ADASS GPU Tutorial 2024
For Windows Users

Keith Shortridge, K&V, 7 October 2024

Introduction

This tutorial is aimed at ADASS attendees who may have sat through numerous talks about how GPUs make everything faster, wondered about making use of their GPUs to speed up compute tasks, and then somehow never found the time to try it. The aim is to give people who have no experience with GPU programming a kick-start towards trying it for themselves on their own laptops. The tutorial will be based around a small set of example C++ command line programs that perform calculations on 2D data, all of which run on MacOS, Linux or Windows. Attendees will be able to build, run, modify, and experiment with these programs, seeing how the GPU performance compares with the CPU. These instructions describe how to download and build these example programs.

You need to do the following:

- Download the source code for the programs.
- If you don't have them on your machine already, install the libraries they use. This should be straightforward.
- Build the programs – this should amount to no more than typing 'nmake' three times in a terminal window.
- Come to the tutorial and bring your laptop.

The tutorial was originally designed for MacOS and Linux, but it became apparent that the programs could also be built on Windows, both using WSL (the Windows Subsytem for Linux), or natively using the Windows Command Prompt from a terminal window. Using WSL is essentially using Linux, and is covered in the original version of these instructions, which cover MacOS and Linux. This document is purely for anyone interested in building the example programs in a native Windows environment.

These instructions have been developed using Windows 11, running Virtual Studio 2022, and assume you will be building 64-bit versions of the programs.

Downloading the source code.

The source code is available on GitHub at:

https://github.com/KnaveAndVarlet/ADASS2024_GPU

Look at the right-hand column under 'Releases' and click on the latest release. Download the .zip file and extract the contents.

This gives you a directory called ADASS2024_GPU-<version>. In it there are two sub-directories, Metal and Vulkan, containing the source files for the Metal and Vulkan version of the example programs. Metal is Apple's GPU programming API, and Vulkan is a cross-platform GPU API from the Khronos group. Metal only runs on Apple devices, while Vulkan should be supported on almost any modern GPU. Windows users will be using Vulkan.

Metal and Vulkan have three sub-directories, Adder, Median, and Mandel, containing the three example programs. Each of these individual program sub-directories is self-contained., with its own copies of any common source files.

Installing the necessary libraries and tools.

Detailed instructions for those that need them are in the appendices that follow, but this section provides an overview. You don't need much.

Programming tools

You will need:

- Microsoft's VisualStudio.
- A source code editor that you're comfortable with. You can use Visual Studio itself, or any other editor you prefer. You could even use Notepad.

If you don't have Visual Studio installed, you can download the free Community version from https://visualstudio.microsoft.com/downloads/ You need to include the Desktop development with C++ option.

It's just the Visual Studio developer command prompt that you'll be using for the example programs, not the full IDE. The example programs were designed to be run as command line programs, and this keeps things as compatible as possible between all the different systems.

GPU Libraries

Windows users will need Vulkan itself, together with the GLFW window library. You will also need a compiler for the glslc shader language, but on Windows this is included with the Vulkan installation.

See the appendix “Installing Vulkan etc” for more details.

Cfitsio and FITS files

One of the examples, Median, supports using FITS files, and for this you will need the cfitsio library. This is a standard building block for astronomical software, and you may already have it. If not, see the appendix “Installing Cfitsio”. (It is possible to build Median without cfitsio, although then it cannot work with FITS files.) It will help if you have a FITS image viewer such as SAOImageDS9 available, just to be able to look at the images produced by the program. There is one small test FITS image included, but you may like to have some data of your own to try. Any FITS file whose main image is a reasonably-sized 2D image should be suitable.

Installation checklist.

You should have:

- The source code for the example programs.
- Visual Studio.
- The Vulkan and GLFW libraries and associated include files, and the glslc compiler that comes with Vulkan.
- The cfitsio library and associated include files (optional, but desirable).
- Ideally, a FITS viewer such as SAOImageDS9.

Building the example programs.

Once you’ve downloaded the source code, and have the necessary libraries installed, building the example programs should be simple.

If you don’t already have a terminal window, you need to start one up configured for 64-bit software development. Click on the Start menu and enter X64 in the search field. You should be offered something like “X64 Native Tools Command Prompt for VS 2022”. That’s what you want.

