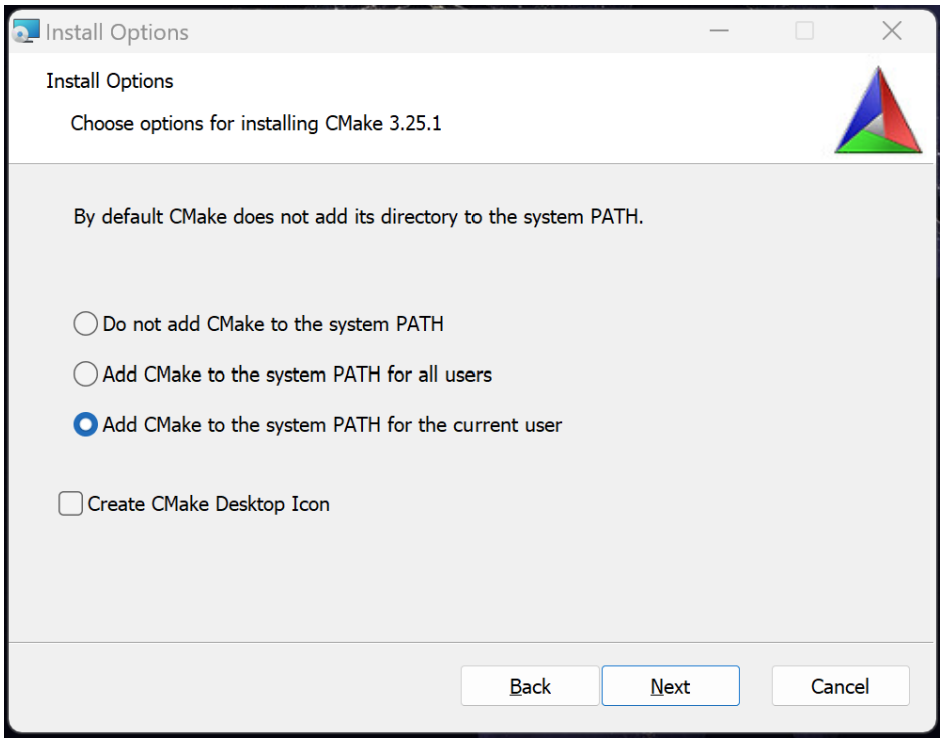
1. Setup

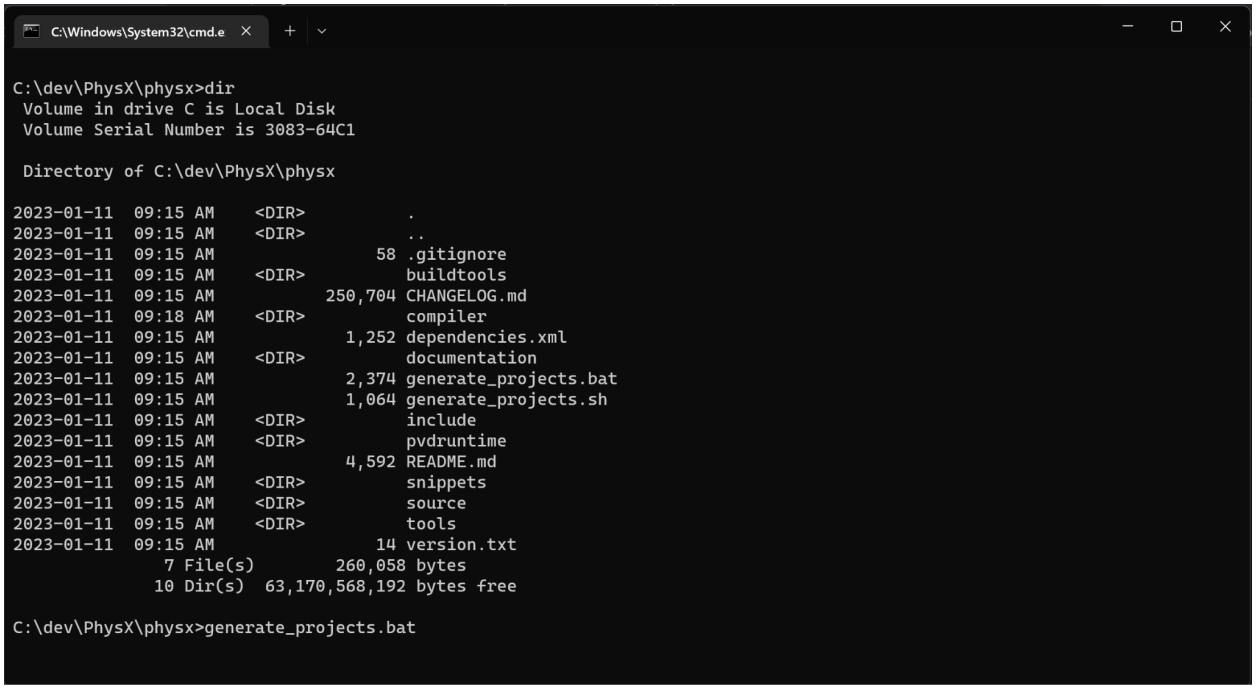
1. Install Visual Studio 2019 (required for building PhysX, but you can use any version for your own projects).

2. Clone the PhysX 5 GitHub repository to your device.

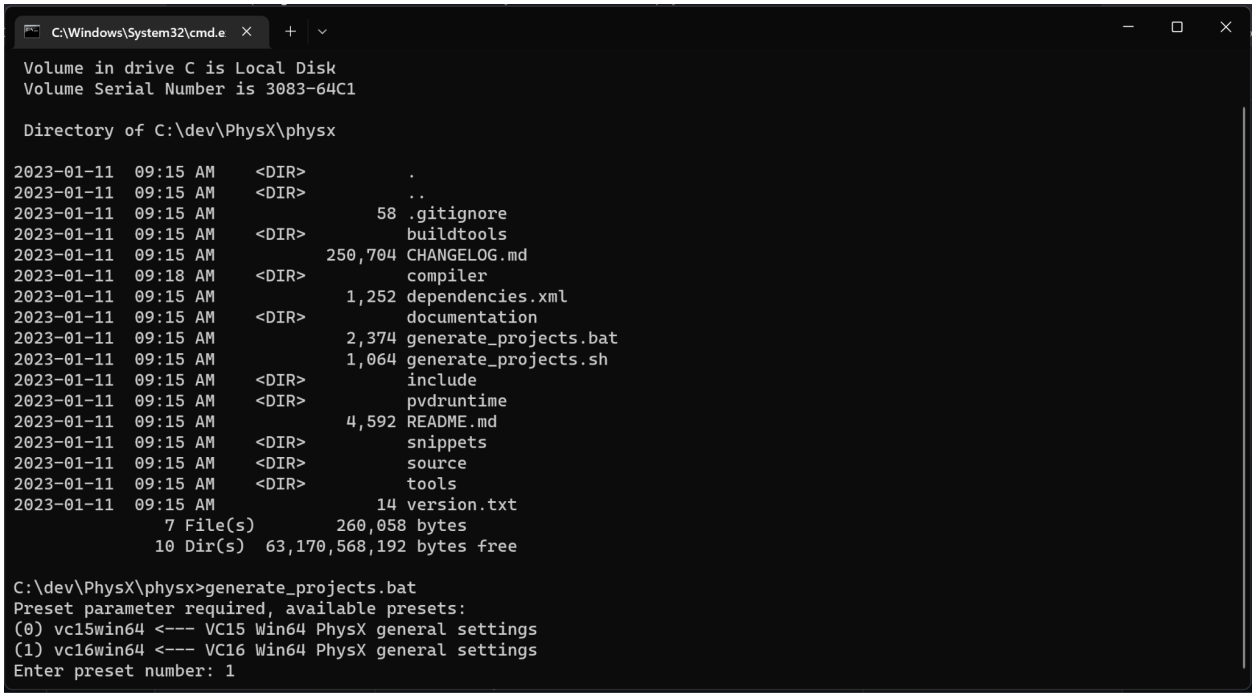
3. Install CMake. Make sure to add it to your system's PATH for at least the current user.



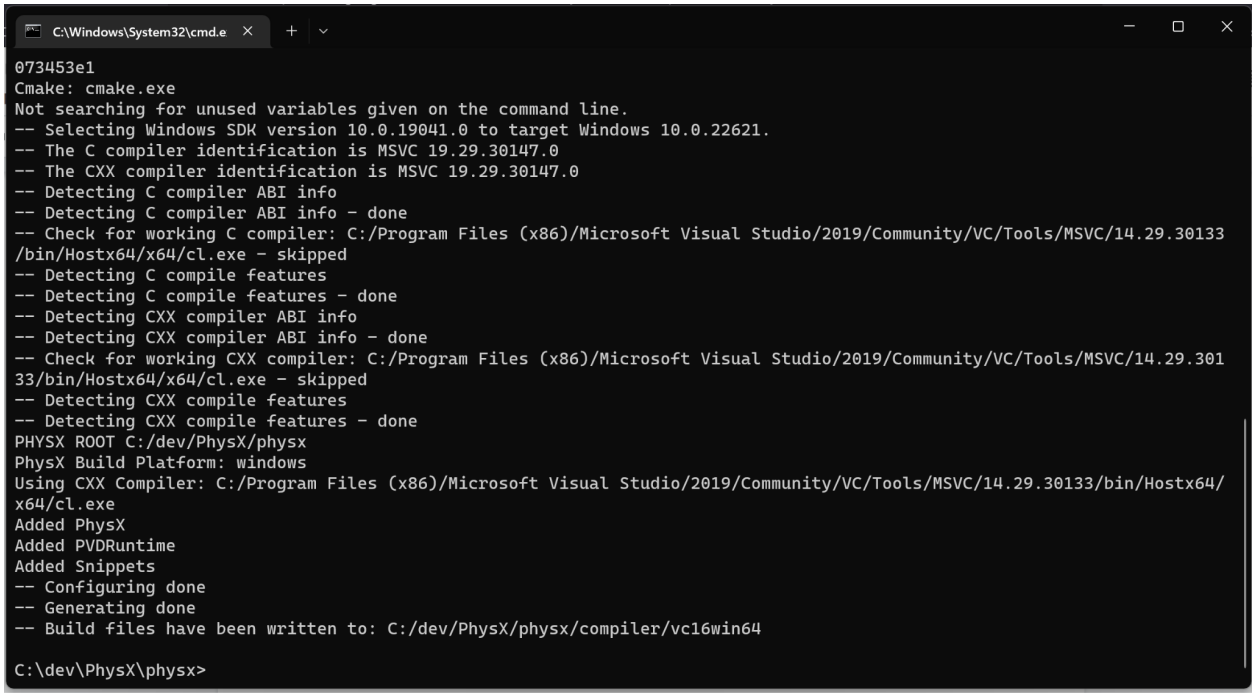
4. Run the PhysX generate_projects.[bat|sh] file in the git directory.



