

# Core Performance Tricks & Common Mistakes

# 3 Types of Client Performance Problems

## CPU/GPU in Profiler

## Issue Type

## Possible Reasons & Examples

CPU - Game Thread **60ms**

CPU - Render Thread **18ms**

GPU **60ms**

### *Game Thread Bound*

- **Too much going on with gameplay!**
- LUA scripting issues, especially with player scaling
- Too many animated meshes
- Many attached objects on player with collision
- Various types of custom animation, ui, or physics

CPU - Game Thread **12ms**

CPU - Render Thread **60ms**

GPU **60ms**

### *Render Thread Bound*

- **Too many objects, or too process rendering!**
- Extremely high mesh counts
- Too many overlapping shadows & lights (esp. terrain)
- Overly complex UI in the wrong context
- Missing distance culling, or showing too much of map

CPU - Game Thread **14ms**

CPU - Render Thread **18ms**

GPU **60ms**

### *GPU Bound*

*(Less common, harder to profile)*

- **Graphics card not powerful enough!**
- High particle density and particle size on VFX
- Overlapping decals or overlapping lights
- High settings on some assets
- Player graphics settings too high\*

CPU - Game Thread **16ms**

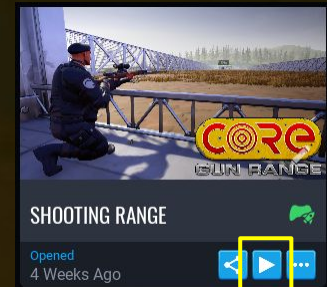
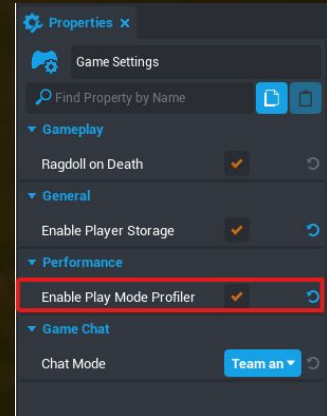
CPU - Render Thread **16ms**

GPU **16ms**

### *Ideal*

- **Will run at 60 FPS, good job!**
- Still can be networking problems creating rubber banding
- Also check the server for issues

# Quick Performance Overview - Profiler (1)



- **Toggle open with F4** once enabled in the **Game Settings Object**. Note it only opens in **local preview** or on the **live server** for a published game.
- Captures information about the CPU & GPU threads contributing to Frame Rate, both for **client** and **server**.
- Very useful for identifying the broad source of performance problems, and especially to identify what the limiting factor is for your game performance - **Game Thread CPU, Render Thread CPU, GPU, or Networking**.
- Target **16.3ms** Frame Time to reach **~60 FPS** for your players.

## Play Mode Profiler Documentation:

<https://manticoregames.atlassian.net/wiki/spaces/CORE/pages/501121025/Play-Mode+Profiler>

## Quick Performance Overview - Hierarchy View (2)

Toggle different stats showed inline in the hierarchy

- Game Thread time
- Render Thread time
- Triangles
- Draw calls
- Load time

Scoreboard snipers	0.002 ms
Chat	0.000 ms
UI Container	0.000 ms
NetworkTransfer (networked)	0.000 ms
Bigglebuns	0.970 ms
Helper_Nameplate	0.000 ms
MRCi 88 (networked)	0.133 ms
KZ99 (networked)	0.039 ms
Hunting Knife (networked)	0.014 ms
Equipment (ni_8ABA1D45C89EEC3E:Hunting Knife)	0.002 ms
Equipment (networked)	0.004 ms
Equipment (networked)	0.000 ms
ScreenObjectGroup	0.000 ms
ScreenObjectGroup	0.000 ms
ScreenObjectGroup	0.000 ms
Generic Reticle (client)	0.081 ms

### Game Thread Times

- Multiple stats about hierarchy objects can be viewed either in preview mode or just at edit time.
  - Game Thread Time** - very helpful for catching pesky LUA scripts doing something inefficiently
  - Render Thread Time** - helpful to identify expensive rendering processes or see expensive parts of the map. Note this is tied to the camera view.
  - Triangles** - can be used at edit time, helpful to identify expensive meshes
  - Draw Calls** - tied to material usage, another tool to identify expensive art assets
  - Load Time** - Hit play to see the load time from various hierarchy elements
- Can view objects in the player in preview mode - this can be extremely useful as well!

Terrain	2.473 ms
Terrain_bg	0.070 s
Game Settings	6.884 ms
UI Settings	0.129 s
Knar	0.179 ms
Keppu(Antti)	2.842 s
Kurtis	0.049 s
Blake	0.059 s
Patrick	0.028 s
Brent	0.054 s
Konz	0.021 s
Gabriel	0.019 s
Bigglebuns	6.577 ms
Divided	0.350 ms
Roland	0.310 ms
Scoreboard snipers	3.746 ms
Chat	0.297 ms
UI Container	0.695 ms
NetworkTransfer (networked)	0.158 ms
Bigglebuns	0.000 ms
Generic Reticle (client)	0.484 ms

