#### 

# **OPTIMISING A AAA VULKAN TITLE ON DESKTOP**

LOU KRAMER



#### LOU KRAMER

#### DEVELOPER TECHNOLOGY ENGINEER AMD



## THE GAME

First Vulkan game using the engine

Engine had existing DX11 and DX12 support on top of an internal rendering API

Once the Vulkan version was somewhat stable, we started to look at the performance side of things ③



## THE GAME

- Best practices
- -> hopefully minor changes only
- Other optimization opportunities?
- -> require probably a bit more work
- -> start early enough, can introduce new problems

## **BEST PRACTICES**

- Is compression enabled for the G-buffer render targets?
- How do the barriers look?
- Can we make use of the copy queue?
- What about the shader building infrastructure?
- ... usage flags, use of correct layouts, etc.

# **BEST PRACTICES**

- Is compression enabled for the G-buffer render targets?
- How do the barriers look?
- Can we make use of the copy queue?
- What about the shader building infrastructure?
- ... usage flags, use of correct layouts, etc.

This is a checklist you can follow through and verify for your own engine

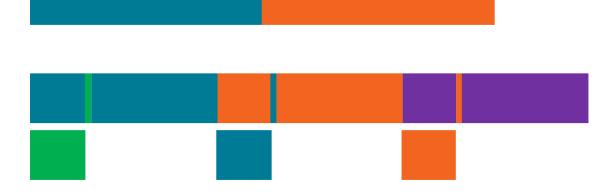
- Very engine specific
- In this particular case, there was a great **async compute** opportunity

- Very engine specific
- In this particular case, there was a great **async compute** opportunity

- Very engine specific
- In this particular case, there was a great **async compute** opportunity



- Very engine specific
- In this particular case, there was a great async compute opportunity



Vulkan specific feature



#### AGENDA

- DCC Delta Color Compression
- Other small things
- Q&A

#### AGENDA OR THE PREVIOUSLY MENTIONED CHECKLIST

- DCC Delta Color Compression
- Other small things
- Q&A

#### AGENDA OR THE PREVIOUSLY MENTIONED CHECKLIST

- DCC Delta Color Compression
- Other small things
- Q&A

+ async compute opportunity

# DCC – DELTA COLOR COMPRESSION

- What is DCC?
- Why do we want it
- -> Performance impact
- How to enable DCC?
- -> the journey of enabling DCC for this game 🏤

#### WHAT IS DCC?

- DCC Delta Color Compression
- Takes advantage of the fact that render targets tend to store slowly varying data
  - E.g. a blue sky will have little variance between the pixels

#### WHAT IS DCC?

- DCC Delta Color Compression
- Takes advantage of the fact that render targets tend to store slowly varying data
  - E.g. a blue sky will have little variance between the pixels



#### WHAT IS DCC?

- DCC Delta Color Compression
- Takes advantage of the fact that render targets tend to store slowly varying data
  - E.g. a blue sky will have little variance between the pixels



- Stores whole blocks one value is stored with full precision, rest is stored as delta
- It's lossless

## WHY DO WE WANT DCC?

- It's a bandwidth saver
- Take a special emphasis in enabling DCC for the G-buffer render targets
  - they usually benefit a lot from bandwidth savings

## WHY DO WE WANT DCC?

- It's a bandwidth saver
- Take a special emphasis in enabling DCC for the G-buffer render targets
  - they usually benefit a lot from bandwidth savings
- How much?
- Depends on workload and varies between graphics card
- But in this particular game title, we observed speed-ups on all tested AMD GPUs, ranging between
  - ~5 10%

## WHY DO WE WANT DCC?

- It's a bandwidth saver
- Take a special emphasis in enabling DCC for the G-buffer render targets
  - they usually benefit a lot from bandwidth savings
- How much?
- Depends on workload and varies between graphics card
- But in this particular game title, we observed speed-ups on all tested AMD GPUs, ranging between
  - ~5 10%



File View Help			
$\blacktriangleleft \triangleright$	START	OVERVIEW	EVENTS
Frame summary			
Barriers		<b>I</b> I C	Color pass #1
Most expensive events		0.000 ms	2.000 ms
Context rolls			
Render/depth targets		Color	
Pipelines			
Device configuration		Color RT # Color RT #	

File View Help			
	START	OVERVIEW	EVENTS
Frame summary			
Barriers		li C	olor pass #1
Most expensive events		0.000 ms	2.000 ms
Context rolls			1
Render/depth targets		Color	
Pipelines			
Device configuration		Color RT #	

Name	Format	Width	Height	Size in memory	Draw calls	Compression	Pixel wavefront ratio	Sample count	Out of order draw calls	Duration
Color RT #0	VK_FORMAT_R8G8B8A8_SRGB	3840	2160	32 MB	1874	OFF	178%	1	0 / 1874	5.044 ms
Color RT #1	VK_FORMAT_A2R10G10B10_UNORM_PACK32	3840	2160	32 MB	1577	OFF	178%	1	0 / 1577	3.761 ms
Color RT #2	VK_FORMAT_R8G8B8A8_UNORM	3840	2160	32 MB	1870	OFF	178%	1	0 / 1870	4.332 ms
Color RT #3	VK_FORMAT_R8G8B8A8_UNORM	3840	2160	32 MB	1871	OFF	178%	1	0 / 1871	4.671 ms

File View Help			
	START	OVERVIEW	EVENTS
Frame summary			
Barriers		<b>I</b> I C	Color pass #1
Most expensive events		0.000 ms	2.000 ms
Context rolls			
Render/depth targets		Color	
Pipelines			
Device configuration		Color RT # Color RT #	

Name	Format	Width	Height	Size in memory	Draw calls	Compression	Pixel wavefront ratio	Sample count	Out of order draw calls	Duration
Color RT #0	VK_FORMAT_R8G8B8A8_SRGB	3840	2160	32 MB	1874	OFF	178%	1	0 / 1874	5.044 ms
Color RT #1	VK_FORMAT_A2R10G10B10_UNORM_PACK32	3840	2160	32 MB	1577	OFF	178%	1	0 / 1577	3.761 ms
Color RT #2	VK_FORMAT_R8G8B8A8_UNORM	3840	2160	32 MB	1870	OFF	178%	1	0 / 1870	4.332 ms
Color RT #3	VK_FORMAT_R8G8B8A8_UNORM	3840	2160	32 MB	1871	OFF	178%	1	0 / 1871	4.671 ms

START	OVER	/IEW	EVENTS
	<b>I</b> 1	Cole	or pass #1
	0.000 ms	2	.000 ms
	Color		
	START	0.000 ms	Col 0.000 ms 2

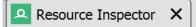
Name	Format	Width	Height	Size in memory	Draw calls	Compression	Pixel wavefront ratio	Sample count	Out of order draw calls	Duration
Color RT #0	VK_FORMAT_R8G8B8A8_SRGB	3840	2160	32 MB	1874	OFF	178%	1	0 / 1874	5.044 ms
Color RT #1	VK_FORMAT_A2R10G10B10_UNORM_PACK32	3840	2160	32 MB	1577	OFF	178%	1	0 / 1577	3.761 ms
Color RT #2	VK_FORMAT_R8G8B8A8_UNORM	3840	2160	32 MB	1870	OFF	178%	1	0 / 1870	4.332 ms
Color RT #3	VK_FORMAT_R8G8B8A8_UNORM	3840	2160	32 MB	1871	OFF	178%	1	0 / 1871	4.671 ms

- You can check the format
  - Float format
  - Integer format

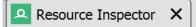
Name	Format	Width	Height	Size in memory	Draw calls	Compression	Pixel wavefront ratio	Sample count	Out of order draw calls	Duration
Color RT #0	VK_FORMAT_R8G8B8A8_SRGB	3840	2160	32 MB	1874	OFF	178%	1	0 / 1874	5.044 ms
Color RT #1	VK_FORMAT_A2R10G10B10_UNORM_PACK32	3840	2160	32 MB	1577	OFF	178%	1	0 / 1577	3.761 ms
Color RT #2	VK_FORMAT_R8G8B8A8_UNORM	3840	2160	32 MB	1870	OFF	178%	1	0 / 1870	4.332 ms
Color RT #3	VK_FORMAT_R8G8B8A8_UNORM	3840	2160	32 MB	1871	OFF	178%	1	0 / 1871	4.671 ms

- You can check the format
  - Float format
  - Integer format
- All of the below are supported

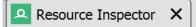
Name	Format	Width	Height	Size in memory	Draw calls	Compression	Pixel wavefront ratio	Sample count	Out of order draw calls	Duration
Color RT #0	VK_FORMAT_R8G8B8A8_SRGB	3840	2160	32 MB	1874	OFF	178%	1	0 / 1874	5.044 ms
Color RT #1	VK_FORMAT_A2R10G10B10_UNORM_PACK32	3840	2160	32 MB	1577	OFF	178%	1	0 / 1577	3.761 ms
Color RT #2	VK_FORMAT_R8G8B8A8_UNORM	3840	2160	32 MB	1870	OFF	178%	1	0 / 1870	4.332 ms
Color RT #3	VK_FORMAT_R8G8B8A8_UNORM	3840	2160	32 MB	1871	OFF	178%	1	0 / 1871	4.671 ms
							$\frown$			



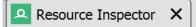
vkCreatelmage	
device	Device 10 d
✔ CreateInfo	VkImageCreateInfo()
sType	VK_STRUCTURE_TYPE_IMAGE_CREATE_INFO
pNext	NULL
flags	VK_IMAGE_CREATE_MUTABLE_FORMAT_BIT
imageType	VK_IMAGE_TYPE_2D
format	VK_FORMAT_R8G8B8A8_SRGB
> extent	VkExtent3D()
mipLevels	1
arrayLayers	1
samples	VK_SAMPLE_COUNT_1_BIT
tiling	VK_IMAGE_TILING_OPTIMAL
usage	VK_IMAGE_USAGE_TRANSFER_SRC_BIT   VK_IMAGE_USAGE_TRANSFER_DST_BIT   VK_IMAGE_USAGE_SAMPLED_BIT   VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT
sharingMode	VK_SHARING_MODE_EXCLUSIVE
queueFamilyIndexCount	0
pQueueFamilyIndices	uint32_t[]
initialLayout	VK_IMAGE_LAYOUT_UNDEFINED



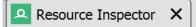
vkCreateImage	
device	Device 10 🖉
✔ CreateInfo	VkImageCreateInfo()
sType	VK_STRUCTURE_TYPE_IMAGE_CREATE_INFO
pNext	NULL
flags	VK_IMAGE_CREATE_MUTABLE_FORMAT_BIT
imageType	VK_IMAGE_TYPE_2D
format	VK_FORMAT_R8G8B8A8_SRGB
> extent	VkExtent3D()
mipLevels	1
arrayLayers	1
samples	VK_SAMPLE_COUNT_1_BIT
tiling	VK_IMAGE_TILING_OPTIMAL
usage	VK_IMAGE_USAGE_TRANSFER_SRC_BIT   VK_IMAGE_USAGE_TRANSFER_DST_BIT   VK_IMAGE_USAGE_SAMPLED_BIT   VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT
sharingMode	VK_SHARING_MODE_EXCLUSIVE
queueFamilyIndexCount	0
pQueueFamilyIndices	uint32_t[]
initialLayout	VK_IMAGE_LAYOUT_UNDEFINED



