**3D Graphics Programming**

**ICA Report**

**Rohit Lalwani**

**V8179602**

**Note:** Maximum lights allowed have been limited to 8 for the purposes of the scene, however only 5 lights have been implemented.

Phong shading has been used to produce the lighting of the terrain and 3D model.

**Controls:**

W, S – Camera movement along z-axis

A, D – Camera movement along x-axis

Q, E – Camera movement along y-axis

Up Arrow, Down Arrow – Point Light movement along the z-axis

Left Arrow, Right Arrow – Point Light movement along the x-axis

RShift, RControl – Point Light movement along the y-axis

**Performance Data:**

**CPU Analysis:**

* 445ms of CPU time during CreateGLFWWindow()
* 296ms of CPU time during Simulation::Initialise()
  + 295ms for Scene::Initialise()
    - 277ms for Model::Initialise()
    - 14ms for Terrain::Initialise()
    - 3ms for Scene::CreateProgram()
    - 1ms for Scene::CreateLights()
* 2ms of CPU time during Simulation::Update()
  + 1ms for Renderer::Render()
  + 1ms for Scene::RenderUpdate()

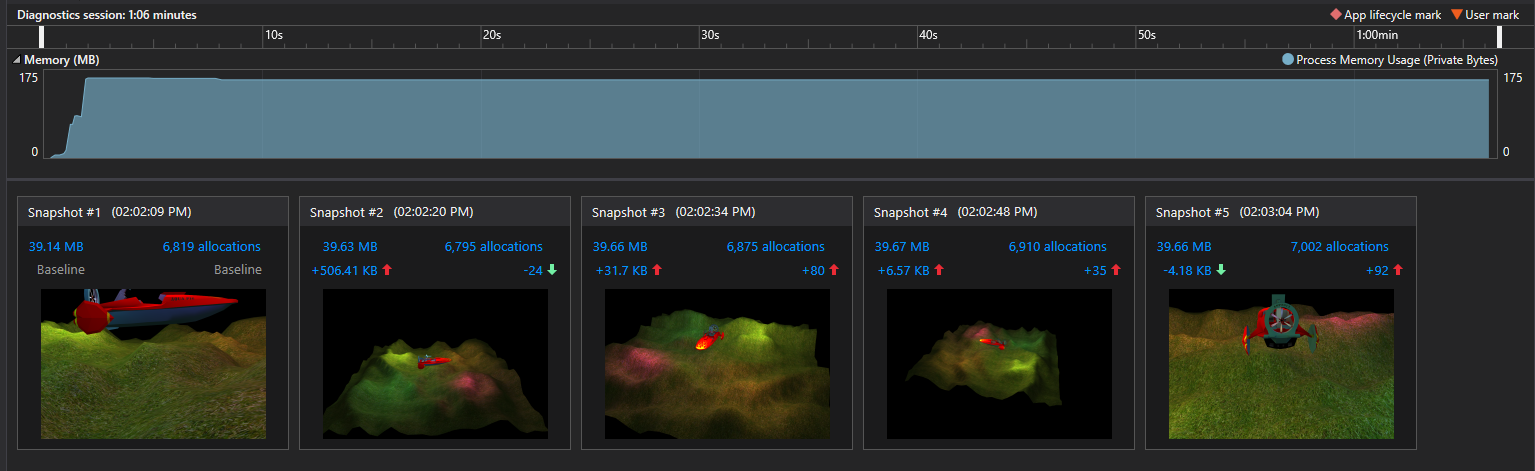
The Initialisation cycle takes up a significant chunk of CPU time, however this is justified as the terrain must be generated and the model is fully loaded at this time.

As seen in the data below, most of the CPU’s time is spend in the Renderer, which would be as expected since that is where the rendering calculations for matrices and such take place. Since the “RenderObjects” method of the Renderer is responsible for the rendering of 3D models on the screen, it is fitting that it takes up the largest chunk of the Render cycle.

A screenshot of a computer

Description automatically generated

**Memory Usage Analysis:**



Baseline Heap size: 39.14MB

Heap size with Terrain partially clipped and all lights visible: 39.63 MB

Heap size with Terrain and lights partially clipped: 39.66 MB

Heap size with nothing clipped: 39.67 MB

Heap size with lights, terrain, and 3D model heavily clipped: 39.66 MB

The consistency in memory usage over time shows that there are no memory leaks present in the program.

The number of triangles drawn to the screen seems to affect the memory used.

There is no memory usage for the animation of the propeller spinning as that is achieved simply by altering the transformation matrix of the propeller.

Since every object in the scene uses the same shader, there is not much memory utilization for them and a more advanced implementation may involve making use of this extra memory to load more varied shaders for a potential visual upgrade.

**Potential Improvements:**

* Visual indicators (such as a small orb) to show where the lighting originates from.
* Skybox to make the scene look less empty
* Make 3D model move around the scene, rather than have a movable light
* Implement a system to import the hierarchy with the model itself, thus creating the hierarchy on load without the need to have a separate function for the same.
* Frustrum culling to speed up rendering
* Dynamically move, add, remove, turn on, or turn off lights in the scene