Consoles

XBOX ONE

Main Processor
- 8 Core CPU custom built by Microsoft

Graphics Processor
- D3D 11.1 chip with 32 MB embedded memory

Memory
- 8GB DDR3

Hard Disk Drive
- Built-in 500GB

Optical Drive
- Blu-ray/DVD combo drive

I/O
- USB 3.0

Communication
- Ethernet
- Three 802.11n radios (for controller and other device connectivity)
- WiFi Direct

AV Output
- HDMI

PS4

Main Processor
- Single-chip custom processor
- CPU: low power x86-64 AMD 'Jaguar', 8 cores

Graphics Processor
- GPU: 1.84 TFLOPS, AMD Radeon™ Graphics Core Next Engine

Memory
- 8GB GDDR5

Hard Disk Drive
- Built-in

Optical Drive
- Blu-ray/DVD combo drive

I/O
- Super-Speed USB (USB 3.0)

Communication
- Ethernet (10BASE-T, 100BASE-TX, 1000BASE-T)
- IEEE 02.11 b/g/n
- Bluetooth® 2.1(EDR)

AV Output
- HDMI
- Analog-AV out
- Digital Output (optical)

Goals?

- Ideal: 1080p @60fps
- Realistic: at least 900p and never drop below 30 fps
- 50GB of data possible on one Blu-ray
- >50GB require some content is downloaded- required internet connection

	Xbox One X	PS4 Pro
		
CPU	Eight custom x86 cores clocked at 2.3GHz	Eight-Core AMD Custom "Jaguar" at 2.1GHz
GPU	Integrated AMD graphics with 6 teraflops of performance	Integrated AMD Polaris graphics with 4.2 teraflops of performance
RAM	12GB GDDR5	8GB GDDR5
Size	11.8 x 9.4 x 2.4 inches	12.8 x 11.6 x 2.1 inches
HDD	1TB	1TB
Weight	8.4 Pounds	7.2 Pounds
Price (USD)	$499	$399

Goals?

- Ideal: 2160p @60fps
- Realistic: at least 900p and never drop below 30 fps
- 4K Blu-rays can hold 100GB of data (these are not BC)

FPS

- Determined by the performance of the console – CPU + GPU (however, GPU is typically the one that causes frame rate hiccups)
- How quickly can everything on the screen be rendered
- The more that has to be rendered the lower the frame rate
- The more calculations taking place per frame the lower the frame rate
- Lowering the resolution increases fps

Film vs games

- Film is usually 30 or less fps (24)
- Film frames are blurred into the next frame
- Films with >30 fps can loose motion blur
- Games do not have a blur between frames – 30 fps means 30 fully rendered frames per second
- Frame rate drops are much more obvious in games than film

Hz vs FPS

- Hz – the refresh rate, how many times your screen draws per second
- Hz - purely a function of your monitor or television
- 60Hz is where the brain can’t see image flickering/judder – TV standard. (120Hz is becoming more and more popular)
- FPS is how many times per second the game renders a frame
- If the FPS is less than your refresh rate the same frame may be redrawn several times
- An FPS greater than your refresh rate will cause tearing
- Capping at 60 FPS prevents tearing

Keep in Mind

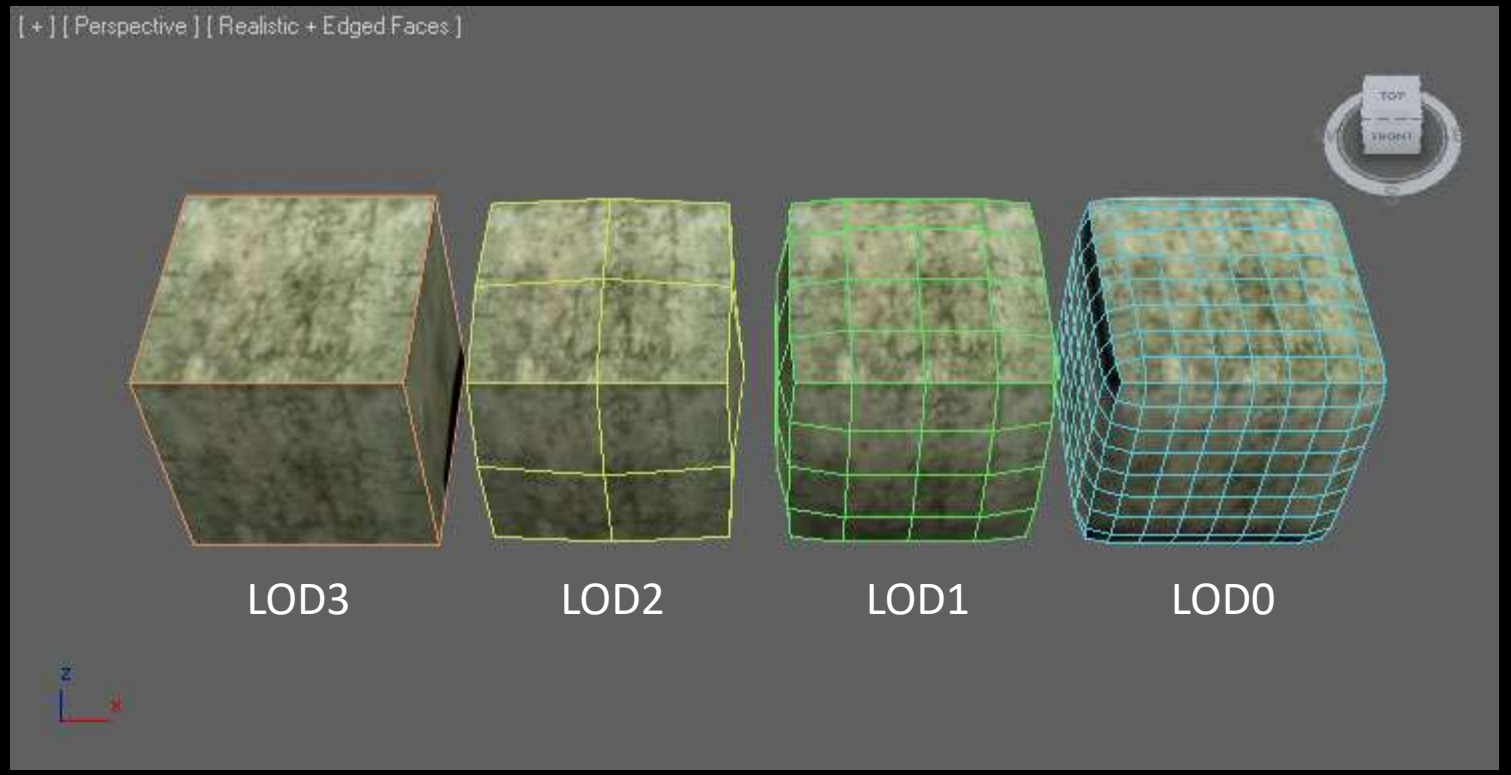
- Games are frequently expected to work on multiple consoles.
- Some Games are on console and PC.

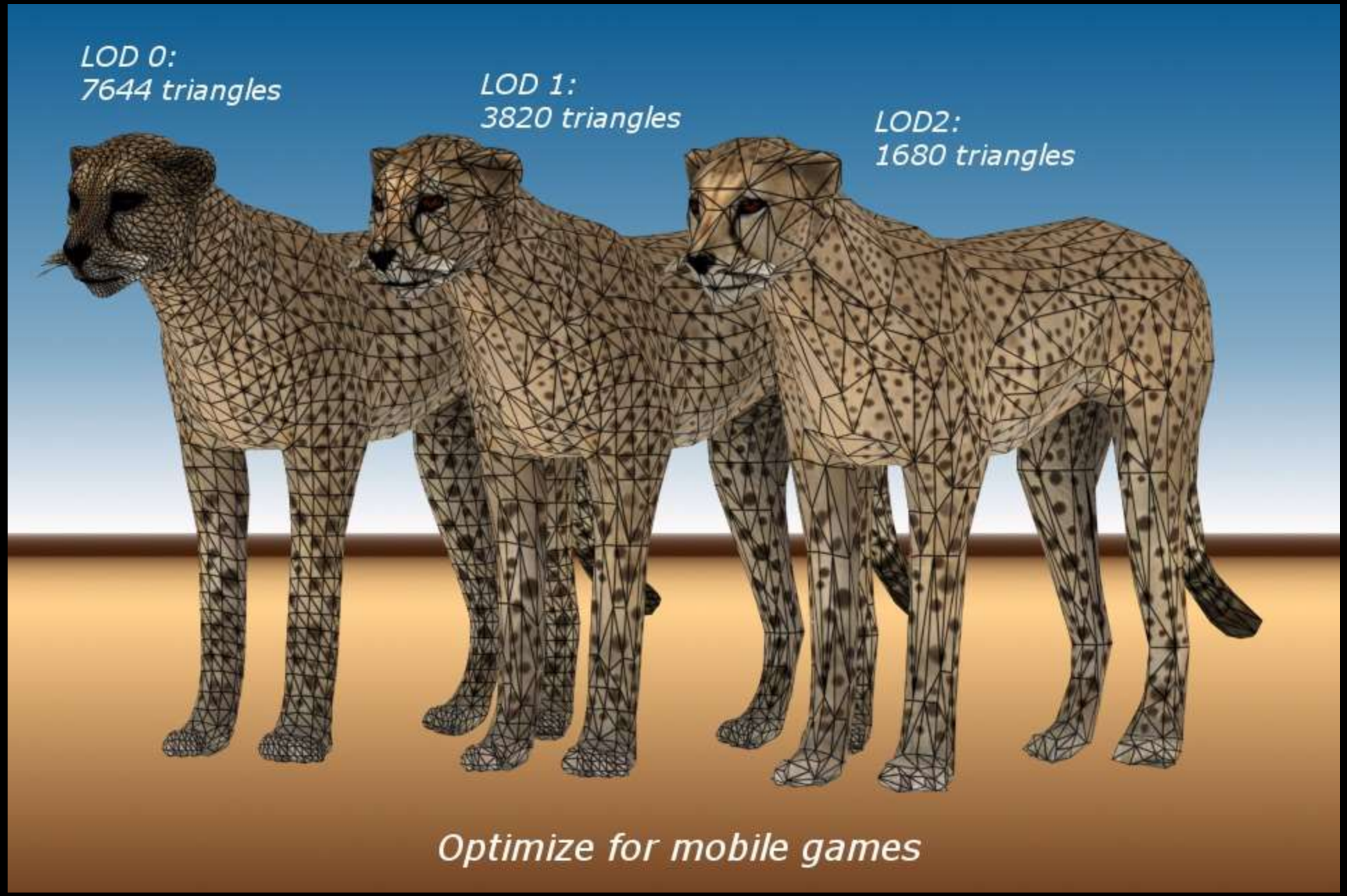
Supporting multiple platforms?

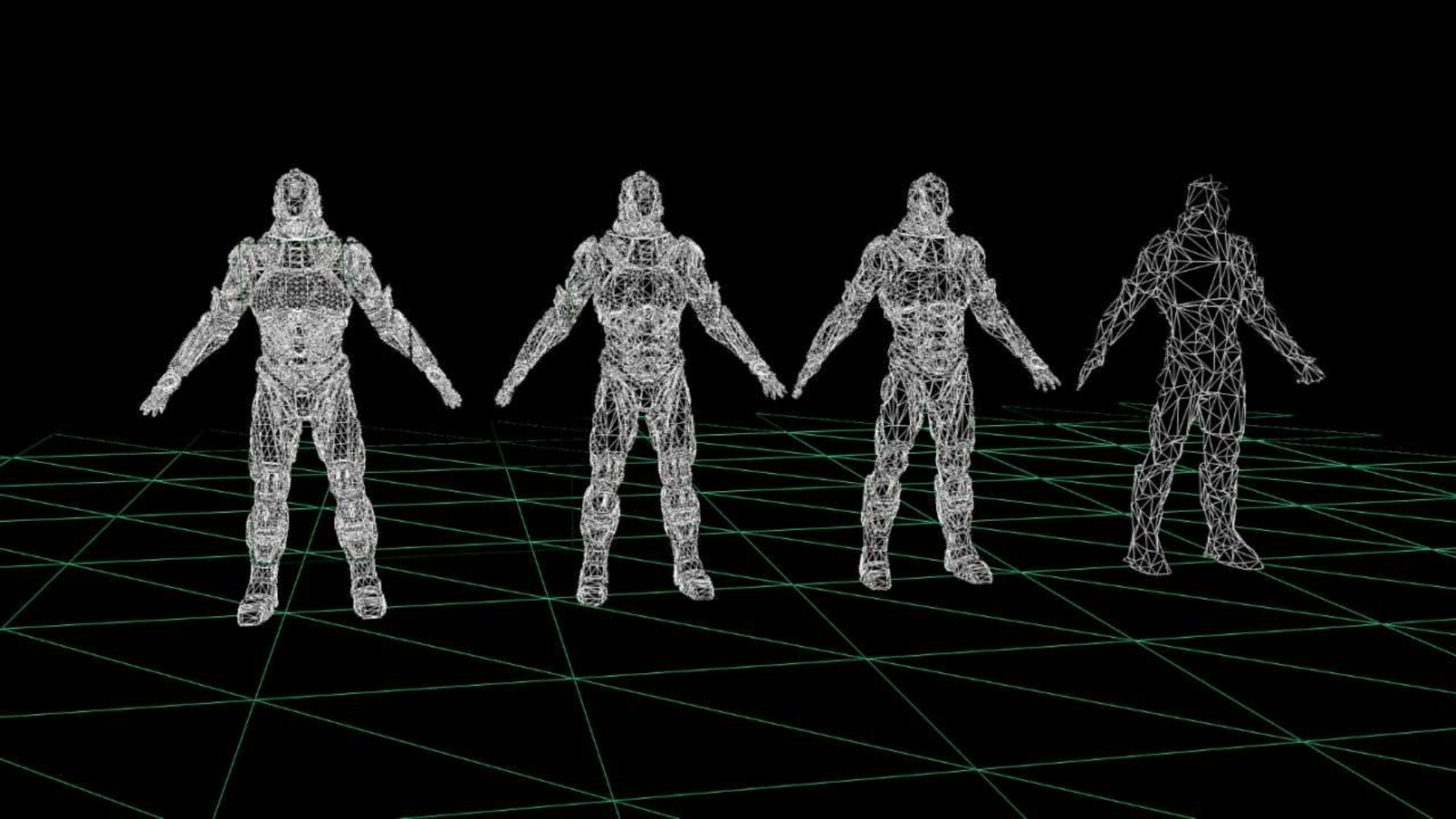
- Scalability
- Content markup
- Switching LODs/adjusting LOD distance
- Changing texture sizes, mip map distance
- Change entire resolution

LODs

- Level of Detail (LOD) is the concept of decreasing the complexity of a 3D model as the camera moves farther away from that model.