• vkCreatelmage							
device	Device 10 d						
✓ CreateInfo	VkImageCreateInfo()						
sType	VK_STRUCTURE_TYPE_IMAGE_CREATE_INFO						
pNext	NULL						
flags	VK_IMAGE_CREATE_MUTABLE_FORMAT_BIT						
imageType	VK_IMAGE_TYPE_2D						
format	VK_FORMAT_R8G8B8A8_SRGB						
> extent	VkExtent3D()						
mipLevels	1						
arrayLayers	1						
samples	VK_SAMPLE_COUNT_1_BIT						
tiling	VK_IMAGE_TILING_OPTIMAL						
usage	VK_IMAGE_USAGE_TRANSFER_SRC_BIT   VK_IMAGE_USAGE_TRANSFER_DST_BIT   VK_IMAGE_USAGE_SAMPLED_BIT   VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT						
sharingMode	VK_SHARING_MODE_EXCLUSIVE						
queueFamilyIndexCount	0						
pQueueFamilyIndices	uint32_t[]						
initialLayout	VK_IMAGE_LAYOUT_UNDEFINED						



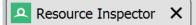
• vkCreateImage			
device	Device 10 a		
✓ CreateInfo	VkImageCreateInfo()		
sType	VK_STRUCTURE_TYPE_IMAGE_CREATE_INFO		
pNext	NULL		
flags	VK_IMAGE_CREATE_MUTABLE_FORMAT_BIT		
imageType	VK_IMAGE_TYPE_2D		
format	VK_FORMAT_R8G8B8A8_SRGB		
> extent	VkExtent3D()		
mipLevels	1		
arrayLayers	1		
samples	VK_SAMPLE_COUNT_1_BIT		
tiling	VK_IMAGE_TILING_OPTIMAL		
usage	VK_IMAGE_USAGE_TRANSFER_SRC_BIT   VK_IMAGE_USAGE_TRANSFER_DST_BIT   VK_IMAGE_USAGE_SAMPLED_BIT   VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT		
sharingMode			
queueFamilyIndexCount	0		
pQueueFamilyIndices	uint32_t[]		
initialLayout	VK_IMAGE_LAYOUT_UNDEFINED		



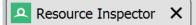
✔ vkCreateImage		
device	Device 10 🖉 VkImageCreateInfo()	
✓ CreateInfo		
sType	VK_STRUCTURE_TYPE_IMAGE_CREATE_INFO	
pNext	NULL	
flags	VK_IMAGE_CREATE_MUTABLE_FORMAT_BIT	
imageType	VK_IMAGE_TYPE_2D	
format	VK_FORMAT_R8G8B8A8_SRGB	
> extent	VkExtent3D()	
mipLevels	1	
arrayLayers	1	
samples	VK_SAMPLE_COUNT_1_BIT	
tiling	VK_IMAGE_TILING_OPTIMAL	
usage	VK_IMAGE_USAGE_TRANSFER_SRC_BIT   VK_IMAGE_USAGE_TRANSFER_DST_BIT   VK_IMAGE_USAGE_SAMPLED_BIT   VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT	
sharingMode		
queueFamilyIndexCount	0	
pQueueFamilyIndices	uint32_t[]	
initialLayout	VK_IMAGE_LAYOUT_UNDEFINED	

Retrieve some more resource details from RenderDoc:

1



vkCreateImage				
device	Device 10 $\mathscr{P}$			
✓ CreateInfo	VkImageCreateInfo()			
sType	VK_STRUCTURE_TYPE_IMAGE_CREATE_INFO			
pNext	NULL	VK_IMAGE_CREATE_MUTABLE_FORMAT_BIT		
flags	VK_IMAGE_CREATE_MUTABLE_FORMAT_BIT			
imageType	VK_IMAGE_TYPE_2D	disables DCC		
format	VK_FORMAT_R8G8B8A8_SRGB			
> extent	VkExtent3D()			
mipLevels	1			
arrayLayers	1			
samples	VK_SAMPLE_COUNT_1_BIT			
tiling	VK_IMAGE_TILING_OPTIMAL			
usage	VK_IMAGE_USAGE_TRANSFER_SRC_BIT   VK_IMAGE_USAGE_TRANSFER_DST_BIT   VK_IMAGE_USAGE_SAMPLED_BIT   VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT			
sharingMode	VK_SHARING_MODE_EXCLUSIVE			
queueFamilyIndexCount	0			
pQueueFamilyIndices	uint32_t[]			
initialLayout	VK_IMAGE_LAYOUT_UNDEFINED			



device	Device 10 d		
✓ CreateInfo	VkImageCreateInfo()		
sType	VK_STRUCTURE_TYPE_IMAGE_CREATE_INFO		
pNext	NULL	VK_IMAGE_CREATE_MUTABLE_FORMAT_BIT	
flags	VK_IMAGE_CREATE_MUTABLE_FORMAT_BIT		
imageType	VK_IMAGE_TYPE_2D	disables DCC	
format	VK_FORMAT_R8G8B8A8_SRGB	WHY	
> extent	VkExtent3D()	VV.	
mipLevels	1		
arrayLayers	1		
samples	VK_SAMPLE_COUNT_1_BIT		
tiling	VK_IMAGE_TILING_OPTIMAL		
usage	VK_IMAGE_USAGE_TRANSFER_SRC_BIT   VK_IMAGE_USAGE_TRANSFER_DST_BIT   VK_IMAGE_USAGE_SAMPLED_BIT   VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT		
sharingMode	VK_SHARING_MODE_EXCLUSIVE		
queueFamilyIndexCount	0		
pQueueFamilyIndices	uint32_t[]		
initialLayout	VK_IMAGE_LAYOUT_UNDEFINED		

#### VK\_IMAGE\_CREATE\_MUTABLE\_FORMAT\_BIT

DCC only works for float **XOR** integer formats -> R8G8B8A8\_SFLOAT, DCC is supported -> R8G8B8A8\_UNORM, DCC is supported

Etc.

#### VK\_IMAGE\_CREATE\_MUTABLE\_FORMAT\_BIT

DCC only works for float **XOR** integer formats -> R8G8B8A8\_SFLOAT, DCC is supported -> R8G8B8A8\_UNORM, DCC is supported

#### Etc.

```
How does the driver know the format of the image?
VkImageCreateInfo imageCreateInfo = {};
imageCreateInfo.format = VK_FORMAT_R8G8B8A8_SRGB;
```

#### VK\_IMAGE\_CREATE\_MUTABLE\_FORMAT\_BIT

DCC only works for float **XOR** integer formats -> R8G8B8A8\_SFLOAT, DCC is supported -> R8G8B8A8\_UNORM, DCC is supported

#### Etc.

```
How does the driver know the format of the image?
VkImageCreateInfo imageCreateInfo = {};
imageCreateInfo.format = VK_FORMAT_R8G8B8A8_SRGB;
```

What happens when the mutable bit is set?

VkImageCreateInfo imageCreateInfo = {};

imageCreateInfo.format = VK FORMAT R8G8B8A8 SRGB;

imageCreateInfo.flags = VK IMAGE CREATE MUTABLE FORMAT BIT;

### Spec:

"VK\_IMAGE\_CREATE\_MUTABLE\_FORMAT\_BIT specifies that the image can be used to create a VkImageView with a **different format** from the image."

```
VkImageCreateInfo imageCreateInfo = {};
imageCreateInfo.format = VK_FORMAT_R8G8B8A8_SRGB;
imageCreateInfo.flags = VK IMAGE CREATE MUTABLE FORMAT BIT;
```

### Spec:

"VK\_IMAGE\_CREATE\_MUTABLE\_FORMAT\_BIT specifies that the image can be used to create a VkImageView with a **different format** from the image."

-> The driver can't rely on the format information from the VkImageCreateInfo struct anymore

-> The driver can't rely on the format information from the VkImageCreateInfo struct anymore

For **float XOR integer**, the driver needs to distinguish between:

- 1. Image views with integer **AND** float formats are used on the image -> DCC must be **disabled**
- 2. Unsupported format is used -> DCC must be **disabled**
- 3. Only integer formats are used, e.g. UNORM and SRGB -> DCC can be enabled
- 4. Only float formats are used -> DCC can be enabled

The driver can't know if enabling DCC is safe by simply looking at the mutable bit.



-> The driver can't rely on the format information from the VkImageCreateInfo struct anymore

For **float XOR integer**, the driver needs to distinguish between:

- 1. Image views with integer **AND** float formats are used on the image -> DCC must be **disabled**
- 2. Unsupported format is used -> DCC must be **disabled**
- 3. Only integer formats are used, e.g. UNORM and SRGB -> DCC can be enabled
- 4. Only float formats are used -> DCC can be enabled

The driver can't know if enabling DCC is safe by simply looking at the mutable bit.



-> The driver can't rely on the format information from the VkImageCreateInfo struct anymore

For **float XOR integer**, the driver needs to distinguish between:

- 1. Image views with integer **AND** float formats are used on the image -> DCC must be **disabled**
- 2. Unsupported format is used -> DCC must be **disabled**
- 3. Only integer formats are used, e.g. UNORM and SRGB -> DCC can be enabled
- 4. Only float formats are used -> DCC can be enabled

The driver can't know if enabling DCC is safe by simply looking at the mutable bit.



The driver can't know if enabling DCC is safe by simply looking at the mutable bit. -> provide additional information by using

VK\_KHR\_image\_format\_list

typedef struct VkImageFormatListCreateInfoKHR {

VkStructureType	sType;
const void*	pNext;
uint32_t	viewFormatCount;
const VkFormat*	pViewFormats;

} VkImageFormatListCreateInfoKHR;

```
VkImageFormatListCreateInfoKHR imageFormatList = {};
imageFormatList.sType = VK_STRUCTURE_TYPE_IMAGE_FORMAT_LIST_CREATE_INFO_KHR
imageFormatList.pNext = ... ;
imageFormatList.viewFormatCount = formatCount;
imageFormatList.pViewFormats = formats; // array of VkFormat
```



```
VkImageCreateInfo imageCreateInfo = {};
imageCreateInfo.format = VK_FORMAT_R8G8B8A8_SRGB;
imageCreateInfo.flags = VK_IMAGE_CREATE_MUTABLE_FORMAT_BIT;
imageCreateInfo.pNext = &imageFormatList;
```

...

### 44 | VULKANISED | 2019

### VK\_IMAGE\_CREATE\_MUTABLE\_FORMAT\_BIT

vkCreateImage							
device	Device 10 🖋						
✔ CreateInfo	VkImageCreateInfo()						
sType	VK_STRUCTURE_TYPE_IMAGE_CREATE_INFO						
✓ pNext	VkImageFormatListCreateInfoKHR()						
sType	VK_STRUCTURE_TYPE_IMAGE_FORMAT_LIST_CREATE_INFO_KHR						
pNext	NULL						
viewFormatCount	2						
✓ pViewFormats	VkFormat[]						
[0]	VK_FORMAT_R8G8B8A8_UNORM						
[1]	VK_FORMAT_UNDEFINED						
flags	VK_IMAGE_CREATE_MUTABLE_FORMAT_BIT						
imageType	VK_IMAGE_TYPE_2D						
format	VK_FORMAT_R8G8B8A8_SRGB						
> extent	VkExtent3D()						
mipLevels	1						
arrayLayers	1						
samples	VK_SAMPLE_COUNT_1_BIT						
tiling VK_IMAGE_TILING_OPTIMAL							
usage VK_IMAGE_USAGE_TRANSFER_SRC_BIT   VK_IMAGE_USAGE_TRANSFER_DST_BIT   VK_IMAGE_USAGE_SAMPLED_BIT   VK_IMAGE_USAGE_COLOR_ATTACHMEN							
sharingMode	VK_SHARING_MODE_CONCURRENT						
queueFamilyIndexCount	3						
> pQueueFamilyIndices	uint32_t[]						
initialLayout	VK_IMAGE_LAYOUT_UNDEFINED						

