Write Up on Project 3C - by Tien Dinh

I. Introduction:

This project runs some experiments to test the performance versus quality in OpenGL. There will be some combination of ball/sphere rendering on screen with a bouncing physic. Each ball is rendered by a certain number of triangles based on its level of detail. The higher level of details meaning more recursive deep, meaning more triangles. The number of triangles for each recursive level is shown below:

- At L1 = 5 level of recursion, a sphere is rendered using 24576 triangles
- At L2 = 6 level of recursion, a sphere is rendered using 98304 triangles
- At L3 = 7 level of recursion, a sphere is rendered using 393216 triangles
- At L4 = 8 level of recursion, a sphere is rendered using 1572864 triangles

The goal is to test the maximum number of triangles that the GPU can handle before dropping FPS. The capped FPS is 60fps.

II. Experiments:

We first test the low range and high range of triangles, then narrows down to find the middle point where FPS just starts to drop.

- At 100 L1 balls: 2,457,600 triangles, FPS stays consistently at 60fps.
- At 100 L2 balls: 9,830,400 triangles, FPS drops down to 37fps.
- At 100 L3 balls: 39,321,600 triangles, FPS drops significantly to 9fps and stops rendering sometimes.

The optimal number of triangles fall between 2.4m and 9.8m. After some trials and errors, some combination of balls show that the optimal number is around 5-5.4 million triangles:

- At 220 L1 balls: 5,406,720 triangles, FPS starts to drop to 58fps.
- At 204 L1 balls: 5,013,504 triangles, FPS stays consistently at 60fps.

To confirm the 5-5.4 million number, more experiments made:

- Combination: 90 L1 + 5 L2 + 5 L3 balls: 4,669,440 triangles, FPS stays at 60fps.
- Combination: 85 L1 + 5 L2 + 10 L3 balls: 6,512,640 triangles, FPS drops to 46fps.

Some experiments with a small number of L4 balls show the same result:

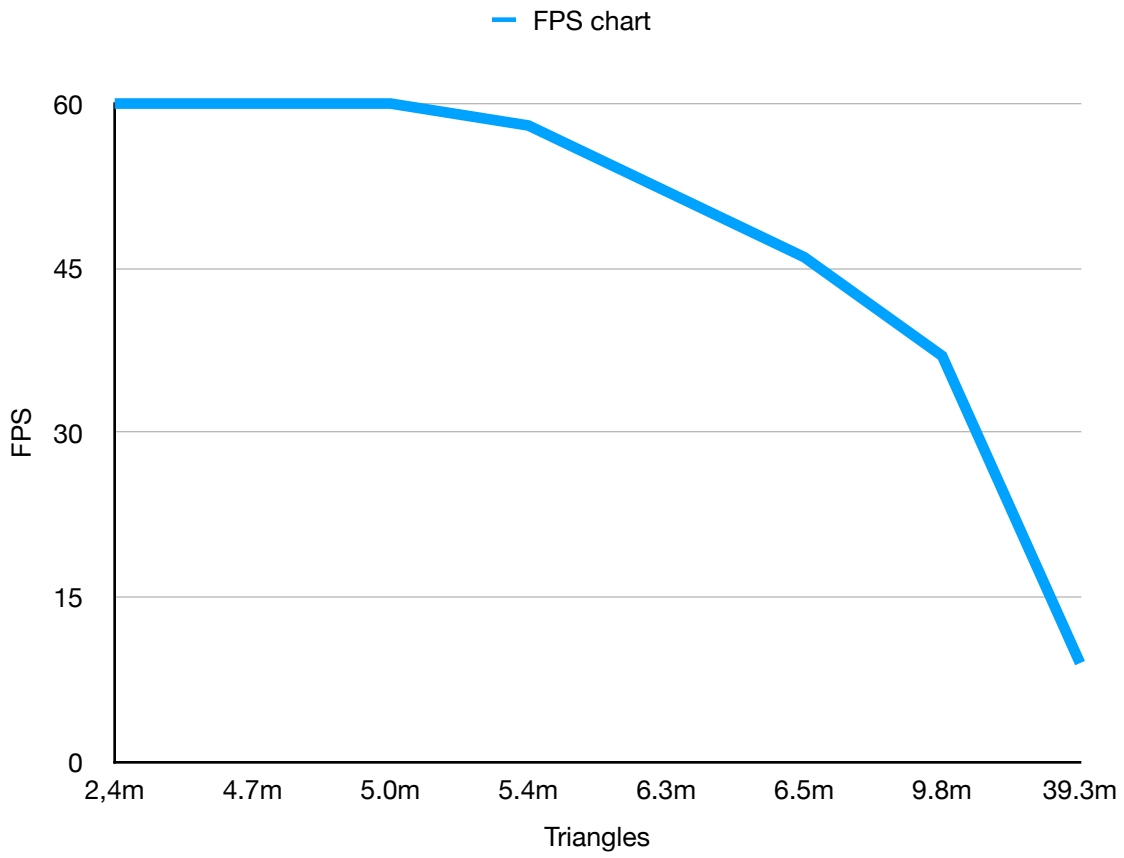
- At 3 L4 balls: 4,718,592 triangles, FPS stays consistently at 60fps.
- At 4 L4 balls: 6,291,456 triangles, FPS drops to 52fps.

From the experiments, we can draw conclusion

III. Conclusion:

There is a trade-off between GPU performance and rendering quality. The more sphere/ triangles that GPU has to render, the lower performance it gets. In our setting (bellow section IV), GPU can handle maximum of 5 to 5.4 million triangles before FPS start to drop below 60fps. This is also considering that the closer the sphere rendering to the camera, the more details (triangles) it renders, thus lower the performance as well. When using 220 L1 balls combination, when the balls group stays closer to the camera, FPS drops to 58fps. However, when the 220 L1 balls stay furthest away from the camera, FPS stays at 60fps. With L4 = 8

recursion level, the number of triangles increase significantly (1,572,864 triangles per L4 balls). FPS starts to drop noticeably when increasing from 3 L4 balls to 4 L4 balls.
All the experiments are described in the below graph:



IV. Machine settings:
The experiments is running on macOS Catalina version 10.15.7
Renderer: Intel Iris OpenGL Engine
OpenGL version supported 4.1 INTEL-14.7.8
Processor: 2.6 GHz Dual-Core Intel Core i5
Memory: 8 GB 1600 MHz DDR3
Graphics: Intel Iris 1536 MB