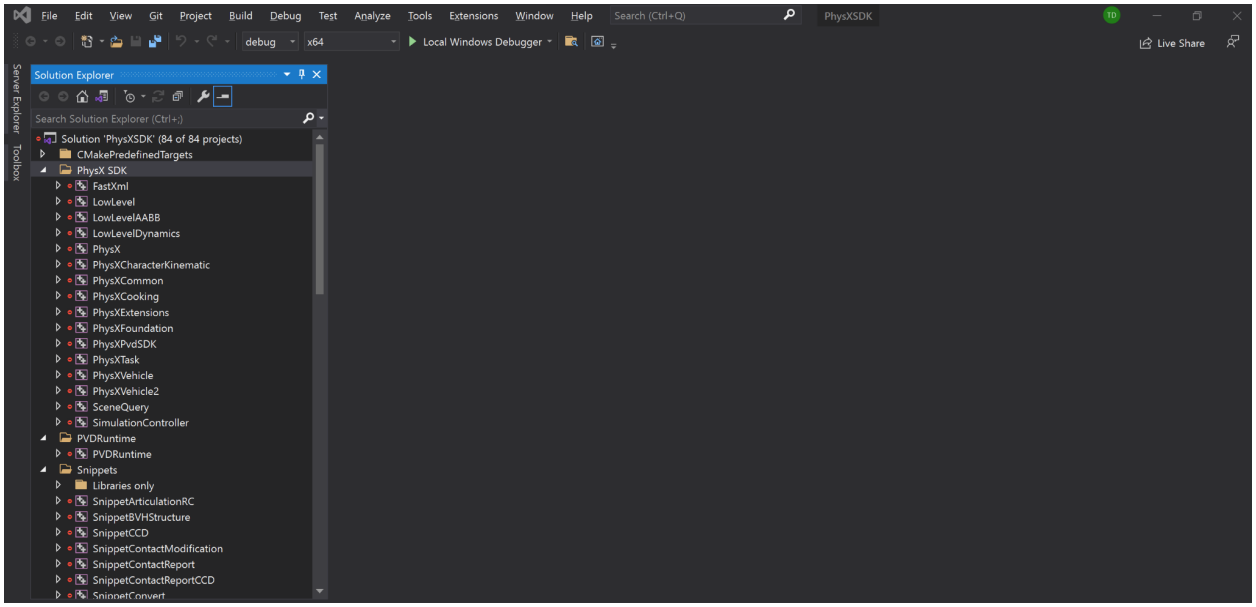
5. Select vc16win64 (the default compiler for Visual Studio 2019).



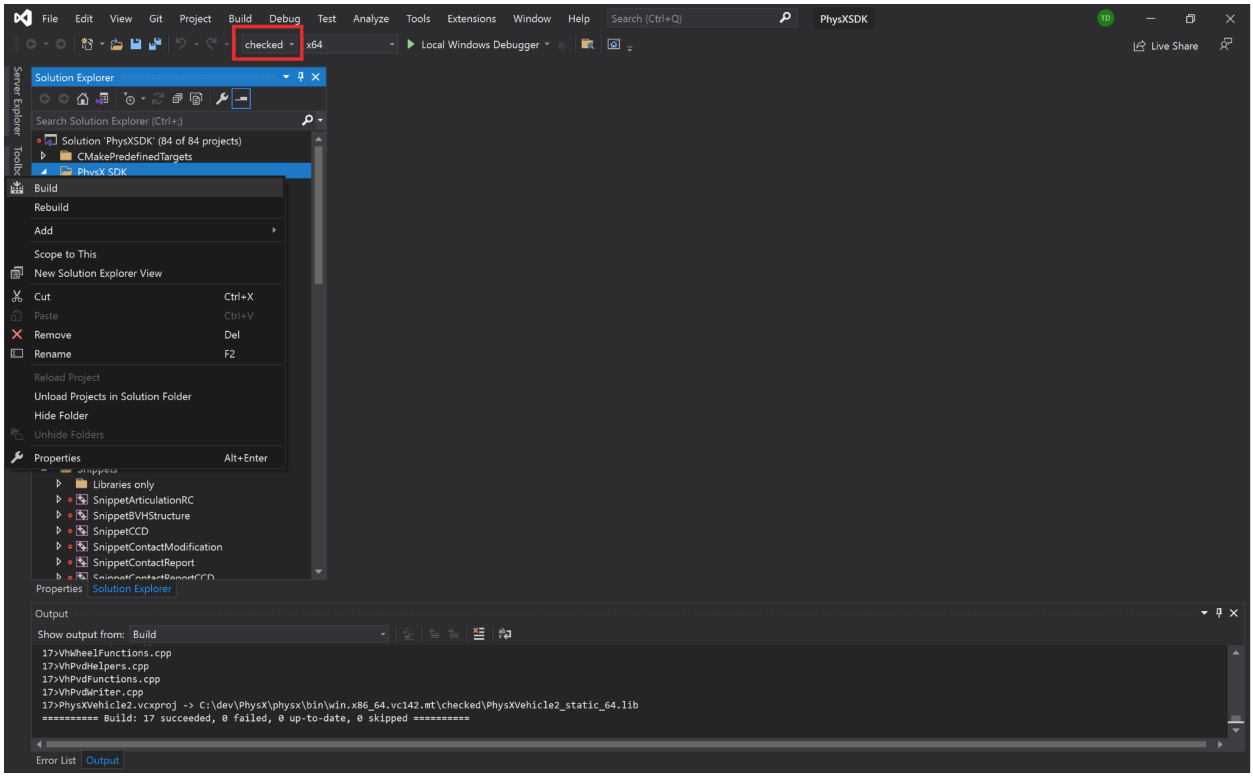
6. Let the script run and you should get an output like this.



7. Navigate to your build directory and open the PhysXSDK.sln file with Visual Studio.

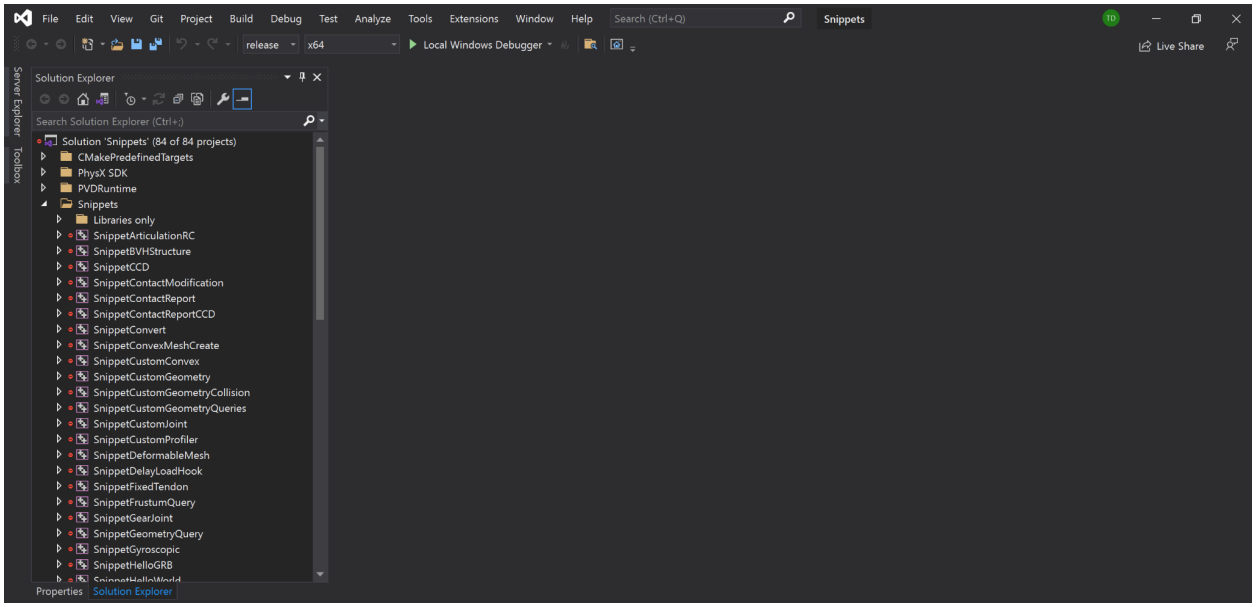


8. Build the PhysX SDK projects using both the “Debug” and “Checked” configurations.

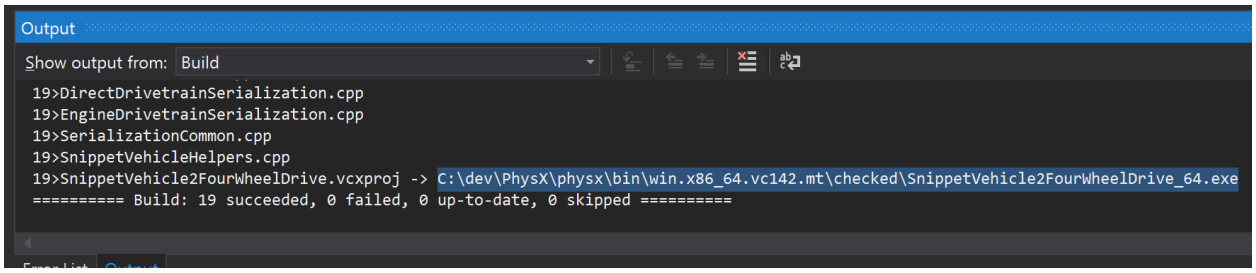


2. Vehicle Snippet

1. Navigate to physx/compiler/vc16win64/sdk_snippets_bin directory and open Snippets.sln in Visual Studio.



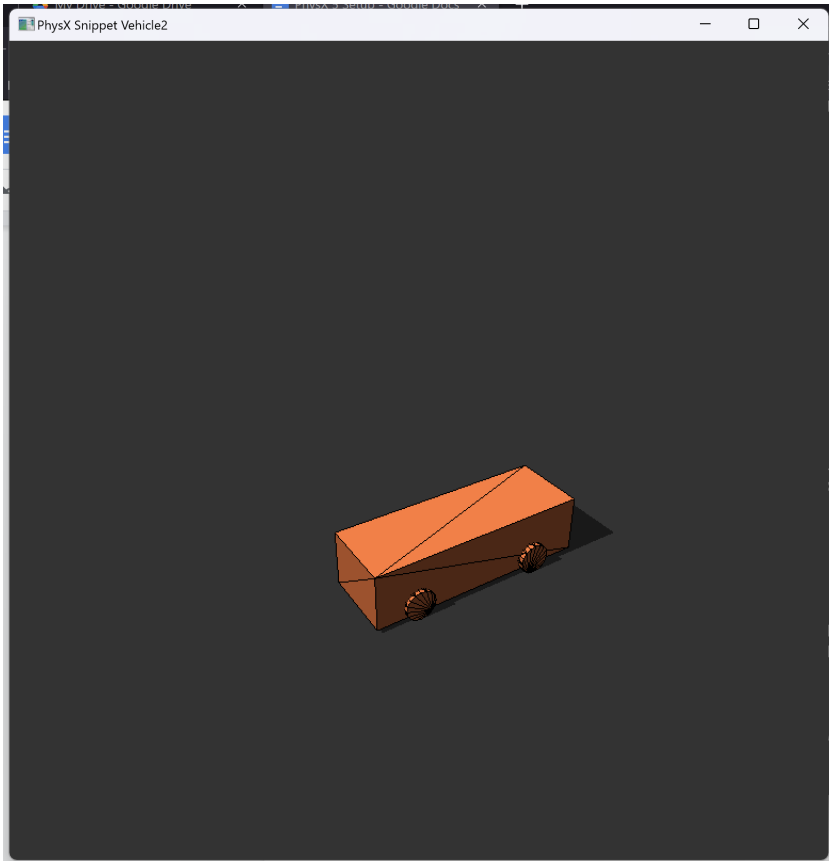
2. Find the SnippetVehicle2FourWheelDrive project in the Solution Explorer. Right-click on the project and select Build.