### Load Times

# Quick Performance Overview - Server Logs (3)

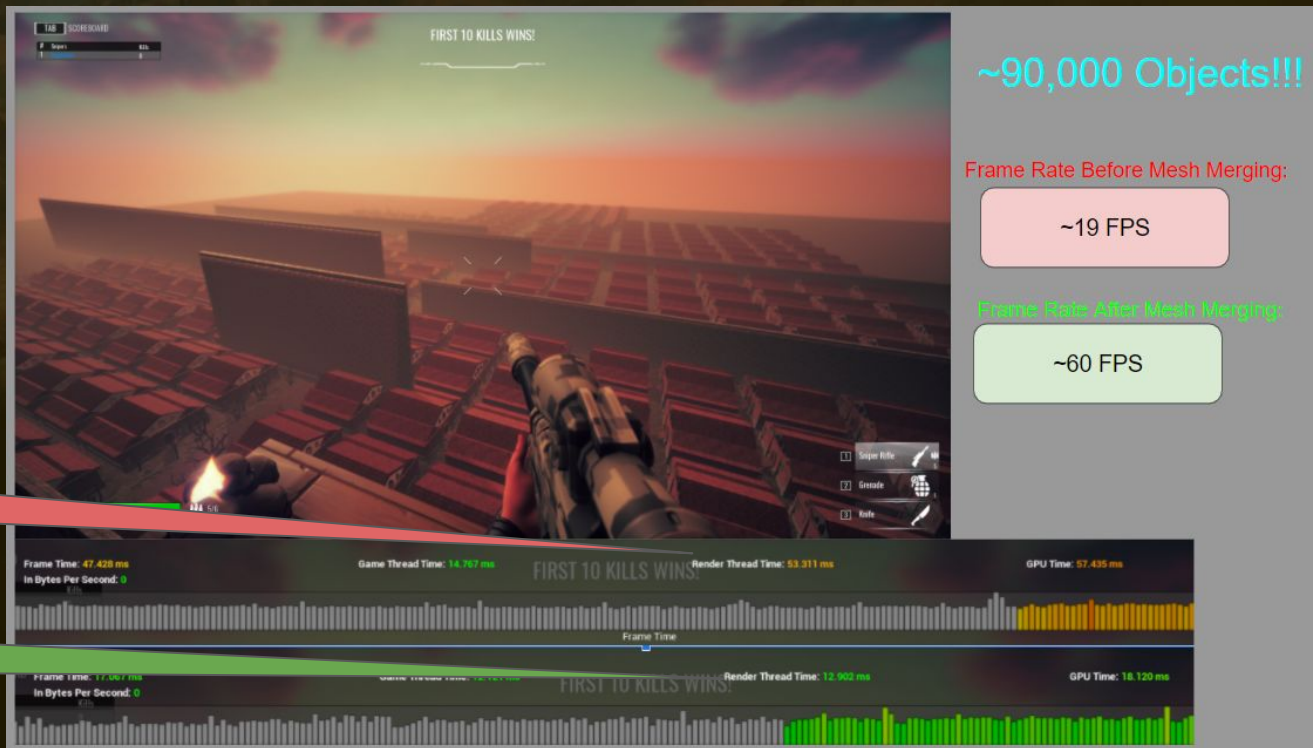
- Access Server Logs if you are the publisher of a game
  - Publish unlisted to test on the live server.
- Print error messages to the server logs for help debugging



# 18 Common Mistakes

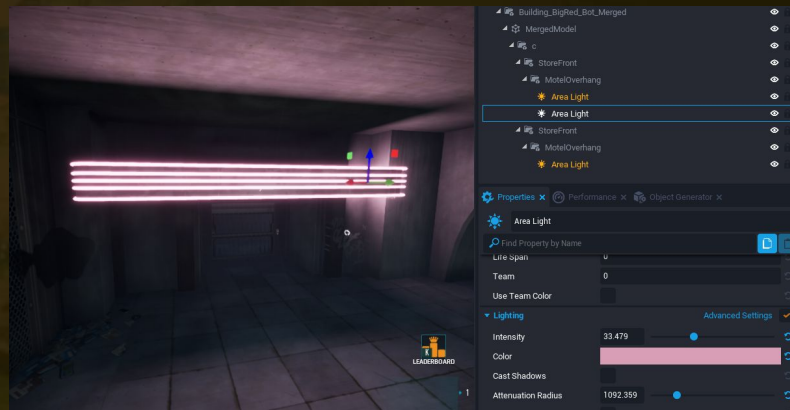
## #18 - Not Mesh Merging!

- Merging meshes will dramatically improve the frame rate for large, object heavy maps especially when many objects need to render simultaneously! See the Render Thread improvements below.



## #17 - Too Many Shadows & Area Lights

- Shadows can be very expensive for GPU and CPU Render times - they are one of the easiest things to change to improve performance!
- Shadow casting lights are especially expensive
- Especially, watch out for shadow casting area lights overlapping with lots of other lights!



### Instance Properties

### Advanced Settings

Cull Distance	Min 100.0 m	Max 200.0 m
Cast Shadows	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Affect Distance Field Lighting	<input type="checkbox"/>	<input type="checkbox"/>
Cast Shadows as Two Sided	<input type="checkbox"/>	<input type="checkbox"/>
Receives Decals	<input checked="" type="checkbox"/>	<input type="checkbox"/>



## #16 - Animating Collidable Objects

- High CPU cost associated with animating objects which still use collision (6x more!)
  - Note we changed client context folders to default with collision off as a way to mitigate this particular issue
- Recommendation - simply make sure anything which is animating has collision off!



house_Windmill_collisionON	0.392 ms	
windmill_geo	0.000 ms	
windmill_base	0.000 ms	
blades	0.392 ms	
house_Windmill_collisionOFF	0.070 ms	
windmill_geo	0.000 ms	
windmill_base	0.000 ms	
blades	0.070 ms	

82% Savings!!

### Total Counts for [ blades ]

Networked: 0 | Non-Networked: 83 | Total: 83

### Child [ Windmill\_center\_cog ]

Networked: 0 | Non - Networked : 82 | Total : 82

### Child [ Object Rotator Continuous ]

Networked: 0 | Non - Networked : 1 | Total : 1

Collision

Force Off

## #15 - Large Moving Triggers or Trigger Based Volumes

- Extremely high CPU cost associated with moving large triggers over areas which have “Interacts with Triggers” checked to be true
  - Note: we changed this setting to default OFF in order to mitigate the issue

The image shows a game in Preview Mode. The main view is a 3D environment with a large, orange, semi-transparent trigger volume. A text overlay in the center reads "Zone Closes in 4 seconds". The top of the screen shows "PREVIEW MODE", "Zone Center 300m", "KILLS 0", "ALIVE 1", and "PHASE 4 - 00:04". On the right side, there is a performance log window titled "Game Logic" with a search bar and several icons. The log shows a list of events and their execution times. A yellow box highlights the "Next Event" entry, which is "Script: ConstrictingPlayZoneClient (1): 8.306ms". Below it, other events are listed with their respective times, such as "Script: RoundStartSkydiveServer (1): 0.148ms" and "Script: DamageDismountsInZone (1): 0.132ms".

Interact with Triggers

## #14 - Complex Player Attachments!

- Objects attached to the player are especially expensive - use these sparingly!
- If these objects have collision on them, the cost will increase CPU game thread costs dramatically - make sure collision (& camera collision) are turned off.



*Early Hunter Costume from upcoming game*

## #13 - Single Player Preview vs. Live Server

- **Single Player Preview** - Client and server are the same princess and run at the same display framerate. There is no networking happening, you know everything immediately.
- **Multiplayer Preview** - Editor is the server and approximates a dedicated server, but only approximates. Still runs at display framerate. Clients won't have lag or packet loss. This gets most of the way to a live server but is not 100% accurate and can still hide problems.
- **Live Server** - In the published game, server is running at 30hz and clients will have packet loss. Order and duration of script based processes can be potentially problematic. Very common problem is trying to do something to a player (i.e. equip something) before other scripts have loaded.

