

# PERFORMANCE OPTIMIZATION CHECKLIST

UNITY 2021+



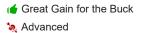
## Index

R		_	4	_	
6	U	a	ι	e	

### Project:

Introduction	3
Unity CPU Optimization	4
∇ Rendering	5
$\triangledown$ User Interfaces	5
▽ Scripting	6
∇ Animation	6
∇ General	7
∇ Physics	7
Unity GPU Optimization	8
∇ General	9
▽ Overdraw	9
∇ Vertex Processing	10
∇ Shaders	10
∇ VR	10
Unity Memory Optimization	11
$\triangledown$ Loading Times and Memory Usage	12
∇ Build Size	12
Bonus Section	13
Performance Trade-Offs	14
How to Get 140 More Performance Tips	15





### **Introduction**

Hey there,

I'm Ruben from The Gamedev Guru.

This is my opportunity to *thank you* for downloading this Performance Checklist for Unity 2021+

This document is the result of *years of experience* shipping games to tenths of millions of players all throughout the world.

As you know, game development is a complicated business.

That means: take the recommendations that help you and leave those that don't.

No amount of expertise will ever replace profiling your own game.

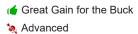
I hope this checklist serves you as well as it did serve me.

Ruben Torres Bonet (The Gamedev Guru)

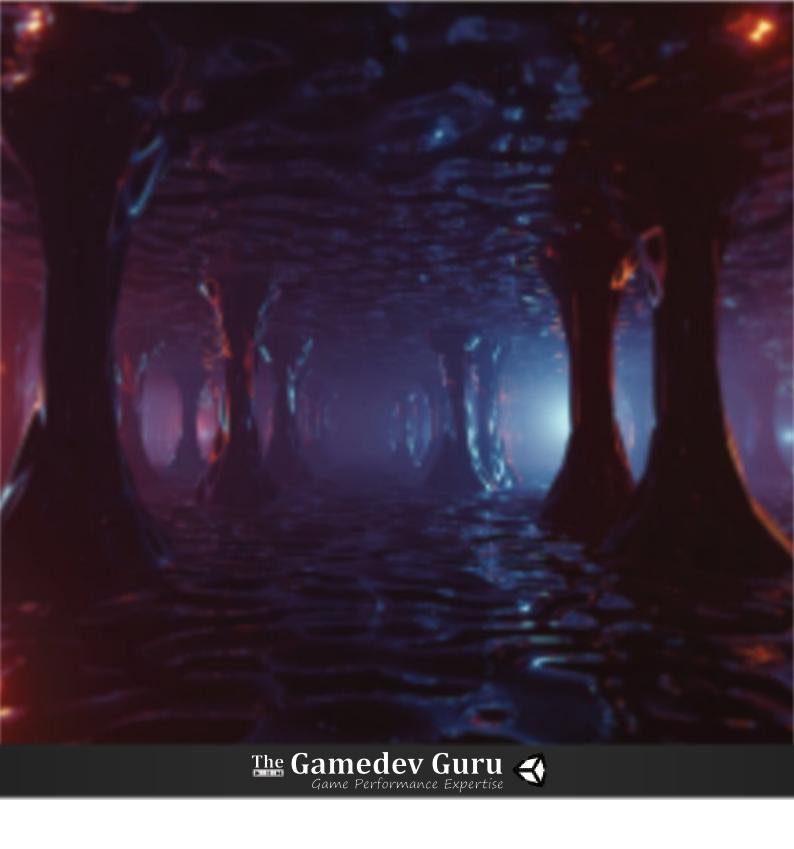
Questions?

Send me an e-mail at <a href="mailto:ruben@thegamedev.guru">ruben@thegamedev.guru</a>





<sup>\*</sup> Some of the assets I recommend here contain affiliate links. Rest assured, I recommend them because they're great and I use them continuously.