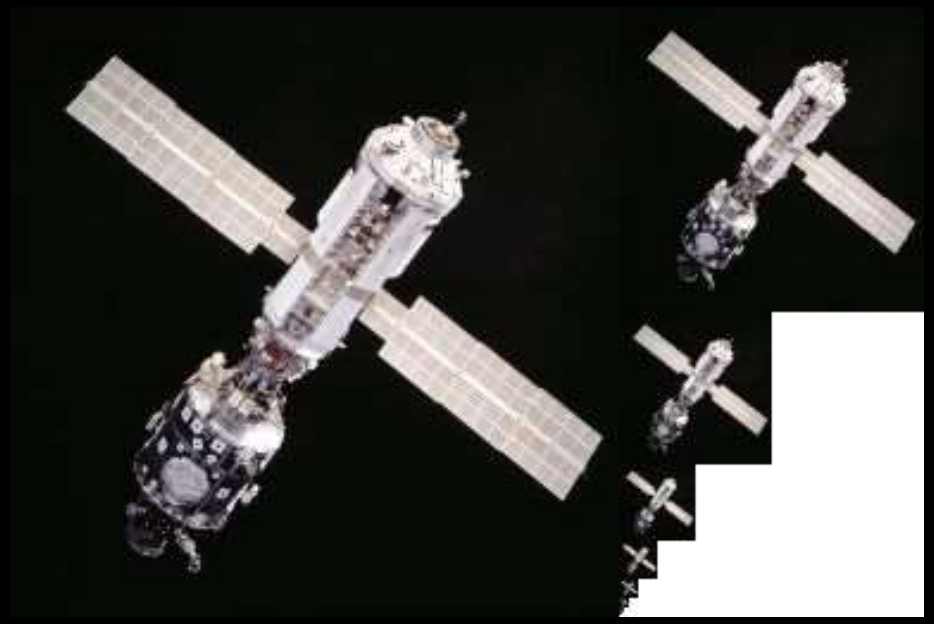



LOD 0:
7644 triangles
LOD 1:
3820 triangles
LOD2:
1680 triangles
Optimize for mobile games



Mipmapping

- Technique where an original high resolution map is scaled and filter into multiple resolutions all stored in the same texture map. The texture changes based on the distance the asset is from the camera.



Virtual Reality

The top virtual reality devices compared



Model	Oculus Rift	HTC Vive	PS VR
In the box	Headset, motion sensor, wireless Xbox One controller, remote, 2x games, and cables	Headset, 2x motion tracking cubes, 2x wireless controllers, link box, earbuds, and cables	Headset, processor unit, camera, earbuds, 2x PS Move controllers, 1x game, and cables
Headset price	~$599	~$799	$460
PC/VR bundle cost	~$1,500	~$1,800	$800+
Release date	3/28/2016	4/5/2016	10/1/2016
Display	AMOLED	OLED	OLED
Resolution	1200 X 1080 (per eye)	2160 X 1200	1920 X 1080
Field of view	110	110	100
Headset weight	~1.03 lbs.	1.21 lbs.	1.34 lbs.
Platform	Oculus, Steam VR	Steam VR	PlayStation
Operating system	Windows 7 or newer	Windows 7 or newer	PlayStation 4
Refresh rate (higher is better)	90Hz	90Hz	120Hz, 90Hz
Controllers	Xbox One wireless controller	Wireless controllers	PS4 wireless controllers
Number of games on launch	30	30+	50 by EOY
Pass-through camera	No	Yes	No
Connections needed	HDMI, 3x USB 3.0, 1x USB 2.0	HDMI, 3x USB 3.0, 1x USB 2.0	HDMI, USB, separate processor unit
Recommended PC requirements	Nvidia GTX 970/AMD R9 290, Intel i5-4590, 8GB RAM, HDMI 1.3	Nvidia GTX 970/AMD R9 290, Intel i5-4590/AMD FX 8350, 4GB RAM, HDMI 1.4	PlayStation 4

TECH INSIDER

Goals?

- > 60 fps (90 fps – match refresh rate)
- Resolution matches the screen
- High screen percentage ~ > 100%

VR vs Console

- VR - Lower frame rate causes sickness – smoothness has a greater importance, ultimate smoothness when Hz = FPS
- Resolution of VR appears worse because it is rendered closer to the eye – screen door effect – when the lines between pixels are visible
- Tearing is more visible in VR
- For VR the screen is rendered twice
- PSVR will double frames at 60fps to increase smoothness when displayed at 120hz

Color Calibration

- A game is made on multiple monitors
- Consumers can play the game on any number of different screens (possibly at different refresh rates)
- Major VR headsets are restricted to specific monitors (they are the screen)

Budgets and Optimization

Game Industry Anecdote Optimization for Halo 4

http://www.gdcvault.com/play/1020641/Technical-Artist-Bootcamp-Halo-4

Terms

- Budget – amount of resources made available for an asset
  - Amount of memory allocated
  - Amount of milliseconds to render allocated
  - Amount of time needed to finish a task
  - Amount of people needed to finish a task
  - Amount of money needed to finish a task (time + people)
- Budget report – break down of the different pieces that make up a full asset and how much memory, time to render, and occasionally physical resources (time and people) are required for that asset
- Many studios have tools that output this data for an asset in a game – “Budget Reporting Tools”

Terms

- Memory – how many bytes a piece of content is
- Performance – relates to how quickly and efficiently the game runs

Memory vs Performance

- Memory relates to how much physical space we have. All content has a “size” or an amount of memory it takes up. Size.
  - Hard data
  - Numerical
  - Pieces of the whole that add up to the total *size* of the game world
- Performance relates to how quickly we can use this content/this memory. Speed.
  - FPS
  - How quickly we can make the image we want to make
  - If the game judders or hiccups it means the performance was bad.
  - Bad performance can be because the game is trying to load too much from memory

GPU vs CPU

- GPU – performs the rendering of assets at each frame – the speed of which reflects in the framerate. Hiccups, sudden changes in framerate, are usually called “perf issues”.
- CPU – controls the game state data, AI calculations, physics, collision, damage, inputs, statistics, audio, etc. Too many calculations for the CPU can also cause sudden frame rate drops
- You can’t borrow memory between the two
- The CPU feeds the GPU
- CPU gives instructions about what the GPU needs to draw
- You don’t want threads on either to be waiting for to long
- You can improve perf by adjusting memory allocation. Similarly you can also improve perf by decreasing the amount of content, decreasing the overall required memory

Render Time per frame

- Each asset type typically is allowed a render time
- This is given in milliseconds
- E.g. an expensive character in a cinematics is allotted 3-4 ms for it to render per frame.
- These are guidelines which should be set early in production per game shot
- You want to leave some unplanned milliseconds as a buffer for more expensive content or unexpected content so you can borrow from these ms later on in the project – a millisecond contingency plan

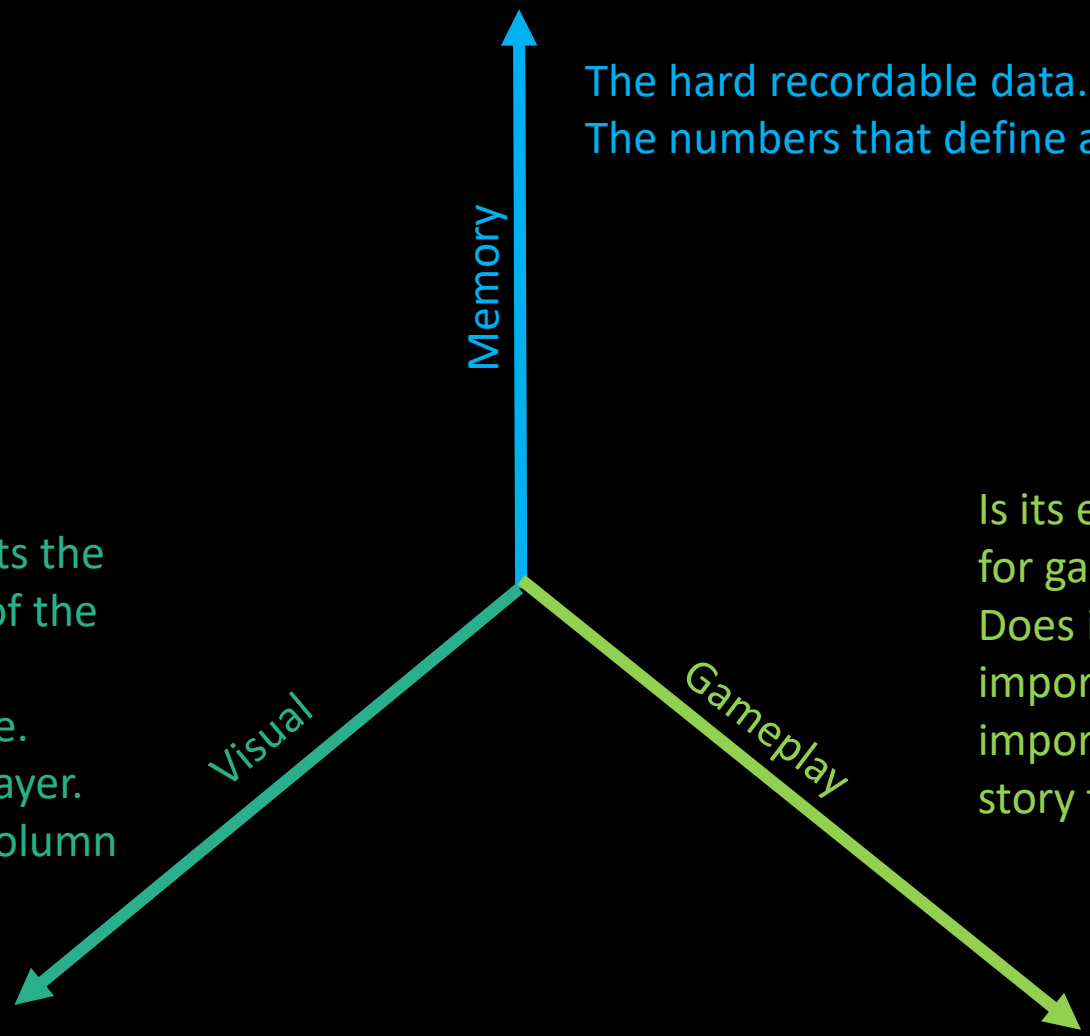
What is a Millisecond?

- 1000 milliseconds = 1 second
- To reach 60 FPS you have 16.6 milliseconds to work with each frame
- Each asset takes a number of milliseconds to render per frame
- Smart teams set guidelines for how many milliseconds are allowed per asset type to render per frame
- i.e. How many ms FX has in the shot

What changes the render time in art?

- Texture size
- Draw calls
- Vertex count
- Shader instruction count
- Lighting

How to determine an assets allocated budget?



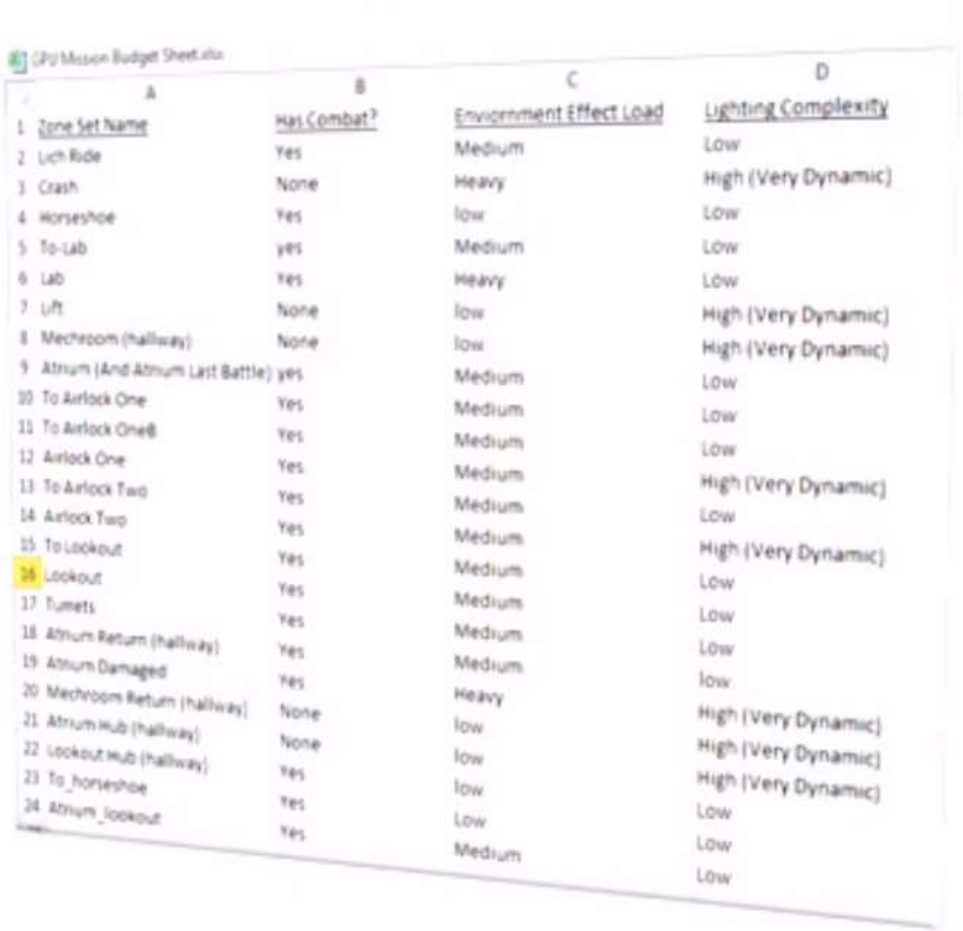
The hard recordable data.
The numbers that define a piece of content

How much it impacts the users visual image of the world.
How big should it be.
How close to the player.
The type of asset, column vs face.

Is its existence important for gameplay and design?
Does it communicate an important location, is it an important character for story telling?

Performance Planning: Frame Budgets

- Identifying critical costs
- Per area budgets
  - Negotiating for ms
  - Finding the right mix
  - AI budgeting
- Allowing overhead


