Learn Vulkan

1 - Introduction

Vulkan has a reputation for being difficult to learn for beginners. The main reason is that it hides less to the programmer than the old API (OpenGL), so you just end up studying low-level details even to draw a single triangle. However, learning Vulkan also has some advantages as it designed to make full use of multi-threading, and to provide finer control over resource management and CPU/GPU synchronization.

Thanks to better support for multi-threading, the CPU workload is more evenly distributed across all cores, which can lead to increased GPU workload, enhancing the overall performance of your graphics applications. And if you understand the use and re-use of resources, then you can minimize data transferts (uploading and copying of data) and reduce the total memory footprint of the application by recycling the same physical memory.

> Observe that if the bottleneck of your application is not CPU-side (that is, if the GPU never waits for the CPU to send work to execute) then Vulkan is not necessarily going to give you a performance boost. Indeed, the functionality exposed by Vulkan is almost identical to that found in OpenGL, and if your application is limited by GPU rendering performance, then it is unlikely that Vulkan will give you better results. However, even if your application isn't CPU-limited, using Vulkan still allows you to free up CPU time, improving the overall system energy efficiency.

Currently, you have three ways to learn Vulkan from scratch: books, online tutorials, and official documentation, each with its own strengths and weaknesses, but with a common thread that binds them together. Indeed, the intended audience of these resources are graphics programmers (that is, they assume you have basic knowledge of computer graphics), so the focus is on the API, leaving little room for the math and theory behind rendering techniques.

Actually, even if you are not a graphics programmer, nothing prevents you from learning Vulkan from resources that are currently available, provided that you are motivated enough to jump from a math textbook to another, and to read dozens of documentation pages and tutorials to collect all the information you need about every single subject.

With this tutorial series I'll try to fill the gap, providing a resource which covers both theory (maths, computer graphics, rendering techniques) and practice (Vulkan API and code samples) in depth. And the only possibility of doing that effectively is to prevent the reader from getting bored, intimidated or discouraged by the amount of information they need to learn and master in order to write even the simplest graphics application. Usually, when people start studying computer graphics, they want to see something on their screen as a reward. For this reason, in every tutorial I will review a different sample to only explain the theory needed to implement it. That way, the reader feels like they are studying to understand the implementation of that specific sample. This allows a gradual and progressive acquisition of new information on which to build knowledge like a puzzle: whenever you read a tutorial, you add another piece.

I'm not claiming this will be the ultimate resource to learn the Vulkan API. By the way, the official documentation (see [1] and [2] in the reference list at the end of this tutorial) will always be the most important and fundamental resource to refer to during your studies. Here, I will simply try to provide as many details as possible to minimize the number of resources you need to look at.

As a reference for writing the samples of this tutorial series, I used the source code already available and maintained by The Khronos Group and Sascha Willems in their repositories. In particular, I will use a minimal framework, trying to follow the Orthodox C++ guidelines as much as possible to increase readability. This will facilitate understanding of the source code.

Khronos Group on GitHub

Sascha Willems on GitHub

In conclusion, if you are looking for a comprehensive, updated resource to learn how to program with the Vulkan API from scratch, then give this tutorial series a chance.

2 - Prerequisites

2.1 - Background knowledge

Of course, I can’t explain everything, so having basic math skills is essential to completely understand the content of the tutorials and the techniques used in the related samples. Fortunately, knowledge of college-level math, including algebra, trigonometry, and calculus, covers about 90% of the math used throughout this tutorial series. For the remaining 10%, I will provide dedicated math appendices.

If your math skills are a bit rusty, don't worry! You can consider [3] as an brilliant math review that starts from the basics and is accessible to everyone. Additionally, I recommend [4] (for self-study) or [5] (for formal education) as excellent references for more advanced topics. Additional textbook recommendations will be provided at the appropriate time.

If you need an introductory textbook on C++ you can take a look at [6] and [7], then use [8] as a reference, and [9] to really understand the low-level details.

