

## Surfing the wave(front)s with Radeon™ GPU Profiler

Dominik Baumeister Developer Technology Engineer Advanced Micro Devices, Inc.

SAME DEVELOPERS CONFERENCE MARCH 18-22, 2019 | #GDC19

## Overview



- Wavefronts & Barriers
- Cross queue synchronization
- Compression
- Wrap-up





SAME DEVELOPERS CONFERENCE MARCH 18–22, 2019 | #GDC19

Now this is how the talk will look like:

 I present you with a problem that has been seen in a loooooot of AAA titles. Both shipped and upcoming ones.

I'm not going to say which titles these are, don't want to offend anyone and sometimes you have to call a solution "good enough". People want to ship games at the end of the day.

- I'll show you how to use RGP to

identify the problem

- I'll explain the underlying issue
- I'll tell you how to fix it
- For the following three topics: Barriers, Async, Compression

All of the information given in this talk is about Vega.

Wavefront occupancy         Event Uming         Ppoline state           ✓ Color by API shader stage         ✓ Color by API shader stage         Zoom to selection         Reset zoom         ©           000000000000000000000000000000000000	START	OVERVIEW EVENTS			
✓ Color by API shader stages         Zoom to selection         Reset zoom         ©           0.000 µs         2,000.000 µs         7,000.000 µs         12,000.000 µs         13,000.000 µs         12,000.000 µs         13,000.000 µs         12,000.000 µs         13,000.000 µs         12,000.000 µs         10,000.000 µs					
			Pipeline state		Zoom to selection Reset zoom
	0.000 µs			s 10,000.000 µs 12,500.000 j	
	0			likaan and and	





You can access this via Events->Wavefront occupancy.

This is probably the view you will spent your first few minutes on to figure out what's going on in your frame.

START	OVERVIEW E	EVENTS						
Wavefront occupancy	Event timing	Pipelin	ne state					
Ƴ Color by API shader stage	✓ GCN shader stage	15				Zoom to selection	Reset zoom	
0.000 µs	2,500.000 µs	5,000.000 µs	7,500.000 µs	10,000.000 µs	12,500.000 µs	15,000.000 µs	17,500.000 µs	
3409 Especto								I 100% 75% 50% 25% 0% 25% 50%
								75%
S VS HS & DS GS	PS CS							100%





You can access this via Events->Wavefront occupancy.

This is probably the view you will spent your first few minutes on to figure out what's going on in your frame.

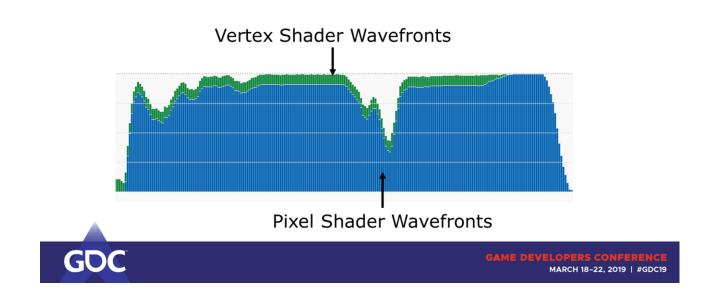
START							
Wavefront occupancy	Event timing	Pipeline state					
✓ Color by API shader stage	✓ GCN shader stages				Zoom to selection	n 🛛 Reset zoom 🗍 🕒 🧲	
	dama da ang ang ang ang ang ang ang ang ang an		հյուստեսուտեսուտ	ւնաստեստունըս	malaanaahaan	mpuntum	
							50%
			a dhahin b				25%
As							25%
sync o			C C 10 309 1 1 19000 0				50%
omp			1° 10				75%
č		-				2	100%
VS HS & DS GS	PS CS						

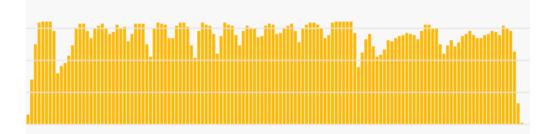




You can access this via Events->Wavefront occupancy.

This is probably the view you will spent your first few minutes on to figure out what's going on in your frame.

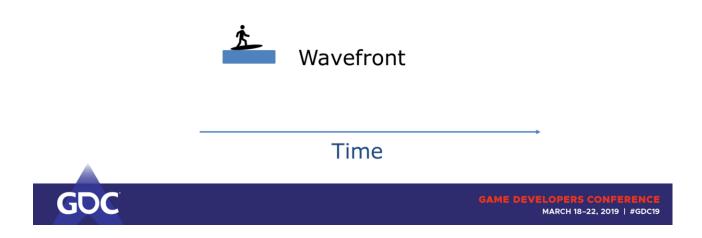




## Compute Shader Wavefronts



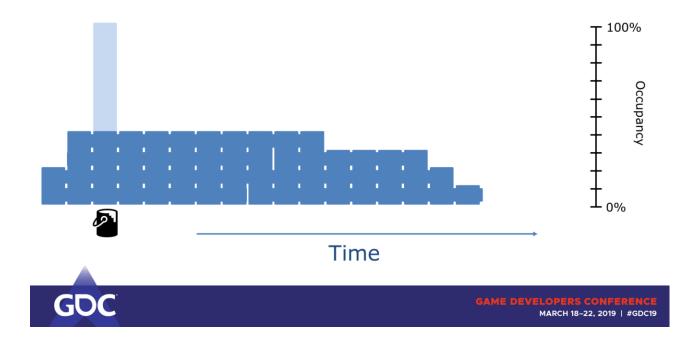
GAME DEVELOPERS CONFERENCE MARCH 18-22, 2019 | #GDC19



May I introduce: a wavefront.

Your dispatch/draw is split up into multiple wavefronts. Each executing your shader program with multiple threads in lockstep.

Once a wavefront finishes executing the shader program its HW resources (registers etc) are freed and new wavefronts can spawn.



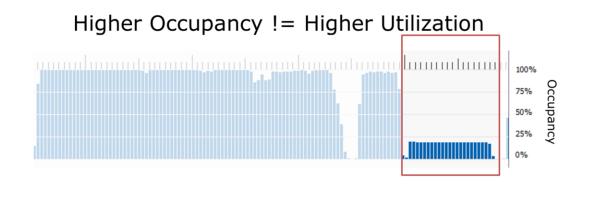
RGP puts them into buckets of a certain duration and counts the total number of wavefronts that run on the GPU during the time slice of each single bucket.

