



Memory Management in the APEX Engine

Lou Kramer

AMD

Daniel Isheden

Avalanche Studios Group

Memory Management in the APEX Engine - | Digital Dragons

16-17 MAY 2022

















Daniel Isheden

Avalanche Studios Group

AMDJ

AGENDA

Basics about memory management and tooling

- Memory types
- Committed / Placed Resources
- Over-Commitment
- Application, Driver, Operating System
- Tools

Memory Management in the APEX Engine

- The APEX Engine
- Resource types
- Problems & Solutions

This talk will focus only on PC with a dedicated GPU

API specific terminology is coloured in





MEMORY TYPES – RECAP

Physically:

VB, IB, CBV, SRV, RT, DS, UAV, ...





MEMORY TYPES – RECAP

Logically:





VB, IB, CBV, SRV, RT, DS, UAV, ...

Default Heap Device Local





void* mappedPtr;

Upload Heap Host Visible | Host Coherent | Readback Heap Host Visible | Host Coherent | Host Cached



CREATE A RESOURCE





COMMITTED RESOURCE

- A separate memory allocation per committed resource
- Each memory allocation comes with an overhead:



• This can take up to several milliseconds (or even seconds)



COMMITTED RESOURCE

- Once the memory is allocated and the resource is created,
- they don't work any slower or different than a placed resource





• On Vulkan[®], there is also a limited maximum number of allocations (e.g., 4096)



PLACED RESOURCE

- A large memory block (e.g., 256 MiB) is allocated when needed
- Sub-allocate parts of them for the placed resource





PLACED RESOURCE





- Over commitment when your video memory is full
 - New allocations may fail
 - Existing allocations can be migrated to system memory
 - ightarrow performance degradation

- Some resources tend to have a high impact on performance
- You really don't want them to migrate to SysRAM, e.g., render targets



- When silently migrated, the whole allocation is affected and all resources associated with it
 - Not just a single resource. Not just a single memory page





- How we can prevent a performance critical resource to get migrated to system memory?
- There is no explicit control
- No way to query for when and what

Application

- Creates resources
- Destroys resources
- Sets a preferred heap
- Sets residency priority
- Knows about all resources and how they are used
- Can use Evict/Make Resident on Direct3D®12
 - Moving things is a fairly expensive operation and can cause stuttering

0 B	32 MiB	64 MiB	96 MiB	128 MiB	160 MiB	192 MiB	224 MiB
Memory type 0 Default pool block 0							
Default pool block 1							
Default pool block 2							
Default pool block 3							
Default pool block 4							
						T	
efault pool block 5							
efault pool block 6							
efault pool block 7							
efault pool block 8							
efault pool block 9							
efault pool block 1	0						
efault pool block 1	1 1						
ustom pool block 0							
Aemory type 1							
Default pool block 0							
efault pool block 1							



Driver

- Sets also residency priorities
- The driver allocates memory for implicit resources
 - Command buffers
 - Descriptors
 - Shader pipelines
 - Internal resources





Operating System (Microsoft[®]'s Video Memory Manager)

- Knows about other applications running in parallel
 - Ensures that each process receives a fair share^[1]
- Can migrate memory blocks to system memory
 - Ensures that the transition of video to system memory is invisible to the application^[2]

-1				
	3.352333000	4.076967300	CpuVisible (0x00	NONE
	3.477154000	3.753860000	CpuVisible (0x00	NONE
	3.827970100	4.105901800	CpuVisible (0x00	NONE
	3.857101600	4.194048800	CpuVisible (0x00	NONE
	3.886722500	4.673828100	CpuVisible (0x00	NONE
	3.945366400	4.461670600	CpuVisible (0x00	NONE
	4.299126700	9,223,372,036.85	CpuVisible (0x00	NONE
	4.540814200	4.823873300	CpuVisible (0x00	NONE
	5.027540200	9,223,372,036.85	CpuVisible (0x00	NONE
	5.148931900	9,223,372,036.85	CpuVisible (0x00	NONE
			1.3.4. Undergroup 0.5.0	

