Real Time Rendering WS19

Submission 2 – Final Demo

Still Dreaming (Group 12)

Michaela Tuscher, 0827032, 033 532

Requirements

OpenGL 4.3 Core Profile, Windows 10.

Illumination

For illumination the Phong lighting model was used. It is possible to set the specular part and shininess individually per object, e.g. eyeball and lamp are shinier than the other objects. Also attenuation is implemented so the objects get darker the farther they are away from the light.

Effects

The implemented effects are:

- Reflections with Real-Time Environment Mapping
- Shadow Mapping with Percentage Closer Filtering
- Edge Detection as Post-Processing Effect

All effects were implemented as described in the lecture slides.
For PCF the descriptions from [Eisemann, Elmar, et al. *Real-time shadows*. AK Peters/CRC Press, 2016] were used, especially chapter 5.1.4, the paragraphs about Regular Nearest Neighbor Sampling and Poisson Disk Sampling.
For Edge Detection the Sobel kernels and formula found on wikipedia were used. (https://en.wikipedia.org/wiki/Sobel_operator)

Real-Time Environment Mapping
The environment map is created by rendering a cubemap from the sphere‘s position with 6 different view vectors for each side of the cube. The perspective projection matrix uses 90 degrees FOV for each direction. The cubemap is rendered in 1 pass by using a geometry shader to render to all 6 sides of the cube. The resolution is 1024x1024 for each side.
Aside from this it is rendered like the normal scene with small differences in the shadow calculations (see point Shadow Mapping) and for cosmetic reasons it is rendered with different ambient and diffuse values so it is brighter than the scene and stands out more.
A reflection vector is calculated according to the formula from the lecture slides to sample from the environment cubemap. At the end of the demo, when the reflective sphere starts to change its size,

the normals do not get normalized anymore to get an effect as if the sphere is swallowing up the eyeball.

Shadow Mapping

In the demo there are two light sources, the moon and the lamp. The moon is implemented as directional light. Therefore a 2D depth texture is used for the depth map which is rendered from the moon‘s position looking at the scene center and using an orthogonal projection matrix. So all light rays are in the same direction. The resolution of the depth map is 1024x1024.

The lamp is implemented as point light, so it uses a depth cubemap rendered in 5 directions of the lamp‘s light position (using 5 different view matrices) and using a perspective projection matrix with 90 degrees FOV for each direction. The upper side of the depth cubemap is ignored in this process because of the shape of the lamp (everything above the light‘s position is in shadow). It is also rendered in 1 pass by using a geometry shader. The resolution is again 1024x1024 for each side. For this shadow map since it is a cubemap the depth is stored as length of the vector between fragment and light source.

The shadow maps are used for rendering shadows in the scene and the environment map. To determine if a fragment is lit by the moon, its position in light space is calculated and its depth then compared to the value in the depth map. If the fragment‘s depth is greater than the value in the depth map, it is in shadow. Similar, for calculating the lamp shadows, the distance between the fragment and the light source is compared to the stored distance in the depth map. Both lights make use of a small bias, which is subtracted from the fragment‘s depth, to prevent shadow acne. It was found by testing with different bias values.

For both light sources‘ shadow calculations PCF was used to make the shadows softer. For the moon it was implemented by also looking up and comparing the shadow values of the 8 neighbouring texels in the shadow map. The values are added and averaged at the end. For the lamp‘s shadows a similar approach was used. 16 values in different directions in a small radius around the original vector are taken from the depth cubemap and compared to the fragment‘s depth. The final value is also averaged at the end.
The resulting shadow values are then multiplied with the light‘s diffuse and specular parts.

For shadows in the environment map the same calculations were used but different bias values were chosen. Those shadows are also calculated with PCF but for both lights the sample sizes are smaller. For the moon 6 values are taken to sample and compare to and for the lamp 8.

Edge Detection

Edge Detection is implemented by rendering the scene with shadows to a 2D texture, adding the edge detection effect in a shader and then displaying the texture on a full screen quad. To get the edge detection effect, the 2D texture is convolved with a horizontal and vertical sobel kernel. The results of the convolutions are squared and added and afterwards the squareroot is calculated.

Models/Textures/Sound

Simple models like floor and walls were created and textured by myself with textures from Total Textures.

Skybox Texture: www.custommapmakers.org/skyboxes/zips/ame_starfield.zip
Most of the other models are from https://free3d.com/. I had to change some of them slightly (i.e. eyeball (cut off face in front of iris), spider (changed color and removed hair on legs)).

