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.
- Tests will be performed without my headset connected. What matters is the profiling data.
- 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.
- Tests will be done *in order*. This means that the (supposedly positive) impact of the previous *optimization technique* will be included in the test to be executed. However, since tests of a specific optimization technique will obviously only be compared to one another (making it relative), it should not affect the results negatively.
- For all tests, the EnemySpawner will be disabled so those calculations do not influence the test results.

Testing setup/steps:

1. Test description
2. Assumptions and pre-conditions
3. Test variables
4. Steps to be executed
5. Expected result
6. Actual result and post-conditions
7. Pass/fail (implement or not implement this specific optimization?)

Researched optimization techniques + tests:

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.

2. **Assumptions and pre-conditions**

   A script is attached to the player (so including the camera). It moves the player forward at 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. Player input will not be required.

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

4. **Steps to be executed**
   4.1 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).
   4.2 Open the independent profiler
   4.3 Run the application
   4.4 Application terminates automatically after Time.
   4.5 Save the profiler data. Possibly screenshot relevant information for this document.

5. **Expected result**

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

   Test 0 (no culling): lag spikes, low average fps. Many drawcalls.

   Test 1 (culling enabled): less/lower intensity lag spikes. Better average fps. Much less drawcalls.

6. **Actual result**
   TBD

7. **Pass/fail (implement or not implement this specific optimization?)**
   Does performance get better, at all? If not, the 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)

1. **Test description**
   Perform 2 tests. One with no object (bullet) pooling at all, and one with bullet pooling.

2. **Assumptions and pre-conditions**
   For simplicity sake, the assault rifle will be the weapon of focus here. For Time, the rifle shoots a bullet every Interval.

   Time = 10 seconds

   Interval = 0.1 seconds

3. **Test variables**
   The performance over time, including lag spikes and starting/ending fps. Garbage collection.

4. **Steps to be executed**
   4.1 Build a new version of the game, changing only the way of bullet spawning, depending on the index of this test (0 = instantiating, 1= bullet pooling).
   4.2 Open the independent profiler
   4.3 Run the application
   4.4 Application terminates automatically after Time.
   4.5 Save the profiler data. Possibly screenshot relevant information for this document.

5. **Expected result**
   Test 0: Slight decrease in fps over time. High garbage collection amount.
   Test 1: No decrease in fps. No/barely any garbage collection.

6. **Actual result and post-conditions**
   TBD

7. **Pass/fail (implement or not implement this specific optimization?)**
   fdfd

Async scene-loading vs standard 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 in 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)

1. **Test description**
   Perform 2 tests. One with standard scene loading, and one with async scene-loading. Scenes to transition from and to will always be menu > level.

2. **Assumptions and pre-conditions**
   Allow easy switches between scene-loading techniques with a simple bool in the SceneLoader script. For an accurate testing-ground, the new scene will be triggered after 10 seconds since startup for both tests. Player input will not be required.

3. **Test variables**
   The time it takes for the Level scene to be fully loaded (= able to move around).

4. **Steps to be executed**
   4.1 Build a new version of the game, changing only the state of static batching, depending on the index of this test (0 = no static batching, 1= static batching).
   4.2 Open the independent profiler
   4.3 Run the application
   4.4 Application terminates automatically after Time.
   4.5 Save the profiler data. Possibly screenshot relevant information for this document.

5. **Expected result**
   Draw calls will have been reduced drastically. Fps improved.
   Test 0 (standard loading): lag spike after triggering the scene loading. Takes a while to transfer to the Level scene.
   Test 1 (async loading): seamless scene transition. Scene will be done loading by the time it gets triggered, resulting in a smooth transition.

6. **Actual result and post-conditions**
   TBD

7. **Pass/fail (implement or not implement this specific optimization?)**
   fdfd

Static Batching

“Static batching is a draw call batching method that combines meshes that don’t move to reduce draw calls. 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 because static batching doesn’t transform vertices on the CPU.” (Unity, 2022)

1. **Test description**
   Perform 2 tests. One with no static batching at all, and one where the whole environment (static objects) are included in static batching.
2. **Assumptions and pre-conditions**
   A script is attached to the player (so including the camera). It moves the player forward at 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. Player input will not be required.

3. **Test variables**
   The performance of the game (average fps, lag spikes), draw calls.