# UNITY CPU OPTIMIZATION



## 1. CPU Optimization

#### ∇ Rendering

- □ Use <u>GPU instancing</u> only for dynamic objects that aren't transformed often
- □ Aim for < 100 draw calls on mobile,
- < 1000 draw calls on desktop
- < □ Disable depth prepass on mobile
  </p>
  - □ Restrict cameras' culling mask to the strictly required layers
  - □ Use <u>occlusion culling</u> in interiors

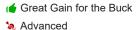
- - □ All your particle systems are procedural
  - □ Use <u>MaterialPropertyBlock</u> instead of instancing materials
  - $\hfill\Box$  Disable precomputed real-time GI
- □ Don't use (realtime) reflection probes

#### ∇ User Interfaces

- Change materials' <u>color property</u> instead of multiple sprites with color variations
- Do not use <u>auto-layout components</u> on dynamic UI: content size fitter, layout element, horizontal/vertical/grid layour group
  - ☐ If you use auto-layout components, disable them once they've done their work
  - ☐ For tables, consider <u>TSTableView</u>

- □ Avoid per-frame changes in UI components to reduce canvas rebuild events: RectTransform, colors, sprites, text and other properties
- Learn UIBuilder and <u>UIElements Runtime</u> for upcoming Unity releases
  - ☐ Use TheGamedevGuru *canvas rebuild detector* to find and resolve canvas rebuilds



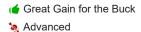


### 1. CPU Optimization

#### abla Scripting ☐ Do not target Mono but *il2cpp* in <u>master</u> ☐ Write C# *jobs+burst* for slow operations (>0.2ms) on multiple elements (>4) mode ☐ Avoid using the Update function; prefer ♠ ☐ Implement <a href="DOTS">DOTS</a> for massive amount of using TheGamedevGuru's BatchUpdate homogeneous elements ☐ Use <u>structs</u> instead of classes for short- $\square$ < 32 bytes of per-frame *allocations* time data storage CPU cost ☐ Disable *script debugging* ☐ Use *String.Empty* instead of "" ☐ Don't use *Instantiate* during gameplay; prefer during loading screens for pooling abla Animation ☐ Enable *Optimize Game Objects* in the <□ □ Reduce bone count for skinning to</li> rigging import settings of your characters maximum 2 on mobile



animators



☐ Reduce *blend tree complexity* in

animations; aim for < 6 blend nodes

avoid them at all costs in UI

☐ Use animators exclusively for characters;