### Single Player Preview



### Multiplayer Preview



### Live Server



#### Strike Team



PLAY

Published	Updated	Max Players	Total Posts
3/4/21	3/25/21	12	55697

2 x XP and Cash continues! Don't miss out on free skins by winning with the devs (if a dev is on the server and you win, you get their favorite skin)

In this fast-paced FPS, two teams compete for control of Strike Points  
[Read More...](#)

Action KingOfTheHill Military

Multiplayer Competitive

Share

768 130

## #12 Outline Object & Callouts

- Depending on the complexity kitbash object, the “Outline Object” asset can be highly CPU intensive, especially if you’re duplicating the asset many times throughout the level!
- Other approaches can be equally effective at highlighting an object at a fraction of the cost, for example “Callout VFX”

Outline Object	0.005 ms	👁
Callout Sparkle	0.000 ms	👁

Approach A:



GatherNode_Sunflower	0.036 ms	👁
Item_SpawnPoint (client)	0.000 ms	👁
GatherTrigger (client)	0.000 ms	👁
ItemModel (client)	0.000 ms	👁
MergedModel (client)	0.000 ms	👁
Outline (client)	0.000 ms	👁
Outline2 (client)	0.000 ms	👁
SparkleVFX (client)	0.036 ms	👁
GatherNode_Sunflower	0.000 ms	👁

Approach B:

ItemModel (client)	0.000 ms	👁
--------------------	----------	---

Total Counts for [ItemModel]  
 Networked: 0 | Non-Networked: 63 | Total: 63

Child [MergedModel]  
 Networked: 0 | Non - Networked : 63 | Total : 63



GatherNode_Sunflower	0.685 ms	👁
Item_SpawnPoint (client)	0.000 ms	👁
GatherTrigger (client)	0.000 ms	👁
ItemModel (client)	0.000 ms	👁
MergedModel (client)	0.000 ms	👁
Outline3 (client)	0.000 ms	👁
Outline (client)	0.630 ms	👁
Outline2 (client)	0.000 ms	👁

**18x cost!**

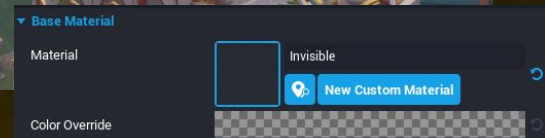
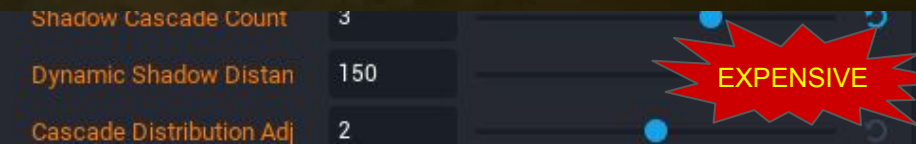
## #11 - Accumulating Listeners

- Stay mindful of “Listener” in the performance profiler - these can lead to large gameplay spikes as seen to the right.
- Scripts destroyed that are connected to equipment will stay in memory until the equipment is destroyed.
- Caching objects from server / client UserData like (x = object.serverUserData) that isn't cleared out will stop Objects from being destroyed.



## #10 - Lighting with Dynamic Shadows

- Avoid increasing Shadow Cascade Count (distance shadows) to a high number as a quick fix for terrain shadow casting.
- Instead, use invisible objects to create distance shadows
  - Visibility on, invisible material



## #9 - Too Many Decals & No Draw Distances!

- Too many decals in an environment will drive up GPU costs significantly - in the case of Strike Team the map uses ~900 decals for the gritty urban atmosphere.
- Profiling the same location with and without any decals shows ~5ms delta to the GPU time.
- **Recommendation:** utilize the draw distance setting to turn off decals which wouldn't be relevant given the player location (i.e. across the map)

### Performance - Decal

Low Distance	4500
Medium Distance	7000
High Distance	9000

### Performance - Vfx

Low Distance	1000
Medium Distance	2500

### Rendering

Relevance	Critical
Smart	Low
Advanced	Medium
Volume Display Color	High
	Critical



### CLIENT LOGS

#### At A Glance



### CLIENT LOGS

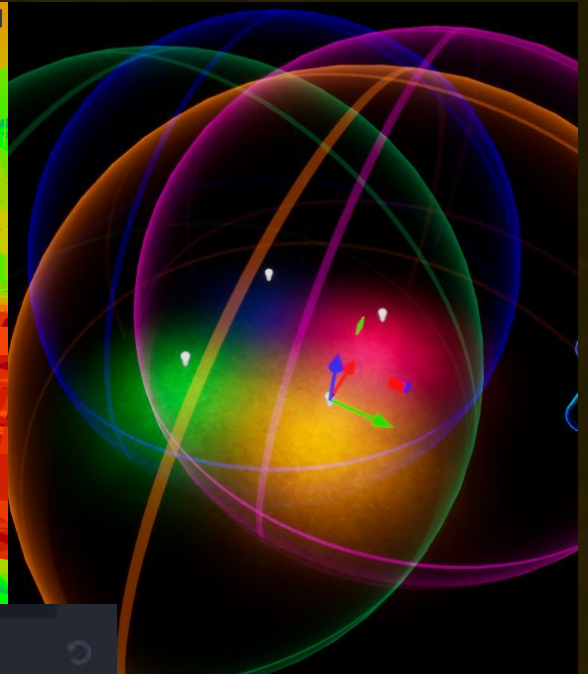
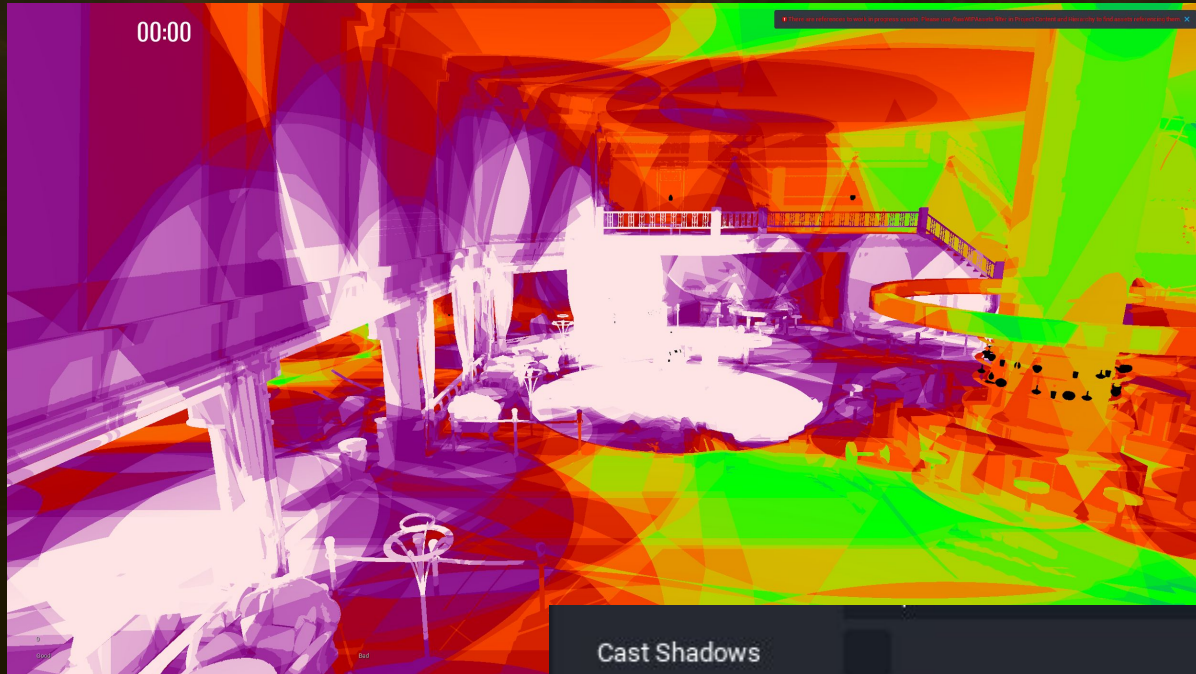
#### At A Glance