GPU Mission Budget Sheet.xlsx

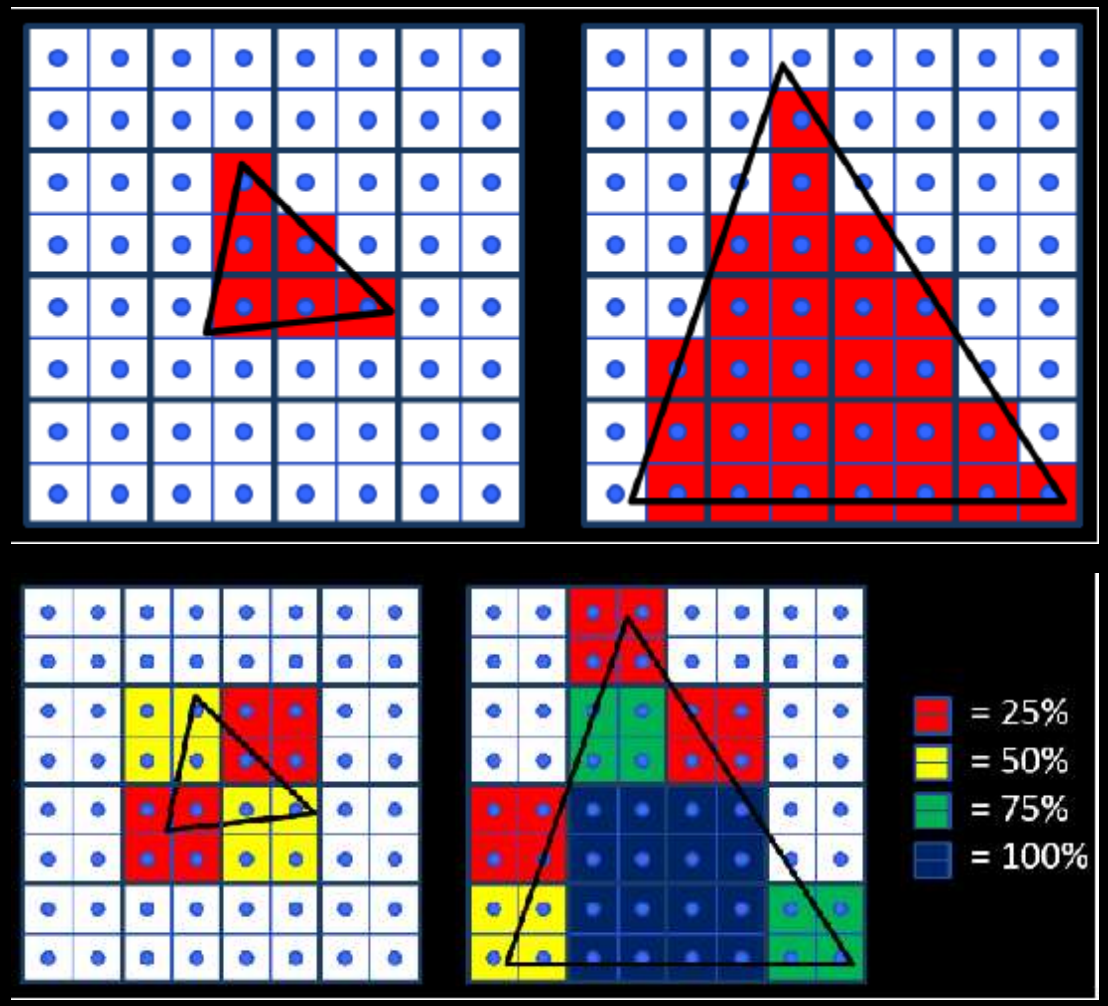
	A	B	C	D
1	Zone Set Name	Has Combat?	Environment Effect Load	Lighting Complexity
2	Lich Ride	Yes	Medium	Low
3	Crash	None	Heavy	High (Very Dynamic)
4	Horseshoe	Yes	low	Low
5	To-Lab	yes	Medium	Low
6	Lab	Yes	Heavy	Low
7	Lift	None	low	High (Very Dynamic)
8	Mechroom (hallway)	None	low	High (Very Dynamic)
9	Atrium (And Atrium Last Battle)	yes	Medium	Low
10	To Airlock One	Yes	Medium	Low
11	To Airlock One8	Yes	Medium	Low
12	Airlock One	Yes	Medium	High (Very Dynamic)
13	To Airlock Two	Yes	Medium	Low
14	Airlock Two	Yes	Medium	High (Very Dynamic)
15	To Lookout	Yes	Medium	Low
16	Lookout	Yes	Medium	Low
17	Turrets	Yes	Medium	Low
18	Atrium Return (hallway)	Yes	Medium	low
19	Atrium Damaged	Yes	Heavy	High (Very Dynamic)
20	Mechroom Return (hallway)	None	low	High (Very Dynamic)
21	Atrium Hub (hallway)	None	low	High (Very Dynamic)
22	Lookout Hub (hallway)	Yes	low	Low
23	To_horseshoe	Yes	Low	Low
24	Atrium_lookout	Yes	Medium	Low

Performance Planning: Level Review

- Early layout review
- Line of sight checking

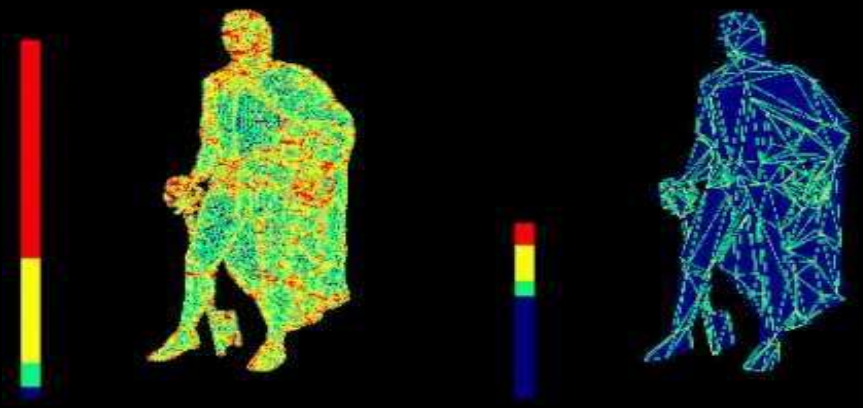
Why are more verts more expensive?

- More verts = more triangles
- Move tris = More overshading
- When shading, a pixel will be drawn if the triangles overlaps the center position of the pixel
- Pixel processing is done per tris



Overdraw

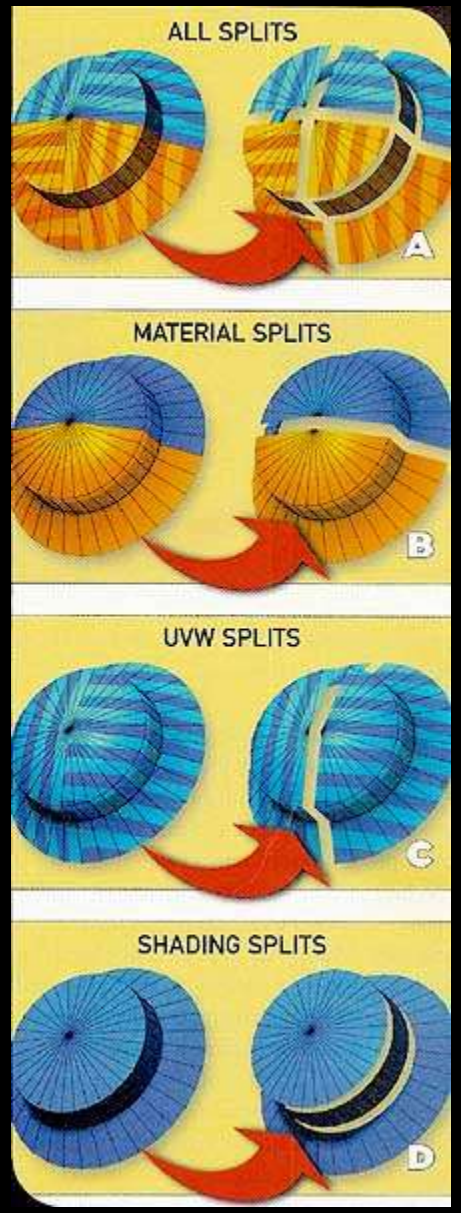
- Overdraw is when the same pixel is rendered multiple times.
  - Can be due to poor tessellation – lots of skinny long triangles or lots of small triangles
  - Happens when transparent materials overlap each other

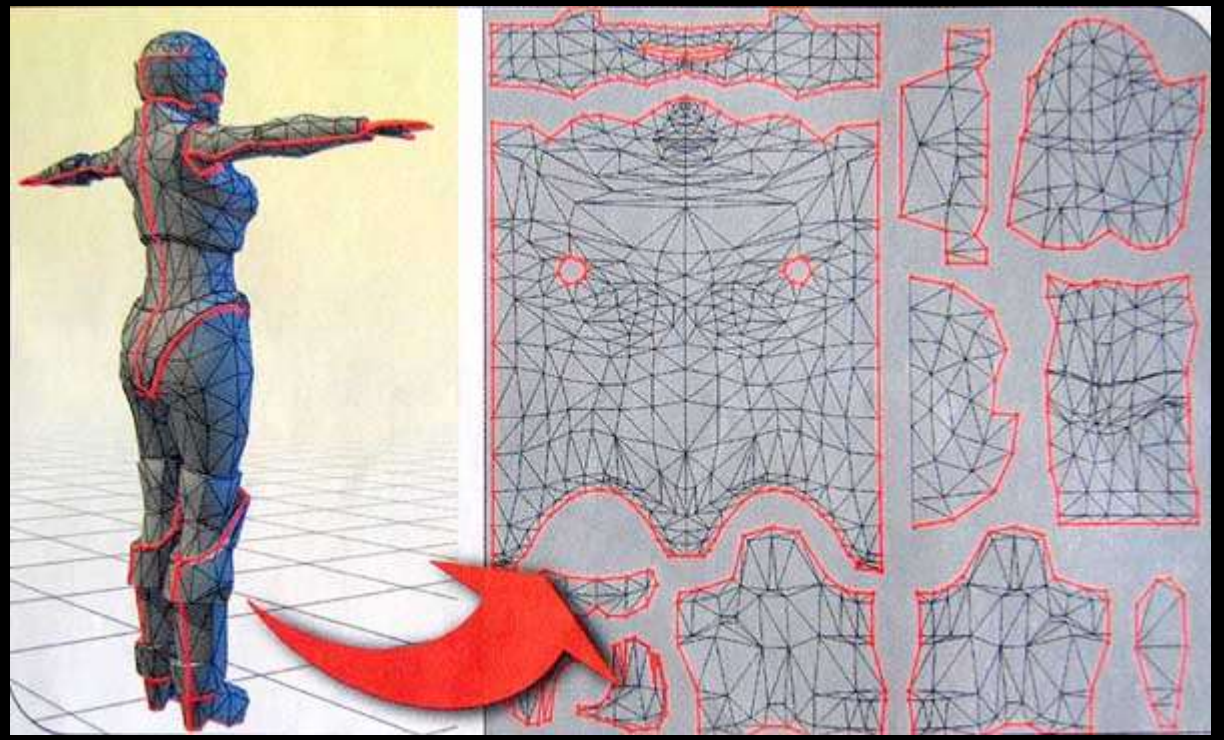


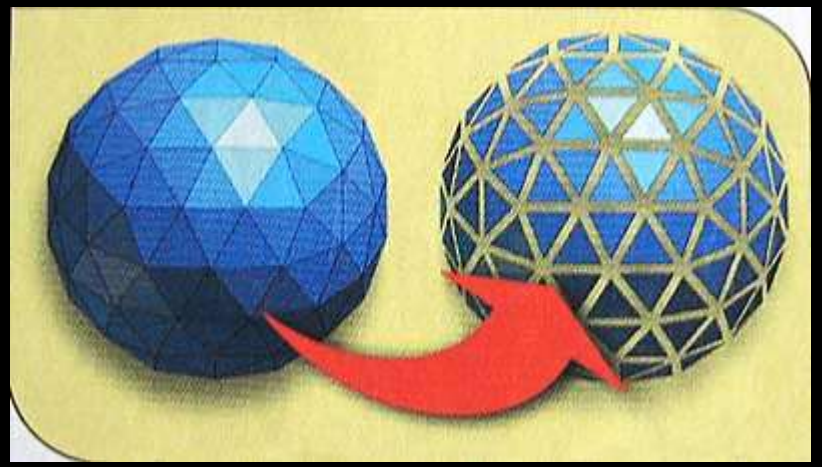
DCC vert count != runtime vert count

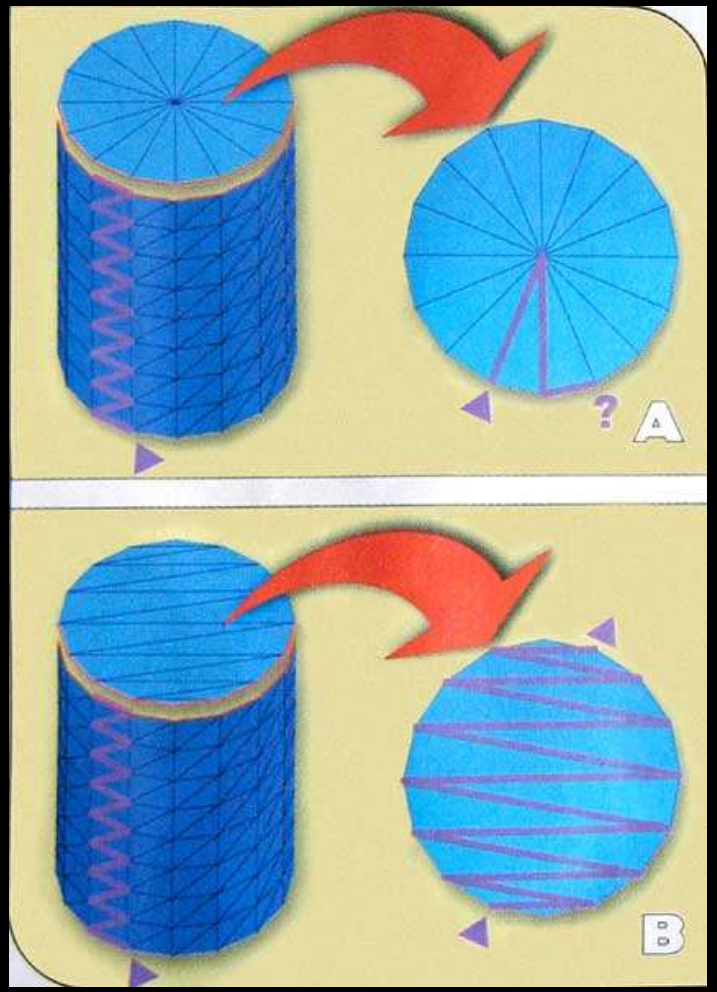
Real cost of my asset

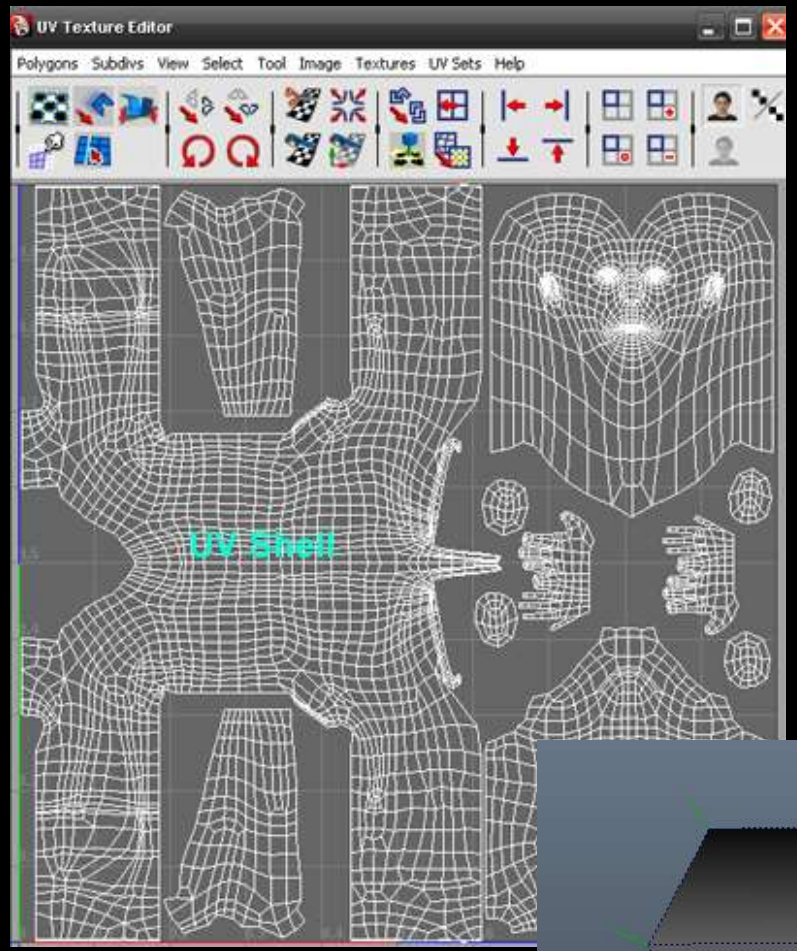
- UV islands
- Smoothing groups (normals)
- Materials
- Vertex Color
- Skin weighting
- Game markup (metadata)

