The Vulkan Sample Code Included with These Notes

Oregon State University

Mike Bailey

mjb@cs.oregonstate.edu

Oregon State University
Computer Graphics

SampleCode.pptx

mjb – December 30, 2022

Sample Program Output
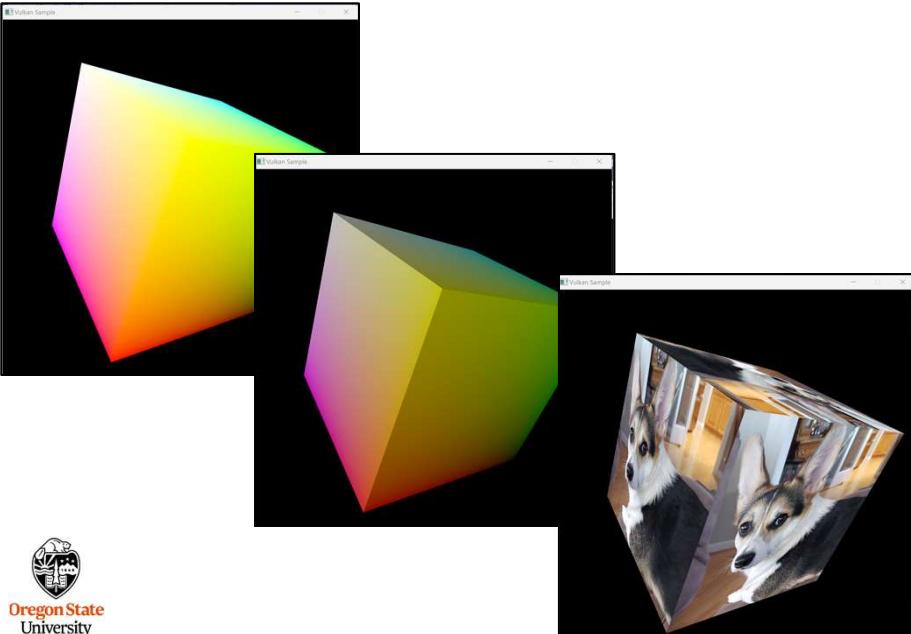
Oregon State University
Computer Graphics

mjb – December 30, 2022

Sample Program Keyboard Inputs

'l' (ell), 'L':	Toggle **l**ighting off and on
'm', 'M':	Toggle display **m**ode (textures vs. colors, for now)
'p', 'P':	**P**ause the animation
'q', 'Q':	**q**uit the program
Esc:	quit the program
'r', 'R':	Toggle **r**otation-animation and using the mouse
'i', 'I':	Toggle using a vertex buffer only vs. an **i**ndex buffer (in the index buffer version)
'1', …,'9','a',…'g'	Set the number of instances (in the instancing version)

Oregon State
University
Computer Graphics

mjb – December 30, 2022

Caveats on the Sample Code, I

1. I've written everything out in appalling longhand.

2. Everything is in one .cpp file (except the geometry data). It really should be broken up, but this way you can find everything easily.

3. At times, I could have hidden complexity, but I didn't. At all stages, I have tried to err on the side of showing you *everything*, so that nothing happens in a way that's kept a secret from you.

4. I've setup Vulkan structs every time they are used, even though, in many cases (most?), they could have been setup once and then re-used each time.

5. At times, I've setup things that didn't need to be setup just to show you what could go there.

Oregon State
University
Computer Graphics

mjb – December 30, 2022

Caveats on the Sample Code, II

5

6. There are great uses for C++ classes and methods here to hide some complexity, but I've not done that.

7. I've typedef'ed a couple things to make the Vulkan phraseology more consistent.

8. Even though it is not good software style, I have put persistent information in global variables, rather than a separate data structure

9. At times, I have copied lines from vulkan_core.h into the code as comments to show you what certain options could be.

10. I've divided functionality up into the pieces that make sense to me. Many other divisions are possible. Feel free to invent your own.

Oregon State
University
Computer Graphics

mjb – December 30, 2022

Main Program

6

```
int
main( int argc, char * argv[ ] )
{
    Width  = 1024;
    Height = 1025;

    errno_t err = fopen_s( &FpDebug, DEBUGFILE, "w" );
    if( err != 0 )
    {
        fprintf( stderr, "Cannot open debug print file '%s'\n", DEBUGFILE );
        FpDebug = stderr;
    }
    fprintf(FpDebug, "FpDebug: Width = %d ; Height = %d\n", Width, Height);

    Reset( );
    InitGraphics( );

    // loop until the user closes the window:

    while( glfwWindowShouldClose( MainWindow ) == 0 )
    {
        glfwPollEvents( );
        Time = glfwGetTime( );          // elapsed time, in double-precision seconds
        UpdateScene( );
        RenderScene( );
    }

    fprintf(FpDebug, "Closing the GLFW window\n");

    vkQueueWaitIdle( Queue );
    vkDeviceWaitIdle( LogicalDevice );
    DestroyAllVulkan( );
    glfwDestroyWindow( MainWindow );
    glfwTerminate( );
    return 0;
}
```

Oregon State
University
Computer Graphics

mjb – December 30, 2022

InitGraphics( ), I

```
void
InitGraphics( )
{
    HERE_I_AM( "InitGraphics" );

    VkResult result = VK_SUCCESS;

    Init01Instance( );

    InitGLFW( );

    Init02CreateDebugCallbacks( );

    Init03PhysicalDeviceAndGetQueueFamilyProperties( );

    Init04LogicalDeviceAndQueue( );

    Init05UniformBuffer( sizeof(Matrices),      &MyMatrixUniformBuffer );
    Fill05DataBuffer( MyMatrixUniformBuffer,    (void *) &Matrices );

    Init05UniformBuffer( sizeof(Light),     &MyLightUniformBuffer );
    Fill05DataBuffer( MyLightUniformBuffer, (void *) &Light );

    Init05MyVertexDataBuffer( sizeof(VertexData), &MyVertexDataBuffer );
    Fill05DataBuffer( MyVertexDataBuffer,         (void *) VertexData );

    Init06CommandPool( );
    Init06CommandBuffers( );
```

Oregon St
Universit
Computer G

mjb – December 30, 2022

InitGraphics( ), II

```
    Init07TextureSampler( &MyPuppyTexture.texSampler );
    Init07TextureBufferAndFillFromBmpFile("puppy.bmp", &MyPuppyTexture);

    Init08Swapchain( );

    Init09DepthStencilImage( );

    Init10RenderPasses( );

    Init11Framebuffers( );

    Init12SpirvShader( "sample-vert.spv", &ShaderModuleVertex );
    Init12SpirvShader( "sample-frag.spv", &ShaderModuleFragment );

    Init13DescriptorSetPool( );
    Init13DescriptorSetLayouts();
    Init13DescriptorSets( );

    Init14GraphicsVertexFragmentPipeline( ShaderModuleVertex, ShaderModuleFragment,
                        VK_PRIMITIVE_TOPOLOGY_TRIANGLE_LIST, &GraphicsPipeline );
}
```

Oregon State
University
Computer Graphics

mjb – December 30, 2022

A Colored Cube

9


```
static GLfloat CubeColors[ ][3] =
{
        { 0., 0., 0. },
        { 1., 0., 0. },
        { 0., 1., 0. },
        { 1., 1., 0. },
        { 0., 0., 1. },
        { 1., 0., 1. },
        { 0., 1., 1. },
        { 1., 1., 1. },
};
```

```
static GLuint CubeTriangleIndices[ ][3] =
{
        { 0, 2, 3 },
        { 0, 3, 1 },
        { 4, 5, 7 },
        { 4, 7, 6 },
        { 1, 3, 7 },
        { 1, 7, 5 },
        { 0, 4, 6 },
        { 0, 6, 2 },
        { 2, 6, 7 },
        { 2, 7, 3 },
        { 0, 1, 5 }
        { 0, 5, 4 }
};
```
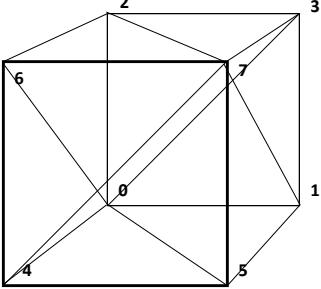
```
static GLfloat CubeVertices[ ][3] =
{
        { -1., -1., -1. },
        {  1., -1., -1. },
        { -1.,  1., -1. },
        {  1.,  1., -1. },
        { -1., -1.,  1. },
        {  1., -1.,  1. },
        { -1.,  1.,  1. },
        {  1.,  1.,  1. }
};
```

