



Currently at Funomena, recently shipped Luna: Moondust Garden for Magic Leap Also part of a small 3 person dev team as Polymath

Previously at Apple doing things

Before that on the prototyping and demos team at Meta AR

Before that spent a few years at intel Prototyping for experimental hardware

Spent a lot of time trying to get things to run faster on things that are underpowered, and overheated.

Not many people can say they've written a shader that made hardware actually explode.



As a tech Artist, particularly as a new tech artist, optimization can sound pretty scary There is a lot of information (and misinformation) out there at every level

- from outdated info
- hunches and superstition
- just plain crazy theories (Reddit, Forums)

# Scope of this Talk

#### NOT about

- Low Level GPU optimization
- Lots of rendering math / counting every instruction
- A tutorial for one of 50 profilers

### Is about

- An approachable intro to graphics optimization
- Making shaders and post effects run faster
- Becoming more aware of your various resources
- Basic Tips and Tricks

I'm not smart enough to teach anyone about deep shader optimization

- This is for entry level to intermediate Tech artists
- Who want an intro or a survey of optimization basic

# **Know Your Enemy**

Understand the limitations of your target hardware

- PC
  - Ballpark your min spec
  - Will you have multiple quality settings?
  - Test on Nvidia, AMD, and Intel GPUs
- Console + Mobile
  - Specific hardware means specific debuggers
  - Emulation will only get you so far



#### PC

- If you're simultaneously releasing on multiple platforms, test on them early. Maybe have a dedicated toaster machine or a few crappy android tablets with various releases. You don't have to build to these all the time, but a sanity check might save your bacon.
- If you have dedicated time and people to port later, don't leave them a huge mess (relying heavily on unsupported features, weird unreadable over-optimization)
- Are you targeting toasters or aircraft carriers?
- What color is the gpu in your dev machine? Are you testing on red and blue ones too?

#### Console

- Mention Playstation, Xbox, Switch equivalents

## Mobile

- Be constantly testing. Lots of things just behave differently or are unsupported on Android.



Jokingly exaggerate separation of the gpu vs cpu "I write all my code on the gpu, I ripped out my cpu years ago" etc

Reiterate our focus on GPU profiling. Suggest working closely with an engineer familiar with cpu Profiling.

# **CPU or GPU**

Here we're focusing on the GPU, though in production it might not be so easy to talk about them separately.

- Try to partner with an engineer for CPU optimization
- Identify CPU waits in the GPU profiler
- Not an exact science



- Kind of a "you got your peanut butter in my chocolate" situation



Lots of profilers and tools I wont mention A whole host of AMD and Intel tools Various degrees of support and usefulness We're going to survey A range of tools from General -> Specific



First up is Renderdoc My favorite for desktop profiling Unity will detect your Renderdoc install! Unreal has a native plugin that can be enabled!



Renderdoc capture of an unreal profiling frame

- Inspecting steps of the deferred rendering pipeline
- We'll return to this later

## Nvidia Tools https://developer.nvidia.com/tools-overview



- Nsight Graphics
  - Many Nsight series tools (Systems, Compute, VS/Eclipse integration)
- Tegra Debugger
  - Specific debug bridge provided by platform partner
- nvProf
  - <u>https://devblogs.nvidia.com/cuda-pro-tip-nvprof-your-handy-universal-gpu-profiler/</u>
  - Command-line profiler, CUDA support

Most time spent with the tegra debugger

There are just as many AMD tools, as well as PIX, and various Console specific tooling used with dev kits.

