and Computer Graphics“ (2nd Edition) von Eric Lengyel.



Hierfür wurden aus den Decal-Texturen entsprechende Texturen mit den Perturbed Normals generiert (http://crazybump.com) was eine praktische Variante darstellt, wenn man keine High-Poly Version seines 3D Modells hat, um daraus eine Normal Map zu generieren.
Um den Lichtvektor und den Viewvektor in den TangentSpace zu rotieren, werden bereits im verwendeten ModelLoader die Tangenten pro Vertex als Vorverarbeitungsschritt berechnet. Dies deshalb, weil für jedes Modell eine Display List generiert wird, und die Tangenten darin verpackt werden und als Texturkoordinaten an das betreffende Vertex Program übergeben werden.
Für das Shading wurden zwei Varianten ausprobiert, die aber in Sachen Optik und Performance keinen erkennbaren Unterschied brachten. Es wurde zum einen der Half-Angle im Vertex Program berechnet und dazwischen interpoliert, und einmal der Licht- und Viewvektor an das Fragment Program übergeben und der Half-Angle pro Fragment berechnet.
Für den Specular Term wurden eigens Texturen angelet, die im Alphakanal der Decal-Texturen gespeichert sind. Dadurch wird das Ergebnis deutlich verbessert, weil die Highlights an das jeweilige Modell angepasst sind.

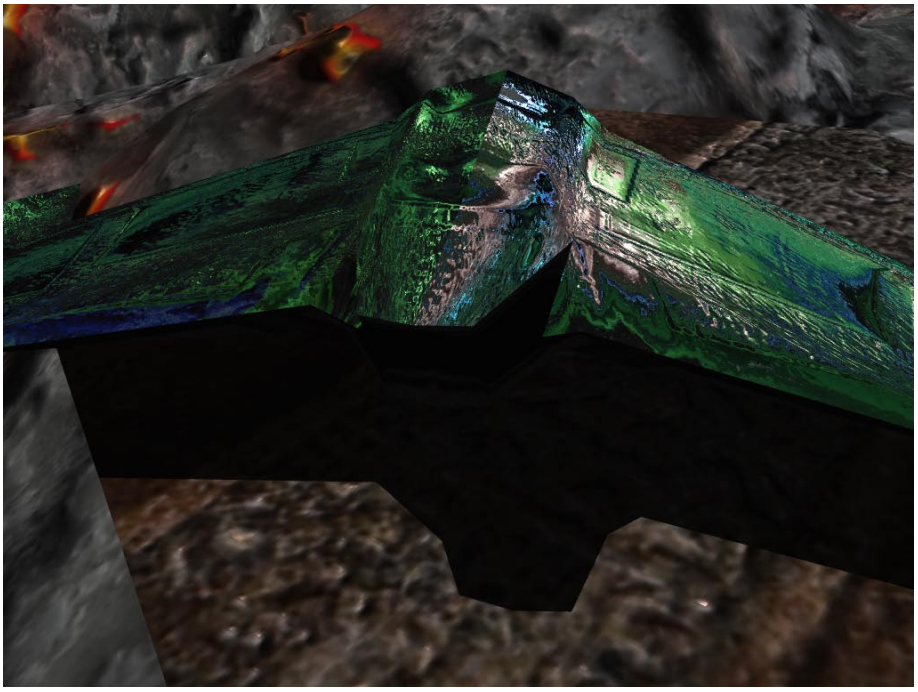
**Wird angewendet auf:** Raumschiff, Asteroiden, Terrain, Säule

**Environment Map Bump Mapping**

Verwendete Quellen:

“The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics”,
http://www.gamasutra.com/features/20050729/lacroix_02.shtml,
Vorlesungsfolien

Für diesen Effekt ist es notwendig, alle beteiligten Elemente in den World Space zu rotieren. Durch die inverse Tangent Space-Matrix und die Model-Matrix des Objektes werden die Normalen, die aus der Normal Map ausgelesen werden, in den World Space transformiert. Danach wird mit diesem Vektor der Lookup in der Cube Map durchgeführt, womit man die korrekt reflektierte Farbe erhält.



**Kamera Steuerung**

Für eine weiche Bewegung der Kamera wurde der Artikel 4.3 „Camera Control Techniques“ von Dante Treglia aus „Game Programming Gems 1“ verwendet.

Darin ist eine Spring-Funktion integriert, die die Kamera, sowie das Raumschiff sanft abbremsen und beschleunigen lässt. Für das selbstlaufende Demo wurden Punkte einer Spline definiert, auf der sich das Raumschiff bewegt.

**Model Loader**

Der verwendete Model Loader kann 3DS-Dateien laden und wurde von uns noch modifiziert. Die Modelle, wie die Planetenoberfläche und die Asteroiden wurden selber mit Autodesk Maya erstellt. Das Raumschiff ist ein freies 3D Modell aus dem Internet.

**Steuerung**

Das Demo startet automatisch im Auto-Mode, dieser kann mit L verlassen werden um die Steuerung des Raumschiffes zu übernehmen. Ein erneutes drücken der L-Taste lässt das Raumschiff wieder am Beginn der Spline starten.

| | |
|---|---|
| WASD + Maus | Raumschiff Steuerung |
| C | Fly Camera aktivieren |
| Pfeiltasten + Maus | Fly Camera steuern |
| Bild ↑ | Fly Camera erhöhen |
| Bild ↓ | Fly Camera senken |
| +/- | Abstand zw. Camera und Model verringern bzw. vergrößern |
| M | Third Person Camera aktivieren |
| K | Flugbahn des Auto-Modes anzeigen (Spline) |
| L | aus-/einschalten des Auto-Modes |

**System Voraussetzungen**

Windows XP
Core2Duo 2 GHz
512 MB RAM
GeForce 7000er

Leider gab es mit dem Release-Build Probleme. Er funktioniert nur durch Starten aus Visual Studio heraus einwandfrei. Ansonsten gehen anscheinend Referenzen auf manche Texturen verloren, die dann nicht angezeigt werden. Die Debug-Version läuft jedoch ohne Probleme und ebenfalls flüssig.