## #8 - Overlapping Light Attenuations!

- High GPU Cost associated with overlapping light attenuations
  - Cost ramps up exponentially especially when shadows are used extensively
- Tempting to use overlapping lights, especially with area lights & spotlights which cover a large amount of visual space and look cool
- Guidelines - use lights intentionally and to evoke a specific effect. Monitor GPU costs to make sure things aren't getting out of hand



# #7 - Not Using UI Contexts!

- Extreme example: ~10k UI objects on screen in a dynamic context
  - Typically worth 1-3ms of CPU time in savings
- Large cost to the CPU thread under UI Tick Time
  - Note that Static Context will only update UI when called explicitly

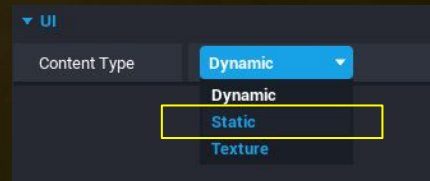
Game Thread Time: 19.496 ms

Dynamic Context

Game Thread Time: 6.132 ms

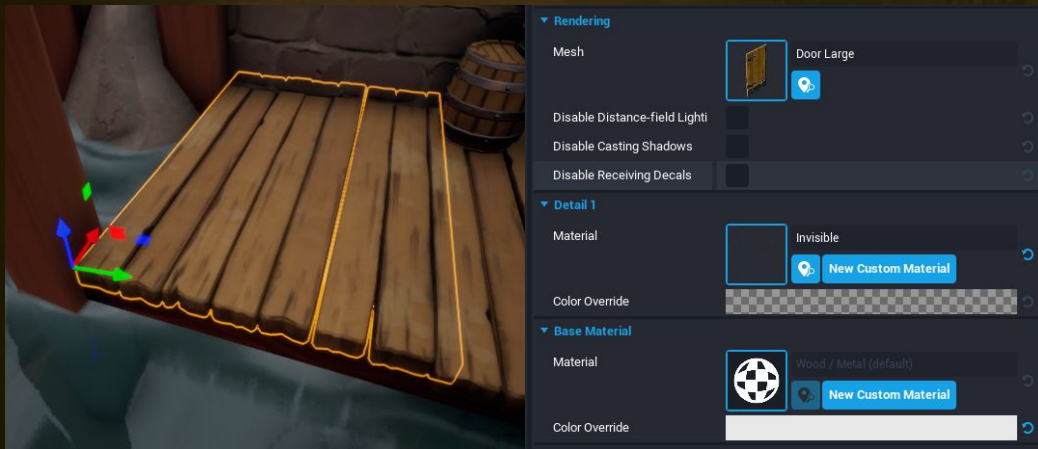
Static Context

Total Counts for [UI Container]  
Networked: 0 | Non-Networked: 12,420 | Total: 12,420

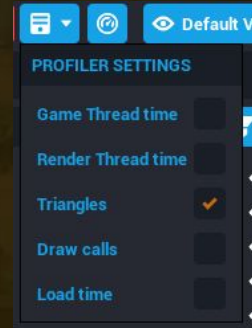


## #6 - High Triangle Counts

- Check triangle counts for objects, especially with curved surfaces, and make sure that you aren't using more triangles than actually required for environment art!
- Recommendation - reduce to or find the minimal number of assets needed to create your environment!
  - If you're really looking to optimize, find creative ways to express your art with the less expensive kitbashing!



Single Mesh vs Individual Planks



Object	Triangle Count	Visibility	Lock
pier_rope	23,608	Visible	Locked
Cylinder	384	Visible	Locked
Cylinder	384	Visible	Locked
Ring - Thick	4,608	Visible	Locked
Ring - Thick	4,608	Visible	Locked
Ring - Thick	4,608	Visible	Locked
Ring - Thick	4,608	Visible	Locked
Ring - Thick	4,608	Visible	Locked

Ring for "Rope" - could be reduced?

