



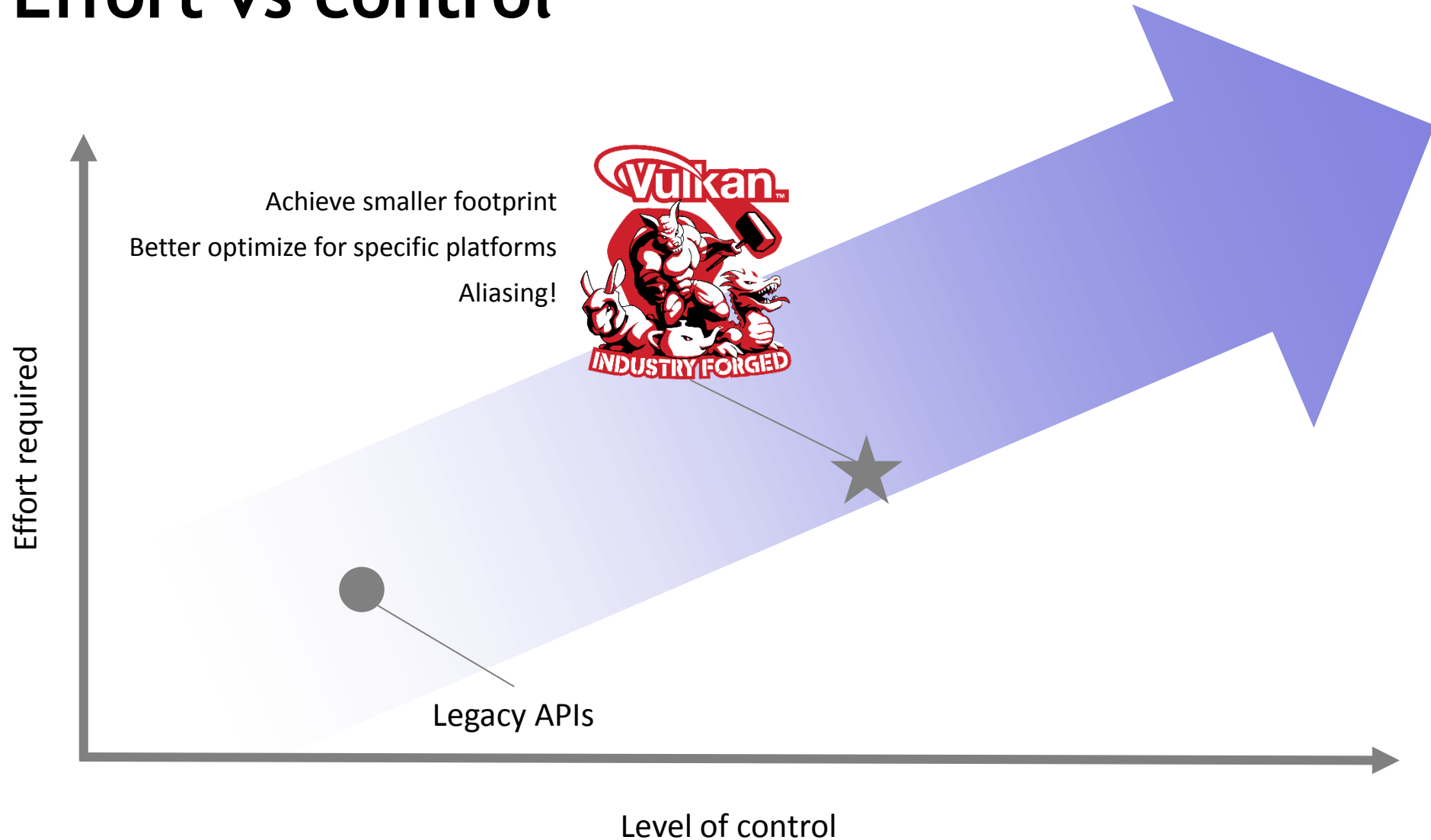
Vulkan Memory Types

Ashley Smith
November 2018

Agenda

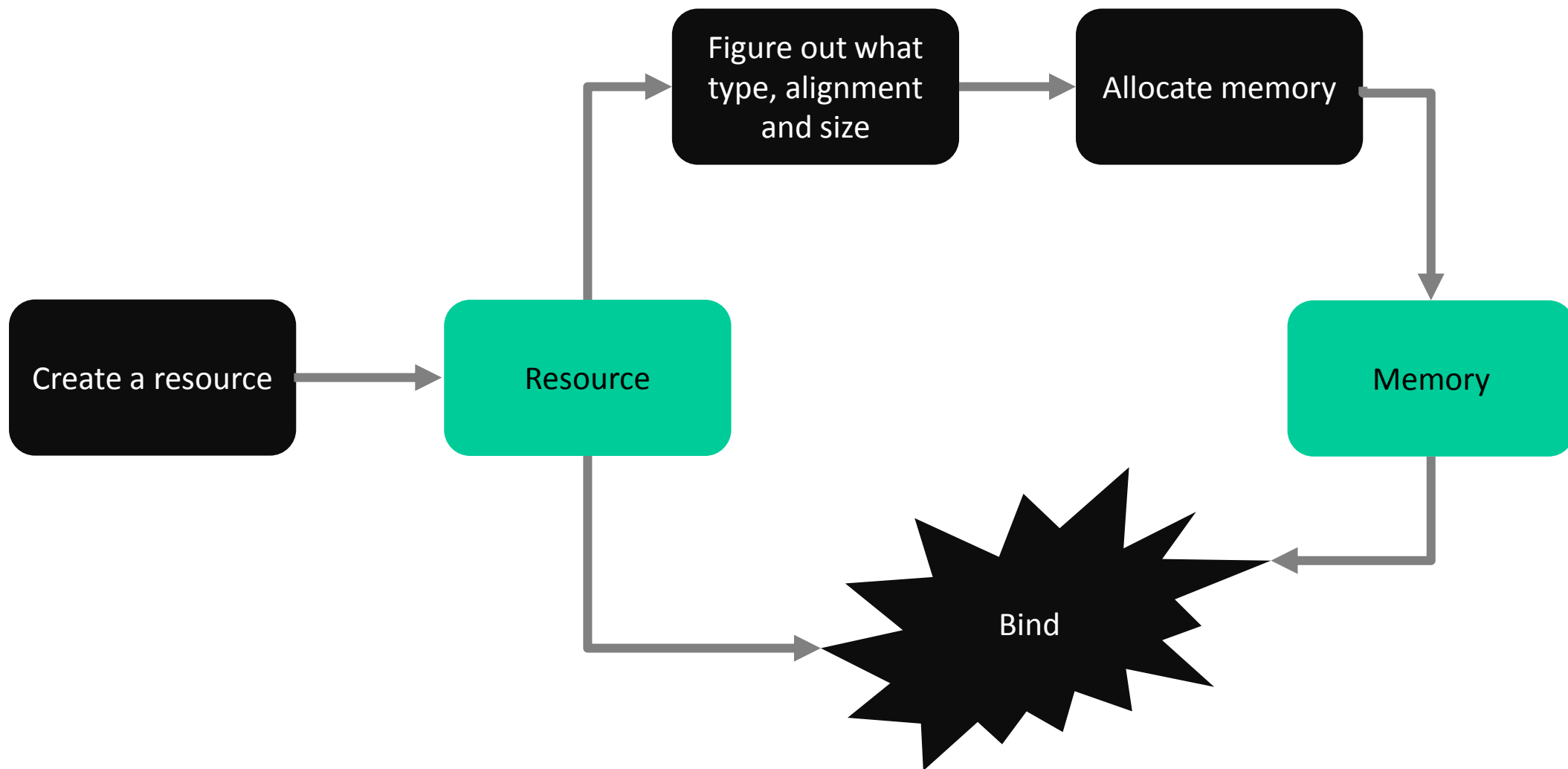
- Heaps and types
- Tips and tricks
- The VMA library
- Conclusion

Effort vs control



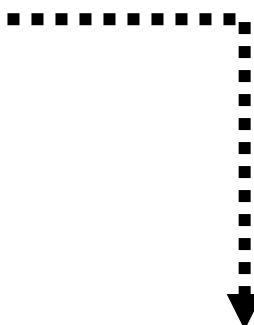
Heaps and types

Allocation



Allocating some memory

```
VkResult vkAllocateMemory(  
    VkDevice device,  
    const VkMemoryAllocateInfo* pAllocateInfo, .....  
    const VkAllocationCallbacks* pAllocator,  
    VkDeviceMemory* pMemory);
```



```
typedef struct VkMemoryAllocateInfo {  
    VkStructureType sType;  
    const void* pNext;  
    VkDeviceSize allocationSize;  
    uint32_t memoryTypeIndex;  
} VkMemoryAllocateInfo;
```

Allocating some memory

```
VkResult vkAllocateMemory(  
    VkDevice  
    const VkMemoryAllocateInfo*  
    const VkAllocationCallbacks*  
    VkDeviceMemory*
```

