Project N.A.R.W.H.A.L.

1. Gameplay

1.1. Goal

The goal is to eat all the nuclear waste before the timer runs out. The player also needs to make sure, not to touch any of the fishing hooks. Points can be earned for eating waste and are lost for touching a hook. The objects are spawned randomly when the game starts.

1.2. Controls

W	move forward
S	move back
A	move left
D	move right
X	move down
SPACE	move up
Mouse	rotate whale
M	mute sound
P	start particle effect (demo)
F1	display help

All function keys are bound as the requirements suggest.

2. Features

2.1. Animated Objects

The fin and the whale itself are two different objects. When the whale moves, the fin moves and rotates up and down. Its position changes relative to the position of the body of the whale.

2.2. View Frustum Culling

In every loop pass, the position of the view frustum is calculated to match the newly calculated camera. The position is then used to calculate 6 panes. We use a spheric hit box for each object so we have to find out if a sphere is outside or inside the hitbox (this can easily be done by calculating the signed distance from a point to a pane (assume

the normals are pointed inwards). If the object is outside every pane of the hitbox, we simply ignore it for the whole loop pass.
Reference:
http://cgvr.cs.uni-bremen.de/teaching/cg_literatur/lighthouse3d_view_frustum_culling/ (22.06.2015)

2.3. Transparency

The transparency of all objects can get turned on and off. For the png images of the images the alpha channel gets used.
Transparency is also used to render text smoothing.

2.4. Experimenting with OpenGL

VBO, VAO, FBO are used in the project.
The required features can be turned on or off via following keyboard buttons:

F2	Frame Time on/off
F3	Wire Frame on/off
F4	Texture-Sampling-Quality: Nearest Neighbor/Bilinear
F5	Mip Mapping-Quality: Off/Nearest Neighbor/Linear
F8	View Frustum-Culling on/off
F9	Transparency on/off

2.5 Text Rendering

We use the FreeType library to render text onto the screen. Every glyph is extracted from the open type font at the beginning. Currently, only one font size is natively used. The different font sizes are only up scalings of the generated bitmaps.
Reference:
http://learnopengl.com/#!In-Practice/Text-Rendering (22.06.2015)

2.6 Random Object Spawning

When the game is started, a predefined number of good and bad objects are spawned randomly throughout the map. The spawning area is defined between two bounds: An outer bound where the hero can't move (so the hero can't move underneath the earth or outside the map) and an inner bound (which is around the spawn point of the hero). This way, every level has to be played a little different.

3. Illumination and Textures

All Objects are illuminated by one light source. The light source is placed in the middle of the field and its effect is visible on all objects. The illumination effect is implemented after the phong

lighting scheme. The lighting effect is visible on the narwhal and its fin. The position and direction of this light source is also used for creating the shadow maps.
The narwhal, its fin, the floor, the nuclear waste and the particles use textures.

4. Effects

4.1. Shadow Maps (with PCF) 1.5p Sebastian

The shadow maps use the same light source as is used for creating the lighting effect for the objects. We used a directional shadow maps effect. We started to implement omni directional light sources, but they did not make it in the final build.
Reference:
http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-16-shadow-mapping (20.06.2015)
http://www.learnopengl.com/#!Advanced-Lighting/Shadows/Shadow-Mapping (20.06.2015)

4.2. CPU particle effect 0.5p Sebastian

Many instances of a PNG image of a bubble with alpha channel get created. The size and direction of the particles get modified. We chose not to change the color. The particles are visible as bubbles on top of the hero whenever the whale feeds on nuclear waste. We had to make sure, that the particles are the last objects to be drawn, or else the transparency would not work correctly.
Reference:
http://www.opengl-tutorial.org/intermediate-tutorials/billboards-particles/particles-instancing/ 20.06.2015

4.3. Glow 1 p Philipp

Objects can be marked as glowing. After the default object rendering pass, all objects will be drawn to a frame object buffer again, this time, only objects that are marked as glowing will be drawn and the other objects will just pass the depth information to the frame buffer object. The resulting image only shows the glowing objects that are actually visible by a camera. The resulting frame buffer is then rendered to both a horizontal and then a vertical gaussian blur shader. The resulting, blurred, image is then added to the default rendering image using `glBlendFunc(GL_ONE, GL_ONE)`
Reference:
http://http.developer.nvidia.com/GPUGems/gpugems_ch21.html (05.08.2007)

4.4. Environment Mapping 1 p Philipp

We use a sky map that is mapped onto a cube while drawing the skybox. This texture is passed to the environmental mapping shader. The shader will calculate the reflection based on the camera and the normal vector. The reflection vector is used to read the corresponding texel in the cube map and thus displays the reflected skybox on its surface.
The environmental mapping is used to get a chromatic look for the fishing hooks. There is also a demo cube somewhere on the map (underneath the hero spawn) that demonstrates the environmental mapping a bit better (the hooks are rather thin).
Reference:

http://www.learnopengl.com/#!Advanced-OpenGL/Cubemaps (21.06.2015)

5. Libraries

- Model loading: **assimp** (http://assimp.sourceforge.net)
- Texture loading: **freeImage** (http://freeimage.sourceforge.net)
- Audio handling: **irrKlang** (http://www.ambiera.com/irrklang)
- Font handling: **freetype** (http://www.freetype.org)
- **GLEW** (http://glew.sourceforge.net)
- **GLFW** (http://www.glfw.org)

6. Special Features

All models and sound effects (not the background music) were created by us. The fin of the whale is animated. When the hero eats nuclear waste, bubbles burst from his back. Objects spawn randomly every time you start the game.

7. Tools

The models for the narwhal and its fin, the floor, the nuclear waste and the fishing hook where created with blender. After that they were exported as collada files, which are used by the game.
The texture for the narwhal was edited with gimp.
The sound effects were edited with reaper.

8. Interaction

The game is pretty simple. Hit the green fuel rods and try not to hit the fishing hooks. Because the level is different every time you start the game, there is no “guide” to win the game.