# **Unity Optimization**

- Frame Debugger
  - <u>https://docs.unity3d.com/Manual/FrameDebugger.html</u>
- GPU Profiler
  - <u>https://docs.unity3d.com/Manual/ProfilerGPU.html</u>
- RenderDoc Integration
- #IFs and Pragmas in shaders
- New LWRP and SRP
  - Still experimental, worth looking at

- Remember all in-engine profiling may be affected by editor overhead



Frame Debugger

- Has its uses nice quick overview of what's happening
- Pretty sparse



This is what you see by default, a generalized view

A Profiler								
Add Profiler - 🔹 Record Deep Profile Profile E	ditor Editor	- Allocation Calls	tacks - Clear on Pl	lav Clear Load Save Fra	me: 189	59/19906	►	*= Current
Copaue C			kauturt, kana					
Hierarchy - CPU:11.36ms GPU:10.74								
				RenderTexture.SetActive			0.02	21.78
				RenderTexture.ResolveA			0.00	1.32
BloomOptimizedWithVignette.OnRenderIma	3.6%	26	0.388					
Graphics.Blit	3.6%	26	0.388					
RenderTexture.ResolveAA	1.0%		0.116					
GC Alloc	0.0%		0.000	Called From	Calls	ALC ALLOC	- nine ins	- 11 70
RenderTexture SetActive	0.4%		0.048	BioomOptimizedWithVign			0.08	41.73
▶ Drawing	0.4%		0.045					
Camera.FireOnPreRender()	0.0%							
LineRenderer.ScheduleGeometryJobs	0.0%		0.000					
		0		filor C				
			ny pro	mer – C	σPU			e
	0.0%		0.00				-	

But you can get fairly specific. The GPU Usage tab is closed by default, open that thing up!

- Here we see a massive chunk of our rendering is our bloom effect. Even with a black frame, screen space effects cost the same.

# **Unreal Optimization**

- Console Commands
  - r.ProfileGPU, r.SetRes, r.ScreenPercentage, r.ABunchOfOtherStuff
  - Shortcut: "Ctrl+Shift+," to show the GPU profiler
  - While running, "stat GPU" to show Live GPU Profiler
- Material Optimization + Feature branching
- Optimization View Modes
- RenderDoc Plugin

- Remember all in-engine profilers are affected by editor overhead

# **Material Stats**

🔊 Stats 🛛 🛛 🔀 Platform Stats

- Base pass shader: 100 instructions
- Base pass shader with volumente Eightmap. The instructions
- Texture samplers: 3/16

- Shows Instruction Count
   Vertex Instructions
- Texture Samples

- All engine profilers are affected by editor overhead
- We'll get to platform stats in a minute