Since there are different wavefront counts on each GPU family, the filling is normalized to the maximum amount of wavefronts.

If our hypothetical GPU could run 10 wavefronts at once, RGP would state

40% occupancy here. (real numbers are in the thousands)

And for completeness sake, this is how RGP would generate the trace for our hypothetical GPU.







Remember: Higher occupancy is simply more waves in flight.

Important: It's occupancy, not utilization! The ALUs may already be stressed to the max.

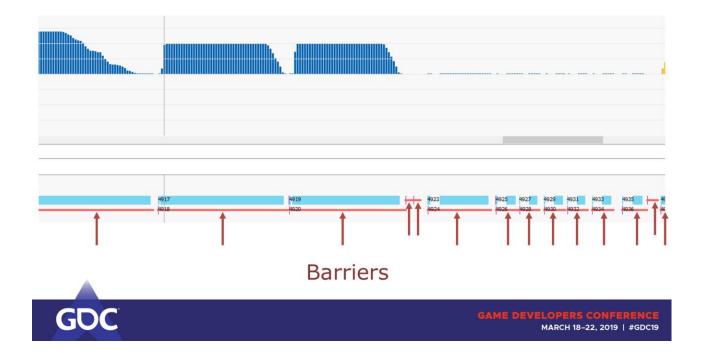
Imagine on a CPU: Dispatching a lot more virtual threads than HW threads doesn't help you much.

→ Fewer waves is not necessarily bad! Not saturating the ALUs/Bandwidth is.



Obviously, if there are NO waves running, then we're not using the shader core to full extent (i.e. we don't get at least one wave onto each SIMD).

Also, usually not all waves are done at the same time. See the occupancy dropping at the end of the workloads.

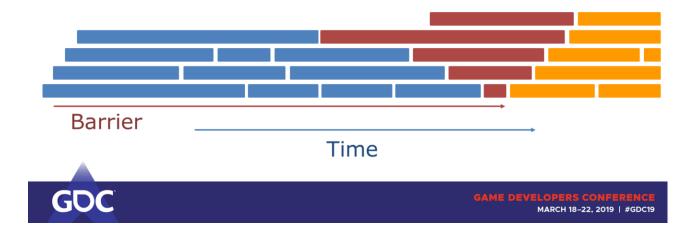


# But what is causing that? RGP knows the answer: Barriers!



## Back to our "zoomed" in view.

Without any synchronization the next compute shader could start here.



Barriers cause the next bunch of work items to wait until the prior work all finished.

They also sometimes cause other work to happen (think of compression/decompression & making work visible to other units => flush caches). Thus, they sometimes have some additional overhead aside from draining the shader core. Now the important part, it's not only that tiny bit of work being lost (1) It's actually that much (2)

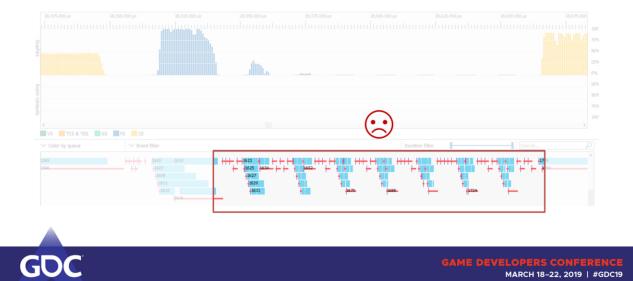
103,450.000 µs 103,500.000 µ				50.000 µs 17,800.000 µs	17,850.000 µs ,900.000			04,0
	,750.000 μs 11,775.000 μs	11,800.000 µs 11,825.00		11,875.000 µs 11,900.000 µs	11,925.000 µs 11,950.00		12,000.000 µs 12,0	
	աստեստանությունում	ահաստահաստություն	taaaadaanaantaaaaa	հատահատահուսահոս	ահատորություն	duuuuduuuuduuuu	վառափառակա	d _
	~							
								-
								_
	<b>iiii</b> 1	de la lite da		and the dealer of the second				
			<					1
								-
								>
VS «							>	E
Cole VS TCS & TES	SS 📕 FS 🦲 CS							-
51 V Color by queue	V Event filter			1	Duration filter	Search		Я
52	· creating			1				1
i190 5192 5194 5	196	5214 5216 5218	5220 5222			54		i.
	197	5215 5217 5219	5221 5223	5225	+	55		
			1					4
			10					



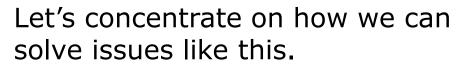
GAME DEVELOPERS CONFERENCE MARCH 18–22, 2019 | #GDC19

So we're up for issue number 1: Using a barrier after each draw.

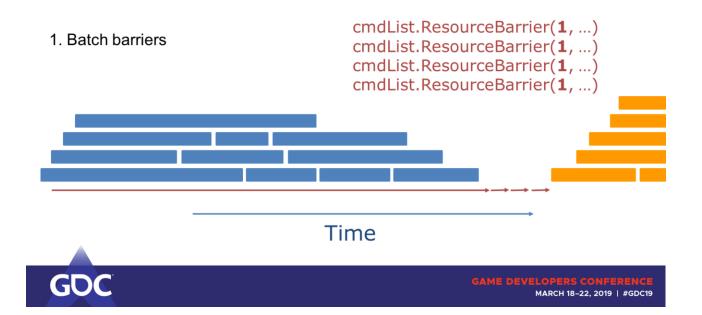
Or even worse: multiple barriers after (almost) each draw.



MARCH 18-22, 2019 | #GDC19



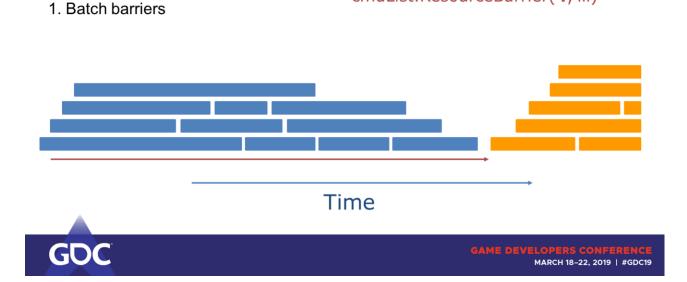
Obviously we can't just get rid of all barriers. Some synchronization will always be necessary.