# DOUBLE-CHECK IF THE CHANGE HAD THE INTENDED EFFECT ...

File View Help			
	START	OVERVIEW	EVENTS
Frame summary			
Barriers		Ca	olor pass #1
Most expensive events		0.000 ms	2.000 ms
Context rolls			
Render/depth targets		Color	
Pipelines			
Device configuration		Color RT #: Color RT #:	
			2

Name	Format	Width	Height	Size in memory	Draw calls	Compression	Pixel wavefront ratio	Sample count	Out of order draw calls	Duration
Color RT #0	VK_FORMAT_R8G8B8A8_SRGB	3840	2160	32 MB	1874	OFF	178%	1	0 / 1874	5.044 ms
Color RT #1	VK_FORMAT_A2R10G10B10_UNORM_PACK32	3840	2160	32 MB	1577	OFF	178%	1	0 / 1577	3.761 ms
Color RT #2	VK_FORMAT_R8G8B8A8_UNORM	3840	2160	32 MB	1870	OFF	178%	1	0 / 1870	4.332 ms
Color RT #3	VK_FORMAT_R8G8B8A8_UNORM	3840	2160	32 MB	1871	OFF	178%	1	0 / 1871	4.671 ms

# DOUBLE-CHECK IF THE CHANGE HAD THE INTENDED EFFECT ...



le View Help			
	START	OVERVIEW	EVENTS
rame summary			
Barriers			olor pass #1
lost expensive events		0.000 ms	2.000 ms
Context rolls			
Render/depth targets		Color	
Pipelines			
Device configuration		Color RT #3	

Name	Format	Width	Height	Size in memory	Draw calls	Compression	Pixel wavefront ratio	Sample count	Out of order draw calls	Duration
Color RT #0	VK_FORMAT_R8G8B8A8_SRGB	3840	2160	32 MB	1874	OFF	178%	1	0 / 1874	5.044 ms
Color RT #1	VK_FORMAT_A2R10G10B10_UNORM_PACK32	3840	2160	32 MB	1577	OFF	178%	1	0 / 1577	3.761 ms
Color RT #2	VK_FORMAT_R8G8B8A8_UNORM	3840	2160	32 MB	1870	OFF	178%	1	0 / 1870	4.332 ms
Color RT #3	VK_FORMAT_R8G8B8A8_UNORM	3840	2160	32 MB	1871	OFF	178%	1	0 / 1871	4.671 ms

# DOUBLE-CHECK IF THE CHANGE HAD THE INTENDED EFFECT ...



File View Help					_
	START	OVER	VIEW	EVENTS	
Frame summary					
Barriers		ll I	Colo	or pass #1	-
Most expensive events		0.000 ms	2	.000 ms	
Context rolls					1
Render/depth targets		Color			
Pipelines					
Device configuration			Color RT #3 Color RT #2		

Name	Format	Width	Height	Size in memory	Draw calls	Compression	Pixel wavefront ratio	Sample count	Out of order draw calls	Duration
Color RT #0	VK_FORMAT_R8G8B8A8_SRGB	3840	2160	32 MB	1874	OFF	178%	1	0 / 1874	5.044 ms
Color RT #1	VK_FORMAT_A2R10G10B10_UNORM_PACK32	3840	2160	32 MB	1577	OFF	178%	1	0 / 1577	3.761 ms
Color RT #2	VK_FORMAT_R8G8B8A8_UNORM	3840	2160	32 MB	1870	OFF	178%	1	0 / 1870	4.332 ms
Color RT #3	VK_FORMAT_R8G8B8A8_UNORM	3840	2160	32 MB	1871	OFF	178%	1	0 / 1871	4.671 ms

### 48 | VULKANISED | 2019

<ul> <li>vkCreateImage</li> </ul>						
device	Device 10 🖋					
✔ CreateInfo	VkImageCreateInfo()					
sType	VK_STRUCTURE_TYPE_IMAGE_CREATE_INFO					
✓ pNext	VkImageFormatListCreateInfoKHR()					
sType	VK_STRUCTURE_TYPE_IMAGE_FORMAT_LIST_CREATE_INFO_KHR					
pNext	NULL					
viewFormatCount	2					
✓ pViewFormats	VkFormat[]					
[0]	VK_FORMAT_R8G8B8A8_UNORM					
[1]	VK_FORMAT_UNDEFINED					
flags	VK_IMAGE_CREATE_MUTABLE_FORMAT_BIT					
imageType	VK_IMAGE_TYPE_2D					
format	VK_FORMAT_R8G8B8A8_SRGB					
> extent	VkExtent3D()					
mipLevels	1					
arrayLayers	1					
samples	VK_SAMPLE_COUNT_1_BIT					
tiling	VK_IMAGE_TILING_OPTIMAL					
usage	VK_IMAGE_USAGE_TRANSFER_SRC_BIT   VK_IMAGE_USAGE_TRANSFER_DST_BIT   VK_IMAGE_USAGE_SAMPLED_BIT   VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT					
sharingMode	VK_SHARING_MODE_CONCURRENT					
queueFamilyIndexCount	3					
> pQueueFamilyIndices	uint32_t[]					
initialLayout	VK_IMAGE_LAYOUT_UNDEFINED					

### 49 | VULKANISED | 2019

vkCreateImage	
device	Device 10 🖋
✔ CreateInfo	VkImageCreateInfo()
sType	VK_STRUCTURE_TYPE_IMAGE_CREATE_INFO
✔ pNext	VkImageFormatListCreateInfoKHR()
sType	VK_STRUCTURE_TYPE_IMAGE_FORMAT_LIST_CREATE_INFO_KHR
pNext	NULL
viewFormatCount	2
✓ pViewFormats	VkFormat[]
[0]	VK_FORMAT_R8G8B8A8_UNORM
[1]	VK_FORMAT_UNDEFINED
flags	VK_IMAGE_CREATE_MUTABLE_FORMAT_BIT
imageType	VK_IMAGE_TYPE_2D
format	VK_FORMAT_R8G8B8A8_SRGB
> extent	VkExtent3D()
mipLevels	1
arrayLayers	1
samples	VK_SAMPLE_COUNT_1_BIT
tiling	VK_IMAGE_TILING_OPTIMAL
usage	VK_IMAGE_USAGE_TRANSFER_SRC_BIT   VK_IMAGE_USAGE_TRANSFER_DST_BIT   VK_IMAGE_USAGE_SAMPLED_BIT   VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT
sharingMode	VK_SHARING_MODE_CONCURRENT
queueFamilyIndexCount	3
> pQueueFamilyIndices	uint32_t[]
initialLayout	VK_IMAGE_LAYOUT_UNDEFINED

vkCreateImage						
device	Device 10 🖋					
✔ CreateInfo	VkImageCreateInfo()					
sType	VK_STRUCTURE_TYPE_IMAGE_CREATE_INFO					
✔ pNext	VkImageFormatListCreateInfoKHR()					
sType	VK_STRUCTURE_TYPE_IMAGE_FORMAT_LIST_CREATE_INFO_KHR					
pNext	NULL					
viewFormatCount	2					
✓ pViewFormats	VkFormat[]					
[0]	VK_FORMAT_R8G8B8A8_UNORM					
[1]	VK_FORMAT_UNDEFINED					
flags	VK_IMAGE_CREATE_MUTABLE_FORMAT_BIT					
imageType	VK_IMAGE_TYPE_2D					
format	VK_FORMAT_R8G8B8A8_SRGB					
> extent	VkExtent3D()					
mipLevels	1					
arrayLayers	1					
samples	VK_SAMPLE_COUNT_1_BIT					
tiling	VK_IMAGE_TILING_OPTIMAL					
usage	VK_IMAGE_USAGE_TRANSFER_SRC_BIT   VK_IMAGE_USAGE_TRANSFER_DST_BIT   VK_IMAGE_USAGE_SAMPLED_BIT   VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT					
sharingMode	VK_SHARING_MODE_CONCURRENT					
queueFamilyIndexCount	3					
> pQueueFamilyIndices	uint32_t[]					
initialLayout	VK_IMAGE_LAYOUT_UNDEFINED					

<ul> <li>vkCreateImage</li> </ul>	
device	Device 10 🖉
✔ CreateInfo	VkImageCreateInfo()
sType	VK_STRUCTURE_TYPE_IMAGE_CREATE_INFO
✓ pNext	VkImageFormatListCreateInfoKHR()
sType	VK_STRUCTURE_TYPE_IMAGE_FORMAT_LIST_CREATE_INFO_KHR
pNext	NULL
viewFormatCount	2
✓ pViewFormats	VkFormat[]
[0]	VK_FORMAT_R8G8B8A8_UNORM
[1]	VK_FORMAT_UNDEFINED
flags	VK_IMAGE_CREATE_MUTABLE_FORMAT_BIT
imageType	VK_IMAGE_TYPE_2D
format	VK_FORMAT_R8G8B8A8_SRGB
> extent	VkExtent3D()
mipLevels	1
arrayLayers	1
samples	VK_SAMPLE_COUNT_1_BIT
tiling	VK_IMAGE_TILING_OPTIMAL
usage	VK_IMAGE_USAGE_TRANSFER_SRC_BIT   VK_IMAGE_USAGE_TRANSFER_DST_BIT   VK_IMAGE_USAGE_SAMPLED_BIT   VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT
sharingMode	VK_SHARING_MODE_CONCURRENT
queueFamilyIndexCount	3
> pQueueFamilyIndices	uint32_t[]
initialLayout	VK_IMAGE_LAYOUT_UNDEFINED

vkCreateImage	
device	Device 10 🖉
✔ CreateInfo	VkImageCreateInfo()
sType	VK_STRUCTURE_TYPE_IMAGE_CREATE_INFO
✓ pNext	VkImageFormatListCreateInfoKHR()
sType	VK_STRUCTURE_TYPE_IMAGE_FORMAT_LIST_CREATE_INFO_KHR
pNext	NULL
viewFormatCount	2
✓ pViewFormats	VkFormat[]
[0]	VK_FORMAT_R8G8B8A8_UNORM
[1]	VK_FORMAT_UNDEFINED
flags	VK_IMAGE_CREATE_MUTABLE_FORMAT_BIT
imageType	VK_IMAGE_TYPE_2D
format	VK_FORMAT_R8G8B8A8_SRGB
> extent	VkExtent3D()
mipLevels	1
arrayLayers	1
samples	VK_SAMPLE_COUNT_1_BIT
tiling	VK_IMAGE_TILING_OPTIMAL
usage	VK_IMAGE_USAGE_TRANSFER_SRC_BIT   VK_IMAGE_USAGE_TRANSFER_DST_BIT   VK_IMAGE_USAGE_SAMPLED_BIT   VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT
sharingMode	VK_SHARING_MODE_CONCURRENT
queueFamilyIndexCount	3
> pQueueFamilyIndices	uint32_t[]
initialLayout	VK_IMAGE_LAYOUT_UNDEFINED

<ul> <li>vkCreateImage</li> </ul>							
device	Device 10 d						
✓ CreateInfo	VkImageCreateInfo()						
sType	VK_STRUCTURE_TYPE_IMAGE_CREATE_INFO						
✓ pNext	VkImageFormatListCreateInfoKHR()						
sType	VK_STRUCTURE_TYPE_IMAGE_FORMAT_LIST_CREATE_INFO	O_KHR					
pNext	NULL	Async compute support was added to the engine					
viewFormatCount	2	royne compare support was added to the origine					
✓ pViewFormats	VkFormat[]						
[0]	VK_FORMAT_R8G8B8A8_UNORM	As a side-effect, now all resources have by					
[1]	VK_FORMAT_UNDEFINED						
flags	VK_PORMAI_ONDEPINED         VK_IMAGE_CREATE_MUTABLE_FORMAT_BIT    default sharing mode concurrent						
imageType	VK_IMAGE_TYPE_2D						
format	VK_FORMAT_R8G8B8A8_SRGB						
> extent	VkExtent3D()						
mipLevels	1						
arrayLayers	1						
samples	VK_SAMPLE_COUNT_1_BIT						
tiling	VK_IMAGE_TILING_OPTIMAL						
usage	VK_IMAGE_USAGE_TRANSFER_SRC_BIT   VK_IMAGE_USAG	;e_TRANSFER_DST_BIT   VK_IMAGE_USAGE_SAMPLED_BIT   VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT 🗸					
sharingMode	VK_SHARING_MODE_CONCURRENT	<b>*</b>					
queueFamilyIndexCount	3						
> pQueueFamilyIndices	uint32_t[]						
initialLayout	VK_IMAGE_LAYOUT_UNDEFINED						