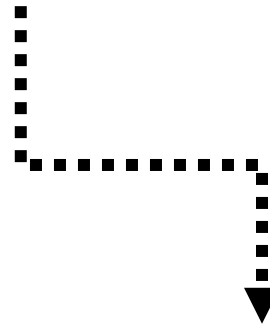
```
device,  
pAllocateInfo, .....  
pAllocator,  
pMemory);
```



```
typedef struct VkMemoryAllocateInfo {  
    VkStructureType    sType;  
    const void*        pNext;  
    VkDeviceSize        allocationSize;  
    uint32_t            memoryTypeIndex;  
} VkMemoryAllocateInfo;
```

Allocating some memory

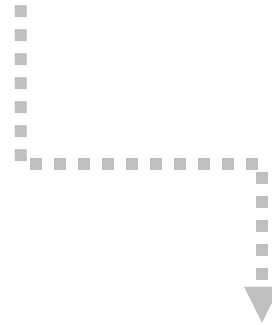
```
vkGetPhysicalDeviceMemoryProperties(  
    VkPhysicalDevice      physicalDevice,  
    VkPhysicalDeviceMemoryProperties* pMemoryProperties);
```



```
typedef struct VkPhysicalDeviceMemoryProperties {  
    uint32_t      memoryTypeCount;  
    VkMemoryType  memoryTypes[VK_MAX_MEMORY_TYPES];  
    uint32_t      memoryHeapCount;  
    VkMemoryHeap  memoryHeaps[VK_MAX_MEMORY_HEAPS];  
} VkPhysicalDeviceMemoryProperties;
```

Allocating some memory

```
vkGetPhysicalDeviceMemoryProperties(  
    VkPhysicalDevice      physicalDevice,  
    VkPhysicalDeviceMemoryProperties* pMemoryProperties);
```



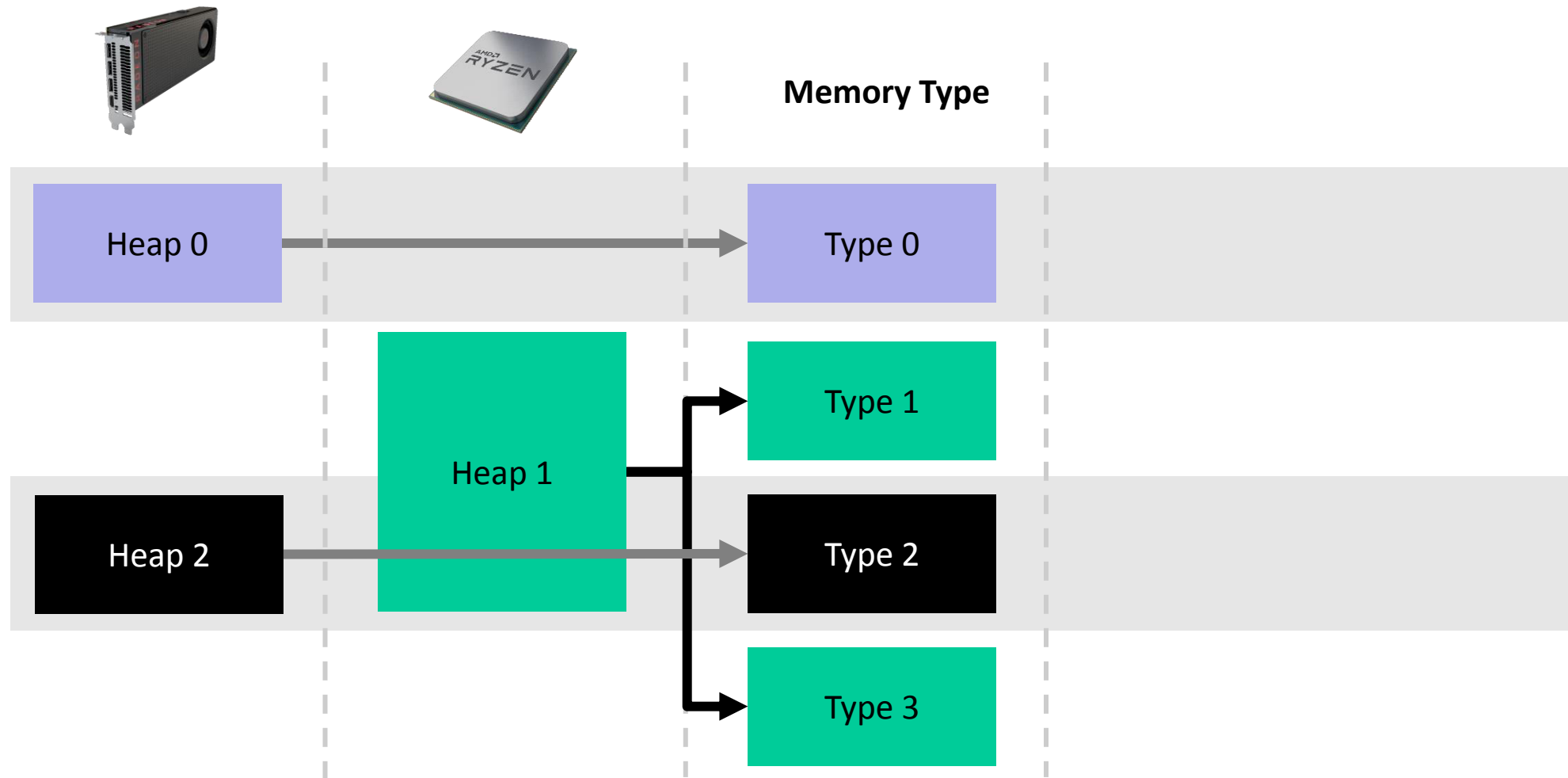
```
typedef struct VkPhysicalDeviceMemoryProperties {  
    uint32_t      memoryTypeCount;  
    VkMemoryType  memoryTypes[VK_MAX_MEMORY_TYPES];  
    uint32_t      memoryHeapCount;  
    VkMemoryHeap  memoryHeaps[VK_MAX_MEMORY_HEAPS];  
} VkPhysicalDeviceMemoryProperties;
```

