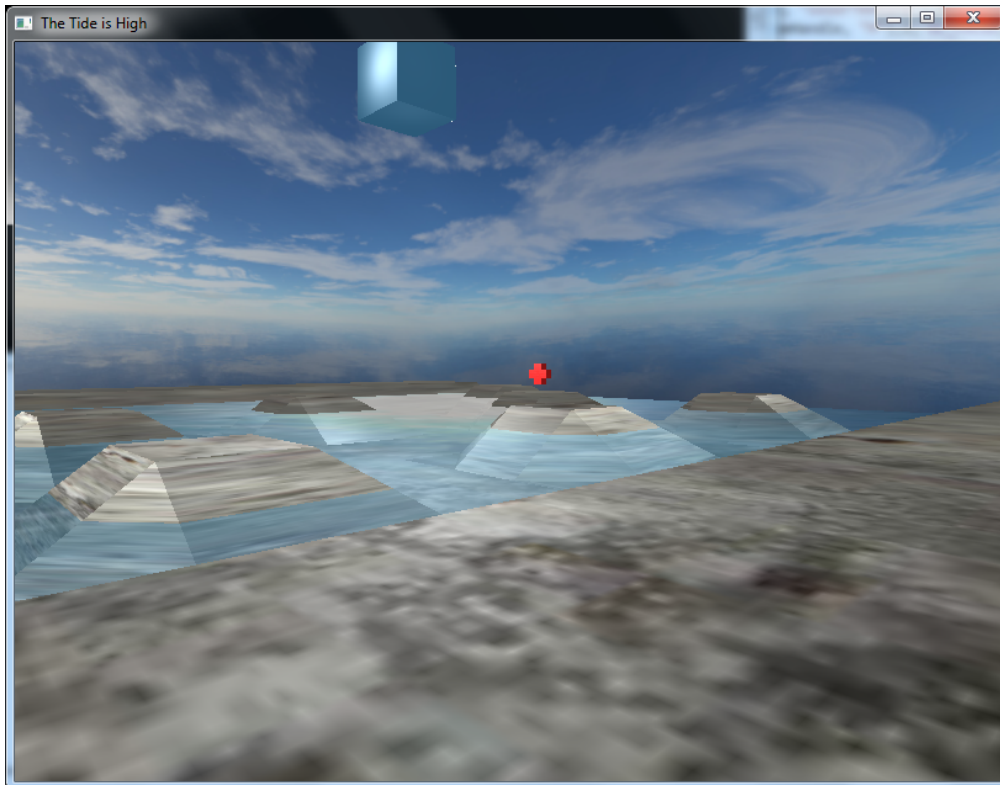


An underwater photograph showing a sandy seabed with scattered rocks and coral. The water is clear and blue, with sunlight filtering through the surface, creating ripples and shadows on the bottom. Three prominent rocks are visible near the surface, and a larger, more complex coral structure is visible in the background.

The Tide Is
High

Optischer Vergleich zur letzten Abgabe

Bei unserer letzten Abgabe sah unser Spiel wie folgt aus:



Inzwischen sieht unser Spiel so aus:



Außerdem kann unser Spiel inzwischen mit Hilfe einer **Config**-Datei auch im **Full Screen** Modus angezeigt werden.

Kurze Beschreibung der Implementierung

Gameplay

„**The Tide is High**“ ist ein Singleplayer Jump’n’Run Game, indem es darum geht vor der bevorstehenden Flut zu flüchten und die andere Uferseite zu erreichen.

Die Spielfigur kann nicht schwimmen, daher muss sie sich über Plattformen, welche aus dem Wasser zwischen den beiden Ufern ragen, auf die andere Uferseite kämpfen.

Diese Plattformen sind verschieden hohe Hügel, welche mittels einer **Heightmap** berechnet und erzeugt werden. Diese Heightmap haben wir in Blender erstellt (http://wiki.blender.org/index.php/Doc:2.4/Tutorials/Textures/Maps/Creating_a_Height_map_from_a_Plane).

