3D Graphics Programming

Simulated Scene

Craig Palmer – V8009358

2019

## Features List

* Generated Terrain Mesh
  + Heights extracted from heightmap
  + Applied Ken Perlin noise to extracted heights
  + Normals calculated
* 3D Models rendered
  + Multiple models placed
  + Model movement via arrow keys
* Skybox
* Lighting
  + Directional
  + Ambient

## Controls

|  |  |
| --- | --- |
|  | Camera  Movement |
|  | Jeep Movement |

A close up of a keyboard

Description automatically generated

# UML Diagram

## 

## Performance Profiling

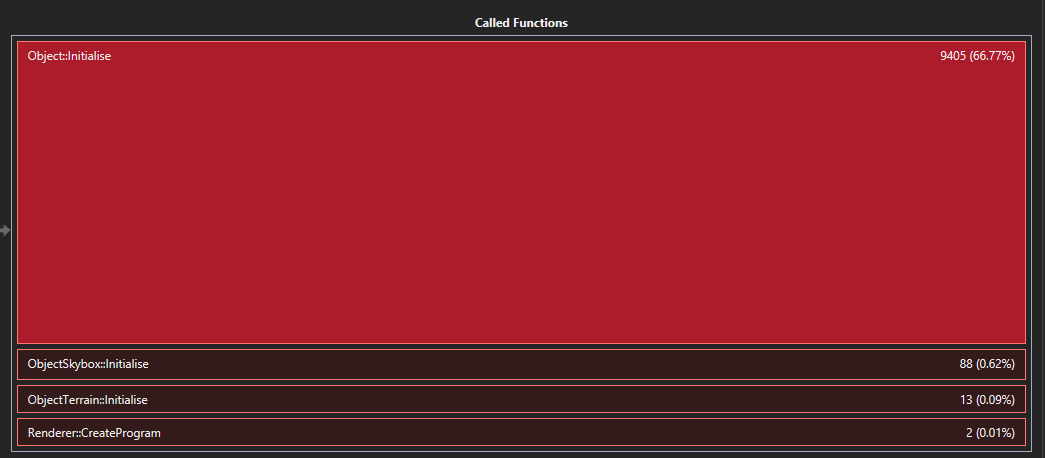
**Main() – 84.10%**

This is to be expected as most of processing is done in loading all the object models and setting up the scene. To improve, I would first look into the Initialise function.



**Renderer::InitialiseGeometry() – 67.50%**

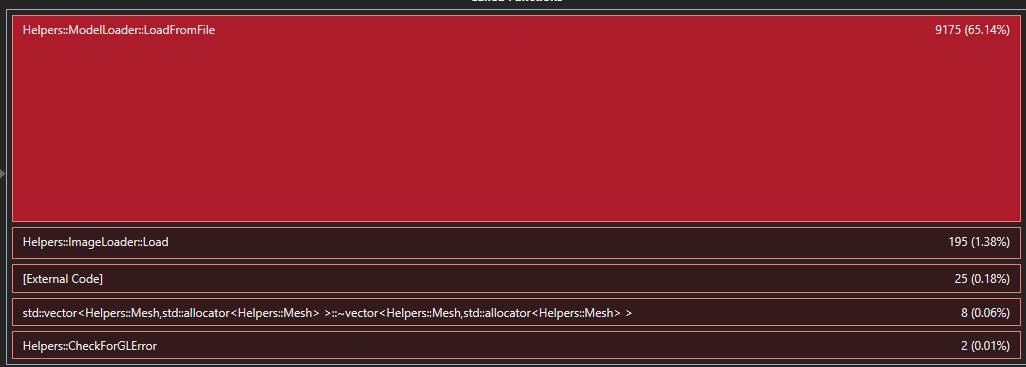
Entering further, most of the processing is taking place in the **Object::Initialise()** calls. Breaking it down further I can see which objects are taking the most time to Initialise.