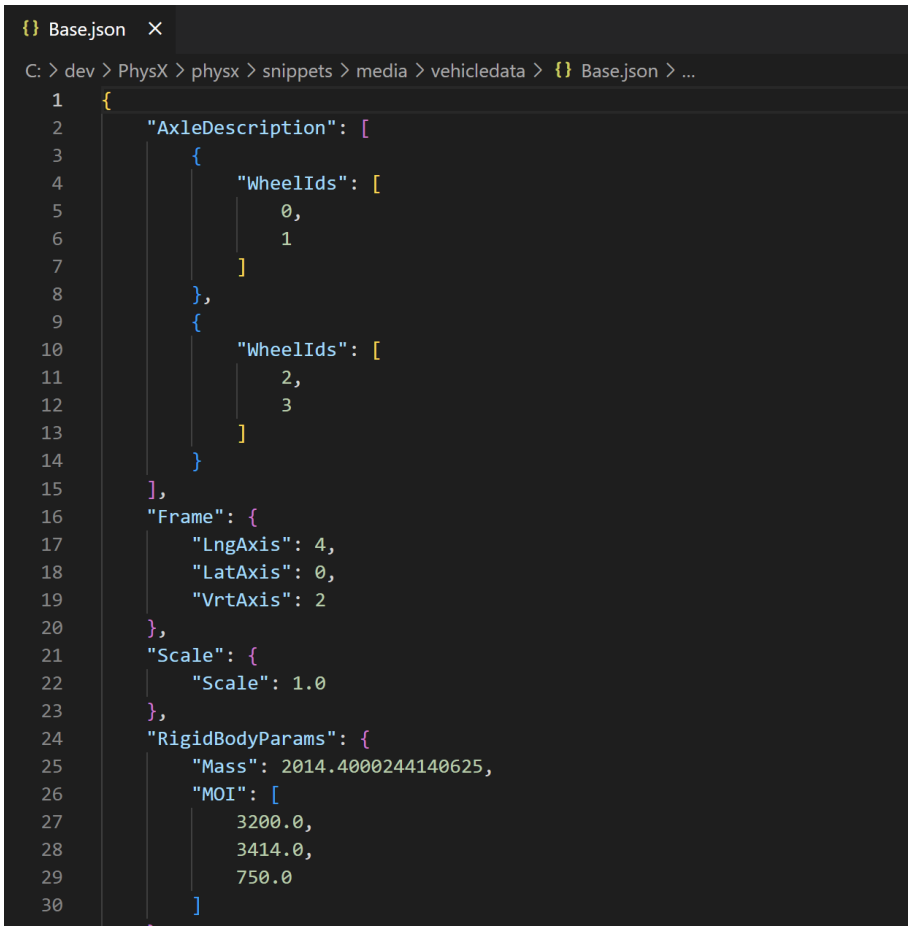
3. Navigate to the specified directory and run the .exe file with the following command:

   SnippetVehicle2FourWheelDrive_64 --vehicleDataPath="../../../snippets/media/vehicledata”

   If everything worked as intended you should get a popup window with a rendered example of a four wheel vehicle that will move around on its own. You can control the camera by left-clicking and dragging.



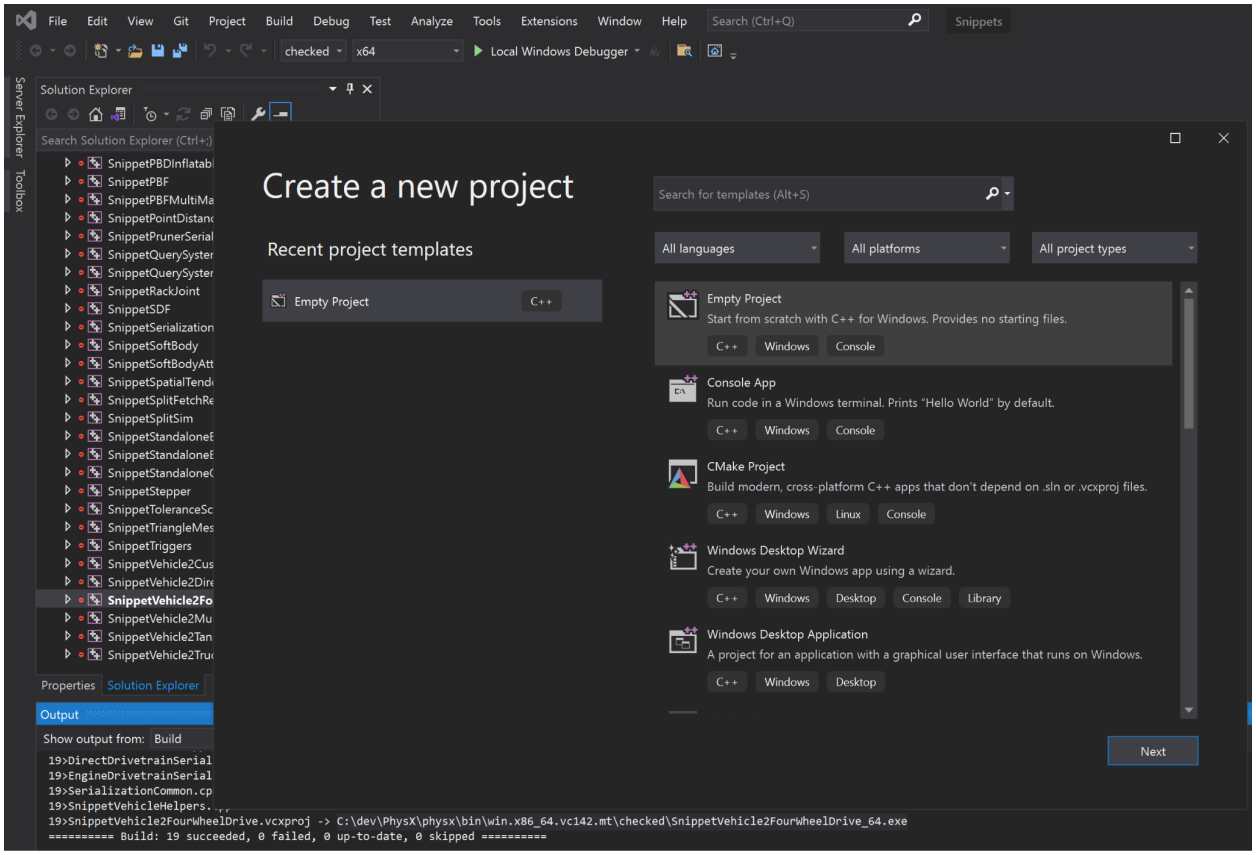
4. The vehicleDataPath we specified above is a directory containing three JSON files outlining various tunable parameters for the vehicles used by the PhysX snippets (newly added in PhysX 5 for your convenience). These list important values that you can change to tweak your driving models in your own project.



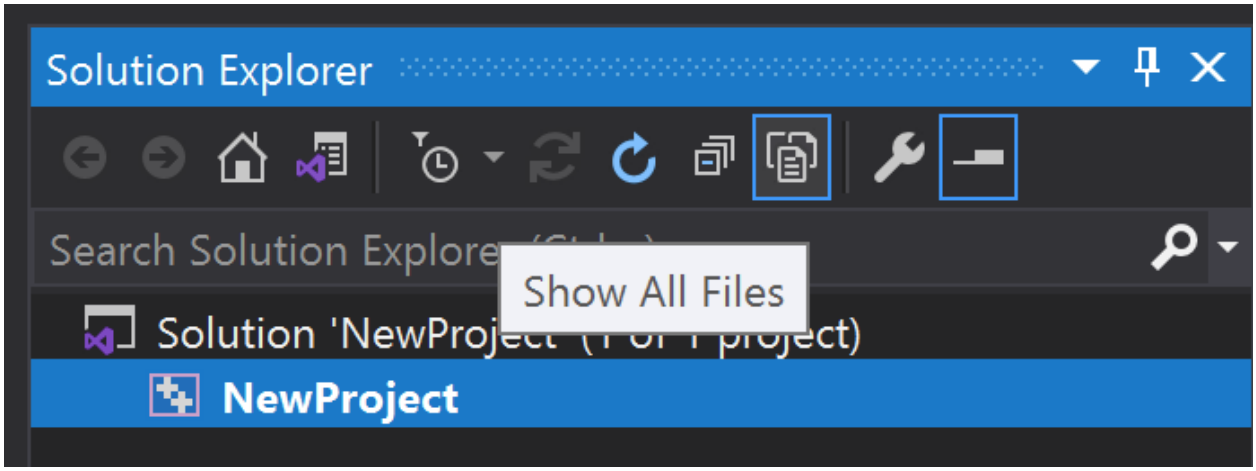
3. Including PhysX in your Project

NOTE: I'll show this from scratch but the process is the same for including in a pre-existing project. Also, feel free to use whatever version of Visual Studio you want for your own projects, but I'll be using 2019 for this.

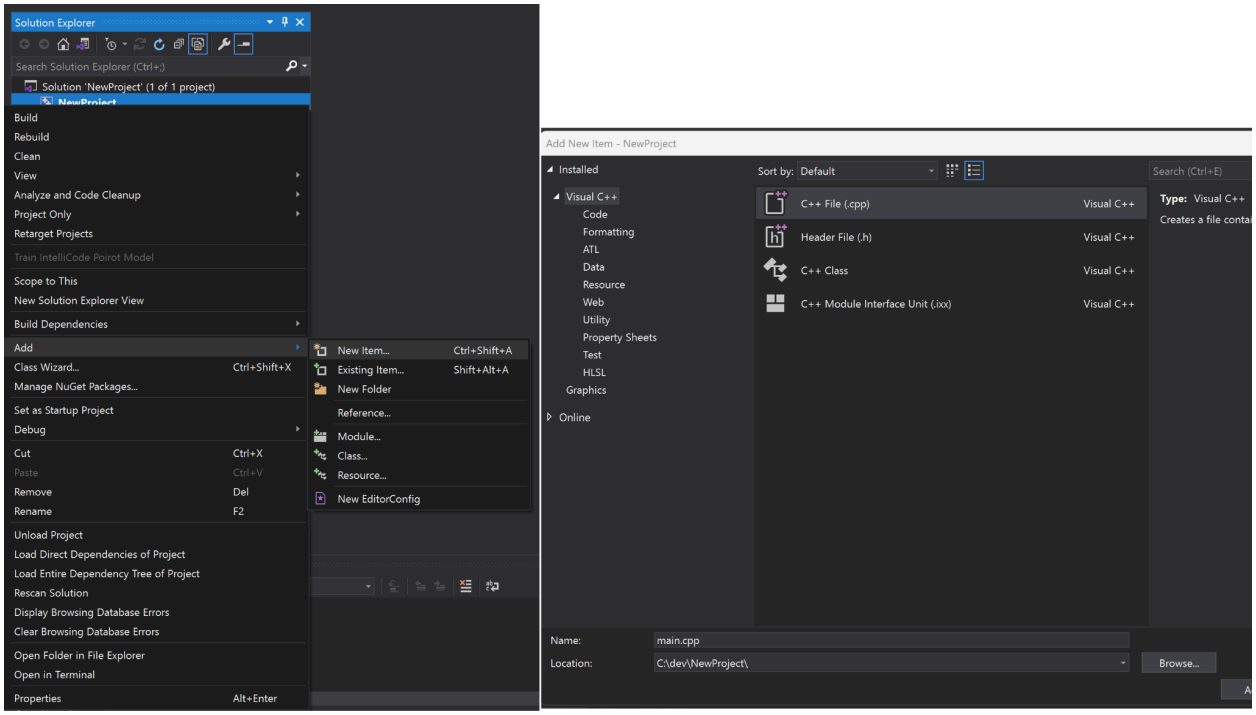
1. Create a new Visual Studio project.