4. **Steps to be executed**
   4.1 Build a new version of the game, changing only the state of static batching, depending on the index of this test (0 = no static batching, 1= static batching).
   4.2 Open the independent profiler
   4.3 Run the application
   4.4 Application terminates automatically after Time.
   4.5 Save the profiler data. Possibly screenshot relevant information for this document.

5. **Expected result**
   Draw calls will have been reduced drastically. Fps improved.
   Test 0 (no static batching): lag spikes, many drawcalls, lower overall fps.
   Test 1 (static batching): less/lower intensity lagspikes, way less drawcalls, better overall fps.

6. **Actual result and post-conditions**
   TBD

7. **Pass/fail (implement or not implement this specific optimization?)**
   fdfd

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)

1. **Test description**
   Perform 2 tests. One with the quality level set to Medium, and one with the quality level set to Ultra.

2. **Assumptions and pre-conditions**

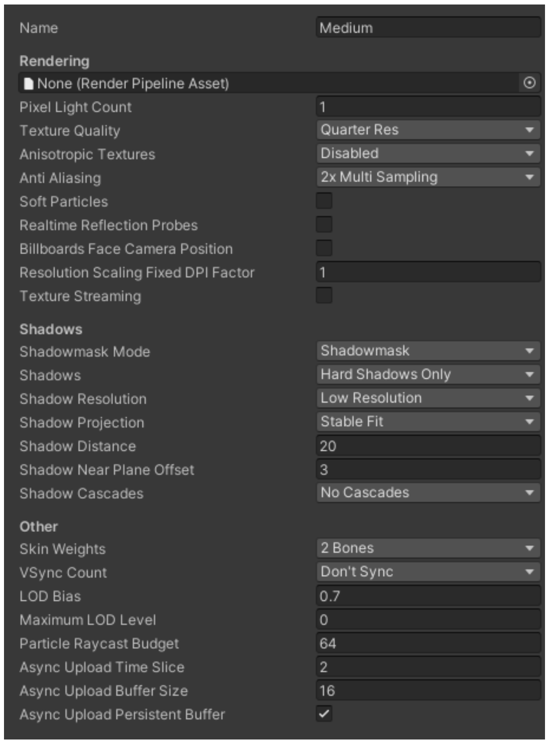
   A script is attached to the player (so including the camera). It moves the player forward at a constant speed, for a constant amount of time.

   Speed = 1 unit/s

   Time = 10 seconds

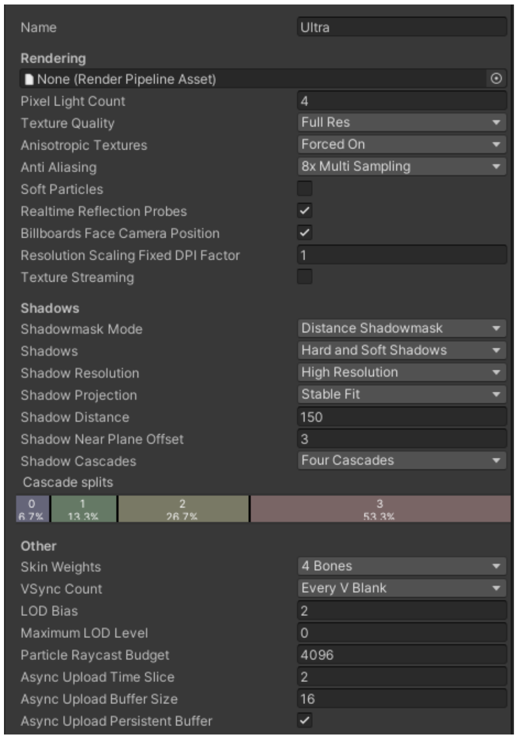
   As with all tests, it's run in a build. Player input will not be required.

