CSCI 420 Computer Graphics
Bonus Lecture

Vulkan

Jernej Barbic
University of Southern California

Overview

- A low-level API for 3D computer graphics by the Khronos group
- First version in 2016; actively developed to this day
- The “successor” of OpenGL
- Steep learning curve, complex user code
- Great control over the GPU

Problems with OpenGL

- Complex drivers
- Error management always active
- Shaders compiled by the drivers
- Cannot parallelize CPU OpenGL calls
- OpenGL still somewhat platform-dependent
- Different OpenGL versions for desktop vs mobile

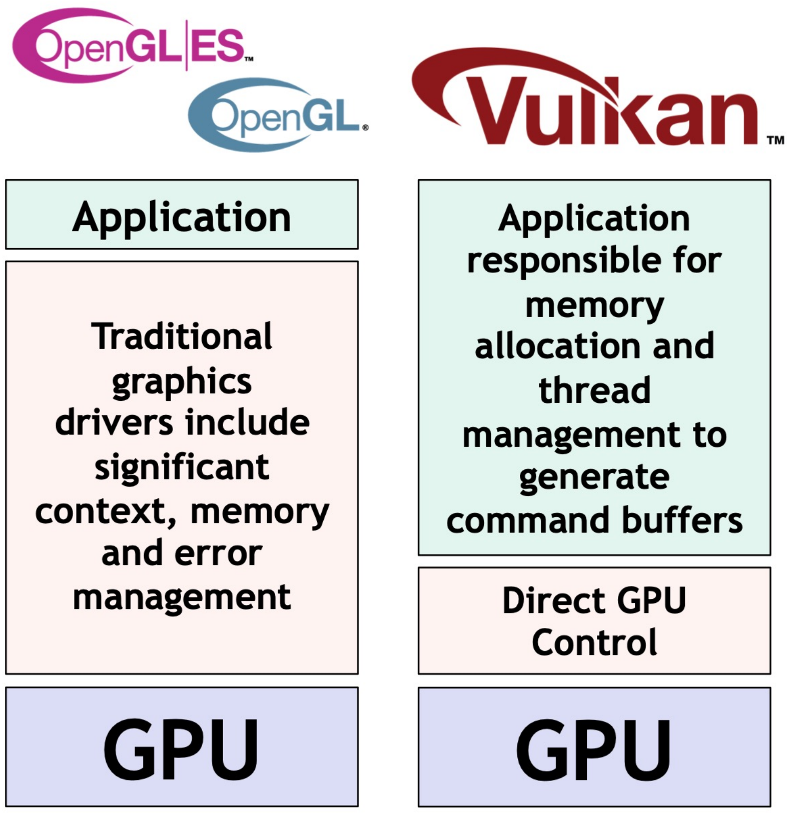
Vulkan Explicit GPU Control

Complex drivers lead to driver overhead and cross vendor unpredictability

Error management is always active

Driver processes full shading language source

Separate APIs for desktop and mobile markets


OpenGL ES
OpenGL
Application
Traditional graphics drivers include significant context, memory and error management
GPU
Vulkan
Application responsible for memory allocation and thread management to generate command buffers
Direct GPU Control
GPU

Simpler drivers for low-overhead efficiency and cross vendor portability

Layered architecture so validation and debug layers can be unloaded when not needed

