Assignment 1 - Report

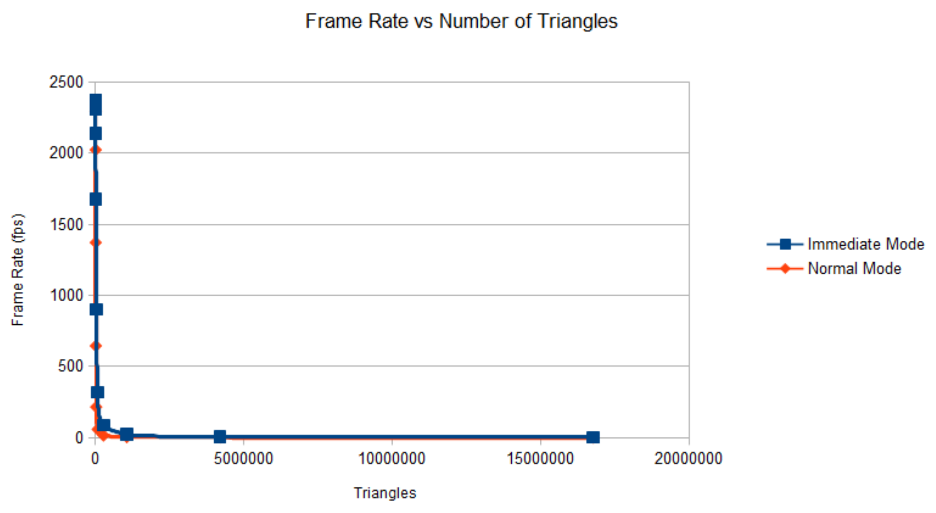
1. When using immediate mode, of pre-computing and storing coordinate and normal data compared to regeneration every frame.

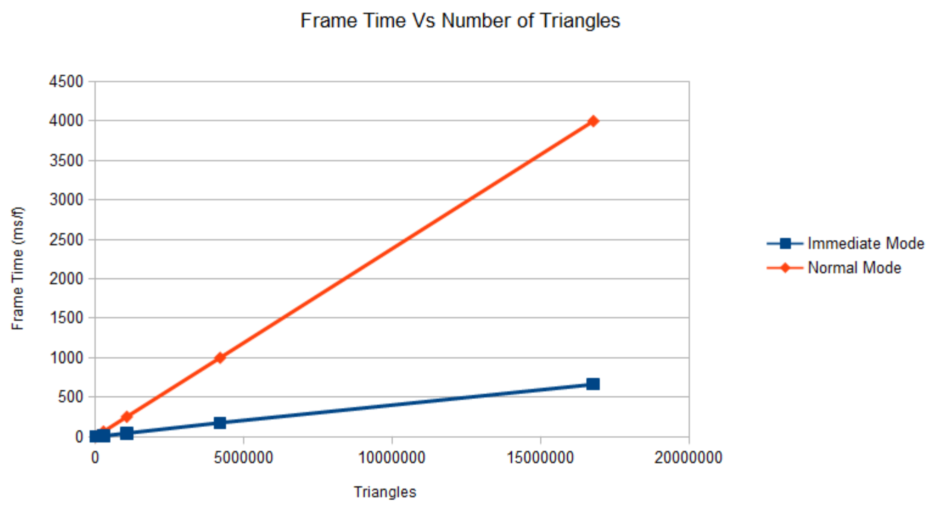
Regenerating data every frame definitely hinders the performance and has a negative impact on the application. Compared to immediate mode, the difference in frame rate is significant, even with as little as 64 polygons. And as we double the tessellation (or quadruple the polygons), the difference in frame rate becomes more and more significant.

Observation – Between tessellation 8 to 512, the frame rate achieved with immediate mode is approximately N times the frame rate of normal mode, with N being incremented by one in each iteration.

