AE GPU SDK

Build Instructions

(5/18/2024)

Note:

We are excited to introduce you the new AE GPU SDK in 16.0!

Please see **SDK_Invert_ProcAmp** in the Effect folder as an example plugin.

More AE GPU SDK API information can be found in After Effects SDK Guide.

Mac:

Setup instructions:

1. The plugin uses BOOST to process GPU kernel files, you need to install BOOST on your machine. You can install BOOST through homebrew, or direct download from boost.org.

2. Once BOOST is installed, obtain the installation path. Usually BOOST installation path looks like this:

   /usr/local/Cellar/boost/1.67.0_1/include

3. Open the SDK_Invert_ProcAmp project in Xcode and go to Preferences -> Locations -> Custom Paths

4. Add this entry:

   Name: **BOOST_BASE_PATH**
   DisplayName: **BOOST**
   Path: **[Your BOOST installation path]**

5. If you see python errors when building, make sure you have python installed for bash (not for zsh).

   If you have installed python3 for bash but still seeing "**python command not found**" error, go to **Project Settings -> Build Rules** and try changing the "**python**" keyword to "**python3**"

Win:

Set up instructions:

1. Install **Boost** from [**boost.org**](boost.org)
   a. Unzip the boost package and run bootstrap.bat
   b. Then run .\b2 to build boost
2. Install the **CUDA SDK** from https://developer.nvidia.com/cuda-downloads.
   Please use the same CUDA version that your AE build is using.
   AE 24.3 uses CUDA 11.8.
3. Install the latest **DirectX Compiler** from https://github.com/microsoft/DirectXShaderCompiler/releases
4. Setup system environment variables:
   CUDA_SDK_BASE_PATH: [CUDA installation path]
   (example: C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.8)
   BOOST_BASE_PATH: [BOOST installation path]
   (example: C:\boost_1_80_0)
   DXC_SDK_BASE_PATH: [DXC installation path]
   (example: C:\dxc_2023_08_14)

Deployment instructions (DirectX):

The DirectXAssets folder generated alongside the binary also has to be copied for the effect to work. The assets are loaded by the application at runtime.

PF_OutFlag2_SUPPORTS_DIRECTX_RENDERING has to be set as a part of Global Flags for DirectX rendering and the corresponding value has to be updated in the PiPL resource file. The effect will fall back to the CPU implementation if the flag is missing.

CUDA Runtime API vs. Driver API:

1. **Utilize CUDA Driver API**
   For best compatibility, we highly recommend utilizing only the CUDA Driver API. Unlike the runtime API, the driver API will be compatible with future drivers. Please note that the CUDA Runtime API is built to handle/automate some of the housekeeping that is exposed and needs to be handled in the Driver APIs, so there might be some new steps and code you would need to implement to migrate from the Runtime API to the Driver API.

2. **Alternative 1: Statically Link to CUDA Runtime**
   If the CUDA Runtime API is needed then we recommend statically linking the CUDA Runtime, cudart_static.lib. This will work with future driver versions.

3. **Alternative 2: Dynamically Link to CUDA Runtime**
   This also works but may be prone to compatibility issues. A compatible CUDA Runtime DLL needs to be available on users' systems to work with future drivers. Currently After Effects ships a copy of the CUDA Runtime DLL corresponding to our recommended CUDA SDK version. This may change in future. If you must dynamically link to the CUDA Runtime then we recommend shipping a copy of the CUDA Runtime DLL with your plugin and use dlopen/LoadLibrary to explicitly load the desired runtimes. For more details, see the CUDA Compatibility section of NVIDIA's GPU Management and Deployment guide: https://docs.nvidia.com/deploy/cuda-compatibility/