Vulkanised 2025

The 7th Vulkan Developer Conference
Cambridge, UK | February 11-13, 2025

Teaching Update & Thesis Vulkan Video Extension for Mesa Lavapipe

Helmut Hlavacs, University of Vienna
Bernhard Clemens Schrenk, University of Vienna

Vulkan at the University of Vienna

universität wien

- Bachelor / Master / PhD
- **Do first:** *Foundations of Computer Graphics*

Vulkan **Tutorial**

- ***Real-Time Computer Graphics***

Vulkan Ray Tracing **Engine**

- *Real-Time Ray Tracing*

Vienna Vulkan **Engine**

- *Gaming Technologies*
- *Cloud Gaming*
- *Bachelor / Master‘s theses*

Repeatability, Communication, Updating

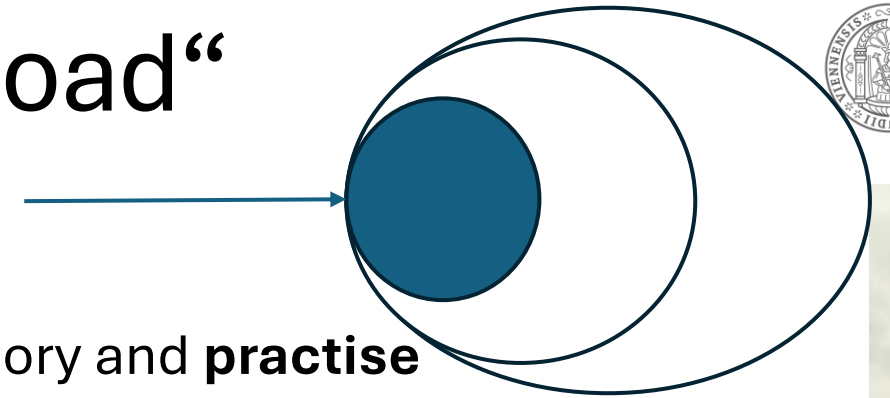
- **Recording the lectures**
- **Moodle page** with examples, resources, slides, dates
- Always know what is expected
- **Forum, active tutors**
- Teach each other
- Who got it working? **Sharing code**
- **Debugging sessions**
- **Keep informed about new developments**

What doesn‘t Work

Theory overload

- Too much (unnessary) theory, too many details
- Too fast progress
- Little or no: practise, time for reflection, understanding
- -> Confusion instead of learning
- -> Unable to use the concepts

Avoid “Theory Overload“



- Focus on essential theory subset
- **Slow** theory progress, **balance** theory and **practise**
- Learn programming by programming -> **habit** formation
- **Scaffolding**
  - Bootstrap, provide **overview** and **guides**, control complexity
  - **Provide working *overall* simple implementation from A to Z**
  - **Look** at existing code, **adapt** this code to **add** new functionality
  - **Reduce line count by fusing lines e.g.**

```
if( condition ) { do_this(); }
```

  - **Grow this base to arrive at your own engine or game**
- Reserve **time** at course **end** to **focus** solely on **programming -> understanding, using, playing around with the subject**

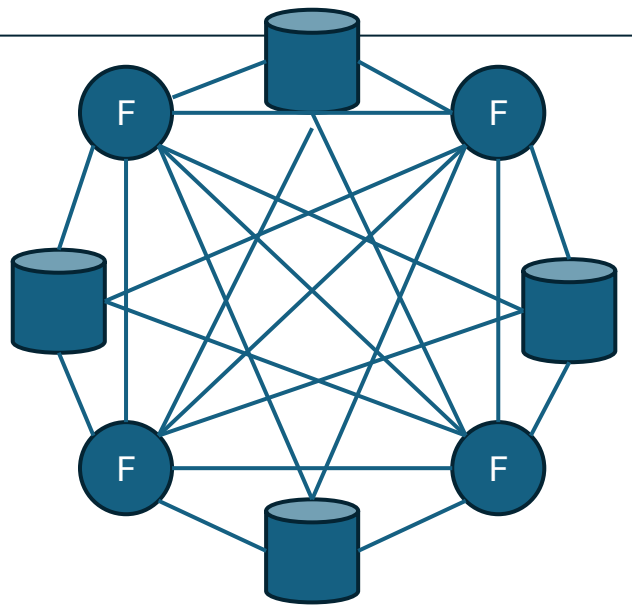
Separation btw. Intent and Implementation

- https://docs.vulkan.org/tutorial/latest/00_Introduction.html
- https://github.com/Overv/VulkanTutorial
- GLFW
- GLM

```
void initVulkan() {
    createInstance();
    setupDebugMessenger();
    createSurface();
    pickPhysicalDevice();
    createLogicalDevice();
    createSwapChain();
    createImageViews();
    createRenderPass();
    createDescriptorSetLayout();
    createGraphicsPipeline();
    createCommandPool();
    createDepthResources();
    createFramebuffers();
    createTextureImage();
    createTextureImageView();
    createTextureSampler();
    loadModel();
    createVertexBuffer();
    createIndexBuffer();
    createUniformBuffers();
    createDescriptorPool();
    createDescriptorSets();
    createCommandBuffers();
    createSyncObjects();
}
```