First, we can batch them, so the driver can manage the synchronization points more precisely.

It's as easy as submitting them together in a single vkCmdPipelineBarrier / CommandList::ResourceBarrier.

The driver can then easily figure out the worst case synchronization or if only a single cache flush is necessary (instead of multiples) etc.



cmdList.ResourceBarrier(4, ...)

Only a small impact, but the effects add up! Don't neglect those.

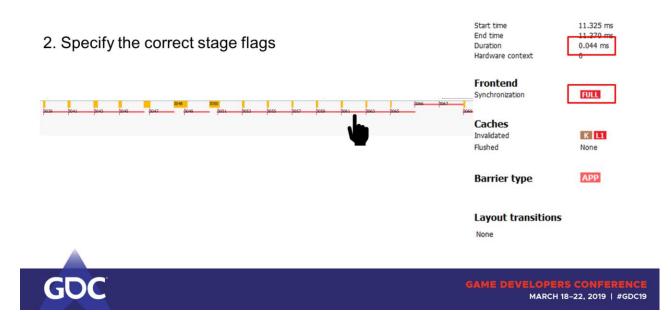
Also, certain compressions may start earlier, that's a net win!



This is how you will see it in RGP.

Note that the first barrier is still exactly as long as before, but we were able to get rid of all the other barriers!

We successfully overlapped the FastClearEliminates (writing cleared values from meta data to every pixel -> decompressing the texture) as well.

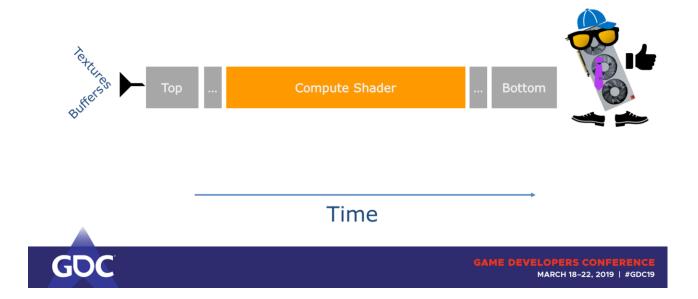


Be very descriptive in how you define the stage flags. In some cases you pay for being too general.

Vulkan specific. Found by a developer we work with.

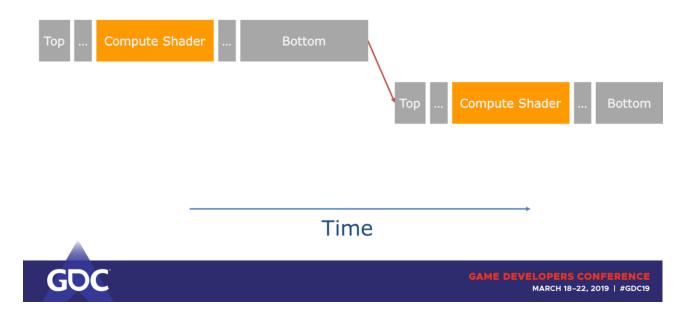
If you click on any of these barriers and look to the right side into the details again you see each barrier taking 0.044ms to execute.

Doing a FULL synchronization each.

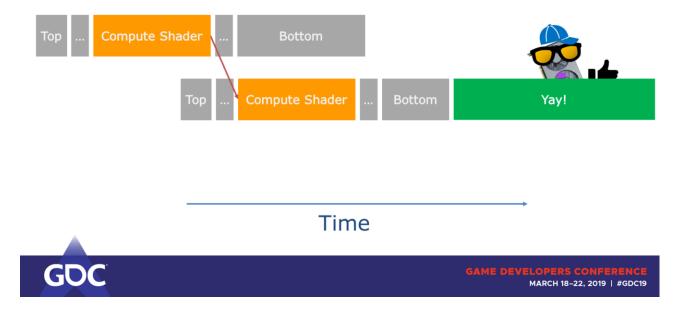


Pretend we would go through the GPU pipeline in a straight line. Compute only here.

We put stuff like textures and buffers in on top, pass a lot of stages until we execute the compute shader until our results drop out at the bottom.

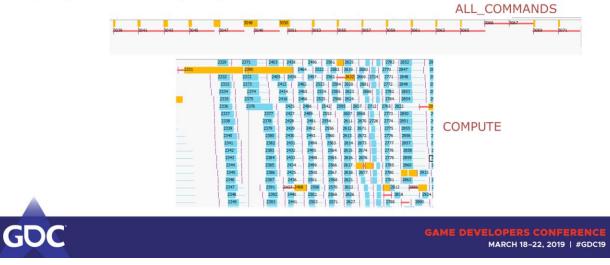


Now let's say in some special corner cases BOTTOM\_OF\_PIPE takes a long time to execute. But we're only interested in dependencies between the CS!



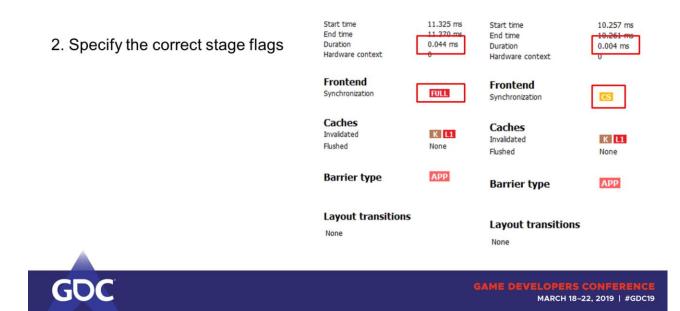
If we would describe the accesses more precisely, we could overlap more.

2. Specify the correct stage flags



Here it was set with ALL\_COMMANDS (contains BOTTOM\_OF\_PIPE) on the async compute queue.

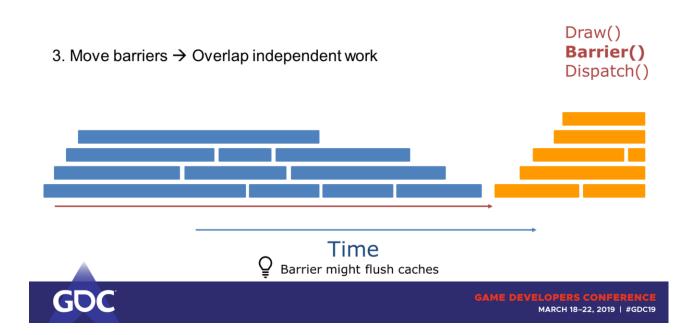
Replacing that with COMPUTE shows a big improvement, here enough to overlap with other draws on the graphics queue.



