OpenCV2X - Configuration

Requirements:

- OMNeT++ 5.4 (or greater)
- Clion IDE → https://www.jetbrains.com/clion/

Configuring the OMNeT++:

- Follow the usual procedure for OMNeT++ install https://doc.omnetpp.org/omnetpp/InstallGuide.pdf
- Ensure you build it with `Make WITH_OSGEARTH=no` never got it working with it on.

Setting up the IDE:

- Install CLion following the below link will describe the installation do not use snap:

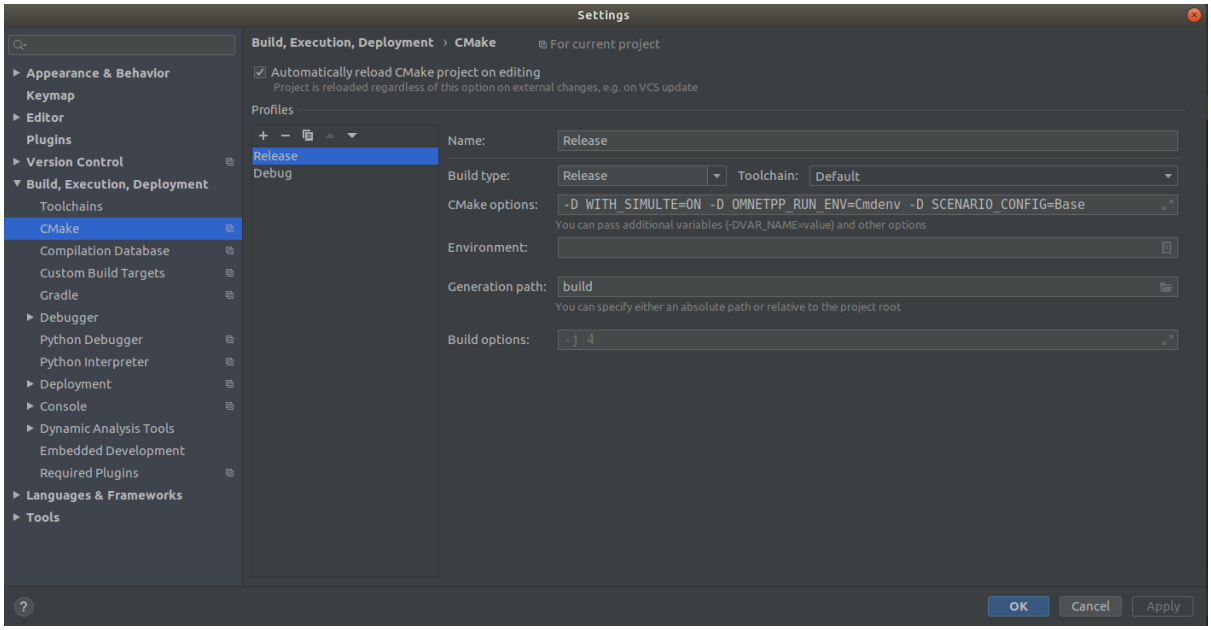
  https://www.jetbrains.com/help/clion/installation-guide.html

  Using snap means the environment variables won't be available and this makes things much more difficult.
- Pull the most recent version of OpenCV2X
- Make all the individual projects from the root of the project:
  - `cd extern/inet & make WITH_OSGEARTH=no MODE=debug`
  - `cd extern/veins & ./configure & make`
  - `make simulte`
  - `make vanetza`

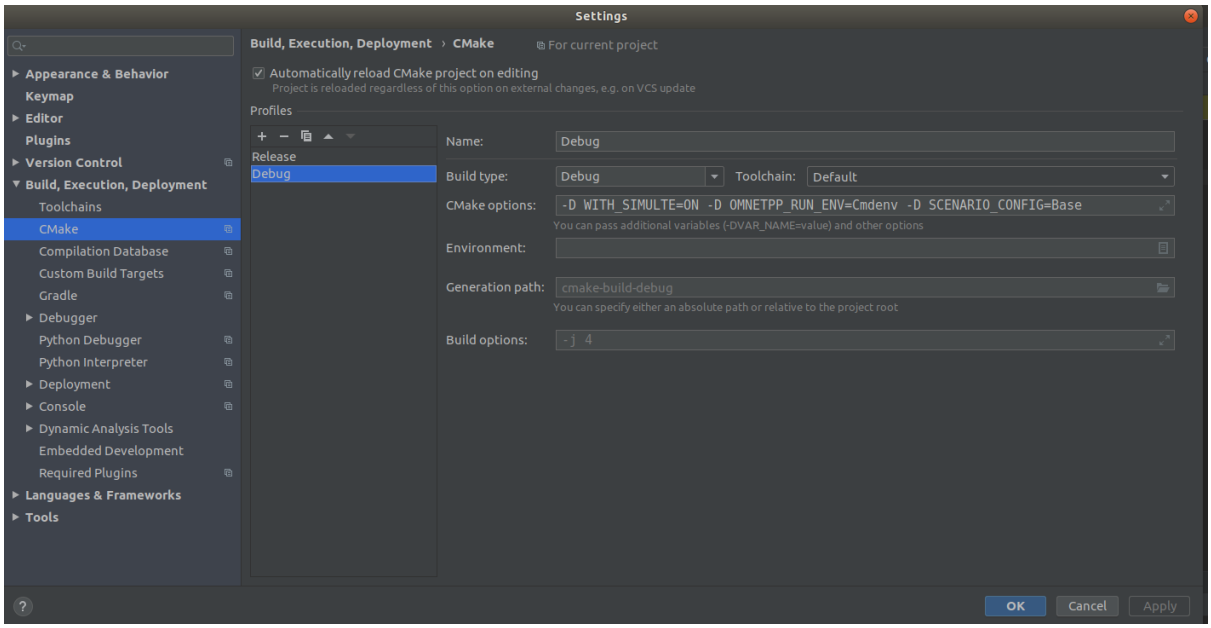
At this point the project should be ready to go now go to `File | Settings | Build, Execution, Deployment | CMake` here you will add a `Debug` and `Release` option

The configuration of each should look like the below images.

Release



Debug



Release configurations:

This is less concerning using the green hammer next to the top right configurations, this will automatically build it the correct way without need to define the configuration parts.

Debug configurations:

While this is complex it only needs to be done once and your system will be ready to run the debug runs and give full breakpoint functionality.

The below are what each field should show for the debug configuration of Mode 4:

Target: `debug_simulte-cars`

Executable: `opp_run_dbg`

This you will find in your OMNeT++ bin folder here is mine as an example:

`/home/brian/omnetpp-5.5.1/bin/opp_run_dbg`

Below is the Program Arguments Section:

```
-u Cmdenv
-c Base
-n /home/brian/git_repos/OpenCV2X/src/artery:/home/brian/git_repos/OpenCV2X/src/traci:/home/brian/git_repos/OpenCV2X/extern/vei
-l /home/brian/git_repos/OpenCV2X/extern/inet/out/gcc-debug/src/libINET_dbg.so
-l /home/brian/git_repos/OpenCV2X/extern/simulte/out/gcc-debug/src/liblte_dbg.so
-l /home/brian/git_repos/OpenCV2X/cmake-build-debug/scenarios/highway-police/libartery_police.so
-l /home/brian/git_repos/OpenCV2X/cmake-build-debug/src/artery/envmod/libartery_envmod.so
-l /home/brian/git_repos/OpenCV2X/cmake-build-debug/src/artery/storyboard/libartery_storyboard.so
-l /home/brian/git_repos/OpenCV2X/cmake-build-debug/src/artery/libartery_core.so
-l /home/brian/git_repos/OpenCV2X/extern/veins/out/gcc-debug/src/libveins_dbg.so
 omnetpp.ini
```

Working Directory: `/home/brian/git_repos/OpenCV2X/scenarios/cars`

The above is the example for my version and will need to change but this is it in essence.

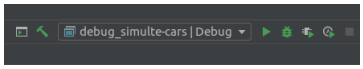
At this point in the top right corner you should be able to select `debug_simulte-cars | Debug` and then click the `bug` symbol to run in debug mode.

Under the file `OpenCV2X/extern/simulte/src/Makefile` change line `19` from `-lINET` to `-lINET_dbg`

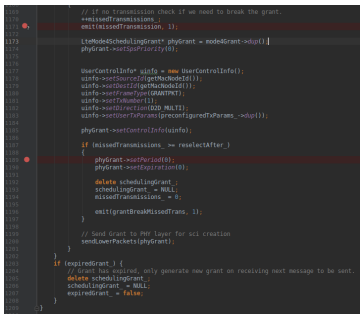
```
# Additional libraries (-L, -l options)
LIBS = $(LDFLAG_LIBPATH)$(INET_PROJ)/out/$(CONFIGNAME)/src  -lINET_dbg
```

At this point you should be able to run the debug configurations of the simulations.

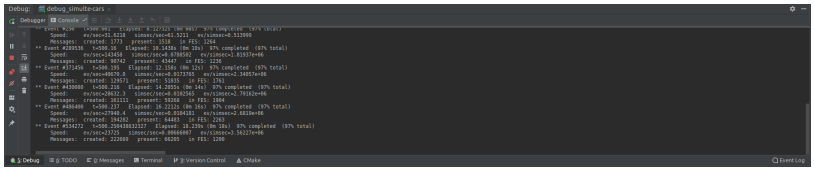
Launching debug simulations, adding breakpoints and debugging.



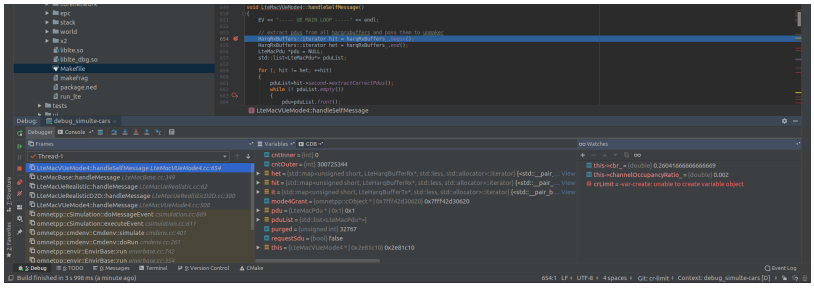
The above is what your launch window should look like with the specific options being set, we use the `debug` option of the `bug` next to the `play` button to launch or debug simulations.



Above shows some breakpoints in the code, when we want to add a breakpoint we simply left click next to the line numbers. A `red circle` in this case indicates a breakpoint.



Above shows what happens when a debug simulation begins the `debug_simulte-cars` option should launch a window at the bottom of the screen, the console will show you how the simulation is running and the debugger window which is shown below will show you things such as variables, function calls and other useful debug information.



The above as described previously is the debug window, in this case we have run into a breakpoint and can now debug the simulation. Most of this is self explanatory.