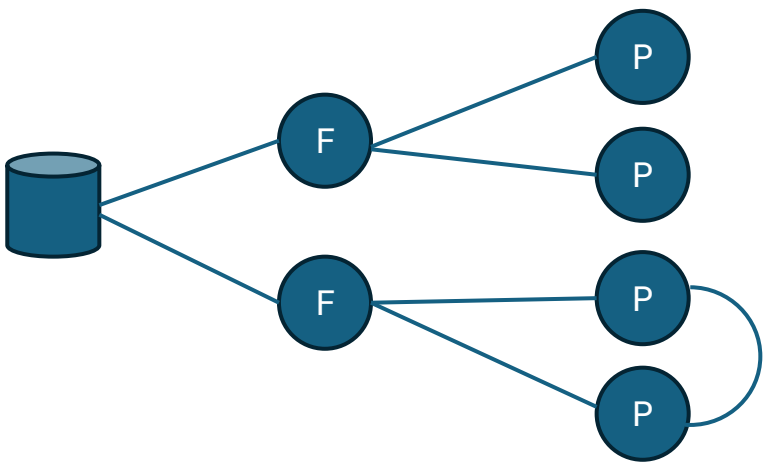
Lower Complexity (!)

```
void function1(…) {
   … // read/write shared data
   function2(…);
   …// read/write shared data
}
```



Intent (stateful)

Implementation (stateless)



Avoid if possible

Adapted Tutorial

- **Pure functions**
- GLM
- **SDL2**
- **Imgui**
- **VMA**
- **Volk**
- **Slang**

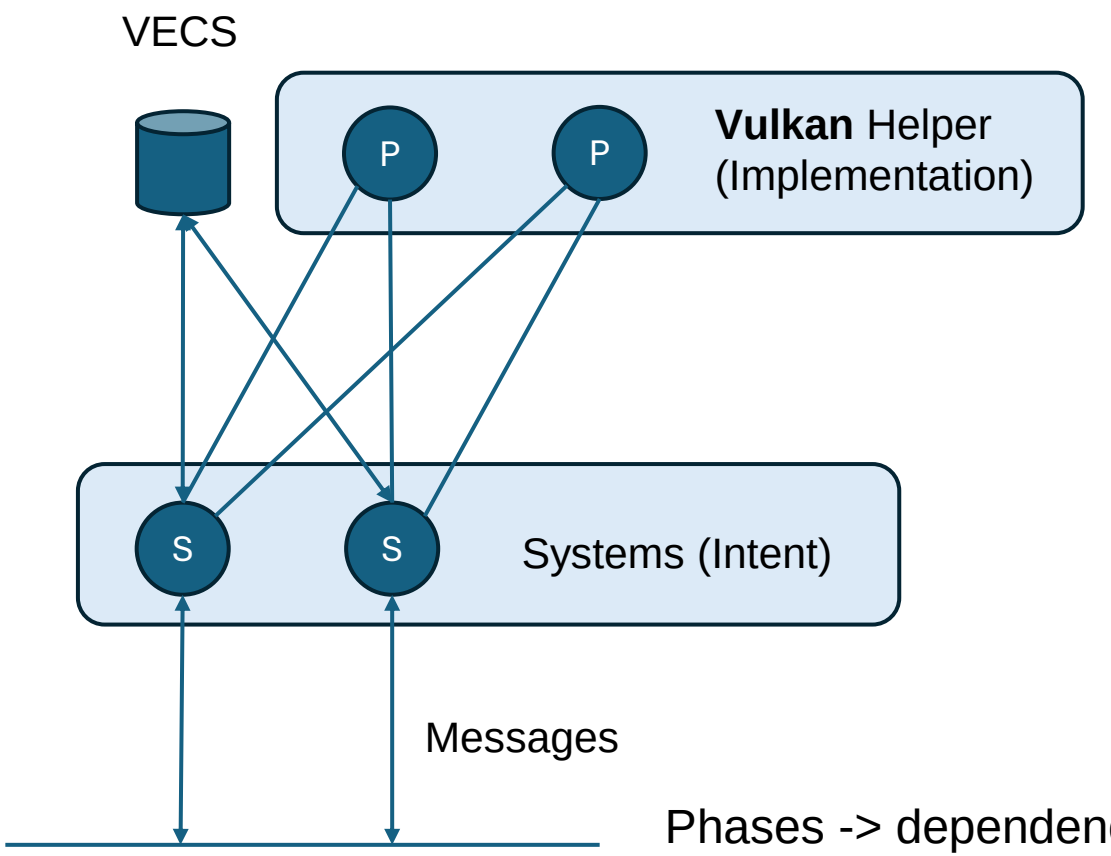
https://github.com/hlavacs/AdaptedVulkanTutorial

```
void initVulkan() {
    createInstance(&m_instance, m_validationLayers);
    setupDebugMessenger(m_instance);
    createSurface(m_instance, m_surface);
    pickPhysicalDevice(m_instance, m_deviceExtensions, m_surfac
    createLogicalDevice(m_surface, m_physicalDevice, m_queueFar
    initVMA(m_instance, m_physicalDevice, m_device, m_vmaAlloca
    createSwapChain(m_surface, m_physicalDevice, m_device, m_sw
    createImageViews(m_device, m_swapChain);
    createRenderPass(m_physicalDevice, m_device, m_swapChain, m
    createDescriptorSetLayout(m_device, m_descriptorSetLayout),
    createGraphicsPipeline(m_device, m_renderPass, m_descriptor
    createCommandPool(m_surface, m_physicalDevice, m_device, m_
    createDepthResources(m_physicalDevice, m_device, m_vmaAlloc
    createFramebuffers(m_device, m_swapChain, m_depthImage, m_i
    createDescriptorPool(m_device, m_descriptorPool);

    createObject(m_physicalDevice, m_device, m_vmaAllocator, m_
      glm::mat4{1.0f}, m_MODEL_PATH, m_TEXTURE_PATH, m_objects)

     createCommandBuffers(m_device, m_commandPool, m_commandBuf
     createSyncObjects(m_device, m_syncObjects);
    setupImgui(m_instance, m_physicalDevice, m_queueFamilies, m
        , m_descriptorPool, m_renderPass);
}
```

Vienna Vulkan Engine 2.0 and Helper Layer

universität wien

Shared global state
(dynamic entities)



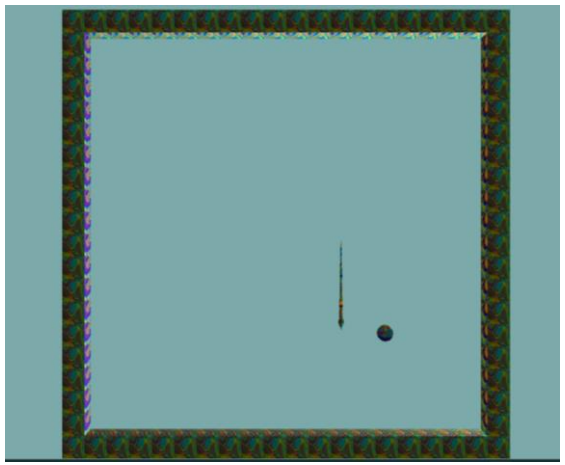
- Pure functions
- Function names show intent and filename
- Single concern
- Stateless,
- Call Vulkan API

- Stateful, single concern
- **No** direct call of Vulkan API
- Show intent, not the details
- Do not offer any public interface
- Isolated

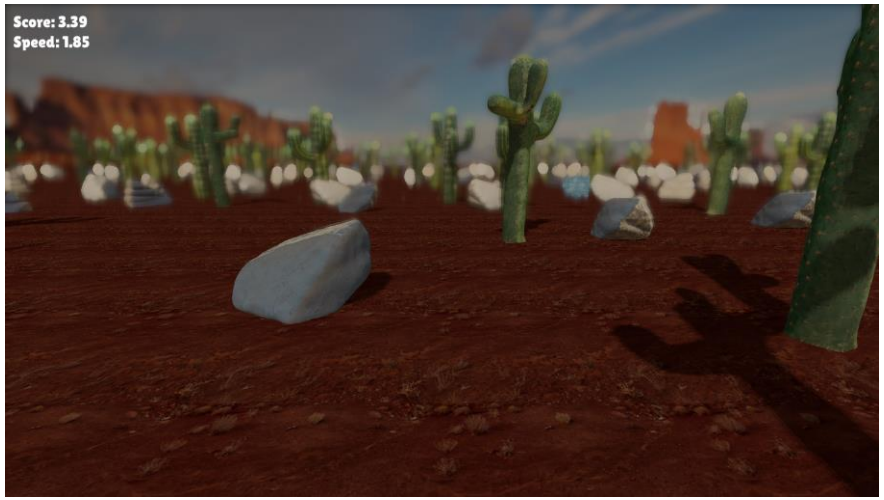
Be Creative and Have Fun

universität wien

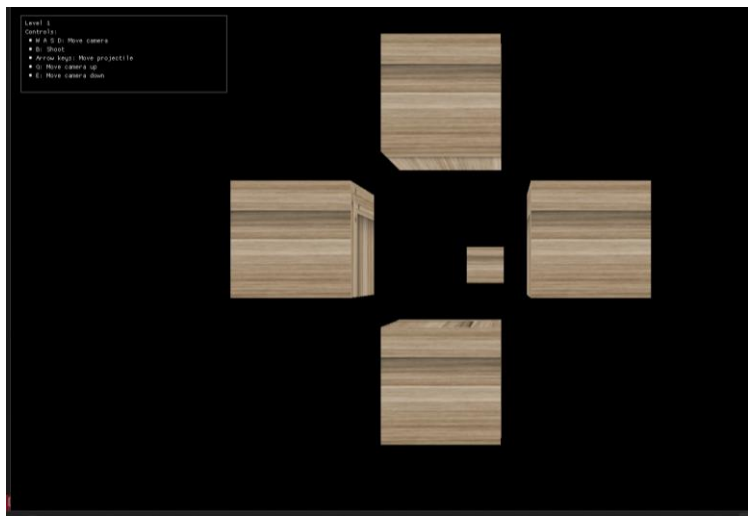
Snake (Tuncer Kerem)