To prove it, here's the detailed view when clicking on these barriers.

We replaced a full synchronization with the proper CS sync, which is a lot less expensive on the async queue.

Almost a magnitude improvement on each barrier.



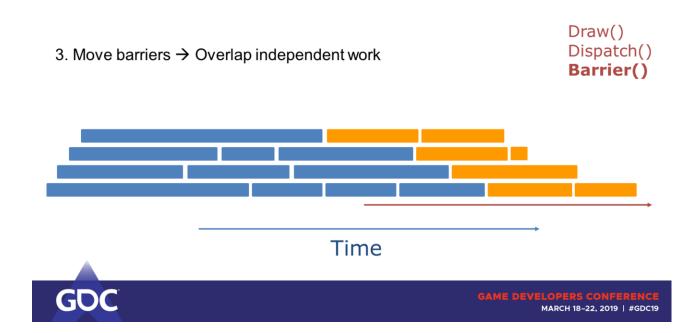
Work that is independent does not necessarily have to wait.

→ Move the barrier after independent workloads.

Added benefit: barriers often flush cashes as well.

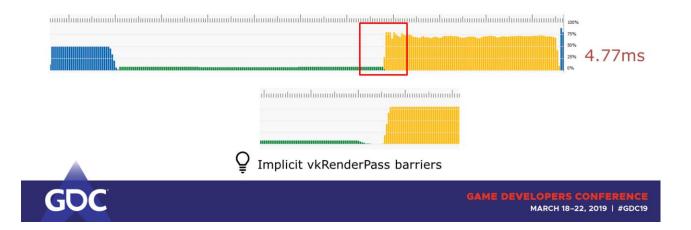
If PS and CS share reads here, then the CS may run slower as it doesn't hit in L2 anymore. Order of submissions here:

- 1. VS/PS
- 2. Barrier
- 3. CS



Order of submission would be:

- 1. PS
- 2. CS
- 3. Barriers



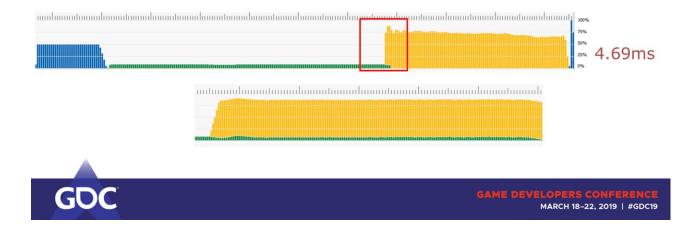
Usually what you want to do is to overlap long running CS waves with raster heavy workloads. In RGP these usually show up as short VS waves with low occupancy.

This is what we start out with. Notice the barrier that causes a gap between the Vertex and the Compute work.

Let's start with moving that barrier behind the CS.

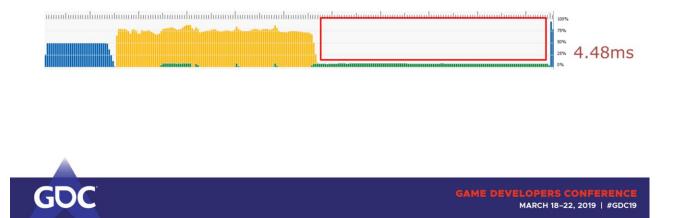
Pro tip these barriers may be due to start/end of RenderPasses: Add explicit TOP to BOTTOM barriers (basically no-op barriers) as subpass dependencies to EXTERNAL

to get rid of implicit barriers at the beginning/end of render passes in Vulkan.



Some tiny overlap at least.

But we can do better: The CS wavefronts are running longer than the VS wavefronts  $\rightarrow$  We should swap CS and VS submits.

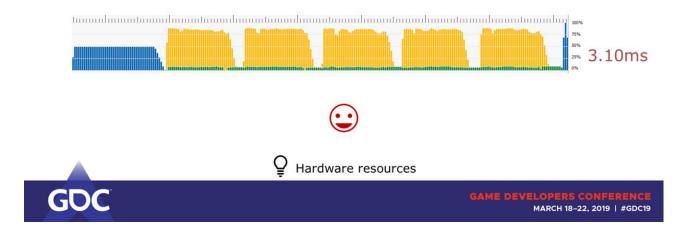


This already looks better than before. Still not ideal as at some point we only spawn new CS waves, no VS waves anymore.

 $\rightarrow$  We can split up the CS work into multiple dispatches and interleave it with VS work.

E.g. if the CS is fullscreen, split it into tiles and interleave it with draws.

Imagine shadow cascades with SSAO.

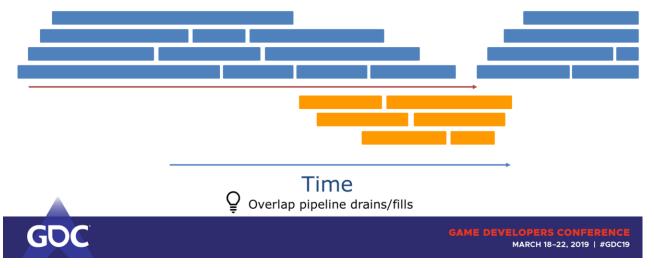


This gives amazing results! And completely without going to the async compute queue.

Works particularly well if you find passes that have significantly different hardware utilization.

E.g. texture fetch heavy SSAO with vertex heavy shadow map rendering.

4. Asynchronous compute



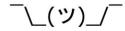
Async queue does not force waits on the graphics queue! Useful for whole compute passes that require syncs to move to the async queue.

Keep attention to queues fighting for resources.

Move passes with a lot of work to the async queue, and keep the number of cross queue syncs to a minimum.

#### 4. Asynchronous compute







# That probably didn't turn out as expected.

#### 4. Asynchronous compute





# But if done correctly, this can work out great!

See how the compute queue can take over while the graphics queue is draining and filling.





```
GAME DEVELOPERS CONFERENCE
MARCH 18–22, 2019 | #GDC19
```

Varying workloads make this a hard problem. Sometimes you end up starving the graphics queue.

Rather aim to keep the graphics queue busy first.





