# Real-Time Rendering and 3D Games Programming

# ASSIGNMENT 1 – REPORT (v1.1)

## INTRODUCTION

Which shape did you choose to draw? Did you derive the algorithm on your own or did you find some other resource to help? List any sources used (books, articles, videos, ...).

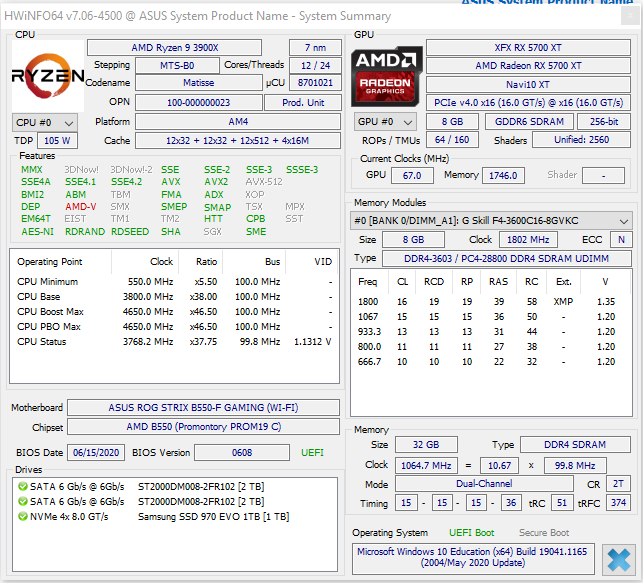
I chose to draw the Menger-Sponge, with the algorithm derived from both (only logic was used):

* <https://antimatroid.wordpress.com/2012/02/01/menger-sponge-in-c-using-opengl/>
* <https://en.wikipedia.org/wiki/Menger_sponge>

And lastly the opengl book provided

* <https://learnopengl.com/book/book_pdf.pdf>

Describe the hardware you used to perform the tests described in this report. Include detailed CPU and GPU information. What screen resolution and refresh rate did you use?



Running at 2560x1440 resolution at 144hz with V-Sync turned off in the GPU settings.

Describe your data structure and algorithm. Are you duplicating vertices that are used by multiple triangles or did you implement shared vertices? Are these cases where multiple faces might overlap? Which OpenGL drawing primitive are you using?

I have a cube object which has a variety of properties, the more important ones being a vector of type vex3 for vertex data and a vector of type int for index data. I generate a straight stream of vertices and corresponding indexes using a recursive method and I push the vertex and index data to each vector. There shouldn’t be any cases where faces overlap as each drawing method draws a GL\_TRIANGLES that is only drawn once per draw loop. So each face is only drawn a singular time.

How did you choose to colour the shape? How many materials did you use and how were they assignment to faces? How did you 'communicate' face material data to the Shaders?

I created a Material object that I attached to the cube for 3 materials which I named; ruby, turquoise and emerald. These were chosen at random and sourced from [here](http://www.it.hiof.no/~borres/j3d/explain/light/p-materials.html). They were assigned to faces via normal values that were generated with the vertex and index data. With a different material applied to the x, y and z axis (3 axis = 3 materials). The materials were passed as uniforms to the shaders in the same way with three uniforms for 3 Material objects, where each shader has a Material struct and the ambient, diffuse, specular and shininess are passed directly onto the object with Unifrom4fv.

How have you decided to position each light source? How did you assign light colours to show off the full capabilities of your lighting model?

There are 8 light sources where light0 or the first source would be the directional light attached to the camera and the rest would be position on each face with lights 1-6 hovering just above each face and then lights 7-8 would be above the model on either side of the x-axis. Each light has a different diffuse value in order to shine a unique colour.

## SCENE 1

Start your testing at subdivision level 1 (base), Lighting On (1 light), Backface Culling On and Depth Testing On.