☐ Prefer *tweening* and custom scripts over

☐ Aim to reduce *triangle count* on animated

spritesheet animations for distant characters

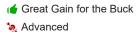
characters to < 3k on mobile

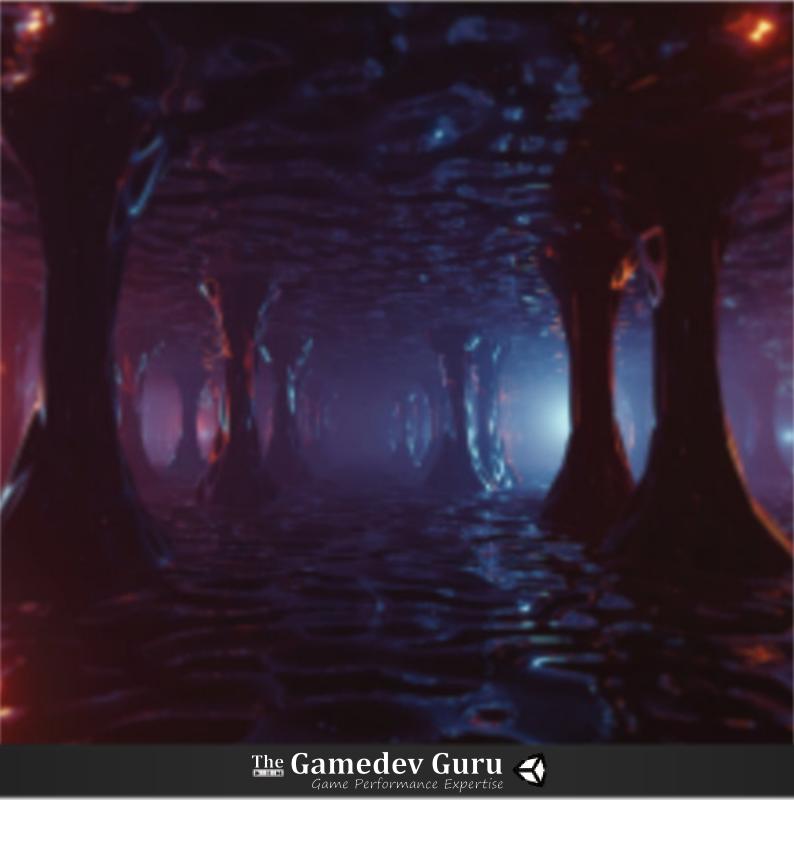
Consider impostor rendering and

# 1. CPU Optimization

∇ General	
☐ Keep <u>scene hierarchies</u> shallow (< 5)	☐ Keep <i>packages</i> up to date, e.g. addressables, TextMesh Pro
☐ Keep <u>scene hierarchies</u> narrow (< 50)	☐ Strip <u>unused shaders</u>
☐ Set <i>static flags</i> on static elements	
☐ Use <u>CullingGroups</u> to pause out-of-screen subsystems	☐ Automate measuring your performance continuously
Subsystems	☐ Short <i>SFX</i> : enable <u>decompress on load</u>
☐ Use the P3 Optimization Framework	
<b>∇</b> Physics	
☐ Disable <u>auto-sync transforms</u>	☐ Adapt the budget allocated to physics on <a href="mailto:Time settings">Time settings</a>
☐ Enable re-use <u>collision callbacks</u>	☐ Try <u>multibox pruning broadphase</u>
☑ Avoid <u>mesh colliders</u> , use instead	
compound colliders of simple shapes	☐ Add more layers and minimize the <u>layer</u> <u>collision matrix</u> enabled pairs
☐ If using mesh colliders: enable <u>read/write</u>	
<u>flag</u> on its mesh importer	







# UNITY GPU OPTIMIZATION



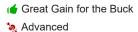
# 2. GPU Optimization

∇ Gener
---------

	☐ Use 0 (or max. 1) real-time <u>per-pixel lights</u> on mobile	D.	☐ Avoid <i>multi-camera</i> setups
	☐ Use 4x <i>MSAA</i> on (most) mobile platforms		☐ Avoid real-time generated <i>render textures</i>
	☐ Full-screen <i>post-processing</i> forbidden on mobile		☐ Optimize texture space usage with color palettes and efficient UVs
	☐ <u>Clear</u> only with pure black color		☐ Improve GPU caching behaviour by using unique big <i>texture atlases</i> with tools such as Mesh Baker
I	☐ Avoid <i>real-time shadows</i> on mobile		
	Overdraw		
	☐ Avoid <i>stacking</i> over 2 layers of UI on top of each other		☐ Create <i>tighter meshes</i> for sprite renderers with the <u>sprite editor</u>
	☐ Use the <u>frame debugger</u> to check if you truly render opaque geometry in <i>front-to-back</i>	***	☐ Favor Sprite Renderers over UI Images
ord	order		☐ Make overdraw less expensive with additive blending instead of alpha blend
	☐ Be willing to disable <u>batching</u> if it breaks optimal object sorting		☐ Reduce <i>particle system</i> particles count
	☐ Remove geometry <u>intersections</u> and		☐ Reduce <i>particle system</i> particle sizes