	Getting Started	Analysis						
em Activity	✓ Gpu Segment Usage Size by 5	ieqmentType, Adapterid, Segmentid * + 📔 👂 🍄						
esses Lifetime By	Process Series	1/	1	welk Outstanding Size using resource time as (Start Tin	na.End Time) (Aggregation: Sum)			
	· sneicopeiencenusca							
	P SearchULese (4003)							
	Kabeonseiningsteien							
utation	k wna.eve (5468)	26						
- A a Alada M	Conf Winfel DX12 R	20						
Martin Martin					Second and a second	and contraction of		
	5	Dog 02 04 05 05	10 12 14 16 19	20 27 24 24 29	0 32 14 24 14		48 50 51	54 58
•	Line # Adapter	Process Service	sent ld Start Time (c) Fr	ad Time (s) Flags Format	Width Height	Lisane Flans	VidMen Global	ine i
	1 b (1) Microsoft Basic Die	inlay Driver		a rate () rage		engi negi		946 175
1	2 = (2) AMD Radeon (TM)	RX 480						5 914 812 416
Λ	3	▶ Idle (0)						610,304
	4	csrss.exe (6408)						153.673.728
y.	5	P. dom.exe (1784)						280,408,064
	6	▶ explorer.exe (7768)						1,306,624
	7	ShellExperienceHost.exe (2004) 1	-1					11,988,992
mory Usage Size by	urpose B	SearchUl.exe (4668) P	-1					8,716,288
	9	RadeonSettings.exe (7036)						31,612,928
	10	Calculator.exe (2590) P	-1					4,485,120
ament Usage Size by SegmentTime Adapt	11	vvpa.exe (5468)						142,966,784
	12	Civ6_Win64_DX12_Release.exe (5, 196, 595, 200 🖂
	13		-		1			44,093,440
- I - I - I - I - I - I - I - I - I - I			1.152333000	4.076967300 CpuVisible (0x00 NONE	0	0	0xFFFF880051556	4,239,350
ferenced Memory - Frame Size	by Type 15		3.477154000	3.753860000 CpuVisible (0x00 NONE	0	0	OxFFFF8804F653	4,259,840
	16		3.827970100	4.105901800 CpuVisible (0400 NONE	0	0	0x7FFF88051879	4,239,840
	10		3.857101000	4.572030100 Couldistie (0x00 NONE	0	0	0.FFFF00030AC	9,690,704
	10		3.880/22.300	4 461670600 Could all the color Monte	0	0	0.6666890031730	4,557,004
terenced Memory - Second Size	by type 15		1 20012/200	222 222 224 PE Could be to 20 Alone		0	D.FFFFBBDEDEDC	4,400,730
	21		4 540814200	4 823873300 CouVisible (0x00 NONE	0	0	0xFEFE88051556	4 570 228
	22		5.027540200 9	223.372.036.85 CouVisible (0x00 NONE	0	0	DVFFFFR8051588	4 554 752
	hu line 23		5.145931900 9	223 372 036.85. CouVisible (0x00. NONE	0	0	0xFFFF880502FA	4 567 040
erenced Memory - All Size	1		5		-			1.191.796.736
erenced Memory - All Size	67							3.862,257,664
erenced Memory - All Size	25		2					
ferenced Memory - All Size	25 26	4	2					98,447,360
ferenced Memory - All Size	25 26 27	P excel.exe (6624)	2					98,447,360

▷ Idle (0)

csrss.exe (6408)
 dwm.exe (1784)

♦ explorer.exe (7768)

SearchUl.exe (4668)

Calculator.exe (2580)

wpa.exe (5468)

RadeonSettings.exe (7036)

ShellExperienceHost.exe (2004)

▼ Civ6 Win64 DX12 Release.exe (...

