Submission 2– WirSindGroot

Gall Alexander, 1225540

Heim Anja, 1226809

Description of implementation:

As requested our demo has automatic camera movement. As illumination model we use the “Phong Illumination Model”, which is implemented in our shaders ShadowMapping.vs and ShadowMapping.fs.

All drawn objects in our project support this illumination model and cast shadows. Our objects are all models of type Wavefront (.obj) and the textures are all .png-files or .jpg-files. The objects were modelled with Blender or taken from the website: “cgtrader” (“https://www.cgtrader.com/”).

We have a landscape with a forest, where the trees cast shadows implemented through omnidirectional shadow mapping. This light is cast by the firefly flying through the scene. Shadow Mapping is implemented by drawing a depth image of the scene from six different camera directions from the lights position. These six images are then stored in a cubemap. This is implemented in the files DepthMapShader.vs, DepthMapShader.gs and DepthMapShader.fs. Then the whole scene is rendered again and this time the light calculations take this cubemap into account to produce correct shadows. As can be seen in the demo the shadow calculation is dynamic. The implementation of omnidirectional shadow mapping is used from the lecture.

We also have different types of axes and shields which all support Normal Mapping. Also a skeleton can be seen which uses Normal Mapping. The implementation of this effect can also be found in the files ShadowMapping.vs and ShadowMapping.fs. Here the tangents of the models are read by the library Assimp and are used in the shaders. As heard in the lecture the bitangents are calculated out of the tangents and the normals. Then these values are used to produce a matrix which is used to bring the fragment’s position and the camera position in the correct space. Then the illumination is calculated in this tangent space.

On the metal of the axes the texture of the model is mixed with Environment Mapping. At the end of the demo we also positioned a sword which mixes its own textures with Environment Mapping. Environment Mapping is implemented in the shaders EnvironmentShader.vs and EnvironmentShader.fs. For this effect the position of the object which uses environment mapping is used. From this position the scene is captured from six different directions. These directions are then stored in a cubemap. This cubemap is the used as texture in the main rendering process.

Furthermore we implemented fog, which is hiding all objects at the beginning. During the demo, the fog lightens until in the end it gets thicker again and swallows the whole scene. The fog effect is also implemented in the files ShadowMapping.vs and ShadowMapping.fs. For each object in eyespace the distance from the camera is calculated and then a grey value is mixed with the colour of the object which is dependent on an exponential function to simulate an atmospheric effect.

We also added Frustum Culling to enhance the computational speed (amount of FPS). Also the animation is frame independent.

Additional libraries:

In our project we use rendering with OPENGL 4.4. Our models are loaded by using the library "Assimp" (https://github.com/assimp/assimp), it is similarly implemented as in the OpenGL tutorial https://learnopengl.com, but with many changes. For texture loading we use the library devil (http://openil.sourceforge.net/download.php). We also use GLFW (http://www.glfw.org/download.html) and GLM (https://glm.g-truc.net/0.9.8/index.html) in our project.

Graphics Card:

We tested our project on Nvidia cards. In the Lab we tested our project on Ghost and we had at least 200 FPS.

Note:

The start of the demo can take longer depending on the hardware, since Assimp takes longer to load high quality models. (Maximum expected loading time is 2 minutes)