overlaps in your 3d meshes

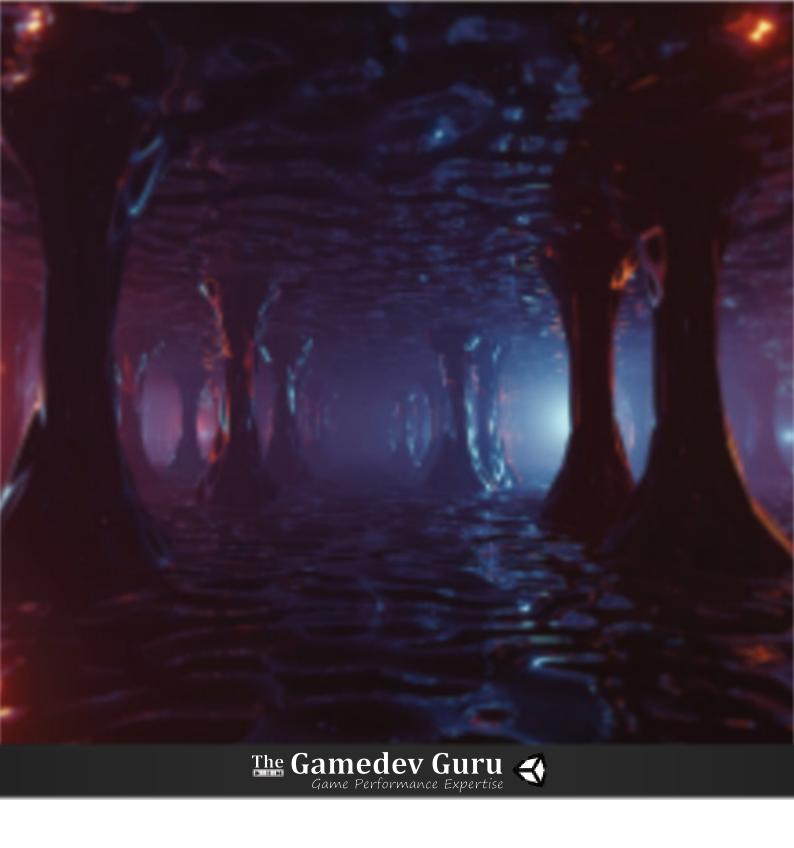


# 2. GPU Optimization

### ∇ Vertex Processing

<b>(6</b> [	☐ Use <u>LOD Groups</u>		☐ Avoid <i>flat-shading</i> (it multiplies vertices)
	☐ Use <u>CullingGroup</u> to disable renderers		☐ Use <u>GPU instancing</u> on dynamic elements to make vertex processing cheaper
	☐ Use <u>Occlusion Culling</u> in interiors to avoid rendering invisible elements	<	☐ Reduce vertices while maintaining quality by using <i>normal</i> and <i>displacement maps</i>
<u>s</u>	☐ If programmer: use tools such as simplygon to automagically reduce vertex count		□ <200k vertex/polygons on mobile
$\bigvee$	Shaders		
	☐ Consider <i>baking</i> lighting information on he diffuse texture itself for static elements		☐ Standard shaders only on high-end HW
V	vithout using normal maps		☐ Use <i>deferred-rendering</i> only on high-end desktop hardware
	☐ Avoid <i>conditional branches</i>		
	☐ Use the smallest <i>variable <u>precision</u></i> you need, e.g. half over float		☐ Measure your <i>shader complexity</i> with tools like <i>Mali offline shader compiler</i>
	□ Avoid <i>multi-pass</i> shaders		☐ Don't use <u>GrabPass</u> on mobile
	☐ Avoid sampling from <i>reflection probes</i>		☐ <u>Pre-compile</u> shaders to avoid hiccups
$\bigvee$	VR		
	☐ Use <u>foveated rendering</u> to reduce ragment shading and ROP complexity		☐ Avoid <i>standard shaders</i> on mobile VR
<b>(6</b> [	☐ Set Oculus <u>GPU <i>hardware levels</i></u> to 4		☐ Consider <i>baking</i> lighting information on the diffuse texture itself for static elements without using normal maps
[ e	☐ Reduce <u>eye/render resolution</u> on emergencies		☐ Use <i>RenderDoc</i> extensively