Eyeball: https://free3d.com/de/3d-model/eyeball--33237.html
Barrier: https://free3d.com/3d-model/concrete-barrier-301778.html
Lamp: https://free3d.com/3d-model/old-table-lamp-304522.html
Mountain: https://free3d.com/de/3d-model/mountain-6839.html
Sphere: https://learnopengl.com/data/models/planet.rar
Rock 1: https://free3d.com/de/3d-model/3d-rock-901394.html
Rock 2: https://free3d.com/de/3d-model/low-poly-rock-4631.html
Tree 1: https://free3d.com/de/3d-model/tree-74556.html
Tree 2: https://free3d.com/de/3d-model/a-tree-88011.html
Statue: https://free3d.com/3d-model/statue-of-liberty-73656.html
Spider: https://free3d.com/de/3d-model/spider-animated-low-poly-and-game-ready-87147.html

Known issues: there will be an error message stating that a texture is missing. It‘s the texture of a faulty object, namely the concrete barrier, which is used twice. One of its walls is missing completely (or gets culled) and the texture for one side too. I left the object in the demo because with the automatic camera it is only shown from perspectives where it looks good.

The sounds are from unity asset store packages.
https://assetstore.unity.com/packages/audio/sound-fx/free-sound-effects-pack-155776
https://assetstore.unity.com/packages/audio/sound-fx/universal-sound-fx-17256

Resolution/FPS and Automatic Camera Movement

Automatic Camera Implementation

The movement is implemented by successively calling functions for moving forward, back, sideways or rotation specifying a timeframe in which this movement is executed as well as a speed and for rotations a rotation angle for x and y-axis.

The resolution of the demo is by default 1920 x 1080 fullscreen when starting it from the .exe file. Both resolution and fullscreen mode on/off can be changed by either starting it from command line with parameters or changing the .bat file which is in the \bin folder. It takes three arguments. The first two are for the screen resolution and the last one is for fullscreen mode. For example for starting it with a resolution of 1280 x 768 in windowed mode the command would be:
start Still_Dreaming_EZG19.exe 1280 768 false
It can be started in any resolution. For fullscreen mode the last parameter has to be „true“.

On my notebook (with Nvidia GeForce GTX 1060 Max-Q) the demo with automatic camera movement runs with 60 FPS with occasional very short drops to 58/59 FPS with the default resolution in fullscreen mode.
However, when I tested it on both Lab Graphic Cards there seemed to be more lags in the automatic camera rotations. When moving around with the debug camera, I did not experience those lags and there are no problems with the effects. Therefore I think that the automatic rotations are causing those problems. DeltaT is considered in the rotations, but maybe something else is wrong. I tried different things but was not able to solve this problem.
Also this leads to the problem, that the automatic camera might end up at different places, especially at the end of the demo. It is intended to rotate around the reflection sphere at the same height while looking at the sphere. However, on the Lab PC it happened that the camera ended up in

a tree at the beginning of the rotation or at the end. I think it was better on the AMD graphics card. All effects were still visible with the automatic camera, but the reflections and environment map are maybe not shown in a way I intended. Again, I did not experience such great differences concerning the camera position when testing on my notebook. When testing on my notebook it sometimes ended up at a slightly different place (a little bit too much on the left/right of the sphere or a little bit too high) but it was not as grave as on the Lab PC and most of the time it ended up looking good.

Controls

When starting the demo, the automatic camera movement is turned on. In automatic camera mode the demo will close as soon as it is finished. It can be closed immediately with the escape key.
The „O“ key can be pressed to make the scene brighter, the „L“ key can be pressed to make it darker.
To go into debug camera mode, „P“ can be pressed. The automatic camera movement will stop then and can not be continued.
The debug camera can be moved with the WASD keys and rotated with the mouse. The escape key closes the demo.
Pressing „E“ in debug camera mode toggles the post processing edge detection effect.
All moving parts of the scene will continue to move or start moving at a certain time in debug camera mode. Also the edge detection effect will be automatically triggered at a certain time in debug mode. The same accounts for all sounds. An exception is the scaling of the reflection sphere. This will not happen in debug camera mode.

Used Libraries

glm (matrices etc.): https://glm.g-truc.net/0.9.9/index.html
Assimp (model loading): http://www.assimp.org/
irrKlang (sound): https://www.ambiera.com/irrklang/
stb_image.h (image loading for textures): https://github.com/nothings/stb/blob/master/stb_image.h
glfw 3 (window creation etc.): https://www.glfw.org/
glad (extension loading): https://glad.dav1d.de/

Tested on

Nvidia and AMD Graphics Card in Vislab
Nvidia GeForce GTX 1060 Max-Q
Nvidia GeForce RTX 2070 Super
Nvidia GeForce GTX 960M