6 Years of Teaching Vulkan with Example for Video Extensions

Helmut Hlavacs, University of Vienna
Bernhard Clemens Schrenk, University of Vienna
Computer Science at University of Vienna, Austria

- Founded in 1365, 10600 staff, 84600 students, 15 Faculties, 5 centers

- Wintersemester: October – January
- Summersemester: March – June
- https://informatik.univie.ac.at/en/

- Bachelor 6 semesters / 180 ECTS credits
- Master: 4 semesters / 120 ECTS credits
- PhD: 6 semesters

Programs:

- Computer Science
- Media Informatics
- Business Informatics
- Data Science
- Medical Informatics
- Business Analytics
- Digital Humanities

Languages:

- German
  Students from Austria, Germany
- English
  International Erasmus Exchange
Courses related to Vulkan API

**Bachelor program:**
- *Electives*
- Bachelor’s thesis

**Master program:**
- *Electives*
- Gaming Technologies *(VVE)*
  - Lab 1
  - Lab 2
  - Master’s thesis

**Cluster Computer Graphics**
- Foundations of Computer Graphics (Gatekeeper)
- Cloud Gaming *(VVE)*
- Real-Time Ray Tracing *(OpenGL)*
- Image Synthesis

Cluster Algorithms
Cluster Data Analysis
....

Real-Time Computer Graphics – Lecture

- Introduction to the Vulkan API and the **Vulkan Tutorial**
- C++ Primer
- Mathematics 1: LinAlg, Rotation, Affine Mappings, Reference Frames
- Mathematics 2: Model - View – Projection, NDC, Viewports
- Vulkan API - Introduction, Instance, Debug, Surface
- Vulkan API - Device + Queue, Swapchain, Command Pools/Buffers
- Vulkan API - Synchronization, Render Pass + Frame Buffer
- Vulkan API - Memory, VMA, Buffers, Images
- Vulkan API - Descriptor Sets and Layouts, Pipeline Objects / Blending
- Vulkan API - Pipeline Objects, GLSL
- Lighting and Shading - BRDF, Light types, Phong
- Lighting and Shading - PBR, Fresnel, Micro Facet, Geometry, Metal/Roughness
- Maps (Texture, Normal, Shadow)

~580 slides
All lectures are **recorded** and can be downloaded from the Moodle: **14x 90 min videos**
A Sample Lecture

Creating Surfaces

E.g., Win32:

```
VkResult vkCreateWin32SurfaceKHR(VkInstance instance, const VkWin32SurfaceCreateInfoKHR* pCreateInfo, const VkAllocationCallbacks* pAllocator, VkSurfaceKHR* pSurface);
```

```
typedef struct VkWin32SurfaceCreateInfoKHR {
    VkStructureType   sType;
    const void*       pNext;
    VkWin32SurfaceCreateFlagsKHR flags;
    HANDLE mInstance;    //Win32 INSTANCE for Win32 window
    HWND hWnd;            //Win32 HND
} VkWin32SurfaceCreateInfoKHR;
```

E.g., GLFW:

```
glfwWindow* window = glfwCreateWindow(640, 480, "Window Title", NULL, NULL);
VkSurfaceKHR surface;
VkResult err = glfwCreateWindowSurface(instance, window, NULL, &surface);
```

Real-Time Computer Graphics  SoSe 2023

Surface Present Modes

```
VkResult vkGetPhysicalDeviceSurfacePresentModesKHR(
    VkPhysicalDevice physicalDevice,
    VkSurfaceKHR surface,
    uint32_t* pPresentModeCount,
    VkPresentModeKHR* pPresentModes);
```

```
typedef enum VkPresentModeKHR {
    VK_PRESENT_MODE_IMMEDIATE_KHR = 0,
    VK_PRESENT_MODE_MAILBOX_KHR = 1,  //1 element in queue, replace previous
    VK_PRESENT_MODE_FIFO_KHR = 2,    //N elements, wait if full (must support)
    VK_PRESENT_MODE_FIFO_RELAXED_KHR = 3,
    VK_PRESENT_MODE_SHARED_DOUBLE_BUFFER_KHR = 1000111000,
    VK_PRESENT_MODE_SHARED_SINGLE_BUFFER_KHR = 1000111001,
    VK_PRESENT_MODE_MAX_ENUM_KHR = 0x7fffffff
} VkPresentModeKHR;
```

Real-Time Computer Graphics  SoSe 2023

Querying Surfaces

```
VkResult vkGetPhysicalDeviceSurfaceCapabilitiesKHR(
    VkPhysicalDevice physicalDevice,
    VkSurfaceKHR surface,
    VkSurfaceCapabilitiesKHR* pSurfaceCapabilities);
```

```
typedef struct VkSurfaceCapabilitiesKHR {
    VkFormat format;
    VkSurfaceFormatKHR colorSpace;
    VkSurfaceFormatKHR depthFormat;
    VkSurfaceFormatKHR mipsFormat;
    VkSurfaceFormatKHR temporalFormat;
} VkSurfaceCapabilitiesKHR;
```

```
VkResult vkGetPhysicalDeviceSurfaceFormatCountKHR(
    VkPhysicalDevice physicalDevice,
    VkSurfaceKHR surface,
    uint32_t* pSurfaceFormatCount,
    VkSurfaceFormatKHR* pSurfaceFormats);
```

```
typedef struct VkSurfaceFormatKHR {
    VkFormat format;
    VkColorSpaceKHR colorSpace;
} VkSurfaceFormatKHR;
```

Real-Time Computer Graphics  SoSe 2023

Time for Code

Create surface
Real-Time Computer Graphics – Lab Assignments

6-7 assignments + a personal game project

1. Install the Vulkan SDK, compile and run the Vulkan Tutorial
2. Scenegraph Worldmatrix and Vulkan Basics
3. Physical Device, Swap Chain, Command Pools and Buffers, Synchronization
4. **Memory, Buffers and Images** + Game Topic
5. Descriptor Sets, Pipeline, View port and GLSL Programming + Game Design Document
6. Blinn-Phong BRDF and more objects (and textures)
7. **Your Game**
A Sample Assignment – Task 04 – Memory, Buffers and Images

1) ...

2) ...

3) ...

4) Change the tutorial code, so that it no longer uses vkCmdDrawIndexed(), but so that it uses vkCmdDraw(). This means you no longer use an index buffer for drawing! Now triangles are defined only by three successive vertices in the vertex buffer!

So after having loaded vertex and index information from disk, create another vertex buffer that holds exactly 3 vertices for each triangle (and therefore many redundant copies of vertices.). Then use this vertex buffer for drawing.

...
Other Vulkan Related Courses and the VVE

**Gaming Technologies**
- **Physics** simulation: Time stepping, rigid body kinematics, collision detection/resolution, ...
- **AI** for Games (NPCs): Movement, Path finding, FSM, Decision/Behavior Trees, MCTS, GOAP, ...
- Your Game (using AI and physics)

**Cloud Gaming**
- Video encoding (FFMPEG) and streaming
- Multimedia Networking: IP 4/6, UDP/TCP, STUN/TURN, RTP, SIP, buffer dynamics, ...
- GUI frameworks
- Audio
- Your Cloud Game (Server, Client)

**The Vienna Vulkan Engine (VVE)**
- C++ rendering framework
- Windows / Linux / ~MacOS
- Vulkan API / GLSL
- GLM / GLFW / Assimp / Nuklear
- Shadow maps
- Threadpool, Screenshots
- **Eventlisteners**
- Rendering, learning Vulkan, basis for implementing new stuff
- [https://github.com/hlavacs](https://github.com/hlavacs)
Quackblast (Lamies Abbas)

Cubemania (Orcun Ilker Döger)

Sphere Fighter (Paul Friedrich Pesak)

Zombie Fighter (Jan Mesner)
Example Bachelor’s Thesis: Graphical Vulkan Editor

https://github.com/Schokolado/GraphicalVulkanEditor

Generate C++ Source code

Create and visualize pipelines
Example Master’s Thesis: 3 Vulkan Renderers


Implementation for the Vienna Vulkan Engine
• Deferred renderer
• Ray Tracing renderer (NVIDIA extension)
• Ray Tracing renderer (Khronos extension)