2.1 - Hardware

Obviously, if you want to run the samples examined in this tutorial series, you need a system with a Vulkan-capable graphics card. Actually, most of the GPUs produced in the last decade should be compatible. However, you can directly verify it by running vkcube, a simple Vulkan test program you can find in the Vulkan-Tools repo, or in the Vulkan SDK (more on this shortly).

Also, make sure to update the drivers for your graphics card to the latest version.

2.2 - Software

2.2.1 - Supported platforms

Windows, Linux and Android provide native support for Vulkan. This means that an OS is able to execute a Vulkan driver (released by a GPU hardware vendor), which is in charge of getting the Vulkan API calls mapped to the hardware. MacOS and iOS don't provide native support for Vulkan, but you can still execute Vulkan applications on those platforms somehow. However, I won't delve into much detail here since the source code of this tutorial series is written to only run on Windows and Linux. This is because I want to avoid code bloat from conditional compilation to support several platforms, and I don't event want to use a multi-platform library just to create a simple window (I won't hide anything to the reader).

> This tutorial series is designed to help you learn how to write graphics applications using the Vulkan API from the ground up. However, please note that it is not intended to be a guide for cross-platform development.

2.2.2 - Headers, libraries and tools

The repository that hosts the code of this tutorial series also includes the header files, import library and tools needed to write and build Vulkan applications. Therefore, you don't need anything else except cloning the repo locally on your PC using the following command:

```
git clone https://github.com/PAMinerva/LearnVulkan
```

However, you still need the Vulkan Loader and other shared libraries to execute Vulkan applications. On Windows, the Vulkan Loader is delivered with the hardware vendor driver update packages. On Linux, it should be the same if proprietary drivers are installed. Otherwise, you must explicitly install it. For this purpose, you can download the Vulkan SDK from the following link.

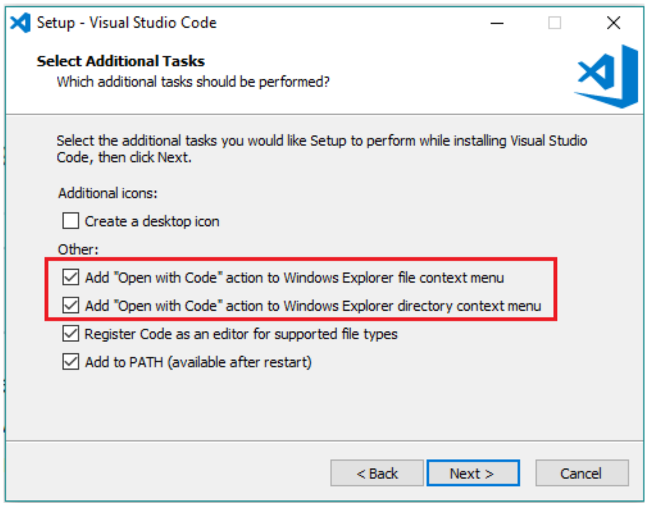
Vulkan SDK

The Vulkan SDK also includes the header files, the layers (as shader libraries), and others interesting tools to validate and compile the shader code. I recommend installing the Vulkan SDK on both Windows and Linux, since it includes the binaries already compiled (included the vkcube program test). Furthermore, the Vulkan SDK will automatically set up your system for development use. This involve creating some registry keys (on Windows), folders and files (both on Windows and Linux) that are often required during development to enable validation and debugging functionality.

Alternatively, you can refer to the Khronos Group repositories to get updated header files, and build the shader libraries and other tools directly from the source code.

2.2.3 - IDE

I will use Visual Studio Code to develop the samples presented in this tutorial series, so you are encouraged to install and use it as well, making sure to check the following additional tasks during installation setup.