Allocating some memory

```
typedef enum VkMemoryPropertyFlagBits {  
    VK_MEMORY_PROPERTY_DEVICE_LOCAL_BIT = 0x00000001,  
    VK_MEMORY_PROPERTY_HOST_VISIBLE_BIT = 0x00000002,  
    VK_MEMORY_PROPERTY_HOST_COHERENT_BIT = 0x00000004,  
    VK_MEMORY_PROPERTY_HOST_CACHED_BIT = 0x00000008,  
    VK_MEMORY_PROPERTY_LAZILY_ALLOCATED_BIT = 0x00000010,  
    VK_MEMORY_PROPERTY_PROTECTED_BIT = 0x00000020,  
} VkMemoryPropertyFlagBits;
```

























Memory types vs. heaps

(RX Vega 64)



Memory types cheat sheet

(RX Vega 64)









Memory Type											Size
							R	W	R	W	
0	✓	✓	✓	✗	✗	✗			✗	✗	Most of VRAM
1	✗	✓	✓	✓	✓	✗					
2	✓	✓	✓	✗	✓	✗					Fixed 256MiB
3	✗	✓	✓	✓	✓	✓					

 - Storage  - Visible  - Cached

 - Fast  - Slow

Memory types cheat sheet

(RX Vega 64)

Memory Type											Size
							R	W	R	W	
0	✓	✓	✓	✗	✗	✗			✗	✗	Most of VRAM
1											
2											Fixed 256MiB
3	✗	✓	✓	✓	✓	✓					























Maps to `VK_MEMORY_PROPERTY_DEVICE_LOCAL_BIT` in `VkMemoryPropertyFlagBits`.

 - Storage  - Visible  - Cached

 - Fast  - Slow

Memory types cheat sheet

(RX Vega 64)

Memory Type											Size
							R	W	R	W	
0	✓	✓	✓	✗	✗	✗			✗	✗	Most of VRAM
1	✗	✓	✓	✗	✗	✗			✗	✗	
2	✓	✓	✓	✓	✓	✓					Fixed 256MiB
3	✗	✓	✓	✓	✓	✓					

























Maps to `VK_MEMORY_PROPERTY_HOST_VISIBLE_BIT` and `VK_MEMORY_PROPERTY_CACHED_BIT` respectively.

 - Storage  - Visible  - Cached

 - Fast  - Slow

Memory types cheat sheet

(RX Vega 64)





























Memory Type											Size
							R	W	R	W	
0	✓	✓	✓	✗	✗	✗			✗	✗	Most of VRAM
1	✗	✓	✓	✓	✓	✗					
2	✓	✓	✓	✗	✓	✗					Fixed 256MiB
3	✗	✓	✓	✓	✓	✓					

 - Storage  - Visible  - Cached

 - Fast  - Slow

Memory types cheat sheet

(RX Vega 64)


Memory Type											Size
							R	W	R	W	
0	✓	✓	✓	✗	✗	✗			✗	✗	Most of VRAM
<div> <p>Okay, not <i>that</i> bad since we benefit from GPU caches, but certainly worse than just reading from <code>DEVICE_LOCAL</code>.</p> </div>	✓	✓	✓	✗	✗	✗					Fixed 256MiB
	✓	✓	✓	✗	✗	✗					
	✓	✓	✓	✗	✗	✗					
	✓	✓	✓	✗	✗	✗					