[1] https://docs.microsoft.com/en-us/windows-hardware/drivers/display/using-memory-segments-to-describe-the-gpu-address-space [2] https://docs.microsoft.com/en-us/windows-hardware/drivers/display/mapping-virtual-addresses-to-a-memory-segment



We can try to increase the likelihood for a performance critical resource to stay in VRAM by:

Having **enough free space** on the VRAM

- Query for budget and stick to it
- DXGI_QUERY_VIDEO_MEMORY_INFO
- VK_EXT_memory_budget → VkPhysicalDeviceMemoryBudgetPropertiesEXT
- Query regularly for usage and budget
- Try to stay at usage < budget



We can try to increase the likelihood for a performance critical resource to stay in VRAM by:

Having enough free space on the VRAM

- Free or evict memory blocks when possible before creating new resources
- Alias Memory





We can try to increase the likelihood for a performance critical resource to stay in VRAM by:

Having enough free space on the VRAM

- Place VB, IB, CBV that are read only once by the GPU to the upload heap (system memory)
 - Save memory for another copy of the resource
 - Can even save time that's needed for the transfer
 - Reading will be slower though
 - Good for buffers





: performance critical resource

: not performance critical resource

We can try to increase the likelihood for a performance critical resource to stay in VRAM by:

Create performance critical resources as committed resources

- After creating critical resources as committed, set them high residency priority
- No need to allocate a new big chunk of memory just the amount that is actually required gets allocated
 - increases chance there is still enough free space
- Critical resources are not scattered in different large allocated memory blocks
 - Whole memory block gets evicted and thus, everything that's in it
 - If every memory block contains a critical resource, you will always loose





We can try to increase the likelihood for a performance critical resource to stay in VRAM by:

Try out different memory block sizes for your placed resources

• The optimal size can vary depending on your specific case (e.g., 256 MiB, 64 MiB, ...)

If your memory is too fragmented:

• Create custom pools for certain resources

- e.g., resources that should certainly be freed when unloading a level
- When streaming resources in and out, fragmentation is expected
 - try to defragment





TOOLS

- VMA/D3D12MA JSON Dumps
- Radeon[™] Memory Visualizer
- Windows[®] Performance Analyzer



	Default pool block 1		
	Default pand March 2		
	Default prod block 3		
	Default pool block 4		a dan a n
	Default pool block 1		
	Default pool block #		
	Default panel block 9		
	Default pool block 10		
	behadt pool block 11		
	Default pool block 12		
	Default prod block 15		
	persan pool block 15		
	Default pool block 17		
	Default pool block 18		
	Default paul Much 19		
	Castern good Minch B		
	Memory type 1 Defendl proof block 2		
	Default pool block 1		
	Default pool block 2	-	
	Seland parel block 2		
	Default pool block 4		
	behavit poor block 5		
	Default pool block 8		
	Default part black 7		
	Default pand block 10		
	Default pool block 11		
	Default pool block 12		
	Default poor block 18		
	Default peor block 14		
	behauft pool block 15		
	behavit poor block 19		
	Default pool block 19		
	Default pard March 20		
	Default pool block 2		
	behadt pool block 22		
Re cal C	Manang Type 2 Dedicated attacation (1		
	Dedicated affectation 3		
Mara para hari	Dedicated allocation 2		
Hara para Landari Bara gara Nagara Man	Default pool block 8		
	Pelasti part Mork 1		
Manage and a set of the set of th	Default pool block 2		
behari goot book b	Menuery Type 2		
	Default pool block #		



VMA/D3D12MA JSON DUMPS

- VMA and D3D12MA are open source memory management libraries for Vulkan [®] and Direct3D[®] 12
- https://gpuopen.com/vulkan-memory-allocator/
- Both come with an auxiliary tool to visualize the internal state of the allocator
- Lists all the memory blocks for each heap and their size
- Shows the resources in each memory block
 - Resource size and type
- Shows free memory in each memory block
- ightarrow useful to analyse fragmentation
- → useful to determine if memory block size is a good fit for the application's resources