GAME DEVELOPERS CONFERENCE MARCH 18-22, 2019 | #GDC19





## Gaps, but no barriers. Where do they come from?

Let's head over to a different view.

Prane summery	OVERVIEW EVENIES SECTIONS System activity
Barroz Histo appendo entre Context-tudo Rondo Elizede kangelis Decisios configuenties	Mentance and Market answert (Market answert (M
	Remaining and a specific and a speci

When you open up your capture you are greeted with the frame summary page.

Anyone working with GPUView might see similarities.

Prane 668 25.000 ms IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII	Sequences of the sequence of t				* Prame 3
G Regional Assessment (College) Manual Assessment (College) Manual Assessment (College) Manual Assessment (College)			(Twister) Ad	Kinesepireleginisti (Jan 2004)     Kinesepireleginisti (Jan 2004)	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
QueueSubert(2) QueueSubert(4 Computer queues	Concentration(3) Concentration(4) Concentration(4)	Mile         QueQuestelen(Q)           Sol         Mile Statement (Q)           QueQuestelen(Q)         QueQuestelen(Q)	2001 (2)	ol (parastalanett) Al (Sametalanett) al Qavetalanet(R	andred B Balance Balance
			Salmut time: 32,934 mi   Salmut a		mittan: 8.000 ms 28.30

Top is the graphics queue.

fan town	System activity The adverse developments The adverse development The adverse d
	GAME DEVELOPERS CONFERE MARCH 18–22, 2019   #G

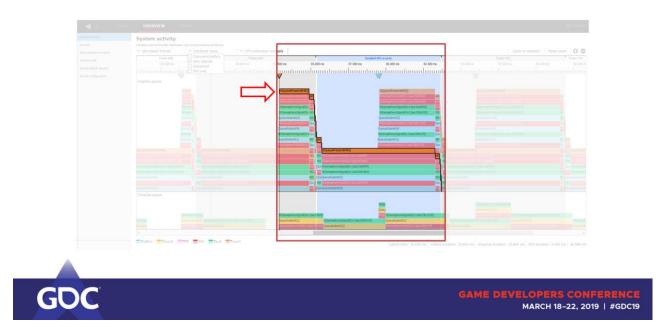
# Bottom is the async compute queue (and/or copy queue if available)

STAR	I OVERVIEW EVENES SETTINGS
Press summer Barriers Mart copensis events Context rate Rander Maph targets Device configuration	System activity         Zoon to setting           Twins more determined to provide provides         Zoon to setting         Reset zoon         ©         ©           Vib/stand finance
	AGA     AGA     AGA     AGA     AGA
	Description     Desc
	Description     Description     Description     Description     Description       Compart gene     Sample and part of the
	Graphice Graphice (DML Computer (DML Computer (DML) Computer (DML) Computer duration: 10.803 ms (DPU duration: 0.000 ms 28.348 ms
GDC	GAME DEVELOPERS CONFERENCE MARCH 18–22, 2019   #GDC

New work item submissions (command buffers, waits, signals, presents) are added on top.

Each time one finishes, the others drop one level.

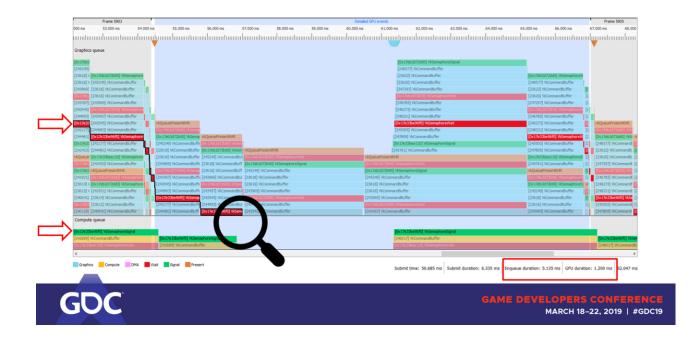
The lowest level are the work items the GPU queue worked on at that time.



RGP marks the frame it captured in light blue, but also shows you the frames prior and after.

The present packets are used to identify the frames.

This page will thus also show you on which queue the present ends up on.



Let's now concentrate on the sync primitives (different trace).

Clicking on one of them shows related signal/waits.

On the bottom right you see how long it took to submit the packet, how long it's been queued up and how long it took the GPU to execute it.

Or in a case of the wait how long it prevented the GPU from progressing

### further.



Marking a selection shows the timespan in the bottom right corner of the view.

You may have noticed these small gaps as well.

- The GPU interrupts the CPU kernel to signal that a command buffer finished.
- CPU side bookkeeping can sometimes cause delays.

#### Async compute recommendations:

 Synchronize seldomly, ideally only 1-2 times per frame Each synchronization point has significant overhead
 Move large continuous workloads to the async queue More opportunity to overlap pipeline drains / fills
 For adventurers: overlap with next frame Usually frames start with raster heavy work and end with compute heavy post processing May add latency!



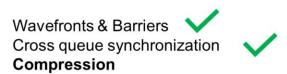


Not much you can do about -> only sync seldomly. Only 1-2 times per frame ideally.

Let it run uninterrupted as long as possible to overlap pipeline drains and fills.

And can even overlap with the next frame if your game can take the added latency.

That works because frames usually start with raster heavy workloads (GBuffer, Shadow maps) and end with compute heavy post processing.





