

## Documentation Loch Ness Safari (Submission 2)

### Gameplay

As a diver you dive through Loch Ness underwater. It is possible to "take a photo" of Nessie (left Mouse Key). When the gamer does that, the distance between Nessie and the Camera is checked. It needs to be lower than a certain threshold (4) for the photo to be of sufficient quality. Furthermore, Nessie needs to be in the View Frustum of the Camera. If such a good photo is achieved, the game is won (as displayed on the text console). In the beginning, the gamer has 5 photos left to take on the SD card. If all of them are misses, the game is lost. By activating the flash light, a higher distance (threshold \* 1.5) is acceptable.

From time to time a bonus SD card can float down through the water. If close enough to the bonus SD card, the gamer can pick it up (by pressing E). This results in having more photos left.

By pressing the space key the "under water flash light" can be toggled on/off. It can only remain lit for a certain amount of time. If this time is over (times are accumulated correctly, if it is toggled off in between), it is automatically switched off and cannot be toggled on again.

#### Features:

- The objects move in a randomized way with their own (very simple) artificial intelligence (This concerns Nessie and the fish underwater, plus some rubber ducks on the surface of the lake).
- In both game end cases (winning and losing) the game stops. The command line output as well as graphical text output gives feedback whether a photo was sufficient or not. (The stopped game represents the final photo.)
- From time to time a bonus SD card comes floating from the top to the bottom. It remains lying there for some seconds, before falling down again at another, randomized, position.
- When the bonus SD card is picked up by the diver, it vanishes and doesn't appear again in the game.
- Every few seconds a sonar sound helps with finding Nessie: The louder the sound, the closer the proximity to the monster.

## Effects

### Shadow Mapping

The effect was implemented, using a depth map via two rendering passes and also using PCF to counter all artifacts. The shadows can be seen for example when fish are swimming above the ground. Shadow Mapping was implemented according to the following Tutorial:

<http://learnopengl.com/#!Advanced-Lighting/Shadows/Shadow-Mapping>

### Caustics

A caustics effect was implemented using animated textures, that were projected to the ground. Therefore, the effect is a combination of animated textures and projected textures.

The texture images used for the animated texture are downloaded from:

<http://www.dgp.toronto.edu/~stam/reality/Research/PeriodicCaustics/index.html>

For implementing projected textures the following articles were used:

<https://www.opengl.org/archives/resources/code/samples/mjktips/projtex/index.html>

[http://www.ozone3d.net/tutorials/gls/ texturing\\_p08.php](http://www.ozone3d.net/tutorials/gls/ texturing_p08.php)

### Lightshafts

A lightshafts effect (volumetric light scattering) was implemented as a post processing effect. It can be toggled on/off by pressing F7. The effect can be seen when the camera is tilted upwards to the sky / light source, especially when swimming closer to the top.

The effect is based on:

[http://http.developer.nvidia.com/GPUGems3/gpugems3\\_ch13.html](http://http.developer.nvidia.com/GPUGems3/gpugems3_ch13.html)

<http://fabiansanglard.net/lightScattering/>

Variation to the algorithm: The effect is only being activated when the sun is within (or close to) the view frustum.

### Spotlight

In the spotlight effect the surface is limited with cones (outer and inner) for a smooth transition. This effect can be activated by pressing SPACE. (Remember that the usage of the spotlight is limited - it can only be turned on for a total of a few seconds!)

References:

<http://learnopengl.com/#!Lighting/Light-casters>

## Additional Features / Implementation Details

### Lighting and Materials

The scene is illuminated by a main light source coming from the direction (0, 20, -1).

There are two different materials in the game, mostly differing in their shininess. Nessie and the bonus object (SD-Card floating from the top) on the one hand reflect the light in a shinier way than the ground on the other hand.

Additionally, a spotlight exists that can be toggled by pressing Space.

The implemented shader uses a Blinn-Phong Lighting Model.

## Fog

The whole scene is affected by a fog effect. This effect is calculated as a linear fog interpolation in dependence of the view space.

## Animated Objects

A few different kinds of fish exist within the lake. Their back part (flapper) is animated using hierarchical modelling. The faster a fish is moving, the faster the flapper will flap.

## Frustum Culling

View Frustum Culling was implemented using the geometrical approach. It can be toggled on/off by pressing F8.

References:

<http://www.lighthouse3d.com/tutorials/view-frustum-culling/>

## Transparency

The surface of the lake consists in a transparent (animated) texture.

## Collision Detection

Collision Detection is implemented with a Height Map. The Height Map itself was created in Blender by applying a height-dependant texture and rendering from above in an orthographic projection. The height map image was then imported using DevIL (as all textures), read in and collision checks implemented manually.

## Controls

- Change the viewing direction via mouse movement
- Move forward, left, backward and right with W, A, S, D
- Activate the "underwater flash light" with M or Space
- Take a photo with the left mouse button or P
- Pick up bonus objects (the SD card) by pressing E
- 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 - Render depth map used for shadow mapping
- F7 - Lightshafts on/off
- F8 - Viewfrustum-Culling on/off
- F9 - Transparency on/off

## Libraries used

- GLFW: <http://www.glfw.org/>
- GLEW: <http://glew.sourceforge.net/>
- Assimp: <http://assimp.sourceforge.net/>
- DevIL: <http://openil.sourceforge.net/>

- FreeType: <https://www.freetype.org/>
- irrKlang: <http://www.ambiera.com/irrklang>

## Resources

The model for Nessie is from: <http://tf3dm.com/3d-model/an-16082.html>

All other models were created manually using blender.

Underwater Sound and Sonar Sound recorded by Mike Koenig (Attribution 3.0 License):

<http://soundbible.com/1660-Underwater-Pool.html>

<http://soundbible.com/1183-Sonar.html>

## Additional References:

Text Rendering with FreeType:

<http://www.learnopengl.com/#!In-Practice/Text-Rendering>

Camera Transformations, Controls:

<http://learnopengl.com/#!Getting-started/Camera>

<https://gist.github.com/23ars/4545671>

Lighting / Materials:

<http://www.opengl-tutorial.org/beginners-tutorials/tutorial-8-basic-shading/>

<http://in2gpu.com/2014/06/19/lighting-vertex-fragment-shader/>

Coordinate Transformations, etc.:

<https://www.opengl.org/archives/resources/faq/technical/transform> HYPERLINK

"<https://www.opengl.org/archives/resources/faq/technical/transformations.htm>" HYPERLINK

"<https://www.opengl.org/archives/resources/faq/technical/transformations.htm>" HYPERLINK

"<https://www.opengl.org/archives/resources/faq/technical/transformations.htm>" ations.htm