Setup - Visual Studio Code

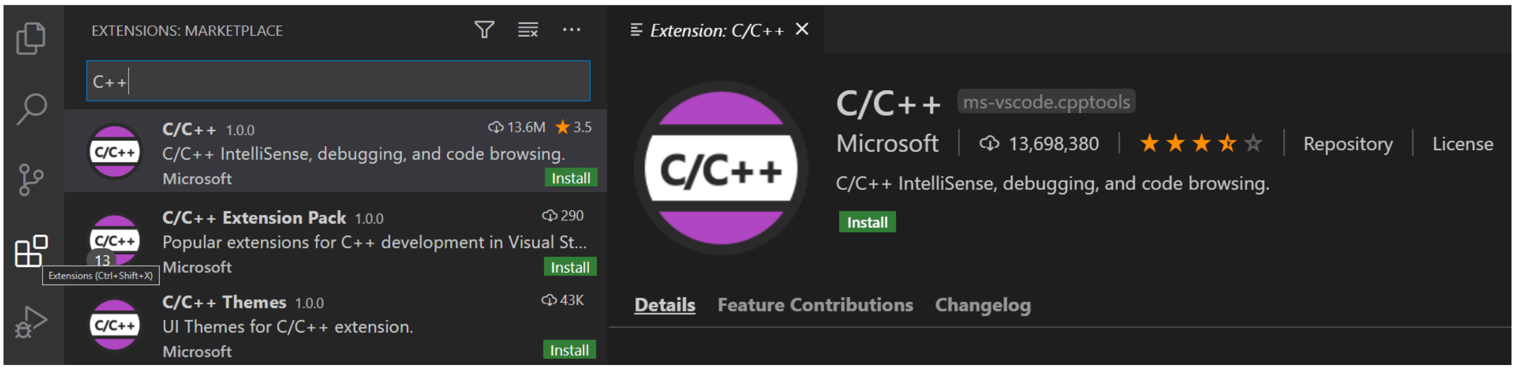
Select Additional Tasks
Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing Visual Studio Code, then click Next.

Additional icons:
- [ ] Create a desktop icon

Other:
- [x] Add "Open with Code" action to Windows Explorer file context menu
- [x] Add "Open with Code" action to Windows Explorer directory context menu
- [x] Register Code as an editor for supported file types
- [x] Add to PATH (available after restart)

You need to install the C/C++ extension for VS Code. You can install it by searching for 'c++' in the Extensions view.



EXTENSIONS: MARKETPLACE

C++

C/C++ 1.0.0 — 13.6M ★3.5
C/C++ IntelliSense, debugging, and code browsing.
Microsoft — Install

C/C++ Extension Pack 1.0.0 — 290
Popular extensions for C++ development in Visual St...
Microsoft — Install