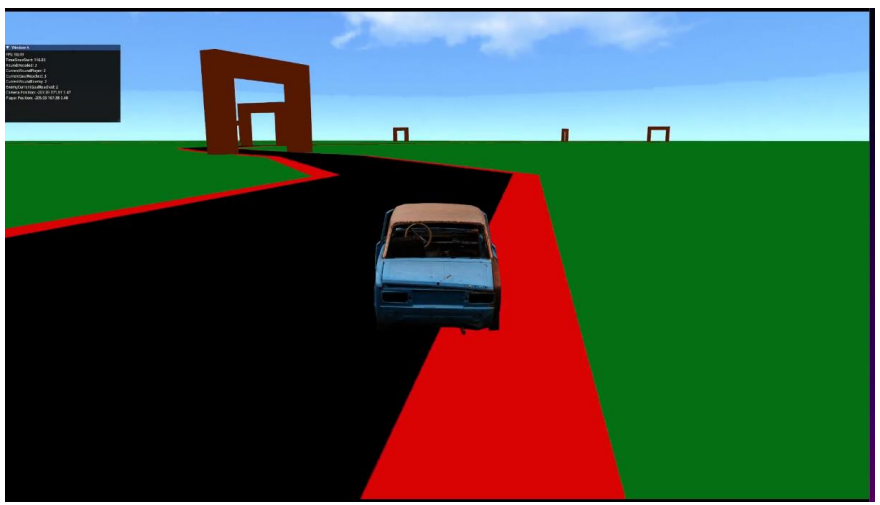
Desert Run (Bernhard C. Schrenk)



Cube Puzzler (Andrei Goje)



Manuel Kart (Manuel Zechmann)



universität wien

University Students vs. Tech Engineers

1.5 - 2 hour lectures + progr.

- Math 1 : Linear algebra
- Math 2: Projection
- C++ primer, C interface
- **Vulkan API**
- Color theory
- Shader language (**Slang**)
- Shading (Phong, PBR, …)
- Lighting, mapping, shadows, GI

20 - 30 minute units + programming

- C++ interface, RAII
- **Vulkan API** + more on
  - Extensions and Layers
  - Maintenance
  - Specification **versions**
  - Synchronization2
  - More examples: MSAA, CS, …
  - Tools

Resources and Contact

universität wien

- Adapted Tutorial: https://github.com/hlavacs/AdaptedVulkanTutorial
- Vienna Vulkan Engine: https://github.com/hlavacs/ViennaVulkanEngine
- **Recorded Lectures**: helmut.hlavacs@univie.ac.at
  Introduction: https://www.youtube.com/watch?v=7Ijalz6RLr4
- Linked-In: https://www.linkedin.com/in/helmut-hlavacs-972b9aa/

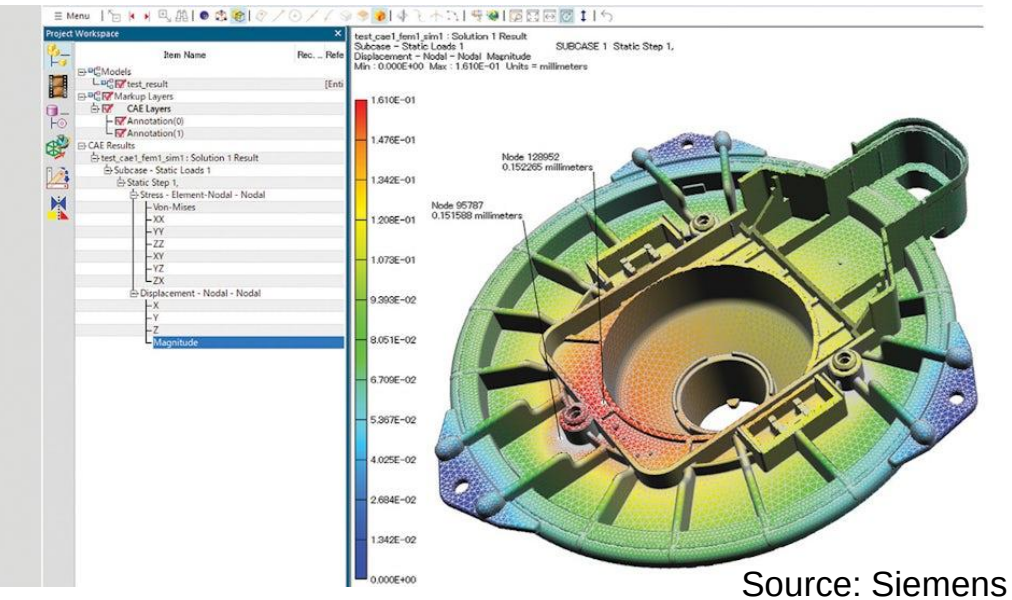
Thesis: Evaluation of the Feasibility of Implementing Vulkan Video Extensions on CPUs

universität wien

➔ Can Vulkan Video Extensions be fully implemented on CPUs?

