- Deferred Rendering
    - We are using a gBuffer with four buffers. The last buffer, gColorInfo, has the material properties for each Object in the scene. We are using a quad to render the “images/textures”.
- Parsing and rendering an existing model format
    - Our model/object loader works by loading each row of an .obj and .mtl files. The information, Vertices and Materials is calculated and sent to the Mesh class.
- Height-map terrain rendering, user can walk on the terrain using WASD keys
    - Our height map Terrain is loaded from raw data and a height map image. This information is calculated and stored in a separate Mesh variable than the objects in the scene. The walking is calculated by each vertices in the terrain with Barycentric coordinate system.
- Normal Mapping
    - The normal mapping image is loaded in the .mtl file. The Tangent is calculated after the normal for the object is done. The bi-tangent is calculated in the Geometry Pass Geometry Shader. These are used in the Geometry Pass Fragment Shader.
- Shadow Mapping
    - An orthographic camera is created with the view perspective of the light in the scene. A depth map is created from this cameras perspective, and sent through the geometry pass to calculate shadows in the scene. The calculated shadow is stored in the w-channel of gNormal and sent to the lighting pass fragment shader to be drawn into the scene.
- Front to back Rendering
    - Front to back rending was done by taking the distance from the camera position to the object position and then comparing it with the distance from the camera position to any other object pos. The shorter the distance means that the object is closer to the camera so we sorted the object container so that the closest object gets rendered first.
- Back face culling using Geometry Shader
    - This is calculated in the Geometry Pass Geometry Shader. This is done by using the dot product on our camera position and a triangles normal.
- Gaussian Filter
    - The technique is that we make a copy of our texture in the lightning pass and send one of them to the Gaussian shader. The Gaussian shader then takes each pixel on the screen that requires blurring and stretches it out horizontally and then vertically.
- Glow-effect
    - When it’s finished blurring the scene with the texture that was sent in, it then passes that blurred texture to the merge shader which already holds the original copy of the texture, the not blurred one. The merge shaders purpose is to merge the two textures to create either a glow feature. Depending on the variables that was sent in to the textures you can get a fully blurred screen, every object glowing or just the normal glow effect on the texture parts that needs glow.