Research questions
• Which rendering algorithm performs better on specific hardware?
• Performance Ray Tracing vs rasterization?
• Performance Nvidia extension vs Khronos extension?
Lessons Learned in 6 Years Wrestling the API

Teacher Instrospection
• Massive work learning the API
• Making the engine is a challenge but good investment
• Recording pays in the long run but lecture rooms get quite lonely
• Bring all to the same level at the start
• Huge amount of content – all or nothing
• Balance with other content? Lighting, shading, mapping, shadows, AO, ...
• C or C++ interface?
• Include Queries? Render Pass?

Extraspection – Students
• Great interest in games related courses
• International students inhomogeneous
• Differences in Mathematics / C++
• Make good use of the Moodle forum
• In summer 2023 20% failed RTCG
• Afterwards students do know the API
• Very creative wrt game ideas
• High motivation, like heavy implementation
• Some wait until the last day – and then some
• Presentations are big fun and surprising
My Computer Graphics Course Journey

1. Foundations of Computer Graphics
2. Real-Time Ray Tracing
3. Gaming Technologies
4. Cloud Gaming
5. Lab 1: Vulkan Video Encoding
6. Lab 2: Vulkan Video Decoding
7. Upcoming: Master Thesis
My experience during the courses

• First: Learning a good mathematical base
• Followed by: State of the art technology and APIs
• Perfectly guided through every topic by small tasks
• Much fun with own projects
• Finding: I am not a designer, but
• I love writing code &
• working on game engines

• Finally: Inspiration where I can continue
  • master thesis
Cloud Gaming / Video Encoding

• Why this topic for my lab courses & master thesis?
• Course Cloud Gaming – easy approach:
  • Grab pictures to host memory
  • Encode with FFMPEG
  • Send to client

• I wanted to improve that -> do it on the GPU
• Game Engine: Vulkan based
• Vulkan Video Extensions just on their way to get released (1 year ago)
Starting Point – Vulkan Video

• Provisional Specifications for Vulkan Video Encode
• NVIDIA Vulkan Beta Driver

• No validation layer support for Video Encode in SDK
• One sample: nvpro-samples/vk_video_samples
  • Encoding from YCbCr raw data file
  • Producing Intra frames only
  • Working with one specific revision
The progress / What to do differently

- Many Ups and Downs
- Extracting the relevant code from the example
- First working own code, took some time
- Even longer way until first P frame (reference picture list)
- Stopped working after every specification revision update
- Many bugs found after final SDK release (with validation layer)

- Next time: Get more in contact with the involved people
- Tip: Read the proposal document containing example snippets
What I learned technically

• Better understanding of Synchronization in Vulkan
• Working with multiple queues
• Working with different image layouts
• Working with provisional Vulkan APIs
• Too much details about H.264

• All this based on a solid base learned during the lectures before
Outcome & Deliverables

• Vulkan Video integrated in Vienna Vulkan Engine
  • Video Textures from H.264 files
  • Get rendered content as H.264 stream
    • GOP structure with I and P frames
• Simple example code for
  • Vulkan Video Encode Extension (updated for the finally released revision)
  • Vulkan Video Decode Extension
  • One CPP file each
• Not a complete H.264 implementation, but trimmed for education
  • As base for other implementations and future experiments
  • github.com/hlavacs/ViennaVulkanEngine/tree/vulkanvideo_encode
  • github.com/clemy/ViennaVulkanEngine/tree/videodecode
Summary as Student

• Vulkan as base for all courses
  • Gaining knowledge in modern computer graphics APIs
  • HW & OS independent: Students can use any platform
  • Immediately use the knowledge for visualization in related courses:
    • AI for NPCs
    • Game Physics
  • Reuse own code and ideas and improve it in every course

• Perfectly supported my interests in
  • Coding
  • Math
  • New Technologies

• Outlook: A solid base for multiple possibilities
  • An academic career
  • A job in the (game) industry