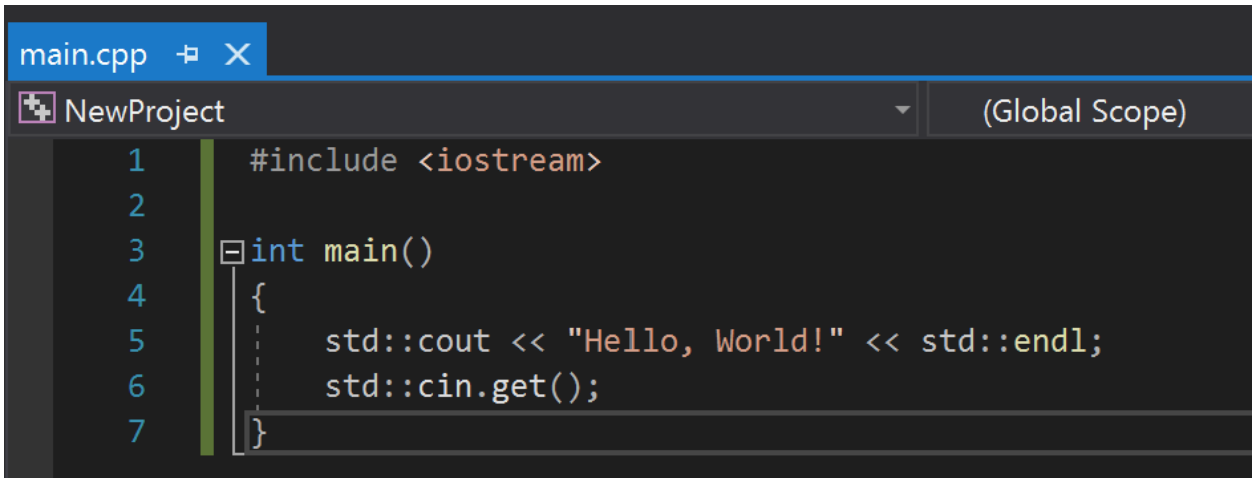
2. If you're not too familiar with Visual Studio I would recommend toggling on the "Show All Files" option in the Solution Explorer to remove the default filters.



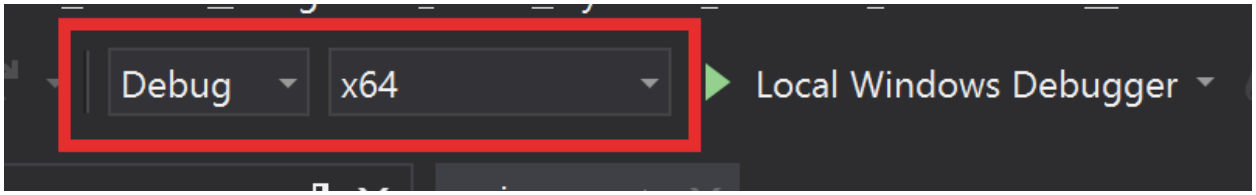
3. Add a new main.cpp file to your project.



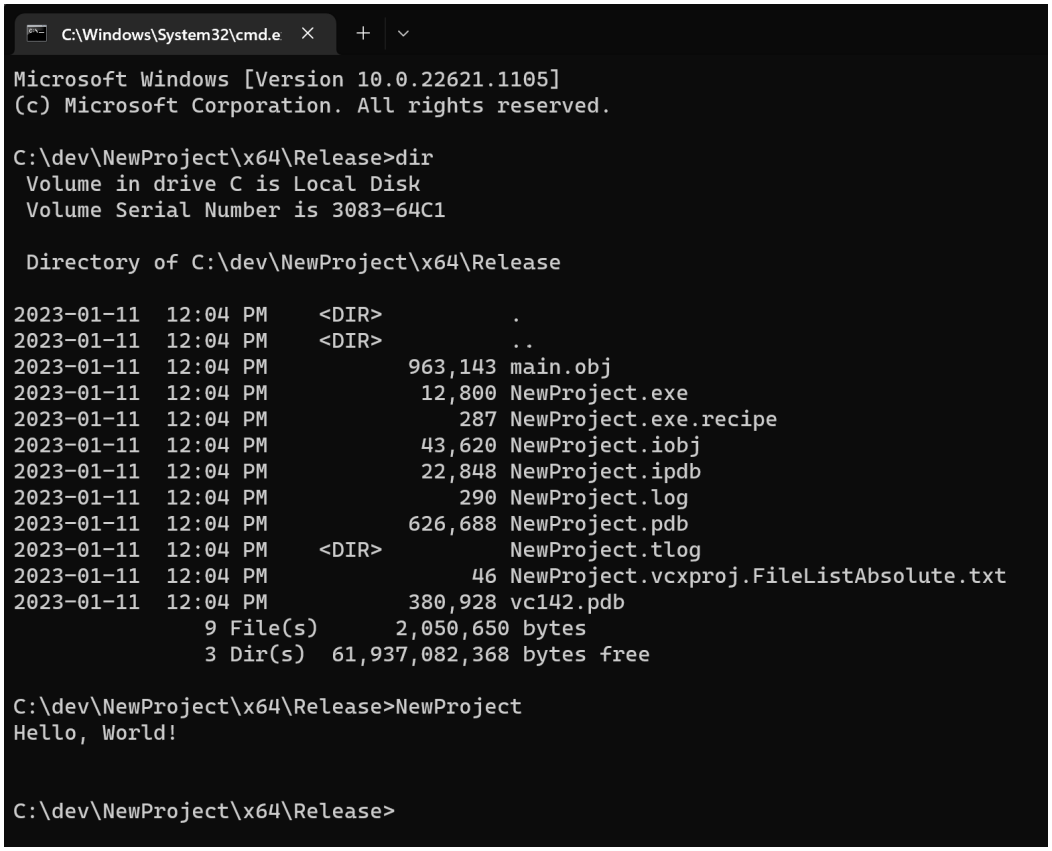
4. Enter the following “Hello, World!” test code into your new .cpp file.



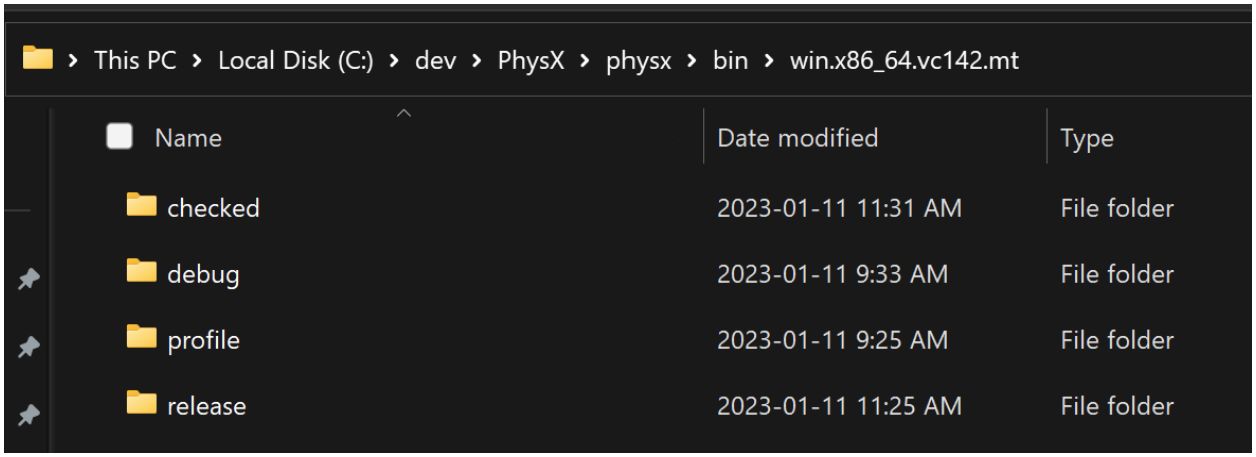
5. Build your project in both the “Release” and “Debug” configurations using x64 (we don’t care about supporting 32-bit systems). This will generate the binaries for your project including the .exe files.



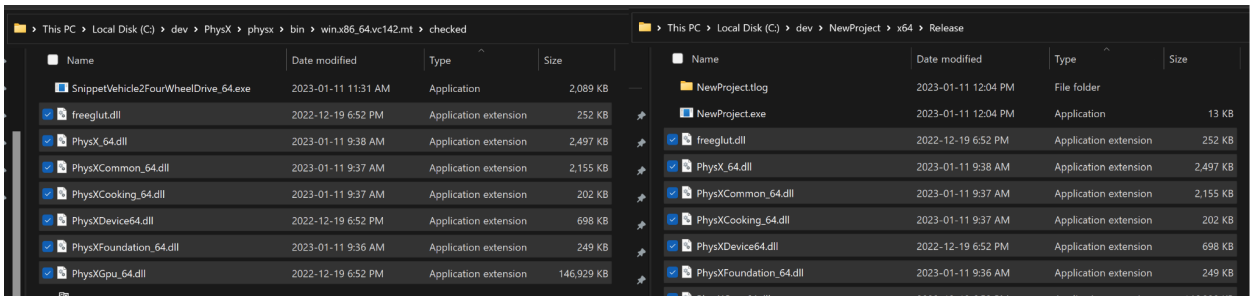
6. Navigate to one of your .exe files and try to run it. You should get something like the following:



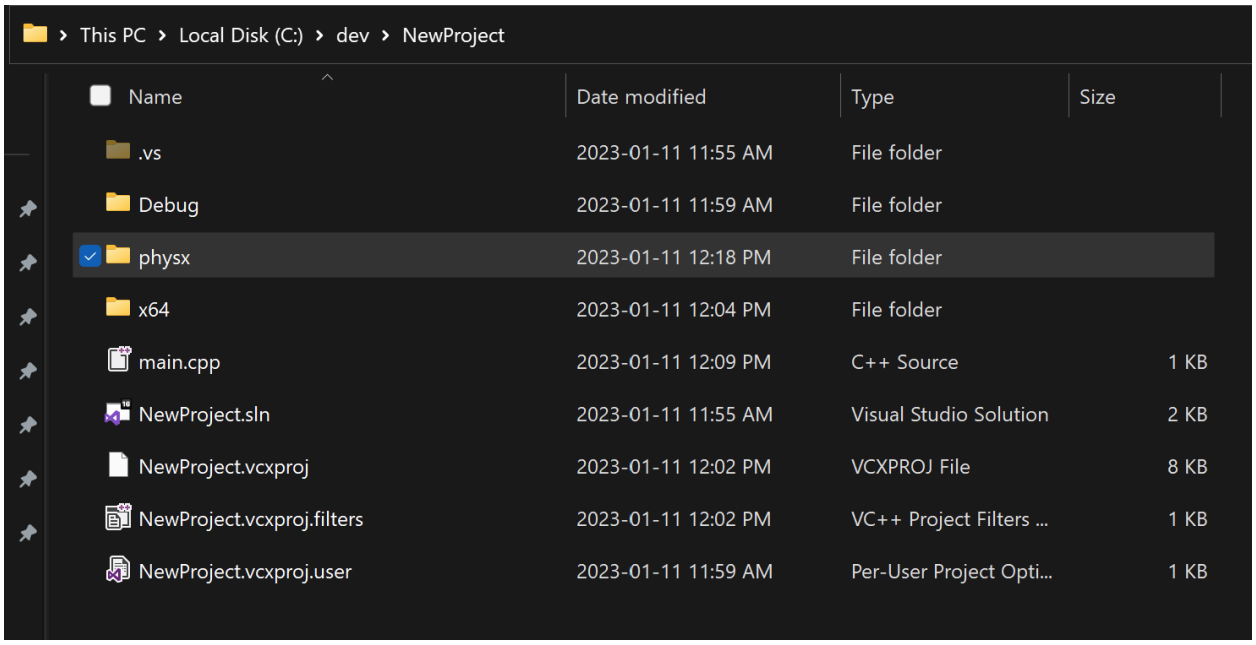
7. In a File Explorer window, navigate to the physx/bin/win.x86_64.vc142.mt directory. Here you will find the PhysX binaries we built earlier in both the “Debug” and “Checked” configurations.



