# Applied GPU Programming, DD2360
# Assignment 4, exercise 4: Discuss, Reflect, Contrast
## Nvidia CUDA vs. OpenCL on performance and easiness to use

Etienne Jodry, Jérémy Kaplan

December $13^{th}$, 2020

CUDA is an application programming interface paired with a compilation chain, while OpenCL operates on a layer right above, wrapping more programming libraries under a single API that can abstract away the pieces of hardware it interacts with. Both allowed us to write and execute code for GPUs using their parallization capabilities. Despite having a lot of similarities, both work in different ways along with slightly different purposes. This is why they differ in their syntax and writing, in their performance and their cost, omitting the fact that OpenCL is compatible with a wider range of hardware.

With consideration of the variety and finickiness of the tasks CUDA and OpenCL are designed to tackle, it is no wonder that none of those two are easy to use. The instructions are complicated and extremely configurable in order to catch with the targeted devices; but as a consequence, OpenCL is more complicated to write. This worked both from our own experience and from the theoretical statement that complexity has to lie somewhere, let it be hidden in the underlying implementation or in the countless parameters of adjustable API calls. For a concrete example, CUDA imposes less overhead code for writing a simple application. Whilst OpenCL requires to call functions (`clGetPlatformIDs()`, `clGetDeviceIDs()`, `...`) and create several objects (`cl_context`, `cl_command_queue`, `cl_program`) before we can launch the kernel. This has the advantage of allowing for more customisation for scalable OpenCL application, but cost more code maintenance. OpenCL only knows scarcely about the targeted devices, so we programmers have to configure our function calls. That is the basic trade-off we could experience while coding. Overall, CUDA was easier to use, less finicky, less messy, in our range of needs.

On the code we created CUDA offered better performances, but it might be because we did our tests on an Nvidia GPU, in this case OpenCL is implicitly using CUDA with an extra layer. Historically OpenGL had better performances on AMD graphic cards but it is hardware dependant since Nvidia added a broader support in its drivers.

However, our work does not cover the immense field of programming. Noticeably, CUDA's support is more narrow than OpenCL's, the latter aims at being applicable for any computer devices, from CPUs to FPGA cards, on x86 or x86-64 as well as ARM. CUDA is owned and maintained by NVidia and only operate with NVidia's GPU. Therefore applications written with OpenCL have the potential of running on a vast range of hardware elements. This feature may allow for a lower cost.

Moreover, OpenCL is open-source, as opposed to CUDA which is property of NVidia corporation. Even if CUDA is freely available for everyone to use, end user will always have less control on the hardware and potential modifications that would best fit his use.