3D Project

DV1541

Walter Karlqvist
Henrik Vik
Hannah Lövberg

2017

Implemented Techniques

Parsing

Parsing is done by a file loaded class that parses obj-files into an instance of the ObjFile class. The class is then converted into a mesh. Paths to materia libraries are exported from the obj instance and parsed by MtlFile. MtlFile are converted into materials.

Gaussian Blur

This is a separate class that uses the ping-pong method to blur with a single shader containing an if-statement. It has a blur method that receives a texture and the amount of blur passes. The texture is blurred and returned.

Deferred Rendering

Our Deferred Rendering uses a framebuffer with four color attachments. Those are world position, normals, color and position in camera space (this is used for the SSAO). It also uses a framebuffer that that renders the light pass into a texture instead of the standard framebuffer, this is needed for our glow-effect.

Displacement Mapping

The tessellator uses the models' diffuse map instead of a separate displacement map. This is done by converting the rgb sample from the diffuse map into grey-scale. It is run during the deferred rendering geometry pass so that all the models receive tessellation.

SSAO

We use the crysis full sphere method. SSAO is done in a class containing the necessary framebuffers and receives a texture of the rendered scene with positions in view space from deferred rendering. It then blurs the SSAO map with the help of the gaussian blur class. The SSAO class returns the SSAO map which is then read by deferred rendering light pass.

Shadow Mapping

Shadow mapping is made up from two steps, where the first part is to create the shadow map, this is done by creating a depth buffer that we simply render the scene to in a very simplified manner. This will be done once for each light in the scene since it is done from the individual lights viewpoint.

This is mainly done in the ShadowMap class in our implementation.

Step two is to render the scene as normal, where you pass the depth buffer to the lightning calculation. Here you check for each fragment if that fragment is in shadow, by measuring the distance from the point to the light, and then converting the point into the light space and looking if the distance is the same as that point in the depth map.

We implemented this in the lightPass function of the DeferedRenderer class.

Front To Back Rendering

This is a simple technique that sort all objects from the closes to the furthest, and enables optimisation on the gpu side of things, this is done at the cost of cpu power. In practice the list of all objects in the scene are sorted by a bubble sort algorithm, since verry little change in the list from frame to frame.

This is implemented in the GameObjectManager class were the sorting algorithm and comparisons located in the FrontToBackSort function.

Back face culling, using Geometry shader

We've implemented back face culling in a geometry shader that is optional to include.

Glow Effect

It receives a texture of the rendered scene and all the 3D-models. It then renders the 3D-models but only samples the alpha channel of the textures to know if it glows. It then blurs the glow texture with the help of the gaussian blur class and merges the glow and the scene texture in the final result.

This is our final step in our pipeline. The final result is rendered to the screen with the help of a fullscreen quad.