<ul> <li>vkCreateImage</li> <li>device</li> </ul>	Device 10 🔗						
✓ CreateInfo	VkImageCreateInfo()						
sType	VK_STRUCTURE_TYPE_IMAGE_CREATE_INFO						
✓ pNext	VkImageFormatListCreateInfoKHR()						
sType	VK_STRUCTURE_TYPE_IMAGE_FORMAT_LIST_CREATE_INFO_KH	1R					
pNext	NULL	Async compute support was added to the engine					
viewFormatCount	2	regine compare support mae added to the engin					
✓ pViewFormats	VkFormat[]						
[0]	VK_FORMAT_R8G8B8A8_UNORM	As a side-effect, now all resources have by					
[1]	VK_FORMAT_UNDEFINED						
flags	VK_FORMAT_UNDEFINED         VK_IMAGE_CREATE_MUTABLE_FORMAT_BIT    default sharing mode concurrent						
imageType	VK_IMAGE_TYPE_2D						
format	VK_FORMAT_R8G8B8A8_SRGB						
> extent	VkExtent3D()						
mipLevels	1						
arrayLayers	1						
samples	VK_SAMPLE_COUNT_1_BIT						
tiling	VK_IMAGE_TILING_OPTIMAL						
usage	VK_IMAGE_USAGE_TRANSFER_SRC_BIT   VK_IMAGE_USAGE_TR	ANSFER_DST_BIT   VK_IMAGE_US					
sharingMode	VK_SHARING_MODE_CONCURRENT						
queueFamilyIndexCount	3						
> pQueueFamilyIndices	uint32_t[]						
initialLayout	VK_IMAGE_LAYOUT_UNDEFINED						

Spec:

"VK\_SHARING\_MODE\_CONCURRENT specifies that concurrent access to any range or image subresource of the object from multiple queue families is supported."

Spec:

"VK\_SHARING\_MODE\_CONCURRENT specifies that concurrent access to any range or image subresource of the object from multiple queue families is supported."

With VK\_SHARING\_MODE\_CONCURRENT **DCC** is disabled

Spec:

"VK\_SHARING\_MODE\_CONCURRENT specifies that concurrent access to any range or image subresource of the object from multiple queue families is supported."

### With VK\_SHARING\_MODE\_CONCURRENT DCC is disabled



### Quick side note on async compute ③



Improved performance of up to ~10%

### Quick side note on async compute ③



Improved performance of up to ~10%

What about DCC?

How to go back to VK\_SHARING\_MODE\_EXCLUSIVE to get DCC enabled? -> Obviously, if a resource is accessed only by **one** queue, just switch back to EXCLUSIVE

But what about resources, which are accessed by several queue families? -> transfer queue family ownership

### TRANSFER QUEUE FAMILY OWNERSHIP

Done in 2 steps

- 1. **Release** the exclusive ownership from the **source** queue family
- 2. Acquire the exclusive ownership for the destination queue family

### TRANSFER QUEUE FAMILY OWNERSHIP

Done in 2 steps

- 1. **Release** the exclusive ownership from the **source** queue family
- 2. Acquire the exclusive ownership for the destination queue family

Example:

Queue family 0 holds currently the exclusive ownership of image A Queue family 1 wants to acquire exclusive ownership of image A

```
VkImageMemoryBarrier imageMemoryBarrier = { };
imageMemoryBarrier.sType = VK STRUCTURE TYPE IMAGE MEMORY BARRIER;
imageMemoryBarrier.srcAccessMask = ...
imageMemoryBarrier.dstAccessMask = 0;
imageMemoryBarrier.oldLayout = oldLayoutImageA;
imageMemoryBarrier.newLayout = newLayoutImageA;
imageMemoryBarrier.srcQueueFamilyIndex = 0;
imageMemoryBarrier.dstQueueFamilyIndex = 1;
imageMemoryBarrier.image = imageA;
imageMemoryBarrier .subresourceRange = subresourceRangeImageA;
...
vkCmdPipelineBarrier(cmdBuf, ...);
...
vkQueueSubmit(queueFamily0,..., submitInfo, ...);
```

```
VkImageMemoryBarrier imageMemoryBarrier = { };
imageMemoryBarrier.sType = VK STRUCTURE TYPE IMAGE MEMORY BARRIER;
imageMemoryBarrier.srcAccessMask = ...
imageMemoryBarrier.dstAccessMask = 0;
imageMemoryBarrier.oldLayout = oldLayoutImageA;
imageMemoryBarrier.newLayout = newLayoutImageA;
imageMemoryBarrier.srcQueueFamilyIndex = 0;
imageMemoryBarrier.dstQueueFamilyIndex = 1;
imageMemoryBarrier.image = imageA;
imageMemoryBarrier .subresourceRange = subresourceRangeImageA;
                              Associated to a commandPool
...
vkCmdPipelineBarrier(cmdBuf, ...);
...
vkQueueSubmit(queueFamily0,..., submitInfo, ...);
```

```
VkImageMemoryBarrier imageMemoryBarrier = { };
imageMemoryBarrier.sType = VK STRUCTURE TYPE IMAGE MEMORY BARRIER;
imageMemoryBarrier.srcAccessMask = ...
imageMemoryBarrier.dstAccessMask = 0;
imageMemoryBarrier.oldLayout = oldLayoutImageA;
imageMemoryBarrier.newLayout = newLayoutImageA;
imageMemoryBarrier.srcQueueFamilyIndex = 0;
imageMemoryBarrier.dstQueueFamilyIndex = 1;
imageMemoryBarrier.image = imageA;
imageMemoryBarrier .subresourceRange = subresourceRangeImageA;
                               Associated to a commandPool
                                                                Associated to queue family 0
...
vkCmdPipelineBarrier(cmdBuf, ...);
...
vkQueueSubmit(queueFamily0,..., submitInfo, ...);
```

```
VkImageMemoryBarrier imageMemoryBarrier = { };
imageMemoryBarrier.sType = VK STRUCTURE TYPE IMAGE MEMORY BARRIER;
imageMemoryBarrier.srcAccessMask = ...
imageMemoryBarrier.dstAccessMask = 0;
imageMemoryBarrier.oldLayout = oldLayoutImageA;
imageMemoryBarrier.newLayout = newLayoutImageA;
imageMemoryBarrier.srcQueueFamilyIndex = 0;
imageMemoryBarrier.dstQueueFamilyIndex = 1;
imageMemoryBarrier.image = imageA;
imageMemoryBarrier .subresourceRange = subresourceRangeImageA;
...
vkCmdPipelineBarrier(cmdBuf, ...);
...
vkQueueSubmit(queueFamily0, ..., submitInfo, ...); 💼
                                                         Semaphore to sync across queues
```

### **ACQUIRE** THE EXCLUSIVE OWNERSHIP

```
VkImageMemoryBarrier imageMemoryBarrier = { };
imageMemoryBarrier.sType = VK STRUCTURE TYPE IMAGE MEMORY BARRIER;
imageMemoryBarrier.srcAccessMask = 0;
imageMemoryBarrier.dstAccessMask = ...
imageMemoryBarrier.oldLayout = oldLayoutImageA;
imageMemoryBarrier.newLayout = newLayoutImageA;
imageMemoryBarrier.srcQueueFamilyIndex = 0;
imageMemoryBarrier.dstQueueFamilyIndex = 1;
imageMemoryBarrier.image = imageA;
imageMemoryBarrier .subresourceRange = subresourceRangeImageA;
                                                                Associated to queue family 1
                               Associated to a commandPool
...
vkCmdPipelineBarrier(cmdBuf, ...);
...
vkQueueSubmit(queueFamily1,..., submitInfo, ...);
```

File View Help	
----------------	--

	START	OVERVIE	W	EVENTS
Frame summary				
Barriers		<b>I</b> I I	Col	or pass #1
Most expensive events		0.000 ms	2	2.000 ms
Context rolls				
Render/depth targets		Color		
Pipelines				
Device configuration			RT #3	

Name	Format	Width	Height	Size in memory	Draw calls	Compression	Pixel wavefront ratio	Sample count	Out of order draw calls	Duration
Color RT #0	VK_FORMAT_R8G8B8A8_SRGB	1920	1080	8 MB	1917	ON	202%	1	0 / 1917	1.853 ms
Color RT #1	VK_FORMAT_A2R10G10B10_UNORM_PACK32	1920	1080	8 MB	1596	ON	202%	1	0 / 1596	1.468 ms
Color RT #2	VK_FORMAT_R8G8B8A8_UNORM	1920	1080	8 MB	1913	OFF	202%	1	0 / 1913	1.617 ms
Color RT #3	VK_FORMAT_R8G8B8A8_UNORM	1920	1080	8 MB	1914	ON	202%	1	0 / 1914	1.722 ms

Use Radeon GPU Profiler (RGP):



File View Help			
	START	OVERVIEW	EVENTS
Frame summary			
Barriers		11 1	Color pass #1
Most expensive events		0.000 ms	2.000 ms
Context rolls			
Render/depth targets		Color	
Pipelines			
Device configuration		Color RT	
5		Color RT	#2

Name	Format	Width	Height	Size in memory	Draw calls	Compression	Pixel wavefront ratio	Sample count	Out of order draw calls	Duration
Color RT #0	VK_FORMAT_R8G8B8A8_SRGB	1920	1080	8 MB	1917	ON	202%	1	0 / 1917	1.853 ms
Color RT #1	VK_FORMAT_A2R10G10B10_UNORM_PACK32	1920	1080	8 MB	1596	ON	202%	1	0 / 1596	1.468 ms
Color RT #2	VK_FORMAT_R8G8B8A8_UNORM	1920	1080	8 MB	1913	OFF	202%	1	0 / 1913	1.617 ms
Color RT #3	VK_FORMAT_R8G8B8A8_UNORM	1920	1080	8 MB	1914	ON	202%	1	0 / 1914	1.722 ms

Use Radeon GPU Profiler (RGP):

The performance increased about ~5-10%, depending on AMD graphics card

### File View Help **START OVERVIEW EVENTS** Frame summary Barriers Color pass #1 0.000 ms 2.000 ms Most expensive events Context rolls Color Render/depth targets Pipelines Color RT #3 Device configuration Color RT #2

Name	Format	Width	Height	Size in memory	Draw calls	Compression	Pixel wavefront ratio	Sample count	Out of order draw calls	Duration
Color RT #0	VK_FORMAT_R8G8B8A8_SRGB	1920	1080	8 MB	1917	ON	202%	1	0 / 1917	1.853 ms
Color RT #1	VK_FORMAT_A2R10G10B10_UNORM_PACK32	1920	1080	8 MB	1596	ON	202%	1	0 / 1596	1.468 ms
Color RT #2	VK_FORMAT_R8G8B8A8_UNORM	1920	1080	8 MB	1913	OFF	202%	1	0 / 1913	1.617 ms
Color RT #3	VK_FORMAT_R8G8B8A8_UNORM	1920	1080	8 MB	1914	ON	202%	1	0 / 1914	1.722 ms