## #5 - VFX, large Particle Size & Density

- Using only 5 VFX smoke volume objects you can easily tank GPU performance by increasing both **particle density** & **particle size** to make it look “thicker”
  - Examples below are 5x5x5, 25 density, and 1-5 particle size
- Particle simulation adds to game thread cost
- Guidance - use minimal values for density/particle size to get the same appearance



Particle Size

1



2



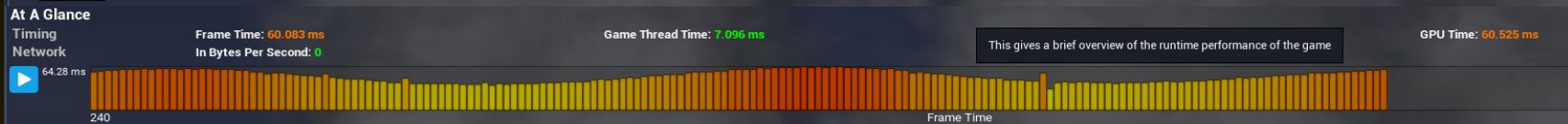
3



4

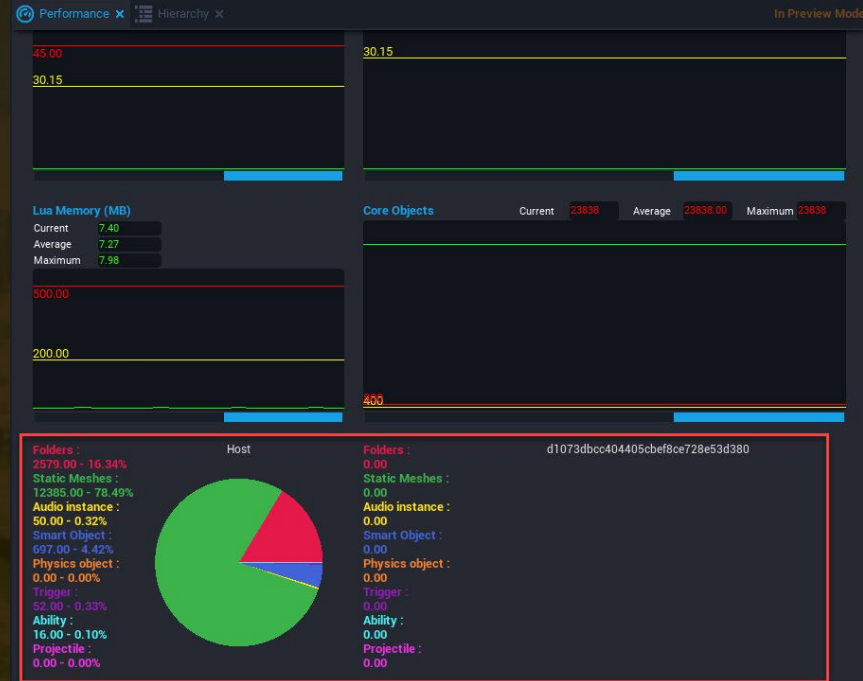
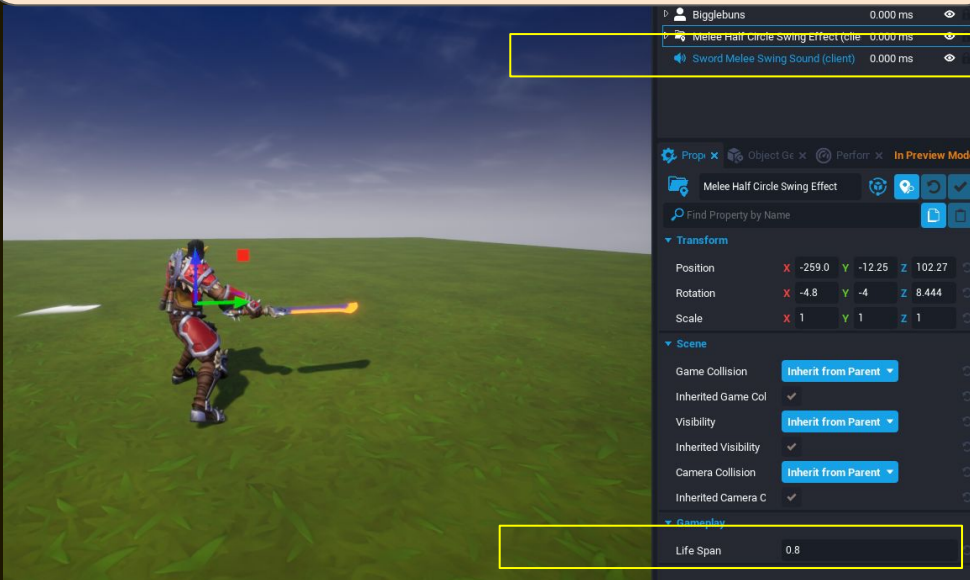


5



## #4 - Missing Lifespans on Spawned FX

- When spawning SFX or VFX, especially for weapons and equipment, **make sure that the hierarchy isn't filling up with objects** that don't expire
  - Default lifespan is set to 0, so you need to enter this manually
  - Alternatively, it is good practice to clean up objects if you have a systematic approach for spawning them
  - Be sure to check both server & client!
- These objects will accumulate over time and eventually crash the game - essentially a memory leak. May be subtle at first!
- Also note that spawning in lots of objects for weapon FX (muzzle flashes, impact FX, etc.) can get very expensive quickly!



### #3 - Too Many Mesh Objects!

- Aim for around 30,000 non-networked objects for your total map size. The actual number of objects depends greatly on the type of objects used.
  - 10k = Almost never an issue
  - 20k = Not an issue for most players
  - 30k = Can see perf issues for some players
  - 40k = Many players will have perf issues
- Mesh merging will help increase the number of total objects possible for your map.
- Most visible on the Render Thread CPU times

The screenshot shows a dark-themed window titled 'OBJECT COUNTS'. At the top, there are menu options: 'Create...', 'Hierarchy Sanity Checks', and 'Count Objects Under Selected Object'. Below these is a 'Last Inspected' field. The main content area displays a tree view of object counts:

- Total Counts for [ Keppu(Antti) ]  
Networked: 0 | Non-Networked: 23,724 | Total: 23,724
- Child [ SAR\_Atmosphere ]  
Networked: 0 | Non - Networked : 43 | Total : 43
- Child [ SAR\_SFX ]  
Networked: 0 | Non - Networked : 46 | Total : 46
- Child [ SAR\_VFX ]  
Networked: 0 | Non - Networked : 45 | Total : 45
- Child [ SAR\_LevelColliders ]  
Networked: 0 | Non - Networked : 69 | Total : 69
- Child [ SAR\_LevelArt ]  
Networked: 0 | Non - Networked : 23,521 | Total : 23,521

## #2 - Incorrect Use of Networking Contexts!

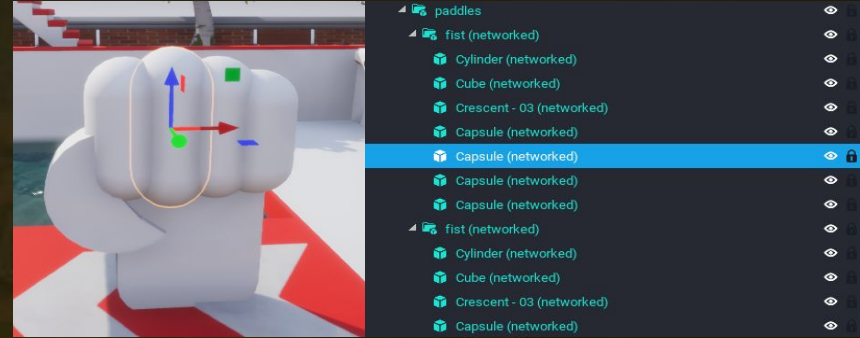
### Networked Contexts

**Client Context** - Networked context for anything which matters to the client, but does not require that all players see the exact same thing at the exact same time. VFX, UI, SFX, etc. are almost always in a client context.