- All engine profilers are affected by editor overhead

Platf	orm Sta	Its					
<ul> <li>Need pla</li> <li>Go grab</li> </ul>	tform specific the Android Mali C	compiler					
≥ Stats ×	Platform Stats ×						
≥ Stats	2 Platform Stats ×						
Stats Settings -	Platform Stats ×	DirectV SM5	DirectV SME	Android CLES 2 C	Android CLES 2	Android CLES 2 1	1 Android CLES 21
2 Stats	Platform Stats ×	DirectX SM5 High	DirectX SM5	Android GLES 2.0	Android GLES 2.	C Android GLES 3.1	1 Android GLES 3.1
Stats Stats Settings - Pixel Shader	Platform Stats ×           Dynamically lit object	DirectX SM5 High 173	DirectX SM5 Low	Android GLES 2.0 High 53	Android GLES 2.0 Low 53	C Android GLES 3.1 High 52	1 Android GLES 3.1 Low 52
Stats ISettings ▼ Pixel Shader	Platform Stats ×  Dynamically lit object	DirectX SM5 High 173	DirectX SM5 Low 173	Android GLES 2.0 High 53	Android GLES 2.0 Low 53	C Android GLES 3.1 High 52	1 Android GLES 3.1 Low 52
€ Stats Settings → Pixel Shader Vertex Shader	Platform Stats       Dynamically lit object Static Mesh	DirectX SM5 High 173 32	DirectX SM5 Low 173 32	Android GLES 2.0 High 53 29	Android GLES 2.0 Low 53 29	C Android GLES 3.1 High 52 25	1 Android GLES 3.1 Low 52 25
Stats -	Platform Stats      Dynamically lit object Static Mesh	DirectX SM5 High 173 32	DirectX SM5 Low 173 32	Android GLES 2.0 High 53 29	Android GLES 2.1 Low 53 29	2 Android GLES 3.1 High 52 25	1 Android GLES 3.1 Low 52 25
2 Stats Settings - Pixel Shader Vertex Shader Samplers	Platform Stats       Dynamically lit object Static Mesh	DirectX SM5 High 173 32 3/16	DirectX SM5 Low 173 32 3/16	Android GLES 2.C High 53 29 2/8	Android GLES 2.4 Low 53 29 2/8	C Android GLES 3.1 High 52 25 2/16	1 Android GLES 3.1 Low 52 25 2/16
Stats  Settings  Pixel Shader  Vertex Shader  Samplers  Texture Lookups (Est.)	Platform Stats ×  Dynamically lit object Static Mesh	DirectX SM5 High 173 32 3/16 VS(0), PS(0)	DirectX SM5 Low 173 32 3/16 VS(0), PS(0)	Android GLES 2.0 High 53 29 2/8 VS(0), PS(0)	Android GLES 2.1 Low 53 29 2/8 VS(0), PS(0)	: Android GLES 3.1 High 52 25 2/16 VS(0), PS(0)	1 Android GLES 3.1 Low 52 25 2/16 VS(0), PS(0)
Stats  Settings  Pixel Shader  Vertex Shader  Samplers  Texture Lookups (Est.) Interpolators	Platform Stats     >      Dynamically lit object Static Mesh	DirectX SM5 High 173 32 3/16 VS(0), PS(0) 0/0	DirectX SM5 Low 173 32 3/16 V S(0), PS(0) 0/0	Android GLES 2.0 High 53 29 2/8 VS(0), PS(0) 0/0	Android GLES 2.1 Low 53 29 2/8 VS(0), PS(0) 0/0	C Android GLES 3.1 High 52 25 2/16 VS(0), PS(0) 0/0	1 Android GLES 3.1 Low 52 25 2/16 VS(0), PS(0) 0/0

- You'll need a platform specific shader copiler to get a preview here.

\_



- If this is useful to you, you probably want to be rooting around in the render pipeline source code anyway
- There's some great write-up's of extending the deferred shader online that I'll link at the end

# <section-header><section-header><complex-block><complex-block>

There are a large number of options here, including visualizing specific gbuffers, to lots of other super useful visualization modes.

- Lighting
- Texel density



Many different mindsets here, I'm just sharing my experiences

# **Pre-Pro Optimization**

- Look for content and performance guidelines for the platform
  - Find out what you're in for
- Predicting Bottlenecks is Difficult
  - Especially on new hardware, a new pipeline, or a new project
- Simple tests can influence art direction and design constraints
  - Custom lighting or Look Dev
  - Custom pipeline or tooling
- Might be wildly off despite best intentions



What can you do to start thinking about optimization in preproduction effectively?

**Platform guidelines** 

- what other projects are doing
- In our case it was a Tegra. Pretty decent, but not magical for VR / AR.

#### Lighting + lookdev

- Maybe you find out you need to use stylized lighting, or no lighting
- Maybe shadows are off limits
- Maybe you find you have room to do something unique

## Pipeline + tooling

- Seeing those needs early can help set your priorities for tools
- Atlasing, batching, baking, polycount, shader complexity

# **Production Optimization**

- Pre-emptive optimization lengthens iteration time
  - Everything takes longer
- Features/Effects will be vital for different projects at different times
  - · Communicating a core gameplay idea might be worth the cost
  - That one system you spent 2 weeks optimizing might go in the trash
- Spotting Pitfalls and Practicing Good habits
  - Knowing when to Refactor and when to wait
  - Legibility and Comments > Micro Optimization
  - Periodic auditing of art resources is a good call
  - If work is done preemptively and cut later, you played yourself