Create a table showing the average frame rate, number of vertices and number of faces at each level of subdivision that your hardware can handle with a frame rate greater than 1 frame per second.

|  |  |  |  |
| --- | --- | --- | --- |
| Division | Frame Rate | Vertices | Faces |
| 1 | 2272 | 160 | 120 |
| 2 | 448 | 3200 | 2400 |
| 3 | 26 | 64000 | 48000 |

Draw a chart showing the average frame rate achieved at each level of subdivision.

Run some tests with Lighting Off while keeping everything else as above. Describe the impact this has on frame rate and why? Use a table and a chart to show the data.

|  |  |  |  |
| --- | --- | --- | --- |
| Division | Frame Rate | Vertices | Faces |
| 1 | 2439 | 160 | 120 |
| 2 | 446 | 3200 | 2400 |
| 3 | 26 | 64000 | 48000 |

This has an impact that the cube seems to render slightly faster, but it doesn’t make too much off a difference. This would be because performance is not linear and frame time is not proportional to the amount of things that are rendered.

Run some tests with Backface Culling On and Off, while keeping everything else as above. Describe the impact this feature has on frame rate and why? Use a table and a chart to show the data.

|  |  |  |  |
| --- | --- | --- | --- |
| Division (Backface On) | Frame Rate | Vertices | Faces |
| 1 | 2564 | 160 | 120 |
| 2 | 452 | 3200 | 2400 |
| 3 | 26 | 64000 | 48000 |

|  |  |  |  |
| --- | --- | --- | --- |
| Division (Backface Off) | Frame Rate | Vertices | Faces |
| 1 | 2380 | 160 | 120 |
| 2 | 446 | 3200 | 2400 |
| 3 | 26 | 64000 | 48000 |

Upon viewing the frame rates with back face culling on/off it doesn’t appear to be much of a difference but it could most likely be the hardware being fast enough where small things like that aren’t noticed until a specific level of triangles are drawn.

Run some tests with Depth Testing On and Off, while keeping everything else as above. Describe the impact this feature has on frame rate and why? Use a table and a chart to show the data.

|  |  |  |  |
| --- | --- | --- | --- |
| Division (Depth Test On) | Frame Rate | Vertices | Faces |
| 1 | 2540 | 160 | 120 |
| 2 | 452 | 3200 | 2400 |
| 3 | 25 | 64000 | 48000 |

|  |  |  |  |
| --- | --- | --- | --- |
| Division (Depth Test Off) | Frame Rate | Vertices | Faces |
| 1 | 2631 | 160 | 120 |
| 2 | 444 | 3200 | 2400 |
| 3 | 2 | 64000 | 48000 |

In this case it appears that the program seems to run a little faster and better with depth testing on. This could be due to have to discard less fragments and saving some rendering power but the difference is minimal.

Run some tests with Backface Culling On and Off in combination with Lighting On and Off, while keeping everything else as above. When Lighting is On is there a difference in Frame Rate when Backface Culling is On vs Off? Describe Why or Why Not and show data to support your answer. Did you expect there to be a difference? Why?

|  |  |  |  |
| --- | --- | --- | --- |
| Division (L=On/Off BC=On/Off) | Frame Rate | Vertices | Faces |
| 1 L=On BC=On | 2439 | 160 | 120 |
| 2 L=Off BC=Off | 456 | 3200 | 2400 |
| 3 L=On BC=Off | 25 | 64000 | 48000 |
| 1 L=Off BC=On | 2631 | 160 | 120 |
| 2 L=Off BC=Off | 452 | 3200 | 2400 |
| 3 L=Off BC=On | 26 | 64000 | 48000 |
| 1 L=Off BC=Off | 2631 | 160 | 120 |
| 2 L=Off BC=On | 450 | 3200 | 2400 |
| 3 L=On BC=Off | 33 | 64000 | 48000 |

Based on the 3rd tier divisions it appears that light has less of an affect on the performance than backface culling. However the difference also appears to be minimal enough that it might just be these low level calculations not making a significant impact. This was expected based on prior tests.