RADEON[™] MEMORY VISUALIZER

- AMD tool to get an insight in how applications use memory for graphics resources from a driver perspective
- <u>https://gpuopen.com/rmv/</u>
- Lists the available heaps and the resources placed in them including driver internal resources
- You can compare two snapshots to find memory leaks

		NE SNAMSHOT COM	PARE								
delta	Snapshe	t 2 🐀 Snapshot 0									
wek dezker.	Mently whi	th resources are present or absent ac	ross anapahats.								
	Pas	nurces unque to anapshot Snapshot	2 🔍 🚾	Resources	common to both se	apshots 🧧	Resources see	ue to snapshot Sna	pahat 0		
	33 (esource	, 161.31 MB	38,40	re sessurces, 5	28.08	4,0	00 resources, 1.82 l	e			
	~ Prafers	d heap 😪 Resource usage Filter b	ly size:	15000					trativine.		Search
	compare I	D Name	VUTLOE ADDRess	Scie	Preserved heap	Committee Dyname	e Converted loca	Committee host	unnapped	Childen	
	00	Resource 0961354356266797778	excoch/poo	3.89 KB	Invadale	2.09 88	0 bytes	0 bytes	0 bytes	Shader pipeline	
	00	Resource 9961058517059964098	0x100366000	4 bytes	Insbie	4 bytes	0 bytes	@ bytes	0 bytes	Event	
	00	Resource 9959900257549406413	0x10f716600	9.13 KB	Invisible	9.03 KB	0 bytes	0 bytes	0 bytes	Shader pipeline	
	0	Resource 9935699756132144503	01/2419(0000	354.00 Kill	Installe	0 bytes	0 bytes	0 bytes	384.50 KB	Testure	
	0	Resource 9958642018131433108	0#1835e008	44,47 KB	Involve	0 bytes	0 bytes	0 bytes	41.47 KB	Hultiuse buffer	
	00	Resource 995839019990411504	Dx10+667800	1.15 KR	Inveble	1.15 KB	0 bytes	0 bytes	0 bytes	Shader ppeline	
	00	Resource 9957640217009596555	Dx105d10000	2.43 KB	treadale	2.41 KB	0 bytes	0 bytes	o byces	Shader ppeline	
	00	Resource 9956473386064440075	0/104323000	3.89 KB	Invisible	3.99 X8	0 bytes	0 bytes	0 bytes	Shader pipeline	
	00	Resource 9956150907034946361	0x10c2ds008	4.55 KB	Invebie	4.55 KB	0 bytes	@ bytes	0 bytes	Shader pipeline	
	00	Resource 9934271158763291823	0x112142000	9,94 KB	Invadria	9.04 KB	0 bytes	0 bytes	0 bytes	Shader ppelme	
	00	Resource 9953731:466057775433	0x1109x2000	2.62 KB	Invisible	2.62 KB	0 bytes	0 bytes	0 bytes	Shader pipeline	
	0	Resource 9952913579730582531	DiGes3fe000	40.00 KB	Invisible	Obytes	0 bytes	0 bytes	40.00 KB	Testure	
	00	Resource 9952411528118487258	Dx102361000	2.85 KB	Institu	2.88 KB	0 bytes	0 bytes	0 bytas	Shadar ppelne	
	00	Resource 9952269475642683035	0x103576808	1.45 KB	Invisible	1.65 KB	0 bytes	0 bytes	0 bytes	Studer pipeline	
	0	Resource 9951725724620581505	91417780005	1.00 MB	Investrie	0 bytes	0 bytes	0 bytes	1.09 M6	Testure	
	00	Resource 9951343739671981183	Dr206563008	2.97 KB	Inistrie	2.97 KB	0 bytes	0 bytes	0 bytes	Shadar pipalina	
	00	Resource 9951131520490827004	0x1091ca008	6.15 KB	Invisible	6.05 KB	0 bytes	@ bytes	0 bytes	Shader pipeline	
	00	Resource 9930368987701317817	Dx1023e6000	1.45 KB	Irvedrie	1.65 88	0 bytes	0 bytes	0 bytes	Shader pipeline	
	0	Resource 9951240886006796214	Dx41588c000	23.71 KB	Invsbie	0 byces	0 bytes	0 bytes	23.71 KB	Hutsuse buffer	
	00	Resource 9958206149656551309	0102236000	1.80 KB	Involte	1.01 KB	0 bytes	0 bytes	0 bytes	Studer pipeline	
	00	Resource 9949248821575844356	0x111bas005	10.73 KB	Invisible	10.73 68	0 bytes	0 bytes	0 bytes	Stader ppelite	
	00	Resource 9949293369253857468	01202860900	1.65 KB	Invisible	1.65 KB	0 bytes	0 bytes	0 01/045	Shadar pipeline	
	00	Resource 9947244497777730228	0x10304c000	7.27 KB	Invisibile	7.27 88	0 bytes	0 bytes	0 bytes	Stader poeline	
	00	Resource 9946051054670203233	Dx102423008	2.42 KB	treakie	2.42 KB	0 bytes	d bytes	O bytes	Shader pipeline	
	00	Resource 9946484395940516869	0x1028cd800	1,65 KB	Insbe	1.65 KB	0 bytes	0 bytes	0 bytes	Shader ppeline	
	Total resour	cest 36,952 Total sze: 4.19 68									