Medium quality level settings:



Name	Medium
Rendering	
None (Render Pipeline Asset)	
Pixel Light Count	1
Texture Quality	Quarter Res
Anisotropic Textures	Disabled
Anti Aliasing	2x Multi Sampling
Soft Particles	☐
Realtime Reflection Probes	☐
Billboards Face Camera Position	☐
Resolution Scaling Fixed DPI Factor	1
Texture Streaming	☐
Shadows	
Shadowmask Mode	Shadowmask
Shadows	Hard Shadows Only
Shadow Resolution	Low Resolution
Shadow Projection	Stable Fit
Shadow Distance	20
Shadow Near Plane Offset	3
Shadow Cascades	No Cascades
Other	
Skin Weights	2 Bones
VSync Count	Don't Sync
LOD Bias	0.7
Maximum LOD Level	0
Particle Raycast Budget	64
Async Upload Time Slice	2
Async Upload Buffer Size	16
Async Upload Persistent Buffer	✓

Ultra quality level settings:



Name	Ultra
Rendering	
None (Render Pipeline Asset)	
Pixel Light Count	4
Texture Quality	Full Res
Anisotropic Textures	Forced On
Anti Aliasing	8x Multi Sampling
Soft Particles	☐
Realtime Reflection Probes	✓
Billboards Face Camera Position	✓
Resolution Scaling Fixed DPI Factor	1
Texture Streaming	☐
Shadows	
Shadowmask Mode	Distance Shadowmask
Shadows	Hard and Soft Shadows
Shadow Resolution	High Resolution
Shadow Projection	Stable Fit
Shadow Distance	150
Shadow Near Plane Offset	3
Shadow Cascades	Four Cascades
Cascade splits	0: 6.7%, 1: 13.3%, 2: 26.7%, 3: 53.3%
Other	
Skin Weights	4 Bones
VSync Count	Every V Blank
LOD Bias	2
Maximum LOD Level	0
Particle Raycast Budget	4096
Async Upload Time Slice	2
Async Upload Buffer Size	16
Async Upload Persistent Buffer	✓

3. **Test variables**
   The performance of the game (average fps, lag spikes), draw calls.

4. **Steps to be executed**
   - 4.1 Build a new version of the game, changing only the level of quality, depending on the index of this test (0 = quality level Medium, 1 = quality level Ultra).
   - 4.2 Open the independent profiler
   - 4.3 Run the application
   - 4.4 Application terminates automatically after Time.
   - 4.5 Save the profiler data. Possibly screenshot relevant information for this document.

5. **Expected result**
   Draw calls will have been reduced drastically. Fps improved.
   Test 0 (Medium level) :
   Test 1 (Ultra level):

6. **Actual result and post-conditions**
   TBD

7. **Pass/fail (implement or not implement this specific optimization?)**
   dsds

Mixed Lighting vs Realtime Lighting

Mixed:

"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)

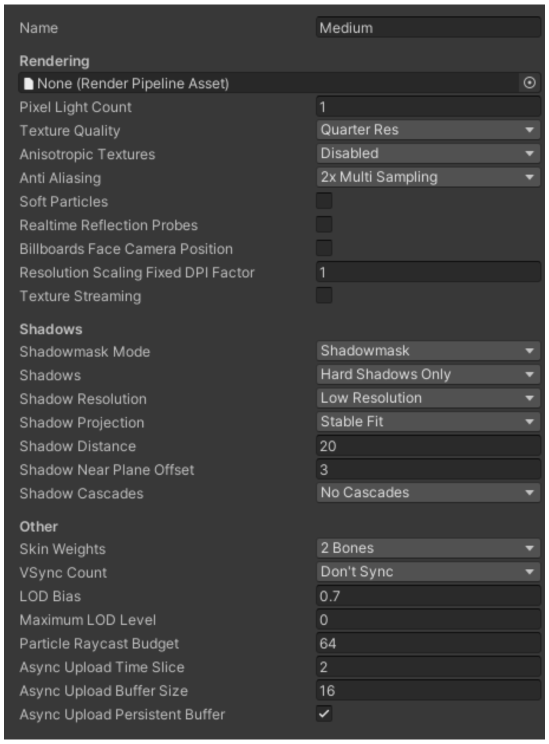
Realtime:

"Unity calculates and updates the lighting of these Lights every frame at run time. They can change in response to actions taken by the player, or events which take place in the Scene. For example, you can set them to switch on and off (like a flickering light), change their Transforms (like a torch being carried through a dark room), or change their visual properties, like their colour and intensity. Real-time Lights illuminate and cast realistic shadows on both static and dynamic GameObjects." (Unity, 2017)

1. **Test description**
   Perform 2 tests. One with all present lights set to Realtime, and one with all present lights set to Mixed.