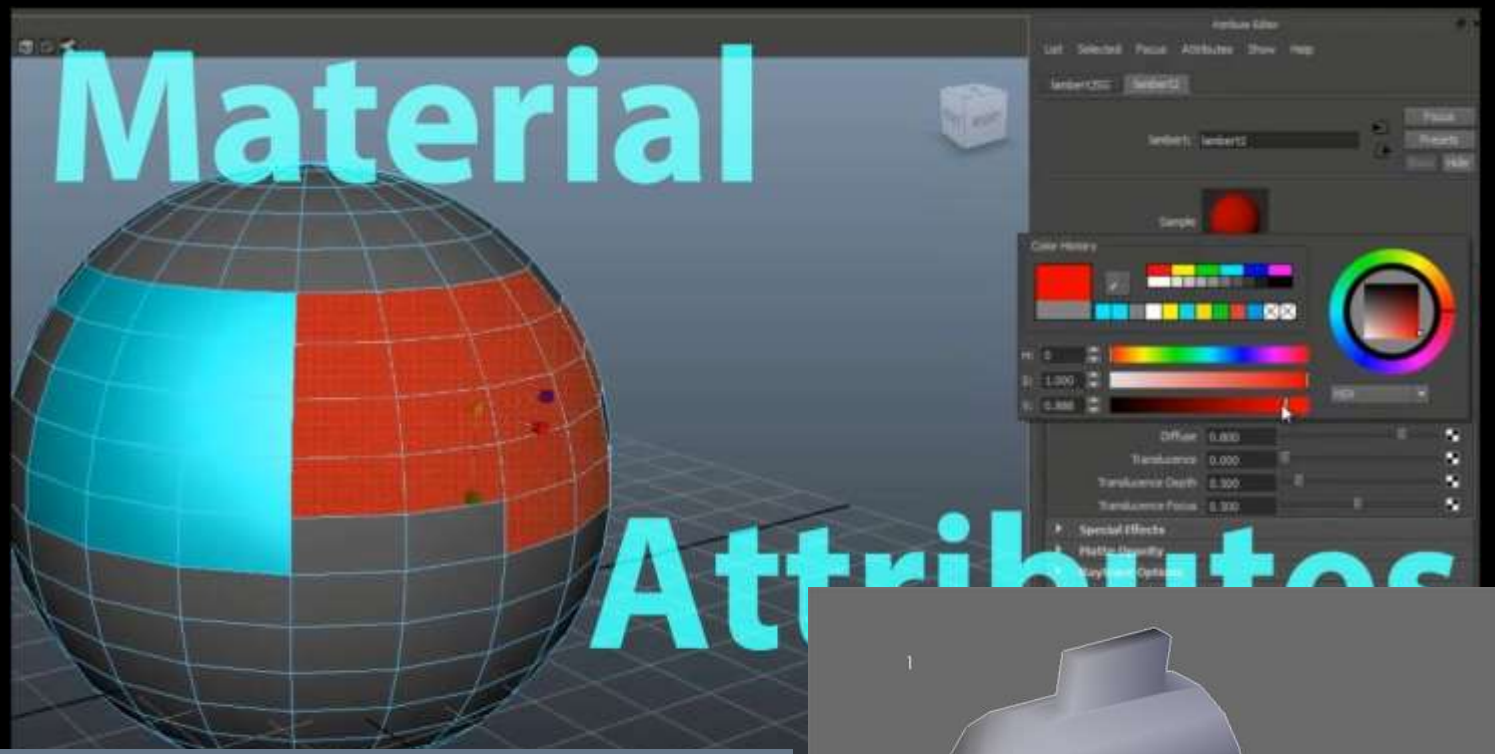

ALL SPLITS
A
MATERIAL SPLITS
B
UVW SPLITS
C
SHADING SPLITS
D

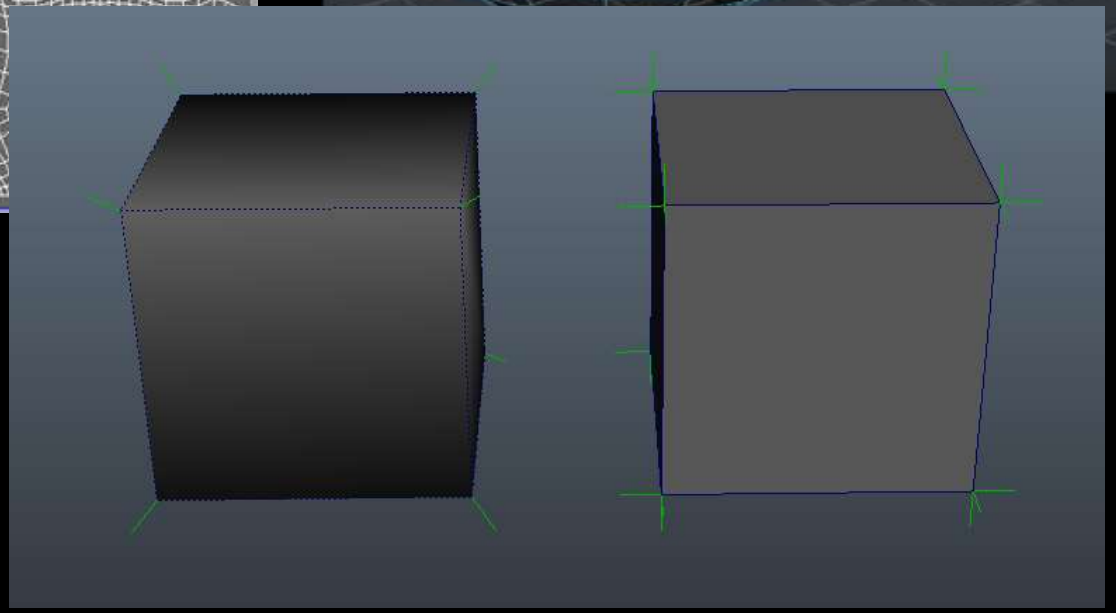


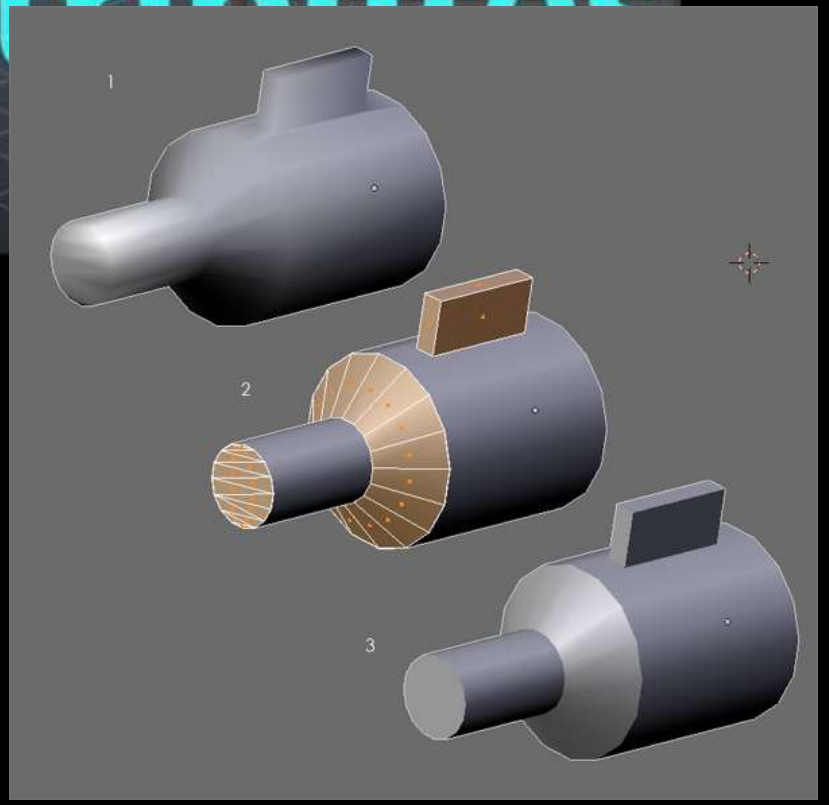



?
A
B


UV Texture Editor
Polygons Subdivs View Select Tool Image Textures UV Sets Help
UV Shell


Material
Attributes




1
2
3

Best Practices for Asset Creation

- To minimize vertex count you should have your uv-map as continuous as possible
- Have smooth normals or share hard normal edges with UV edges
- Share the same material across you mesh as much as possible
- Lower poly models with high rez textures are more performant
- Create edges that look smooth on low poly meshes by smoothing your normals
- Try to avoid sharing a single vertex with too many tris
- Keep texel density consistent to reduce extra mipping calculations

Rendering

CPU

- To render objects on the screen, the CPU has a lot of processing work to do: working out which lights affect that object, setting up the shader and shader parameters, and sending drawing commands to the graphics driver, which then prepares the commands to be sent off to the graphics card.
- All this “per object” CPU usage is resource-intensive, so if you have lots of visible objects, it can add up
- To optimize CPU performance, combine multiple objects that share the same textures together
- CPU also processes vertex calculations such as skinning, cloth simulation, and particles

GPU

- GPU is often limited by **fillrate** or memory bandwidth.
- **fillrate** refers to the number of pixels a video card can render to screen and write to video memory or ram in a second
- If lowering your resolution increases your framerate most likely the fillrate is your bottleneck
- The GPU could have too many vertices to process. The number of vertices that is acceptable to ensure good performance depends on the GPU and the complexity of vertex shaders
- Too much overdraw! Reduce transparency, reduce highly tessellated meshes (especially those at a distance)
- Optimize your shaders, reduce calculations and texture reads to increase GPU performance

Optimization Techniques

Optimizing does not mean removing content.

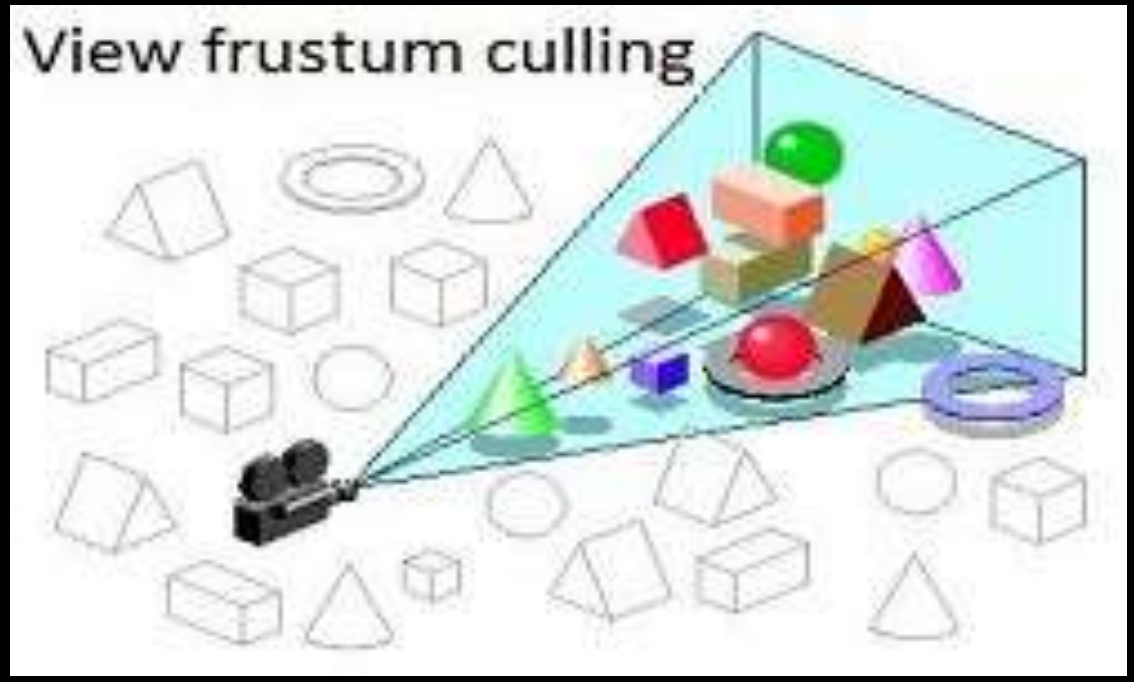
The goal of a technical artist is to optimize without sacrificing visual quality

Work Modularly

- Modularity
  - Objects out of view are culled out

Frustum culling

- **frustum culling** is the process of removing objects that lie completely outside the viewing **frustum** from the rendering process.



Strategically place object to cull

- Occlusion culling – uses the current view of the scene to determine what is hidden behind other objects and then chooses not to render those objects
- Occlusion Markup – decide that some objects are just not rendered at all from a distance
- Backface culling – if an object is completely opaque, surfaces facing away from the camera never need to be drawn

Use LODs and Mip-Maps

- LOD - LOD's reduce the numbers of instructions sent to the renderer for objects farther away.
  - Less instruction = better performance (less overdraw)
  - Less instructions = lower visual quality
- Mip-Maps - Mip-mapping lowers the resolution of the textures sent to the renderer based on distance of the asset

Reduce Draw calls

- Every shader require a separate draw call.
- Draw calls are calls to the graphics card to draw the mesh, reducing them reduces overhead for the GPU
- Too many draw calls will hurt the performance

Reduce draw calls

- Merge textures so several object use the same texture map and material/shader
- Use the same shader across many assets, especially when the assets are far enough away
  - Atmospheric fog can be an excuse for why everything past some distance becomes the same color
- Reduce amount of transparences overlapping

Reduce Overlapping Transparences

- Reduce transparency usage as much as possible
- Render transparency at a lower screen percentage
- Put opaque object under transparent objects
- Sky Box - Only render parts of the sky box that are visible
- Water - Only Render transparency in water where it is shallow enough

Reduce culling distance

- The culling distance of draw distance is the maximum distance an object can be from the camera in order to be drawn by the renderer.
- Polygons beyond the draw the distance are not drawn to screen.
- This distance can be adjusted to improve performance.

Use Proxy Geo

- Proxy geo is low-res geometry that matches the form of the higher res geometry.
- From a distance proxy geo can look identical to the geometry it is trying to represent. Only up close can users tell the difference.
- The textures from the higher res geometry is typically projected onto the proxy geo.
- The proxy geo texture will also be lower.

Use Occlusion Cards

- Geometry that fades out as the camera approaches it.
- Used to prevent geometry from rendering until the player is within a specific distance of that geometry.

Compression

- Use Compressed Textures!!
- Results in faster load times and a smaller memory footprint
- Being able to read memory faster increases rendering performance, limits the amount of texture data transferred when the GPU is rendering
- You can improve perf by adjusting memory allocation. Similarly you can also improve perf by decreasing the amount of required memory of your content (and this can be done with compression)

Make Shaders Cheaper

- Avoid complex mathematical operations such as pow, exp, log, cos, sin, tan.
  - Use lookup tables instead
- Use the built in functions – they are typically optimized for the engine.
- Round floating point values.
- Reduce texture calls.
- Reduce instructions.

Animation Optimization

- Make sure things that can’t be seen (are occluded) aren’t animating.
- Remove skin weights for objects that are far away.
- Only calculate motion for major joints.
  - Ex – at a distance the head still turns and rotates, but the facial joints no longer animate.
- Combine skinned meshes whenever possible.
- Avoid scale animations, they are more expensive than translation and rotation.
- Don’t import the rig with your skeleton.