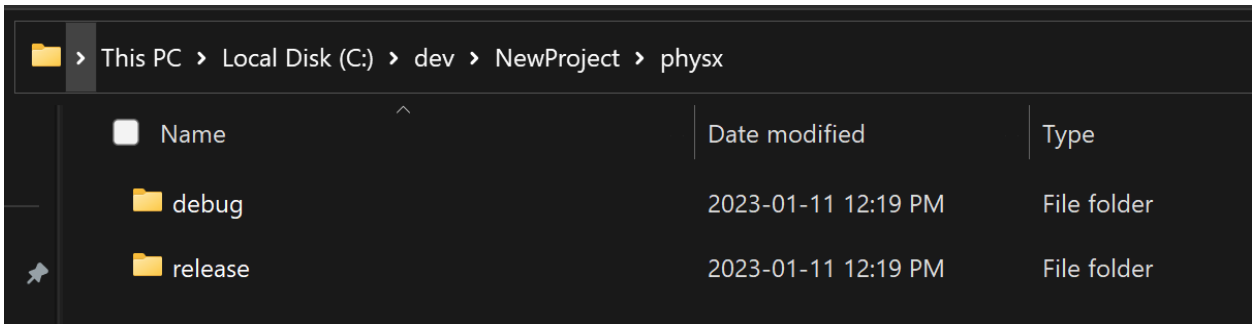
8. In a second File Explorer window, navigate to where the release configuration .exe file we just generated for your project is. Copy all the .dll files from the checked PhysX binaries into the same directory as your release configuration .exe file. Likewise, copy all the .dll files from the debug PhysX binaries into the same directory as your debug configuration .exe file.



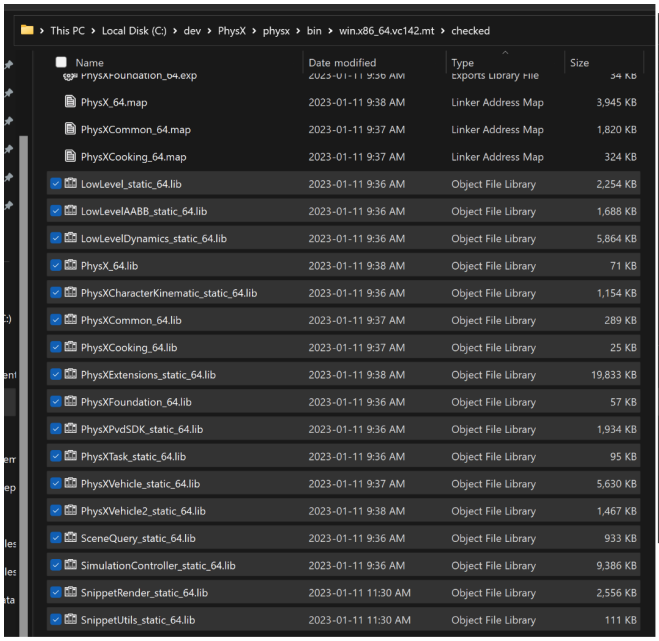
9. In a second File Explorer window create a new directory in your project to hold the PhysX static library binaries.



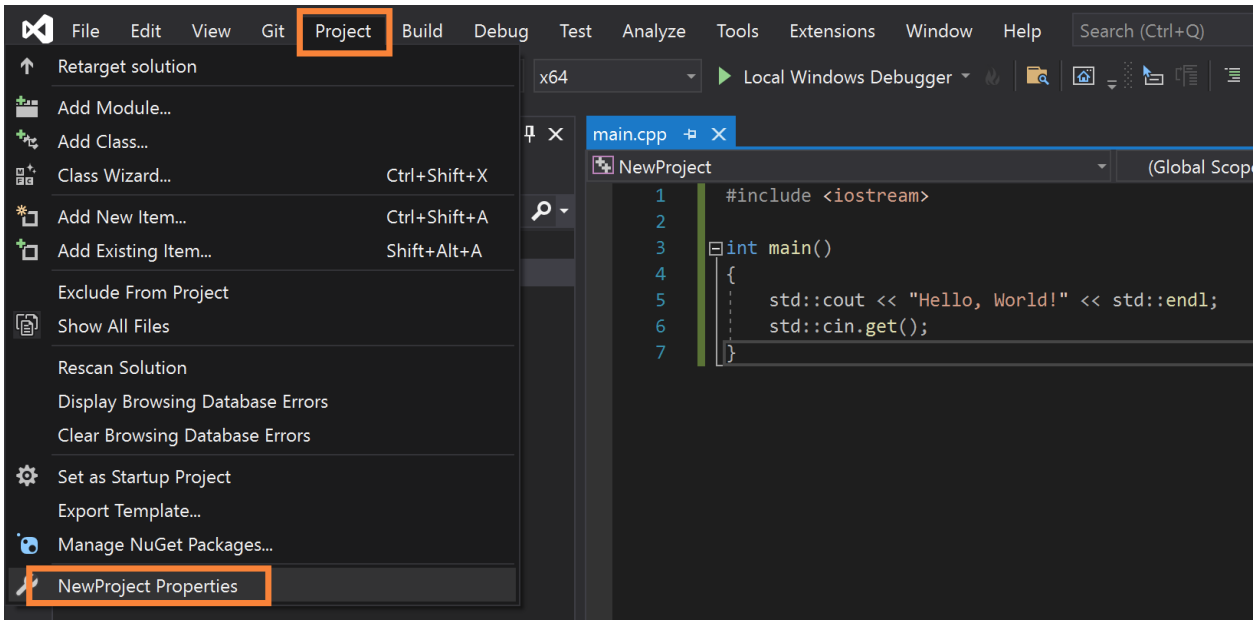
10. And inside of the directory make two new directories called Debug and Release.



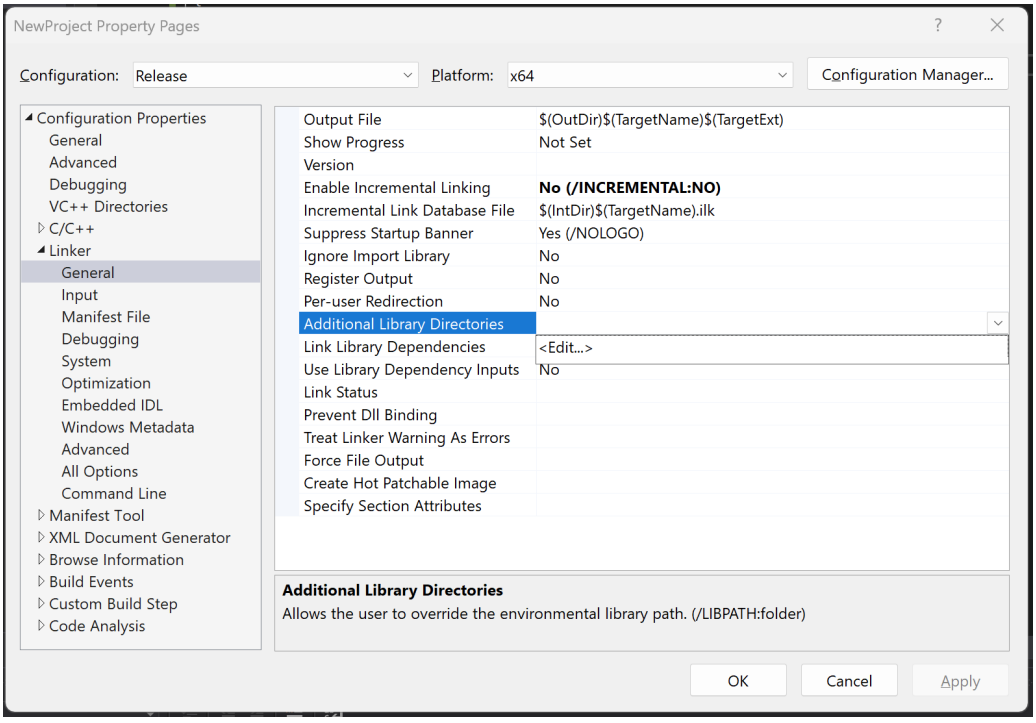
11. Copy the static .lib files from the PhysX checked binaries to your project's release directory. Likewise, copy the static .lib files from the PhysX debug binaries to your project's debug directory.



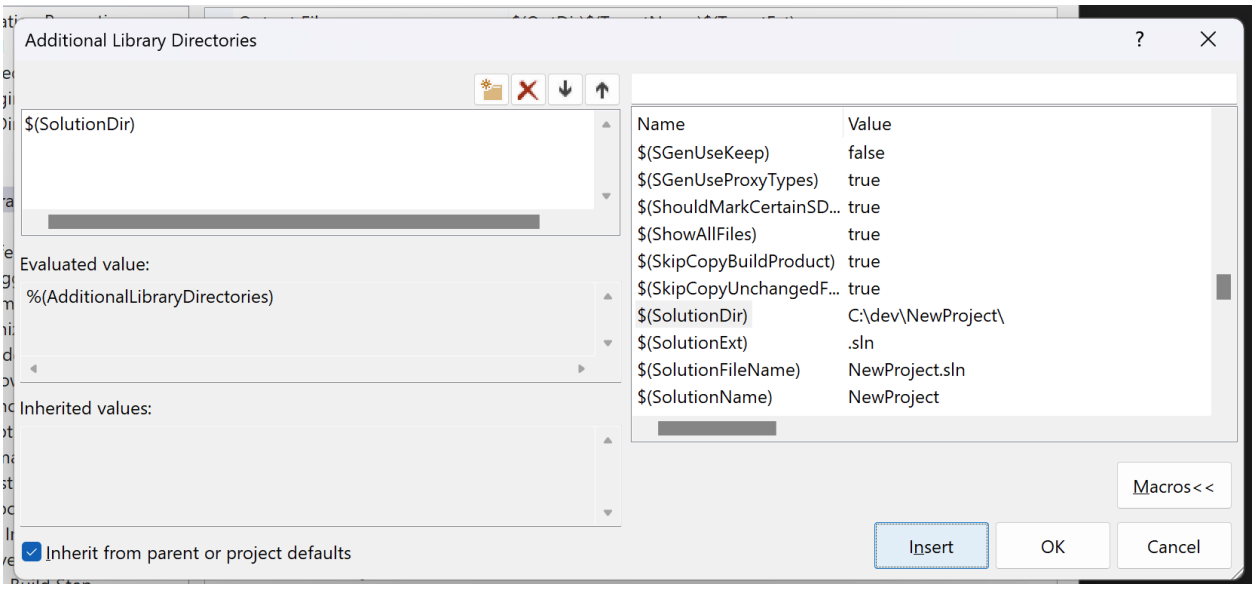
12. Back in Visual Studio, open your project's properties menu.