- **Non-Academic Relevance?**

  There is need in the industry:
  - Testing and Reference implementations
  - Legacy systems
  - Visualization Systems without video encoder hardware



Source: Siemens

Video Codec APIs – A Survey

- Grouped by **Availability:**
  - Vendor specific (Nvidia Video Codec SDK, Intel VPL, ...)
  - Operating-System specific (Direct3D 12 Video, ...)
  - Vendor & OS neutral (Vulkan Video Extensions)

- Key **Challenges:**
  - **One API usable everywhere** -> Reduces application implementation effort
  - **Software Codec (CPU) fallback** -> Works without dedicated video HW

- Survey **Conclusion:**
  - ***Missing:*** Vendor & OS independent API with software fallback

Vulkan Video Extensions Use Cases

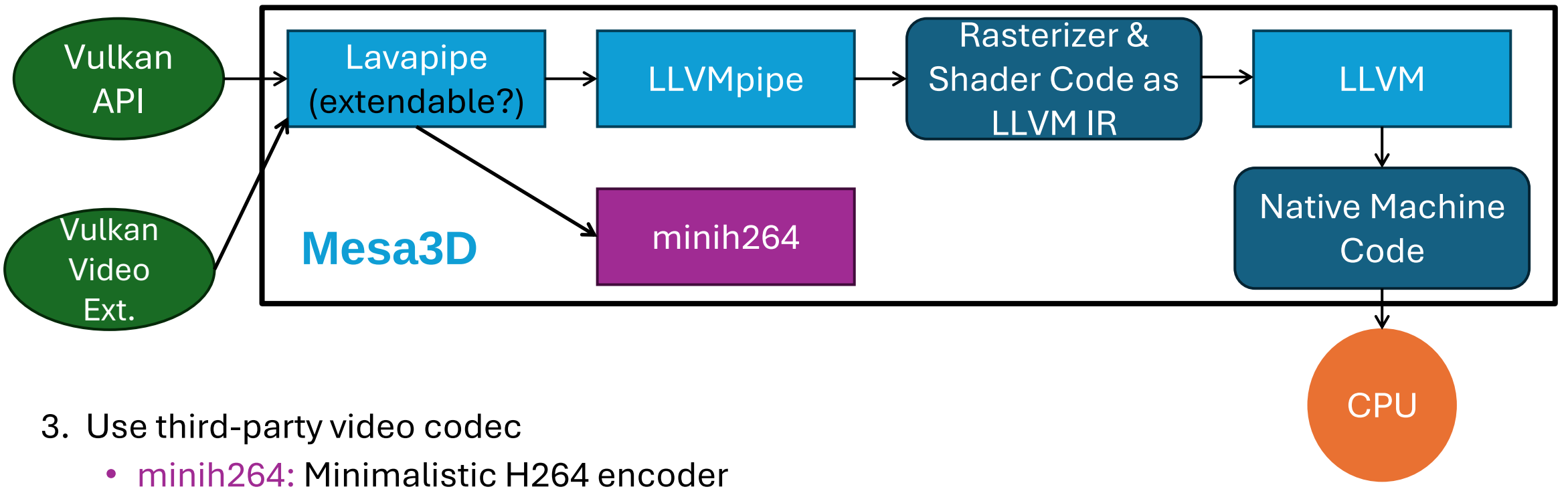
- **Game Streaming** / Cloud Gaming: low latency
- In-Game **Video Playback:** see video
- **Scientific** Data **Visualization** with video output
- **Video Editing:** live preview, live filter (shader)
- **Video Transcode**
- **Surveillance:** decode multiple video streams
- **AI Video Applications:** decode video & apply AI algorithms directly on the GPU
- and many more…

From my last year's talk:
Vienna Vulkan Engine with
Vulkan Video Decode support:
Streaming video directly onto a texture

Where to begin?

universität wien

1. Define a subset to begin with (H.264 Baseline Profile, Encode only)
2. Use existing Vulkan graphics API implementation as base
   - **Mesa3D Lavapipe:** Software (CPU-based) rasterizer (3D renderer) with Vulkan API



3. Use third-party video codec
   - minih264: Minimalistic H264 encoder

universität wien

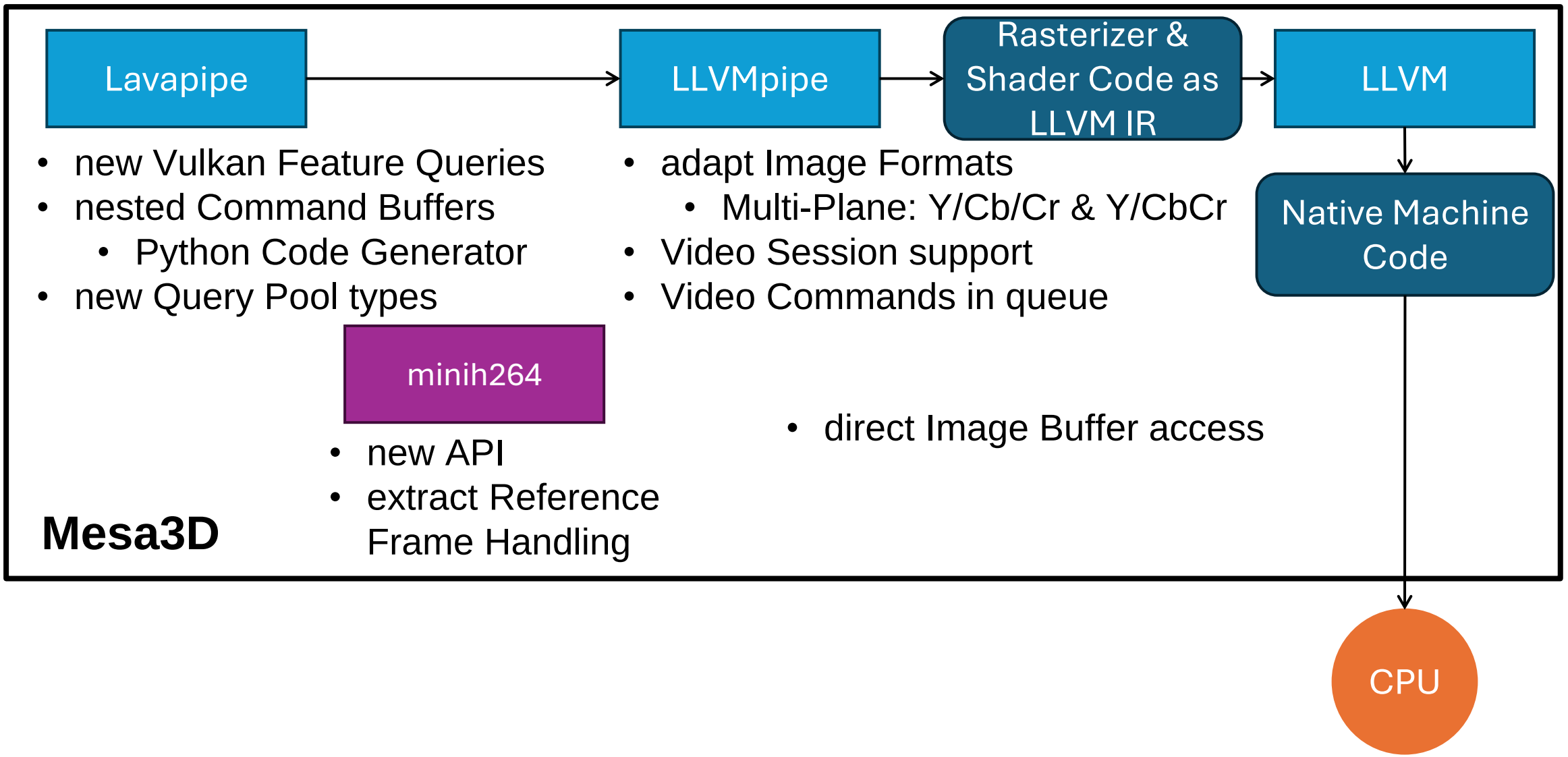
Mesa3D/Gallium Lavapipe

CPU-based Rasterizer with Vulkan API

- mesa
  - src
    - gallium
      - drivers
        - llvmpipe
      - frontends
        - lavapipe
    - vulkan

- available for Windows & Linux
- No need for dedicated hardware (but much slower)
- Vulkan 1.4 compliant
- Good platform for experiments with Vulkan Extensions (driver side)
- Minimum dependencies – easy to compile

What to add?

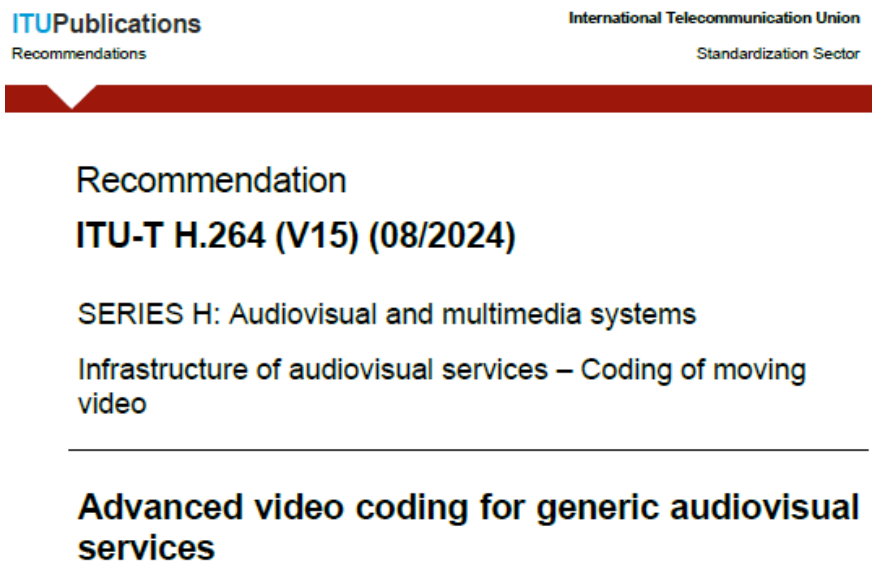

Lavapipe
LLVMpipe
Rasterizer & Shader Code as LLVM IR
LLVM
Native Machine Code
CPU
• new Vulkan Feature Queries
• nested Command Buffers
• Python Code Generator
• new Query Pool types
• adapt Image Formats
• Multi-Plane: Y/Cb/Cr & Y/CbCr
• Video Session support
• Video Commands in queue
minih264
• new API
• extract Reference Frame Handling
• direct Image Buffer access
Mesa3D