 - Storage  - Visible  - Cached

 - Fast  - Slow

Memory types cheat sheet

(RX Vega 64)



Memory Type	Storage	Visible	Cached	Storage	Visible	Cached	Fast	W	R	W	Size
0	✓	✓	✓	✗	✗	✗	🐰	🐰	✗	✗	Most of VRAM
1	✗	✓	✓	✓	✓	✗	🐢	🐢	🐢	🐢	
2	✓	✓	✓	✗	✓	✗					
3	✗	✓	✓	✓	✓	✓	🐢				

On current GPUs & drivers, PC Windows® everything that is HOST_VISIBLE is also marked COHERENT.

On other architectures you may need:
`vkInvalidateMappedMemoryRanges`
 before reads and
`vkFlushMappedMemoryRanges` after writes.

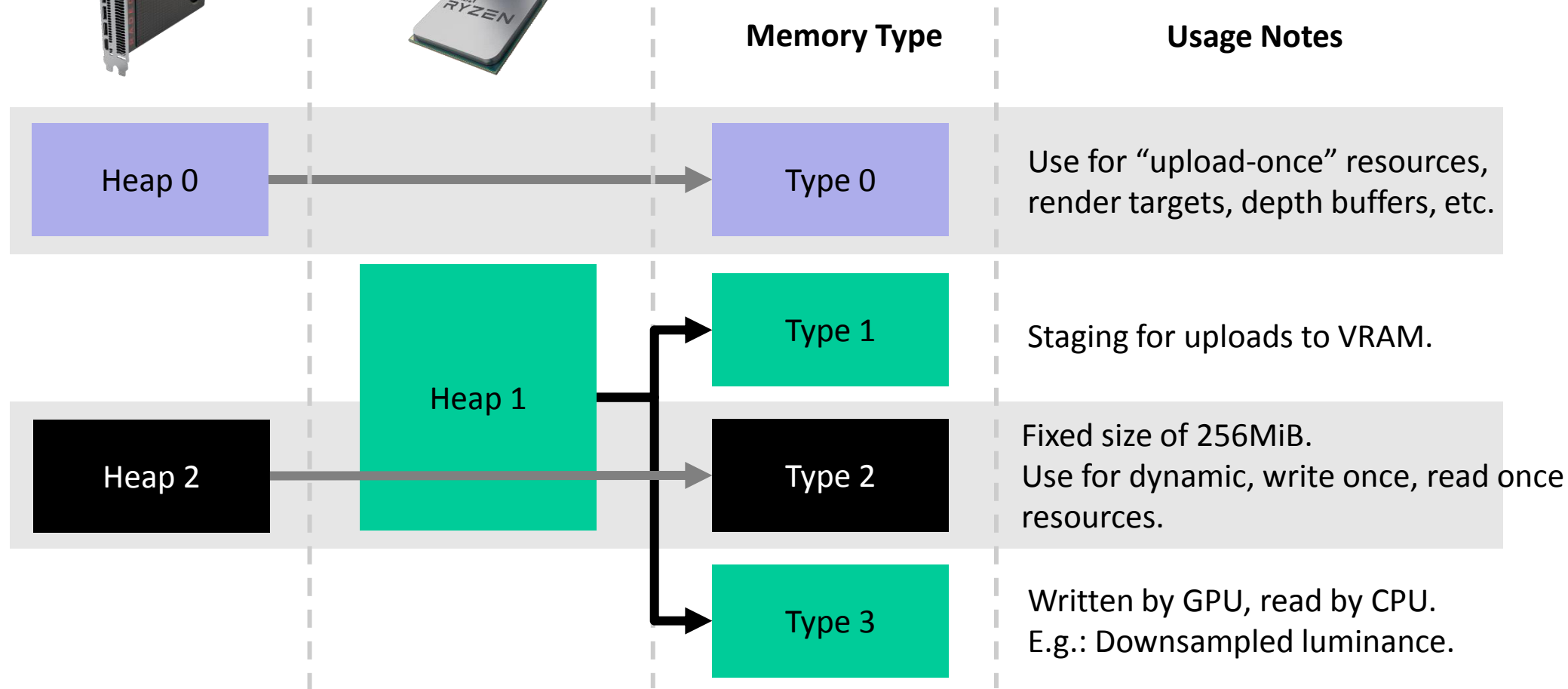
BEWARE: Unmapping won't do this for you!