Oregon State
University
Computer Graphics

mjb – December 30, 2022

A Colored Cube
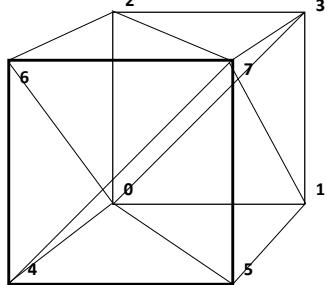
10

```
struct vertex
{
    glm::vec3    position;
    glm::vec3    normal;
    glm::vec3    color;
    glm::vec2    texCoord;
};

struct vertex VertexData[ ] =
{
    // triangle 0-2-3:
    // vertex #0:
    {
        { -1., -1., -1. },
        {  0.,  0., -1. },
        {  0.,  0.,  0. },
        {  1., 0. }
    },

    // vertex #2:
    {
        { -1.,  1., -1. },
        {  0.,  0., -1. },
        {  0.,  1.,  0. },
        {  1., 1. }
    },

    // vertex #3:
    {
        {  1.,  1., -1. },
        {  0.,  0., -1. },
        {  1.,  1.,  0. },
        {  0., 1. }
    },
```

Oregon State
University
Computer Graphics
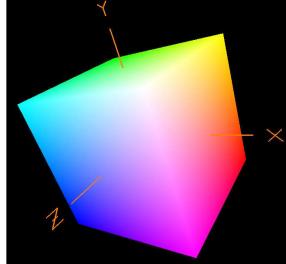
mjb – December 30, 2022

The Vertex Data is in a Separate File that is #include'd into sample.cpp   11

#include "SampleVertexData.cpp"

```
struct vertex
{
    glm::vec3      position;
    glm::vec3      normal;
    glm::vec3      color;
    glm::vec2      texCoord;
};

struct vertex VertexData[ ] =
{
    // triangle 0-2-3:
    // vertex #0:
    {
        { -1., -1., -1. },
        {  0.,  0., -1. },
        {  0.,  0.,  0. },
        {  1., 0. }
    },

    // vertex #2:
    {
        { -1.,  1., -1. },
        {  0.,  0., -1. },
        {  0.,  1.,  0. },
        {  1., 1. }
    },
    . . .
```

Oregon State
University
Computer Graphics

mjb – December 30, 2022

What if you don't need all of this information?   12

```
struct vertex
{
    glm::vec3      position;
    glm::vec3      normal;
    glm::vec3      color;
    glm::vec2      texCoord;
};
```

For example, what if you are not doing texturing in this application? Should you re-do this struct and leave the texCoord element out?

As best as I can tell, the only costs for retaining vertex attributes that you aren't going to use are some GPU memory space and possibly some inefficient uses of the cache, but not gross performance. So, I recommend keeping this struct intact, and, if you don't need texturing, simply don't use the texCoord values in your vertex or fragment shaders.

Oregon State
University
Computer Graphics

mjb – December 30, 2022

Vulkan Software Philosophy

Vulkan has lots of typedefs that define C/C++ structs and enums

Vulkan takes a non-C++ object-oriented approach in that those typedef'ed structs pass all the necessary information into a function. For example, where we might normally say using C++ class methods:

result = **LogicalDevice->**vkGetDeviceQueue (queueFamilyIndex, queueIndex, OUT &Queue);

Vulkan has chosen to do it like this:

result = vkGetDeviceQueue (**LogicalDevice**, queueFamilyIndex, queueIndex, OUT &Queue);

Oregon State
University
Computer Graphics

mjb – December 30, 2022

Vulkan Conventions

VkXxx is a typedef, probably a struct

vkYyy( ) is a function call

VK_ZZZ is a constant

My Conventions

"Init" in a function call name means that something is being setup that only needs to be setup once

The number after "Init" gives you the ordering

In the source code, after main( ) comes InitGraphics( ), then all of the InitxxYYY( ) functions in numerical order. After that comes the helper functions

"Find" in a function call name means that something is being looked for

"Fill" in a function call name means that some data is being supplied to Vulkan

"IN" and "OUT" ahead of function call arguments are just there to let you know how an argument is going to be used by the function. Otherwise, IN and OUT have no significance. They are actually #define'd to nothing.

Computer Graphics

mjb – December 30, 2022

**Querying the Number of Something and
Allocating Enough Structures to Hold Them All**

15

```
uint32_t  count;
result = vkEnumeratePhysicalDevices( Instance, OUT &count, OUT (VkPhysicalDevice *)nullptr );

VkPhysicalDevice * physicalDevices = new VkPhysicalDevice[ count ];
result = vkEnumeratePhysicalDevices( Instance, OUT &count, OUT &physicalDevices[0]  );
```

This way of querying information is a recurring OpenCL and Vulkan pattern (get used to it):

	How many total there are	Where to put them
result = vkEnumeratePhysicalDevices(Instance,	&count,	nullptr);
result = vkEnumeratePhysicalDevices(Instance,	&count,	&physicalDevices[0]);

Oregon State
University
Computer Graphics

mjb – December 30, 2022

Your Sample2019.zip File Contains This

16

Linux shader compiler — glslangValidator

Windows shader compiler — glslangValidator.exe

Double-click here to launch Visual Studio 2019 with this solution — Sample.sln

The "19" refers to the version of Visual Studio, not the year of development.

