Into The Night optimization document Nils Meijer 466301

Important notes:

* My game is not meant to run on a standalone Oculus Quest. Always stay connected to a VR-capable PC.

My specifications:

* + AMD R7 3700X (8 cores, 3.59 GHz)
  + RTX 2070
  + 16GB
  + Windows 11
* Tests must always be performed in a build of the game, *not* in the editor. The editor is most often not suitable for performance analysis. Use the standalone unity profiler to monitor performance.
* LOD groups will not be tested, as the environment asset pack I used has these implemented already.
* I am not using HDRP, the mentioned asset pack threw shader errors when I tried upgrading its materials. Still have to try URP. It’s fine if either doesn’t work, the purpose is to learn VR development, making a beautiful game is the 2nd priority in this case.
* For these tests, when I mention “performance”, I always mean:
  + Average FPS over a constant span of time (can be converted to average milliseconds per frame).
  + Amount & intensity of lag spikes.

Testing setup/steps:

1. Test case ID
2. Test description
3. Assumptions and pre-conditions
4. Test data (what are the variables I’m going to test?)
5. Steps to be executed
6. Expected result
7. Actual result and post-conditions
8. Pass/fail (implement or not implement this specific optimization?)

**Researched optimization techniques:**

Occlusion culling

“The process which prevents Unity from performing rendering calculations for GameObjects that are completely hidden from view (occluded) by other GameObject.” (Unity, 2021)

1. **Test description**

Perform 3 tests. Check if performance increases & if objects that should(n’t) be rendered do get or do not get rendered.

Tests:

1. Occlusion culling disabled
2. Occlusion culling enabled with “starting voxel” size.
3. Occlusion culling enabled with smaller “voxel” size.
4. **Assumptions and pre-conditions**

A script is attached to the player (so including the camera). It moves the player forward with a constant speed, for a constant amount of time.

Speed = 1 unit/s

Time = 10 seconds

As with all tests, it’s run in a build.

1. **Test data (what are the variables I’m going to test?)**

The performance (average fps, lag spikes) of the game, using different occlusion culling parameters.

1. Steps to be executed
2. Build a new version of the game, changing only parameters of occlusion culling, depending on the index of this test (0 = no culling, 1 = some culling, 2 = full culling).
3. Open the independent profiler
4. Run the application
5. Application terminates automatically after Time.
6. Save the profiler data. Possibly screenshot relevant information for this document.
7. Expected result

Best performance with full culling (test 3 out of 3)

1. Actual result

TBD

1. Pass/fail (implement or not implement this specific optimization?)

Does performance get better, at all? If not, test failed (no optimization possible)

Object pooling

“Object pooling is where you pre-instantiate all the objects you’ll need at any specific moment before gameplay – for instance, during a loading screen. Instead of creating new objects and destroying old ones during gameplay, your game reuses objects from a “pool”. (Placzek, 2016)

Async scene-loading

“Load Scene Async loads the scene in the background and is spread over multiple frames. In general, it is recommended to use the Async method since it is much more efficient spreading the loading over several frames instead of one, it works perfectly in a Player Build, however in the editor itself it might stutter and freeze because the Editor does not support background operations very well.” (Coppens, 2021)

Static Batching

“Static batching is a [draw call batching](https://docs.unity3d.com/Manual/DrawCallBatching.html) method that combines meshes that don’t move to reduce [draw calls](https://docs.unity3d.com/Manual/optimizing-draw-calls.html). It transforms the combined meshes into world space and builds one shared vertex and index buffer for them. Then, for visible meshes, Unity performs a series of simple draw calls, with almost no state changes between each one. **Static batching** doesn’t reduce the number of draw calls but instead reduces the number of render state changes between them.

Static batching is more efficient than [dynamic batching](https://docs.unity3d.com/Manual/dynamic-batching.html) because static batching doesn’t transform vertices on the CPU.” (Unity, 2022)

Graphics Quality Levels

“Unity allows you to set the level of graphical quality it attempts to render. Generally speaking, quality comes at the expense of framerate and so it may be best not to aim for the highest quality on mobile devices or older hardware since it tends to have a detrimental effect on gameplay.” (Unity, 2017)

Baked Lighting

“Mixed Lights combine elements of both real-time and baked lighting. You can use Mixed Lights to combine dynamic shadows with baked lighting from the same light source, or when you want a light to contribute direct real-time lighting and baked indirect lighting.” (Unity, 2022)

# Bibliography

Coppens, G. (2021, 9 19). *Loading Scenes In Unity*. Retrieved from medium.com: https://medium.com/geekculture/loading-scenes-in-unity-98e446756497

Placzek, M. (2016, 11 23). *Object Pooling in Unity*. Retrieved from raywenderlich.com: https://www.raywenderlich.com/847-object-pooling-in-unity

Unity. (2017, 9 18). *Quality*. Retrieved from docs.unity3d.com: https://docs.unity3d.com/Manual/class-QualitySettings.html

Unity. (2021). *Occlusion Culling*. Retrieved from docs.unity3d.com: https://docs.unity3d.com/Manual/OcclusionCulling.html

Unity. (2022, 6 3). *Light Mode: Mixed*. Retrieved from docs.unity3d.com: https://docs.unity3d.com/Manual/LightMode-Mixed.html

Unity. (2022, 6 3). *Static batching*. Retrieved from docs.unity3d.com: https://docs.unity3d.com/Manual/static-batching.html