With tessellation 8 - FR (IM) = 3*FR (NM)

With tessellation 16 - FR (IM) = 4*FR (NM) and so on, until a tessellation of 512





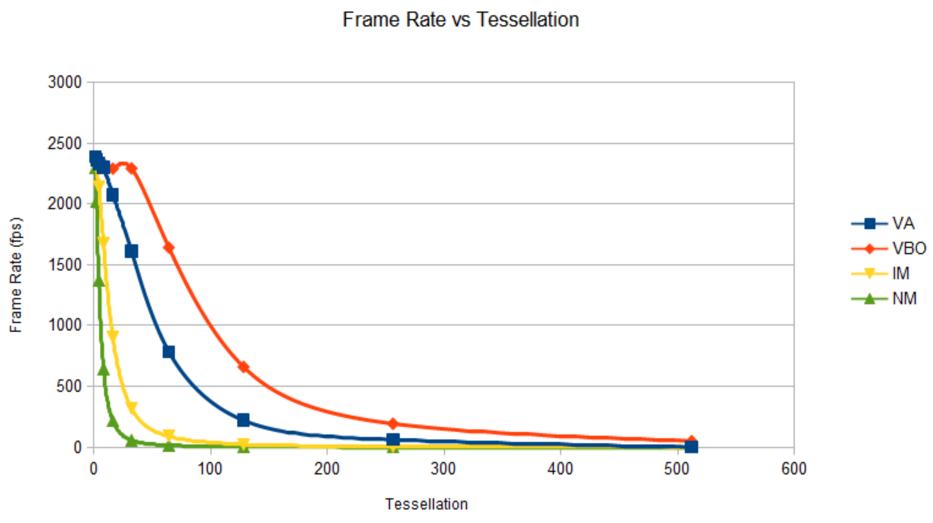
2. Increasing tessellation

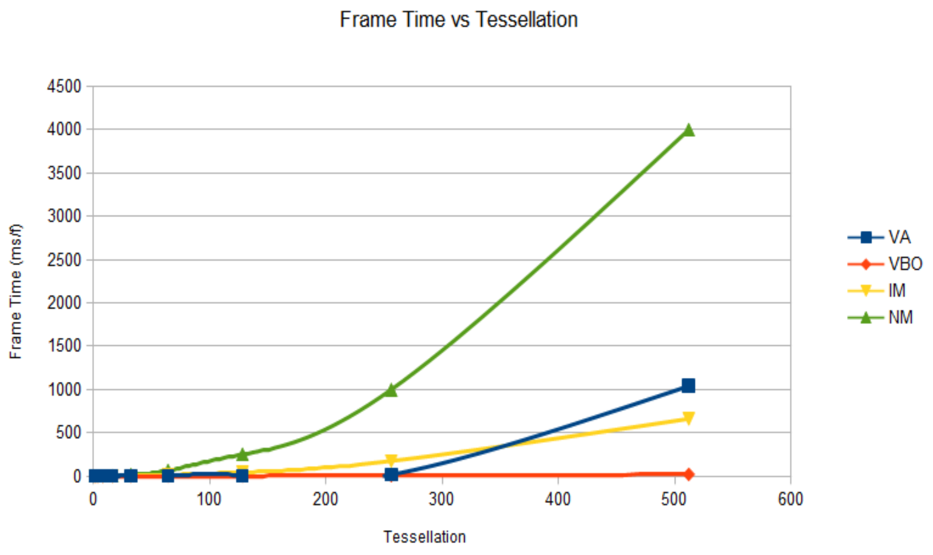
In normal mode (NM), increasing the number of polygons reduces the frame rate by about the same factor.

In immediate mode (IM), frame rate drop is not as much as normal mode initially, but over time, as the number of polygons increases; a similar trend is observed.

