**Grog**

Fuerst Rene, 0627916, 033 532, [rene.fuerst@live.com](mailto:rene.fuerst@live.com)  
Vogl Wolf-Dieter, 0626355, 033 532, [w.d.v@gmx.at](mailto:w.d.v@gmx.at)

# Treatment

Grog soll ein Geisterschiff fahrend auf dem Meer darstellen. Dabei sollen gespensterähnliche Gestalten auftreten und Furcht und Schrecken erregen. Es gibt eine Außenszene mit einer Kamerafahrt rund um das Schiff und eine Innenszene, die durch die verschiedenen Räume im Schiff führen.

# Effekte

## Global

* **High Dynamic Range Tone Mapping** mit adaptiver Helligkeitsanpassung und Bloom-Effekt (Separable Gauss-Filter).

*HDRRendering NVIDIA SDK*

*HDRToneMappingCS11 DirectX SDK*

* **Skeletal Animation**
* **Parallel Split Variance Shadow Maps (PSVSM):** Die Schatten auf dem Schiff werden mit Hilfe von Variance Shadow Maps berechnet. Es werden 2 Parallel-Split Shadow Maps verwendet, um Artifakte bei größeren Entfernungen zu vermeiden.

*GPU Gems 3 Chapter 8. Summed-Area Variance Shadow Maps*

*GPU Gems 3 Chapter 10. Parallel-Split Shadow Maps on Programmable GPUs*

* **Per Pixel Lighting, Lambert und Phong Surfaces**

## Außenszene

* **Vertex Displacement Mapping:** Die Wellen werden durch Vertex-Displacement Mapping erzeugt und schattiert. Ein einfaches, radiales LOD, wie in Chapter 18 der GPU-Gems 2 beschrieben, wird verwendet. Das Vertex-Displacement ist besonders gut am Schiffsrumpf zu erkennen.

*GPU Gems 2 Chapter 18*. *Using Vertex Texture Displacement for Realistic Water Rendering*

* **Normal Mapping:** Auf mehreren Texturen im und am Schiff wird Normal Mapping angewendet. Aufwändigere Verfahren wie Steep Parallax Mapping oder Parallax Occlusion Mapping werden aus aus Performancegründen nicht verwendet. Die Tangenten- und Binormalkoordinaten werden vom Programm berechnet.

*Parallax Mapping NVIDIA SDK*

*Computing Tangent Space Basis Vectors for an Arbitrary Mesh:* [*http://www.terathon.com/code/tangent.html*](http://www.terathon.com/code/tangent.html)

* **Reflection und Refraction:** Reflektion der Skybox und des Schiffes auf die planare Wasseroberfläche, sowie Refraktionen des Teiles des Schiffsrumpfes, der sich unter Wasser befindet. Der Grad der Reflektion wir durch einen approximierten Fresnel-Term bestimmt.

*GPU Gems 2 Chapter 19. Generic Refraction Simulation.*

*Realtime Water Rendering:* [*http://keepcoding.bplaced.com/kcdev/tutorials/Realtime%20Water%20Rendering.pdf*](http://keepcoding.bplaced.com/kcdev/tutorials/Realtime%20Water%20Rendering.pdf)

* **Volume Renderer:** Für den Totenschädel wurde ein Raycasting Volume Renderer implementiert, der auch mit 1D-Transferfunktionen arbeitet.

*VO Visualisierung*

## Innenszene

* **Volume Renderer:** Für volumetrische Elemente, wie Wasser und Feuer wurde ein Raycasting Volume Renderer implementiert, der auch mit Levelsets arbeitet.

*VO Visualisierung*

* **Volumetrisches Feuer und Wasser:** Das Feuer und Wasser wurden mit Navier-Stokes Gleichungen berechnet. Die Grundlagen dafür ist im Chapter 30 der GPU-Gems beschrieben, wobei die Randbedingungen der Gleichung selbst erarbeitet wurden. Für Wasser wurde zusätzlich noch Levelsets und Levelset-Advektion auf der GPU implementiert. Die simulierten volumetrischen Ergebnisse werden durch einen Volume-Renderer und einen Levelset-Renderer dargestellt.

*GPU Gems 3 Chapter 30. Real-Time Simulation and Rendering of 3D Fluids*

* **Spherical Harmonic Lighting:** Die Lichtausbreitung im Innenraum wird mit dem Spherical Harmonic Lighting Verfahren approximiert. Dazu wird die Geometrie in einem Vorverarbeitungsschritt mit Hilfe von Spheres angenähert.

*Approximative Real-time Soft Shadows and Diffuse Reflections in Dynamic Scenes* [*http://www.cg.tuwien.ac.at/research/publications/2007/guerrero-2008-dip/guerrero-2008-dip-thesis.pdf*](http://www.cg.tuwien.ac.at/research/publications/2007/guerrero-2008-dip/guerrero-2008-dip-thesis.pdf)

# Anmerkungen

Für das Demo wurde ein simples zeitgesteuertes Skripting entwickelt, damit der Ablauf durch verschiedene Events (Audio, Texteinblendungen, Fade In/Out…) einfach zu erweitern und adaptieren ist.

Die Kamerafahrt folgt einem B-Spline, gesteuert durch zeitliche Kontrollpunkte für Position und Orientierung.

Es stehen 2 verschiedene Versionen des Programmes zur Verfügung: x64 und x86 mit Optimierungen für Prozessoren mit 3DNow! bei der Vektor/Matrizenmathematik.

Die Models wurden großteils von Vania Petkova von Quantm modeliert. Selber wurden auch Models hinzugefügt und teilweise abgeändert.