**Static Context** - Used for spawning in chunks of collision without individually networking the meshes.

**Networked Context** - The most precious commodity for multiplayer games! Used for gameplay critical elements which need to be synchronized for all players in the game, for example a soccer ball which all players can interact with in a game.

Common mistake involves having more networked than is actually necessary - this will lead to rubber banding and other lag issues!



*Could be in Static Context!*

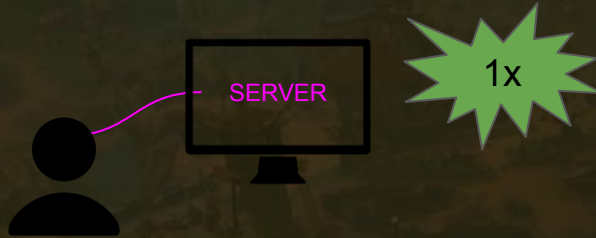


*Correct!*

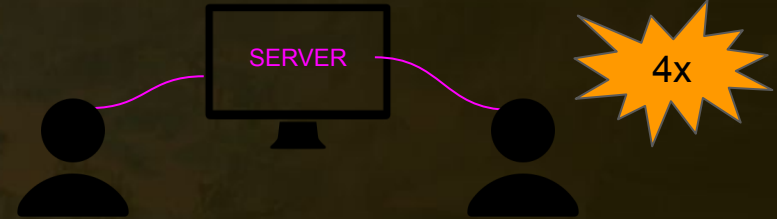


*Very Incorrect & Terribly Expensive!!!*

# #1 - Assumptions about Player Count!



Player 1 Network Object



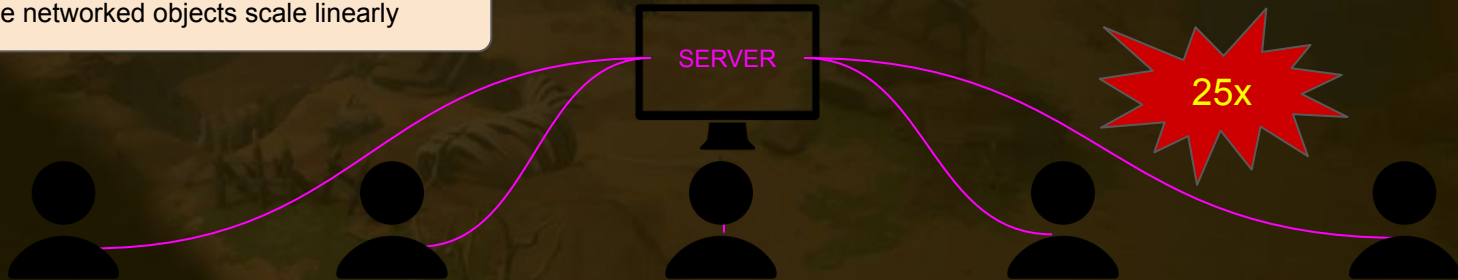
Player 1 Network Object

Player 2 Network Object

Player 1 Network Object

Player 2 Network Object

- Cost scales quadratically with player count!
- Consider what happens in a \*full game\* with your desired player count
- Scene networked objects scale linearly



Player 1 Network Object

Player 1 Network Object

Player 1 Network Object

Player 1 Network Object

Player 1 Network Object

Player 2 Network Object

Player 2 Network Object

Player 2 Network Object

Player 2 Network Object

Player 2 Network Object

Player 3 Network Object

Player 3 Network Object

Player 3 Network Object

Player 3 Network Object

Player 3 Network Object

Player 4 Network Object

Player 4 Network Object

Player 4 Network Object

Player 4 Network Object

Player 4 Network Object

Player 5 Network Object

Player 5 Network Object

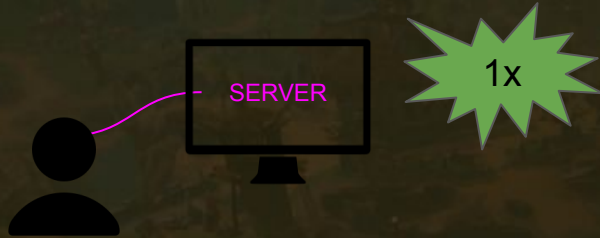
Player 5 Network Object

Player 5 Network Object

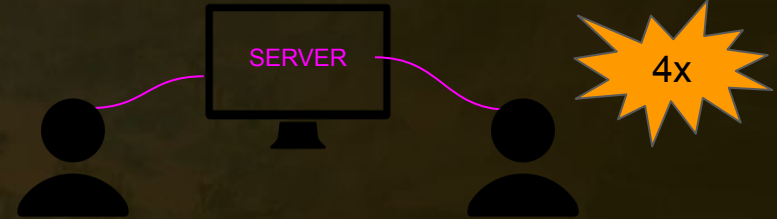
Player 5 Network Object



# #1 - Assumptions about Player Count!



Player 1 Network Object



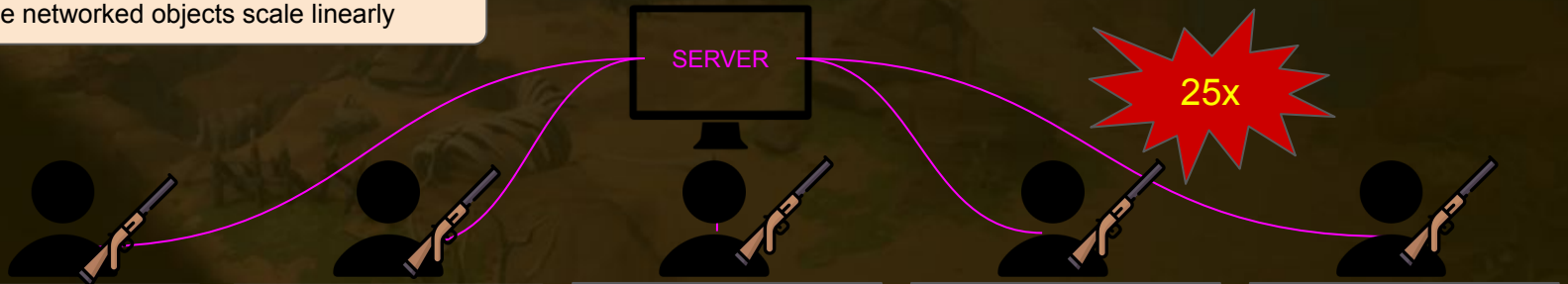
Player 1 Network Object

Player 2 Network Object

Player 1 Network Object

Player 2 Network Object

- Cost scales quadratically with player count!
- Consider what happens in a \*full game\* with your desired player count
- Scene networked objects scale linearly



