Making use of New Vulkan Features
Dr. Matthäus G. Chajdas, Advanced Micro Devices, Inc.
New features!

- The Vulkan working group implemented many features directly based on community feedback
- Too many to cover exhaustively, so I picked ...:
  - VK_EXT_scalar_layout
  - VK_EXT_memory_budget, VK_EXT_memory_priority
  - VK_KHR_8bit_storage and VK_KHR_shader_int8_float16
  - VK_EXT_inline_uniform_block
  - VK_KHR_memory_model
- Ecosystem improvements happening everywhere - a quick look at VMA
EXT_scalar_layout

- The long journey for HLSL compatible layout comes to an end 😊
- Not going to cover this in detail, but we finally have HLSL (and C) compatible layout, and there was much rejoicing!

```cpp
layout(set = 0, binding = 0, scalar) buffer foo {
  vec3 x;
  float y;
};
```

- This is now sizeof(vec4)
- This also applies to uniform buffers!
VK_EXT_memory_*

- Memory management remains a hot topic
- Eviction in Vulkan is completely hidden from the developer
- Two extensions to help handling low-memory situations
  - VK_EXT_memory_priority
  - VK_EXT_memory_budget
VK_EXT_memory_budget

• Provides a way to find out how much memory is available to the application and how much is used

• This relieves applications from tracking it manually and allows them to track memory usage across processes

• Very simple to use:
  - vkGetPhysicalDeviceProperties2KHR
  - Chain in: VkPhysicalDeviceMemoryBudgetPropertiesEXT

• Call it once per frame or every few frames

• If you notice you’re over budget, try to either move things to host memory, or release memory (dropping mip-map levels is usually the best !/$)
VK_EXT_memory_budget
Let’s assume the same situation as you just faced

Let’s zoom in on your app

Let’s assume you don’t check your budget, so the OS decides to clean up. What if the not so important data could get moved out automatically?

<table>
<thead>
<tr>
<th>Your app</th>
</tr>
</thead>
<tbody>
<tr>
<td>Important data</td>
</tr>
</tbody>
</table>
VK_EXT_memory_priority

• Specify a priority per VkMemory at creation time
• When memory becomes scarce, OS will now prioritize what to move to host memory
• Rules of the thumb:
  - Anything you’re writing to should be high priority (render targets, images, etc.)
  - Anything you’re not writing to is probably ok with normal/lower priority, depending on how often you read it
  - If you haven’t measured anything yet, try medium priority for textures and low priority for vertex data – but please measure 😊
• Make sure to double-check that the feature is supported!
• ... and finally: This is all a hint! Try to budget properly, and use this as a “defense in depth”
VK_KHR_8bit_storage

- Sometimes, 8 bit is enough (masks, material attributes, etc.)
- There has been some community demand to access this
- 8-bit storage allows 8-bit values in:
  - Push constants
  - Storage buffers
  - Uniform buffers
- Separate feature bits for each!
- Similar to 16-bit storage
- Note that if you want to use them in shaders, you also want to check for VK_KHR_shader_float16_int8 so you can do arithmetic, etc. without converting first!
VK_KHR_inline_uniform_block

- Not all uniforms are equal

- What if your uniform buffer is just a few floats?
VK_KHR_inline_uniform_block

- Let’s try this instead

<table>
<thead>
<tr>
<th>Uniform buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Descriptor 2</td>
</tr>
<tr>
<td>Descriptor 3</td>
</tr>
<tr>
<td>Descriptor 4</td>
</tr>
<tr>
<td>Descriptor 5</td>
</tr>
<tr>
<td>Descriptor 6</td>
</tr>
<tr>
<td>Descriptor 7</td>
</tr>
<tr>
<td>Descriptor 8</td>
</tr>
</tbody>
</table>

Uniform buffer
VK_KHR_inline_uniform_block

- Saves one indirection when accessing uniforms
- Between push constants and real buffers, but often easier to manage
- Don’t overdo it though, otherwise it degrades into driver managed uniforms
- No visible change on the shader side

![Diagram](image-url)
VK_KHR_memory_model

- Various folks are targeting Vulkan/SPIR-V with their compilers & need assurances about the memory model to reason about behavior
- Game developers prefer optimizations to not break
- The memory model is an implementation detail, but it has various benefits for end users
  - Simple switch for GLSLang/DXC
  - Higher quality & more consistency in the compiler
  - Compilers can do load/store optimizations safely
  - Better testing - helped AMD identify & resolve driver bugs
VMA

- **Vulkan Memory Allocator** - built due to community feedback!
  - Single header file
  - Designed to run on conformant Vulkan implementations
  - Works across multiple operating systems (Windows, Linux, Android, etc.)
  - Used in many community projects

- **New highlights:**
  - Defragmentation on both CPU and GPU
  - New linear and buddy allocators
  - New sparse binding support
  - Record & replay to inspect memory usage over time

- Community provided [Rust bindings](#)
Your feedback matters

- Please keep providing feedback: GitHub, pigeons, etc.
- New extensions ship directly based on user feedback!
DISCLAIMER AND ATTRIBUTION

DISCLAIMER
The information contained herein is for informational purposes only, and is subject to change without notice. This document 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’s products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale. GD-18

ATTRIBUTION
AMD, the AMD Arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Khronos is a registered trademark of the Khronos Group, Inc. Windows is a registered trademark of Microsoft Corporation. Linux is a registered trademark of Linux Torvalds. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.