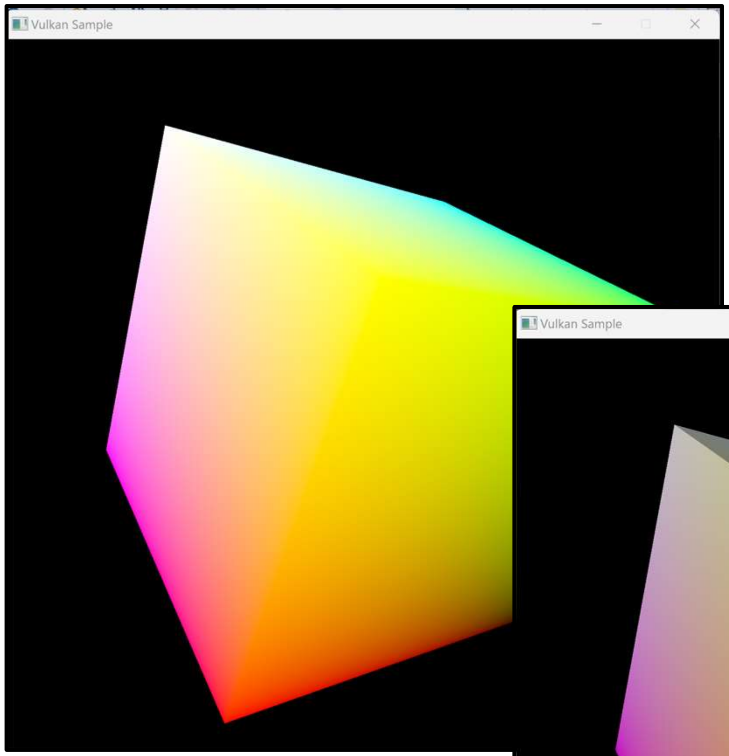
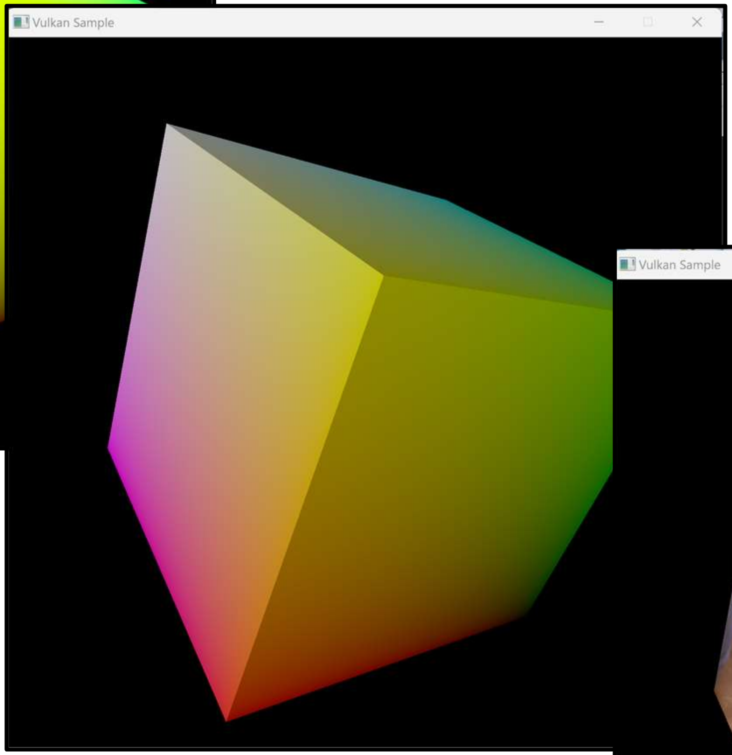
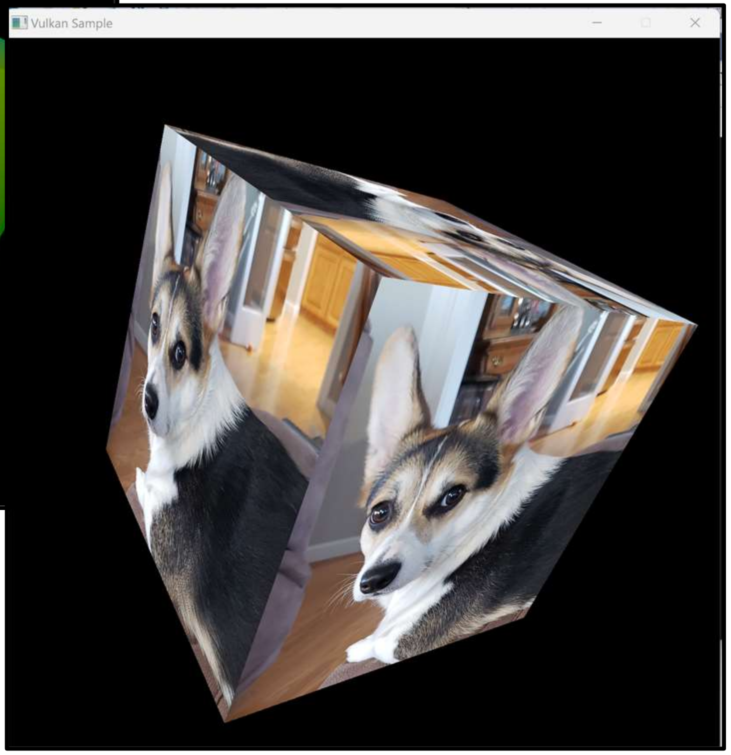
The Vulkan Sample Code Included with These Notes