Eigentlich handelt es sich dabei aber nur um ein Überlebenstraining, wobei ein würfelförmiger Raum zu einer Illusion von Himmel, Wasser und Plattformen gemacht dient. Die große Kugel in der Mitte stellt den Mond dar, weshalb die Flut bald kommen wird. Wir haben unsere Welt eher klein und simpel gehalten, um das Spielprinzip einfach erkennbar und gut sichtbar zu machen.

Der Spieler hat zu Beginn drei Leben, welche durch die drei Herzen in der rechten oberen Ecke symbolisiert werden. Bei jedem Sprung ins Wasser verliert der Spieler eines davon. Wenn alle drei Leben verloren wurden wird „Game Over“ angezeigt. Der Spieler kann dann aber sofort mit drei neuen Leben von vorne beginnen. Das rote Plus dient nur zum Zeigen der Effekte (Reflexion, Schatten) und hat keine Auswirkung auf das Gameplay.

Controls

Unser Spiel kann mit Hilfe der Tasten W,A,S und D, sowie der Leertaste gesteuert werden. Um die Bewegungen frame-unabhängig zu halten werden alle Bewegungen mit ΔT berechnet.

Mit Hilfe der Maus kann man sich im Spiel umsehen, sprich die Kamera bewegen. Außerdem kann man sich in Richtung der aktuellen Blickrichtung vorwärts und rückwärts bewegen.

Eingabe	Effekt
Mausbewegung	Umsehen
W	Bewegung nach vorne
A	Bewegung nach links
S	Bewegung nach hinten
D	Bewegung nach rechts
Leertaste	Springen
F2	Frame Time on/off
F3	Wire Frame on/off
F4	Textur-Sampling-Quality: Nearest Neighbor/Bilinear
F5	Mip Mapping-Quality: Off/Nearest Neighbor/Linear
F6	Lichtquelle rotieren on/off
F9	Transparency on/off

Effekte

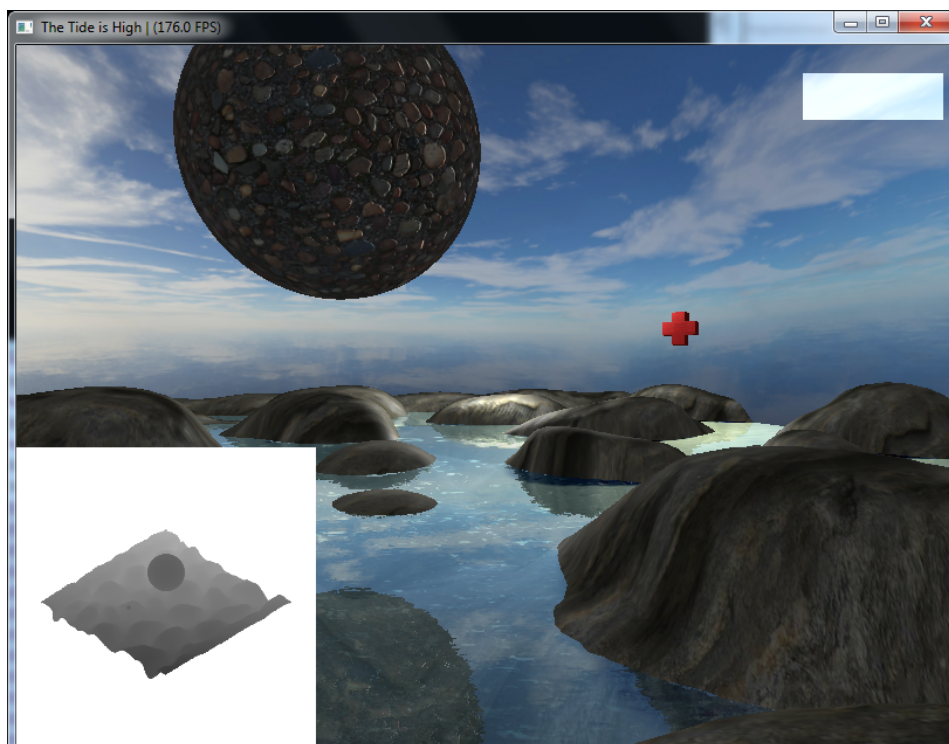
Wir haben uns folgende Effekte aus der Effekt-Liste ausgesucht:

Shadow Maps (with PCF)	1.5	As presented in the lectures, shadow maps calculate the shadows via rendering the scene from the light source. PCF samples the shadow map several times to improve quality. Make sure to counter all artifacts.
Water (+Fresnel-Shading, Normal Mapping)	0.5	Render a water surface - with moving waves using the - fresnel-shading-algorithm and - axis-aligned normal mapping.
+ Reflection	1	Your water reflects environment, shadows, and moving objects based on the waters normal map
Normal Mapping	1	Alter the normal vector by reading an value from a texture (=normal map). It requires the understanding of Tangent Space (which should be used for the calculations). This effect only counts if it is applied to complex objects and not on axis-aligned quads or boxes!

Shadow Maps (with PCF)

Zum Testen ist die Taste F6 sehr hilfreich, welche das Licht rotieren lässt.

Für das Shadow Mapping haben wir im ersten Pass die Tiefenwerte von der Lichtquelle aus (bzw. aus Sicht der Lichtquelle) mittels Frame Buffer Object (FBO) in eine Textur gerendert, wie in folgendem Screenshot zu sehen.



Anschließend haben wir diese Information an unseren Shader übergeben und dort die Position und Stärke der Schatten berechnet. Die Schatten werden dabei öfters gesampled (PCF).

Auf folgendem Screenshot sind die Schatten der Kugel, des Kreuzes und einiger Hügel sehr gut sichtbar.



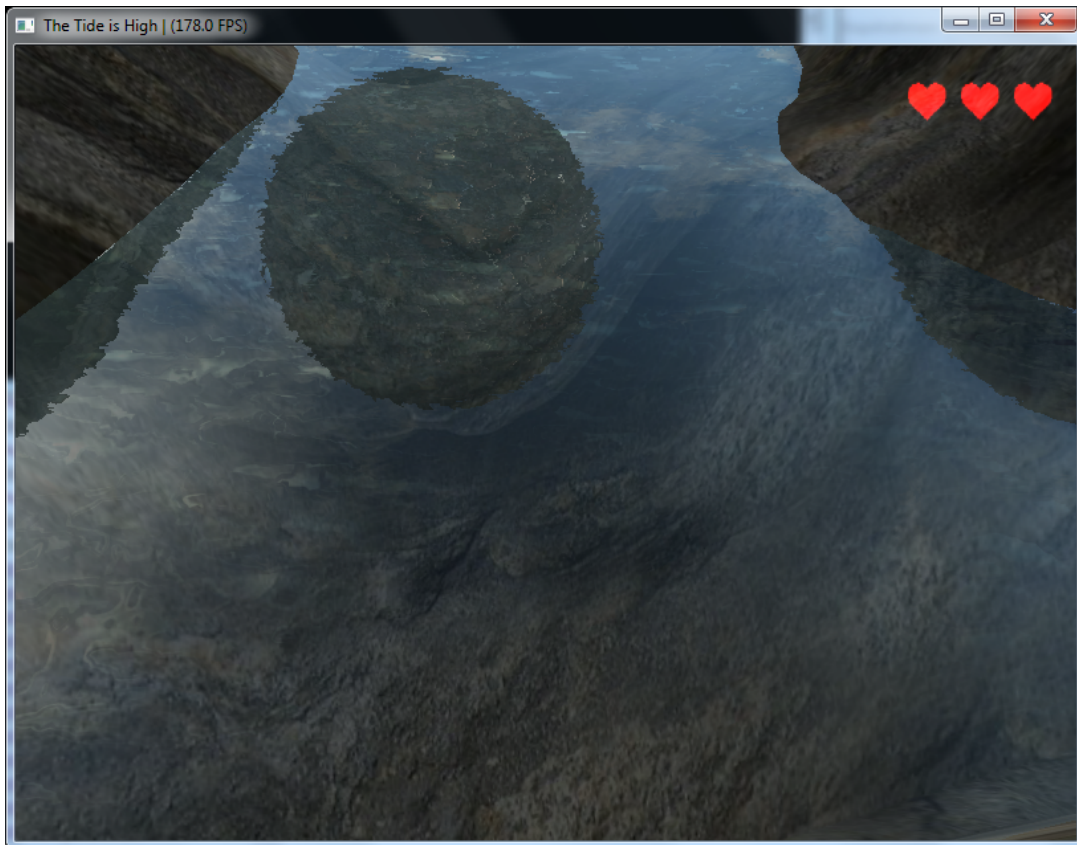
Quellen:

- Folien zu Shadow Mapping
- <http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-16-shadow-mapping/>

Water (+Fresnel-Shading, Normal Mapping)

Unser Wasser besteht aus einem dünnen hellblauen Quader. Der Eindruck von Wellen wird mittels Normal Mapping erzeugt. Mit Hilfe der Zeit-Variablen werden die Koordinaten der Normal Map bewegt, wodurch die bewegten Wellen entstehen.

Damit die Transparenz des Wassers realistischer wirkt wird diese mit Hilfe des Fresnel-Terms je nach Blickwinkel bestimmt. Folgende Abbildung zeigt die erhöhte Transparenz bei steilerem Blickwinkel.

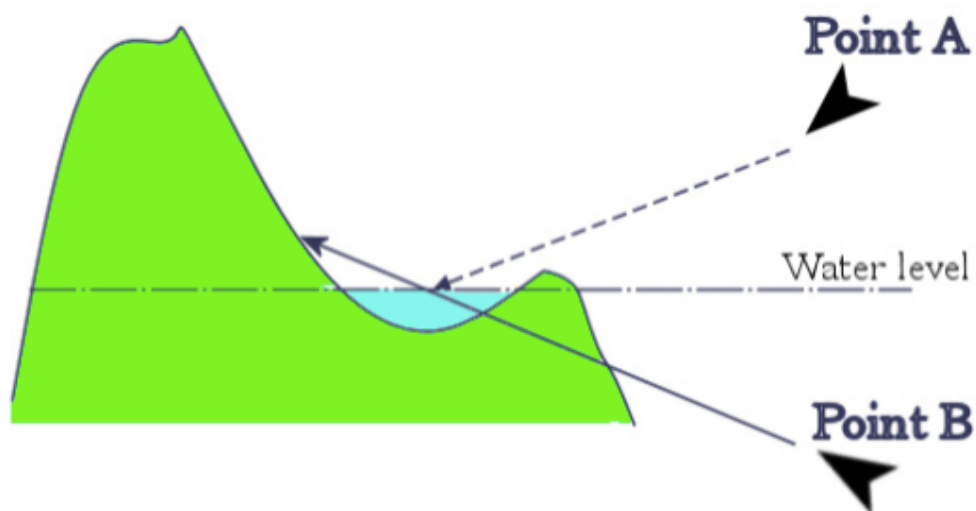


Quellen:

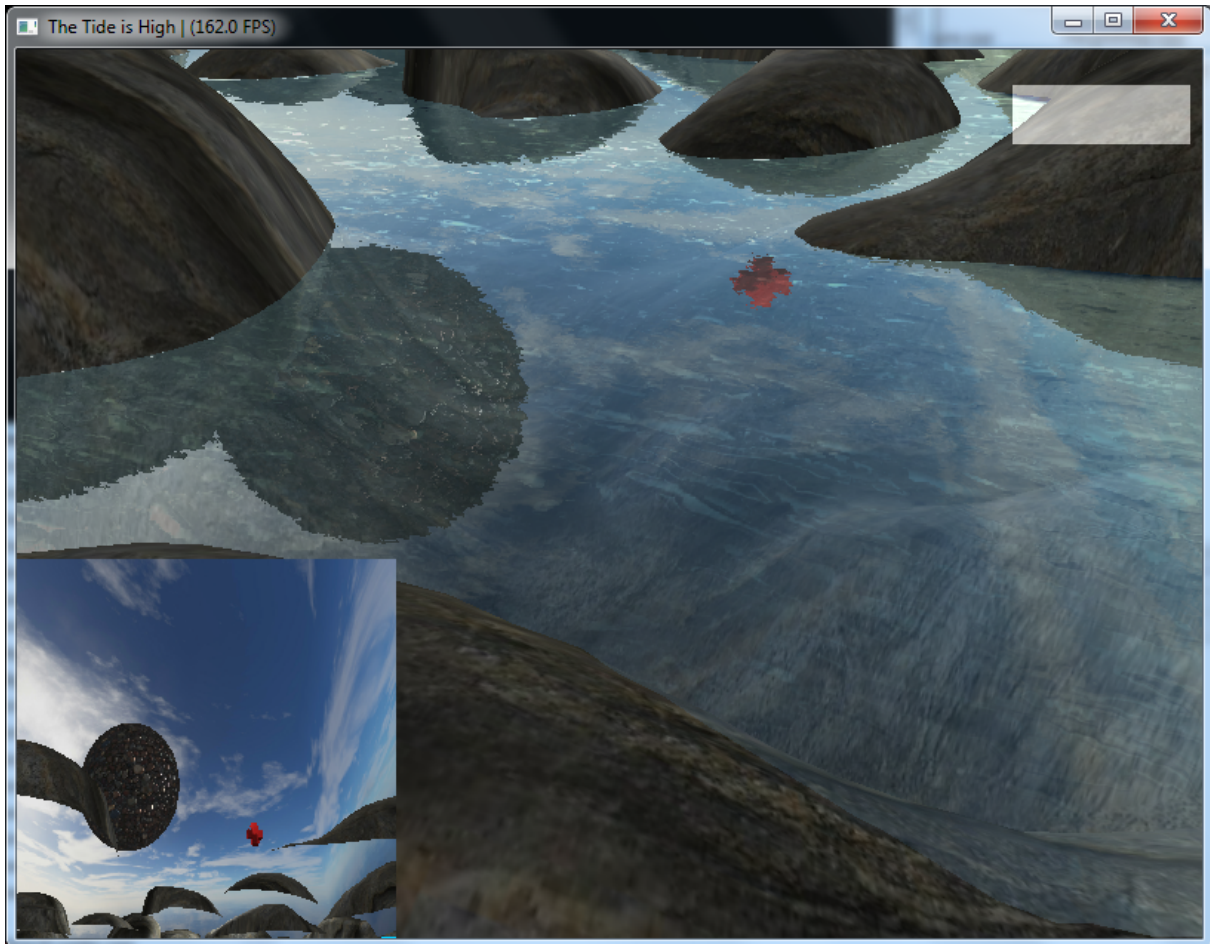
- <http://www.virtual-vision.net/19.html>
- <http://29a.ch/slides/2012/webglwater>
- <http://habibs.wordpress.com/lake/>

Reflektion

Für die Reflexion verschieben wir unsere Kamera unter die Wasseroberfläche, wie in folgender Abbildung zu sehen (Quelle: <http://habibs.wordpress.com/lake/>).



Aus dieser Perspektive wird dann unsere Welt oberhalb der Wasseroberfläche mittels FBO in eine Textur gerendert, wie in folgendem Screenshot zu sehen.



Anschließend berechnen wir in unseren Shadern die richtige Position der Reflexionen. Auch die Reflexionen von Kugel, Kreuz und Hügeln sind im Screenshot oberhalb gut zu erkennen.

