|  |
| --- |
|  |
| Project Report |
|  |
| **3D-Programming, DV1541, 2015/2016** |

[Välj datum]

Written by : Martin Clementson and David Wigeluis

Technical Artist

Year 2, 2015/2016

Project Report

3D-Programming, DV1541, 2015/2016

# Comments and Considerations

The following project is a 3D Application that was done using C++ programming language.

We utilized the Direct X 11 API by Microsoft to achieve the final result.

The libraries we have used in this projects is:

* DirectXMath
  + This was used to be able to utilize the mathematical functions needed for our various operations such as matrix multiplication and vector operations.
* Algorithm
  + This was added to compliment DirectXMath with more functions such as MIN and MAX.
* Diput
  + This library enabled us to create our custom input handler.
* Vector
  + This library was used to utilize the standard template “Vector” that we used as our primary container.
* Fstream
  + This library was used to open and read from files. We used it for operations such as parsing our “.OBJ” and “.MD5” files, etc.
* String
  + This library was used to get the “String” type as well as the overloaded operations it provides.
* All the necessary DirectX libraries
  + DirectX has several libraries that were included for it to work properly
* DirectXToolKit
  + An unofficial library that provided the “WICtextureLoader” function that we used when loading textures and normal maps.

In our project we have implemented certain functions that can be turned on and off during runtime.

We have mapped these to hotkeys that the user can press to switch.

The keys and its functions are:

* W: This makes the camera move forwards in the direction it is looking.
* A: Makes the camera strafe to the left.
* S: This makes the camera move backwards.
* D: Makes the camera strafe to the right.
* The camera can be rotated using either the mouse or the arrows on the keyboard.
* Left Shift: Toggles the mouse visibility.
  + The camera cannot be rotated with the mouse when it’s in the visible mode (but the keyboard arrows is still functional).
* TAB: This toggles the mini map and the surrounding edge overlay on and off.
* T: When the user is above the terrain, pressing T toggles walking on top of the terrain.
* G: Toggles the post processing render pass. In this case Gaussian blur filtering is used in the post process.

# Core Techniques

## Skeletal Animation

# Geometry Techniques

## Parsing and rendering of .OBJ files

The concept of rendering a mesh from an “.OBJ” file is to assemble it from its own instructions. Usually achieved by reading the file using libraries such as “fstream.h” and assembling it within the code, then parsing the mesh through the graphical pipeline.

We achieved this by first creating a class called “OBJHandler” which main objective was, as the name suggests.

To handle the reading of the “.OBJ” file.

## Height map

# Texturing and Lighting

## Normal mapping

# Projection Techniques

## Dynamic cubic environment mapping

# Acceleration Techniques

## View frustum culling against a quad tree

## Back face culling using Geometry Shader

# Other Techniques

## Gaussian filter using a Compute shader