GAME DEVELOPERS CONFERENCE MARCH 18-22, 2019 | #GDC19

START	OVERVIEW EVENTS					
Prane summary						
Sarriers.						Zoom to selection Reset a
Must expensive events				Color pass #11		
Context rols	.200 ms 4.300 ms 4.4		4.700 ms 4.800 ms	4.900 ma 5.000 ma	5.100 ms 5.200 ms 5.30	
Render/depth targets		anataanahaanahaanahaanahaana	homotonooloonotooo	daamtaanalaanataa	adaanta madaana daga daga daga daga daga daga dag	matanandananatanandananatananaharanata
Device configuration	Color					
	Color RT #30 Color RT #30	Color RT #30		Color RT #30	Color RT #30	Color RT #30
	Color RT #29 Color RT #29 Color RT #28 Color RT #28	Color RT #29 Color RT #28		Calor RT #29 Calor RT #28	Color RT #29 Color RT #28	Cdor RT #29 Cdor RT #28
	Color RT #27 Color RT #27	Cdar RT #27		Color R7 #27	Color RT #27	Citiz RT #27
				A PROPERTY AND A PROPERTY		and the second second
	Depth					
	Depth II Trans ExperimINT 45	Depth IIT #5		Depth KT #5	Depth RT #5	Depth 87 #3
	and the second se	Septe RT #5		Depth RT #5	Depth ат #5	Depth RT #5
	Depth RT #5 Depth RT #5	Cvgdh RT #3		Depth AT #3	Depth RT #5	Decivitit vis
	and the second se	Depit IIT #5		Clash 87 #5	Digit 87 #5	2e6517.0
	and the second se	General es		desch KT #5	Ckydr AT 25	Day6.47 #3
	and the second se	Create AT #5		Dagh RT #5	Depts 17 45	page at as
	and the second se	0od 81 45				Darin 47.45
	Other	*-+	, au			Den ST AL
	Other   2445	as ↓ t (✓ Select columns				joan () - jess
	Other 	as ↓ t (✓ Select columns		•	÷	joan () - jess
	Other jata Autor jata Depand tree V Group by targ Rame For V Group V Group by targ	t  √ Select columns mat Wdb	Height Draw calls	Compression Sample count	Out of order draw calls Duration	joan () - jess
	Other  24	t  ∨ Select columns nat work 1500 1500 1500	Height Draw calls 1000 2147	Compression Sample count OFF 1	Out of order draw calls Doration 0/2147 2.444m	joan () - jess
	Ditter  Espand tree  Company from  Company f	ti	Height Draw calls 1080 2147 2080 2135	Compression Sample count OPF 1 OPF 1	Out of order draw calls         Duration           0/2147         2.464m           0/225         2.327m	joan () - jess



GAME DEVELOPERS CONFERENCE MARCH 18-22, 2019 | #GDC19

					~
					0
	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,				
Color					
Color RT #30 Color RT #30 Color RT #30		Color RT #30	Color RT #30	Color RT #30	
Color RT #29 Color RT #29 Color RT #29 Color RT #28 Color RT #28 Color RT #28 Color RT #28		Calor RT #29 Calor RT #28	Color RT #29 Color RT #28	Cilor RT #25 Color RT #25	
Color RT #27 Color RT #27 Color RT #27		Color R7 #27	Color RT #27	Celor RT #27	
		-			
Color RT #30					
Color RT #30		Deels 27 40 -	Spell R. a)	Spint at	
10.00			(see kta)	Street	
10.00			Stations)	and the second se	
Color RT #30 Color RT #29			Sawitzak	Den el A	
Color RT #29		Constitution of the second	See 172	para la para	
Color RT #29		County of a	(1000-000) (1000-000)		
Color RT #29 Color RT #28			(See to 2.1)		
Color RT #29		_	Out of order doue calls Deuxtion	haar 11-1 haa	
Color RT #29 Color RT #28		_	Dut of order done calls Deration 0,2200 2,044m	haar 11-1 haa	
Color RT #29 Color RT #28		_	Out of index data         Desistent           0(2107         1.401m           0(2107)         3.201m	haar 11-1 haa	
Color RT #29 Color RT #28 Color RT #27	100 100 100 100 100 100 100 100 100 100	_	Dut of order done calls Deration 0,2200 2,044m	haar 11-1 haa	



```
GAME DEVELOPERS CONFERENCE
MARCH 18-22, 2019 | #GDC19
```

This view is not split into queues, but rather accesses to resources.

Color render targets on top, depth targets below and barriers & buffers on the bottom.

											00
Concent and Concent and Concen				Cale KT acts Cale KT acts Cale KT acts Cale KT acts			Comer ACT at 20 Comer ACT at 20 Comer ACT at 20 Comer ACT at 21		Cale III all Like III all Cale III all		
Depth											
Depth Costinanes Costinan	ия реректия			Deph AT #5			South II.T #5		Depth RT #5		
Depth RT #5 Cleath RT				Deph RT #3	_		Depith 87 #5				
Depth RT	<b>#5</b> Dity beiget (→ delect columns					+				David,	
Constants Depth RT Logard Inst Notes	<b>#5</b> big barget → failert calures raread	width Heigh		Compression	Samjir cond	Dist of order draw culls	Duration			Deed	
Depth RT	#5 Frend V field adverse Frend V field adverse Field Adverse V field Adverse V field Adverse Field Adverse V field Adverse V field Adverse Field Adverse V field Adv		6 Dise cile 210 233		Siample count	+				Dara	
Contract of the sector	<b>#5</b> big barget → failert calures raread	width Heigh		- - Campirraidon - Crit	Samjile cound 1 1	pat Out of order draw calls $Q(2)\Phi$	Duration 1.40+m			See	



```
GAME DEVELOPERS CONFERENCE
MARCH 18–22, 2019 | #GDC19
```

This view is not split into queues, but rather accesses to resources.

Color render targets on top, depth targets below and barriers & buffers on the bottom.

The part you see here is the Gbuffer pass (4 color targets + 1 depth target).

Side note: May want to take care of these barriers splitting up the Gbuffer pass.

								0   Q
CONTRACT (20 CHar)							Calor 87 #29	
Case 47 A 20 Case A							Case 17, 423	
DerATATE DerA				Calif. Kr. 822.7		Cour AT ALT	Emr 87.42	
	10)					Prod of the	Destroy	
-				. In the			- Detroit	
-				Terri (1999)			- Perce	
-								_
-							003   +   wes	
0ther   2415   2 <	- <del>200  </del>		4				(ens	
Other 	<mark>jater j</mark> up tij target V Securi column			766 -	-		(ell)	
Dher deale deale Appendiate of the Law	1250 - 1 up by target Armout	width the	jat Dysecolo	Compression Sample cm	ent. Out of order drove calls	a Develop	(00.   -  )#1	
Coner ( ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) )	i passe	20400 He (120 10	plot Orsow calls 1 2147	Compression Sample on OPT 1	ent Out of order draw call 0 (2007	Denstron 2.40+m		
Coher	later	1000 No. 100 1020 No. 1020 No.	4 ght Dysw.colls 5 2(47 2 203)	Compression Sample on OTP 1 OTP 1	■ Out of order draw call 0 (294) 0 (233)	<ul> <li>Developing</li> <li>2.49 m</li> <li>1.82 m</li> </ul>	(00) () () (00)	
Coner ( ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) )	i passe	20400 He (120 10	plot Orsow calls 0 2147 2 233 2 2142	Compression Sample on OPT 1	ent Out of order draw call 0 (2007	Denstron 2.40+m		



GAME	DEVELOPERS CONFERENCE
	MARCH 18-22, 2019   #GDC19

This view is not split into queues, but rather accesses to resources.