Vertex arrays (VA) performs significantly better than immediate mode. The frame drop is very little up until a tessellation of 16 and then the frame rate drops to half, then one third and then one forth of the previous iteration. Thereafter, the frame rate becomes consistent for the next two iterations until at last when it drops to a single frame with a tessellation of 512. It must be noted that when the data was plotted, IM performed better than VA for the last observation (at a tessellation of 512), which is contrary to my expectations.

Vertex buffer objects (VBO) have the best performance due to the obvious reason that the data is stored on GPU. The frame rate is substantially high and consistent up until tessellation of 32. Thereon, the frame rate drop becomes quite steep. However, as the number of polygons increase to large numbers, the frame rate becomes more consistent.



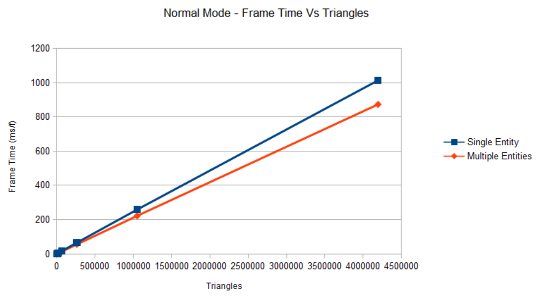


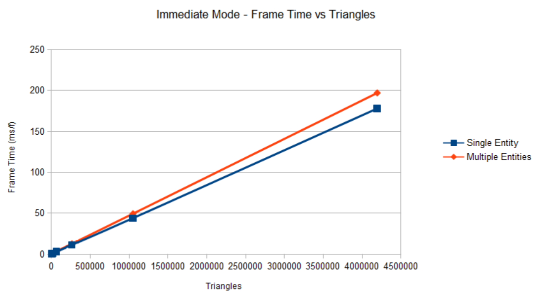
3. Increasing the number of objects being drawn.

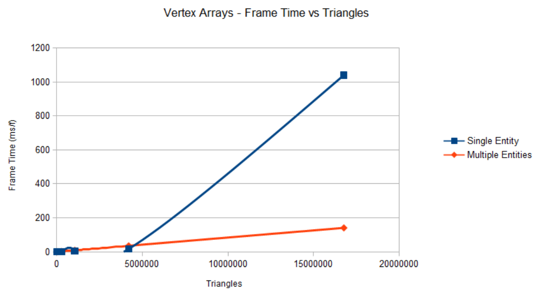
Normal and immediate mode don't show much of a difference with increasing the number of objects. In normal mode, a slightly higher frame rate is observed with increasing the number of objects compared to increasing the tessellation on one sphere/torus. Immediate mode displays otherwise: the frame rate is slightly lower compared to a highly tessellated sphere.

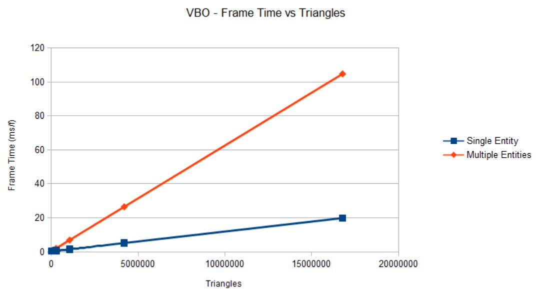
In vertex arrays, the frame rate drops to almost half of the frame rate with the same number of polygons in one sphere, but with a high tessellation of 512. It performs better with multiple objects compared to a single object.

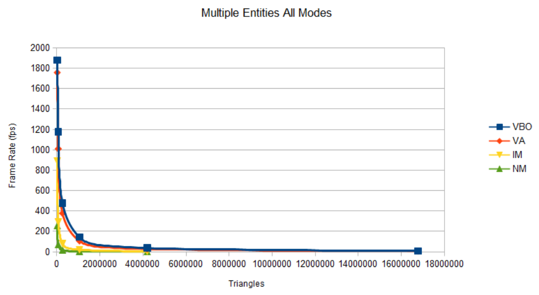
Vertex buffer objects show a similar trend to vertex arrays. The frame rate drops to one forth of the frame rate of a sphere with the same number of polygons.