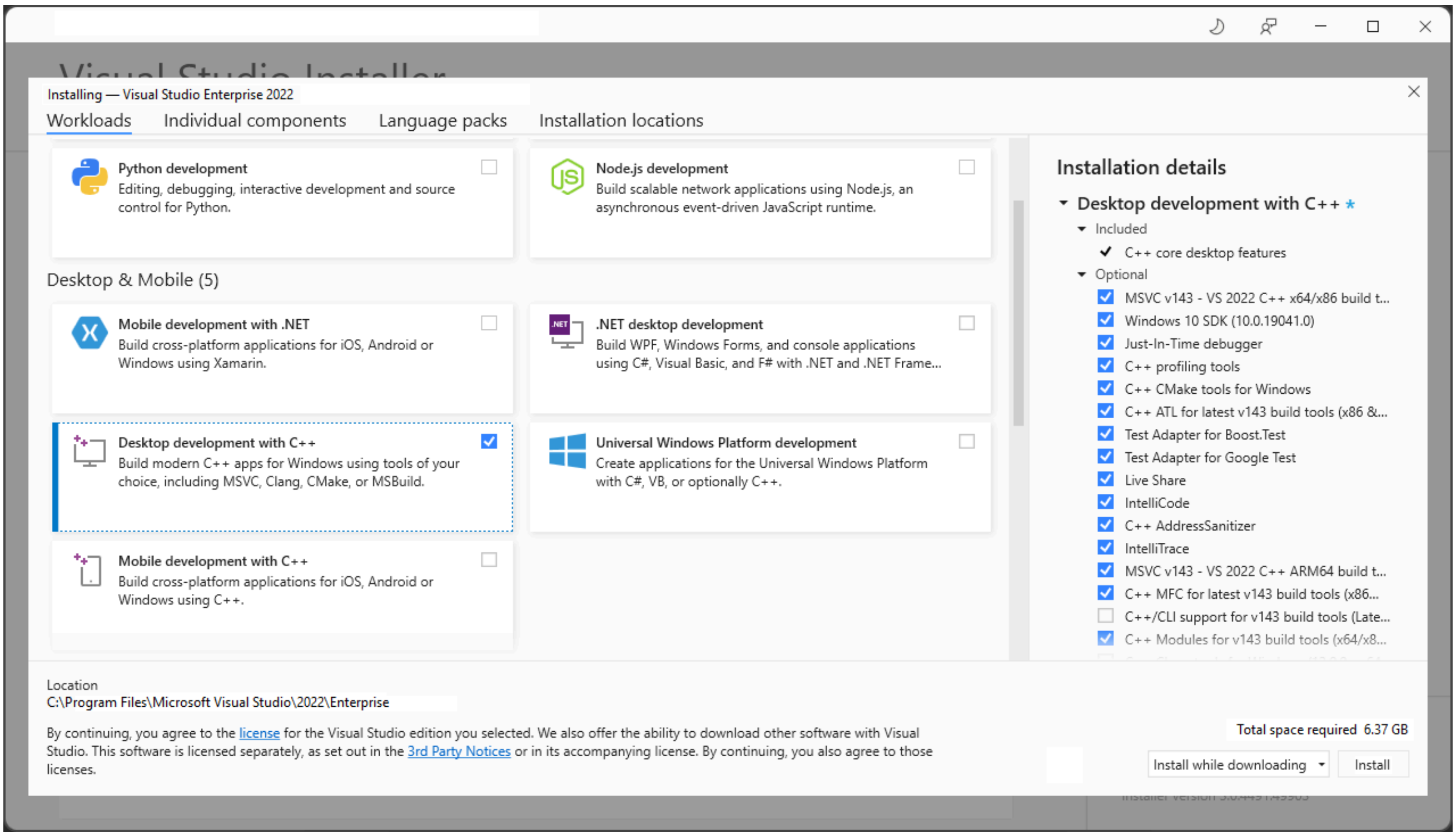
C/C++ Themes 1.0.0 — 43K
UI Themes for C/C++ extension.
Microsoft — Install

Extension: C/C++

C/C++ ms-vscode.cpptools
Microsoft | 13,698,380 | Repository | License
C/C++ IntelliSense, debugging, and code browsing.
Install

On Windows you also need to install the Microsoft Visual C++ (MSVC) compiler toolset.

If you have a recent version of Visual Studio, open the Visual Studio Installer from the Windows Start menu and verify that Desktop development with C++ is checked. If it's not installed, then check the box and select the Modify button in the installer.


Installing — Visual Studio Enterprise 2022

Workloads | Individual components | Language packs | Installation locations

Python development ☐
Editing, debugging, interactive development and source control for Python.

Node.js development ☐
Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Desktop & Mobile (5)

Mobile development with .NET ☐
Build cross-platform applications for iOS, Android or Windows using Xamarin.

.NET desktop development ☐
Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET and .NET Frame...

Desktop development with C++ ✓
Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.

Universal Windows Platform development ☐
Create applications for the Universal Windows Platform with C#, VB, or optionally C++.

Mobile development with C++ ☐
Build cross-platform applications for iOS, Android or Windows using C++.

