

***A Frenetic First-Person Shooter (FFPS)***

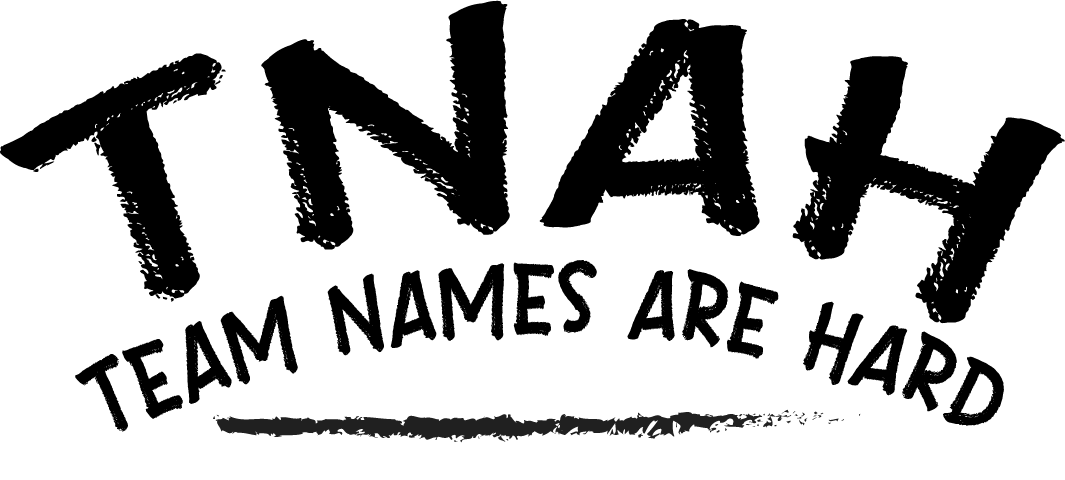
**Group Self-Assessment**

Version 1.0

Produced by Dylan Blereau, Bryce Standley, and Christopher Logan

Created: March 31, 2021

Updated: April 9, 2021



# Group self-assessment which provides an assessment of the group's work as a whole as well as each member's own work. As part of the self assessment, you need to include the tasks breakdown list which shows each task, estimated time taken and actual time taken and the names of people involved with each task. A template file for the task breakdown list can be downloaded [here](https://www.it.murdoch.edu.au/~S900432D/ag896gYi899G31QHaNRca3y_p,xtq82275ql/assignments/TaskBreakdownList.doc). Each group will be submitting this as part of the self assessment. In addition to the task breakdown list, the self assessment should address the following issues:

# Testing Details

The testing has done using both visual aids, such as seeing something and analysis in within the window, and also using debugging within the command line.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Testing Window | | | | | |
| Test Case | Description | Data | Expected result | Actual result | Pass/Fail |
| 1 | Setting up the window and setting it to GLFW works, adding a name, and size from a script | String name  Int height  Int width | The window will correctly output with the correct height and name | The same as the expected result | Pass |
| 2 | The camera will be able to go to the correct position and display objects | Camera set | The camera will correctly do as the description advertises | The camera correctly does as the description advertises | Pass |
| 3 | The input system works and registers, by displaying the exit screen, moving the camera, and using the wireframe | WASD for movement  K for wireframe  X for exit | They will all correctly work when clicked | They correctly work when clicked | Pass |
| 4 | The input script can correctly remap keys | Forward key for W (87) will be swapped for P (80) and still work the same way when P is press | Will move the camera on P click forward | Moves the camera forward on P click | Pass |
| 5 | Resizing the widow keeps the scale of the whole world | Changing window size in windows | The rendering will not be significantly distorted | It worked | Pass |
| Engine Testing | | | | | |
| 6 | Will correctly create the correct number of scenes from scripts and output the amount it created (For testing) | Amount = 2 | There will be 2 scenes created | It worked  Scenes: 2 | Pass |
| 7 | Successful initializes glfw and glad setting the renderer and window |  | Clarification outputs to the command will tell us if they worked or not | It worked  GLFW init = 1  GLAD init = 1 | Pass |
| Scene Testing | | | | | |
| 8 | Loading Game Objects From a file | 10 Game objects are set into the scene1.lua script file these are all tree | 10 trees will spawn on the map and render correctly | It worked | Pass |
| 9 | The terrain will correctly load in and render | The terrain with final game settings will load and render correctly, based of specifications in terrain.lua | The terrain will correctly load in and display | It worked | Pass |
| 10 | The skybox correctly loads in and renders | The skybox with the settings from skybox.lua correctly loads in and displays | The skybox will render and display correctly | It worked | Pass |
| Collision Testing | | | | | |
| 11 | Test static objects by walking into them | Load the scene1.lua script from our game | Load in the standard scene we will have in our game, which will have about 30 or so objects, and walk into each reachable tree and see if the collision works | It worked | Pass |
| 12 | The token collision testing, it disappears on collision | Walk into each token based of scene1.lua | Each token will disappear once walked into, leaving none | Walked through each token and none of them remained | Pass |
| 13 | The enemy collision testing, they disappear on collision | Walk into each enemy based of scene1.lua | Each enemy will disappear once walked into, leaving none | Walked through each enemy and none of them remained | Pass |
| 14 | The terrain collision | Walk around the terrain | The player will have their height adjusted based on the terrain they are currently on | It worked | Pass |