Performance Checklist

- ❑ Reduce vertex count visible per frame
  - ❑ Tune Occlusion
    - ❑ Culling distance
    - ❑ Impostors
    - ❑ Line of sight
    - ❑ Occlusion cards
    - ❑ Modular geo used
  - ❑ LOD distance objects
    - ❑ Proxy Geo
  - ❑ Reduce object count
    - ❑ Reduce large meshes to multiple smaller meshes
  - ❑ Remove small triangles
  - ❑ Reduce transparency usage
    - ❑ Reduce transparency overlapping
- ❑ Reduce number of different materials per frame
  - ❑ LOD meshes so they all share the same material after some distance
  - ❑ Don’t apply too many shaders to a single mesh
  - ❑Reuse materials as much as possible
- ❑ Bake lighting
  - ❑ Limit dynamic lights
    - ❑ Decrease radius
    - ❑ Reduce shadow casting
- ❑ Use compressed texture formats
  - ❑ Use 16-bit textures over 32-bit
  - ❑ Use Mip Maps so textures sizes reduce at a distance
  - ❑Share channels in textures for multiple uses
- ❑ Reduce shader complexity
  - ❑ Minimize texture calls
  - ❑ Minimize complex mathematical operations
- ❑Optimize Animation
  - ❑LOD joints
  - ❑Remove animation after a certain distance

Resources

- https://www.simplygon.com/knowledge-base/tutorials/why-optimize
- http://www.ericchadwick.com/examples/provost/byf2.html
- http://www.cgmascot.com/design/low-poly-tips-2/
- https://docs.unrealengine.com/latest/INT/Engine/Performance/Guidelines/index.html
- https://docs.unity3d.com/Manual/ModelingOptimizedCharacters.html

Profiling Tools in UE4 and Unity

Performance and Profiling In Unreal Engine 4

- Profile in game directly (editor objects such as the Content Browser add rendering costs)
- Profile in ms not fps
- Disable Vsync - **CONSOLE:** r.Vsync

Performance and Profiling In Unreal Engine 4

CONSOLE: stat unit

The actual **Fame** time is limited by either

Game: CPU game thread

Draw: CPU render thread

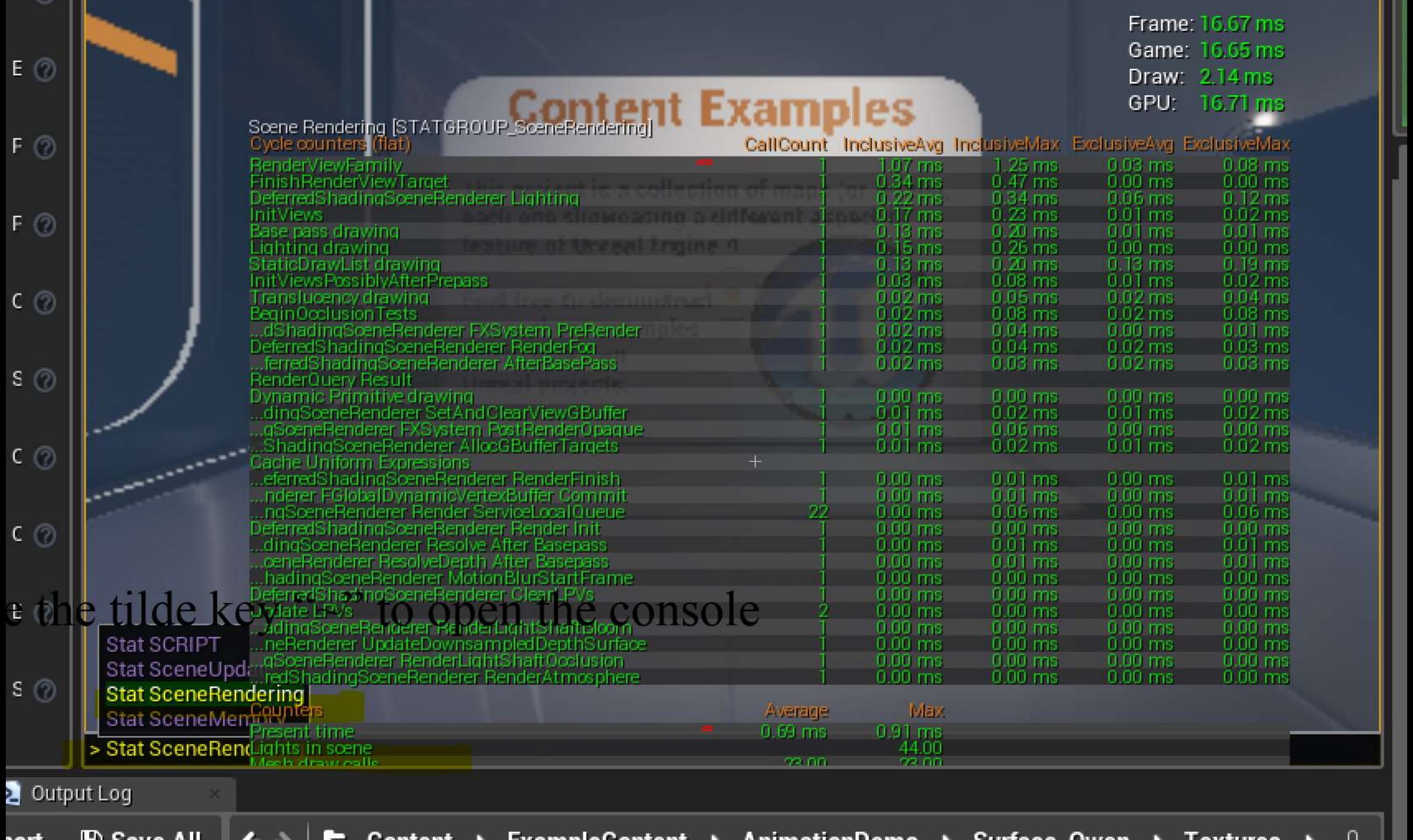
GPU

Which ever of those three is the highest is the limiting factor


ExampleProjectWelcome
File
Edit
Window
Help
Modes
Search Classes
Recently Placed
Basic
Lights
Cinematic
Visual Effects
Geometry
Volumes
All Classes
Save
Source Control
Content
Marketplace
Settings
Blueprints
Cinematics
Build
Pause
Stop
Simulate Stands
demo stands may require you to play or
te the game in the editor to see the
le. This is done with the Play button
in the toolbar, above the viewport.
is now.
Use the tilde key "~" to open the console
Stat UnitGraph Draws simple unit time graph
Stat UNIT Shows hardware unit framerate
> stat unit_

Are you CPU bound?

- **CONSOLE: stat SceneRendering**
- Probably too many draw calls
  - Reduce object count
  - Reduce view distance
  - Reduce material counts
  - Use LOD models that contain smaller num of materials or elements
  - Reduce objects that cast shadow
- **CONSOLE: stat Game**
- Too much game code, or poorly written game code
  - Check for unnecessary loops in blueprints
  - Check for bad raycast geometry
  - Too much physics
  - Too much AI



e the tilde key `~` to open the console

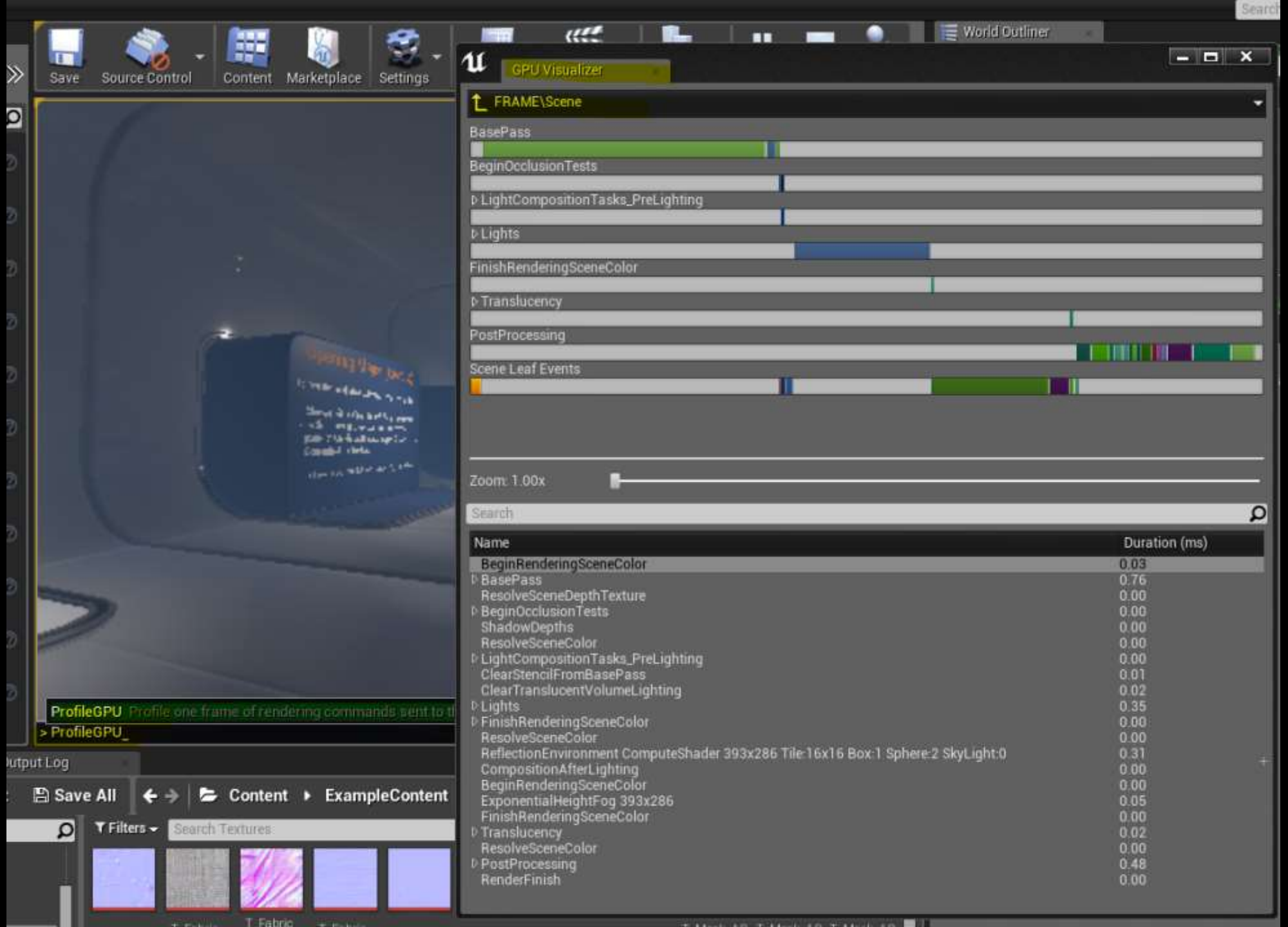

PROFILING WITH AI LOGGING ON!
PROFILING WITH GC VERIFY ON!
Frame: 16.67 ms
Game: 16.65 ms
Draw: 2.13 ms
GPU: 16.66 ms
Game [STATGROUP_game]
Cycle counters (flat)
CallCount InclusiveAvg InclusiveMax ExclusiveAvg ExclusiveMax
World Tick Time 2 0.62 ms 0.80 ms 0.04 ms 0.08 ms
Tick Time 2 0.50 ms 0.68 ms 0.00 ms 0.00 ms
Blueprint Time 8 0.18 ms 0.30 ms 0.01 ms 0.01 ms
PlayerController Tick 1 0.10 ms 0.18 ms 0.00 ms 0.00 ms
Char Movement Total 1 0.06 ms 0.09 ms 0.00 ms 0.00 ms
GC Mark Time
MoveComponent(Primitive) Time 1 0.06 ms 0.10 ms 0.00 ms 0.00 ms
GT Tickable Time 2 0.02 ms 0.07 ms 0.02 ms 0.06 ms
Post Tick Component Update 1 0.02 ms 0.07 ms 0.00 ms 0.00 ms
UpdateOverlaps Time 2 0.02 ms 0.04 ms 0.00 ms 0.01 ms
Update Camera Time 2 0.02 ms 0.04 ms 0.02 ms 0.03 ms
Queue Ticks 1 0.02 ms 0.06 ms 0.02 ms 0.06 ms
Transform or RenderData 1 0.01 ms 0.02 ms 0.00 ms 0.00 ms
Nav Tick Time 2 0.01 ms 0.02 ms 0.01 ms 0.02 ms
Blueprint Latent Actions 8 0.00 ms 0.02 ms 0.00 ms 0.02 ms
MoveComponent(SceneComp) Time
Reset Async Trace Time 1 0.00 ms 0.02 ms 0.00 ms 0.02 ms
TeleportTo Time 1 0.00 ms 0.00 ms 0.00 ms 0.00 ms
GC Sweep Time 1 0.00 ms 0.00 ms 0.00 ms 0.00 ms
Net Tick Time 2 0.00 ms 0.02 ms 0.00 ms 0.02 ms
Finish Async Trace Time 1 0.00 ms 0.00 ms 0.00 ms 0.00 ms
EndScopedMovementUpdate Time 1 0.00 ms 0.00 ms 0.00 ms 0.00 ms
Net Post BC Tick Time 2 0.00 ms 0.00 ms 0.00 ms 0.00 ms
Cooldown Dequeuing 2 0.00 ms 0.00 ms 0.00 ms 0.00 ms
Net Broadcast Tick Time 2 0.00 ms 0.00 ms 0.00 ms 0.00 ms
Recreate 0 0.00 ms 0.06 ms 0.00 ms 0.00 ms
UpdatePhysicsVolume Time
ScheduleCooldowns
Counters Average Max
Ticks Queued 16.20 17.00
TimerManager Heap Size 5.00 5.00
e tilde key ~ to open the console
Stat GAME Displays game performance stats
Stat Game
> stat game_

Are you GPU bound?

- **CONSOLE: ProfileGPU**
- Draw call heavy?
- Materials are too complex?
- Triangle meshes are too dense?
- View distance is too far?

GPU Bottleneck

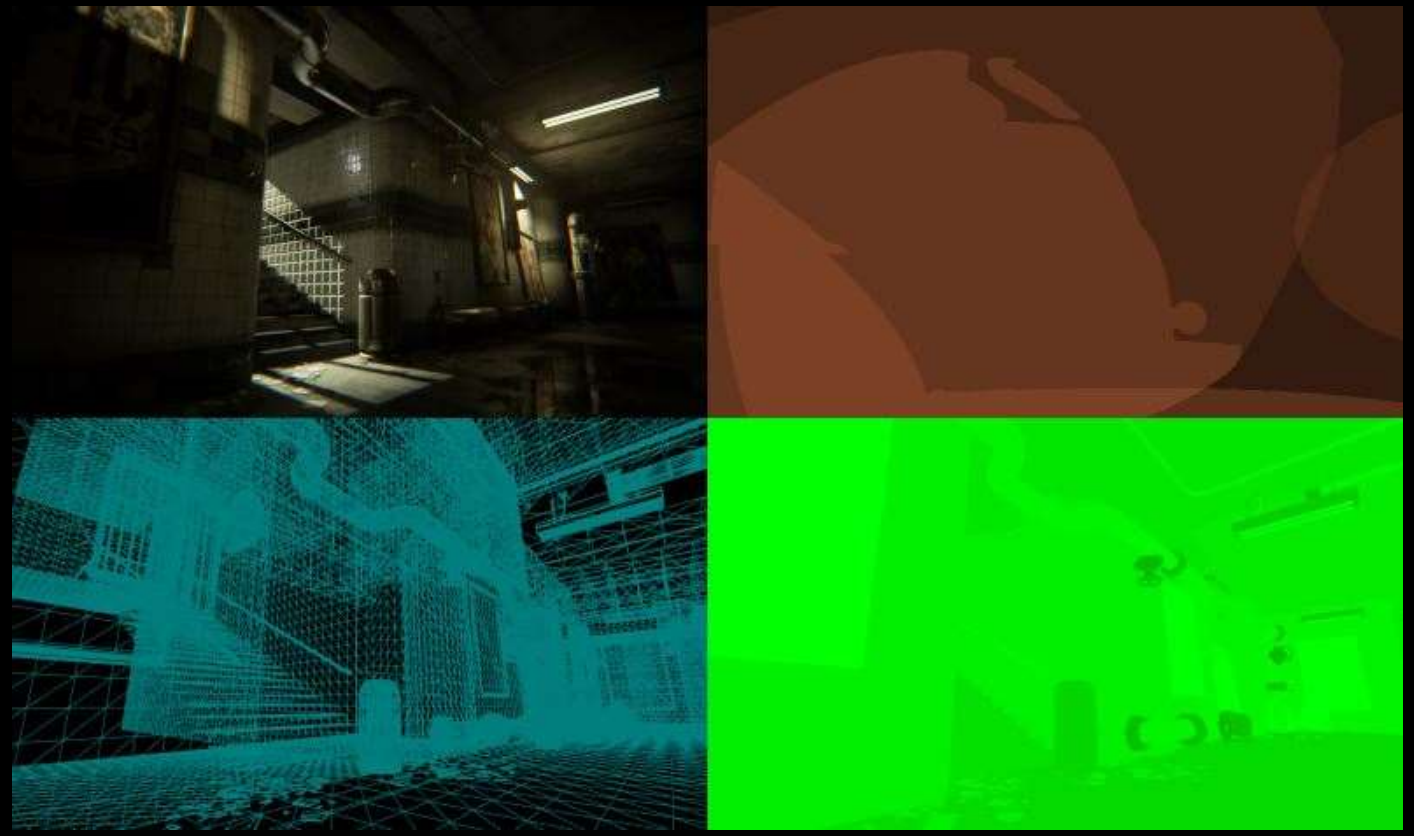
- Is it pixel based? Check by changing r.ScreenPercentage
- Is it lighting based? Check by changing r.Shadow.MaxResolution
- Is it bound by vertex processing? Harder to check for…..