UNITY MEMORY OPTIMIZATION

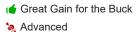


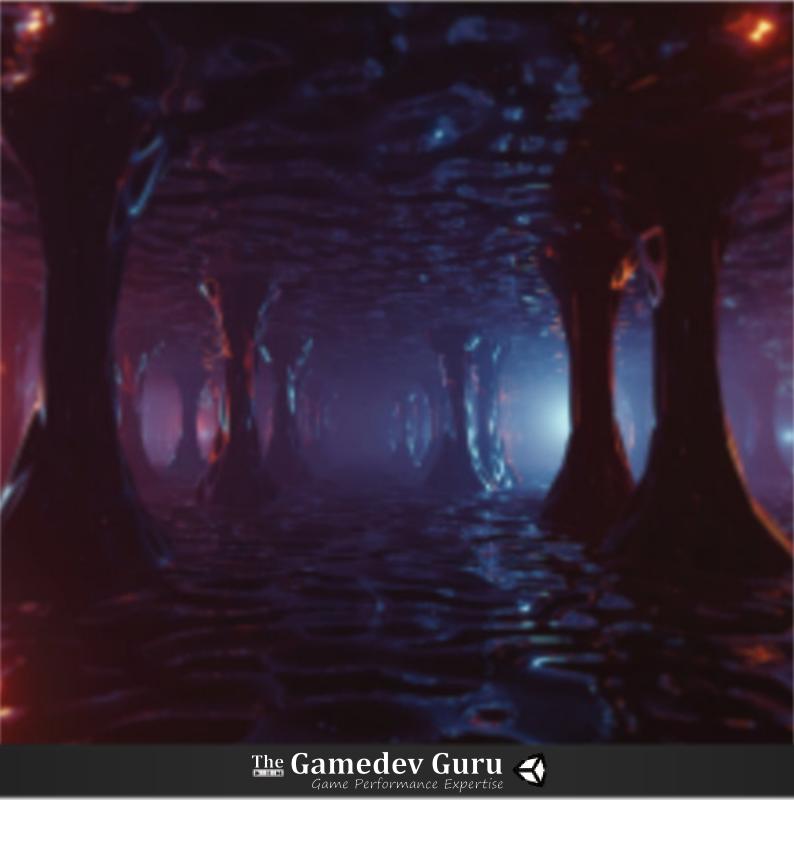
## 3. Memory Optimization

#### ∇ Loading Times and Memory Usage

•	☐ Do NOT use the <i>Resources directory</i> ; Migrate to Addressables if you do	intended optimized formats with tools such a A+ Asset Explorer
	☐ Use an almost-empty animated initial scene for loading screens while you load the next scene asynchronously	● □ Atlas your <i>textures</i> and reduce their sizes Tools like <u>Mesh Baker</u> will help you
	☐ Make an exhaustive inventory <i>of shaders</i> and use a small set of unique shaders,	☐ Be selective about AudioClip import settings: do not overuse <i>Decompress on Load</i>
	stripping out the rest. Use tools such as A+ Asset Explorer to confirm.	☐ Try <i>Load in background</i> and <i>streaming</i> for long AudioClips
	☐ Pre-compile the few shaders you use	☐ Learn the <u>Addressables system</u>
	$\square$ Confirm <u>all</u> your assets are using the	
	Build Size	
	☐ Measure the output build size components with <i>build analyzers</i> such as <u>Build Report</u> <u>Tool</u>	☐ Detect and remove <i>duplicated content</i> wit tools such as <u>A+ Asset Explorer</u>
•	☐ Delegate content to <u>CDNs</u> and download in run-time using Addressables	☐ Disable <i>mipmaps</i> where possible to gain 33%
	☐ Use <u>LZ4</u> for general fast compression with addressable asset bundles	<ul> <li>Consider <i>Unity Tiny</i> for ultra-small builds</li> <li>■ Merge materials and textures with tools</li> </ul>
	☐ Use <u>LZMA</u> for maximum (but slow) compression with addressable asset bundles	like <u>Mesh Baker</u>



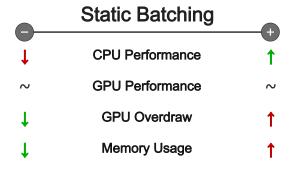




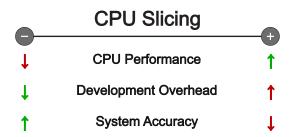
# BONUS SECTION

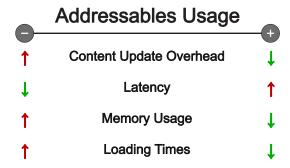


## Performance Trade-Offs

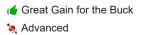


SpriteRenderer vs Image	
CPU Performance	<b>1</b>
GPU Performance	1
Ease of Use for UI	1
Supports Tight Meshes	X
	CPU Performance GPU Performance Ease of Use for UI









### Next Steps

Dear Game Developer,

If you liked my *Unity Performance Checklist - Lite Edition*, you can find out more about my other game performance level-up programs:

- Game Performance Taskforce: community-driven weekly lessons on game performance, including live q&a and live lessons
- Unity Performance Checklist PRO (2x the size & features)
- The P3 Optimization Framework: know your next steps in optimization
- Addressables for the Busy Developer: nullify loading times and excessive memory usage. Deliver your content over CDNs for 10x faster iteration times.

Ruben (The Gamedev Guru)