Run some tests with Depth Testing On and Off in combination with Backface Culling On and Off, while keeping everything else as above. When Depth Testing is On is there a difference in Frame Rate when Backface Culling is On vs Off? Describe Why or Why Not and show data to support your answer. Did you expect there to be a difference? Why?

|  |  |  |  |
| --- | --- | --- | --- |
| Division (DT=On/Off BC=On/Off) | Frame Rate | Vertices | Faces |
| 1 DT=On BC=On | 2439 | 160 | 120 |
| 2 DT=Off BC=Off | 452 | 3200 | 2400 |
| 3 DT=On BC=Off | 25 | 64000 | 48000 |
| 1 DT=Off BC=On | 2501 | 160 | 120 |
| 2 DT=Off BC=Off | 450 | 3200 | 2400 |
| 3 DT=Off BC=On | 36 | 64000 | 48000 |
| 1 DT=Off BC=Off | 2631 | 160 | 120 |
| 2 DT=Off BC=On | 448 | 3200 | 2400 |
| 3 DT=On BC=Off | 25 | 64000 | 48000 |

Based on the tests ran it would appear that with both depth testing and backface culling off the machine ran slightly better however again the difference is too minimal in all divisions to notice.

Discuss the performance characteristics of adding lights to the scene. Include a chart showing impact on frame rate for number of lights from 0 to 9. Discuss the shape of the curve and what it means.

Is there anything you found interesting or unexpected while running the above tests? Explain why.

No based on what the prior tests have shown there is nothing out of the ordinary and the number of lights doesn’t seem to affect the scene performance much.

## SCENE 2

Start your testing at subdivision level 1 (base), Lighting On (1 light), Backface Culling On and Depth Testing On.

Describe how you have decided to handle normal vectors. Are you specifying them per-vertex or per-face? Are you calculating them on the CPU or GPU? If CPU, how do you communicate them to the GPU? Are you storing them in a data structure or are you calculating them when needed in the shader?

The normals are calculated per face using a normalized cross product of each triangle. However normal aren’t stored and just instead set at generation. Depending on the Scene the normal is calculated either on the CPU or directly in the geometry shader. Then passed around via in and outs.

Vary the subdivision level and move around the scene. Describe the performance characteristics you're seeing at the different levels of subdivision? Is the scene getting smoothly animated as you move around? Does it seem to speed up and slow down depending on what's currently being rendered? Why? At what level of subdivision do you start to notice that your machine is struggling with the drawing load? What are some things that **might** be causing it to 'struggle'?

The scene moves smoothly when moving around despite which division is currently loaded. There is no slowdown on camera movement around the scene. The only time slowdown occurs is during the loading of divisions 4 and 5. However after the initial loading the movement is smooth and fine. The slowdown on load could be due to how many vertices are being passed to the shader. Causing the program to hang until all vertices are passed.

Create a table showing the average frame rate, number of vertices and number of faces at each level of subdivision that your hardware can handle with a frame rate greater than 1 frame per second.

|  |  |  |  |
| --- | --- | --- | --- |
| Division | Frame Rate | Vertices | Faces |
| 1 | 3029 | 160 | 120 |
| 2 | 2785 | 3200 | 2400 |
| 3 | 2777 | 64000 | 48000 |
| 4 | 1041 | 1280000 | 960000 |
| 5 | 62 | 25600000 | 19200000 |

Draw a chart showing the average frame rate achieved at each level of subdivision. Compare this to the results you had for Scene 1. What is the data telling you about Immediate Mode vs Modern Mode? What sort of speed-up are you seeing?

As shown above there is already a massive increase in power in modern mode. There is a speed up on every division and every on division 5 it still runs fine for the most part where immediate mode couldn’t even reach division 5. This shows that using Modern mode has obvious improvements and is a power horse.

Run some tests with Lighting Off while keeping everything else as above. Are the performance characteristics similar as for Scene 1? Why or Why Not? Use a table and a chart to show the comparison.

|  |  |  |  |
| --- | --- | --- | --- |
| Division | Frame Rate | Vertices | Faces |
| 1 | 2325 | 160 | 120 |
| 2 | 2564 | 3200 | 2400 |
| 3 | 2272 | 64000 | 48000 |
| 4 | 1041 | 1280000 | 960000 |
| 5 | 63 | 25600000 | 19200000 |

As see above even with lighting off there appears to be not that much of a difference even with immediate mode as there is an equal difference in scene 1.