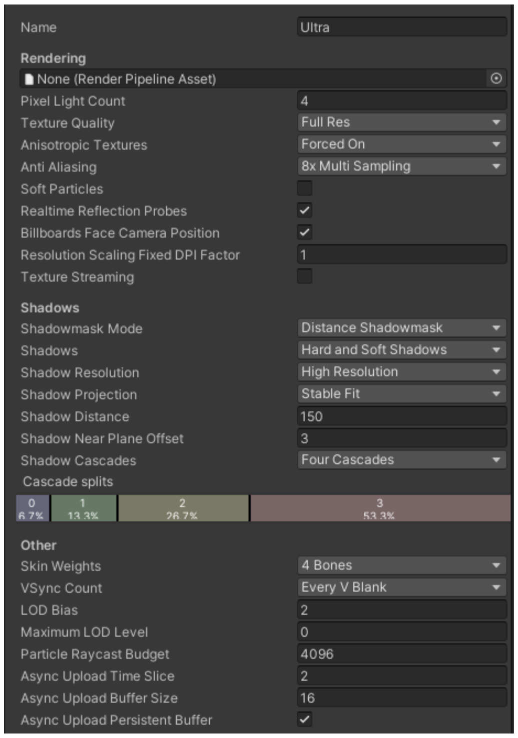
2. **Assumptions and pre-conditions**
   A script is attached to the player (so including the camera). It moves the player forward at a constant speed, for a constant amount of time.
   Speed = 1 unit/s
   Time = 10 seconds
   As with all tests, it's run in a build. Player input will not be required.

Medium quality level settings:



Name	Medium
Rendering	
None (Render Pipeline Asset)	
Pixel Light Count	1
Texture Quality	Quarter Res
Anisotropic Textures	Disabled
Anti Aliasing	2x Multi Sampling
Soft Particles	☐
Realtime Reflection Probes	☐
Billboards Face Camera Position	☐
Resolution Scaling Fixed DPI Factor	1
Texture Streaming	☐
Shadows	
Shadowmask Mode	Shadowmask
Shadows	Hard Shadows Only
Shadow Resolution	Low Resolution
Shadow Projection	Stable Fit
Shadow Distance	20
Shadow Near Plane Offset	3
Shadow Cascades	No Cascades
Other	
Skin Weights	2 Bones
VSync Count	Don't Sync
LOD Bias	0.7
Maximum LOD Level	0
Particle Raycast Budget	64
Async Upload Time Slice	2
Async Upload Buffer Size	16
Async Upload Persistent Buffer	✓

Ultra quality level settings:



Name	Ultra
Rendering	
None (Render Pipeline Asset)	
Pixel Light Count	4
Texture Quality	Full Res
Anisotropic Textures	Forced On
Anti Aliasing	8x Multi Sampling
Soft Particles	☐
Realtime Reflection Probes	✓
Billboards Face Camera Position	✓
Resolution Scaling Fixed DPI Factor	1
Texture Streaming	☐
Shadows	
Shadowmask Mode	Distance Shadowmask
Shadows	Hard and Soft Shadows
Shadow Resolution	High Resolution
Shadow Projection	Stable Fit
Shadow Distance	150
Shadow Near Plane Offset	3
Shadow Cascades	Four Cascades
Cascade splits	0: 6.7% / 1: 13.3% / 2: 26.7% / 3: 53.3%
Other	
Skin Weights	4 Bones
VSync Count	Every V Blank
LOD Bias	2
Maximum LOD Level	0
Particle Raycast Budget	4096
Async Upload Time Slice	2
Async Upload Buffer Size	16
Async Upload Persistent Buffer	✓

3. Test variables

The performance of the game (average fps, lag spikes), draw calls.

4. Steps to be executed

4.1 Build a new version of the game, changing only the level of quality, depending on the index of this test (0 = quality level Medium, 1 = quality level Ultra).
4.2 Open the independent profiler
4.3 Run the application
4.4 Application terminates automatically after Time.
4.5 Save the profiler data. Possibly screenshot relevant information for this document.

5. Expected result

Draw calls will have been reduced drastically. Fps improved.
Test 0 (Realtime):
Test 1 (Mixed):

6. Actual result and post-conditions

TBD

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

dsds

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. (2017, 6 8). *Real-time lighting*. Retrieved from docs.unity3d.com: https://docs.unity3d.com/560/Documentation/Manual/LightMode-Realtime.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