You then need to go to the directory where you download the example programs. Start at the top-level ADASS2024_GPU directory. You should see the Metal and Vulkan sub-directories.

```
cd Vulkan
```

You should see the three directories, Adder, Median and Mandel.

```
cd Adder
```

The Adder directory as released contains two Makefiles. The one just called 'Makefile' is for MacOS/Linux. The one called Makefile.win is for Windows. You can run nmake to build the Adder program using:

```
nmake /F Makefile.win
```

During the tutorial, you may need to rebuild the program after making changes to the code, and you may find it simpler to rename the Makefiles.

```
rename Makefile Makefile.unix
rename Makefile.win Makefile
nmake
```

This should build the Adder example program. If it doesn't, this may be because you have some of the include files or libraries installed in a different directory to the one assumed by Makefile.win. Look at the start of Makefile.win, and see if there are any obvious changes needed to the INCLUDE and LIBRARY settings there. Adder only uses Vulkan itself, any you may need to change VULKAN_VERSION.

Then repeat this in the Median and Mandel directories. Median uses the cfitsio and zlib libraries, and you may need to check that they are where Makefile.win expects. Mandel uses the GLFW and GLM libraries, so you may need to check their locations. (If you've not installed cfitsio, use nmake to build Medianx instead of the default Median – `nmake /F Makefile.win medianx`.)

Hopefully, you now have all three executable programs: Adder, Median, and Mandel. They need to be run from their specific sub-directories. You can see if they run – Adder and Median should just run and output something to the effect that everything ran OK, while Mandel should open a graphics window and draw a fairly well-known shape in it. If that doesn't happen, let me know (e-mail to keith@knaveandvarlet.com.au). You can play with them more if you like, but that might spoil any surprise the tutorial has in store.

Possible problems

If you have any problems building the programs, some of the libraries or their associated include files may be in a non-standard location and the compiler or linker isn't finding them. You may find you have to modify the Makefile.win files to allow for this. All the Makefile.win files start by setting values for INCLUDES and LIBRARIES, as well as VULKAN_VERSION, and you may need to modify these. Do feel free to e-mail keith@knaveandvarlet.com.au preferably with as many details (error messages copied from the terminal, for example) as possible.

If any of the programs build but don't run, they may not be able to find all the shared libraries. The Vulkan installation should have put the Vulkan libraries in a standard location, and the Makefile.win files link against a non-shareable GLFW library. For Median, the Makefile.win file copies the cfitsio and zlib shareable libraries into the Median directory itself, where the program should have no trouble finding them – it might be worth checking they were copied across as expected.

It's also always worth checking that your graphics drivers are up to date.

Help, support and feedback

Any problems, e-mail keith@knaveandvarlet.com.au

If you do hit a problem and fix it yourself, please let me know the details. Someone else may hit the same problem and this will help me to help them. And please feel free to tell me if you got things to run without problems; it's always nice to know! The Windows versions (and these instructions) are relatively new, and I'm relatively inexperienced with the Windows command prompt, so there may well be some initial problems.

There may be corrections and changes to the example programs between now and the actual tutorial, in which case new releases should appear on GitHub.

Appendix: Installing Vulkan etc

These instructions are a simplified version of more detailed instructions from the excellent vulkan-tutorial site. For more details:

https://vulkan-tutorial.com/Development_environment

Go to the LunarG website at https://vulkan.lunarg.com and click on the 'Download Developer Tools' button with the various OS symbols. This takes you to a window that offers you a choice of versions. Select the latest Windows version – at the time of writing, this is 1.3.290.0. Download and run the installer.

This will ask where you want Vulkan to be installed. The default is C:\VulkanSDK\1.3.290.0 (or a later version number). This is the location assumed by the Makefile.win files, so use it unless you have a good reason not to. The standard default installation should be all you need.

At this point, it might be as well to check that Vulkan is supported on your machine. Open up a terminal window, and try:

```
cd C:\VulkanSDK\1.3.290.0\bin
```

```
vkcube
```

You should see a spinning cube. If you don't, you may need to check that your graphics drivers are installed and your GPU is supported. You may find the command:

```
vulkaninfo –summary
```

tells you something useful.

You now need to install the GLFW library.

GLFW can be downloaded from https://www.glfw.org/download.html which will offer you the latest version – this is 3.4 at the time of writing, and that is what is assumed by the Makefile.win files. Download the 64-bit Windows binaries and open the .zip file.