Oregon State University
Mike Bailey
mjb@cs.oregonstate.edu

Sample Program Output







Oregon State University Computer Graphics

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)

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.

Caveats on the Sample Code, II

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.

Main Program

```
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;
}
```

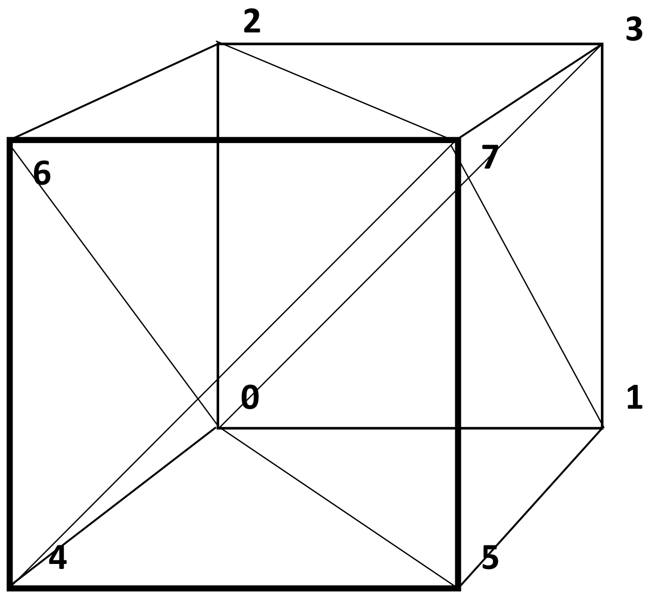
InitGraphics( ), I

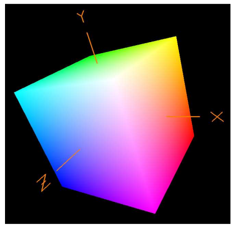
```
void
InitGraphics( )
{
        HERE_I_AM( "InitGraphics" );

        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( );
```