Installation details

- Desktop development with C++ *
  - Included
    - ✓ C++ core desktop features
  - Optional
    - ✓ MSVC v143 - VS 2022 C++ x64/x86 build t...
    - ✓ Windows 10 SDK (10.0.19041.0)
    - ✓ Just-In-Time debugger
    - ✓ C++ profiling tools
    - ✓ C++ CMake tools for Windows
    - ✓ C++ ATL for latest v143 build tools (x86 &...
    - ✓ Test Adapter for Boost.Test
    - ✓ Test Adapter for Google Test
    - ✓ Live Share
    - ✓ IntelliCode
    - ✓ C++ AddressSanitizer
    - ✓ IntelliTrace
    - ✓ MSVC v143 - VS 2022 C++ ARM64 build t...
    - ✓ C++ MFC for latest v143 build tools (x86...
    - ☐ C++/CLI support for v143 build tools (Late...
    - ✓ C++ Modules for v143 build tools (x64/x8...

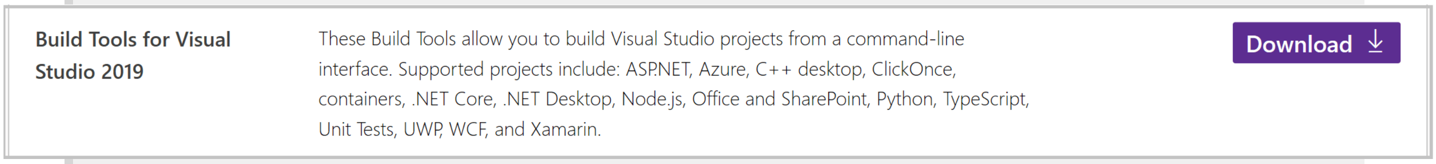
Location
C:\Program Files\Microsoft Visual Studio\2022\Enterprise

By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

Total space required 6.37 GB

Install while downloading | Install

> You can also install the Desktop development with C++ workload without a full Visual Studio IDE installation. From the Visual Studio Downloads page, scroll down until you see Tools for Visual Studio 2022 under the All Downloads section and select the download for Build Tools for Visual Studio 2022. This will launch the Visual Studio Installer.


Build Tools for Visual Studio 2019

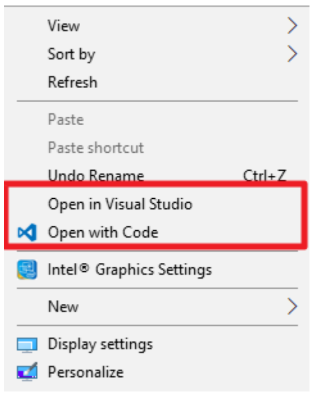
These Build Tools allow you to build Visual Studio projects from a command-line interface. Supported projects include: ASP.NET, Azure, C++ desktop, ClickOnce, containers, .NET Core, .NET Desktop, Node.js, Office and SharePoint, Python, TypeScript, Unit Tests, UWP, WCF, and Xamarin.

On linux you need to check if GCC and GDB are already installed. To verify whether they are, open a Terminal window and enter the following commands:

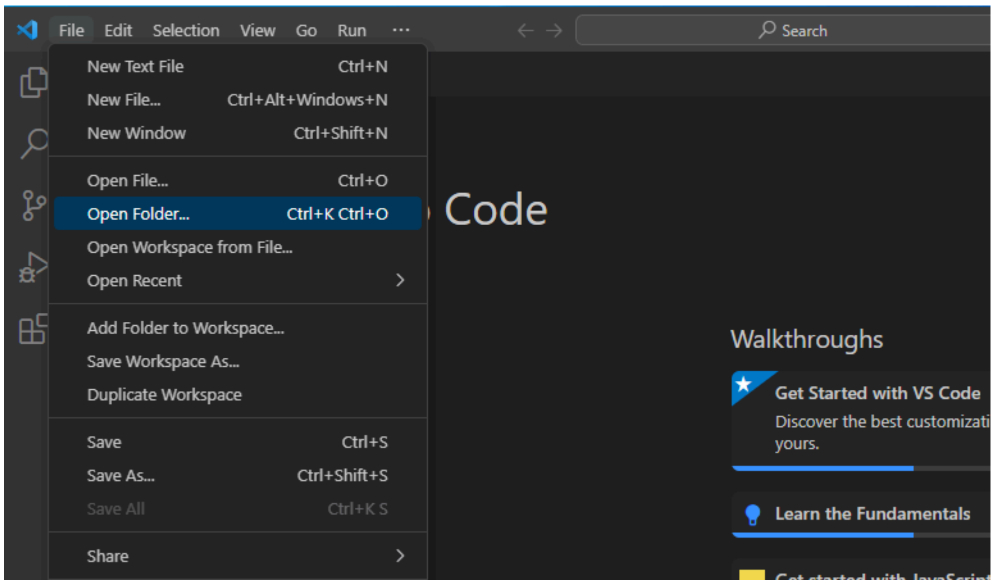
```
gcc -v
gdb -help
```

If GCC and\or GDB aren't installed, refer to the appropriate documentation for instructions on installing them for your specific Linux distribution. Also, make sure the libx11 development package is installed as well.

To open a sample, right-click on the related folder and select "Open with Code" from the context menu.



If the option to "Open with Code" is not present in the context menu, you can alternatively navigate to File and select "Open Folder...".

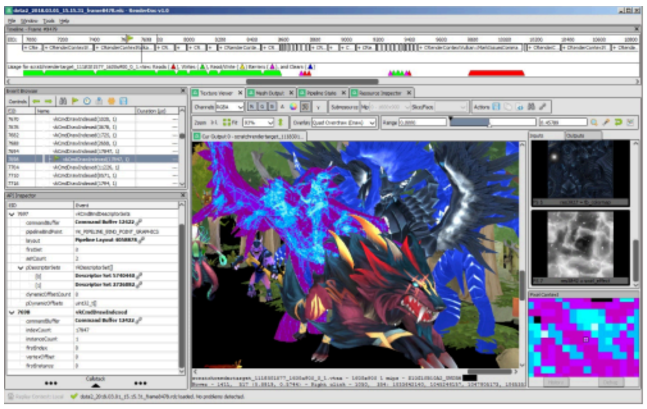


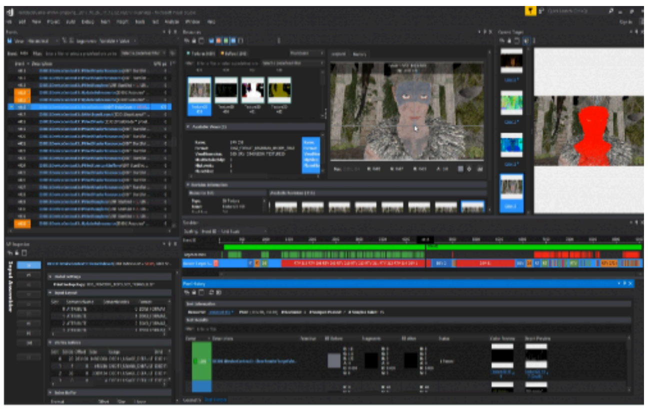
2.3 - Graphics Debugger

In addition to an application debugger, such as GDB or the one included in the MSVC toolset, it's also necessary to have a graphics debugger that allows you to capture a frame of your Vulkan application for inspecting every individual event involved in generating the frame. For this purpose, I recommend installing RenderDoc and/or NVIDIA Nsight Graphics, which can be found at the following links.

RenderDoc

NVIDIA Nsight Graphics





References

[1] Vulkan API Specifications

[2] Khronos APIs Specifications

[3] Engineering Mathematics (Stroud, Booth)

[4] Advanced Engineering Mathematics (Stroud, Booth)

[5] Advanced Engineering Mathematics (Kreyszig)

[6] Beginning C++ Through Game Programming (Dawson)

[7] A Tour of C++ (Stroustrup)

[8] The C++ Programming Language (Stroustrup)

[9] Write Great Code - Vol. 1 and 2 (Randall Hyde)

If you found the content of this tutorial somewhat useful or interesting, please consider supporting this project by clicking on the Sponsor button. Whether a small tip, a one time donation, or a recurring payment, it's all welcome! Thank you!

Sponsor