Easier with experience - You'll spot your own bad habits over time

Trying to plan too far ahead can be very counterproductive Much of the work happens, well, during the project

# **Deadline Optimization**

- The project needs are definitely clearer
  - You know the size of the bag (90fps, must fit on Game & Watch)
  - Your to-do list + priorities are obvious
- Collect all the low hanging fruit
  - Texture size
  - Particle Overdraw
  - Lighting and Shadow Settings
  - Neuter Post Effects (4x MSAA to 2x MSAA)
  - Uncheck the MakeGameSlower boolean
- You know how much time is left
  - Makes it easier to throw things in the bin
  - You are hopefully "feature complete"



Go through Lighting and Shadow Settings Double check everything you can Audit everything you can

# Uh-Oh

- When to raise a red flag
  - your producer or lead has a higher level view by design
  - If something needs to be bubbled up early, don't be afraid to do so
  - it's better to have a feature die earlier than later





Need to add oculus flowchart for troubleshooting here, "when in doubt..." the black box method

# **Process + Prioritizing**

- Understand the Problem
  - Identify problem areas using the tools
  - Form a hypothesis and attempt to verify
  - Repeat
- Understand the Fix
  - Remove the feature and look at the delta • The fix will never be faster than that
  - Is it an easy fix, or a tough fix?
  - Is it worth fixing?
  - Is it okay to just cut the feature?
  - Is it more or less important than your other heavy hitters?



# **Effects as UI**

- Effects communicate core gameplay concepts
  - Success vs Failure
  - Valid vs Invalid (actions, ranges, locations)
  - Inventory, counts, and amounts
  - Cooldowns and timing
- Modifying these effects can change player perception
  - This should be an ongoing conversation
  - Can be dangerous late in production
  - Artists, Animators, Designers, and Producers all bring insight to the table



If this looks very familiar to you, good. It's a great chart. This graph is recreated from an oculus presentation – It describes a process

# **Utilization: Vertex or Pixel shader?**

- Where are your biggest hits?
  - Find out if any of them are dumb mistakes
  - Look for patterns
- Fluctuations over the app lifetime
  - Actors or particles spawn, different levels
  - Are you looking at a particularly expensive effect?
- Can you shuffle work around to even the load?
  - Moving instructions from pixel to vertex shader
  - Computing things offline instead of runtime
    - Baking textures instead of using procedurals
    - Baking lighting, using vertex color instead of ID masks



Spoiler Alert: It's almost always the pixel shader

# Tips + Tricks

- Shader Branching, Loops, Expensive math
- Overdraw
  - Know your blend modes and passes
- Drawcalls
  - Instancing, batching
- Texture Samples
- Atlasing, ID Maps
  Channel Packing and Precision Crimes
- Utilization
  - It's probably the pixel shader
- Hardware Weirdness
  - Tiled Rendering paths on mobile, Android shenanigans



<ul> <li>Channel Packing</li> <li>Substance has a great interface for this</li> <li>Be aware of various compression profiles</li> </ul>	Tips + T	ricks		
Current preset: GAME (Paded) Output maps Create: Gray RGB R+G+B RGB+A R+G+B+A Smesh_ColorAD S	<ul> <li>Channel Pac</li> <li>Substance</li> <li>Be aware of</li> </ul>	king has a great interface fo of various compression	or this profiles	
Output maps     Create:     Gray     RGB     R+G+B     RGB+A     R+G+B+A       \$mesh_ColorAO     \$     \$     \$     \$	Current preset: GAME (Pag	cked)		and the second sec
Smesh_ColorAD S RGB A Ra	Output maps	Create: Gray RGB R+G+B RGB+A R	+G+B+A	Clear in the
	\$mesh_ColorAO	\$ RGB		
\$mesh_NormalEmissive \$	\$mesh_NormalEmissive	\$ RGB A		S' 30
śmesh_SpecPowOpac \$ R G B A 🕅	\$mesh_SpecPowOpac	\$ <mark>R G</mark> B A		

