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 choose the Menger Sponge. I used my own algorithm where I have a Cube data structure that can possibly contain children. If a subdivision is requested, then the Cube would have 8 children create at each corner and they all take 1/9th of the volume of the original.

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?

The Graphics Card used for testing was a Nvidia RTX 2080 Super, the CPU used for testing was an AMD Ryzen 7 5800X, both at stock speeds. The RAM was clocked at 2133MHz. The screen resolution was 2160px width by 1440px height and the refresh rate was 144fps. For performance testing, Vsync was turned off.

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 used my own algorithm where I have a Cube data structure that can possibly contain children. If a subdivision is requested then the Cube would have 8 children create at each corner and they all take 1/9th of the volume of the original.

All vertices are duplicated for every face of the cube, in total there would be 32 vertices comprised of 6 faces and 12 triangles.

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?

There are 6 materials, one for each face. Due to each face having duplicate vertices the materials produced a flat look with specular highlighting done in the fragment shader. A material ID was passed in the vertex buffer object.

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?

For a total of 8 lights including the directional light, one light was placed in the center of the menger sponge. The other 6 lights were placed 0.2 units to the normal of the face of the root cube.

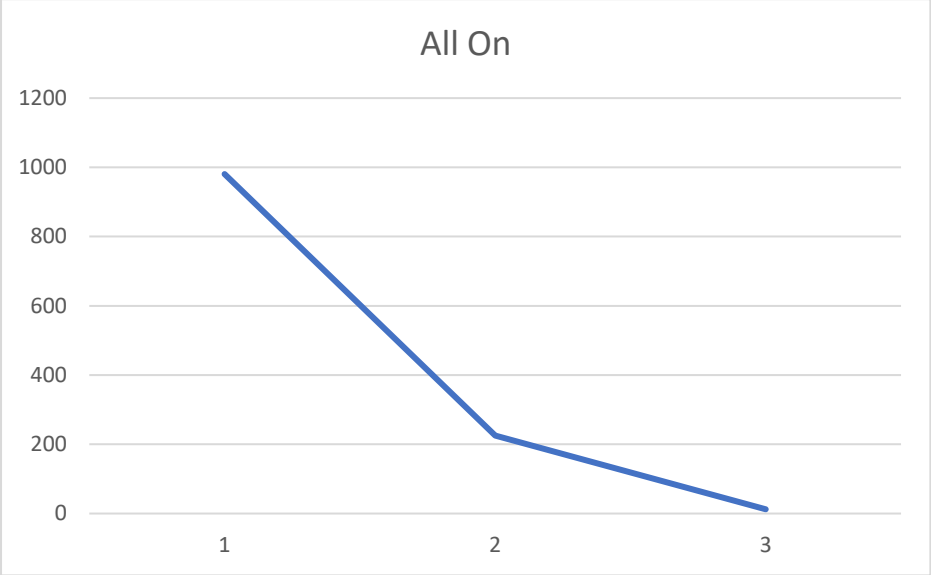
SCENE 1

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.

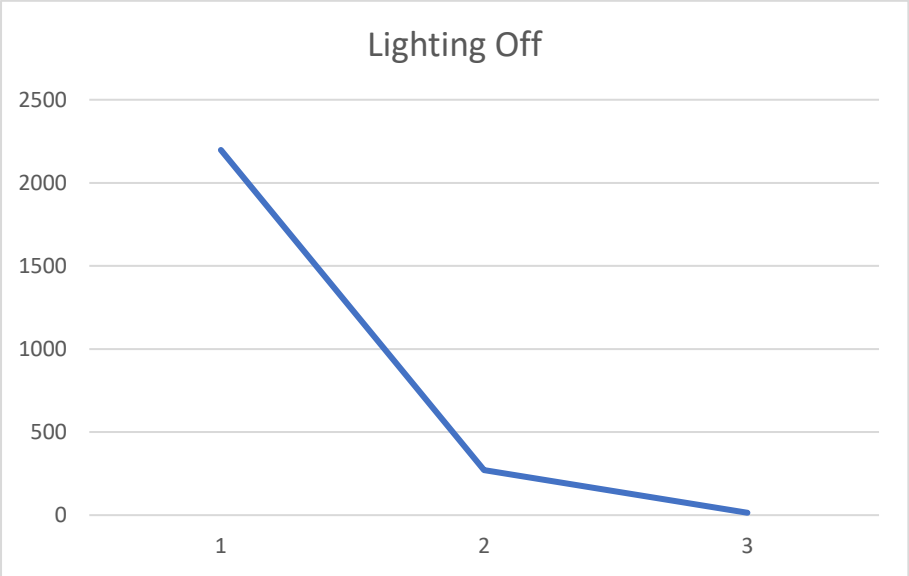
Subdivision	All On
1	980.583467
2	225.098886
3	12.236337