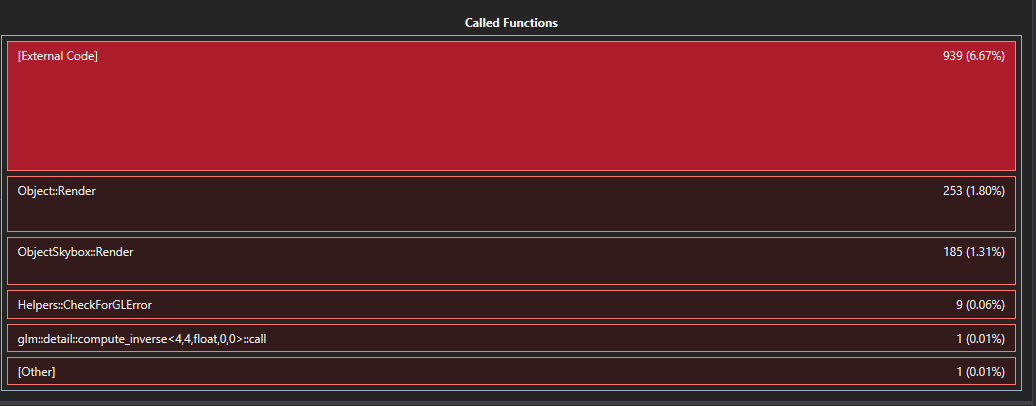
From **Object::Initialise(),** I created a graph to see which objects are taking the most time to execute. In the demo there are 10 trees, 10 rocks and 1 jeep.

Looking at the performance of the function, 97.55% is the **Helpers::ModelLoader::LoadFromFile()** function loading the tree model. I can assume then it is due to the object file having too many vertices. In which case to improve I would suggest either using a smaller object file or I would go back to the artist and say this object file is slowing down the code too much.

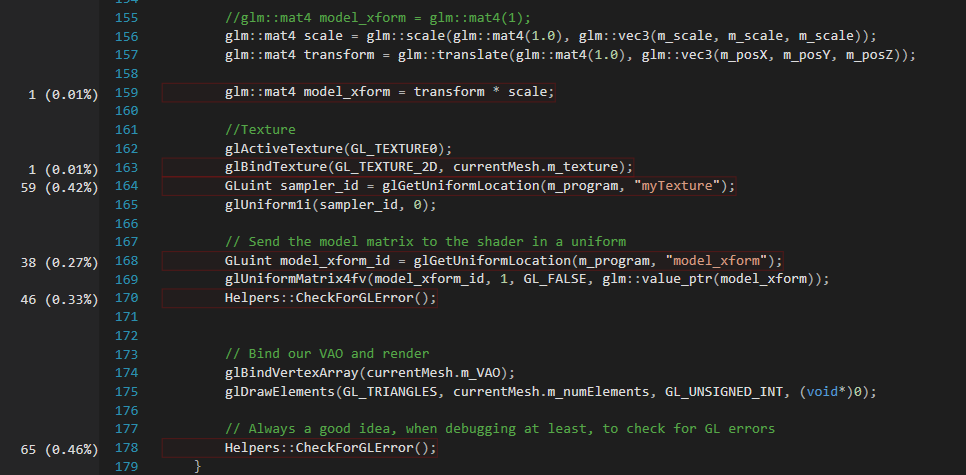
Alternatively, I only really have to load the model once and just have multiple instances of that same model. This could reduce the loading time down to 1/10.



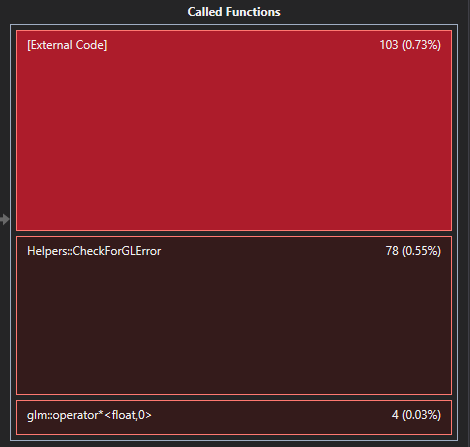
**Renderer::Render() – 9.87%**



Inside of the Object::Render function, no apparent improvements are clear. Multiplying matrices is expensive but required. Small increases could be made by removing the **Helpers::CheckForGLError()** but would only equate to 0.06% so the performance increase would not outweigh the benefits of the function checking for errors.



Similar can be said for ObjectSkybox::Render(). Majority of improvements could be made by removing Helpers::CheckForGLError calls but the increase of 0.55% would not be significant enough.



In summary to improve performance I would locate an alternative model to use for scenery, or I could potentially reuse the model 10 times instead of loading the same tree 10 times. I could reduce the number of Helpers::CheckForGLErrors() but that would be a minimal and not generate many gains. Additional features, I would like to implement a hierarchy system, have children to specific objects transforms and would require building a node system for that. I could also include spotlights from models to help generate a more realistic look.