START	TIMELINE SNAPSHOT COMPARE			
Heap overview	Local (Smart Access Memory is disabled)			
Resource overview Alocation overview Resource list Alocation explorer Resource details	The heap is in local (video) memory. It is mappable by the CP Total size 256.00 MB Requested [30.00 NB Bound] 32.01 KB Committed \$50.00 KB	U, but does not use the CPU cache. Physical location of heap: CPU Cached: GPU Cached: GPU Vable: Smallst allocation: Largest allocation: Hean allocation:	Video memory No Yes Yes 8.00 KB 256.00 KB 17.3.3 KB	Internal
	Invisible The heap is hoal (video) memory. It is not mappable by th Total size Requested Bound Sali GB Committed S.7.6 GB	e CPU. Physical location of heap: CPU Cached: GPU Cached: GPU Cached: GPU Vable: Smallest allocation: Largest allocation: Mean allocation:	Vdeo memory No Yes 122.00 KB 6.11 M6	Texture Render target Deph stend buffer Trick buffer Vertex buffer Other
You are currently viewing: - Snepshot 0 (126,235,883 clk)	Host The heap is in host (system) memory. It is intended for write Total size 23.0508 Requested 70.00 /08 Bound 123.03 /08 Committed 132.00 /08	-only data on the CPU side. Physical location of heap: S CPU Cached: CPU Vable: GPU Vable: Smales: allocation: Largest allocation: Mean allocation: 2	System memory Yes Yes Yes Koo KB 255.00 KB 118.00 KB	Command buffer Internal



WINDOWS® PERFORMANCE ANALYZER

 Windows[®] Performance Analyzer is part of the Windows[®] Performance Toolkit, which is part of the Windows[®] 10 SDK

- Shows all current processes
- Shows how much memory each process allocated
- Lists all evicted memory blocks from VRAM to SysRAM under GPU Segment -1.