Run some tests with Backface Culling On and Off, while keeping everything else as above. Are the performance characteristics similar as for Scene 1? Why or Why Not? Use a table and a chart to show the comparison.

|  |  |  |  |
| --- | --- | --- | --- |
| Division (Backface On) | Frame Rate | Vertices | Faces |
| 1 | 2564 | 160 | 120 |
| 2 | 2172 | 3200 | 2400 |
| 3 | 1204 | 64000 | 48000 |
| 4 | 378 | 1280000 | 960000 |
| 5 | 47 | 25600000 | 19200000 |

|  |  |  |  |
| --- | --- | --- | --- |
| Division (Backface Off) | Frame Rate | Vertices | Faces |
| 1 | 2380 | 160 | 120 |
| 2 | 2083 | 3200 | 2400 |
| 3 | 1136 | 64000 | 48000 |
| 4 | 350 | 1280000 | 960000 |
| 5 | 48 | 25600000 | 19200000 |

As shown above just like in scene one there doesn’t seem to be much of a difference between the two where backface culling on runs just slightly better but both are powerful. This could be again that these smaller computations don’t seem to impact the larger picture (at this scale).

Run some tests with Depth Testing On and Off, while keeping everything else as above. Are the performance characteristics similar as for Scene 1? Why or Why Not? Use a table and a chart to show the comparison.

|  |  |  |  |
| --- | --- | --- | --- |
| Division (Depth Testing On) | Frame Rate | Vertices | Faces |
| 1 | 3125 | 160 | 120 |
| 2 | 3124 | 3200 | 2400 |
| 3 | 3025 | 64000 | 48000 |
| 4 | 1041 | 1280000 | 960000 |
| 5 | 63 | 25600000 | 19200000 |

|  |  |  |  |
| --- | --- | --- | --- |
| Division (Depth Testing Off) | Frame Rate | Vertices | Faces |
| 1 | 2972 | 160 | 120 |
| 2 | 2857 | 3200 | 2400 |
| 3 | 2706 | 64000 | 48000 |
| 4 | 1030 | 1280000 | 960000 |
| 5 | 63 | 25600000 | 19200000 |

This graph shows that depth testing on wins out in performance but again the difference between the two is minimal at this scale of drawing. This is the same again as scene one however albeit more powerful that scene one.

Run some tests with Backface Culling On and Off in combination with Lighting On and Off, while keeping everything else as above. When Lighting is On is there a difference in Frame Rate when Backface Culling is On vs Off? Describe Why or Why Not and show data to support your answer. Did you expect there to be a difference? Why?

|  |  |  |  |
| --- | --- | --- | --- |
| Division (BC=On/Off L=On/Off) | Frame Rate | Vertices | Faces |
| 3 BC=On L=On | 2856 | 64000 | 48000 |
| 4 BC=Off L=Off | 1020 | 1280000 | 960000 |
| 5 BC=On L=Off | 63 | 25600000 | 19200000 |
| 3 BC=Off L=On | 2736 | 64000 | 48000 |
| 4 BC=On L=On | 1041 | 1280000 | 960000 |
| 5 BC=Off L=On | 62 | 25600000 | 19200000 |
| 3 BC=On L=Off | 3124 | 64000 | 48000 |
| 5 BC=Off L=Off | 62 | 25600000 | 19200000 |

Discuss the performance characteristics of adding lights to the scene. Include a chart showing impact on frame rate for number of lights from 0 to 9. Discuss the shape of the curve and what it means. Is there any difference between these results and Scene 1 results?

Is there anything you found interesting or unexpected while running the above tests? Explain why.

So it appears that adding and removing lights has a significant change on the performance especially around the 3rd division. This wasn’t actually what was expected and was interesting that lighting has this much of an effect on the performance of the program when pushed through the gpu.

