Students: Vendela Eklund 980812 & Erik Eliasson 980323 Date: 22-03-2019
Program: PAACI17
Course: DV1542

Students: Vendela Eklund 980812 & Erik Eliasson 980323 Date: 22-03-2019
Program: PAACI17
Course: DV1542

Table of Content

Students: Vendela Eklund 980812 & Erik Eliasson 980323 Date: 22-03-2019
Program: PAACI17
Course: DV1542

1. Techniques

1.1 Core Techniques

1.1.1 Deferred Rendering

The deferred rendering technique starts by creating two kinds of framebuffers, one for all the geometries and one for the “final scene”. It starts with creating the geometry-framebuffer and bind it while creating five g-buffers, the first stores positions, the second stores normals, the third stores colors, the fourth stores texture coordinates also known as uv, and the last one stores the depth. After creating and binding those textures to the geometry-framebuffer it creates the final-framebuffer and bind it for creation of the final-texture that is going to store all the information that has passed through the LightPass.

When everything is created it store information in all of the textures. Starting with GeometryPass, where it stores the information from all of our objects to the position-, normal-, color-, uv- and depth-texture by creating objects - one by one - and then store the information. After that is done the BeginLightPass and the LightPass begins, where all of our lights are calculated and information to the final-texture is created and stored, and finally put the final-texture on a fullscreen quad.

1.2 Geometry

1.2.1 Parsing And Rendering an Existing Model Format

This technique reads a .obj file and parse it into a renderable object that can be used. It also gathers information about a .mtl file in the .obj file where it gathers information about textures, kd and ks.

1.2.2 Heightmap Terrain Rendering

It uses a Image Loader library that is able to read and store information from a .bmp texture, most importantly pixel data. The heightmap is created by looping and reading pixels from the image and create heights depending on the value of every pixel, then it creates vertices with those heights and placing them all in a grid. Then connects all of the vertices with index data.

Students: Vendela Eklund 980812 & Erik Eliasson 980323 Date: 22-03-2019
Program: PAACI17
Course: DV1542

1.3 Texture And Lighting

1.3.1 Normal Mapping

Our normal map is created by changing the normals depending on the NormalMapping-texture located in the shader, then saves those normals to the deferred normal-texture.

1.4 Projection Techniques

1.4.1 Shadow Mapping

The shadow map that is used works on directional light and are using a orthographic view. The depth projection matrix is describing a axis-aligned bounding box with the values (-15,25),(-15,25),(-15,30) for its sides. The view matrix uses the directional lights inverse direction as the first vector and the origin as the second vector while the last vector only is the “up” vector. The model matrix is a default matrix with the values of 1.

The resolution of the shadow map that is used is the same as the window size therefore the shadows has a good quality but it will suffer a bit on the performance side.

1.5 Acceleration Techniques

1.5.1 View Frustum Culling Against a Quadtree

The implemented frustum culling are working together with a quadtree that sorts the objects after the position in the world. The way this work is that when the objecthandler gets a addobject call, it creates the objects and add it to its own quadtree. Therefore each objecthander got a own quadtree to cull from. When a quadtree gets a object it will add it to its instance if it is in bounding box but if the quadtree already got a object inside it will use its four childs to distribute both the object being added and the object already inside the tree.

When the object is added and the handler containing the object gets a request to render all its object it will pass this request to the quadtree that uses a function “inView” to see if this quadtree bounding box is inside of the frustum.

But to do this the function needs a farplane and nearplane to build two planes that goes parallel with the edge of the view. The planes are created the same way but the only difference is the distance to its centre, where the farplane uses a distance of 500 unit length and the nearplane 0.01 unit length.

1.5.2 Back face culling using Geometry Shader

This is a very small technique that is used in the geometry shader of our geometry pass in deferred rendering.

1.6 Other Techniques

1.6.1 Water Effect

This technique are have been implemented by letting objects more their uv-coordinates automatic after 0.08s has passed. Therefore each render it will check if 0.08s has passed, else it will add it to the "timeSinceChanged". When it is time to move the texture it moves all the uvs by the default uvs. Our water uses the uv-coordinates (1,0.0625), therefore it will move the v-coordinate +0.0625 each 0.08s until it reaches 1 where it is reseted to 0.

1.6.2 Glow Effect

The glow technique receives the color texture of deferred rendering and checks if the objects colors are bright enough to glow. The texture created of the glow technique is later passed through a blur filter and the blurred texture is later merged with the final deferred rendered texture.