- In certain compression schemes various channels are discarded or compressed more harshly
- Sometimes green is given more range, use it for something like roughness
- Be careful with normal maps
- Sub channel packing is possible, works best with masks
- Trading unique samplers for dragging everyone along for the ride, usually worth it

# Tips + Tricks

## Build a "Benchmark" for your hardware

- Add increasingly more meshes over time
- Make a bunch of unique materials that wont get instanced
- Make a spaghetti procedural mesh
- Render 10 transparent full screen quads
- Add some giant particles for flavor
- Make sure to cast lots of dynamic shadows
- Dump FPS and any relevant data to a log file

Build a stress test scene that spits out metadata and framerate. Have it be manipulated via key press or controller, if in headset. For bonus points log everything. Do this early, and kill some assumptions and bad intuition with COLD - HARD - DATA.

✓ Default Clear Meshes Cyc	le Material Draw Meshes	Example Cheesepoof							
Steps	5	Ben	ichmark Tool						
₽ Meshes	1,500 Array element 🛨 💼								
Spacing	300.0								
Mesh Scale	200.0	u vermanti	×						
Mesh Count		Perspective VII Show	🔂 🕹 🖉 🚯 🖓 🗮 10 🛕 10° 🗾 0.25 🔳 4 🕰 🕫						
	20 💽 🖻								
	10 2	ALL DE LE							
	10 🔊 🖻								
Mesh	ProfilingSphere_760		-						
▲ Materials	3 Array elements 🕂 前 🤊								
	Profiling_MAT + + ● =								
	NewMaterial								
	Example_MAT								
		6	Level: Untitled (Persistent)						

I made this silly little guy for looking at overdraw and general shader complexity Useful at the beginning of a project to benchmark new hardware

- Super simple, just adds a 3D array of mesh components, can cycle through an array of materials
- Logs total Number of tris shown, FPS, The current material
- Could use particles instead, could add post process quads

2019.03.07	-22.37.30_mmme12703.mbc = NenderD0C V1.2	
File Window	/ Icols Help	
Timeline - Frame	: #12765	
EID:	1000 2000 3000 4000 5000 6000 7000	8000 9000 10000 11000 12000 13000 14000 🏲 14916 16000 17000 18000 19000 20000 21000 22000 23000 24000
- Scene	t Discourse (Example on Way)	I a facilitation
Usage for GBuf	ferA: Reads ( 🛕 ), Writes ( 🛕 ), Read/Write ( 🛕 ), and Clears ( 🔺 )	
1		
Event Browser		
		Similar Level      Alternative and a second and as
EID.	Nume	Channels RGEA 🗸 R G B A 🥥 🐰 V Subresource Mp 0 - 1120/004 U Sice/Face 🗸 Actions 🛃 🕛 🙃 🕅 🖉 🌧
14820	Drawindexed(11904)	
14823	Dynamic	Zoom 111 Barts String Vorenzy Voren
14827	API Calls	Range 0.00
14830-14832	> GPUParticles_PostRenderOpaque	
14833-14838	✓ ResolveSceneDepthTexture	Cur Input 2 - GBuffer A
14838	API Calls	
14839-14842	LightCompositionTasks_PreLighting	
14848	ClearDepthStencilView(S=00)	
14850-14871	ClearTranslucentVolumeLighting	
14872-14961	✓ Lights	
14873-14961	✓ DirectLighting	
14874-14955	✓ NonShadowedLights	22 DOLLES - CONTEX
14875-14916	<ul> <li>StandardDeferredLighting</li> </ul>	
14876-14880	✓ BeginRenderingSceneColor	
14880	API Calls	
14916	DrawIndexed(3)	
14918-14955	<ul> <li>InjectNonShadowedTranslucentLighting</li> </ul>	
14940	DrawInstanced(4, 24)	
14951	Drawinstanced(4, 24)	2 3 toxtre3 + CBuffer8
14900	API Calls	
14937-14938	<ul> <li>IndirectLighting</li> </ul>	
API Inspector		
EID	Event	
> 14881	ID3D11DeviceContext::RSSetViewports	
> 14882	ID3D11DeviceContext::RSSetScissors	
> 14883	ID3D11DeviceContext:IASetInputLayout	Pikel Context
> 14884	ID3D11DeviceContext:/VSSetShader	
> 14885	ID3D11DeviceContext:PSSetShader	
> 14886	ID3D11DeviceContext::GSSetShader	
> 14887	ID3D11DeviceContext::DMSetBlendState	
\$ 1,1000	ID2D11DaviesCententsDSSetCenstantRuffeer	
	Calstack	DBufferA - 1120x704 1 mips - R10010B10A2 UNORH