Getting Started 1 Analy	sis										
▲ Gpu Segment Usage Size by SegmentTyp	pe, Adapterid, Segmentid * 👻 🛛 🌶	o 💠									
Series	66-			Peak Outstanding S	ize using resource tim	e as [Start Time,End	Time] (Aggregation:	: Sum)			
b Search III ave (4669)											
BadaanSattings ava	45										
Calculator are (2590)	40-										
V Calculator.exe (2500)											
= Civ6 Win64 DV12 P	26-										
	-										
	0 <u> </u>	chine in the second sec	Leiter	Truthin		drada	1111				
3	0.0 0.2 0.4 0.6	0.8 1.0 1	1.2 1.4 1.6	1.8 2.0 2.2	2.4 2.6	2.8 3.0	3.2 3.4	3.6 3.8 4.0	4.2 4.4 4.6	4.8 5.0 5	2 5.4 5.6 5.8
.ine # Adapter	Process	Segment Id	Start Time (s)	End Time (s)	Flags	Format	Width	Height	Usage Flags	VidMm Global	Size _{Sum} Legend
1 ▷ (1) Microsoft Basic Display Driver	r										946,176
2 🔻 (2) AMD Radeon (TM) RX 480											5,914,812,416
3	▶ Idle (0)										610,304
4	csrss.exe (6408)										153,673,728
5	◊ dwm.exe (1784)										280,408,064
6	explorer.exe (7768)										1,306,624
7	ShellExperienceHost.exe (2004)	Þ	-1								11,988,992
8	SearchUl.exe (4668)	Þ	-1								8,716,288
9	RadeonSettings.exe (7036)										31,612,928
10	Calculator.exe (2580)	Þ	-1								4,485,120
11	▶ wpa.exe (5468)										142,966,784
12	▼ Civ6_Win64_DX12_Release.exe (5,196,595,200
13		-	-1								44,093,440 🚞
14			3.352333000	4.07696730	0 CpuVisible (0x00.	NONE		0	0	0xFFFF88051556	4,239,360
15			3.477154000	3.75386000	0 CpuVisible (0x00.	NONE		0	0	0xFFFF8804F653	4,259,840
16			3.827970100	4.10590180	0 CpuVisible (0x00.	NONE		0	0	0xFFFF88051B79	4,259,840
17			3.857101600	4.19404880	0 CpuVisible (0x00.	NONE		0	0	0xFFFF88050AC	4,296,704
18			3.886722500	4.67382810	0 CpuVisible (0x00.	NONE		0	0	0xFFFF88051738	4,337,664
19			3.945366400	4.46167060	0 CpuVisible (0x00.	NONE		0	0	0xFFFF88051B4A	4,468,736
20			4.299126700	9,223,372,036.85.	. CpuVisible (0x00.	NONE		0	0	0xFFFF8805083C	4,530,176
21			4.540814200	4.82387330	0 CpuVisible (0x00.	NONE		0	0	0xFFFF88051556	4,579,328
22			5.027540200	9,223,372,036.85.	. CpuVisible (0x00.	. NONE		0	0	0xFFFF88051588	4,554,752
23			5.148931900	9,223,372,036.85.	. CpuVisible (0x00.	NONE		0	0	0xFFFF880502FA	4,567,040
24		Þ	5								1,191,796,736
25		Þ	3								3,862,257,664
26		Þ	2								98,447,360
27	excel.exe (6624)										82,448,384

• The other GPU Segments map to the heaps you see in RMV



DISCLAIMER & ATTRIBUTIONS

DISCLAIMER

The information contained herein is for informational purposes only and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale. GD-18

© 2022 Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo, Radeon, Ryzen, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Vulkan and the Vulkan logo are registered trademarks of the Khronos Group Inc. PCIe and PCI Express are registered trademarks of the PCI-SIG Corporation. DirectX is a registered trademark of Microsoft Corporation in the US and other jurisdictions.

Other product names used in this publication are for identification purposes only and may be trademarks of their respective owners.



AMDJ



Avalanche Studios Group





CONTRABAND



AVALANCHE OPEN WORLD ENGINE















AVALANCHE OPEN WORLD ENGINE









- Many categories of resources
- Varying properties
 - Size
 - Number of resources
 - Lifetime
 - CPU/GPU usage
 - Performance importance
- D3D12MA and VMA



Render targets

- GPU read/write
 - Performance critical
- Resolution-dependent
 - 500-2000 MB
- Created at startup
 - Occasionally resized







GPU storage buffers

- Temporary buffers
 - Pre-skinned vertices
 - Compute shader generated terrain mesh
 - GPU-generated vegetation instance data
- GPU read/write
 - Sometimes performance critical
- ~ 100 MB
- Some allocated on startup, some on demand