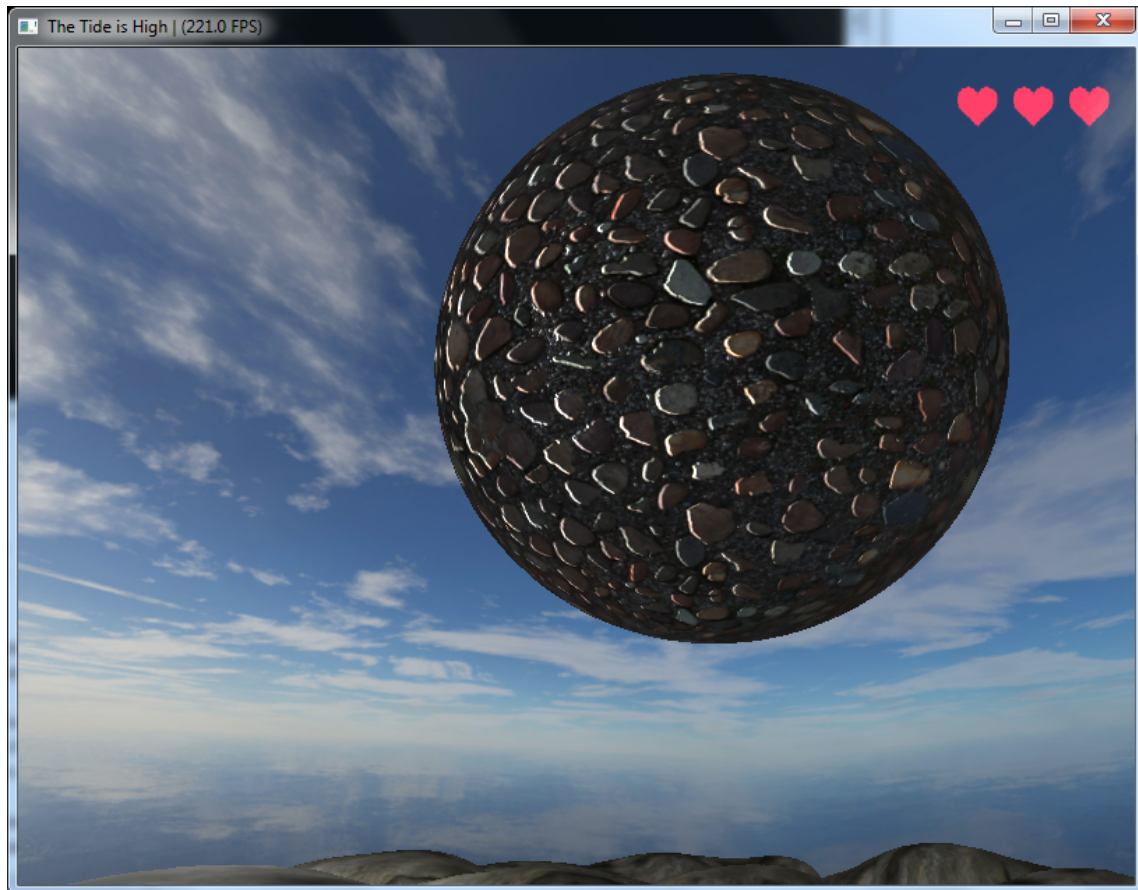
Quellen:

- <http://www.virtual-vision.net/19.html>
- <http://29a.ch/slides/2012/webglwater>
- <http://habibs.wordpress.com/lake/>

Normal Mapping

Zum Testen ist die Taste F6 sehr hilfreich, welche das Licht rotieren lässt.

Die große schwebende Kugel in der Mitte unsres Spiels stellt den Mond dar. Diesen haben wir mit einer Steintextur versehen, um das Normal Mapping deutlich darzustellen. Für das Normal Mapping wird zu der Textur die passende Normal Map Textur benötigt. Aufgrund der Oberflächennormalen des Meshes werden dann die Tangenten und Bitangenten berechnet und in Kombination mit den Werten der Normal Map Textur die beleuchteten Stellen erzeugt. Obwohl das Normal Mapping auf der Kugel dynamisch besser erkennbar ist, ist es auf dem Screenshot unterhalb auch zu erkennen.



Quellen:

- <http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-13-normal-mapping/>
- http://www.swiftless.com/tutorials/gsl/8_bump_mapping.html
- <http://ogldev.atSPACE.co.uk/www/tutorial26/tutorial26.html>

Experimentieren mit OpenGL

„Advanced Functionalities of OpenGL should be implemented“

- Vertex-Buffer-Objects (VBO) ✓
- Vertex Array Objects (VAO) ✓
- Buffer-Objects: FBO (Frame Buffer Object) ✓
- Mip Mapping (on/off) ✓
- Textur-Sampling-Quality (Bi/Trilinear Filtering) ✓

Unsere Funktionalitäten zum experimentieren mit OpenGL sind wie gewünscht unter folgenden Tasten verfügbar und es werden jeweils Feedback-Messages angezeigt.