 - Storage
  - Visible
  - Cached

 - Fast
  - Slow

Memory types vs. heaps

(RX Vega 64)



Warning

```
VkMemoryAllocateInfo x = {  
    .memoryTypeIndex = 2  
};  
vkAllocateMemory(&x, &mem);
```

Warning

```
vkGetPhysicalDeviceMemoryProperties(&properties);  
VkMemoryAllocateInfo x = {  
    .memoryTypeIndex = findAppropriateMemoryHeap(properties);  
};  
vkAllocateMemory(&x, &mem);
```

Warning

- **Not a good idea to hardcode the memory type indices**
 - Driver may change in future
 - May be different on other/newer hardware
 - Magic numbers
- **Query the info using `vkGetPhysicalDeviceMemoryProperties`**
- **Map to engine-specific enums**

Tips and tricks

Allocation strategies

Is this a good idea?

```
for( int i = 0; i < 1000; i++ ) {  
    my_new_objects[i] = new x;  
}
```

Allocation strategies

Is this a good idea?

```
for( int i = 0; i < 1000; i++ ) {  
    my_new_objects[i] = new x;  
}
```



Allocation strategies

Is this a good idea?

```
//for( int i = 0; i < 1000; i++ ) {  
//  my_new_objects[i] = new x;  
//}
```

```
my_new_objects = new x[1000];
```



Allocation strategies

- Same idea on GPU for similar reasons:
 - Fragmentation
 - Performance
 - Data locality
- Allocate reasonably large chunks of memory (256MiB)
 - Just 16 allocations fills 4GiB of VRAM
 - Good balance between flexibility and performance
 - On Windows®7 Vulkan memory allocations have larger overhead
- Sub-allocate the memory for resources from these blocks

Allocation strategies

Linear allocator



Stack allocator



Double stack allocator



Block allocator



Ring buffer



 - Used memory  - Free memory

Over subscription

- When you run out of memory:
 - VK_ERROR_OUT_OF_DEVICE_MEMORY
 - VK_SUCCESS

Over subscription

VK_ERROR_OUT_OF_DEVICE_MEMORY

- Allocation fails
- Application must handle out-of-memory conditions
- Out-of-memory potentially changes per driver/hardware

Over subscription

VK_SUCCESS

- Allocation succeeds
- Some blocks are silently migrated to system memory
- Why would you want this?
 - Useful for development purposes - Artists don't always stick to budgets
 - Some of your blocks might get paged anyway (you're not alone on the machine)
 - Application doesn't have to handle out-of-memory
- Accessing blocks migrated to system memory can degrade GPU performance

Over subscription

- No way is exposed to control residency manually
- No way is exposed to query the used/free memory
- To make things worse, there are other implicit resources which need memory too:
 - Swap chains
 - Command buffers
 - Descriptors
 - Shaders / PSOs
 - Query results
- Use `VkMemoryHeap::size` then apply some “informed adjustments”:

Flags	Adjustment
DEVICE_LOCAL	<code>VkMemoryHeap::size * 0.8f</code>
DEVICE_LOCAL HOST_VISIBLE	<code>VkMemoryHeap::size * 0.66f</code>

Vulkan memory allocator (VMA)

Vulkan memory allocator (VMA)

- Free
- Open source
- MIT license
- Single header
 - <https://github.com/GPUOpen-LibrariesAndSDKs/VulkanMemoryAllocator>
- Simple, C99 interface. Same style as Vulkan™
- Battle tested, already getting some love in the community

Vulkan memory allocator (VMA)

- Function that help to choose the correct and optimal memory type based on intended usage
- Functions that allocate memory blocks, reserve and return parts of them to the user
- Allocation tracker, look at used/unused, and fragmentation
- Respects alignment and buffer/image granularity

Vulkan memory allocator (VMA)

```
VkBufferCreateInfo bufferInfo = { VK_STRUCTURE_TYPE_BUFFER_CREATE_INFO };  
bufferInfo.size = 65536;  
bufferInfo.usage = VK_BUFFER_USAGE_VERTEX_BUFFER_BIT | VK_BUFFER_USAGE_TRANSFER_DST_BIT;  
  
VmaAllocationCreateInfo allocInfo = {};  
allocInfo.usage = VMA_MEMORY_USAGE_GPU_ONLY;  
  
VkBuffer buffer;  
VmaAllocation allocation;  
vmaCreateBuffer(allocator, &bufferInfo, &allocInfo, &buffer, &allocation, NULL);
```

Vulkan memory allocator (VMA)

- Even has some tooling!
- VMA can dump allocator state to JSON
- Python script generates PNG file which shows the allocator contents