Use Radeon GPU Profiler (RGP):

The performance increased about ~5-10%, depending on AMD graphics card

### File View Help **START OVERVIEW EVENTS** Frame summary Barriers Color pass #1 0.000 ms 2.000 ms Most expensive events Context rolls Color Render/depth targets Pipelines Color RT #3 Device configuration Color RT #2

Name Color RT #0	Format VK_FORMAT_R8G8B8A8_SRGB	<b>Width</b> 1920	Height 1080	Size in memory 8 MB	Draw calls 1917	Compres ON	What about this one?	Out of order draw calls 0 / 1917	Duration 1.853 ms
Color RT #1	VK_FORMAT_A2R10G10B10_UNORM_PACK32	1920	1080	8 MB	1596	ON	202% 1	0 / 1596	1.468 ms
Color RT #2	VK_FORMAT_R8G8B8A8_UNORM	1920	1080	8 MB	1913	OFF	202% 1	0 / 1913	1.617 ms
Color RT #3	VK_FORMAT_R8G8B8A8_UNORM	1920	1080	8 MB	1914	ON	202% 1	0 / 1914	1.722 ms

### AND ONCE AGAIN ... ③

### Color RT #2 – G-buffer resource #2

✓ CreateInfo	VkImageCreateInfo()
sType	VK_STRUCTURE_TYPE_IMAGE_CREATE_INFO
<b>&gt;</b> pNext	VkImageFormatListCreateInfoKHR()
flags	VK_IMAGE_CREATE_MUTABLE_FORMAT_BIT
imageType	VK_IMAGE_TYPE_2D
format	VK_FORMAT_R8G8B8A8_UNORM
> extent	VkExtent3D()
mipLevels	1
arrayLayers	1
samples	VK_SAMPLE_COUNT_1_BIT
tiling	VK_IMAGE_TILING_OPTIMAL
usage	VK_IMAGE_USAGE_TRANSFER_SRC_BIT   VK_IMAGE_USAGE_TRANSFER_DST_BIT   VK_IMAGE_USAGE_SAMPLED_BIT   VK_IMAGE_USAGE_STORAGE_BIT   VK_IMAGE_USAGE_COLOR_ATTACHMENT_BI
sharingMode	VK_SHARING_MODE_EXCLUSIVE
queueFamilyIndexCount	0
pQueueFamilyIndices	uint32_t[]
initialLayout	VK_IMAGE_LAYOUT_UNDEFINED

#### AND ONCE AGAIN ... ③

#### Color RT #2 – G-buffer resource #2

✓ CreateInfo	VkImageCreateInfo()
sType	VK_STRUCTURE_TYPE_IMAGE_CREATE_INFO
<b>&gt;</b> pNext	VkImageFormatListCreateInfoKHR()
flags	VK_IMAGE_CREATE_MUTABLE_FORMAT_BIT
imageType	VK_IMAGE_TYPE_2D
format	
> extent	VkExtent3D()
mipLevels	1
arrayLayers	1
samples	VK_SAMPLE_COUNT_1_BIT
tiling	VK_IMAGE_TILING_OPTIMAL
usage	VK_IMAGE_USAGE_TRANSFER_SRC_BIT   VK_IMAGE_USAGE_TRANSFER_DST_BIT   VK_IMAGE_USAGE_SAMPLED_BIT   VK_IMAGE_USAGE_STORAGE_BIT   VK_IMAGE_USAGE_COLOR_ATTACHMENT_B
sharingMode	
queueFamilyIndexCount	0
pQueueFamilyIndices	uint32_t[]
initialLayout	VK_IMAGE_LAYOUT_UNDEFINED

### **USAGE FLAGS**

#### Color RT #2 – G-buffer resource #2

		due to async compute
✓ CreateInfo	VkImageCreateInfo()	
sType	VK_STRUCTURE_TYPE_IMAGE_CREATE_INFO	
<b>&gt;</b> pNext	VkImageFormatListCreateInfoKHR()	-> VK_IMAGE_USAGE_STORAGE_BIT
flags	VK_IMAGE_CREATE_MUTABLE_FORMAT_BIT	is now required for G-buffer resource #2
imageType	VK_IMAGE_TYPE_2D	is now required for O-builer resource #2
format	format VK_FORMAT_R8G8B8A8_UNORM	
> extent	VkExtent3D()	
mipLevels	1	
arrayLayers	1	
samples	VK_SAMPLE_COUNT_1_BIT	
tiling	VK_IMAGE_TILING_OPTIMAL	
usage	VK_IMAGE_USAGE_TRANSFER_SRC_BIT   VK_IMAGE_USAGE_TRANSFER_DS	_BIT   VK_IMAGE_USAGE_SAMPLED_BIT VK_IMAGE_USAGE_STORAGE_BIT VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT
sharingMode	VK_SHARING_MODE_EXCLUSIVE	
queueFamilyIndexCount	0	
pQueueFamilyIndices	uint32_t[]	
initialLayout	VK_IMAGE_LAYOUT_UNDEFINED	

Post process moved to the compute queue

### **USAGE FLAGS**

#### Color RT #2 – G-buffer resource #2

II		due to async compute	
✓ CreateInfo	VkImageCreateInfo()		
sType	VK_STRUCTURE_TYPE_IMAGE_CREATE_INFO		
<b>&gt;</b> pNext	VkImageFormatListCreateInfoKHR()	-> VK_IMAGE_USAGE_STORAGE_BIT	
flags	VK_IMAGE_CREATE_MUTABLE_FORMAT_BIT	is now required for G-buffer resource #2	
imageType	VK_IMAGE_TYPE_2D	is now required for O-bullet resource #2	
format	format VK_FORMAT_R8G8B8A8_UNORM		
> extent	VkExtent3D()	WHY?	
mipLevels	1		
arrayLayers	1		
samples	VK_SAMPLE_COUNT_1_BIT		
tiling	VK_IMAGE_TILING_OPTIMAL		
usage	VK_IMAGE_USAGE_TRANSFER_SRC_BIT   VK_IMAGE_USAGE_TRANSFER_V	ST_BIT   VK_IMAGE_USAGE_SAMPLED_BIT VK_IMAGE_USAGE_STORAGE_BIT VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT	
sharingMode	VK_SHARING_MODE_EXCLUSIVE		
queueFamilyIndexCount	0		
pQueueFamilyIndices	uint32_t[]		
initialLayout	VK_IMAGE_LAYOUT_UNDEFINED		

Post process moved to the compute queue

# VK\_IMAGE\_USAGE\_STORAGE\_BIT

#### Spec:

"VK\_IMAGE\_USAGE\_STORAGE\_BIT specifies that the image can be used to create a VkImageView suitable for occupying a VkDescriptorSet slot of type VK\_DESCRIPTOR\_TYPE\_STORAGE\_IMAGE "

#### Spec:

" A storage image (VK\_DESCRIPTOR\_TYPE\_STORAGE\_IMAGE) is a descriptor type associated with an image resource via an image view that load, **store**, and atomic operations can be performed on."

# VK\_IMAGE\_USAGE\_STORAGE\_BIT

#### Spec:

"VK\_IMAGE\_USAGE\_STORAGE\_BIT specifies that the image can be used to create a VkImageView suitable for occupying a VkDescriptorSet slot of type VK\_DESCRIPTOR\_TYPE\_STORAGE\_IMAGE "

#### Spec:

" A storage image (VK\_DESCRIPTOR\_TYPE\_STORAGE\_IMAGE) is a descriptor type associated with an image resource via an image view that load, **store**, and atomic operations can be performed on."



### **USAGE FLAGS**

Usage flags influencing DCC:

- VK\_IMAGE\_USAGE\_STORAGE\_BIT disables DCC
- VK\_IMAGE\_USAGE\_SAMPLED\_BIT makes DCC less efficient

### **USAGE FLAGS**

Usage flags influencing DCC:

- VK\_IMAGE\_USAGE\_STORAGE\_BIT disables DCC
- VK\_IMAGE\_USAGE\_SAMPLED\_BIT makes DCC less efficient

Always use what you need, but not more



#### SUMMARY

VK\_IMAGE\_CREATE\_MUTABLE\_FORMAT\_BIT

use VK\_KHR\_image\_format\_list

VK\_SHARING\_MODE\_EXCLUSIVE

- don't use sharing mode concurrent in production ready code
- use SHARING\_MODE\_EXCLUSIVE and transfer queue family ownership when required

USAGE FLAGS

• set all the usage flags you need, but not more

### **OTHER NIT-PICKS CONCERNING DCC**

#### Decompression

- During transfer operations
- General layout

Depth targets

- Compressed differently
- Above guidelines don't apply here

There is no rule without expection

There might be some tweaks in the driver for specific cards

### **OTHER NIT-PICKS CONCERNING DCC**

#### Decompression

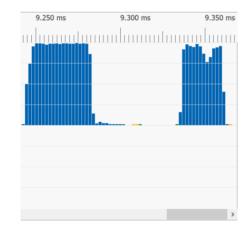
- During transfer operations
- General layout

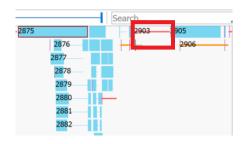
#### Depth targets

- Compressed differently
- Above guidelines don't apply here

There is no rule without expection

There might be some tweaks in the driver for specific cards







# SYNCHRONIZATION

Barriers

- Placing
- Batching
- Pipeline stage masks

Cross queue synchronization

#### BARRIERS

- Experience with barriers in this particular game
- Most of the issues likely have their roots in the original engine structure, which is DX11-based

#### BARRIERS

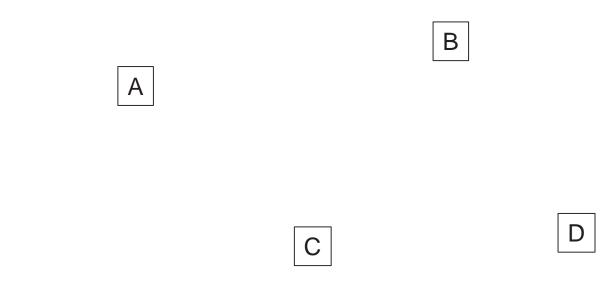
- Experience with barriers in this particular game
- Most of the issues likely have their roots in the original engine structure, which is DX11-based
- -> Rearranging barriers to get more overlap between the drawcalls / passes

-> Batching barriers to save some additional time

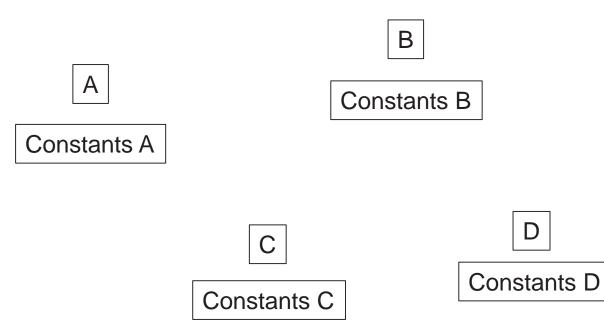
#### BARRIERS

- Experience with barriers in this particular game
- Most of the issues likely have their roots in the original engine structure, which is DX11 based
- -> Rearranging barriers to get more overlap between the drawcalls / passes
- -> Batching barriers to save some additional time
- Other findings
- -> Where specifying barriers as precise as possible really pays of

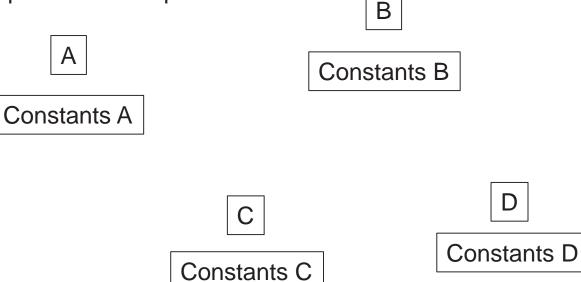
 The rendering work is logically organized in components – e.g. one shadow map component, one lighting component etc.