Player 1 Network Object

Player 1 Network Object

Player 1 Network Object

Player 1 Network Object

Player 1 Network Object

Player 2 Network Object

Player 2 Network Object

Player 2 Network Object

Player 2 Network Object

Player 2 Network Object

Player 3 Network Object

Player 3 Network Object

Player 3 Network Object

Player 3 Network Object

Player 3 Network Object

Player 4 Network Object

Player 4 Network Object

Player 4 Network Object

Player 4 Network Object

Player 4 Network Object

Player 5 Network Object

Player 5 Network Object

Player 5 Network Object

Player 5 Network Object

Player 5 Network Object

Be mindful of things which don't impact the Player Experience, but are expensive for game performance. Use common sense!

- Does the player care about X/Y/Z?
- Is there a cheaper version of X/Y/Z which creates an equivalent player experience?
- What can I pull out of my game without the experience toppling over



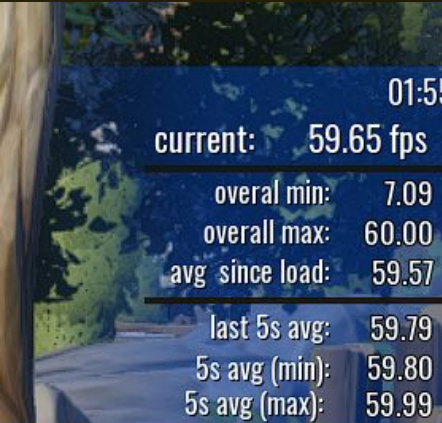
# Useful Tools!

# FPS Tracker - Coming soon to CC!

## FPS Tracker - average out FPS for high level adjustment

(will be available on Community Content!)

- Great to use for simple A/B comparisons or high level assessments of frame rates for games
- Use “/fps show” to turn on the panel
- Use “/fps reset” to start the clock back at 0
- Tracks a few categories:
  - Time since load
  - Current FPS
  - Overall min/max FPS
  - Average FPS since load (very useful!)
  - Last 5s average (very useful!)
  - Last 5s min/max



01:55

current:	59.65 fps
overall min:	7.09
overall max:	60.00
avg since load:	59.57
last 5s avg:	59.79
5s avg (min):	59.80
5s avg (max):	59.99

FPS Meter & Chathooks

FPSMeter\_v2

FPSMeterServer

FPSMeter

FPSMeterClient (client)

UI Container (client)



00:07

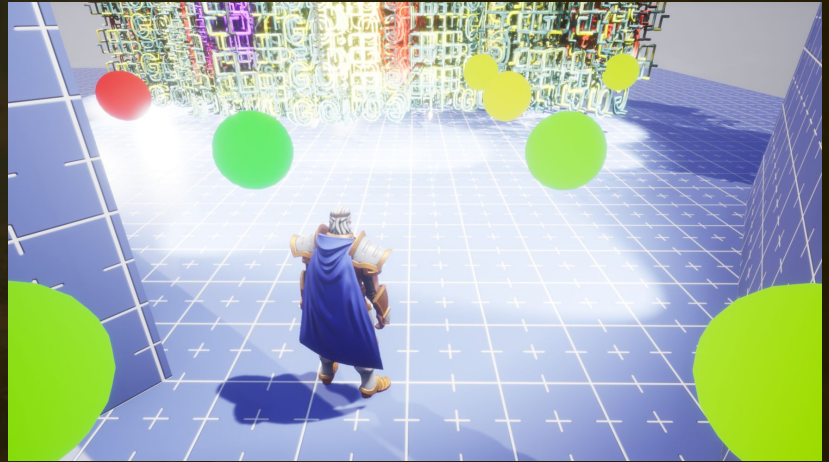
current:	143.02 fps
overall min:	44.98
overall max:	144.01
avg since load:	142.67
last 5s avg:	142.47
5s avg (min):	44.98
5s avg (max):	144.01

# Performance Mapper

## Performance Mapper Tool

(will be available on Community Content!)

- Tracks and stores player data over time for frame rate given a specific location on that map.
- Colors correspond to the quality of the frame rate for all accumulated data.
- Great for testing a large, complex map with lots of potential “hotspots” suffering from worse performance.
  - Utilize chat commands to download, clear, display data



# Other Misc Tips!

# Low Fidelity Versions (part 1)

## Use high fidelity and low fidelity versions of complex assets (tanks!)

- For your own tank, render the high fidelity version (more objects, better VFX, etc.)
- For other player tanks, typically far away, use a lower fidelity model with fewer overall objects.
  - Check if you have the local player and equip the corresponding object.



Total Counts for [TANK\_SKIN\_Default\_GER\_Panzer3\_LP]  
Networked: 0 | Non-Networked: 241 | Total: 241

Child [AdjustmentPoint]  
Networked: 0 | Non - Networked : 239 | Total : 239

Child [AllyOutline]  
Networked: 0 | Non - Networked : 1 | Total : 1

Child [EnemyOutline]  
Networked: 0 | Non - Networked : 1 | Total : 1

Total Counts for [TANK\_SKIN\_Default\_GER\_Panzer3]  
Networked: 0 | Non-Networked: 443 | Total: 443