Heres that same render doc image from before using our test benchmarking tool

- We've drilled down into the lighting section, where we can find the gbuffers unreal is using to light the scene.
- Some of These gbuffers are also exposed in engine through the viewmodes, but not all, and not with this detail.

# **Example Problem**

- What's going on here?
  - We're using Unity
  - We're on an AR Device
  - Our Terrain Mesh Is taking most of the frame
  - It's drawing twice
    - Its not due to stereo rendering
    - It's not a multipass shader
    - There's no duplicate components

Heres a an example problem, with some clues. I'll show one at a time and give you a second to think about it.

We're on an Ar Device, We know a major hit is the scanned terrain mesh. However, this isn't a CPU problem.

The Terrain Come in as chunks... This is a clue

Even though the Terrain is Opaque, it's thowing up in the transparent Queue... This is a clue.



This is the default view in renderdoc after capturing a frame from the unity editor

We can see a list of any frames we've captured, with timestamps. The Timeline view is currently collapsed

· Unity 2	01949917_1724	timestosak - R	experibles with 2																	- 8 X
Die Mite																				
Timenie - Pr	me #37750	1995		104	4000	100	1000	22444	200	200	1446	1000	4444	10000	1100	100			Man inter store	104
100		a Camera Rando	2000	200 200	1000			0.00		1000 000				1000		110	P7 14997		100 1000 1100 1000	
		1 # Channy								ACC DR	14									+ C
		Kenderforse	Rosale Rander		Enternantional State As	endor				1000	ar/coaceceaners						Concert Transporte	112001237		
		Reidoffered	disanderLang Job		Enderforms d.Render	Laspitob														
						0.6-014-0-0														
Channel Start			(A) Section (A) and Second (A)																	
<																				2
I	-					Testire in	1965 X 😫 I	notive State: X	Meth Output	🕻 🛃 Launch Applic	stor × 💶 kab	est - Only PID 121	ing 🗙 📕 Ne	ourse inspector in					Prevenue calaurs singlayed in make range 0.00 - 1.00 with INTE-IN	
Tan I	Dent R	Marrie Marrie				Corents II	* A Succe	iens 🏚 kowi	offer 0 Olive	rice IT I Ven IT IS									Deadler date to seen to an event	
4803	997	MEL P	anin desert 13413	100000000		15 Deput						FiOurput	GLES OUR	A					Right dick to debug an event, or hide failed events.	
4805			nots deved(12)			933	125	POSTTONE			NCENSEA)	VIE		ST_NONTION				2	Evera	
4809			mindered[780]				1	-9-13464	1.13054	1.411	6.04745	11 <sup>-</sup> 189		-1.21163	8,7588	11,20776	-1014172			
4612			nvt+deved(12)																	
4815			neli dese 8(12)																	
4010			even deces (12)																	
4834			windere KAS()			1														
4827			mindered(942)					-0.64105	0.02516	1.50319										
4830			ents deved(12)															_		
4833			nvindered(18)					-1-16575	0.01023	1_40914	0.02444									
4836	708		ewindeeed()51						0.71501		0.00242									
4839			nath develop (12)					-1.25100												
4845		- De	evindenod(1074)					-L 16575	0.01020	1-40914	0.02444									
4543													4					•		
4855			naindexed(18)									» «								
4854			enfedeend(304)			10 in														
4857		1	windered(12)				·	and the second		-		-								
4863		D	exhibition ([12]					and the state				With a								
4555																				
4090			extraced(30)																	
4872			natis deserd (241)																	
4870		0	entra deseal (2021)																	
4885		De	awin devel(\$69)																	
4884	724	0	an in deved (310)		1444															
488.7	10 11 100	TH PP	Diserche=101771	4																
API Impect																				
10																				
> 4815			itest://SetVertesBuffers																	
> 4886		D3011DeviceCor	rest:/ASetindesBuffer																	
22410		ID2011DeviceG	antest Drewlasleand																	
		***	Calitocit																1.	
		2 100 x 2010.00	17 17.34 forme 0755 atc insteal No sublem detected																	
and the second second		Consideration (12)	and a second second second and presenting description																	

