# 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 a Menger Sponge to draw. My algorithm was mainly based on the following video: <https://www.youtube.com/watch?v=LG8ZK-rRkXo>.

Although it’s not the exact same, as I had to make some tweaks to fit within the assignment specification.

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?

CPU

Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz

Base speed: 2.21 GHz

Sockets: 1

Cores: 6

Logical processors: 12

L1 cache: 384 KB

L2 cache: 1.5 MB

L3 cache: 9.0 MB

GPU

NVIDIA GeForce GTX 1060

Dedicated GPU memory 6.0 GB

Shared GPU memory 7.9 GB

GPU Memory 13.9 GB

Screen resolution: 1920 x 1080

Refresh Rate: 60hz

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?

GL\_TRIANGLES are being used to render the sponge. A mix of shared vertices and duplicating vertices has been used. For each face/square in the sponge, a vertex has been specified, including those vertices shared by two separate faces (so four vertices per face have been specified). Then the order to draw these vertices was stored in separate storage. Although for each face there are six vertices to visit, only four are specified. However shared vertices between faces are specified multiple times and therefore specified multiple times. Therefore, vertices are duplicated but could have been duplicated further without using an elements storage.

Multiple faces overlap as when the sponge vertices are generated, they are generated cube by cube meaning that cubes with faces touching will have their faces overlap.

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 used blue to colour the faces facing in the z axis direction, red for the x, and green for the y. A total of three materials were made. Each frame a uniform in the shaders for each axis (named xMaterial, yMaterial, and zMaterial) were set to the appropriate colour.

For immediate mode the information for normals was used to generate a colour. For example, a normal of (1.0, 0.0, 0.0) will generate a red colour.

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?

Light positions are determined randomly my using srand() and rand. This is also the case for light colours as the RGB values are each assigned a random value between 0 and 1. A maximum of 10 lights can be placed as there needed to be a set number of lights that can be stored as a uniform in the fragment shader.

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

All tests will be completed with vsync off.

|  |  |  |  |
| --- | --- | --- | --- |
| Subdivision | Average Frame Rate | Number of Vertices | Number of Faces |
| 1 | 2176 | 1440 | 280 |
| 2 | 261 | 28,800 | 5600 |
| 3 | 17 | 576,000 | 112,000 |

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.

|  |  |  |  |
| --- | --- | --- | --- |
| Subdivision | Average Frame Rate | Number of Vertices | Number of Faces |
| 1 | 2172 | 1440 | 280 |
| 2 | 273 | 28,800 | 5600 |
| 3 | 15 | 576,000 | 112,000 |

As we can see from the table and chart, removing a single light has almost no impact on average frame rate. This could be due to a single directional light having little impact on the lighting calculations.

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.

Backface culling is already on for previous tests, so we will create a table with Backface Culling off.

|  |  |  |  |
| --- | --- | --- | --- |
| Subdivision | Average Frame Rate | Number of Vertices | Number of Faces |
| 1 | 2173 | 1440 | 280 |
| 2 | 257 | 28,800 | 5600 |
| 3 | 14 | 576,000 | 112,000 |

Once again disabling backface culling has little to no impact on the frame rate.

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.

Depth Testing Off:

|  |  |  |  |
| --- | --- | --- | --- |
| Subdivision | Average Frame Rate | Number of Vertices | Number of Faces |
| 1 | 2083 | 1440 | 280 |
| 2 | 245 | 28,800 | 5600 |
| 3 | 13 | 576,000 | 112,000 |

Deactivating Depth Testing finally appears to have a noticeable effect on frame rate. This would most likely be due to the depth buffer no longer discarding information relating to what shapes should be drawn in front of the other, resulting in a large amount of information taking up the buffer.

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?

Previous test results will be copied into this section for clarity.

Backface Culling: On

Lighting: On

|  |  |  |  |
| --- | --- | --- | --- |
| Subdivision | Average Frame Rate | Number of Vertices | Number of Faces |
| 1 | 2172 | 1440 | 280 |
| 2 | 273 | 28,800 | 5600 |
| 3 | 15 | 576,000 | 112,000 |

Backface Culling: Off

Lighting: On

|  |  |  |  |
| --- | --- | --- | --- |
| Subdivision | Average Frame Rate | Number of Vertices | Number of Faces |
| 1 | 2173 | 1440 | 280 |
| 2 | 257 | 28,800 | 5600 |
| 3 | 14 | 576,000 | 112,000 |