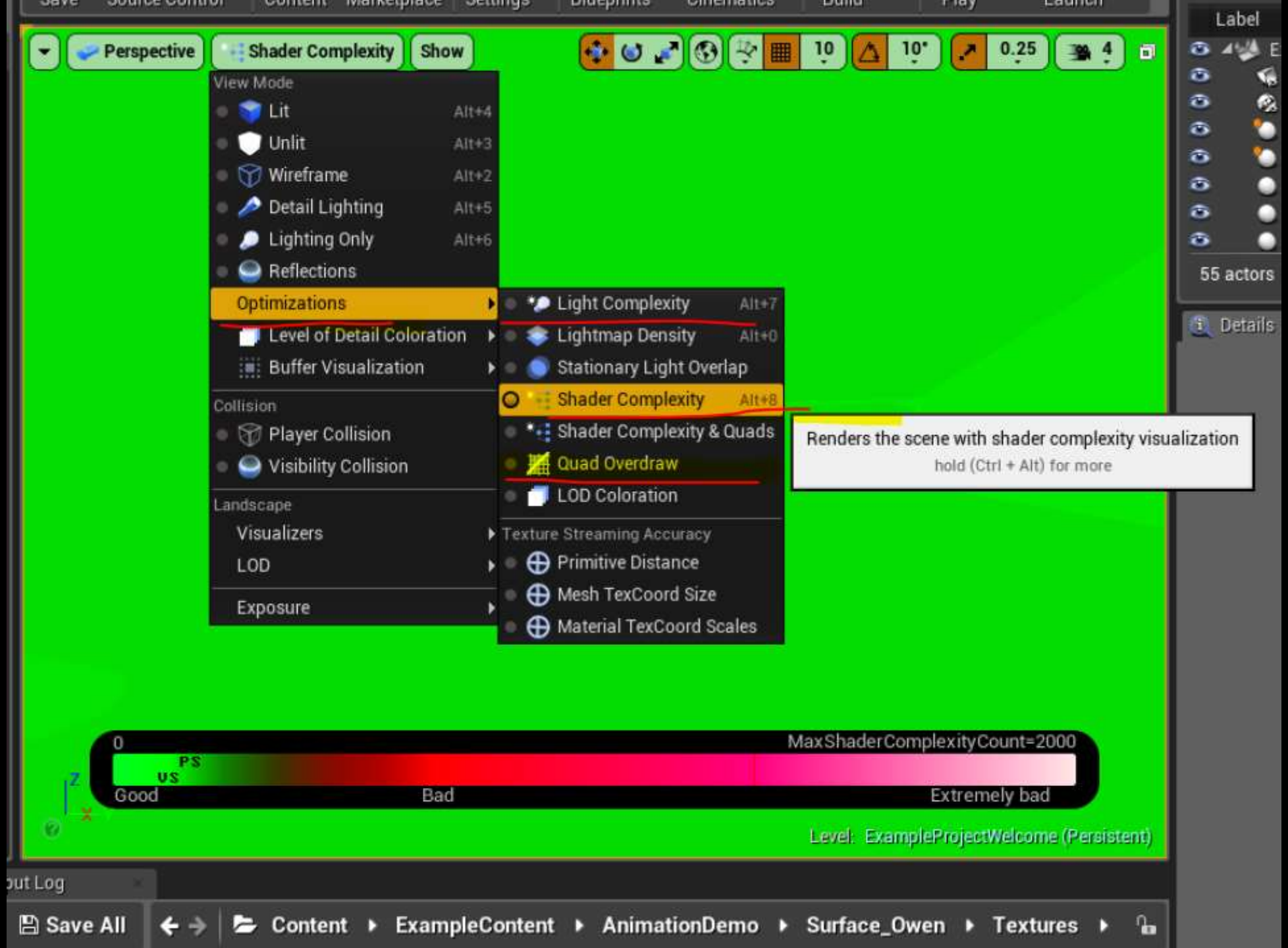

Save
Source Control
Content
Marketplace
Settings
World Outliner
GPU Visualizer
FRAME\Scene
BasePass
BeginOcclusionTests
LightCompositionTasks_PreLighting
Lights
FinishRenderingSceneColor
Translucency
PostProcessing
Scene Leaf Events
Zoom 1.00x
Search
Name
Duration (ms)
BeginRenderingSceneColor 0.03
BasePass 0.76
ResolveSceneDepthTexture 0.00
BeginOcclusionTests 0.00
ShadowDepths 0.00
ResolveSceneColor 0.00
LightCompositionTasks_PreLighting 0.00
ClearStencilFromBasePass 0.01
ClearTranslucentVolumeLighting 0.02
Lights 0.35
FinishRenderingSceneColor 0.00
ResolveSceneColor 0.00
ReflectionEnvironment ComputeShader 393x286 Tile:16x16 Box:1 Sphere:2 SkyLight:0 0.31
CompositionAfterLighting 0.00
BeginRenderingSceneColor 0.00
ExponentialHeightFog 393x286 0.05
FinishRenderingSceneColor 0.00
Translucency 0.02
ResolveSceneColor 0.00
PostProcessing 0.48
RenderFinish 0.00
ProfileGPU Profile one frame of rendering commands sent to t
> ProfileGPU_
Output Log
Save All
Content
ExampleContent
Filters
Search Textures
T_Fabric

Show Flag	Description
ScreenSpaceReflections	Toggles screen space reflections, can cost a lot of performance, only affects pixels up to a certain roughness (adjusted with r.SSR.MaxRoughness or in postprocess settings).
AmbientOcclusion	Screen Space Ambient Occlusion (Some scenes only benfit very little, for static objects you bake AO in lightmass).
AntiAliasing	Toggle any antialiasing (TemporalAA and FXAA), use TemporalAA to switch to FXAA (faster, lower quality).
Bloom	Affects image based lens flares and the bloom feature.
DeferredLighting	Toggle all deferred lighting passes.
DirectionalLights PointLights SpotLights	Toggles the different light types (useful to see what kind of light type is having what effect and performance impact.
DynamicShadows	Toggles all dynamic shadows (shadowmap rendering and shadow filtering/projection).
GlobalIllumination	Toggles baked and dynamic indirect lighting (LPV).
LightFunctions	Toggles the light functions rendering.
PostProcessing	Toggles all post-processing passes.
ReflectionEnvironment	Toggles reflection environment passes.
Refraction	Toggles the refraction pass.
Rendering	Toggles rendering altogether.
Decals	Toggle decal rendering.
Landscape Brushes StaticMeshes SkeletalMeshes Landscape	Toggles what geometry is rendered.
Translucency	Toggles translucency rendering.
Tessellation	Toggles tessellation (still runs tessellation shaders but spawning more triangles).
IndirectLightingCache	Toggles if dynamic objects or static objects that have invalidated lightmaps are using the Indirect Lighting Cache.
Bounds	Show the bounding volume of the objects selected in the editor.
Visualize SSR	Renders all pixels affected by screen space reflections is bright orange (slow), see below.

View Modes

- *Lit, LightCompexity (darker is better), Wireframe, Shader Complexity (green is good)*

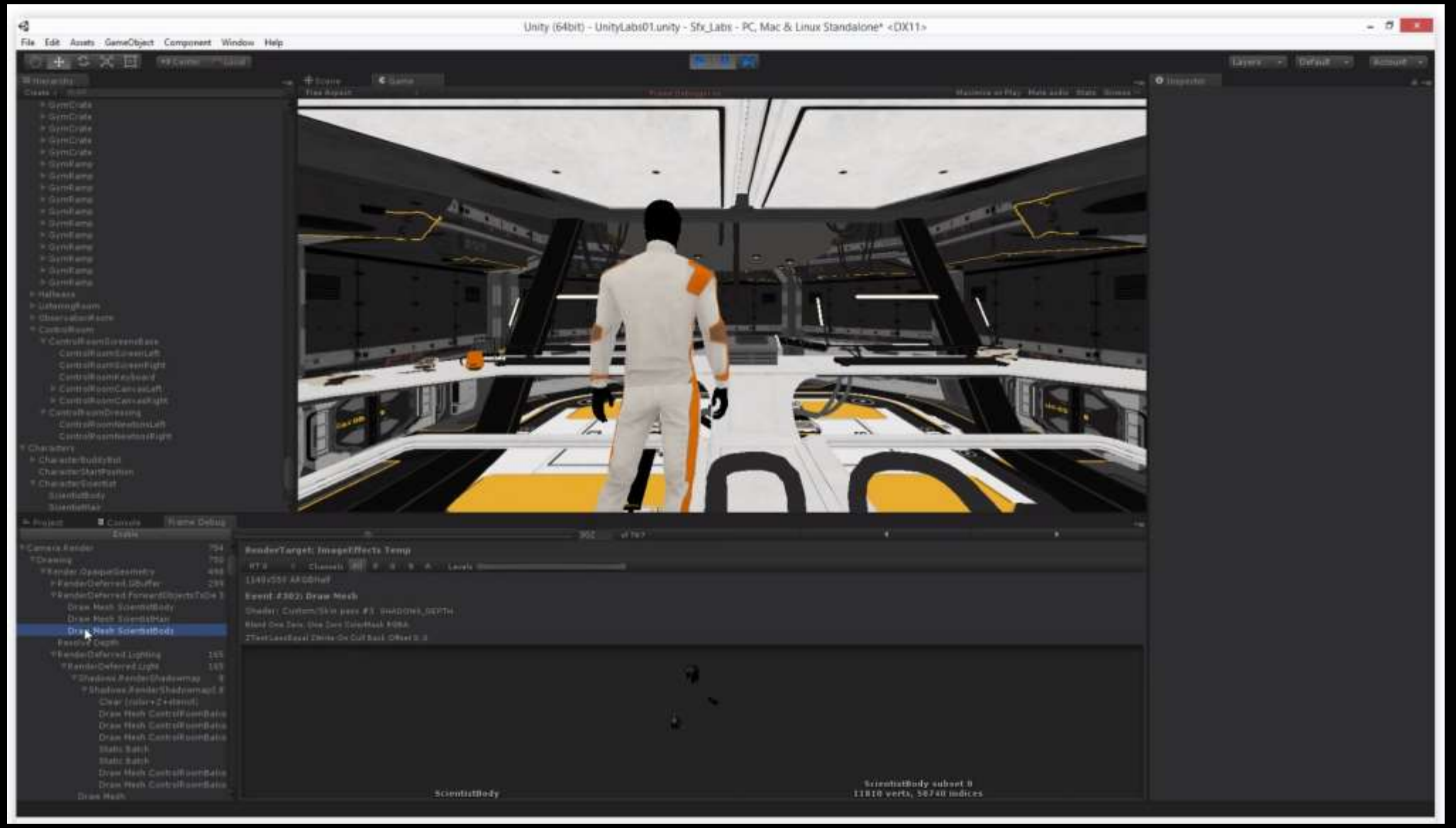


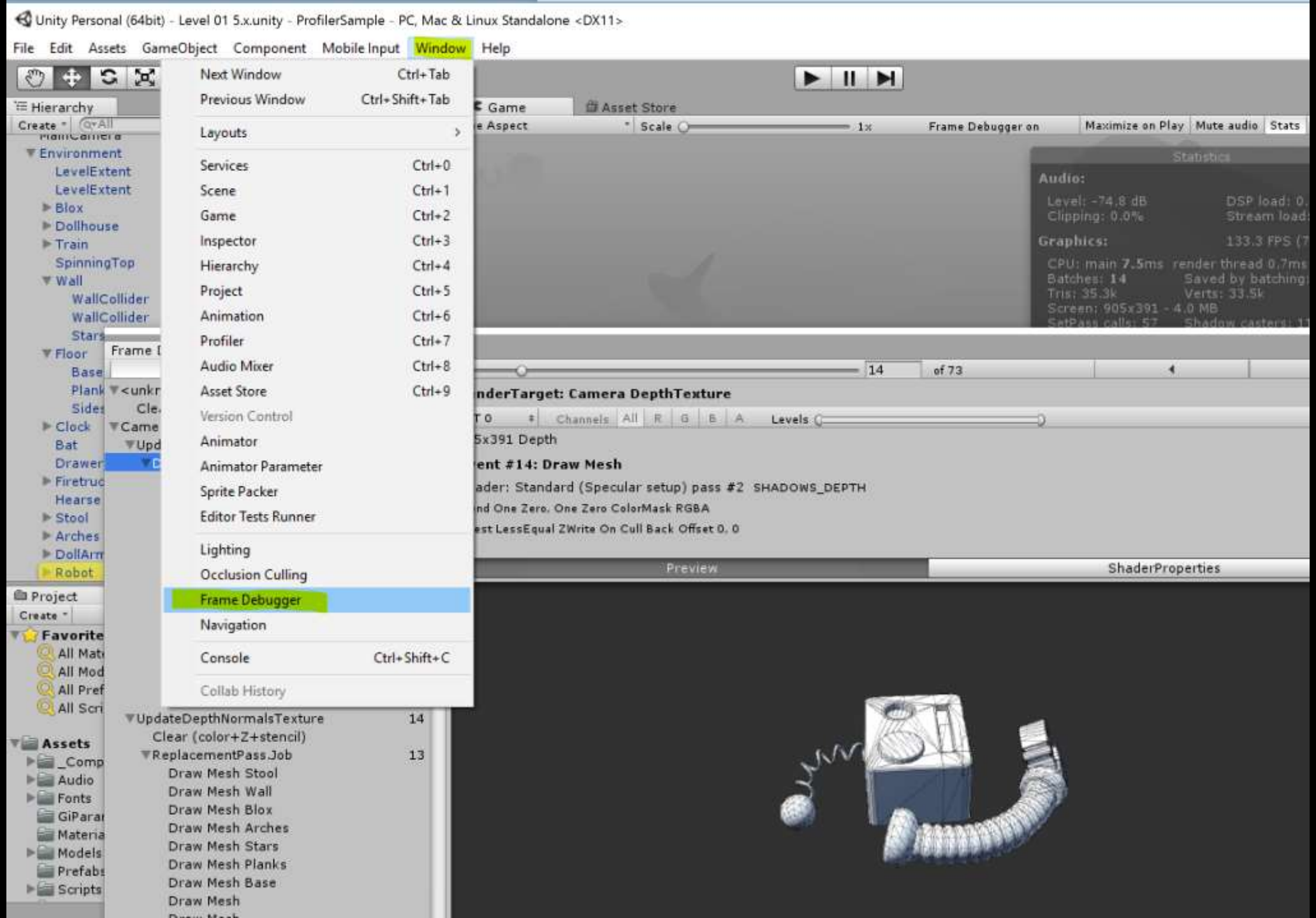

Perspective
Shader Complexity
Show
10
10°
0.25
4
View Mode
Lit Alt+4
Unlit Alt+3
Wireframe Alt+2
Detail Lighting Alt+5
Lighting Only Alt+6
Reflections
Optimizations
Level of Detail Coloration
Buffer Visualization
Collision
Player Collision
Visibility Collision
Landscape
Visualizers
LOD
Exposure
Light Complexity Alt+7
Lightmap Density Alt+0
Stationary Light Overlap
Shader Complexity Alt+8
Shader Complexity & Quads
Quad Overdraw
LOD Coloration
Texture Streaming Accuracy
Primitive Distance
Mesh TexCoord Size
Material TexCoord Scales
Renders the scene with shader complexity visualization
hold (Ctrl + Alt) for more
0
MaxShaderComplexityCount=2000
PS
VS
Good
Bad
Extremely bad
Level: ExampleProjectWelcome (Persistent)
Label
55 actors
Details
Save All
Content
ExampleContent
AnimationDemo
Surface_Owen
Textures