Step 1: Turn on dark mode, it makes you smarter and more powerful

Here we've expanded the timeline view/

We can ignore half because we know we're using multipass stereo, so half of the frame is a duplicate.

We can also see That about half the frame is used to draw our opaque geo and half for our transparent geo. But we know our terrain meshes should be drawing in the transparent queue at all, so what's the deal?

Unity_2019.03.17_17.24	frame37755.rdc - RenderDoc v1.2							– a x
Elle Window Eachs H								
Titleline - Frane #37756								×
	100 1500 2000 2500 300 Camera Ander	e 3500 +130 9000	5300 600	0 600 700 700	1000 1000	9000 9500 10000 11000 11000	11500 12 12206	100 1300 1300 1400 1400 1000
	é - Drawing				+ C Draving			¥C
	Ender.OpageGeonetry     EnderforceardToxa e Ender	Ender TransparentGeor     Ender TransparentGeor	metry		+ Rander/OpaqueGeanetry		<ul> <li>Render, TransparentOcometry</li> <li>Render PorsantAirdon Tender</li> </ul>	Y
	RenderForward.RenderLoopJob	- Renderforward.RenderL	000000				Rende Forward Rende Loop	tab
Usage for No Resource: Res	ds ( 🔺 ), Tintes ( 🔺 ), Read/Write ( 📥 ), and Clears ( 🖒 )							
e								
Dvent Browser			< Texture Viewer X	Repeire State X Reh Output X	Launch Application 🗙 📕 localhest - U	nity PID 12156] X 📕 Resource Inspector X		
Cantrols 🔶 🛶 👪	► • · · · · · · · · · · · · · · · · · ·		Contrast De Bornel	Perchant Dance Different Dance Different	t historios			
ED Draw#	Name	Duration (µs)						Double click to jump to an event. Right click to debug an event, or lide field events.
12197 2025	Drawin deced(32)		IA	🚽 VS 🛏 Hull Shader	🗕 DS 🛏 G	S Rasterizer Pixel Shader	OM CS	
12173 2027	- Drawindered(32)							
12176 2028	Drawle dexed(72)		State					
12179 2029			Ploorshot/Cursor,	🖉 🔶 🖓 🔂	* 🔜 Save			
12182 2030	Drawindexed(72)		T Beatron					
12188 2037	Davisterel 21		Set					
12191 2033	- Drawin deced(12)							
12194 2034								
12197 2035	Drawindexed(21)							
12200 2036	Drawindeced(160)							
12205 2038	Provindered (\$43)							
12209 2039	Drawindexed(32)							
12212 2040	Drawindexed(554)							
12213 2041	Drawindcood (2)							
12221 2043	Drawindexed(12)	-						
12224 2044	- Drawindexed(12)		<ul> <li>Samplers</li> </ul>					
12227 2045	Drawindexed(32)		SH	Sampler Addressing	Filter	LOD Clamp LOD	Biasi	
12230 2040	Drawindored 121							
12236 2048								
12239 2049	Drawindexed(993)							
12242 2050	Drawindexed().()							
12248 2052	- Drawindexed().2)							
12251 2053	Drawin deced(31)							
API Impector								
ED.	Event							
3 12204	D1D110exiceContext://ASetilentext/uffers							
> 12205	D3D11DeviceContext:/ASetIndex8uffer		544					
7 12200	DJD11DeviceContest_Drawindexed		0: cbatter0	ConstantBuffer-222-336 e <sup>R</sup>	0 - 4096	10 Variables, 160 bytes needed, 336 provided	•	
			1: cbuffer1	ConstantBuffer-193-160 e <sup>0</sup>			•	
			2 cbutter2	ConstantBuffer-194-168 g <sup>D</sup>	0 - 4096	21 Variables, 336 bytes needed, 368 provided	•	
	Celetect							
Replay Context Local	Unity_2019.03.17_17.24_finame37755.rdc loaded. No problems detect	ed.						