Child [AdjustmentPoint]  
Networked: 0 | Non - Networked : 441 | Total : 441

Child [AllyOutline]  
Networked: 0 | Non - Networked : 1 | Total : 1

Child [EnemyOutline]  
Networked: 0 | Non - Networked : 1 | Total : 1

## #15 - Low Fidelity Versions (part 2)

Game Thread time

- Self: 0.882 ms
- Children: 0.197 ms

TANK\_CHASSIS\_Default\_GE\_Stug3G (1): 1.022 ms

- Script: VehicleDamageEffectsClient (1): 0.057 ms
- fxbp\_smooth\_tankTread\_trail\_C Particles (2): 0.035 ms
- Wheel Rotate (22): 0.033 ms
- Cannon Rotate (2): 0.031 ms
- fxbp\_smoke\_volume\_vfx\_C Particles (2): 0.031 ms
- CannonGuide Rotate (1): 0.010 ms
- fxbp\_local\_outline\_C (2): 0.003 ms
- fxbp\_env\_tank\_treads\_C (4): 0.002 ms
- PFHealthComponent (1): 0.001 ms
- Turret Rotate (1): 0.000 ms
- Tank Camera (1): 0.000 ms
- Sniper Camera (1): 0.000 ms

434 objects total

Game Thread time

- Self: 0.800 ms
- Children: 0.232 ms

TANK\_CHASSIS\_Default\_GE\_Stug3G (1): 0.640 ms

- Cannon Rotate (2): 0.055 ms
- Script: VehicleDamageEffectsClient (1): 0.051 ms
- fxbp\_smoke\_volume\_vfx\_C Particles (2): 0.038 ms
- fxbp\_smooth\_tankTread\_trail\_C Particles (2): 0.031 ms
- Wheel Rotate (22): 0.030 ms
- CannonGuide Rotate (1): 0.008 ms
- fxbp\_env\_tank\_treads\_C (4): 0.005 ms
- fxbp\_local\_outline\_C (2): 0.002 ms
- Tank Camera (1): 0.002 ms
- PFHealthComponent (1): 0.001 ms
- Sniper Camera (1): 0.000 ms
- Turret Rotate (1): 0.000 ms

1.031 ms

Game Thread time

- Self: 0.561 ms
- Children: 0.279 ms

TANK\_CHASSIS\_Default\_GE\_Stug3G (1): 0.355 ms

- Cannon Rotate (2): 0.055 ms
- fxbp\_fire\_volume\_vfx\_C Particles (1): 0.039 ms
- fxbp\_smooth\_tankTread\_trail\_C Particles (2): 0.038 ms
- Wheel Rotate (22): 0.036 ms
- fxbp\_smoke\_volume\_vfx\_C Particles (2): 0.032 ms
- fxbp\_env\_tank\_treads\_C (4): 0.003 ms
- fxbp\_local\_outline\_C (2): 0.003 ms
- Script: VehicleDamageEffectsClient (1): 0.003 ms
- CannonGuide Rotate (1): 0.003 ms
- Tank Camera (1): 0.001 ms
- PFHealthComponent (1): 0.000 ms
- Turret Rotate (1): 0.000 ms
- Sniper Camera (1): 0.000 ms

271 objects total

Game Thread time

- Self: 0.422 ms
- Children: 0.194 ms

TANK\_CHASSIS\_Default\_GE\_Stug3G (1): 0.327 ms

- Wheel Rotate (22): 0.030 ms
- fxbp\_smooth\_tankTread\_trail\_C Particles (2): 0.027 ms
- Cannon Rotate (2): 0.026 ms
- fxbp\_smoke\_volume\_vfx\_C Particles (2): 0.024 ms
- fxbp\_env\_tank\_treads\_C (4): 0.003 ms
- fxbp\_local\_outline\_C (2): 0.002 ms
- CannonGuide Rotate (1): 0.002 ms
- Script: VehicleDamageEffectsClient (1): 0.001 ms
- Tank Camera (1): 0.001 ms
- PFHealthComponent (1): 0.000 ms
- Sniper Camera (1): 0.000 ms
- Turret Rotate (1): 0.000 ms

0.616 ms

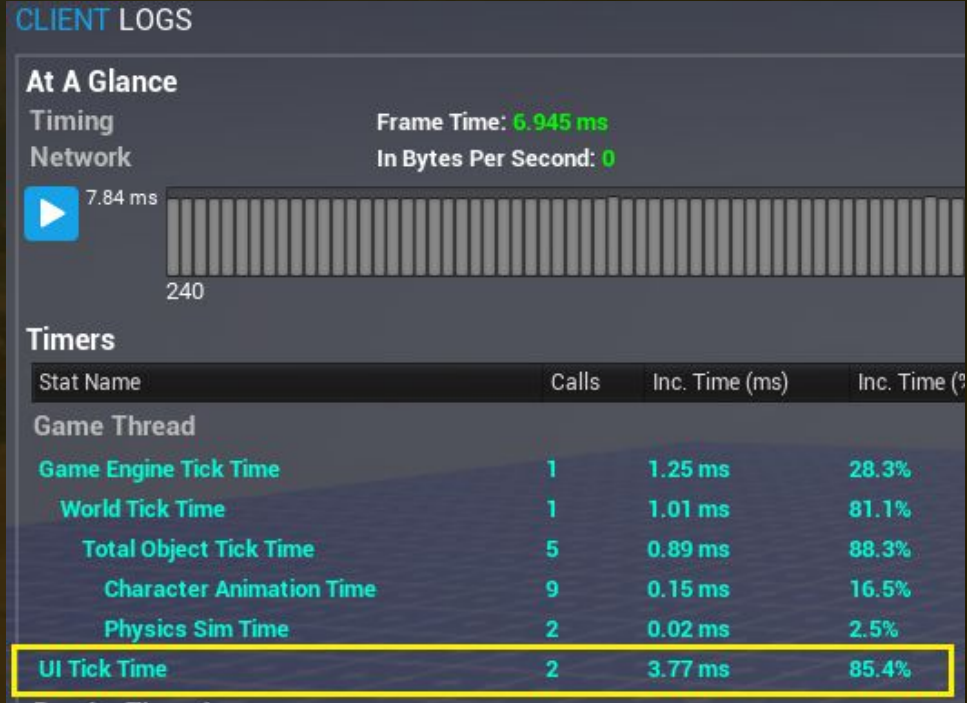
~.4ms faster  
(-161 objects)



# UI Tick Time Breakdown

## Profiler Tool & UI Tick Time

- Note that the “UI Tick Time” field has a cost associated with having the profiler open (~2-4ms) to actually render the UI visuals. This is categorized under the Game Thread CPU time.
- When closed, this cost is not incurred - although there is a very small cost to having it activated in your game generally.
- Therefore - you should be aware that the Game Thread CPU time is actually lower than displayed in the aggregate ms count. It may be better to optimize the render thread before the game thread given this information.





# Questions!