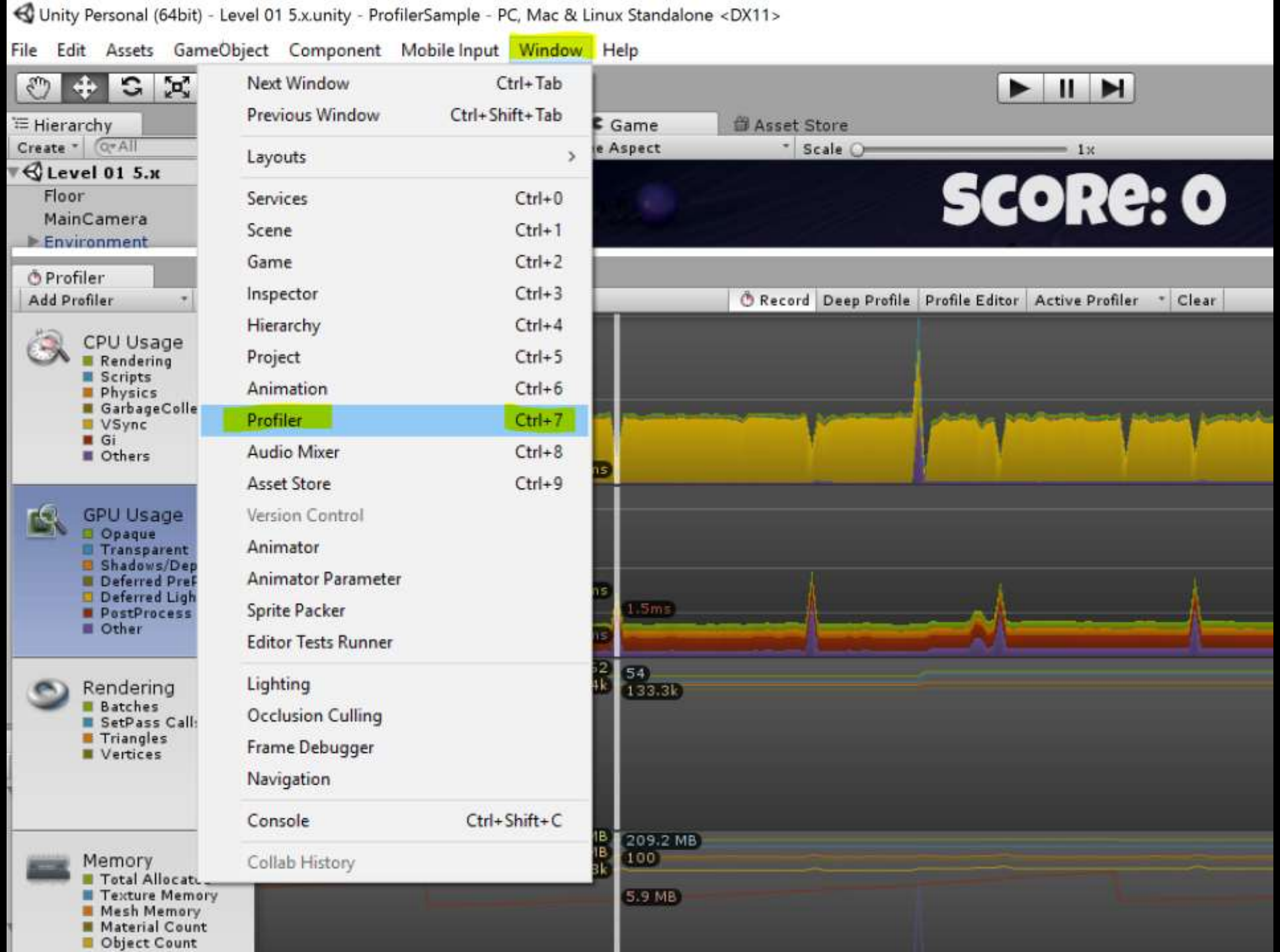
Performance and Profiling in Unity

Performance and Profiling in Unity

- Use the Frame Debugger

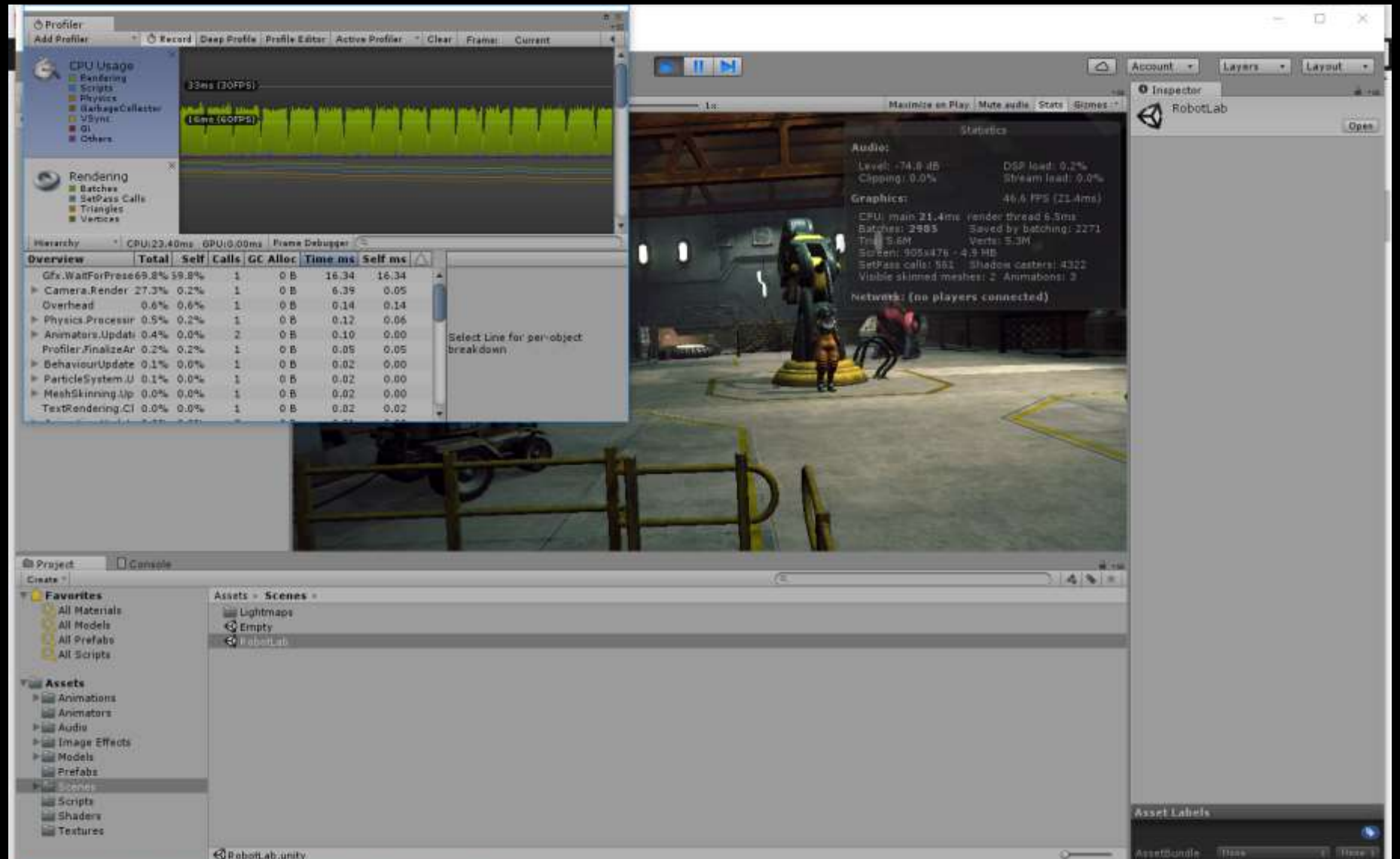



Unity Personal (64bit) - Level 01 5.x.unity - ProfilerSample - PC, Mac & Linux Standalone <DX11>
File Edit Assets GameObject Component Mobile Input Window Help
Next Window Ctrl+Tab
Previous Window Ctrl+Shift+Tab
Layouts
Services Ctrl+0
Scene Ctrl+1
Game Ctrl+2
Inspector Ctrl+3
Hierarchy Ctrl+4
Project Ctrl+5
Animation Ctrl+6
Profiler Ctrl+7
Audio Mixer Ctrl+8
Asset Store Ctrl+9
Version Control
Animator
Animator Parameter
Sprite Packer
Editor Tests Runner
Lighting
Occlusion Culling
Frame Debugger
Navigation
Console Ctrl+Shift+C
Collab History
Hierarchy
Game
Asset Store
Frame Debugger on
Maximize on Play
Mute audio
Stats
Statistics
Audio:
Graphics:
Preview
ShaderProperties
UpdateDepthNormalsTexture 14
Clear (color+Z+stencil)
ReplacementPass.Job 13
Draw Mesh Stool
Draw Mesh Wall
Draw Mesh Blox
Draw Mesh Arches
Draw Mesh Stars
Draw Mesh Planks
Draw Mesh Base
Draw Mesh


Unity Personal (64bit) - Level 01 5.x.unity - ProfilerSample - PC, Mac & Linux Standalone <DX11>
File Edit Assets GameObject Component Mobile Input Window Help
Next Window Ctrl+Tab
Previous Window Ctrl+Shift+Tab
Layouts
Services Ctrl+0
Scene Ctrl+1
Game Ctrl+2
Inspector Ctrl+3
Hierarchy Ctrl+4
Project Ctrl+5
Animation Ctrl+6
Profiler Ctrl+7
Audio Mixer Ctrl+8
Asset Store Ctrl+9
Version Control
Animator
Animator Parameter
Sprite Packer
Editor Tests Runner
Lighting
Occlusion Culling
Frame Debugger
Navigation
Console Ctrl+Shift+C
Collab History
Hierarchy
Create
Level 01 5.x
Floor
MainCamera
Environment
Game
Asset Store
Scale
1x
SCORE: 0
Profiler
Add Profiler
Record
Deep Profile
Profile Editor
Active Profiler
Clear
CPU Usage
Rendering
Scripts
Physics
VSync
Others
GPU Usage
Opaque
Transparent
Other
Rendering
Batches
Triangles
Vertices
Memory
Texture Memory
Mesh Memory
Material Count
Object Count
1.5ms
54
133.3k
209.2 MB
100
5.9 MB

Performance and Profiling in Unity

- Use the **Profiler**

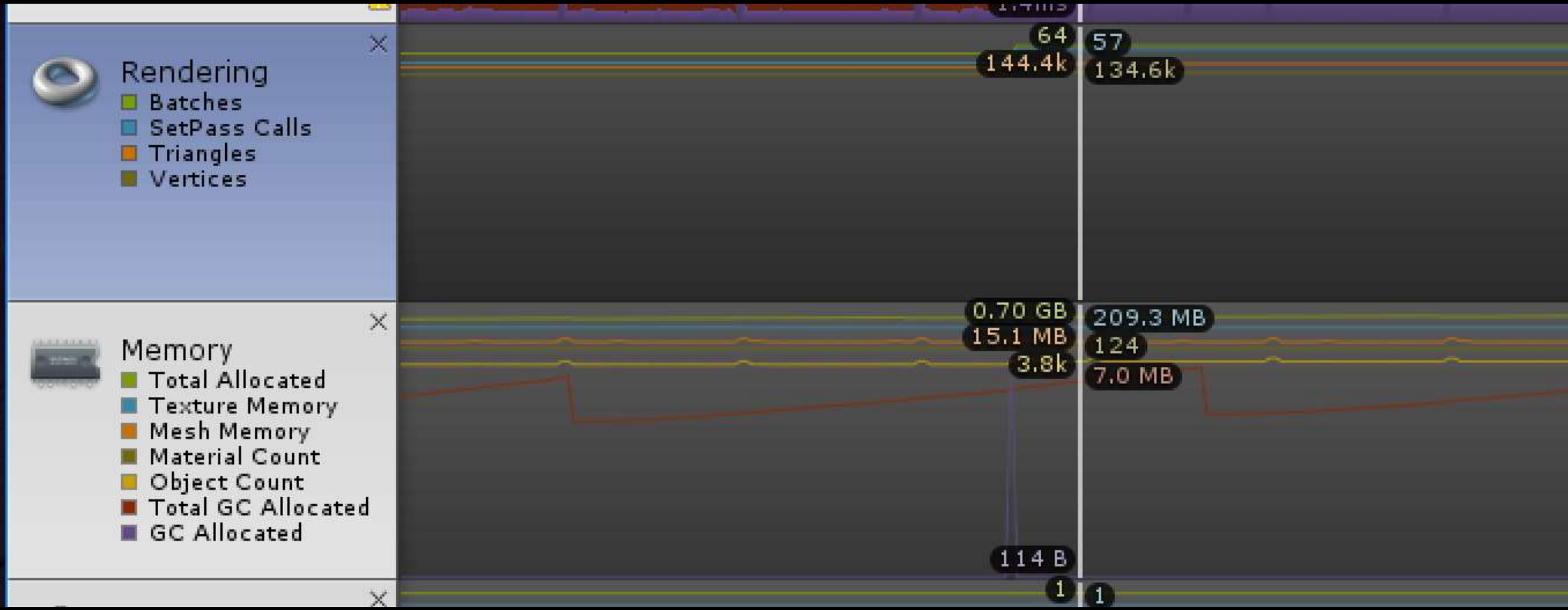


The Profiler

- Gfx.WaitForPressed…. Means GPU bound, CPU is done waiting for GPU
- If both GPU and CPU are waiting at any point it mean vsync is on and the frames are rendering faster then the refresh rate