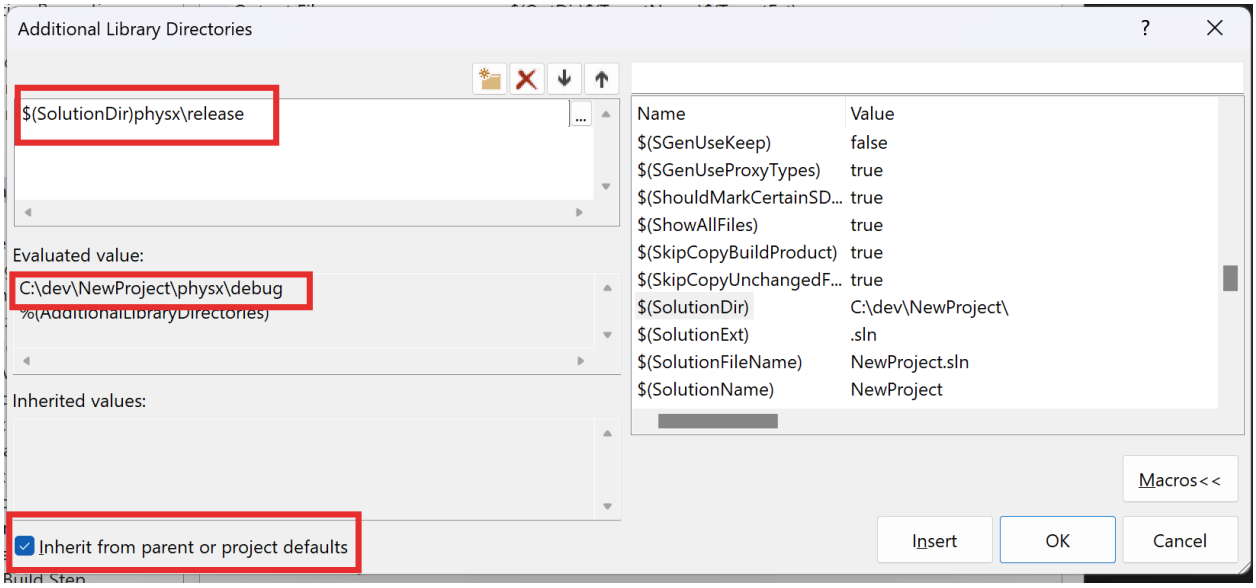
13. In the properties menu, change the configuration to Release and Platform to x64. Click on the Linker tab in the left-hand menu and click on the dropdown menu for Additional Library Directories and select <Edit…>.



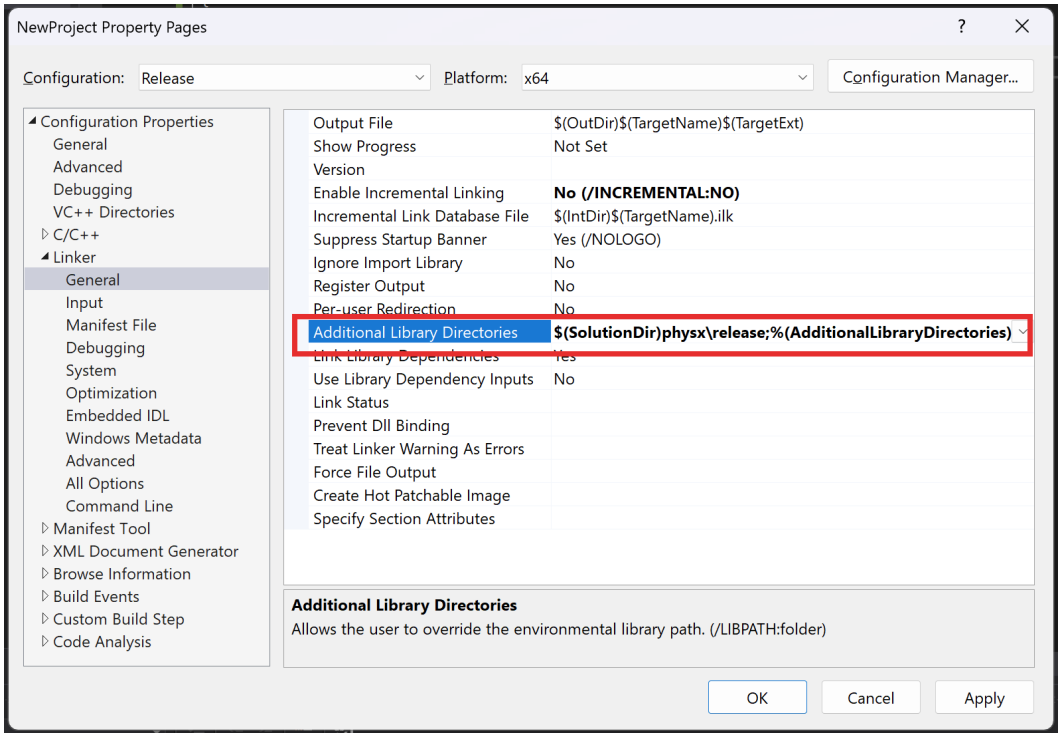
14. In the new popup menu, click on the Macros>>> button to expand the window. In the box on the right scroll down to $(SolutionDir) and press the Insert button.



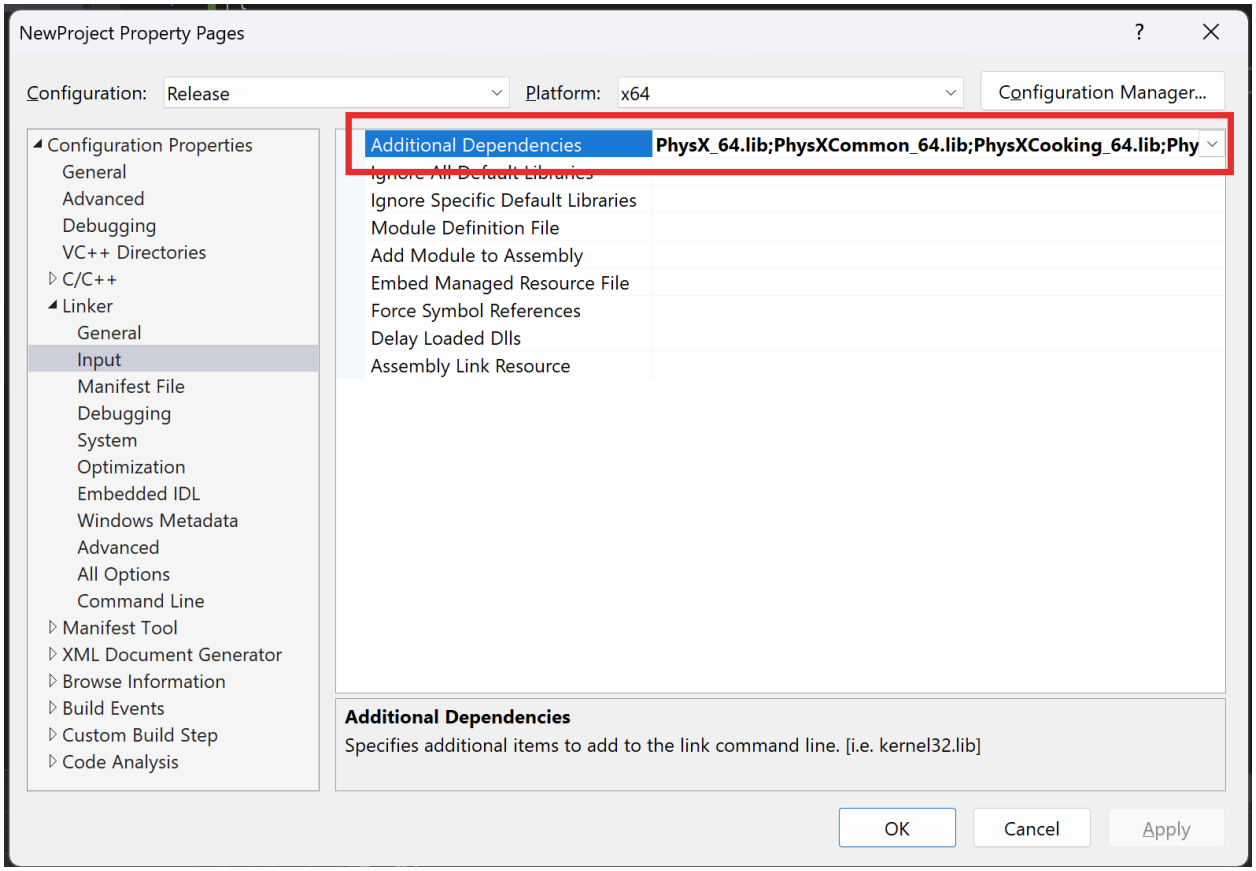
15. This will add the macro $(SolutionDir) to the path on the left. This macro makes a relative path to the directory containing your project's .sln file which will help ensure that your application is able to find its dependencies on other devices in the future. Further edit this path so that it leads to the release directory we made just a while ago. The Evaluated value area should display the correct absolute path to this directory. Finally, ensure Inherit from parent or project defaults is also ticked.



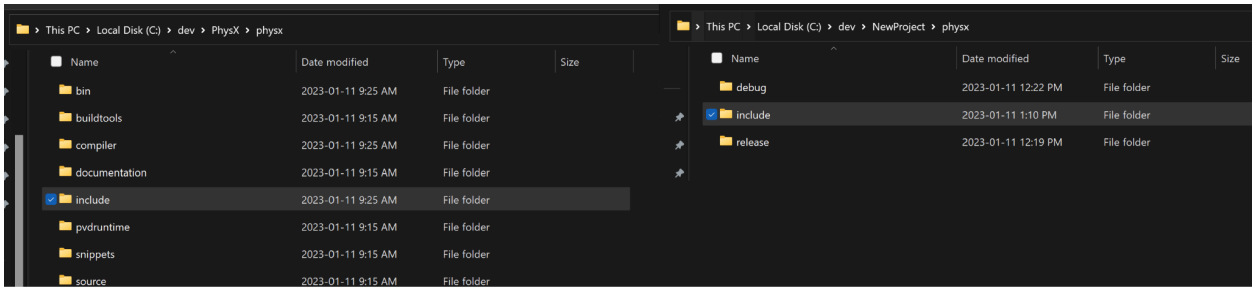
16. Hit Ok to save this. The new relative path should now appear in the Additional Library Directories area.