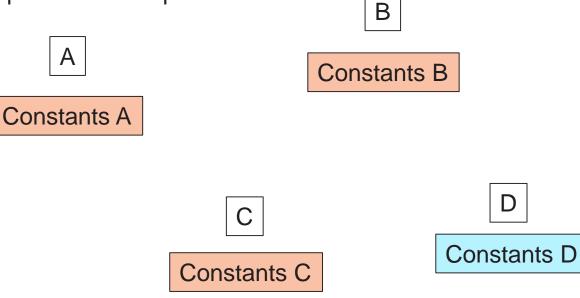
 The rendering work is logically organized in components – e.g. one shadow map component, one lighting component etc.



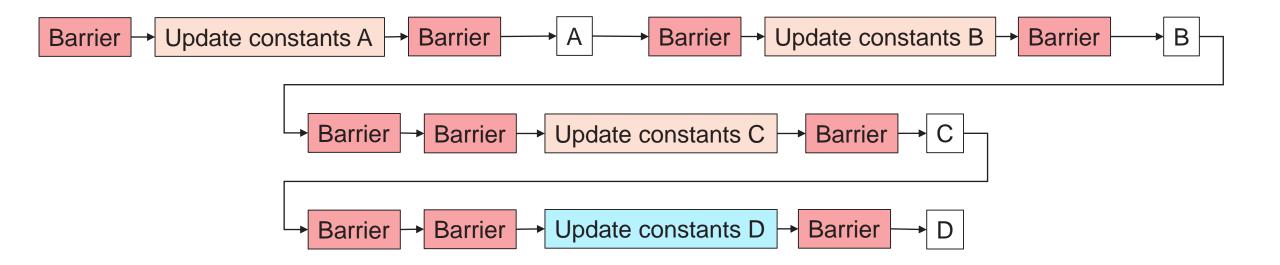
- Constants information is gathered on the CPU side in the beginning of each frame
- Constants A, B and C are equal, constants D are different
- Component A is independent from Component B
- Component C depends on Component A and B
- Component D depends on Component C



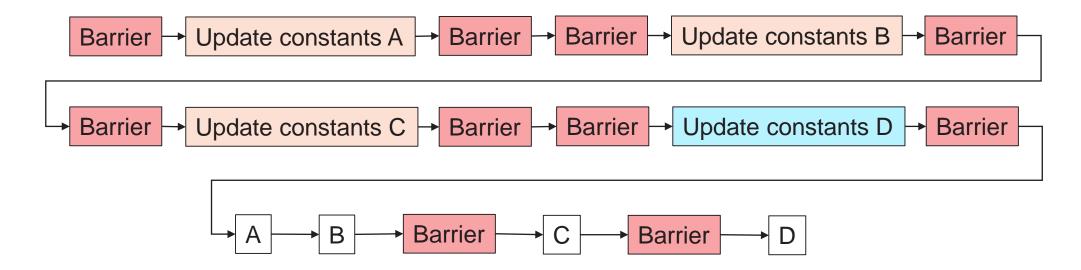
- Constants information is gathered on the CPU side in the beginning of each frame
- Constants A, B and C are equal, constants D are different
- Component A is independent from Component B
- Component C depends on Component A and B
- Component D depends on Component C



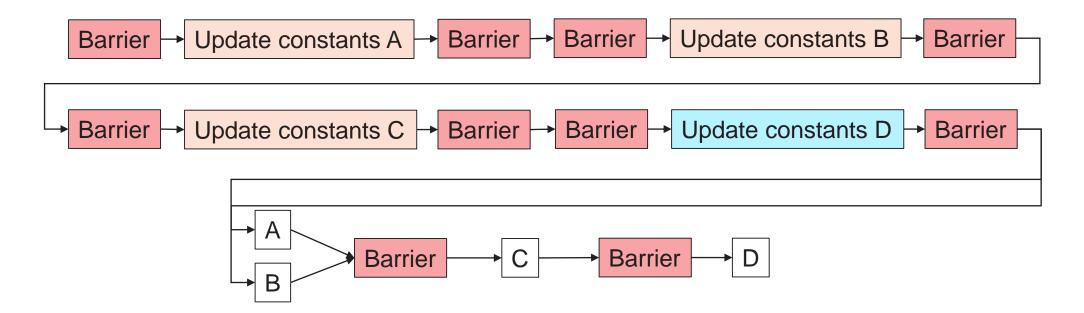
- Constants information is gathered on the CPU side in the beginning of each frame
- Constants A, B and C are equal, constants D are different
- Component A is independent from Component B
- Component C depends on Component A and B
- Component D depends on Component C



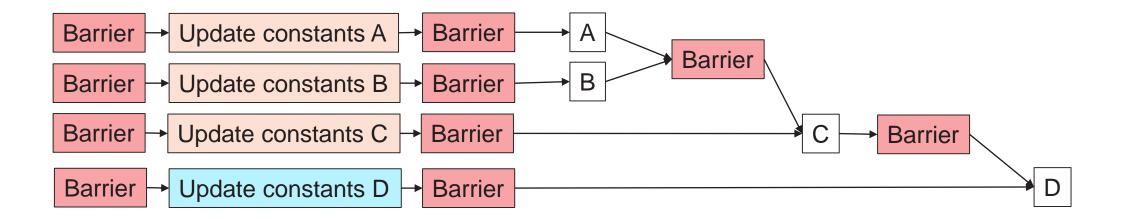
- Constants information is gathered on the CPU side in the beginning of each frame
- Constants A, B and C are equal, constants D are different
- Component A is independent from Component B
- Component C depends on Component A and B
- Component D depends on Component C



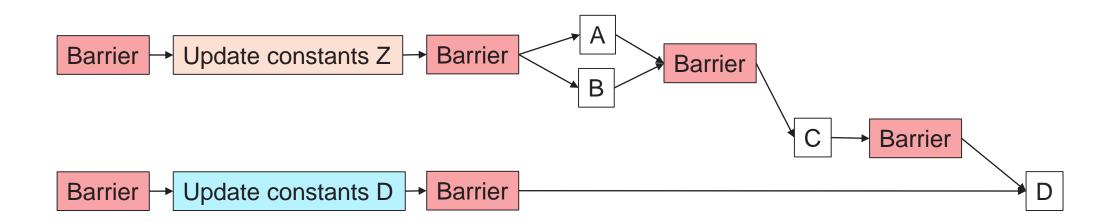
- Constants information is gathered on the CPU side in the beginning of each frame
- Constants A, B and C are equal, constants D are different
- Component A is independent from Component B
- Component C depends on Component A and B
- Component D depends on Component C



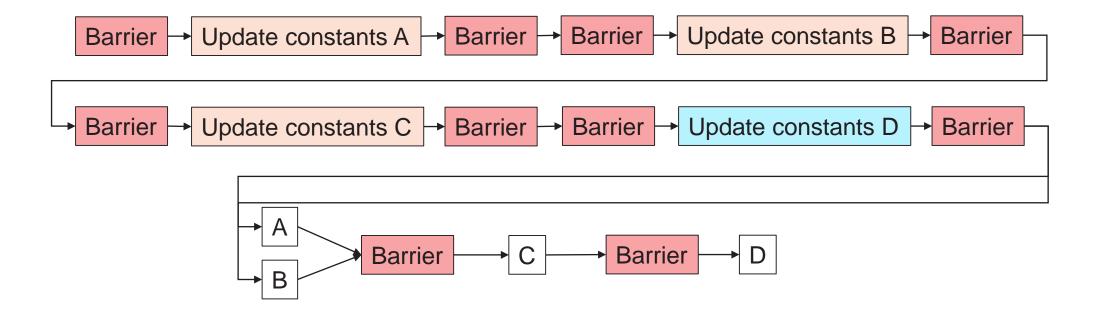
- Constants information is gathered on the CPU side in the beginning of each frame
- Constants A, B and C are equal, constants D are different
- Component A is independent from Component B
- Component C depends on Component A and B
- Component D depends on Component C

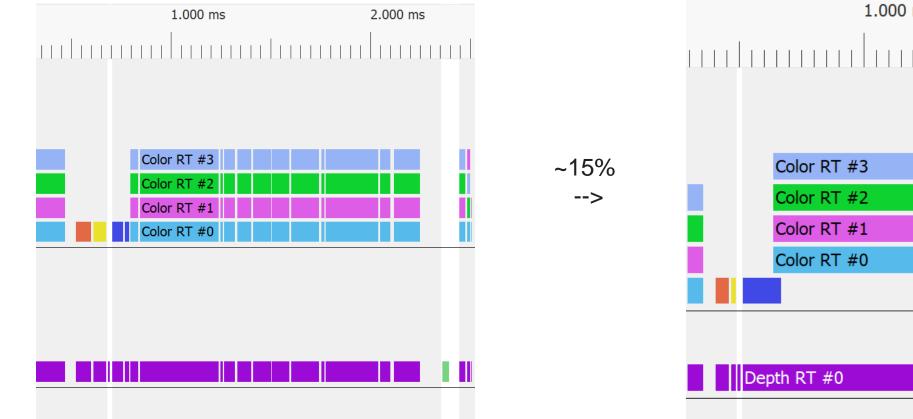


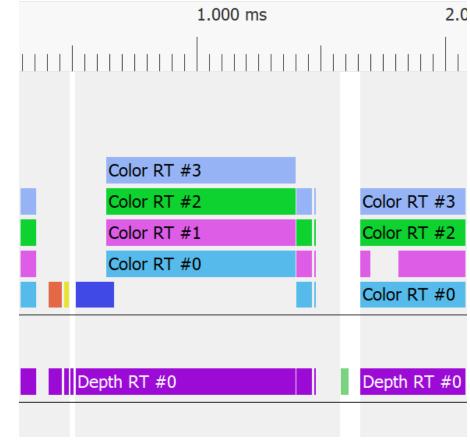
- Constants information is gathered on the CPU side in the beginning of each frame
- Constants A, B and C are equal, constants D are different
- Component A is independent from Component B
- Component C depends on Component A and B
- Component D depends on Component C

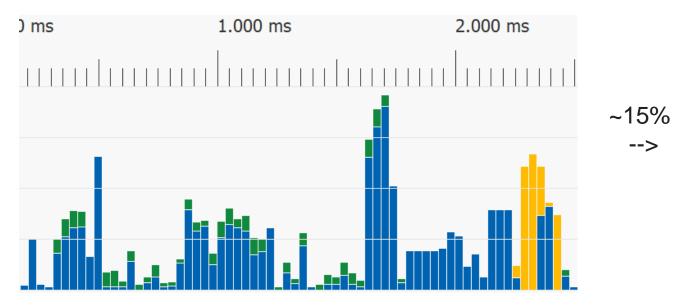


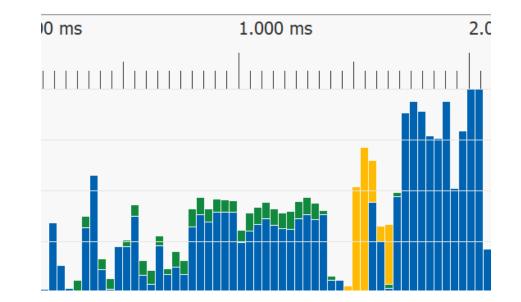
• This is what we ended up with – but it already had observable changes