## SCENE 3

Create a table and chart showing the frame rate for each level of subdivision your machine can handle with a frame rate greater than 1 frame per second.

|  |  |  |  |
| --- | --- | --- | --- |
| Division | Frame Rate | Vertices | Faces |
| 1 | 764 | 1440 | 1080 |
| 2 | 740 | 28800 | 21600 |
| 3 | 680 | 576000 | 432000 |
| 4 | 689 | 11520000 | 8640000 |

Is this what you expected? Why or Why Not?

This curve was expected however what was not expected was how well the machine would hold up rendering 9 cubes so well. Where one cube takes about 2800 frames it was expected that 9 cubes would render at roughly 310 frames. (2800/9). What was also not expected was how well the machine would perform at 4 divisions.

Use a table and a chart to show the difference in performance between using GL\_STATIC\_DRAW and GL\_DYNAMIC\_DRAW in your calls to glBufferData(). Run the tests manually by changing the code and recompiling your project.

|  |  |  |  |
| --- | --- | --- | --- |
| Division (Static) | Frame Rate | Vertices | Faces |
| 1 | 764 | 1440 | 1080 |
| 2 | 740 | 28800 | 21600 |
| 3 | 680 | 576000 | 432000 |
| 4 | 689 | 11520000 | 8640000 |

|  |  |  |  |
| --- | --- | --- | --- |
| Division (Dynamic) | Frame Rate | Vertices | Faces |
| 1 | 763 | 160 | 1080 |
| 2 | 781 | 3200 | 21600 |
| 3 | 719 | 64000 | 432000 |
| 4 | 65 | 1280000 | 8640000 |

Discuss the results and whether it is what you expected and, if the two differ, why you think they differ.

The results were not expected as I didn’t expect there to be much of a change but it appears that dynamic draw handles smaller divisions better than static where static can handle larger divisions better that dynamic. This may be due to at 2 or less divisions the way both calls act doesn’t really have any impact on performance due to the relatively small amount of verities. As that increases dynamic tends to struggle were static can cope.

## SCENE 4

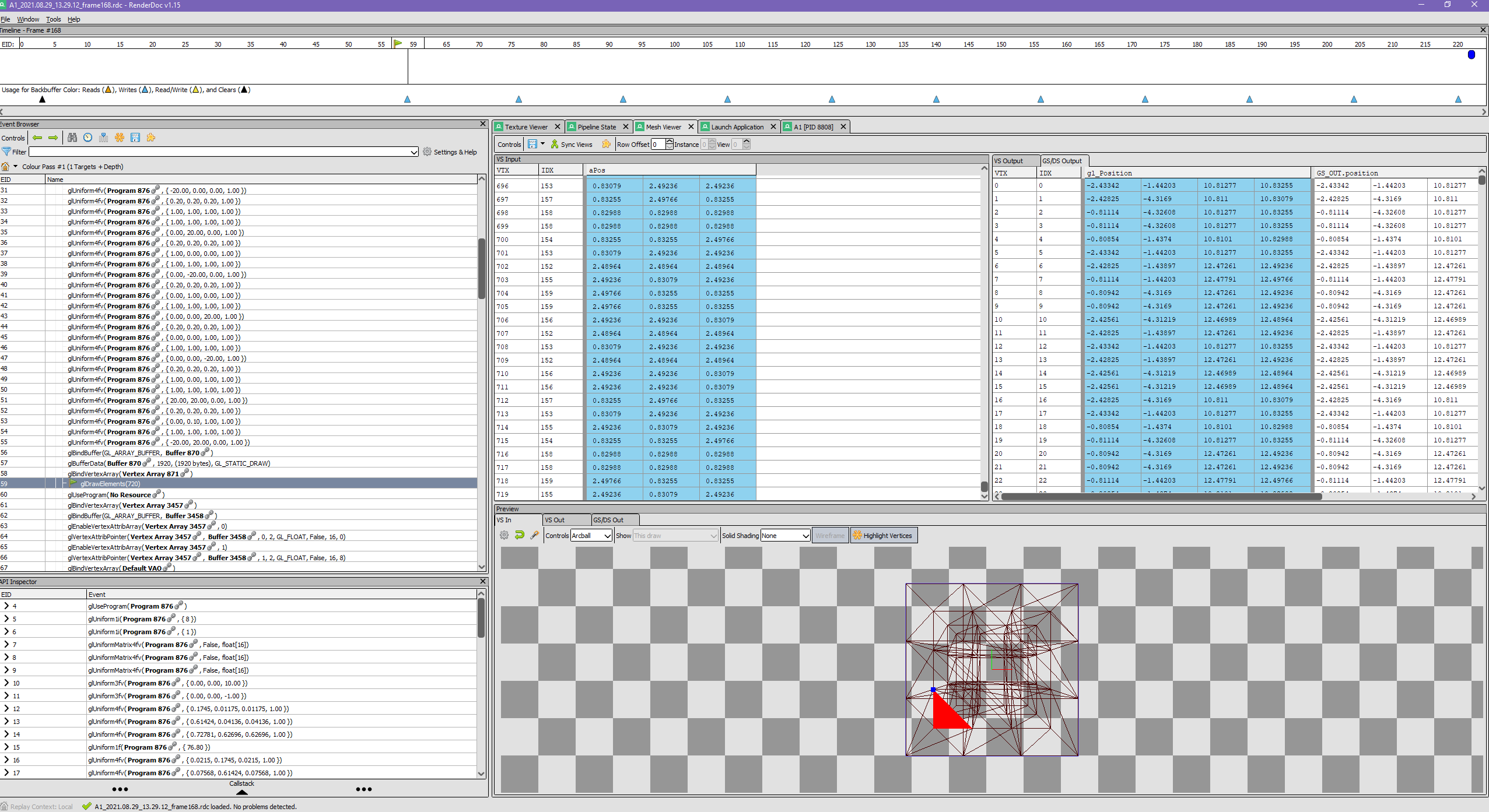
Is there any difference in performance compared to Scene 3? Is this what you expected? Why or Why Not?