Color render targets on top, depth targets below and barriers & buffers on the bottom.

The part you see here is the Gbuffer pass (4 color targets + 1 depth target).

Side note: May want to take care of these barriers splitting up the Gbuffer pass.

								Contraction 1
Convikt #20 Date htt #2							Like B	
Courses and Constant of								
Eleventy star. Elevents	Dame all well			Caler bit #27 :			Carls	
	A- David (17.4)			Capital Station				
	Description			Depth 47 at			East	
	Paral 1943			Dert H1 at -			Eser	
Dent.								
	2005.0)		,					
Dent.			ii I					
Constanting and a constant			Height Draw calls					
Columnia Columnia Digund tree	langet ( <sup>™</sup> Select columns	Width H	1	28				
Constant Constant Const Expand there Rease	izen	Width H 1920 2	Height Draw calls	Compression Sample court	Dut of order draw calls	Duration		
Column Column Expand tree Name Name Name Name Name Name Name Na	lotanui by target  √ Select columns format vc_r00wst_stacassa.goca	Width H 1920 9 1920 5	Height Draw calls	Compression Sample count OPF 1	Out of order deaw calls 0 / 2147	Duration 2.404 ms		
Converting and a converting and a converting and the set of the converting and the conver	by target: ✓ Select columns Format w(_700w1_,XS1051893_JA00H_/AOC2	width         H           1920         3           1920         5           1920         3	Height Draw calls 1060 2147 1080 2135	Compression Sample count OPP 1 OPP 1	Out of order draw calls 0/2447 0/2135	Duration 2-494m 2.307 m		



GAME DEVELOPERS CONFERENCE MARCH 18-22, 2019 | #GDC19

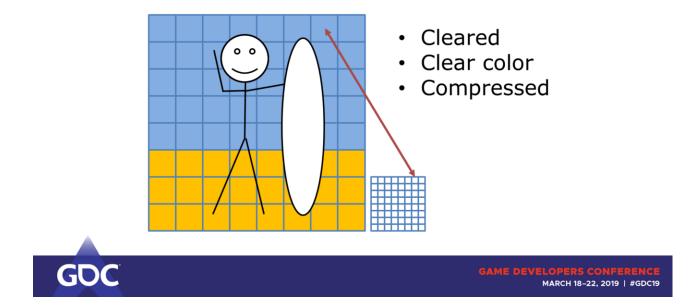
# Below it you can see details that unroll once you click on a RT.



	Name	Format	Width	Height	Draw calls	Compression	Sample count	Out of order draw calls	Duration
>	Color RT #27	VK_FORMAT_R8G8B8A8_SRGB	1920	1080	2777	OFF	1	0 / 2777	2.414 ms
>	Color RT #28	VK_FORMAT_A2R10G10B10_UNORM_PACK32	1920	1080	2766	OFF	1	0 / 2766	2.100 ms
>	Color RT #29	VK_FORMAT_R8G8B8A8_UNORM	1920	1080	2773	OFF	1	0 / 2773	2.170 ms
>	Color RT #30	VK_FORMAT_R8G8B8A8_UNORM	1920	1080	2774	OFF	1	0 / 2774	2.416 ms



GAME DEVELOPERS CONFERENCE MARCH 18-22, 2019 | #GDC19



Let's make this a quick (barely scratching the surface) introduction to texture compression as it's used for render targets.

Let's start out with a texture and cut it into blocks.

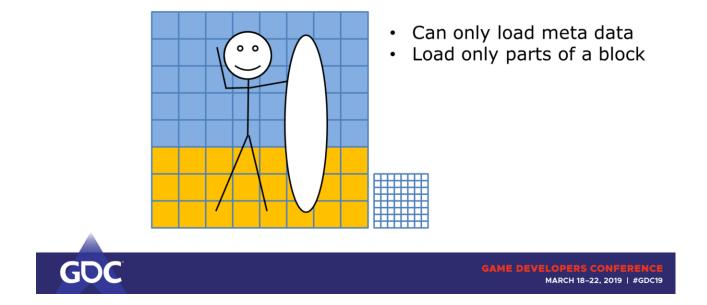
We attach some meta data to the texture that describes attributes per block.

Like

- Cleared
- Clear Color
- Compressed / Decompressed

In compressed state we can overwrite the contents of each original block. Everything's lossless compressed so we can restore the correct color per pixel.

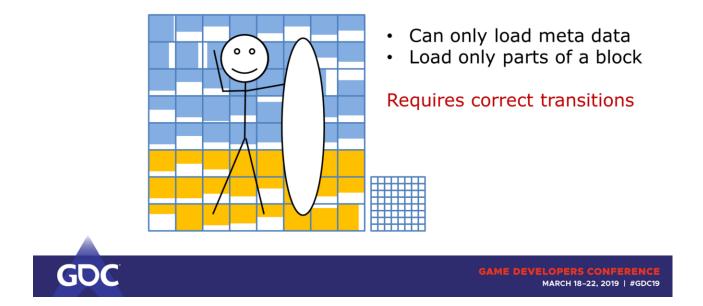
Can be done during creation of that texture, meaning rendering into it. Controlled by transitioning the texture, sometimes may need to decompress it.



The cool thing is:

- Can skip loading the actual pixels
- Or only load parts of a pixel block!

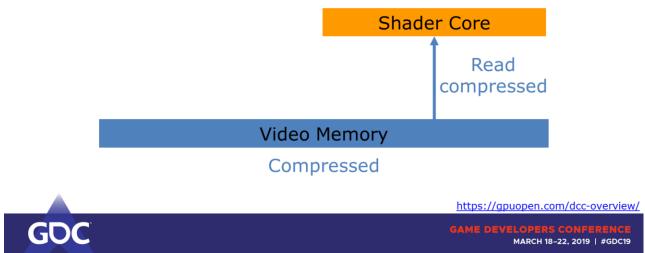
Make sure to transition correctly, or you may end up seeing the compressed blocks  $\rightarrow$  corruptions.



Make sure to transition correctly, or you may end up seeing the compressed blocks  $\rightarrow$  corruptions.