Run-time only has to ingest SPIR-V intermediate language

Unified API for mobile, desktop, console and embedded platforms

Source: Khronos group

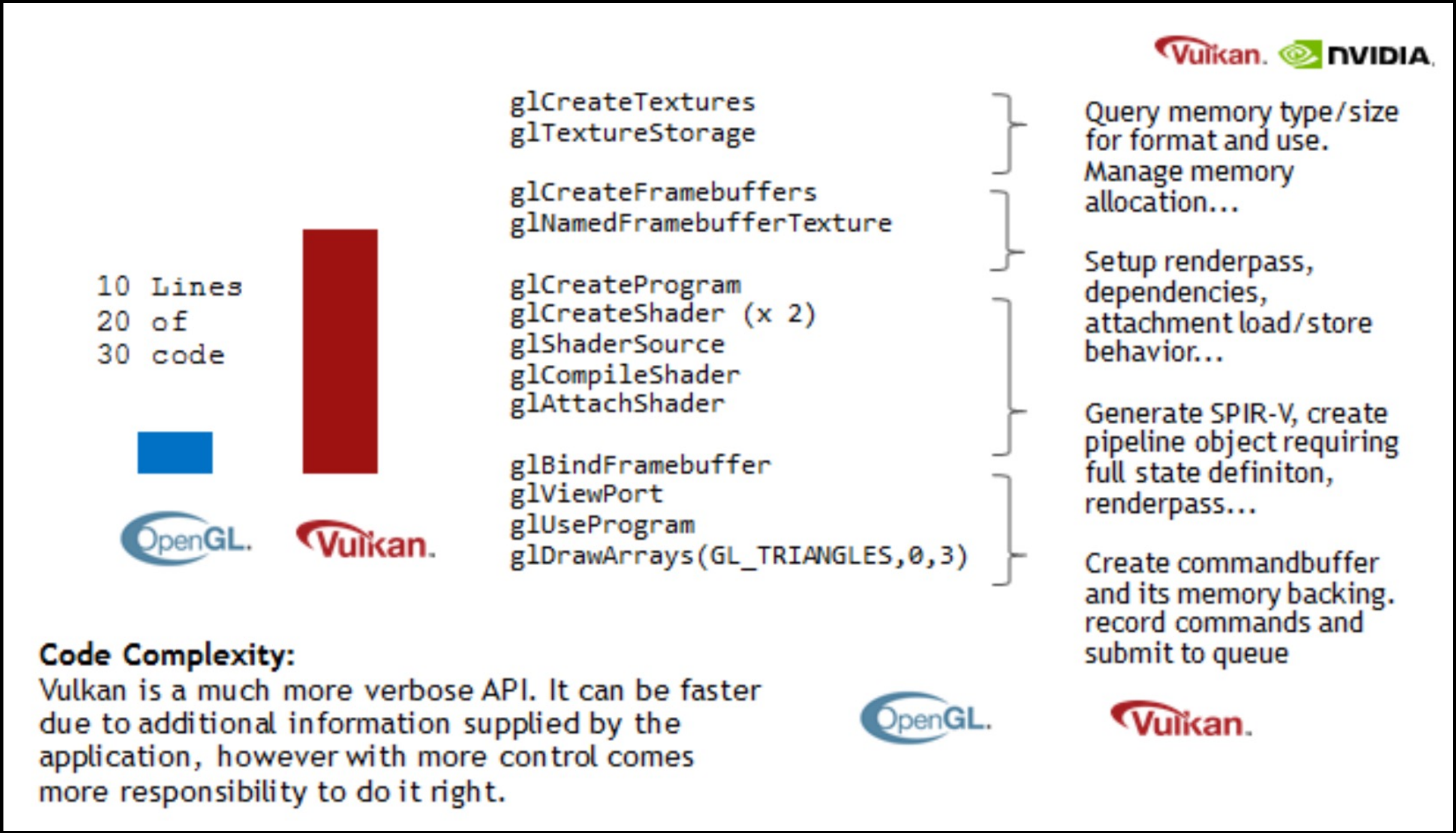
Vulkan Target Audience

- Vulkan is not for everyone
- For programmers enthusiastic about high-performance computer graphics
- If your focus is game development, you may stick with Direct3D or OpenGL.
- Major game engines use Vulkan without exposing it to you.

SPIR-V

- “Standard Portable Intermediate Representation”
- High-level intermediate language (exchanged in binary form)
- Used in Vulkan, and OpenCL
- Removes the need for the graphics driver to include a shading language compiler
- In Vulkan, one can use GLSL or HSLS => converted to SPIR-V

Code complexity: OpenGL vs Vulkan


Vulkan. NVIDIA.
10 Lines
20 of
30 code
OpenGL.
Vulkan.

```
glCreateTextures
glTextureStorage

glCreateFramebuffers
glNamedFramebufferTexture

glCreateProgram
glCreateShader (x 2)
glShaderSource
glCompileShader
glAttachShader

glBindFramebuffer
glViewPort
glUseProgram
glDrawArrays(GL_TRIANGLES,0,3)
```

Query memory type/size for format and use. Manage memory allocation...

Setup renderpass, dependencies, attachment load/store behavior...

Generate SPIR-V, create pipeline object requiring full state definiton, renderpass...

Create commandbuffer and its memory backing. record commands and submit to queue

OpenGL. Vulkan.

Code Complexity:
Vulkan is a much more verbose API. It can be faster due to additional information supplied by the application, however with more control comes more responsibility to do it right.

Source: Nvidia

Summary

- Vulkan is much more low-level than OpenGL
- Greater control over the GPU, at the cost of complex programming
- Vulkan is actively developed by the Khronos group
- OpenGL and Direct3D are not going away any time soon.