|  |  |  |  |
| --- | --- | --- | --- |
| Division | Frame Rate | Vertices | Faces |
| 1 | 2322 | 1440 | 1080 |
| 2 | 1892 | 28800 | 21600 |
| 3 | 649 | 576000 | 432000 |
| 4 | 77 | 11520000 | 8640000 |

There doesn’t appear to be that much of a difference which was unexpected because I assumed that doing the new transformations would have a bigger impact on the performance.

## SCENE 5

There are two sets of position coordinates in your C++ vertex array for this Scene, with three floats each, representing "home position" and "morphed position" for each vertex. You have changed the Vertex Array Object to use the morphed position as the position attribute that is used by the vertex shader. Use RenderDoc to find this data and confirm whether, on the GPU, only the morphed position is being sent across (3 floats) or both the morphed position and the home position (6 floats). Include a screenshot from RenderDoc showing this.



Scene 5 only has the morphed position being sent across to the GPU.

Is this what you expected? Why or Why Not?

Yes as the calculation should’ve been performed on the CPU and then the new vertices should’ve been sent across. As there is no need to send both sets of vertices across.

## SCENE 6

Show a table and a chart comparing the performance (frames per second) of Scene 5 and Scene 6 at different model subdivisions.

|  |  |  |  |
| --- | --- | --- | --- |
| Division (Scene 5) | Frame Rate | Vertices | Faces |
| 1 | 2083 | 160 | 120 |
| 2 | 769 | 3200 | 2400 |
| 3 | 55 | 64000 | 48000 |

|  |  |  |  |
| --- | --- | --- | --- |
| Division (Scene 6) | Frame Rate | Vertices | Faces |
| 1 | 2322 | 160 | 120 |
| 2 | 2564 | 3200 | 2400 |
| 3 | 2272 | 64000 | 48000 |
| 4 | 476 | 1280000 | 960000 |

Discuss what the data is showing.

The difference in performance when a calculation to transform the cube into a spherical object is done on the CPU vs the GPU. Difference in divisions due to exceeding memory capacity.