Oregon State
University
Computer Graphics

mjb – December 30, 2022

Reporting Error Results, I

17

```
struct errorcode
{
        VkResult       resultCode;
        std::stringmeaning;
}
ErrorCodes[ ] =
{
        { VK_NOT_READY,                              "Not Ready"              },
        { VK_TIMEOUT,                                "Timeout"                },
        { VK_EVENT_SET,                              "Event Set"              },
        { VK_EVENT_RESET,                            "Event Reset"            },
        { VK_INCOMPLETE,                             "Incomplete"             },
        { VK_ERROR_OUT_OF_HOST_MEMORY,               "Out of Host Memory"     },
        { VK_ERROR_OUT_OF_DEVICE_MEMORY,             "Out of Device Memory"   },
        { VK_ERROR_INITIALIZATION_FAILED,            "Initialization Failed"  },
        { VK_ERROR_DEVICE_LOST,                      "Device Lost"            },
        { VK_ERROR_MEMORY_MAP_FAILED,                "Memory Map Failed"      },
        { VK_ERROR_LAYER_NOT_PRESENT,                "Layer Not Present"      },
        { VK_ERROR_EXTENSION_NOT_PRESENT,            "Extension Not Present"  },
        { VK_ERROR_FEATURE_NOT_PRESENT,              "Feature Not Present"    },
        { VK_ERROR_INCOMPATIBLE_DRIVER,              "Incompatible Driver"    },
        { VK_ERROR_TOO_MANY_OBJECTS,                 "Too Many Objects"       },
        { VK_ERROR_FORMAT_NOT_SUPPORTED,             "Format Not Supported"   },
        { VK_ERROR_FRAGMENTED_POOL,                  "Fragmented Pool"        },
        { VK_ERROR_SURFACE_LOST_KHR,                 "Surface Lost"           },
        { VK_ERROR_NATIVE_WINDOW_IN_USE_KHR,         "Native Window in Use"   },
        { VK_SUBOPTIMAL_KHR,                         "Suboptimal"             },
        { VK_ERROR_OUT_OF_DATE_KHR,                  "Error Out of Date"      },
        { VK_ERROR_INCOMPATIBLE_DISPLAY_KHR,         "Incompatible Display"   },
        { VK_ERROR_VALIDATION_FAILED_EXT,            "Valuidation Failed"     },
        { VK_ERROR_INVALID_SHADER_NV,                "Invalid Shader"         },
        { VK_ERROR_OUT_OF_POOL_MEMORY_KHR,           "Out of Pool Memory"     },
        { VK_ERROR_INVALID_EXTERNAL_HANDLE,          "Invalid External Handle" },
};
```

Oregon State University
Computer Graphics

mjb – December 30, 2022

Reporting Error Results, II

18

```
void
PrintVkError( VkResult result, std::string prefix )
{
        if (Verbose  &&  result == VK_SUCCESS)
        {
                fprintf(FpDebug, "%s: %s\n", prefix.c_str(), "Successful");
                fflush(FpDebug);
                return;
        }

        const int numErrorCodes = sizeof( ErrorCodes ) / sizeof( struct errorcode );
        std::string meaning = "";
        for( int i = 0; i < numErrorCodes; i++ )
        {
                if( result == ErrorCodes[i].resultCode )
                {
                        meaning = ErrorCodes[i].meaning;
                        break;
                }
        }

        fprintf( FpDebug, "\n%s: %s\n", prefix.c_str(), meaning.c_str() );
        fflush(FpDebug);
}
```

Oregon State University
Computer Graphics

mjb – December 30, 2022

Extras in the Code

```
#define REPORT(s)          { PrintVkError( result, s );  fflush(FpDebug); }


#define HERE_I_AM(s)       if( Verbose )  { fprintf( FpDebug, "***** %s *****\n", s );  fflush(FpDebug); }


bool          Paused;


bool          Verbose;




#define DEBUGFILE          "VulkanDebug.txt"

errno_t err = fopen_s( &FpDebug, DEBUGFILE, "w" );


const int32_t  OFFSET_ZERO =  0;
```

Oregon State
University
Computer Graphics