#### Early builds had several consecutive barriers:

167 vkCmdDispatch(25	0.001 ms
168 vkCmdPipelineBarri	0.002 ms
169 vkCmdPipelineBarrier()	0.001 ms
170 vkCmdPipelineBarrier()	0.001 ms
171 vkCmdPipelineBarrier()	0.002 ms
172 vkCmdPipelineBarrier()	- 0.001 ms
173 vkCmdPipelineBarrier()	0.002 ms
174 vkCmdPipelineBarrier()	0.001 ms
175 vkCmdPipelineBarrier()	0.001 ms
176 vkCmdPipelineBarrier()	0.001 ms
177 vkCmdPipelineBarrier()	0.001 ms

#### Early builds had several consecutive barriers:

	167 vkCmdDispatch(25 168 vkCmdPipelineBarri
169	vkCmdPipelineBarrier()
170	vkCmdPipelineBarrier()
171	vkCmdPipelineBarrier()
172	vkCmdPipelineBarrier()
173	vkCmdPipelineBarrier()
174	vkCmdPipelineBarrier()
175	vkCmdPipelineBarrier()
176	vkCmdPipelineBarrier()
177	vkCmdPipelineBarrier()
.77	vkCmdPipelineBarrier()

0.001 ms
0.002 ms
0.001 ms
0.001 ms
0.002 ms
- 0.001 ms
0.002 ms
0.001 ms
0.001 ms
0.001 ms
0.001 ms
_

void vkCmdPipelineBarrier(	
VkCommandBuffer	CO
VkPipelineStageFlags	sr
VkPipelineStageFlags	ds
VkDependencyFlags	de
uint32_t	me
const VkMemoryBarrier*	рM
uint32_t	bu
const VkBufferMemoryBarrier*	pВ
uint32_t	im
const VkImageMemoryBarrier*	pI

commandBuffer, srcStageMask, dstStageMask, dependencyFlags, memoryBarrierCount, pMemoryBarriers, bufferMemoryBarrierCount, pBufferMemoryBarriers, imageMemoryBarrierS);

#### Early builds had several consecutive barriers:

0.001 ms

0.001 ms

0.002 ms

0.002 ms

0.001 ms

0.001 ms

167 vkCmdDispatch(25
168 vkCmdPipelineBarri
169 vkCmdPipelineBarrier()
170 vkCmdPipelineBarrier()
171 vkCmdPipelineBarrier()
172 vkCmdPipelineBarrier()
173 vkCmdPipelineBarrier()
174 vkCmdPipelineBarrier()
175 vkCmdPipelineBarrier()
176 vkCmdPipelineBarrier()
177 vkCmdPipelineBarrier()

#### Example: 2 image layout transitions

void vkCmdPipelineBarrier(	
VkCommandBuffer	COMM
VkPipelineStageFlags	srcS
VkPipelineStageFlags	dstS
VkDependencyFlags	depe
uint32_t	memo
const VkMemoryBarrier*	pMem
uint32_t	buff
<pre>const VkBufferMemoryBarrier*</pre>	pBuf
uint32_t	image
const VkImageMemoryBarrier*	pIma

commandBuffer, srcStageMask, dstStageMask, dependencyFlags, memoryBarrierCount, pMemoryBarriers, bufferMemoryBarrierCount, pBufferMemoryBarriers, imageMemoryBarrierS);

vkCmdPipelineBarrier(..., 0, NULL, 0, NULL, 0, &imageBarrierA); vkCmdPipelineBarrier(..., 0, NULL, 0, NULL, 0, &imageBarrierB);

#### Early builds had several consecutive barriers:

0.001 ms

0.002 ms

0.001 ms

0.002 ms

0.002 ms

0.001 ms

0.001 ms

167 vkCmdDispatch(25
168 vkCmdPipelineBarri
169 vkCmdPipelineBarrier()
170 vkCmdPipelineBarrier()
171 vkCmdPipelineBarrier()
172 vkCmdPipelineBarrier()
173 vkCmdPipelineBarrier()
174 vkCmdPipelineBarrier()
175 vkCmdPipelineBarrier()
176 vkCmdPipelineBarrier()
177 vkCmdPipelineBarrier()

#### Example: 2 image layout transitions

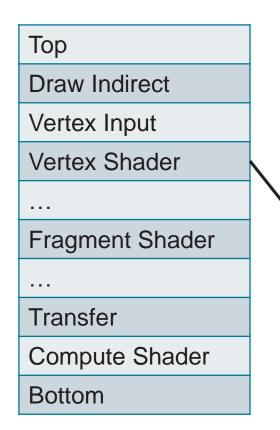
void vkCmdPipelineBarrier(	
VkCommandBuffer	commandBuffer,
VkPipelineStageFlags	srcStageMask,
VkPipelineStageFlags	dstStageMask,
VkDependencyFlags	dependencyFlags,
uint32_t	<pre>memoryBarrierCount,</pre>
const VkMemoryBarrier*	pMemoryBarriers,
uint32_t	<pre>bufferMemoryBarrierCount,</pre>
<pre>const VkBufferMemoryBarrier*</pre>	pBufferMemoryBarriers,
uint32_t	<pre>imageMemoryBarrierCount,</pre>
<pre>const VkImageMemoryBarrier*</pre>	<pre>pImageMemoryBarriers);</pre>

```
vkCmdPipelineBarrier(..., 0, NULL, 0, NULL, 0, &imageBarrierA);
vkCmdPipelineBarrier(..., 0, NULL, 0, NULL, 0, &imageBarrierB);
```

VkImageMemoryBarrier[2] imageBarriers = {imageBarrierA, imageBarrierB}; vkCmdPipelineBarrier(..., 0, NULL, 0, NULL, 2, &imageBarriers);

->

#### **PIPELINE STAGE MASKS**



Тор
Draw Indirect
Vertex Input
Vertex Shader
Fragment Shader
Transfer
Compute Shader
Bottom

### **PIPELINE STAGE MASKS**

Ton



**Draw Indirect** 

Vertex Input

Vertex Shader

. . .

 $\sim \sim \sim$ 

Fragment Shader

Transfer

Compute Shader

Bottom

юр
Draw Indirect
Vertex Input
Vertex Shader
Fragment Shader
Transfer
Compute Shader
Bottom

#### ALL\_COMMANDS\_BIT

#### Spec:

"VK\_PIPELINE\_STAGE\_ALL\_COMMANDS\_BIT is equivalent to the logical OR of every other pipeline stage flag that is supported on the queue it is used with."

Тор		
Draw Indirect		
Vertex Input		
Vertex Shader		
Fragment Shader		
Transfer		
Compute Shader		
Bottom		

#### ALL\_COMMANDS\_BIT

Spec:

"VK\_PIPELINE\_STAGE\_ALL\_COMMANDS\_BIT is equivalent to the logical OR of every other pipeline stage flag that is supported on the queue it is used with."

<b>T</b> <sub>2</sub>
Тор
Draw Indirect
Vertex Input
Vertex Shader
Fragment Shader
Transfer
Compute Shader
Bottom

ALL\_COMMANDS\_BIT

Spec:

"VK\_PIPELINE\_STAGE\_ALL\_COMMANDS\_BIT is equivalent to the logical OR of every other pipeline stage flag that is supported on the queue it is used with."

The bottom bit adds a wait on end of pipe + timestamp -> can take up to  $\sim$ 64k cycles on the async queue  $\otimes$ 

Тор	
Draw Indirect	
Vertex Input	
Vertex Shader	
Fragment Shader	
Transfer	
Compute Shader	
Bottom	T

->

#### ALL\_COMMANDS\_BIT

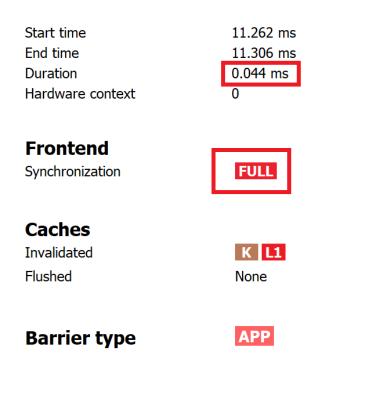
#### Spec:

"VK\_PIPELINE\_STAGE\_ALL\_COMMANDS\_BIT is equivalent to the logical OR of every other pipeline stage flag that is supported on the queue it is used with."

Use the specific pipeline stage mask instead of all\_commands, e.g.: VK\_PIPELINE\_STAGE\_COMPUTE\_SHADER\_BIT | VK\_PIPELINE\_STAGE\_TRANSFER\_BIT

The bottom bit adds a wait on end of pipe + timestamp -> can take up to ~64k cycles on the async queue 🙁

->



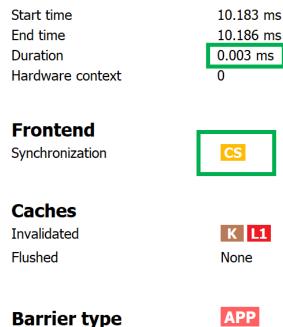
#### Layout transitions

None

VK\_PIPELINE\_STAGE\_ALL\_COMMANDS\_BIT on async compute queue

### **ALL\_COMMANDS\_BIT – COMPUTE PIPELINE**

->



#### Layout transitions None

VK\_PIPELINE\_STAGE\_COMPUTE\_SHADER\_BIT | VK\_PIPELINE\_STAGE\_TRANSFER\_BIT on async compute queue

### **CROSS QUEUE SYNCHRONIZATION**

#### The engine used to have ~7 command buffers per frame

Π						
	vkQueuePresentKHR					
	[531201] VkCommandBuffer	vkQueuePresentKHR				
- 1	[25658] VkCommandBuffer	[531201] VkCommandBuffer	vkQueuePresentKHR			
	[25656] VkCommandBuffer	[25658] VkCommandBuffer	[531201] VkCommandBuffer	vkQueu	aPresentKHR	£.,
	[530689] VkCommandBuffer	[25656] VkCommandBuffer	[25658] VkCommandBuffer	[531201	] VkCommandBuffer	
5	[530177] VkCommandBuffer	[530689] VkCommandBuffer	[25656] VkCommandBuffer	[25658]	VkCommandBuffer	V
2	[529665] VkCommandBuffer	[530177] VkCommandBuffer	[530689] VkCommandBuffer	[25656]	VkCommandBuffer	C
	[529153] VkCommandBuffer	[529665] VkCommandBuffer	[530177] VkCommandBuffer	[530689	] VkCommandBuffer	I

### **CROSS QUEUE SYNCHRONIZATION**

#### The engine used to have ~7 command buffers per frame

vkQueuePresentKHR			
[531201] VkCommandBuffer	vkQueuePresentKHR		
[25658] VkCommandBuffer	[531201] VkCommandBuffer	vkQueuePresentKHR	
[25656] VkCommandBuffer	[25658] VkCommandBuffer	[531201] VkCommandBuffer	vkQueuaPresentKHR
[530689] VkCommandBuffer	[25656] VkCommandBuffer	[25658] VkCommandBuffer	[531201] VkCommandBuffer
[530177] VkCommandBuffer	[530689] VkCommandBuffer	[25656] VkCommandBuffer	[25658] VkCommandBuffer
[529665] VkCommandBuffer	[530177] VkCommandBuffer	[530689] VkCommandBuffer	[25656] VkCommandBuffer
[529153] VkCommandBuffer	[529665] VkCommandBuffer	[530177] VkCommandBuffer	[530689] VkCommandBuffer