Buffer Contents									
Element	m_ModelIndex	m_LodIndex	m_VisibilityE	m_NumTypes	m_TypeId				
0			13	257					
1	216	1	13	257	56				
2	28	1	13	257	15				
3	18	1	13	257	94				
4	93	0	13	257	2				
5	28	1	13	257	249				
6	201	0	13	257	0				
7	18	2	13	257	170				
8	18	2	13	257	130				
9	216	2	13	257	64				
10	18	2	13	257	73				
11	18	3	13	257	8				
12	18	2	13	257	28				
13	18	2	13	257	91				
14	18	2	13	257	178				





Streamed vertex/index/material buffers

- GPU read-only
 - Copied from staging buffer
- Almost negligible size
- Streamed in and out







Streamed textures

- GPU read-only
 - Copied from staging buffer
- Loose memory budget
 - 500 2000+ MB
 - Not manual resource placement
 - Simply tracks total memory usage
 - Reference counted
- Large range of sizes
- Streamed in and out







Shader pipelines

- They do take up VRAM!
 - Driver-managed
 - Visible in Radeon Memory Visualizer!
- Many, many shader permutations
- Almost 100 MB
 - ~256 B 20 KB each





Constant buffers

- Camera matrices, constants, etc
- CPU generated each frame
- CPU RAM mapped memory
 - Caching on GPU hides cost
- Rotating buffers
 - Linear suballocator





Staging buffers

- Temporary buffers in RAM
- Used to initialize buffers/textures
- Similar system to constant buffers





Performance issues

- Sudden drops to <10 FPS
 - Often after window resize
- GPU-limited
 - Abnormally high PCI-E bus load?
- Performance critical resource spilled to RAM!
 - Confirmed with Radeon Memory Visualizer





Problems

- Memory usage spikes
- Fragmentation
- Simply running out of VRAM



Problem - Memory spikes

- Cannot destroy resource in use by GPU
 - Engine defers deletion for 1-2 frames
- Resized render targets?
 - Massive VRAM usage spike
- Radeon Memory Visualizer useful



Solution - Immediate resource destruction

- Wait for GPU to finish all pending work
- Delete resources immediately
- Then create new resources





Solution - Resource recreation

- Avoid delete, create, delete, create, ...!
- First delete all resources
- THEN recreate all resources
 - Avoids potential fragmentation





Problem - Fragmentation

- Resources streamed in and out
- Fragmentation
- Inflates memory usage
 - Higher risk of paging
- Critical resources mixed in
 - Risk being paged out with the entire block
- Radeon Memory Visualizer useful





Solution - Committed resources

- Separate out performance critical resources
 - Create as committed resources
- Individual residency
 - No longer causes or suffers from fragmentation
- Which resources?
 - Render targets
 - Unordered access textures/buffers
 - "Large" textures
- Reminder: Max allocation limit!



Solution - Defragmentation

- Background defragmentation
 - New D3D12MA feature
- No/few level switches
- Must update resource references
 - Resource tables/descriptor sets
 - Bindless resources





Problem - Simply running out of VRAM

- VRAM is scarce and highly contested
 - Multiple high resolution monitors
 - Heavy 3D artist software
 - Web browsers
 - Video recording software
 - etc...
- Spilling is still possible
 - Especially for our artists
- Can happen at any time
 - Want to minimize performance impact







3ds Max





Optimal resource priorities

- 1. Depth buffers
- 2. Render targets/unordered access textures
- 3. Compute shader intermediate buffers
- 4. Read-only textures
- 5. Read-only buffers



"Solution" - Assigning priorities

- Simple heuristic
 - High and normal priority resources
- Critical resources
 - Depth/color render targets
 - Unordered access textures/buffers
- Can only assign priorities to entire blocks
 - Critical resources are committed resources
- We don't use Evict()/MakeResident()
 - Resource unusable after eviction (*not* the same as paging)
 - Difficult to identify eviction candidates





Avalanche Studios Group











Lou Kramer



Daniel Isheden

Avalanche Studios Group