How to Evaluate?

universität wien

- Does the implementation *conform* to
  1. the *Vulkan specification* and
  2. the *video encoding standards*?
- **Environment:**
  - **Linux & Windows -** Extended Lavapipe used as Vulkan Graphics Driver
- **Functionality Tests:**
  - **Vulkan Video Sample Code** (1. Nvidia, 2. my own)
  - **Vienna Vulkan Engine** with Video Encoding support
- **Conformance Tests:**
  - **Vulkan Conformance Test Suite** (Vulkan CTS) for Drivers & Hardware
  - Decompressing encoded video and comparison (**Reconstruction Quality Metric**)

Vulkan® 1.4.304 - A Specification (with all registered extensions)


ITUPublications
Recommendations

International Telecommunication Union
Standardization Sector

Recommendation
ITU-T H.264 (V15) (08/2024)

SERIES H: Audiovisual and multimedia systems

Infrastructure of audiovisual services – Coding of moving video

Advanced video coding for generic audiovisual services

universität wien

Vulkan CTS

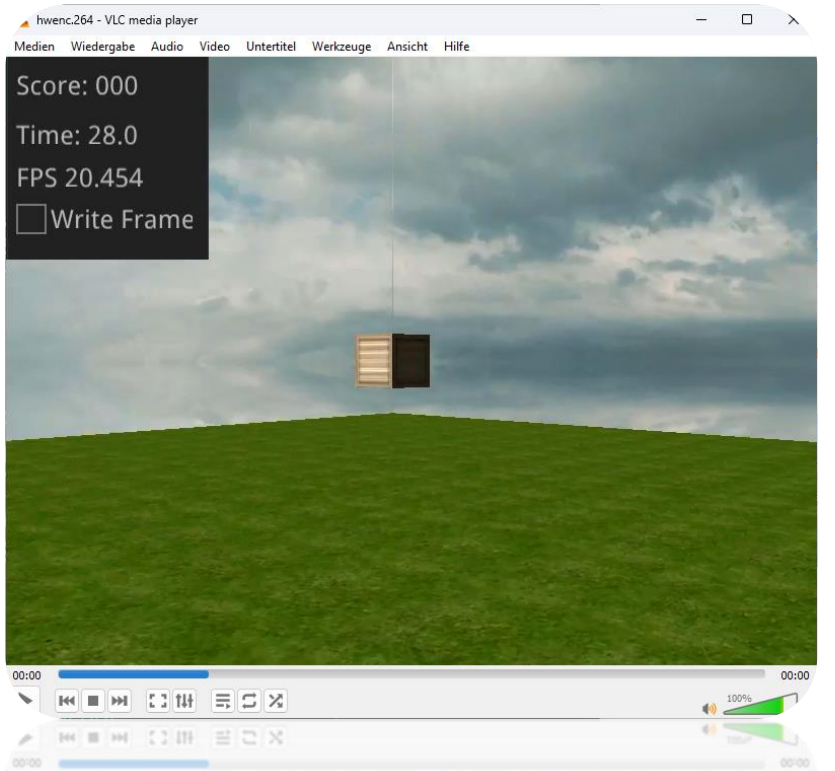
Vulkan Conformance Test Suite

- Thousands of automated tests for Vulkan drivers
- Windows & Linux
- Growing number of tests for video extensions
- Easy setup – own script for fetching dependencies
- Required to pass for conformant products
- Also suitable as example code

Results

Functional Tests:

- Encoder Examples: working ✅
- Rendering & Video Encoding with Vienna Vulkan Engine: working ✅
- Advantage: **Modifying the application not necessary** -> simply use Mesa driver as Vulkan device