With the call to draw a piece of the transparent terrain selected, lets check out what shaders its using in the Pipeline State tab. Hmmm. HMMMMMMM



AHA! Why is our terrain mesh rendering with the cursor material? Well, our cursor is projected onto the terrain but... Oh. Oh no.



SO here's the culprit. That box is not a bounding box. That's the culling volume unity's Projector Component uses to determine... Which meshes to DUPLICATE, and render the projector material on.

It's currently set to 2... meters.

Early in development the magic leap meshing api brought in meshes in tiny chunks Later they updated to large chunks... We increased the culling size to make sure the cursor always drew on enough chunks.

# **Example Problem**

## • What happened?

- It wasn't what we suspected (intuition failed)
  - We thought it was the for loop used in our terrain metaballs (it wasn't)
- We used the Unity Projector Component for the game cursor
  - We enlarged the culling volume after the terrain chunks became less granular
  - This was a vital feature and while we had solutions, we had no time to implement
     <u>http://blog.wolfire.com/2009/06/how-to-project-decals</u>
- We decided to optimize as much as we could and eat the cost
  - The effect was vital feedback for gameplay

Sometimes, the correct answer is to let it be.

In this case, we found it so late in production, we had to tighten knobs everywhere else to make room for a vital feature.

We hit our performance targets by optimizing in other areas.

# **Talking Optimization With Your Team**

- Gaining Perspective
  - Put your work in context by communicating across disciplines
- Beware of the Implications
  - Changing things may have consequences



Especially later in production

# **Talking Optimization With Your Team**

- Artists might see a "Black Box"
  - May have partial knowledge or superstitions
    - Good Intentions but poor implementations
    - Many dubious ways to skin a cat
  - Empowering them can have a big impact
    - Junior artists might be all over the place
    - Senior artists might be more conservative than necessary



MOVIECLIPS, COM

# **Talking Optimization With Your Team**

- Engineers have different priorities
  - They may not be as focused on the gameplay or aesthetic implications
    You may need to advocate for an expensive feature
  - Trust but Verify
    - Don't take a passing statement from your engineer as gospel
    - Follow up with them and do your own benchmarking

If we disable all rendering performance increases dramatically!

## **More Stuff**

- LearnOpenGL.com
  - "What the heck does any of this mean?!"
- Shader and C# tutorials
  - Alan Zucconi
  - Catlike Coding
  - Unity is a great tool for learning about shaders
     Scriptable Render Pipeline is pretty exciting
- Unreal Deferred Pipeline Breakdown
  - Matt Hoffman UE4 Rendering

Further reading and resources

# Questions, Comments, grievances

- garrettorious@gmail.com
- @creatosaurus 🌱



Your shaders after this talk