Backface Culling: On

Lighting: Off

|  |  |  |  |
| --- | --- | --- | --- |
| Subdivision | Average Frame Rate | Number of Vertices | Number of Faces |
| 1 | 2172 | 1440 | 280 |
| 2 | 273 | 28,800 | 5600 |
| 3 | 15 | 576,000 | 112,000 |

Backface Culling: Off

Lighting: Off

|  |  |  |  |
| --- | --- | --- | --- |
| Subdivision | Average Frame Rate | Number of Vertices | Number of Faces |
| 1 | 2210 | 1440 | 280 |
| 2 | 266 | 28,800 | 5600 |
| 3 | 23 | 576,000 | 112,000 |

Based on the previous tests, I did not expect to see much change in framerate with this test.

A small change in frame rate can be seen when lighting is on, but backface culling is on and off with level subdivision of 2. There is a 16 frame difference between these two levels. I was not expecting the frame rate to drop with backface culling turned off. This could be explained due to the way that the sponge is drawn. As the sponge is drawn cube by cube with faces touching, there are still faces inside the sponge being rendered, meaning that there aren’t many faces being culled.

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?

Depth Testing: On

Backface Culling: On

|  |  |  |  |
| --- | --- | --- | --- |
| Subdivision | Average Frame Rate | Number of Vertices | Number of Faces |
| 1 | 2176 | 1440 | 280 |
| 2 | 261 | 28,800 | 5600 |
| 3 | 17 | 576,000 | 112,000 |

Depth Testing: On

Backface Culling: Off

|  |  |  |  |
| --- | --- | --- | --- |
| Subdivision | Average Frame Rate | Number of Vertices | Number of Faces |
| 1 | 2172 | 1440 | 280 |
| 2 | 274 | 28,800 | 5600 |
| 3 | 14 | 576,000 | 112,000 |

Depth Testing: Off

Backface Culling: On

|  |  |  |  |
| --- | --- | --- | --- |
| Subdivision | Average Frame Rate | Number of Vertices | Number of Faces |
| 1 | 2083 | 1440 | 280 |
| 2 | 245 | 28,800 | 5600 |
| 3 | 13 | 576,000 | 112,000 |

Depth Testing: Off

Backface Culling: Off

|  |  |  |  |
| --- | --- | --- | --- |
| Subdivision | Average Frame Rate | Number of Vertices | Number of Faces |
| 1 | 2073 | 1440 | 280 |
| 2 | 259 | 28,800 | 5600 |
| 3 | 14 | 576,000 | 112,000 |

There appears to be more of a difference in framerates for these tests, which I was not expecting based on the previous tests.

Of course with depth testing disabled, the sponge takes a strange appearance. The tests with lower framerates appear to be the ones with depth testing turned off. The reason for this would most likely be the same as stated in a previous test. Depth testing is most likely the main reason for the variance in framerates, as backface culling in previous tests had little impact on the average framerate due to the way that the sponge is drawn.

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.

As with the other tests, I didn’t expect to see much variety in the results. This would be the same case with these tests. With little to no variance except from the difference between adding 1 light and 0 lights on the level 1 sponge (which would most likely have been caused by an outlier factor), there is little to say about the performance about adding more lights. This could be due to how the fixed-function pipeline calculates lighting, but I cannot identify a specific reason why the curve is flat.

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

I expected the tests to show a great deal of variance in framerate, and so I was not expecting the difference in frame rates to be the size they were. Perhaps my expectations were placed too high, but these difference in framerates were disappointing.

## 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?

I am currently specifying the normal for each vertex on the CPU. They are stored in the same array as the vertices and sent in the same VBO with glVertexAttribPointer being used to identify the normals for the vertex shader. With the sponge faces consisting of squares that face axis directions, I thought it would be easier to place them straight into the same array as the vertices, without having to waste calculating them each frame.

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 speed of “flying around” would vary because the time it takes between rendering frames is used in the calculation in determining the speed of movement. With a higher frame rate there would be less time in between frame generation, and that would make the movement slower. If this speed was not scaled, the camera would move extremely fast. My computer begins to struggle at a subdivision level of 4. Of course with the amount of faces being rendered this would begin to slow down rendering.

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.

|  |  |  |  |
| --- | --- | --- | --- |
| Subdivision | Average Frame Rate | Number of Vertices | Number of Faces |
| 1 | 2702 | 1440 | 280 |
| 2 | 1587 | 28,800 | 5600 |
| 3 | 221 | 576,000 | 112,000 |
| 4 | 9 | 11,520,000 | 2,240,000 |

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?

There is a significant difference in framerate especially in Scene 2, with Modern Mode (MM) showing an increase in performance by more than 6 times and then an increase in performance by a factor of 13 in Scene 3. Scene 4 is able to hold an average frame rate of 9 for MM while Immediate Mode (IM) was not able to keep a framerate of more than 1.

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.

|  |  |
| --- | --- |
| Subdivision | Frame rate |
| 1 | 2953 |
| 2 | 1784 |
| 3 | 228 |
| 4 | 10 |

Obviously comparing MM to IM in terms of frame rate for each subdivision, MM has significantly higher frame rates than IM. However the decrease in framerate appears similar, as it drops significantly with each subdivision as the framerate flattens out in the chart above towards the end.

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.

|  |  |  |
| --- | --- | --- |
| Subdivision | Frame rate with backface culling ON | Frame rate with backface culling OFF |
| 1 | 2957 | 2954 |
| 2 | 1814 | 1757 |
| 3 | 227 | 227 |
| 4 | 7 | 10 |

With different framerates, the similarity between the scenes is still the drop between higher subdivisions.

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.

Lighting is off as per the previous question

|  |  |  |
| --- | --- | --- |
| Subdivision | Frame rate with depth testing ON | Frame rate with depth testing OFF |
| 1 | 2953 | 2924 |
| 2 | 1784 | 1754 |
| 3 | 228 | 227 |
| 4 | 10 | 6 |

As with the last question, the framerates are not the same of course but the rate in which framerate decreases is similar to scene 1. Once again there doesn’t seem to be much difference between having depth testing on and off most likely due to the sponge drawing algorithm.

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?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Subdivision | Lighting: ON  Backface: OFF | Lighting: On  Backface: On | Lighting: Off  Backface: On | Lighting: Off  Backface: Off |
| 1 | 2957 | 2970 | 2941 | 2971 |
| 2 | 1781 | 1739 | 1781 | 1784 |
| 3 | 228 | 221 | 227 | 229 |
| 4 | 9 | 9 | 6 | 10 |

Framerates for subdivision 1 vary between 2700 and 3000 so it’s hard to determine an average framerate, so subdivision 2 and 3 are usually more useful. Having both lighting and backface culling meant a lower framerate which should be surprising, but previous results usually show backface culling having little effect. Having lighting and backface culling off resulted in the highest framerate, due to the computer having to do no lighting calculations. However the difference is still very little.

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?

The rate of decrease is similar to what we’ve seen in previous tests, with framerate decreasing sharply in lower subdivisions as there are less cubes to draw but then performance decreasing at a lower rate with high subdivisions. The number of cubes in any subdivision can be calculated with 20n, where n is the level of subdivision. So It makes sense that the shape of the curve would be similar to an exponential as the number of cubes per level is also an exponential.

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

I found the difference in framerates between scene 1 and scene 2 to be very interesting. Even though scene 2 uses modern techniques (with modern almost always meaning better), its still interesting to compare performance between older and more modern rendering techniques.

## 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.

|  |  |
| --- | --- |
| Subdivision | Framerate |
| 1 | 2043 |
| 2 | 392 |
| 3 | 25 |

Is this what you expected? Why or Why Not?

I was expecting much lower performance giving that the computer is rendering 9 sponges compared to scene 2 which only renders 1. The framerate is definitely surprising but the sharp decrease with each subdivision is not.

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.

|  |  |  |
| --- | --- | --- |
| Subdivision | STATIC\_DRAW Framerate | DYNAMIC\_DRAW Framerate |
| 1 | 2043 | 2041 |
| 2 | 392 | 394 |
| 3 | 25 | 28 |

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

There is little difference here between using STATIC\_DRAW and DYNAMIC\_DRAW. I believe changing the two only gives the GPU hints at how the data storage will be modified, so there is no guarantee that changing these two will significantly change performance.

## SCENE 4

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

The framerate is significantly higher in Scene 4. Scene 4 is able to get to subdivision 4 with a framerate of 8, and lower subdivisions have much better performance than scene 3. With instancing, this is exactly what I was expecting. With scene 3, we must send 9 times the amount of data (9 sponges) in a buffer compared to scene 4, in which we send enough data for 1 sponge. With such a difference in data being sent over each frame, I was expecting a significant change.

## 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.

Answer here.

Is this what you expected? Why or Why Not?

Answer here.

## SCENE 6

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

Answer here.

Discuss what the data is showing.

Answer here.