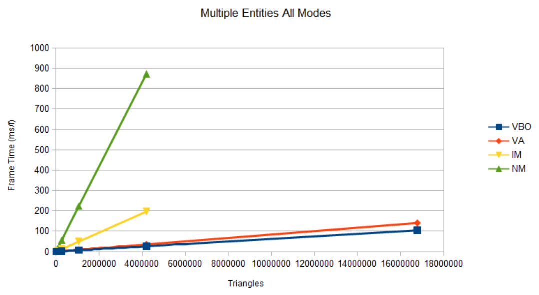












4. Using immediate mode, vertex arrays and vertex buffer objects.

Immediate mode has extremely poor efficiency as compared to vertex arrays and vertex buffer objects. Making multiple vertex calls to GPU hinders performance, is not efficient and can become a bottleneck when millions of polygons are involved. Vertex arrays provide a pointer to data to the GPU, which is then copied to server space for rendering and improves the performance as there is less overhead on the bus. Vertex buffer objects are even better as all of the data is stored on the GPU and instantly accessible, requiring no commits over the wire. Vertex arrays should generally be used when data changes constantly whereas Vertex buffer objects should be used for geometry that is less likely to change and is constantly required for the scene.

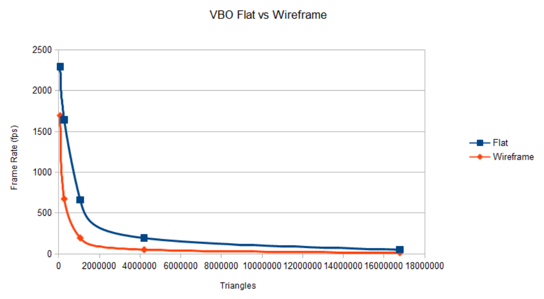
5. Changing rendering mode from wireframe to flat to smooth shaded.

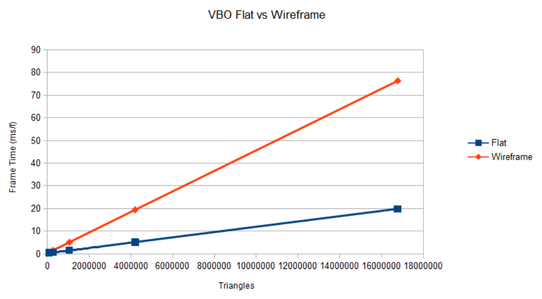
There is no/minimal difference between rendering mode flat and smooth shaded.

In immediate and normal mode, no/minimal difference is observed between flat and wireframe mode.

In vertex arrays, a lower frame rate is observed with wireframe until 2560000 polygons. With further increase in polygons, frame rate is more or less consistent with flat/smooth shaded.

Vertex buffer objects show a consistent drop in frame rate with wireframe mode. As the number of polygons increase, frame rate difference increases between flat and wireframe mode. At tessellation 16, frame rate drop is 20 percent to that of flat mode, and at tessellation 512, frame rate drops by 70 percent to that of flat mode.

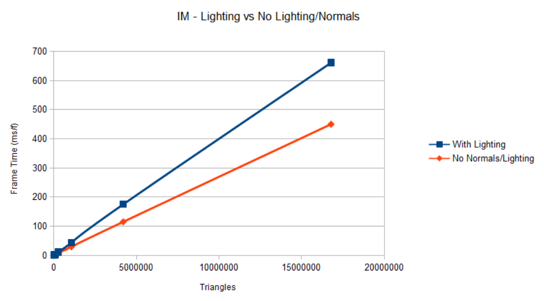


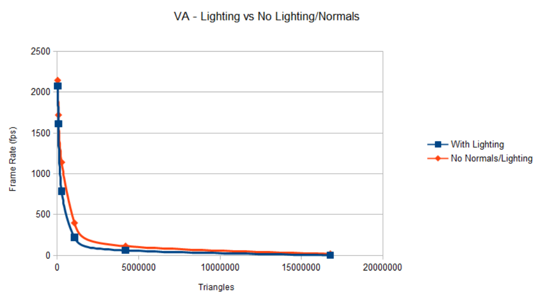


6. Turning lighting on and off, and specifying or not specifying normals accordingly.

No difference is seen in performance with Vertex buffer objects with lighting on or off.