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  );
}
```

A Colored Cube





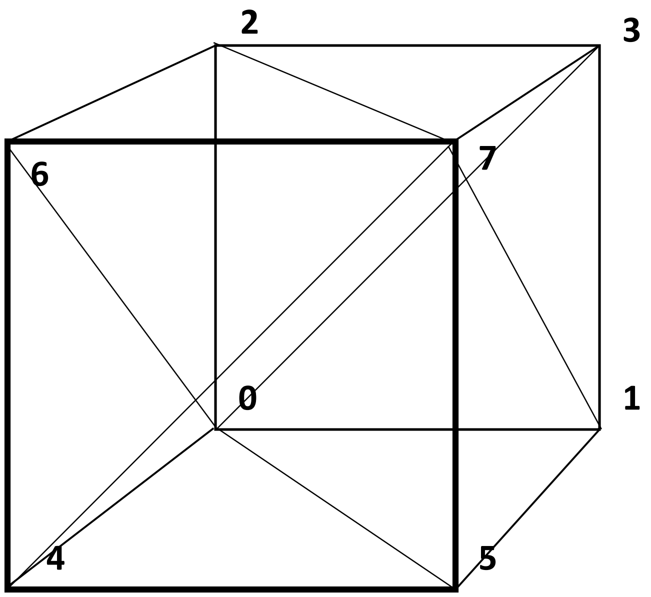
```
static GLfloat CubeColors[ ][3] =
{
		{ 0., 0., 0. },
		{ 1., 0., 0. },
		{ 0., 1., 0. },
		{ 1., 1., 0. },
		{ 0., 0., 1. },
		{ 1., 0., 1. },
		{ 0., 1., 1. },
		{ 1., 1., 1. },
};
```

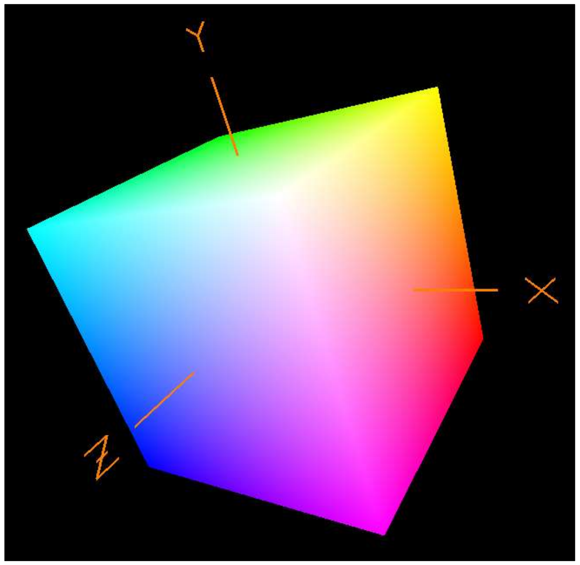
```
static GLfloat CubeVertices[ ][3] =
{
		{ -1., -1., -1. },
		{  1., -1., -1. },
		{ -1.,  1., -1. },
		{  1.,  1., -1. },
		{ -1., -1.,  1. },
		{  1., -1.,  1. },
		{ -1.,  1.,  1. },
		{  1.,  1.,  1. }
};
```

```
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 }
};
```

A Colored Cube

```
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. }
        },
```





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

```
#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. }
	},
. . .
```

What if you don’t need all of this information?

```
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.

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 );
```

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.

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

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

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

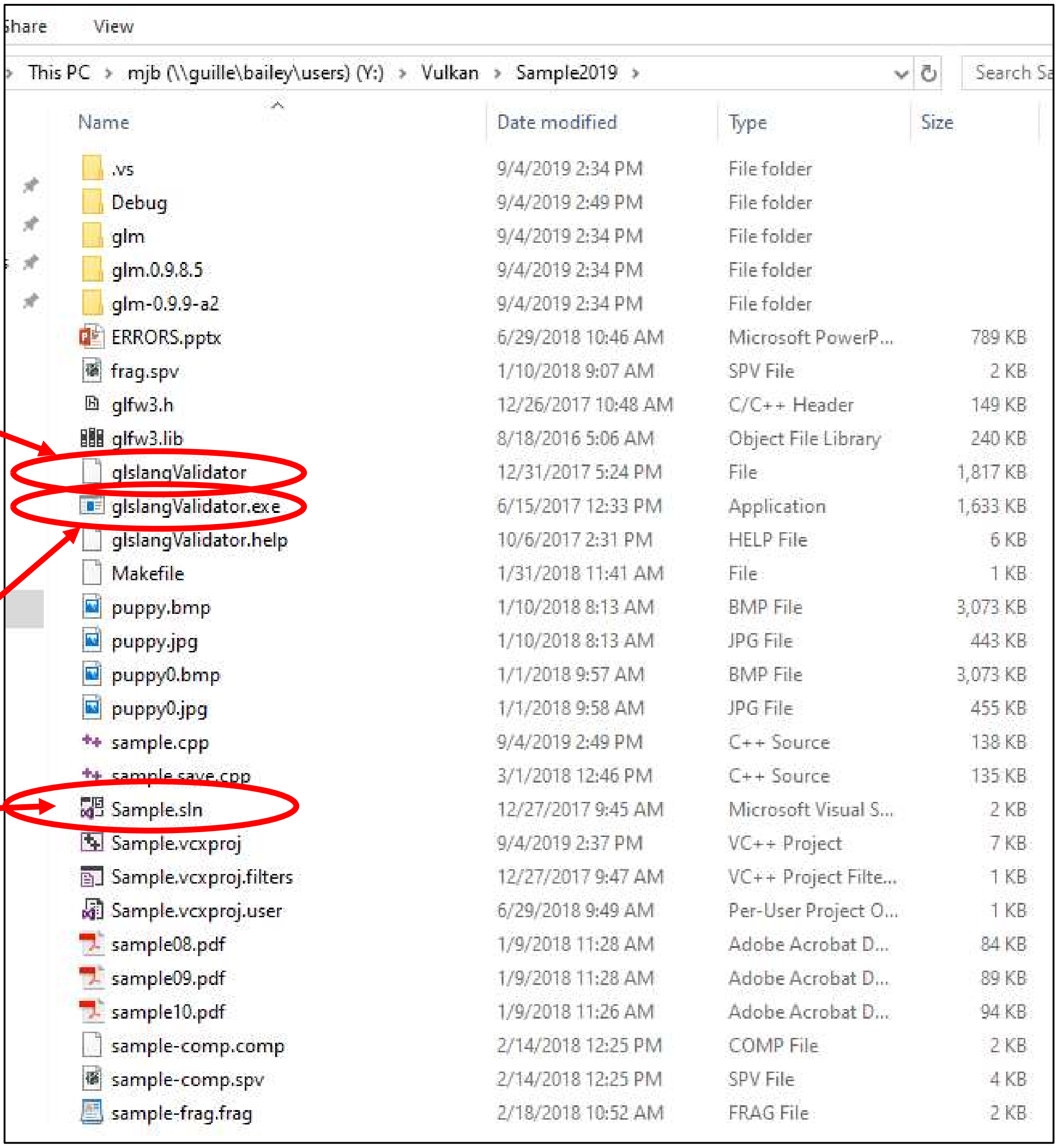
This way of querying information is a recurring OpenCL and Vulkan pattern (get used to it):

How many total there are

Where to put them