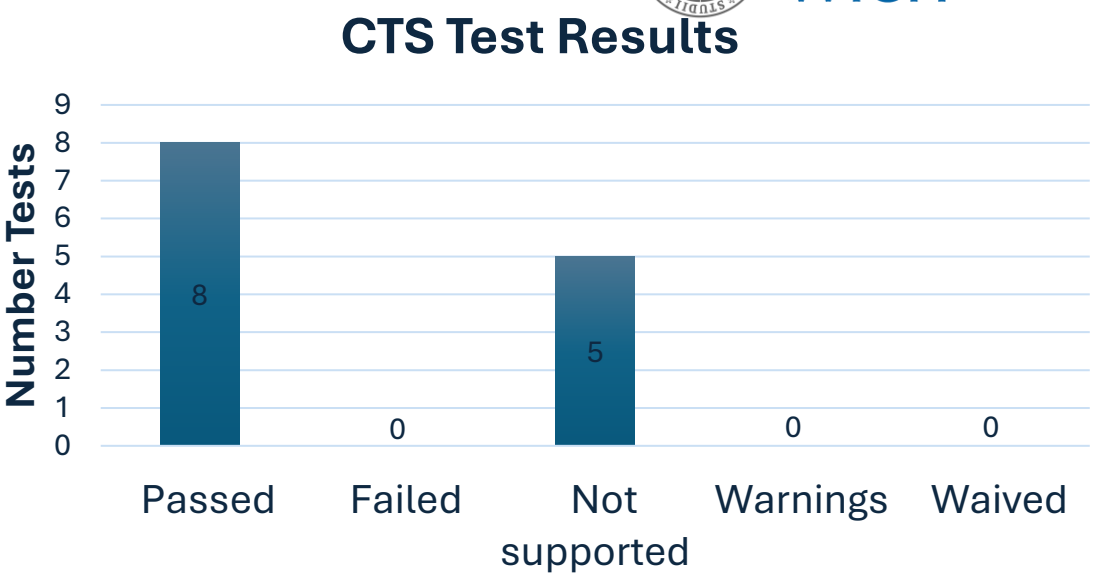


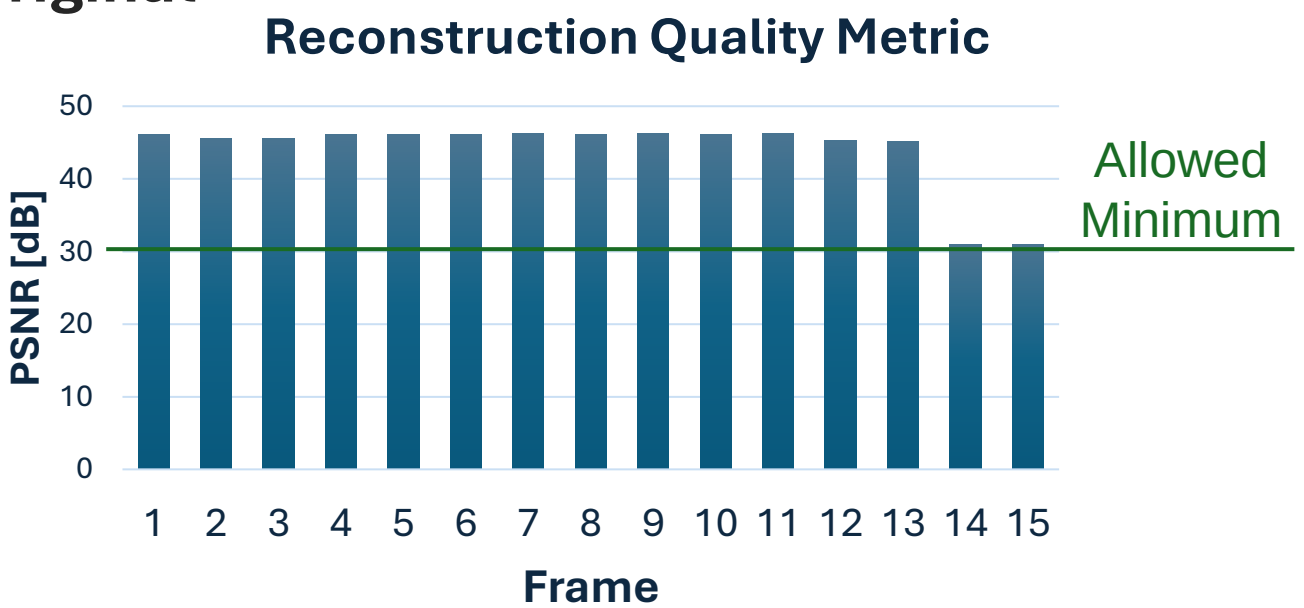
H.264 video file encoded on CPU by
Vienna Vulkan Engine
with Vulkan Video support
running with Mesa3D Lavapipe driver

Results

Conformance Tests:

- Vulkan Conformance Test Suite:
  Test Set:
  “dEQP-VK.video.encode.h264.*”
  100% of supported Tests passed
- Encoded Video is **equivalent to original** video
  - CTS criteria: PSNR > 30 dB (Peak Signal to Noise Ratio)
  - Frame 14+15: reduced resolution test





Outcome

universität wien

- CPU based **H.264 encoding** as **Vulkan Video Extension** in **Lavapipe**
- Can be used together with LLVMpipe rasterizer or with another video source
- Provides a **viable solution** for systems **without dedicated** video compression **hardware**
- Can be used as second Vulkan device to combine with GPUs not supporting Video Extensions

- What can come next**:**
  - Add **video decode** support
  - **Extend feature** support (higher profiles, B frames, …)
  - **Performance** (Speed) Evaluation & Optimizations
  - Integrate with **faster codecs** depending on licensing situation (x264 is under GPL)
  - Extend support to **additional coding standards** (AV1)
  - **Merge** support **upstream** into official Mesa drivers