Kelvin … 15051161  
Eimantas … ---  
Niels van der Meer 15112411

KB-01 documentation

The Hague University | 3D-engine design

2018

# The engine

For the KB-01 course of The Hague University we got the assignment to design a 3D-engine. By the end of this course the engine must have multiple distinct features, or requirements, that must be working before the deadline. These requirements include:

* A Skybox
* Moving/rotating assets
* A height map
* A bitmap
* Camera movement

The engine was made with Microsoft Visual Studio software. Multiple files with code have been made, all with a different class. The main program can be started from our ExecutableProject folder, with the other files being stored in a GameEngine3D folder.

# Classes

For the engine to run we need to have multiple classes. Every class we have is split between a header file and a .ccp file.

**AssetLoader**This class is connected to the AssetLoader header, as well as the Asset header file. The code in this class makes is so that we are able to get our assets form our Assets folder in the project map. It creates and loads the assets into our engine through our trajectory path and puts them in the virtual space.

**Bitmap**A bitmap is a file with the structure of the pixel grid making an image. We use our Bitmap files to chart the pixels of our bitmap file. We make sure out file had a bit count of 24. If that’s not the case, we log an error for that reason. We can log it due to the connection with our Debug file.

**Camera**Our camera is connected with our Entity header file. With that connection we’re able to define movement for our camera inside our engine using both the rotation of our camera and movement.

**Debug**Inside our Debug files we are able to create our debug logger. Inside our logger we can document things that happen within our engine, such as information about what is happening and warning or error messages when that is needed. An example of this is getting a message when input is failing.

**Entity**Our Entity works from our Transform header file. The transform file can give us the position of an entity in the engine, like the camera. We can get both the X, Y and Z position, but also the current rotation of that entity and how large it is. With those values, the entity file can set the position, scale and rotation of the entity inside the engine.

**FPSCounter**With this class we can get the FPS, Frames Per Seconds, at which the engine is running. Using the Timer header file, we can count the frames that have passed within a certain amount of time using our Update section within the file. This number is eventually returned to us with via GetFPS.

**Input**With our input file we can monitor when certain keys are pressed and we move our mouse button. We use these inputs for our camera so we can move around in the world our 3D-engine creates. When input isn’t working, we can send an error message to our Debug logger so we can see where things go wrong.

**Kernel**The Kernel is one of the most important files in the engine. Form this file we manage our camera control, the opening of our windows and the rendering of everything within out engine. Also from this file we can send error messages to our logger if something isn’t working the way it should or an instance is missing. The loop that keeps on rendering every frame also is put in motion within this file.

**Scene**From our Scene and SceneLoader files we actually create and load in everything within our scenes in the engine. From here we call upon the assets to load in, the skybox and the terrain so it can all form inside the engine. While the Scene file calls forth these files, the SceneLoader sets the scenes into position for use.

**Timer**The Timer file is for keeping track of the time within the engine. It is mostly used for keeping track of the FPS of the engine and the Performance it is giving.

**Window**You can’t see the engine without a window to look through. This is where the Window class comes into play. In this class we define how big the window is we open and are also able to define to be able to shut down the window.

# Main Project Function

At the beginning of the main function we start the code in the Kernel file. In this file we make sure that a window is opened where we can display our engine. Also our Debuglogger is called through our Kernel file.

After the Kernel and Debug logger we get to the code that controls our camera movements. We can freeroam the digital space with input on our WASD keys for actual movement in the faced direction and use input on the arrow keys for the rotation of the camera. This is called via the CameraController.

After that we get to the AssetLoader file. Here we load in the assets and textures to shape our virtual world. The skybox code is within our main file. It gives us a virtual box that lets us have an orientation of our position within the virtual world. The skybox gets a set rotation and set position after which it gets rendered.  
The terrain is also loaded after that. We load in our heightmap with textures to create depth in the terrain of our engine.