Conclusion

- Vulkan is lower-level and requires explicit memory management
 - Creating resources is a multi-stage process
 - Former driver magic is now under your control
- You need to deal with differences between GPUs
- By following good practices you can achieve optimal performance on any GPU
- Vulkan Memory Allocator (VMA) is battle-tested and can really help a lot

Thank-you

- Adam Sawicki
- Dominik Baumeister
- Lou Kramer
- Matthäus G. Chajdas
- Rys Sommefeldt
- Timothy Lottes
- Nicolas Thibieroz
- Alon Or-Bach

Memory aliasing

- As resolutions get larger, render targets follow suit
- As many resources are transient, aliasing can be a solution to keep render target/UAV memory in check

G-Buffers:

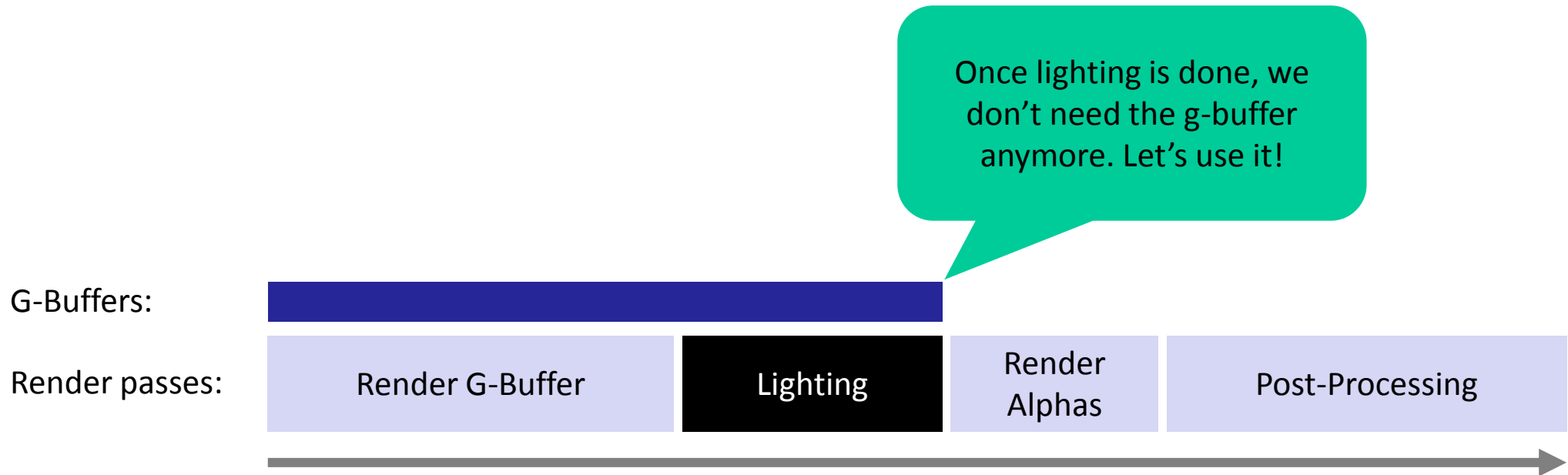


Render passes:



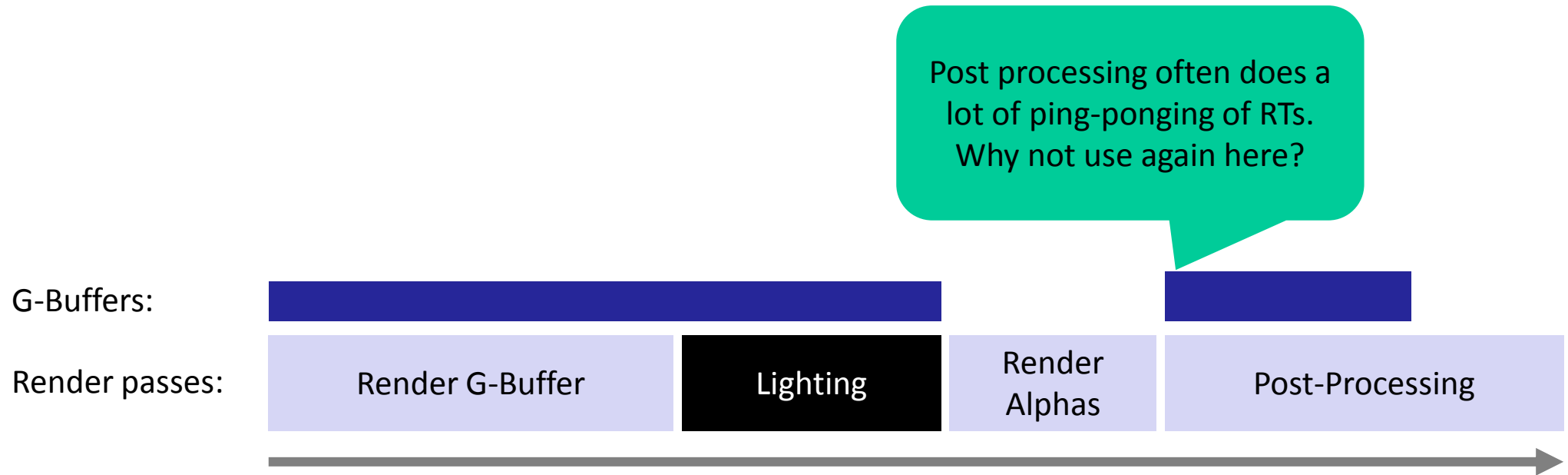
Memory aliasing

- As resolutions get larger, render targets follow suit
- As many resources are transient, aliasing can be a solution to keep render target/UAV memory in check



Memory aliasing

- As resolutions get larger, render targets follow suit
- As many resources are transient, aliasing can be a solution to keep render target/UAV memory in check



Memory aliasing

- As resolutions get larger, render targets follow suit
- As many resources are transient, aliasing can be a solution to keep render target/UAV memory in check

Maybe some compute shader in here needs a nice big UAV for something. No need to allocate, alias with a Render target.

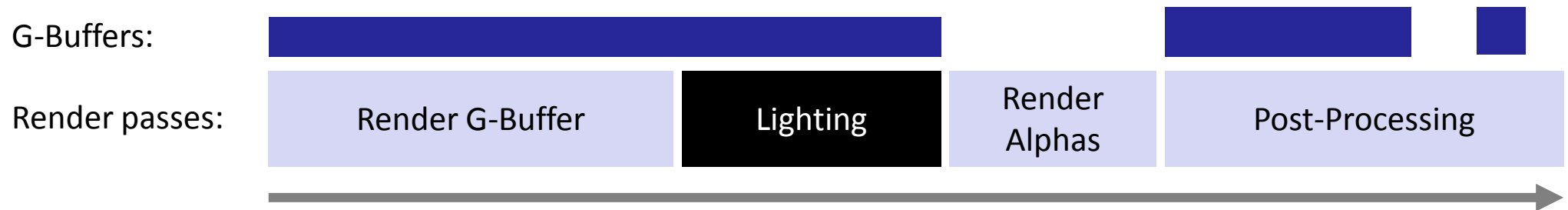
G-Buffers:

Render passes:



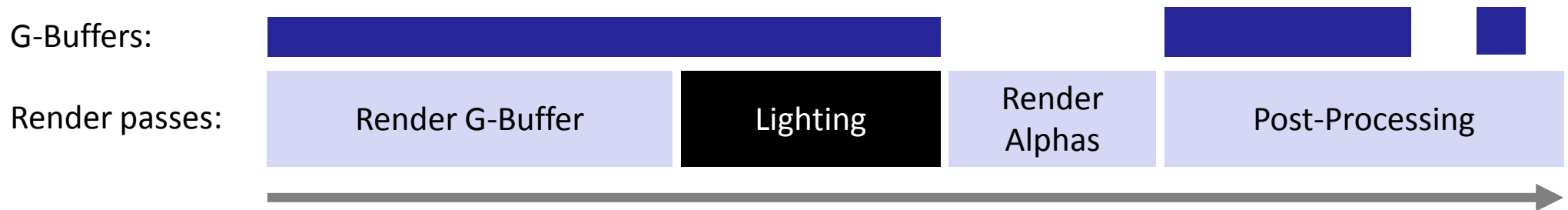
Memory aliasing

- As resolutions get larger, render targets follow suit
- As many resources are transient, aliasing can be a solution to keep render target/UAV memory in check
- For second, third, etc. use of aliased resource best to assume it contains garbage



Memory aliasing

- As resolutions get larger, render targets follow suit
- As many resources are transient, aliasing can be a solution to keep render target/UAV memory in check
- For second, third, etc. use of aliased resource best to assume it contains garbage
- >50% memory saved in some titles [ODonnell17]



Conclusion

- Vulkan is lower-level and requires explicit memory management
 - Creating resources is a multi-stage process
 - Former driver magic is now under your control
- You need to deal with differences between GPUs
- By following good practices you can achieve optimal performance on any GPU
- Vulkan Memory Allocator (VMA) is battle-tested and can really help a lot

Thank-you

- Adam Sawicki
- Dominik Baumeister
- Lou Kramer
- Matthäus G. Chajdas
- Rys Sommefeldt
- Timothy Lottes
- Nicolas Thibieroz
- Alon Or-Bach