An increase in frame rate is observed with lighting off in vertex arrays, immediate and normal mode.





7. Increasing the number of lights, up to a maximum of 8.

No difference is seen with increasing the number of lights in any mode.

8. Increasing the window size and zooming in, thereby increasing the number of pixels to be filled.

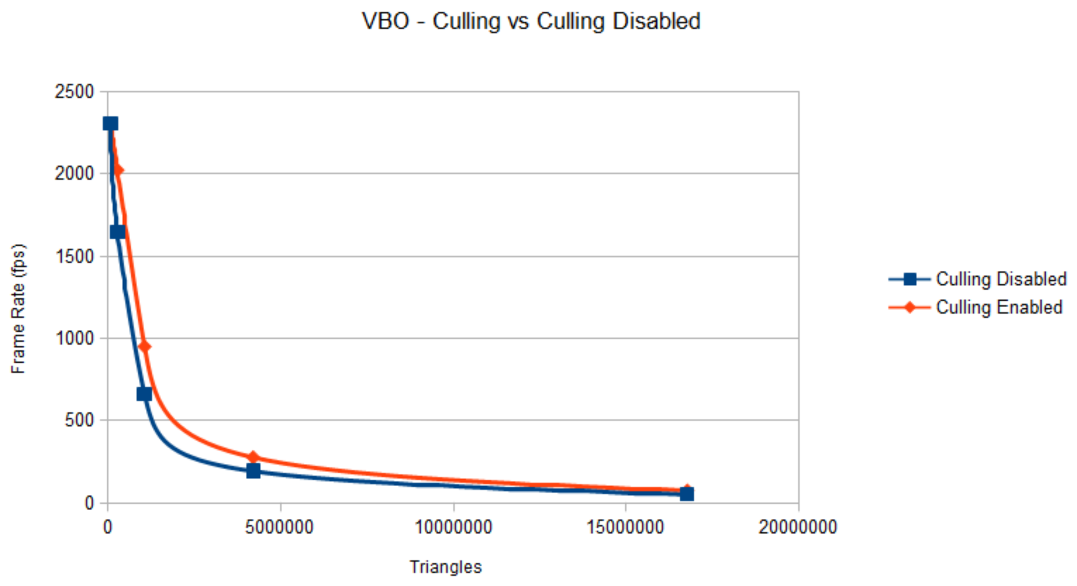
Increasing the window size and zooming in shows no difference in normal, immediate or vertex arrays.

When the camera is zoomed in enough to clip the scene, an increase in frame rate is observed with Vertex buffer objects. However, the increase in frame rate is not great enough.

9. Backface culling.

Backface culling has no effect on normal, immediate or vertex arrays.

An increase in frame rate is observed with vertex buffer objects with backface culling.



10. Number of lit shaded polygons drawn at frame rate of around 30fps.

Normal Mode – 123904 polygons at frame rate of around 31 (tessellation = 44)
Immediate Mode – 774400 polygons at frame rate of around 31 (tessellation = 110)
Vertex Arrays Mode – 7840000 polygons at frame rate of around 31 (tessellation = 350)
Immediate Mode – 27040000 polygons at frame rate of around 31 (tessellation = 650)

11. Time taken to render a single lit object of a million polygons.

Normal Mode – 245.4 ms
Immediate Mode – 40.8 ms
Vertex Arrays – 4.34 ms
Vertex Buffer Object – 1.47ms