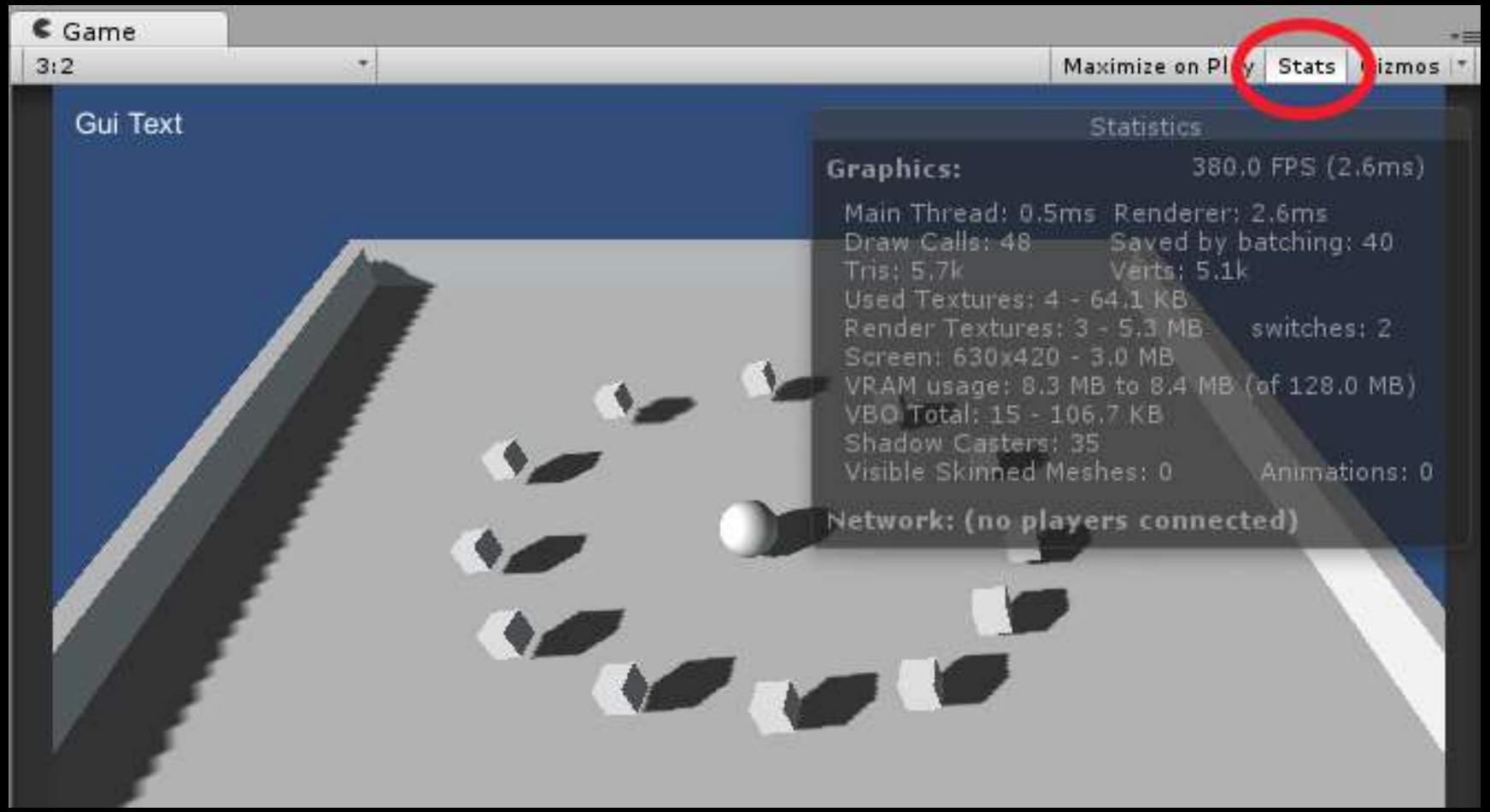
Additional Profiling Details

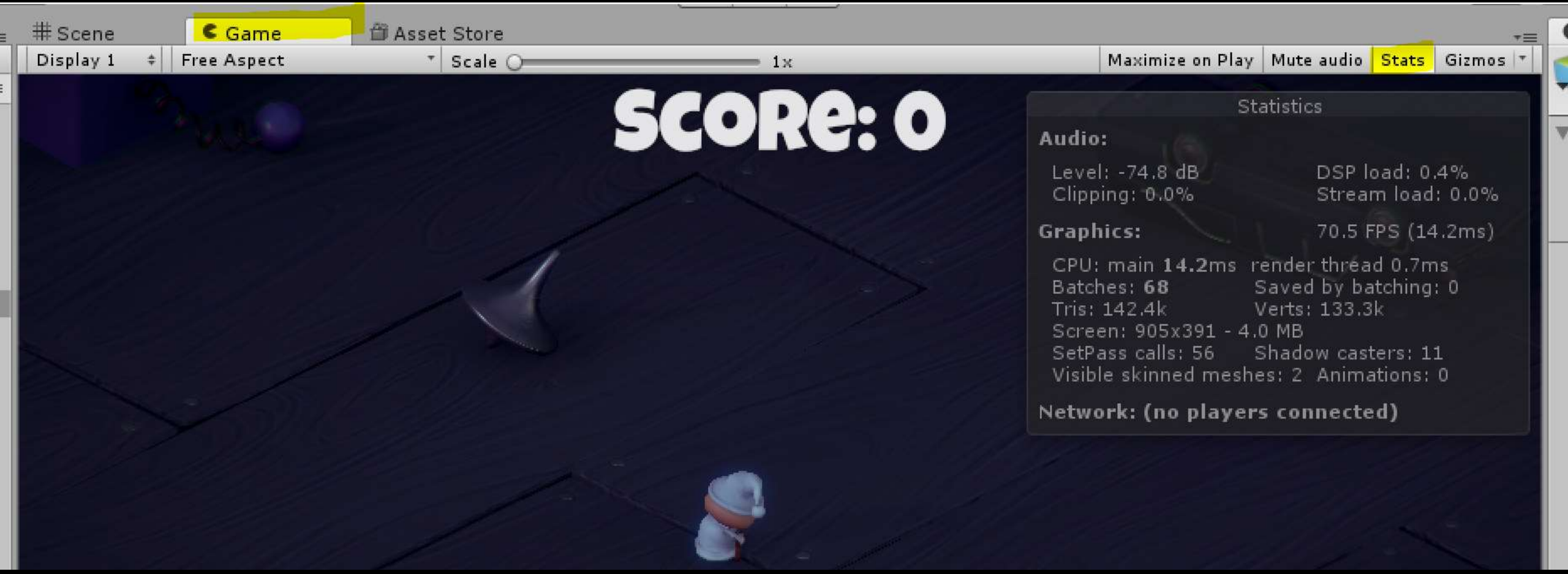
- **Rendering Profiler** – where to see the current triangles and vertices count for the rendered frame
- **Memory Profiler –** shows statistics for common asset types such as textures and meshes


Rendering
Batches
SetPass Calls
Triangles
Vertices
64
57
144.4k
134.6k
Memory
Total Allocated
Texture Memory
Mesh Memory
Material Count
Object Count
Total GC Allocated
GC Allocated
0.70 GB
209.3 MB
15.1 MB
124
3.8k
7.0 MB
114 B
1
1

Performance and Profiling in Unity

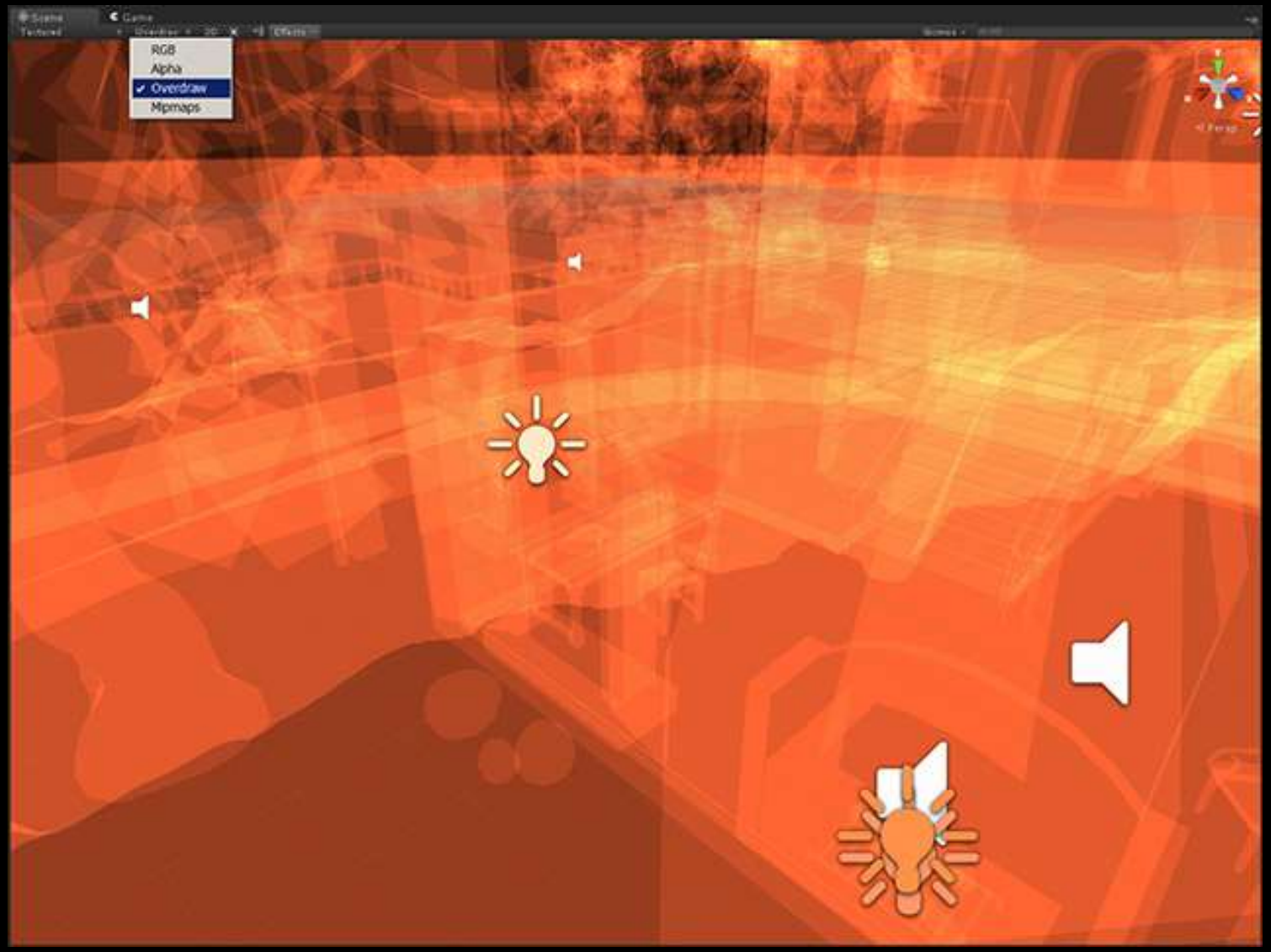
- Use the **Rendering Stats**

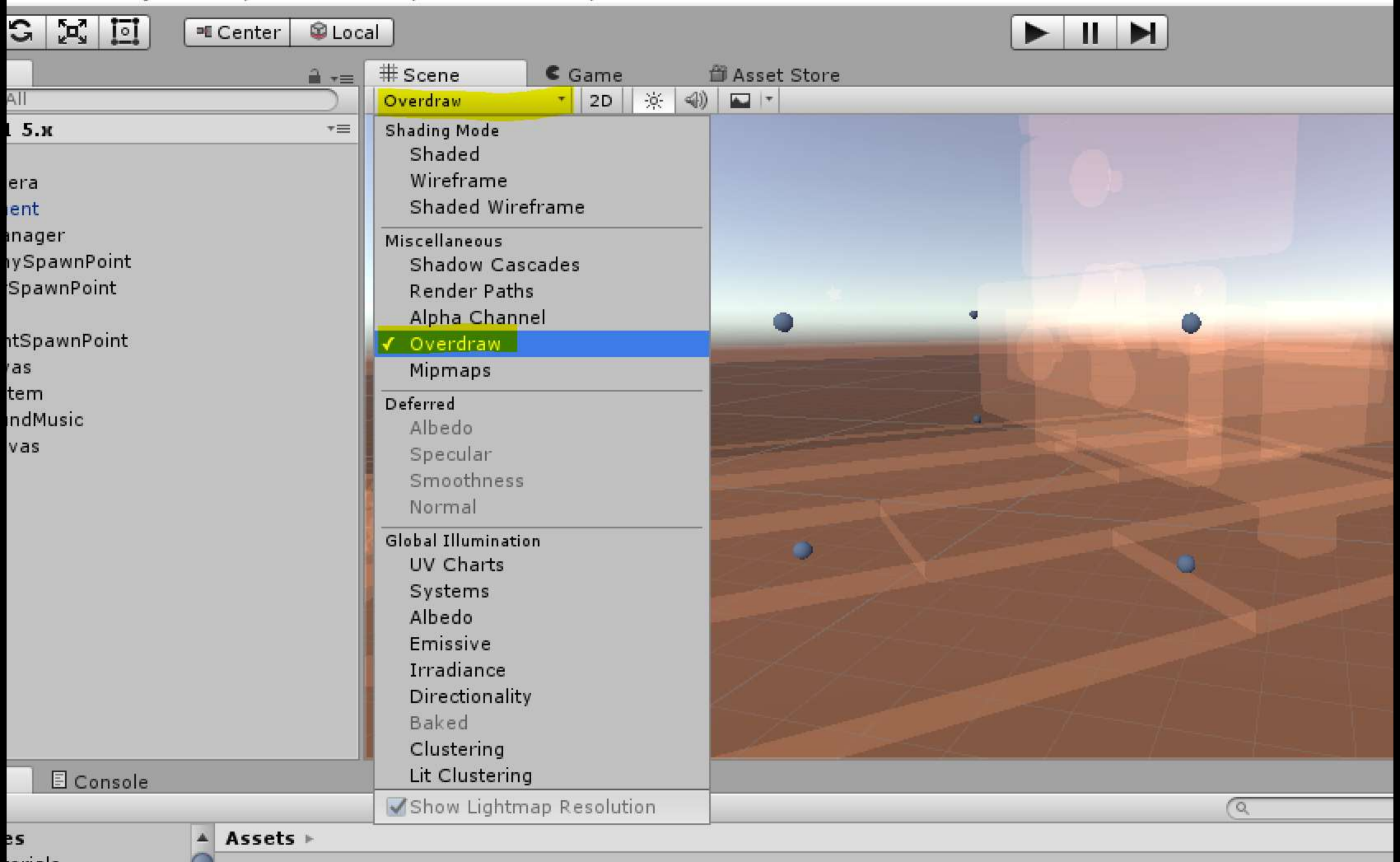



Scene
Game
Asset Store
Display 1
Free Aspect
Scale
1x
Maximize on Play
Mute audio
Stats
Gizmos
SCORE: 0
Statistics
Audio:
Level: -74.8 dB
DSP load: 0.4%
Clipping: 0.0%
Stream load: 0.0%
Graphics:
70.5 FPS (14.2ms)
CPU: main 14.2ms render thread 0.7ms
Batches: 68
Saved by batching: 0
Tris: 142.4k
Verts: 133.3k
Screen: 905x391 - 4.0 MB
SetPass calls: 56
Shadow casters: 11
Visible skinned meshes: 2
Animations: 0
Network: (no players connected)

Performance and Profiling in Unity

- Use the **GPU Overdraw** - brighter colors indicate more overdraw




Center
Local
Scene
Game
Asset Store
Overdraw
2D
Shading Mode
Shaded
Wireframe
Shaded Wireframe
Miscellaneous
Shadow Cascades
Render Paths
Alpha Channel
Overdraw
Mipmaps
Deferred
Albedo
Specular
Smoothness
Normal
Global Illumination
UV Charts
Systems
Albedo
Emissive
Irradiance
Directionality
Baked
Clustering
Lit Clustering
Show Lightmap Resolution
Console
Assets