You now need to copy the files to somewhere the Makefiles can pick them up. The Vulkan-tutorial website suggests putting them in a sub-directory of Visual Studio, but if you're working on command line, this leads to very long filenames with spaces in them, which I find awkward. Instead, I have created a folder in my home directory called Libraries – so mine is C:\Users\keith\Libraries – and used File Explorer to copy from my Downloads folder into there. So I now have a folder called C:\Users\keith\Libraries\glfw-3.4.bin.WIN64. It's an awkward name, but the Makefile.win files will pick it up automatically.

Although the Vulkan tutorial says you need the GLM library, these example programs don't use it. If you go further with Vulkan, especially for graphics, you may need a library like GLM, but not for this tutorial.

Appendix: Installing Cfitsio

Cfitsio is a standard library for astronomical software, and you may already have it on your laptop. For Windows, this needs to be built from source, together with the Zlib compression library. This tutorial only uses cfitsio for the Median example program, which has the option of working on either dummy data or an actual image read from a FITS format file. While it's fun to see this work on real data, you may prefer to treat this as an optional extra.

I hope you do chose to install cfitsio. If you do, the following instructions mostly duplicate those in the cfitsio installation instructions. The only difference is that what follows installs the zlib and cfitsio files where the Makefile.win file for the Median program expects them. If you put them somewhere different, you may need to modify that Makefile.win file to match.

(If you chose not to install cfitsio, when you build the Median example program, use 'nmake medianx' instead of 'nmake median'. The version with the 'x' at the end builds without cfitsio – and will not be able to read or write FITS files, which is a pity, but can still run a median filter through dummy data.)

The cfitsio source can be downloaded from NASA's HEASARC page:

https://heasarc.gsfc.nasa.gov/docs/software/fitsio/fitsio.html

which has a link in the first section called 'Windows PC cfit-4.5.0.zip file' which will download the latest Windows version. Expand this, and you'll have a directory called cfit-4.5.0, which contains a directory cfitsio-4.5.0. (4.5.0 is the latest version at time of writing. If this has changed, replace 4.5.0 in these instructions with the latest version number).

You also need the Zlib compression library. At the end of the Zlib home page https://zlib.net there is a permalink for the most recent Zlib version. Download the .zip file and expand it. That should give you a directory called zlib131, which contains a directory zlib-1.3.1 (again, the version numbers may be different).

You need to build Zlib first, then cfitsio.

If you don't already have a terminal window, you need to start one up configured for 64-bit software development. Click on the Start menu, and enter X64 in the search field. You should be offered something like "X64 Native Tools Command Prompt for VS 2022". That's what you want.

You need to have cmake installed on your laptop, but you should have that as part of the C++ development options with Visual Studio.

Set your default directory using 'cd' to that zlib131 directory in the expanded download files. Now:

```
mkdir zlib.build
cd zlib.build
cmake ..\zlib-1.3.1
    -DCMAKE_INSTALL_PREFIX=C:\Users\<yourname>\Libraries\zlib
cmake --build . --config Release
cmake --install .
```

You need to change `<yourname>` so that the files are installed into the Libraries folder in your home directory that should already contain the glfw-3.4.bin.WIN64 directory with the GLFW files – assuming you installed these as described in the previous appendix. (Note that the long cmake command shown above needs to be all on one line – don't just copy and paste it as it stands from this PDF file. If you do mistype a cmake command, I've found the best thing is often to just delete the zlib.build directory and start again.)

Now to build cfitsio.

Set your default directory using ‘cd’ to the cfitsio-4.5.0 directory in the expanded cfitsio download files. Now:

```
mkdir cfitsio.build
cd cfitsio.build
cmake ..\cfitsio-4.5.0
  -DCMAKE_PREFIX_PATH=C:\Users\<yourname>\Libraries\zlib
  -DCMAKE_INSTALL_PREFIX=C:\Users\<yourname>\Libraries\cfitsio
cmake --build . --config Release
cmake --install .
```

Ignore warnings from the compiler as it builds cfitsio.

In your `C:\Users\<yourname>\Libraries` directory you should now have cfitsio and zlib directories (along with the GLFW directory from when you installed that). The cfitsio and zlib directories should each have bin, include, and lib sub-directories.