After async compute support was added, the number of command buffers doubled

			[705537] VkCommandBuffer		
		[708097] VkCommandBuffer	[7 [707329] VkCommandBuffer	[70!	
		[24602] VkCommandBuffer	[70 [705025] VkCommandBuffer	[70;	
		[24600] VkCommandBuffer	vkQue [7( [706817] VkCommandBuffer	[70!	5
		[708609] VkCommandBuffer	[70805 vkt [704513] VkCommandBuffer	· [70ŧ	E .
		[708353] VkCommandBuffer	[24602 [70 vkQueuePresentKHR	[704	r.
3'	[704001] VkCommandBu	[706049] VkCommandBuffer	[2460C [24 [708097] VkCommandBuffer	vkQ	[70
1:	[703233] VkCommandBL	[704001] VkCommandBuffer	[7086C [24 [24602] VkCommandBuffer	[708	[70
8 [699393] VkCommandBuffer	[708865] VkCommandBuff	[703233] VkCommandBuffer	[70835 [7( [24600] VkCommandBuffer	[24	[70
6 [701185] VkCommandBuffer	[699393] VkCommandBuffer	[708865] VkCommandBuffer	[70604 [7( [708609] VkCommandBuffer	[24	[70
3 [698881] VkCommandBuffer	[701185] VkCommandBuffer	[699393] VkCommandBuffer	[7040C [7C [708353] VkCommandBuffer	. [70]	[70
[700673] VkCommandBuffer	[698881] VkCommandBuffer	[701185] VkCommandBuffer	[70323 [7 [706049] VkCommandBuffer	[70	
		Protocol 14 Concerning of Concerning	[705793] VkCommandBuffe		
		[706305] VkCommandBuffer	[704769] VkCommandBuffer	U IV	
er		[704257] VkCommandBuffer	[706305] VkCommandBuffer	[7 [705793] VkComn	
er [699649] VkCommandBuffer			[704257] VkCommandBuffer	[7 [704769] VkComn	mandB
[698625] VI			[699649] VkCommandBuffer	[706305]	

### **CROSS QUEUE SYNCHRONIZATION**

Cross queue synchronization is only possible at submission boundaries

					[705537] VkCommandBuffer	
					[0x14f57f729e0] VkSemaphoreWait	[705537] Vk
					[0x14ef7796c20] VkSemaphoreSignal	[0x14f57f729 [70
			[0x14f57f72700] VkSemaphoreSignal		[707329] VkCommandBuffer	[0x14ef7796 [0x
			[708097] VkCommandBuffer		[0x14ef7796660] VkSemaphoreWait	[707329] Vk <mark>[0</mark> x
			[24602] VkCommandBuffer	[7	[705025] VkCommandBuffer	[0x14ef7796 [70
			[24600] VkCommandBuffer	[0	[0x14ec22c8e10] VkSemaphoreWait	[705025] Vk <mark>[0</mark> x
			[708609] VkCommandBuffer	vkQue [7	[706817] VkCommandBuffer	[0x14ec22c8 [70
			[0x14ef7797bf0] VkSemaphoreWait		[0x14ef7795f30] VkSemaphoreSignal	[706817] Vk <mark>[0x</mark>
			[708353] VkCommandBuffer	[0x14f [0	[704513] VkCommandBuffer	[0x14ef7795 [70
			[0x14ef7796940] VkSemaphoreWait	[70809 [0	vkQueuePresentKHR	[704513] Vk <mark>[0</mark> x
		[704001] VkCommandB	[706049] VkCommandBuffer	[24602 [7	[0x14f57f72700] VkSemaphoreWait	vkQueuePre [70
		[703233] VkCommandBi	[0x14ef7796ab0] VkSemaphoreSignal	[24600 [2	[0x14f57f72700] VkSemaphoreSignal	[0x14f57f72] vkC
699393] VkCommandBuffer		[0x14ec22c9820] VkSem	[704001] VkCommandBuffer	[7086C [2	[708097] VkCommandBuffer	[0x14f57f72]
0x14f57f72590] VkSemaphoreWait	[	708865] VkCommandBuf	[703233] VkCommandBuffer	[0x14e [7	[24602] VkCommandBuffer	[708097] Vk <mark>[0</mark> x
0x14ef7796c20] VkSemaphoreSignal	[699393] VkCo	rnmandBuffer	[0x14ec22c9820] VkSemaphoreSignal	[70835 [0	[24600] VkCommandBuffer	[24602] VkC [70
701185] VkCommandBuffer	[0x14f57f72590	] VkSemaphoreWait	[708865] VkCommandBuffer	[0x14e [7	[708609] VkCommandBuffer	[24600] VkC [24
0x14ef7796660] VkSemaphoreWait	[0x14ef7796c2	[] VkSemaphoreSignal	[699393] VkCommandBuffer	[70604 [0	[0x14ef7797bf0] VkSemaphoreWait	[708609] Vk [24
698881] VkCommandBuffer	[701185] VkCo	rnmandBuffer	[0x14f57f72590] VkSemaphoreWait	[0x14e [7	[708353] VkCommandBuffer	[0x14ef7797 [70
0x14ec22c8e10] VkSemaphoreWait	[0x14ef779666	0] VkSemaphoreWait	[0x14ef7796c20] VkSemaphoreSignal	[7040C <b>[0</b>	[0x14ef7796940] VkSemaphoreWait	[708353] Vk <mark>[0</mark> x
[700673] VkCommandBuffer	[698881] VkCo	rnmandBuffer	[701185] VkCommandBuffer	[70323 [7	[706049] VkCommandBuffer	[0x14ef7796 [70
					[0x14ef7797bf0] VkSemaphoreSignal [705793] VkCommandBuffer [0x14ef7796c0] VkSemaphoreSignal [0x14ef7796660] VkSemaphoreSignal [704769] VkCommandBuffer	10
			[0x14ef7796940] VkSemaphoreSignal	_	[0x14ef7795f30] VkSemaphoreWait	[7 [0x14ef7]
			[706305] VkCommandBuffer	[0x14ef7	796940] VkSemaphoreSignal	[0 [705793]
			[0x14ef7796ab0] VkSemaphoreWait	[706305]	VkCommandBuffer	[0 [0x14ef77 [0x14ef
The second second			[704257] VkCommandBuffer	[0x14ef7	796ab0] VkSemaphoreWait	[7 [0x14ef7; [705793
[0x14et779]			[0x14ec22c9820] VkSemaphoreWait	[704257]	VkCommandBuffer	[0 [704769] [0x14ef
[699649] VI				[0x14ec2	2c9820] VkSemaphoreWait	[0 [0x14ef77 [0x14ef
[0x14ef779; [699649] VI [0x14ef779, [0x14ef7797bf0] VkSemaphoreSignal [0x14ef779; [699649] VkCommandBuffer					2c9820] VkSemaphoreWait 797bf0] VkSemaphoreSignal	[0 [0x14ef7; [0x14ef [7 [0x14ef7; [704769

#### SUMMARY

- Check your barriers if you can rearrange them
- Batch consecutive barriers to a single barrier
- Specify your barriers as precise as possible
- Cross queue synchronization is only possible at submission boundaries

# **OTHER SMALL THINGS**

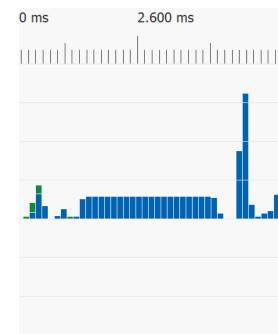
- Copy queue
- Compute queue & the swapchain
- Shader building infrastructure

### **COPY QUEUE**

Resource was copied from GPU to CPU

- Generated on GPU during previous frame
- After the copy overwritten with updated data from current frame

This copy blocked the whole GPU. -> ~1-2% of frame time vkCmdCopyImage() 0.199 ms



### **COPY QUEUE**

By using the copy queue, we won the time previously spend for vkCmdCopyImage() back.

		[24602] VkCommandBuffer [24600] VkCommandBuffer	[7( [705025] VkComm vkQue [7( [706817] VkComm		[70; [70;
		[708609] VkCommandBuffer	[70805 vkt [704513] VkComm		[706
		[708353] VkCommandBuffer	[24602 [7( vkQueuePresentKH		[704
	[704001] VkCommandB	[706049] VkCommandBuffer	[2460C [24 [708097] VkComm	andBuffer	vkQ
	[703233] VkCommandB	[704001] VkCommandBuffer	[70860 [24 [24602] VkComma	ndBuffer	[708
699393] VkCommandBuffer	[708865] VkCommandBuf	f [703233] VkCommandBuffer	[70835 [70 [24600] VkComma	ndBuffer	[246
701185] VkCommandBuffer	[699393] VkCommandBuffer	[708865] VkCommandBuffer	[70604 [70 [708609] VkComm	andBuffer	[246
598881] VkCommandBuffer	[701185] VkCommandBuffer	[699393] VkCommandBuffer	[70400 [70 [708353] VkComm	andBuffer	[708
700673] VkCommandBuffer	[698881] VkCommandBuffer	[701185] VkCommandBuffer	[70323 [7 [706049] VkComm	andBuffer	[70
		[706305] VkCommandBuffer	[705793] VkComn [704769] VkCommar		
		[706305] VkCommandBuffer [704257] VkCommandBuffer		dBuffer [7	15793] VkComma
[699649] VkCommandBuffer			[704769] VkCommar	dBuffer [7 [7]	15793] VkComma 14769] VkComma

#### **COMPUTE QUEUE & SWAPCHAIN**

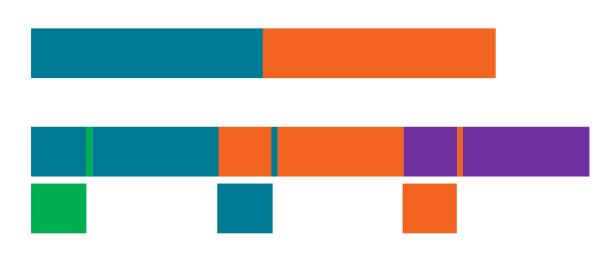
Write directly from compute to the swapchain

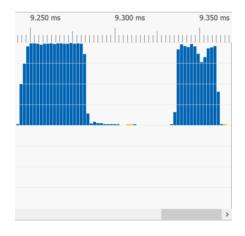


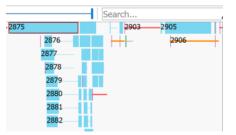
#### **COMPUTE QUEUE & SWAPCHAIN**

Write directly from compute to the swapchain 2875 vkCmdDraw(3, 1, 0, 0)

0.041 ms







### **COMPUTE QUEUE & SWAPCHAIN**

Write directly from compute to the swapchain

Possibly present from compute



#### Vulkan specific feature

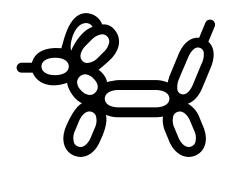


#### SHADER BUILDING INFRASTRUCTURE



#### SUMMARY

- Check for compression, especially for the G-buffer render targets
- Take special care of the barriers ③
- Can you make good use of the copy queue?
- The compute queue can write directly to the swapchain
- Use the DXC compiler



## **THANKS TO**

**Dominik Baumeister** 

Matthäus Chajdas

**Tobias Hector** 

Adam Sawicki

Rys Sommefeldt

Steven Tovey

Marco Weber

# REFERENCES

https://www.khronos.org/registry/vulkan/specs/1.1-extensions/html/

https://gpuopen.com/dcc-overview/

https://gpuopen.com/vulkan-barriers-explained/

https://github.com/GPUOpen-LibrariesAndSDKs/VulkanMemoryAllocator

https://gpuopen.com/reducing-vulkan-api-call-overhead/

### Q&A

lou.kramer@amd.com

@lou\_auroyup

https://gpuopen.com/



#### **DISCLAIMER & ATTRIBUTION**

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

#### ATTRIBUTION

© 2019 Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo and combinations thereof are trademarks of Advanced Micro Devices, Inc. in the United States and/or other jurisdictions. Other names are for informational purposes only and may be trademarks of their respective owners.

# AMD