# Suggestions for Improvement

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task** | **Allocated completion time** | **Actual completion time** | **Allocated to (the name of the group member)** | **Comments** |
| Collision Detection | 1 week | On Going | Bryce Standley | Simple system in place, More complex system is work in progress |
| Bounding Boxes | 1 day | On Going | Bryce Standley | Created but still being debugged and developed |
| Exit Screen | 3 hours | 1 day | Bryce Standley / Chris Logan | Created and rendering |
| Renderer | 3 Weeks | 3 Weeks | Chris Logan / Bryce Standley / Dylan Blereau | Rendering for all objects in a scene |
| Game Input | 3 hours | 2 hours | Chris Logan / Bryce Standley | Per engine update and event driven input |
| Scene Management | 1 Week | 4 days | Chris Logan | Management for scene and its objects |
| Window Management | 5 days | 5 days | Chris Logan | Management of the application object |
| Graphics facade | 1 day | 1 day | Chris Logan | Façade of graphics API (OpenGL) |
| Terrain | 1 week | 1 week | Dylan Blereau / Bryce Standley | Generation and texturing of terrain though heightmaps |
| Camera | 2 hours | 2 hours | Dylan Blereau | Camera system controlled though game input |
| Skybox | 2 hours | 2 hours | Dylan Blereau | Skybox within a scene |
| Mesh/Model Import/Rendering | 4 days | 7 days | Dylan Blereau / Chris Logan | Assimp mesh loading and rendering |
| Debugging | 2 hours | 2 hours | Bryce Standley | Debugging tools |
| Game Objects | 1 day | 1 day | Chris Logan | Base object within a world |
| Game Asset Factory | 2 hours | 2 hours | Chris Logan | Creator of game objects and other assets |
| Lua Scripting | 3 hours | 4 hours | Dylan Blereau | Ability to script game elements in lua |
| Texture Loading | 2 hours | 2 hours | Dylan Blereau | Loading terrain and other textures |
| Player Management | 2 days | 2 days | Chris Logan / Bryce Standley | Management of the player and player movement |
| Game Assets | 2 days | 1.5 days | Bryce Standley | Creation/Collection of assets used within the game |
| Documentation | 3 days | 3days | Dylan Blereau / Chris Logan /Bryce Standley | All related documents |
| Testing | 2 days | 2 days | Chris Logan | Testing of particular game and engine systems |
| Shaders | 2 days | 1 day | Bryce Standley / Dylan Blereau / Chris Logan | Different vertex and fragment shaders for each type of object in the game world |
| Engine Design  Planning | 1 week | 1 week | Chris Logan | Planning the overall structure of the engine |

## Time Management

Overall, I think everyone in the team was largely successful at getting the work done in the allotted time. However, improvements can be made in some areas, as some tasks took far longer than anticipated.

An example of this was when Dylan and Chris worked on the ASSIMP model loader. A task that they planned only to take 4 days took a week in part due the first build having too many bugs and needing to be redone. While it is expected that not everything will be smooth sailing, putting more effort into debugging code when each team member has issues would go a long way to saving some much-needed time.

Thankfully our task breakdown list suggests that most tasks were done before or on time, which means that we are already doing a good job. There is always room for improvement however, so we will be making a more concerted effort to debug code better, so that we don’t end up with messy code that needs to be scrapped.

## Communication

While the group are all good friends and communicate almost daily, we feel improvement could be made at providing one another with a bit more support to ensure we all stay on task. Making sure everyone has a role to do ensures that development of the assignment will go smoothly. We feel as a group that communicating with one another should be a bit more focused in the second assignment, to ensure we aren’t crunching and rushing to get the assignment finished.

## Commenting as you code

Every member of the group was somewhat lax when it came to commenting the code as it was being written. We all would leave basic comments that would help us understand what the code was doing, but our failure to implement proper doxygen comments as we went along with development meant that we has to spend a lot of time at the end of the project commenting code. We aim to improve by doing doxygen comments as we code in assignment 2.

# Software Reuse

The code in the program has been designed with reusability in mind, with most classes being capable of working in other programs outside the engine, with very little adjustment to the class themselves. The scene class has been designed so that it is easy to implement more of them when requirement, it also holds many game Objects which again has been designed so that they are easily adaptable. The whole program itself is designed with the intention of users being able to use it for their own game purposes, such as users using the many lua scripts to change the functionality and visuals of the game, specifically at this time the terrain, engine, scene, skybox and inputs of the game. The code for skyboxes, terrain and models could also be reused in other game engines for the same purposes, and the rendering system allows for a good interface to add a new API, such as Vulkan, or in our case easily take the OpenGL one we have an easily put in into another engine.

The assets we use are also reusable in many ways, as they are all a standardized form, such as the models and textures all being fbx, png and jpg files. This means that it is firstly easy to adapt them to new software being made (such as reusing our code) and also easy to adapt and edit them in existing editing software, such as photoshop for the images, and blender for the fbx files. The shaders could also be used in other OpenGL projects, but the scripts are designed with this engine in mind, but they are easily adaptable for the current game engine.