

# Cliffhanger

Peter Messenger, 0327186, [messenger.mine@gmail.com](mailto:messenger.mine@gmail.com)  
Matthias Mühlich, 0308771, [mmuehlich@gmx.li](mailto:mmuehlich@gmx.li)

## Treatment

Das Set-up für Cliffhanger besteht aus einer Felswand und der Hauptfigur, welche kurz vor dem Abgrund steht.

Während dies im bekannten Actionfilm Sylvester Stallone ist, wird der Protagonist in unserem Werk nicht ganz unpassend zu einem Baum abstrahiert... Dieser steht auf einem einsamen Fels in der Brandung, umgeben von Wasser und einem malerischen



Sonnenuntergang. Schatten und Lichtreflexionen verstärken den Eindruck der Szene.

## Effekte und Details

Einige Effekte wurden eingesetzt um die Szene realistischer und eindrucklicher erscheinen zu lassen. Sowohl Baum als auch Felswand werfen Schatten (auch auf sich selbst), welche mittels Shadow-Mapping realisiert werden. Die Details der Felswand und des Wassers werden durch Normal-Mapping hervorgehoben, für den Fels wird nur die diffusen Lichtanteile berechnet, für das Wasser wird auch die Spiegelung (das Specular Component) berücksichtigt. Jedoch kann der Rendering-Modus für das Wasser sowie für die Felswand interaktiv geändert werden.

## Schatten / Shadow Mapping:

Als Schatten Effekt wurde ein einfaches Shadow Mapping implementiert. Im ersten Pass wird die Szene von der eingestellten Lichtposition und -richtung aus ohne Effekte (nur Geometrie) gerendert. Dabei ist die Auflösung der Framebuffertextur in einem fixen Faktor (2) zur Bildschirmauflösung gehalten um eine höhere Tiefenauflösung zu erhalten. Mit

verschiedenen Distanzen der z-near und z-far planes wurde auch experimentiert, jedoch wurden keine bedeutenden Verbesserungen der Tiefenauflösung festgestellt.

Im zweiten pPss werden pro Fragment die in Lichtkoordinaten transformierten Tiefenwerte der gerenderten Szene mit denen im Depth Map gespeicherten Werten verglichen. Ist die Tiefe höher, wird das Fragment schwarz eingefärbt. Der Vergleich wird in Hardware durchgeführt. Um Selbstschattierungen zu minimieren werden im ersten Pass die frontfaces geculled und im zweiten pass ein kleiner Polygon Offset eingesetzt.

## **Normal Mapping / Bump Mapping:**

Die Normal-Mapping Implementierung ist angelehnt an Søren Dreijers oder Jérôme Guinots Tutorial (siehe Quellenangaben).

Beim Laden der Objekte werden die Normalen und Tangentialvektoren pro Vertex berechnet und dem Shader übergeben. Im Vertex Shader wird ein lokales Koordinatensystem mit Normalvektor, Tangentialvektor und Bitangentialvektor erstellt, welches die lokale Oberfläche an jedem Vertex beschreibt. Sowohl Licht- als auch Sichtrichtung werden in dieses Koordinatensystem umgerechnet.

Im Fragment Shader wird das Normalmap geladen, welches zur Einfachheit dieselben Texturkoordinaten verwendet wie die Farbtextur. Die Farbwerte (rgb) des Normalmaps werden in Normalvektoren umgerechnet (xyz) und zur Berechnung des Lichtes verwendet.

## **Player / Recorder:**

Das Programm verfügt über einen einfachen Player/Recorder. Im Record-Modus (Start mit 'R' - Ende mit 'R') werden alle gedrückten Tasten sowie die Anzahl Frames dazwischen aufgenommen und in eine Datei (*rec.cfg*) gespeichert. Die Mouse wird nicht aufgenommen, da sich dies als sehr buggy herausgestellt hat. Mit 'P' kann die zuletzt aufgenommene Sequenz abgespielt werden - dabei wird jedoch weder die Mouse gesteuert (diese kann der User selbst steuern) noch wird die Ausgangsposition oder Lichtrichtung gesetzt. Es wird deshalb empfohlen sowohl das Recording als auch das Playback mit der Ausgangsposition bei Programmstart zu beginnen.

Die Szene wird per default in der Datei *rec.cfg* gespeichert und bei jedem neuen Recording-Vorgang überschrieben.

## **Config-File:**

Die Datei *config.cfg* enthält einige konfigurierbare Einstellungen wie Fenstergrösse, Vollbildmodus und OpenGL- oder GLSL-Version.

## Quellen und Links:

### - Shadow Maps:

Timo Aila; Samuli Laine: Alias-Free Shadow Maps  
([http://www.tml.tkk.fi/~samuli/publications/aila2004egsr\\_paper.pdf](http://www.tml.tkk.fi/~samuli/publications/aila2004egsr_paper.pdf))

Daniel Scherzer; Stefan Jeschke; Michael Wimmer: Pixel-Correct Shadow Maps with Temporal Reprojection and Shadow Test Confidence  
(<http://www.cg.tuwien.ac.at/research/publications/2007/Scherzer-2007-PCS/Scherzer-2007-PCS-Preprint.pdf>)

Fabien Sanglard: Shadow Mapping with GLSL  
(<http://fabiansanglard.net/shadowmapping/index.php>)

Flashbang.se: Simple FBO rendering  
(<http://www.flashbang.se/index.php?id=18>)

### - Bump Mapping / Normal Mapping:

Mark J. Kilgard: A Practical and Robust Bump-mapping Technique for Today's GPUs  
(<http://www.cg.tuwien.ac.at/courses/Realtime/slides/PracticalBumpMap.pdf>)

Paul's Projects: Shadow mapping and bump mapping using the fixed pipeline  
(<http://www.paulsprojects.net/tutorials/tutorials.html>)

Fabien Sanglard: Bump Mapping with GLSL  
(<http://www.fabiansanglard.net/bumpMapping/index.php>)

Søren Dreijer: Bump Mapping Using CG (3rd Edition)  
([http://www.blacksmith-studios.dk/projects/downloads/bumpmapping\\_using\\_cg.php](http://www.blacksmith-studios.dk/projects/downloads/bumpmapping_using_cg.php))

Jérôme Guinot: Bump Mapping using GLSL  
([http://www.ozone3d.net/tutorials/bump\\_mapping.php](http://www.ozone3d.net/tutorials/bump_mapping.php))