### Sample()

#### Decompressed

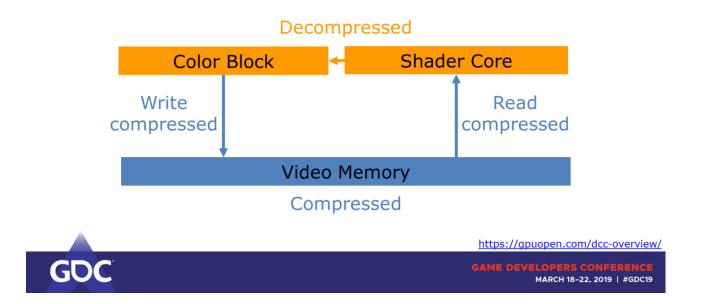


Cycle through sampling a texture and exporting color values to a render target.

 $\rightarrow$  We save on bandwidth both on read AND write to VMEM.

The texture itself isn't going to be stored more compactly – in fact we even need to attach more data to hold meta information!

### Export()



Cycle through sampling a texture and exporting color values to a render target.

 $\rightarrow$  We save on bandwidth both on read AND write to VMEM.

The texture itself isn't going to be stored more compactly – in fact we even need to attach more data to hold meta information!

#### Compression checklist:

- Use exclusive queue ownership
   With shared ownership the driver must assume that it's used
   on hardware blocks that can't read or write compressed

   Explicitly state image format
   UNKNOWN / MUTABLE can prevent compression
   Can work around with VK\_KHR\_image\_format\_list

   Use only the required image usages
   Otherwise the resource can end up
   in less than optimal compression levels
- Clear render/depth targets Clearing resets meta data



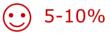
GAME DEVELOPERS CONFERENCE MARCH 18–22, 2019 | #GDC19

- Rendering many triangles into a target can turn a block to decompressed state. Clearing resets meta data and thus improves compression ratio.
- Shared ownership disables compression (certain blocks can't handle compression)
- Quirks with UNKNOWN/MUTABLE prevent compression. Can work around it with VK\_KHR\_image\_format\_list

- Important if many draws render to that target.

	Name	Format	Width	Height	Draw calls	Compression	Sample count	Out of order draw calls	Duration
>	Color RT #27	VK_FORMAT_R8G8B8A8_SRGB	1920	1080	2777	OFF	1	0 / 2777	2.414 ms
>	Color RT #28	VK_FORMAT_A2R 10G 10B 10_UNORM_PACK32	1920	1080	2766	OFF	1	0 / 2766	2.100 ms
>	Color RT #29	VK_FORMAT_R8G8B8A8_UNORM	1920	1080	2773	OFF	1	0 / 2773	2.170 ms
>	Color RT #30	VK_FORMAT_R8G8B8A8_UNORM	1920	1080	2774	OFF	1	0 / 2774	2.416 ms

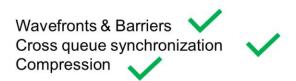
	Name	Format	Width	Height	Draw calls	Compression	Sample count	Out of order draw calls	Duration
>	Color RT #27	VK_FORMAT_R8G8B8A8_SRGB	1920	1080	2779	ON	1	0 / 2779	2.207 ms
>	Color RT #28	VK_FORMAT_A2R10G10B10_UNORM_PACK32	1920	1080	2768	ON	1	0 / 2768	1.959 ms
>	Color RT #29	VK_FORMAT_R8G8B8A8_UNORM	1920	1080	2775	ON	1	0 / 2775	2.014 ms
>	Color RT #30	VK_FORMAT_R8G8B8A8_UNORM	1920	1080	2776	ON	1	0 / 2776	2.218 ms





GAME DEVELOPERS CONFERENCE MARCH 18-22, 2019 | #GDC19

### Generous 10% win in this case.





GAME DEVELOPERS CONFERENCE MARCH 18-22, 2019 | #GDC19

## Wrap-Up

#### Barriers

- Batch barriers
- □ Use the proper pipeline stage flags
- Overlap independent work
- Use async compute queue





AME DEVELOPERS CONFERENCE MARCH 18–22, 2019 | #GDC19

## Wrap-Up

#### Cross queue synchronization

- Sync seldomlyPrefer large workloads



MARCH 18-22, 2019 | #GDC19



## Wrap-Up

#### Compression

- Use exclusive queue access
- Explicit image format
- □ Use only required image usage
- □ Clear render / depth targets

. ◄ ►	START	OVERVIEW	EVENTS						
Frame summary									
Barriers									
Host expensive events									
Context rolls			1.300 ms 4.400 m			00 ms	4.800 ms		
Render/depth targets		coor							
Device configuration									
		Color RT #30	Color RT #30	Color RT #30					
		Color RT #29 Color RT #28	Color RT #29	Color RT #29 Color RT #28					
		Color RT #27	Color RT #28 Color RT #27	Color RT #27					
		Depth							
		Depth							
		Depth RT #5	Depth RT #S	Depth RT #5					
		Other							
		2419	2506	+					
		¢							
		Expand tree	✓ Group by target	Select columns					
		Nan	e Formal		Width	Height	Draw call		
		> Colo		MAT_RBG888A8_SRG8	1920	1080	2147		
		> 📕 Colo		HAT_A2R 10G 108 10_UNORM_PACK32	1920	1080	2135		
		> 📕 Colo	RT #29 VK_FOR	MAT_REGEBBAS_UNORM	1920	1080	2143		
		> 📕 Colo	RT #30 VK_FOR	MAT_REGEBBAS_UNORM	1920	1080	2143		
		> Dept	h RT #6 VK_FOR	HAT_D32_SPLOAT_S8_UDNT	960	540	34		



GAME DEVELOPERS CONFERENCE MARCH 18-22, 2019 | #GDC19

## There is more

- More information on **barriers** <u>https://gpuopen.com/vulkan-barriers-explained/</u>
- More information on DCC <u>https://gpuopen.com/dcc-overview/</u>
- Check out RGPs new features on Wednesday 5pm!
- And a more detailed look at barriers by Matt Pettineo at 4pm today <sup>(2)</sup>



GAME DEVELOPERS CONFERENCE MARCH 18–22, 2019 | #GDC19

## Thanks!

- To the AMD tools team
- To all reviewers
- ... and to you Thank you for your attention!

Time for questions :)



AME DEVELOPERS CONFERENCE MARCH 18–22, 2019 | #GDC19