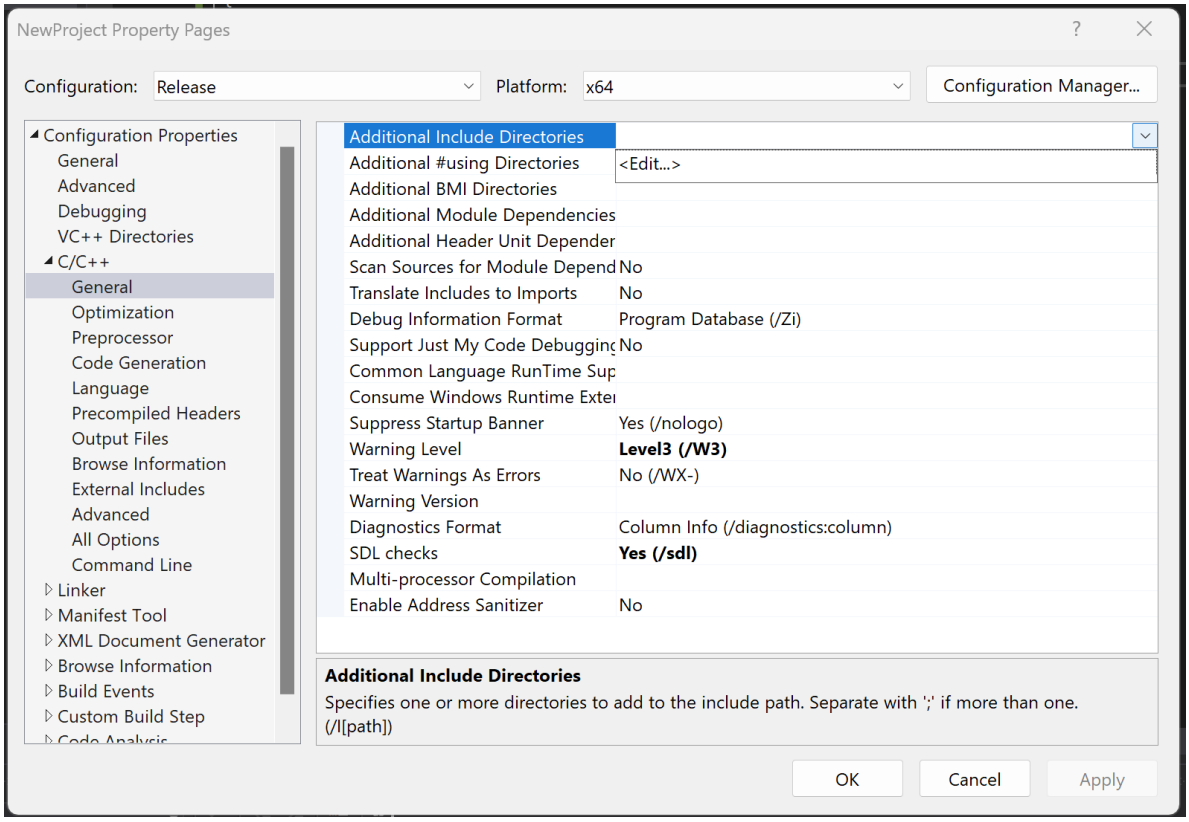
17. Now click on the Input header in the left-hand menu of the properties menu and add the names of the static .lib files we copied to our project into the Additional Dependencies area delimited by a semicolon (leave the ones already there). Then hit Apply to save.



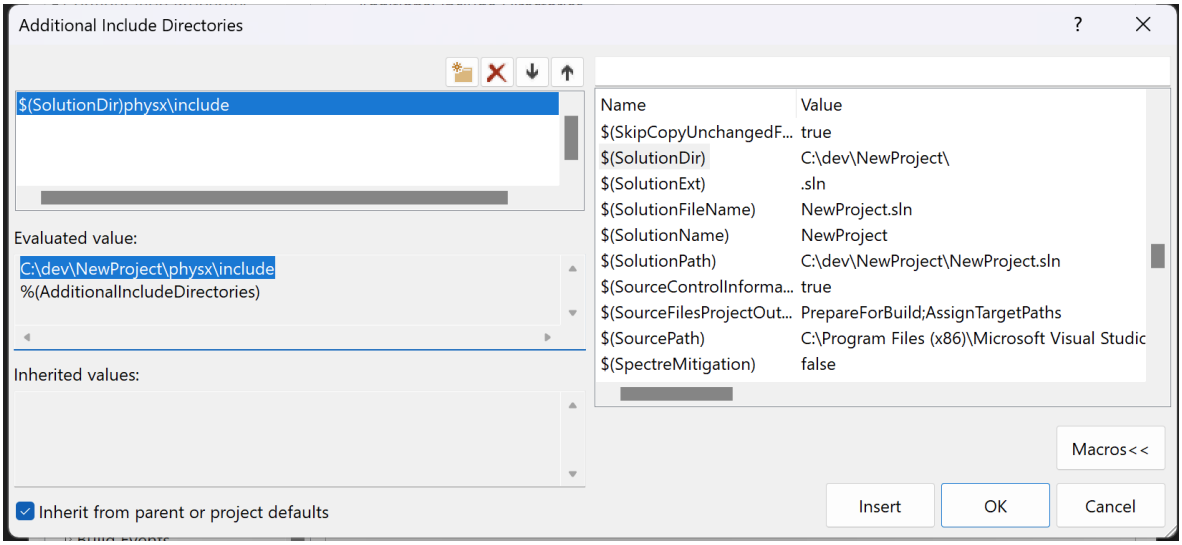
18. Head back to the PhysX project in the File Explorer and in the physx/ directory copy the entire include directory into your project where you made the debug and release folders.



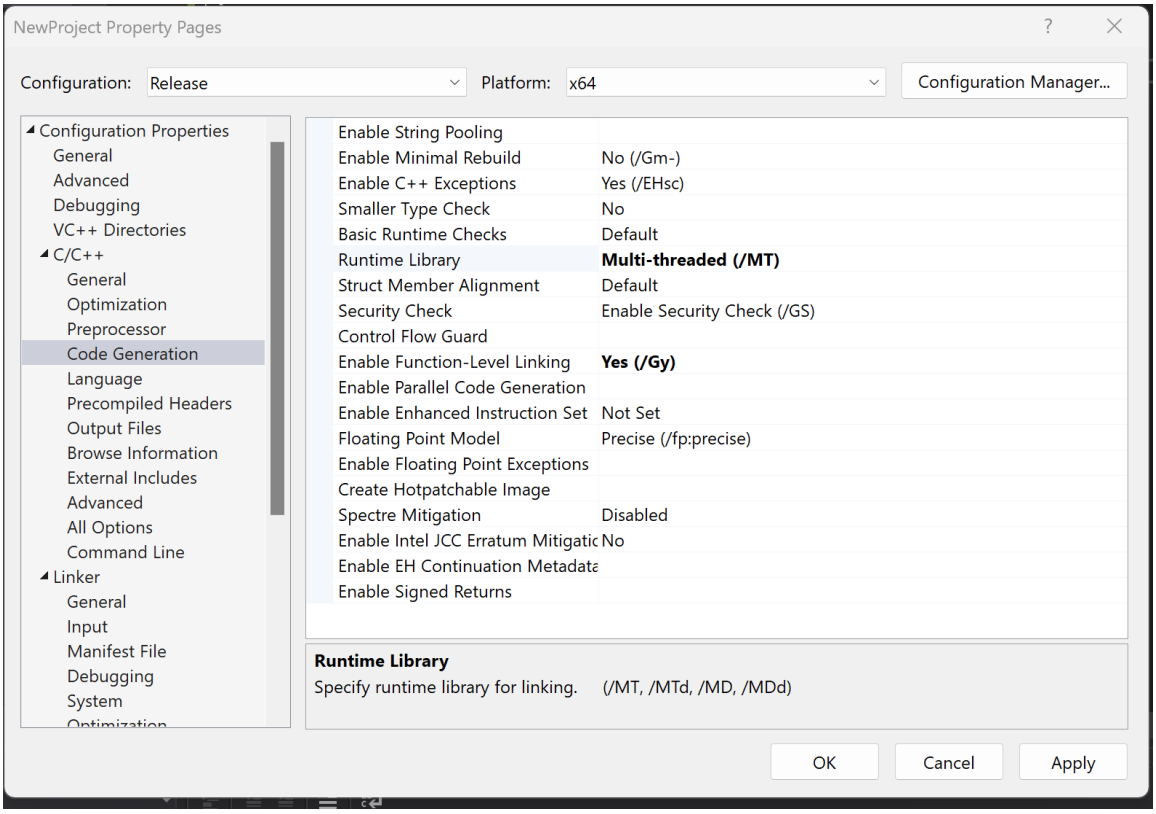
19. Back in Visual Studio, open your project's properties menu again and click on the C/C++ tab in the left-hand menu and go to the general header. Edit the Additional Include Directories as we did the libraries directory before.



20. As before, use the $(SolutionDir) macro to create a relative file path to the include directory you just copied into your project.



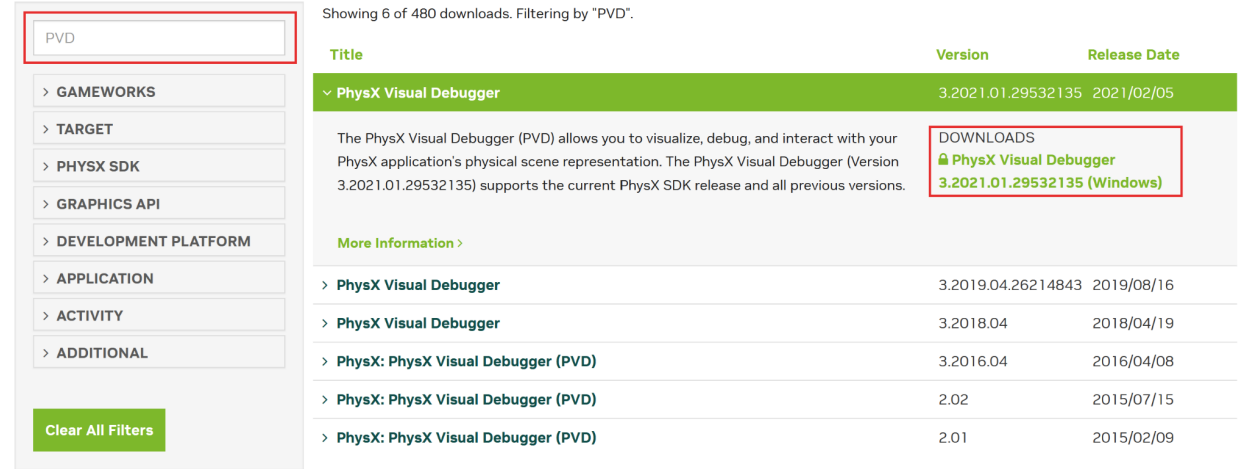
21. Finally, in the properties menu navigate to the Code Generation header under the C/C++ tab in the left-hand menu and find the Runtime Library selection. Set this to Multi-threaded (/MT) for the release configuration and Multi-threaded Debug (/MTd) for the debug configuration of your project. NOTE: If you have previously built and included libraries in your project you may need to ensure they are built using the same Runtime Library configuration to avoid errors.