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



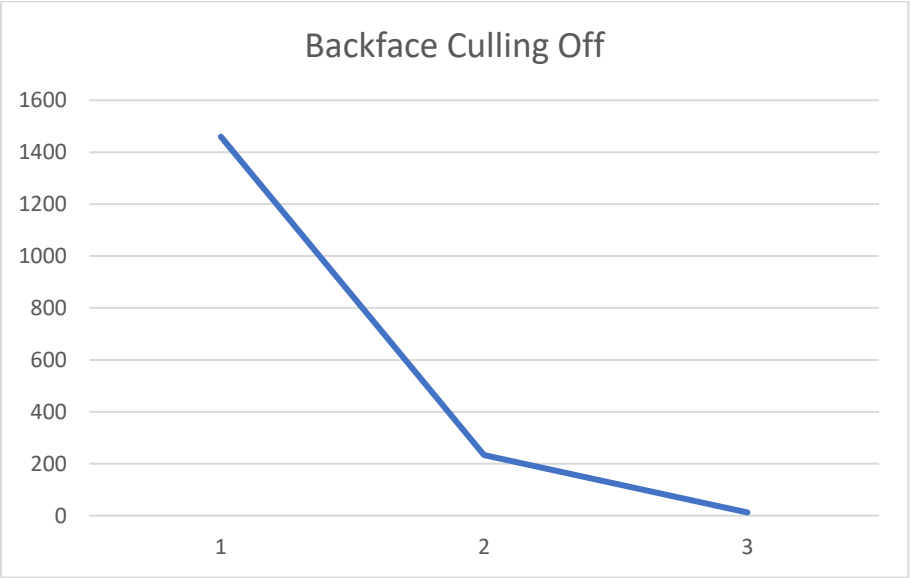
Lighting Off
2197.609001
271.827031
14.250254

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.



Comparing the graph to the original for scene 1, with lighting off the framerate has increased. This is because lighting performs additional calculations.

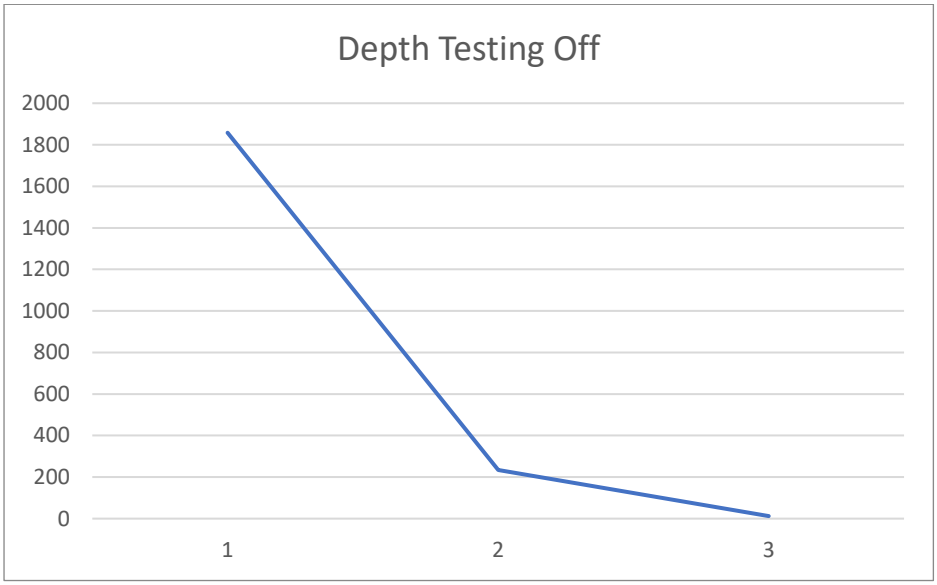
Run some tests with Backface Culling On and Off, while keeping everything else as above. Describe the impact this feature has on frame rate and why? Use a table and a chart to show the data.



Backface Culling Off
1459.660104
233.258947
12.324723

Comparing the graph to the original for scene 1, backface culling has no impact on the performance in immediate mode.

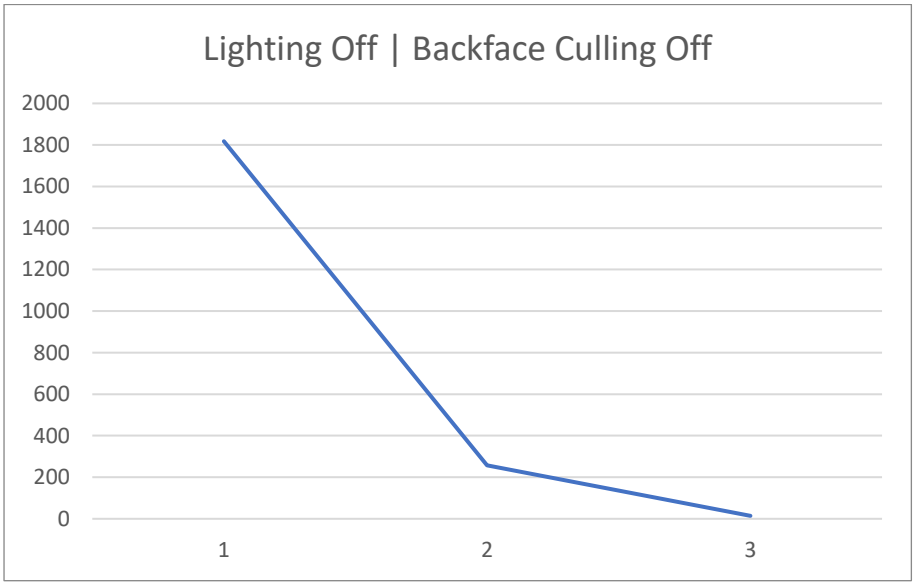
Run some tests with Depth Testing On and Off, while keeping everything else as above. Describe the impact this feature has on frame rate and why? Use a table and a chart to show the data.



Depth Testing Off
1857.934868
233.874746
12.206779

Comparing the graph to the original for scene 1, depth testing has no impact on the performance in immediate mode.

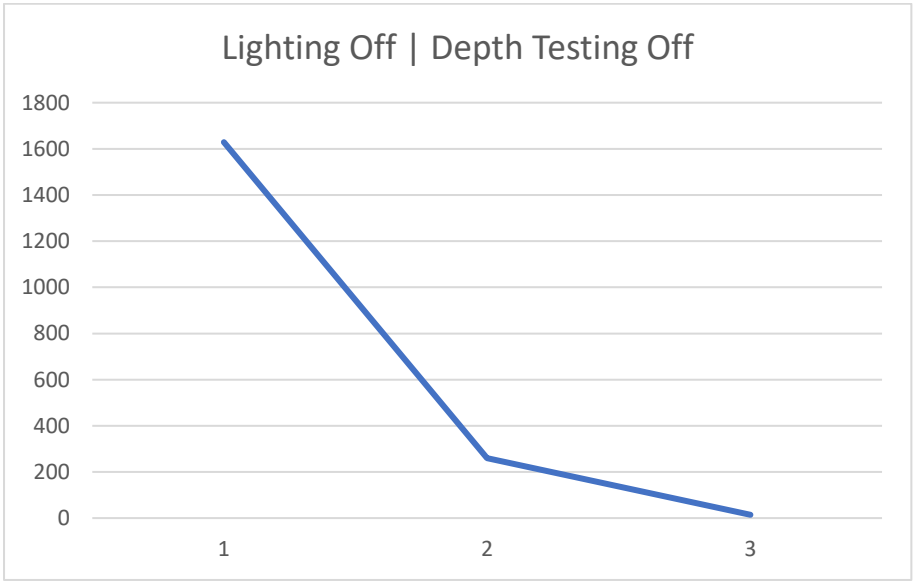
Run some tests with Backface Culling On and Off in combination with Lighting On and Off, while keeping everything else as above. When Lighting is On is there a difference in Frame Rate when Backface Culling is On vs Off? Describe Why or Why Not and show data to support your answer. Did you expect there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



Lighting Off | Backface Culling Off
1817.76208
257.617956
14.260995

Comparing the graph to the “lighting off” graph, backface culling now effects performance at subdivision 0. It has no effect on other subdivisions. I didn’t expect a difference due to thinking that the fixed function calls would have more overhead than the performance improvement from backface culling.

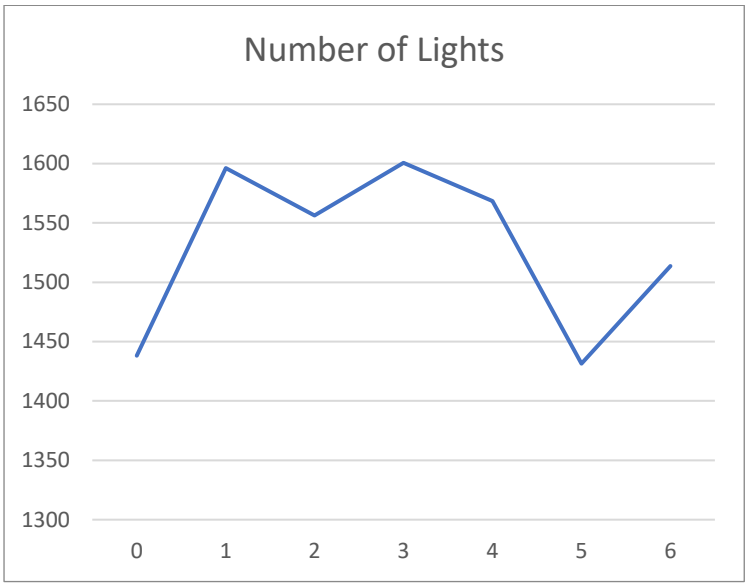
Run some tests with Depth Testing On and Off in combination with Backface Culling On and Off, while keeping everything else as above. When Depth Testing is On is there a difference in Frame Rate when Backface Culling is On vs Off? Describe Why or Why Not and show data to support your answer. Did you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



Lighting Off | Depth Testing Off
1628.895299
259.875665
14.274753

Comparing the graph to the “lighting off” graph, depth testing culling now effects performance at subdivision 0. It has no effect on other subdivisions. I didn’t expect a difference for depth testing as it doesn’t attempt to prevent anything from being drawn, only organizes based on a very simple ‘if statement’.

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.



Number Lights	FPS
0	1438.298435
1	1596.166646
2	1556.185292
3	1600.655629
4	1568.438008
5	1431.477328
6	1513.660789

The shape of the graph indicates that lighting in immediate mode doesn't have much effect on performance compared to the other OpenGL instructions.

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

Answer here.

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?

Normal vectors are stored in a separate array. For every vertex position, there is one normal vector, in other words the normal vectors are per vertex. Each vertex is duplicated 3 times to make the 3 separate but connecting faces of a cube.

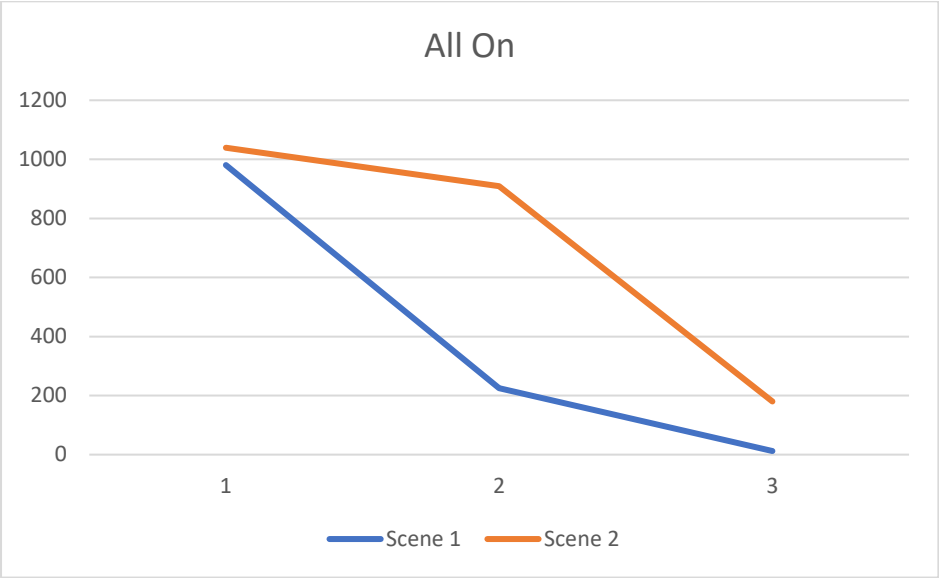
*Vary the subdivision level and move around the scene. Describe the performance characteristics you're seeing at the different levels of subdivision? Is the scene getting smoothly animated as you move around? Does it seem to speed up and slow down depending on what's currently being rendered? Why? At 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'?*

At a subdivision of level 4 my machine slows down to less than 1fps. I suspect the cause is not a fault with the program but a hardware fault. For every other subdivision the program runs smoothly. Turning off vsync shows frames consistently higher than the refresh rate.

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.

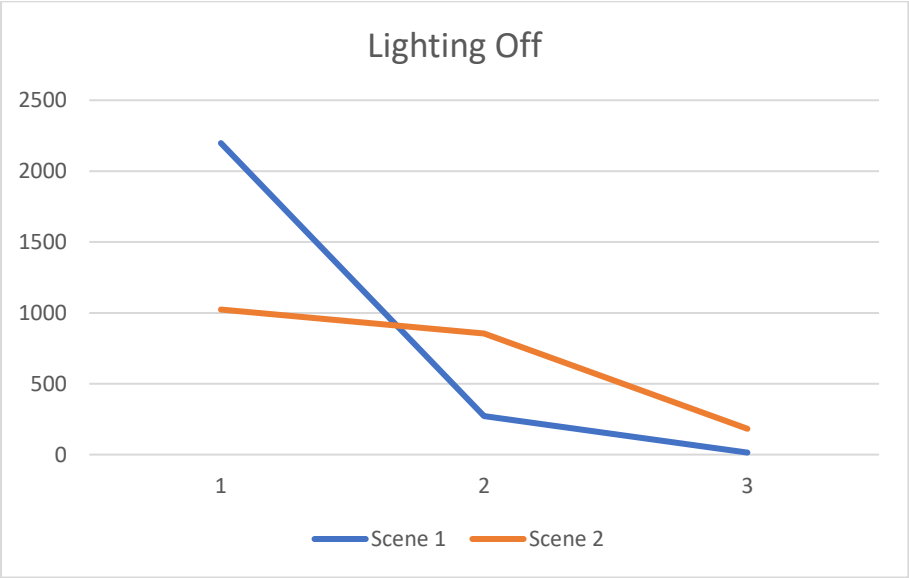
Subdivision	Scene 2
1	1039.288212
2	908.571371
3	180.072915

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?



The graph tells me that immediate mode is better suited for a low number of vertices and 'glBegin' operations. For anything else, modern OpenGL will give greater performance.

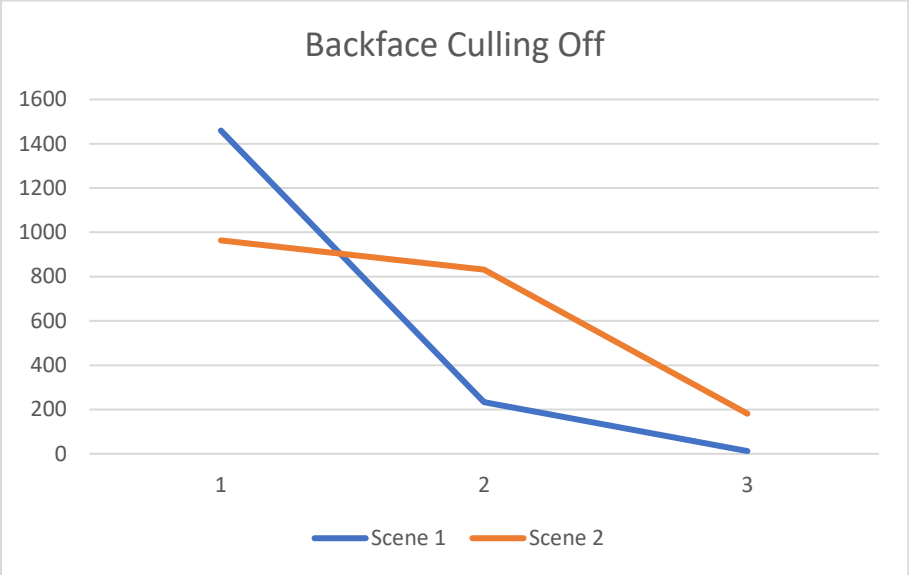
Run some tests with Lighting Off while keeping everything else as above. Are the performance characteristics similar as for Scene 1? Why or Why Not? Use a table and a chart to show the comparison.



Lighting Off	
Scene 1	**Scene 2**
2197.609001	1023.408421
271.827031	854.144566
14.250254	182.243605

Turning off lighting improves performance more-so for immediate mode. Shaders offer greater performance but have an initial overhead per-frame.

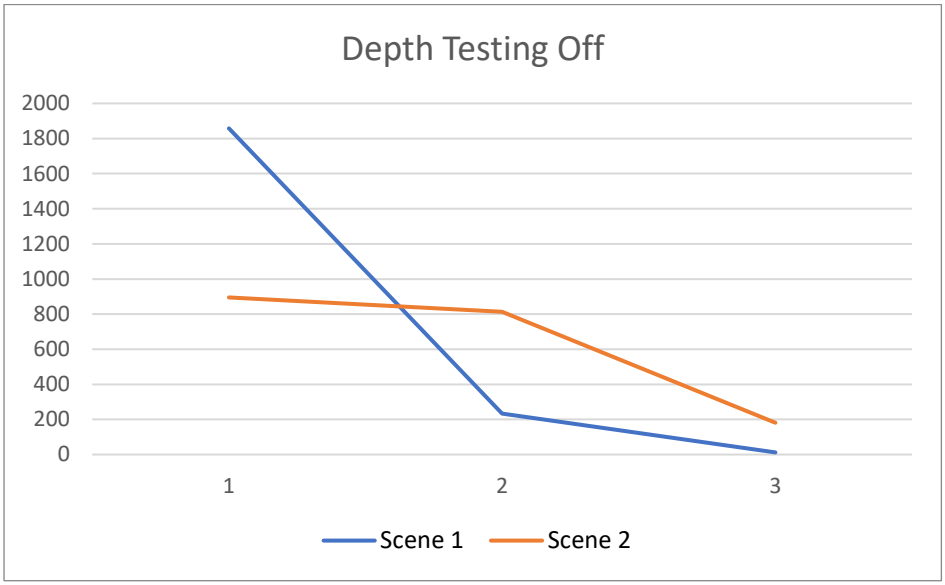
Run some tests with Backface Culling On and Off, while keeping everything else as above. Are the performance characteristics similar as for Scene 1? Why or Why Not? Use a table and a chart to show the comparison.



Backface Culling Off	
Scene 1	**Scene 2**
1459.660104	963.572155
233.258947	832.100438
12.324723	181.19149

Backface culling has no effect on performance. The performance characteristics are like scene 1. This was not expected.

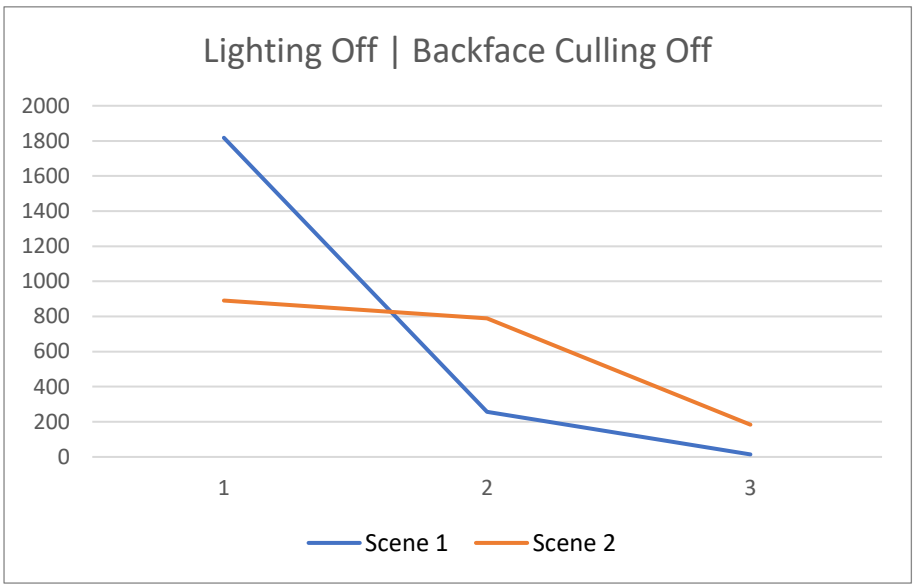
Run some tests with Depth Testing On and Off, while keeping everything else as above. Are the performance characteristics similar as for Scene 1? Why or Why Not? Use a table and a chart to show the comparison.



Depth Testing Off	
Scene 1	**Scene 2**
1857.934868	894.902635
233.874746	812.469126
12.206779	181.21477

Backface culling has no effect on performance. The performance characteristics are like scene 1. This was expected.

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?

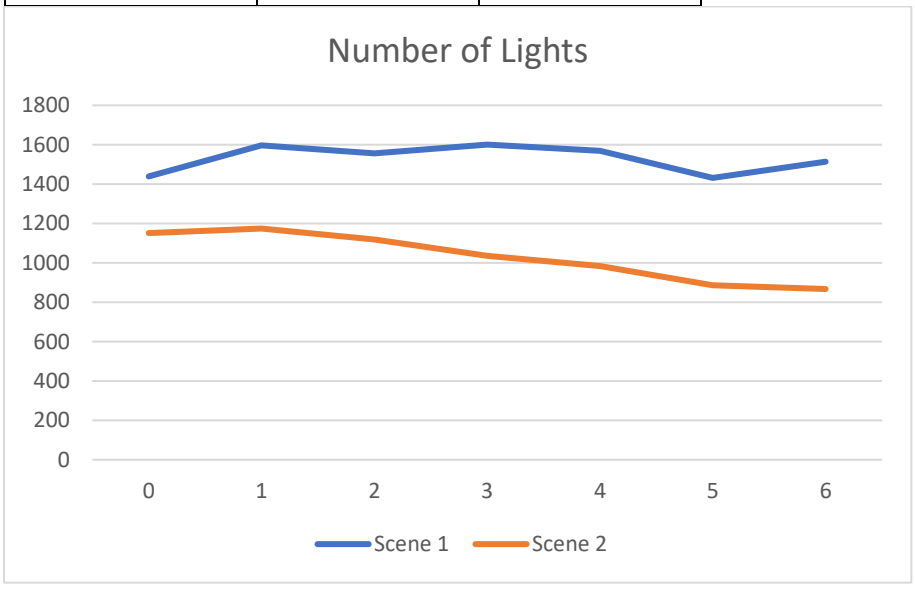


Lighting Off | Backface Culling Off	
Scene 1	**Scene 2**
1817.76208	890.5917
257.617956	789.986136
14.260995	183.622018

Backface culling has no effect on performance. The performance characteristics are like scene 1. This was not expected.

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?

FPS		
Number Lights	**Scene 1**	**Scene 2**
0	1438.298435	1151.679667
1	1596.166646	1174.297447
2	1556.185292	1118.650818
3	1600.655629	1035.026326
4	1568.438008	983.986602
5	1431.477328	885.90673
6	1513.660789	867.129714



The frames are following a straight linear path instead of being random. The frames are consistently lower but this is to be expected, the fixed function pipeline was intended to be fast for as few operations whereas modern opengl requires preparation per frame.

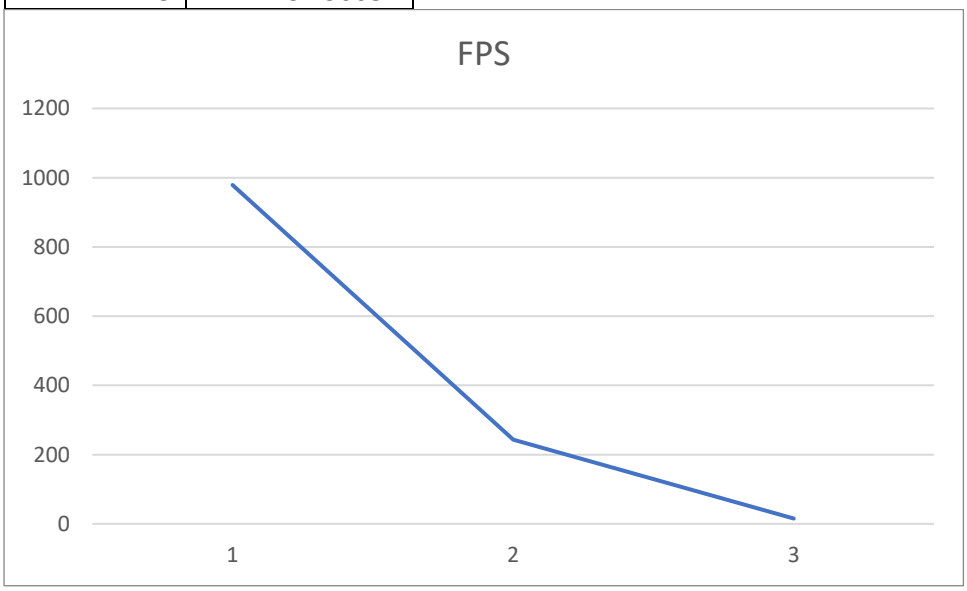
Is there anything you found interesting or unexpected while running the above tests? Explain why.

Answer here.

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	Default
1	978.898856
2	243.495208
3	15.290681

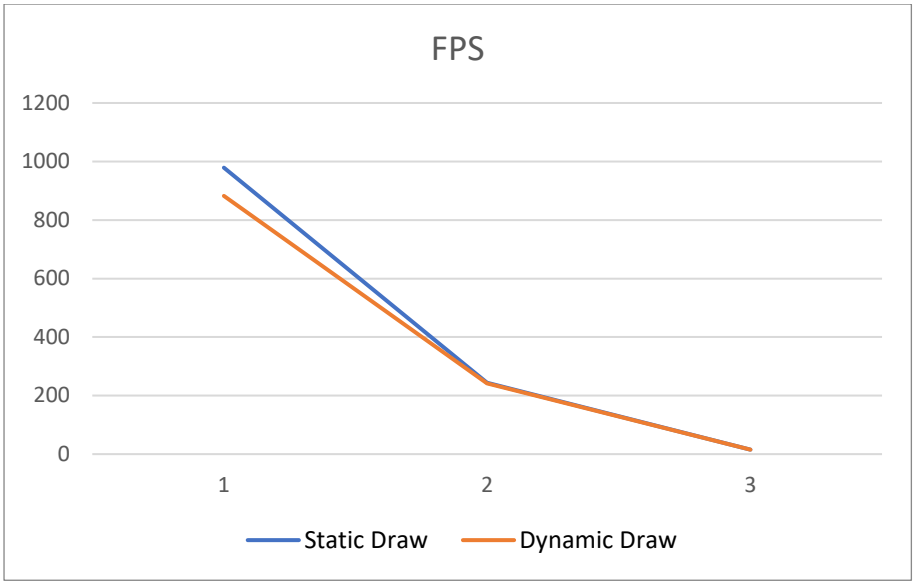


Is this what you expected? Why or Why Not?

This result was expected due to the rotation math that is being ran. It impacts the framerate greatly. If it were disabled the frames would improve, but not nearly as much as instancing.

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	Dynamic Draw
1	978.898856	882.508938
2	243.495208	241.234387
3	15.290681	15.27214



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

The results are what were expected, a slight performance loss. This is because dynamic draw is an unfixed size so operations have to happen to allow for the data to be reallocated when requested.

SCENE 4

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

Yes, there is an improvement that allows for subdivision 4 to be reached. I expected this due to instancing reduces the number of function calls that need to be called via the OpenGL.

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.

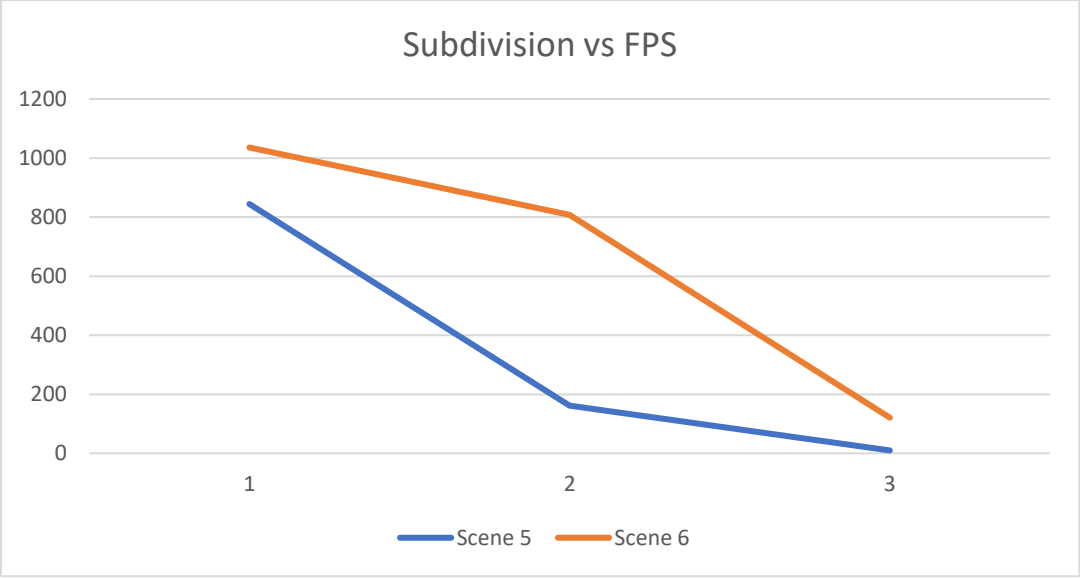
Is this what you expected? Why or Why Not?

Yes, because the original vertex data does not exist anymore and is being constantly overridden using ‘glSubBufferData’.

SCENE 6

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

Subdivision vs FPS		
Subdivision	**Scene 5**	**Scene 6**
1	844.216694	1035.720981
2	161.370748	807.866683
3	9.789917	120.764619



Discuss what the data is showing.

The data shows that updating the vertex positions on the GPU through the vertex shader would bring higher frames. This is because a GPU is built to iterate over all vertices and perform the same calculation on many cores. A CPU needs to halt, iterate over the vertices, and then finally send the data.