```
result = vkEnumeratePhysicalDevices( Instance,    &count,              nullptr   );


result = vkEnumeratePhysicalDevices( Instance,    &count,         &physicalDevices[0] );
```

Your Sample2019.zip File Contains This



Share View

This PC › mjb (\\guille\bailey\users) (Y:) › Vulkan › Sample2019 ›

Search Sa

Name	Date modified	Type	Size
.vs	9/4/2019 2:34 PM	File folder	
Debug	9/4/2019 2:49 PM	File folder	
glm	9/4/2019 2:34 PM	File folder	
glm.0.9.8.5	9/4/2019 2:34 PM	File folder	
glm-0.9.9-a2	9/4/2019 2:34 PM	File folder	
ERRORS.pptx	6/29/2018 10:46 AM	Microsoft PowerP...	789 KB
frag.spv	1/10/2018 9:07 AM	SPV File	2 KB
glfw3.h	12/26/2017 10:48 AM	C/C++ Header	149 KB
glfw3.lib	8/18/2016 5:06 AM	Object File Library	240 KB
glslangValidator	12/31/2017 5:24 PM	File	1,817 KB
glslangValidator.exe	6/15/2017 12:33 PM	Application	1,633 KB
glslangValidator.help	10/6/2017 2:31 PM	HELP File	6 KB
Makefile	1/31/2018 11:41 AM	File	1 KB
puppy.bmp	1/10/2018 8:13 AM	BMP File	3,073 KB
puppy.jpg	1/10/2018 8:13 AM	JPG File	443 KB
puppy0.bmp	1/1/2018 9:57 AM	BMP File	3,073 KB
puppy0.jpg	1/1/2018 9:58 AM	JPG File	455 KB
sample.cpp	9/4/2019 2:49 PM	C++ Source	138 KB
sample.save.cpp	3/1/2018 12:46 PM	C++ Source	135 KB
Sample.sln	12/27/2017 9:45 AM	Microsoft Visual S...	2 KB
Sample.vcxproj	9/4/2019 2:37 PM	VC++ Project	7 KB
Sample.vcxproj.filters	12/27/2017 9:47 AM	VC++ Project Filte...	1 KB
Sample.vcxproj.user	6/29/2018 9:49 AM	Per-User Project O...	1 KB
sample08.pdf	1/9/2018 11:28 AM	Adobe Acrobat D...	84 KB
sample09.pdf	1/9/2018 11:28 AM	Adobe Acrobat D...	89 KB
sample10.pdf	1/9/2018 11:26 AM	Adobe Acrobat D...	94 KB
sample-comp.comp	2/14/2018 12:25 PM	COMP File	2 KB
sample-comp.spv	2/14/2018 12:25 PM	SPV File	4 KB
sample-frag.frag	2/18/2018 10:52 AM	FRAG File	2 KB

Linux shader compiler

Windows shader compiler

Double-click here to launch Visual Studio 2019 with this solution

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

Reporting Error Results, I

```
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“ },
};
```

Reporting Error Results, II

```
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);
}
```

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;
```