F2	Frame Time on/off
F3	Wire Frame on/off
F4	Textur-Sampling-Quality: Nearest Neighbor/Bilinear
F5	Mip Mapping-Quality: Off/Nearest Neighbor/Linear
F6	Lichtquelle rotieren on/off
F9	Transparency on/off

„How and which objects were illuminated (description of light sources) or textured.“

Texture Mapping

Die Objekte und Texturen laden wir mit Hilfe der **Assimp-Bibliothek** und der **FreeImage-Bibliothek**. Um geladen werden zu können, muss jedes der verwendeten Objekte die Position, Indizes, Normalen und UV-Koordinaten beinhalten.

Wir haben unsere Models mit dem **3D-Modellierungsprogramm Blender** erstellt und beim Exportieren darauf geachtet, dass all diese Werte im Objekt abgespeichert werden. Außerdem haben wir die Option „Triangulate Faces“ gewählt, damit unsere Objekte nur aus Dreiecken gerendert wird.

Die Texturen werden in unserer Projektstruktur in der Klasse „Texture“ erstellt, wo auch die gewünschten Bilder für die Texturen geladen werden. Hier wird auch eine MipMap generiert und je nach Anwendungsfall der Minimum- oder Maximum-Filter ausgewählt. Beim Rendern der Schrift, welche beim Erreichen des Ziels eingeblendet wird, hat sich herausgestellt, dass die Verwendung von GL_NEAREST geeigneter als die von GL_LINEAR ist.

Für das Rendern der Schrift haben wir die Klasse „Text“ erstellt. Hier werden die Positionen der gewünschten Buchstaben in einem Bild mit einer Auswahl an diversen Buchstaben berechnet und anschließend angezeigt.

Die SkyBox ist ein Würfel, welcher auf den inneren Flächen eine Textur hat, welche Himmel darstellt. Um das zu bewerkstelligen, haben wir die Normalen nach innen gedreht.

Lighting

In unserem Spiel gibt es eine Lichtquelle, welche die Sonne darstellen soll. Von dieser Lichtquelle werden alle Meshes bis auf die Skybox beleuchtet. Wir haben für die Lichtquelle derzeit die Position (2,2,2) gewählt.

Die Eigenschaften der Materialien, wie beispielsweise der ambiente, der diffuse und der spiegelnde (specular) Anteil, wird bei uns im Shader festgelegt und nicht direkt über das Objekt-File gelesen. Zur Zeit haben die meisten unserer Modelle dieselben Materialeigenschaften. Diese können über uniforms im Shader angepasst werden. Weiters sind zu jedem unserer Modelle die Normalen gespeichert, wodurch die passende Beleuchtung berechnet werden kann.

„What additional libraries (e.g. for collision, object-loader, sound, ...) were used, including references (URL)?“

- Assimp-Bibliothek (<http://assimp.sourceforge.net/>)
- FreeImage-Bibliothek (<http://freeimage.sourceforge.net/>)

„What Tools have you used to create the Models (Maya, 3DS MAX, ...).“

- Blender

Generelle Quellen/Referenzen:

- <http://www.spieleprogrammierer.de/15-2d-und-3d-grafik/18444-text-ausgeben-rendern-mit-opengl/>
- <http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-11-2d-text/>
- <http://www.codehead.co.uk/cbfg/>
- [http://www.spacesimulator.net/wiki/index.php?title=Tutorials:Bitmapped_Fonts_\(OpenGL3.3\)](http://www.spacesimulator.net/wiki/index.php?title=Tutorials:Bitmapped_Fonts_(OpenGL3.3))
- <http://freeimage.sourceforge.net/>
- https://www.opengl.org/discussion_boards/showthread.php/164502-freeImage-example-code
- http://nehe.gamedev.net/tutorial/texture_mapping/12038/
- <http://www.lighthouse3d.com/cg-topics/code-samples/importing-3d-models-with-assimp/>
- <http://assimp.sourceforge.net/>
- <http://www.mbsoftworks.sk/index.php?page=tutorials&series=1>