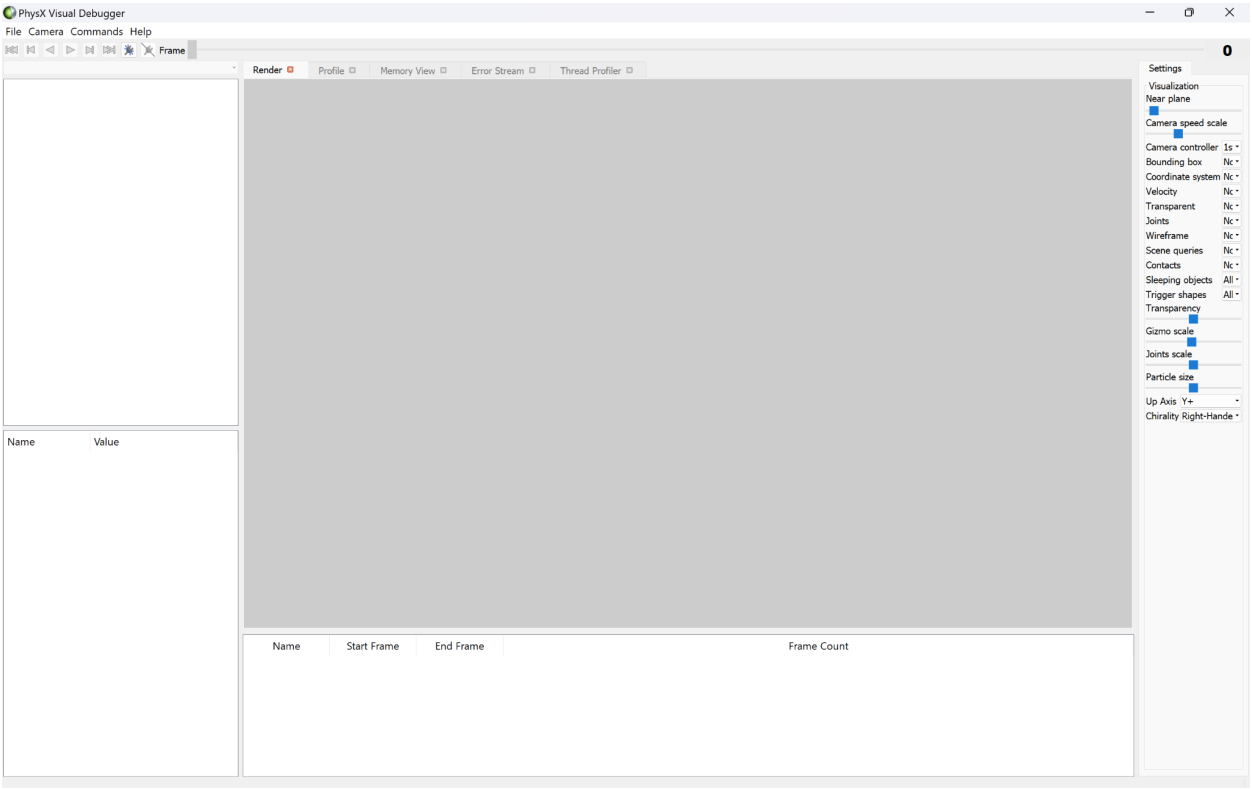
4. Getting Started with PhysX

1. Install the PhysX Visual Debugger application. You will need to make an NVIDIA developer account to access the download. Once you've made an account, click on the download link at the above website. Then search for PVD on the next page and download the newest version available and install.

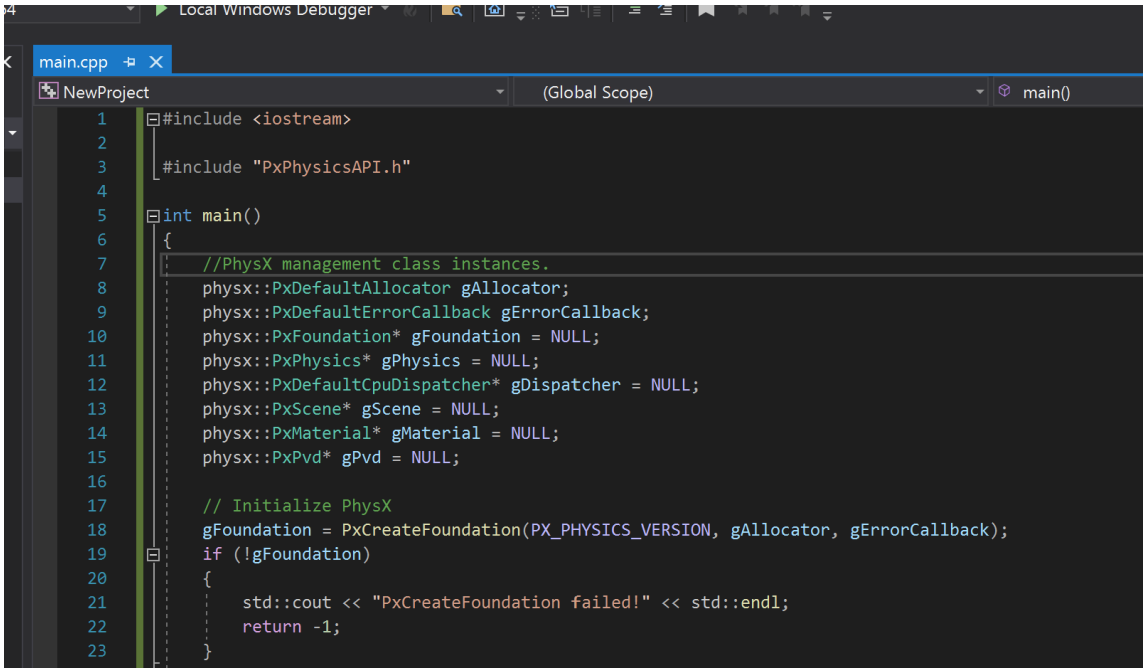
Gameworks Download Center



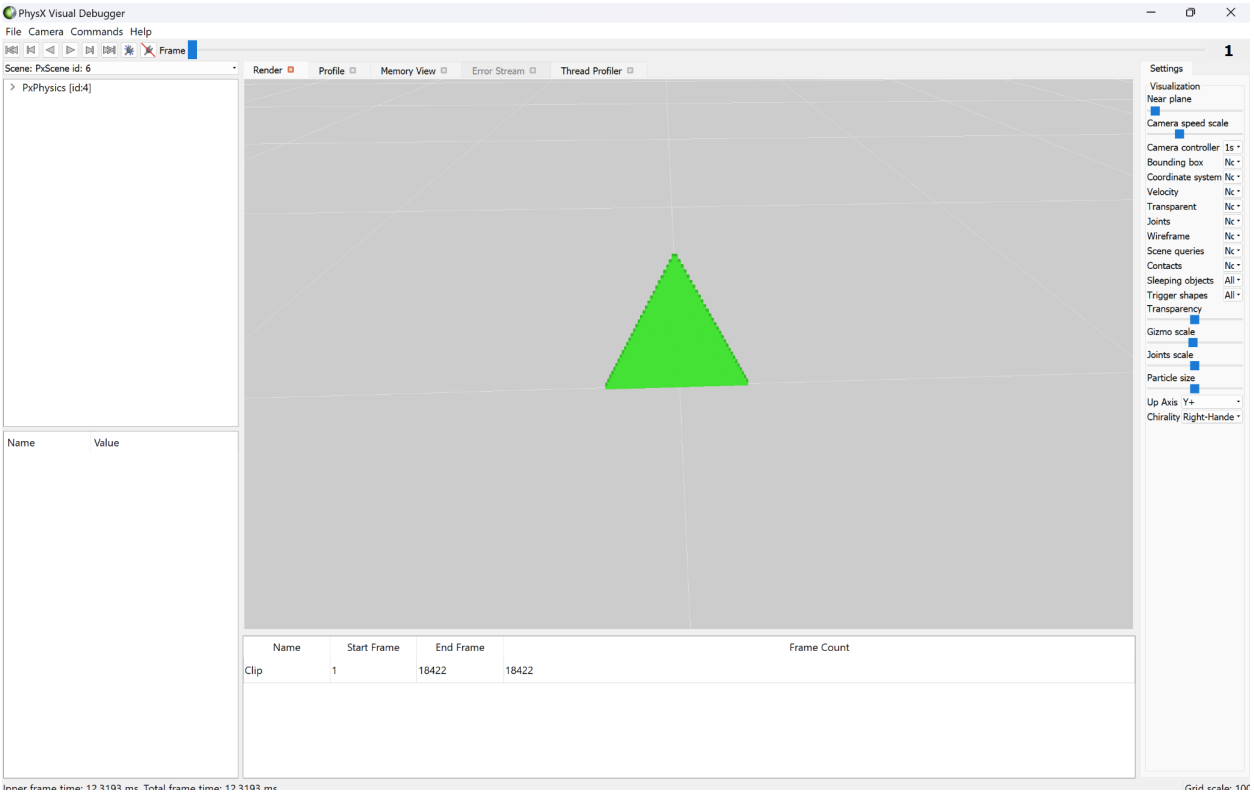
2. Once the PVD is installed, run it to see an application like this:



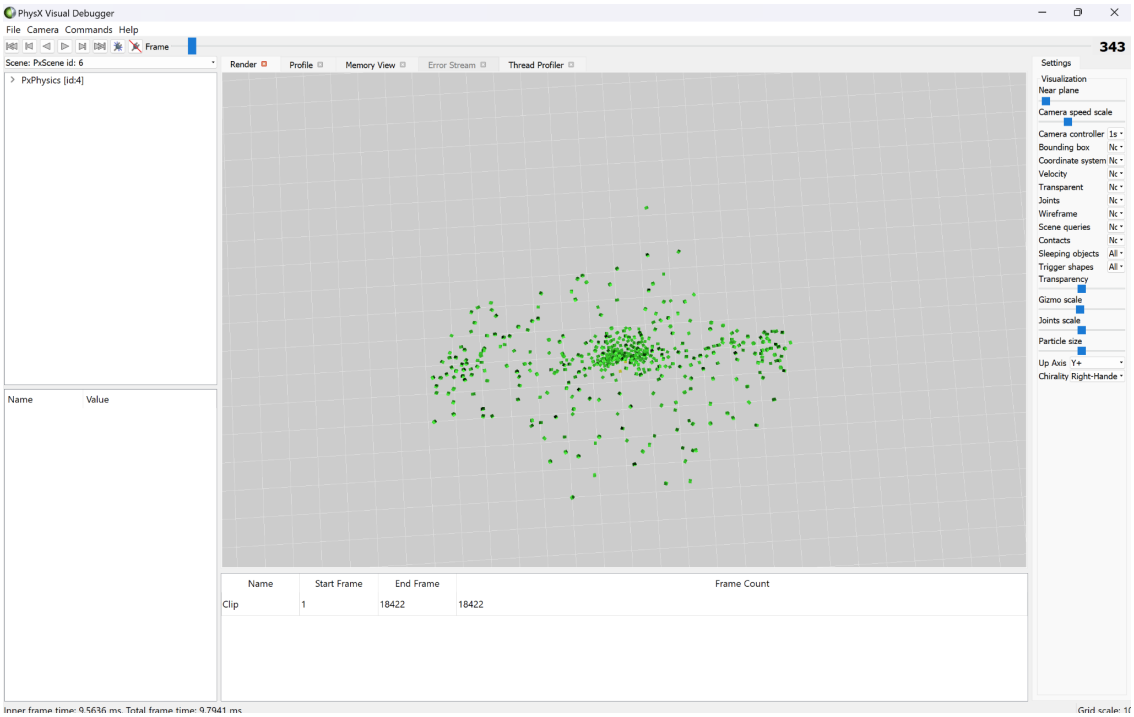
3. To make sure PhysX is working in our project we'll copy the provided main.cpp file into our project and attempt to run it while PVD is also open.



4. If everything is working as intended you should see a pyramid of boxes (in some state) in the PVD display.



5. The boxes forming the pyramid have rigid bodies and the force of gravity should be exerted on them during the simulation causing them to fall to the ground plane and spill apart:



6. At the top of the PVD window there are playback controls you can use once you stop the application and there is a list of objects including some of their parameters in the left-hand menu: