Getting Setup with MOOS

Hello, I’m Dean, the following is a recount of the setup I did to get MOOS working on a Windows PC. I’ll also include my understanding of the architecture and a handy list of resources for more information.

I had major difficulty with getting a GUI on Ubuntu with my computer, I hope this guide simplifies it.

You might also choose to change operating system to Linux or dual boot, it’s the platform many developers use because it’s open source, lightweight and used on many platforms.

Skip to step 5 if you’re running Linux or Ubuntu already :D

# Setup

1. Install WSL 2 program:
   1. Use <https://docs.microsoft.com/en-us/windows/wsl/wsl2-kernel>
   2. This is mainly required for as a pre-requisite for the following program. (It creates a virtual directory on your computer network, on a blank address bar type “[\\wsl$](file:///\\wsl$)” no quotation marks)
2. Install Ubuntu app:
   1. Download and install from: <https://sourceforge.net/projects/vcxsrv/>
   2. This will be your main Ubuntu command line (with no GUI).
   3. This will utilize and install itself within the “[\\wsl$](file:///\\wsl$)” virtual directory.
3. Install XLaunch:
   1. This will act as your main GUI terminal, allowing you to visualize the MOOS simulations.
   2. Open the program and select the following options:
      1. “One large window”, display number -1, NEXT ->
      2. “Start no client”, NEXT ->
      3. Check “Disable access control” and leave other defaults, NEXT->
      4. “Save configuration”, preferable to desktop, you’ll use this to open the program from now on.
4. Open the Ubuntu app:
   1. Install the following packages in order.
      1. sudo apt install git subversion
      2. sudo apt install build-essential
      3. sudo apt install cmake
      4. sudo apt install libfltk1.3-dev
      5. sudo apt install libtiff5-dev
      6. sudo apt-get update && sudo apt-get upgrade
      7. sudo apt-get install tasksel
      8. sudo apt-get install xfce4 slim
   2. Update the “.bashrc” script to connect to the GUI on startup
      1. Ensure you’re in the home directory by typing “cd”.
      2. Type “nano .bashrc”
      3. Use the arrow keys to navigate down to the last line of the document
      4. Type the following line: export DISPLAY=”`grep nameserver /etc/resolv.conf | sed ‘s/nameserver //’`:0”
         1. NOTE: character sensitive, ` = (not an apostrophe).
      5. Exit and save by pressing: ctrl + x (then y to save)
      6. Make sure the XLaunch window is open from before
      7. Type source .bashrc to reload the file
         1. The Ubuntu app should now be connected to the GUI
   3. Whenever you want to open it from now on, just open the XLaunch window first, then the Ubuntu app and you’ll automatically be setup :D
5. Installing MOOS
   1. From the home directory, type: svn co https://oceanai.mit.edu/svn/moos-ivp-aro/trunk/ moos-ivp
   2. Change directory to the new folder: cd moos-ivp
   3. sudo ./build-moos.sh install
   4. sudo ./build-ivp.sh install
6. Starting your first simulation up
   1. Navigate to a mission directory
      1. cd moos-ivp/ivp/missions/s1\_alpha
      2. ./launch.sh

# Architecture

Below is my understanding of MOOS architecture so far, it may be a good starting point but you’ll no doubt surpass me quickly.

* MOOS framework is developed by Oxford.
  + <https://ori.ox.ac.uk/moos/>
* MOOS IvP which utilises and controls MOOS developed by MIT
  + <https://oceanai.mit.edu/moos-ivp>
* MOOS – Mission Oriented Operating Suite
  + MOOSDB - MOOS Database, it simulates the parameters of a vessel, AUV or the shoreside base.
  + MOOS Community – consists of a MOOSDB file, functionalities implemented through applications and behaviours.
    - The network is setup in a star topology, MOOSDBs communicate through to apps, the apps do NOT communicate to each other.
    - For more information:
  + A simulation can use multiple MOOS Communities
* Applications are where most MOOS development gets done.
  + Sets up a priority list of actions to decide the actions of the vessel.
* Behaviours are the fundamental movements and readings the vessel can make, such as moving to waypoint, tracking, returning etc.
  + Many pre-written ones exist in the common library.

# Optimisations

1. Use .bashrc to change directory to an easier file location via going back through the mnt.
   1. /mnt/c/Users/…
   2. You can then open the folder using visual studio and view all the files
2. Using Visual Studio
   1. Colours the code and makes editing easier
   2. To save files from visual studio, the simulation must NOT be running and you need to run the ./clean.sh app to remove log files.

# Resources

1. If you struggled to get GUI running:
   1. <https://techcommunity.microsoft.com/t5/windows-dev-appconsult/running-wsl-gui-apps-on-windows-10/ba-p/1493242>
2. The guides I used to get setup:
   1. <https://github.com/moos-tutorials>
   2. <https://docs.microsoft.com/en-gb/windows/wsl/install-win10>
   3. <https://phoenixnap.com/kb/how-to-install-a-gui-on-ubuntu>
3. The two main hubs of documentation:
   1. <https://ori.ox.ac.uk/moos/>
   2. <https://oceanai.mit.edu/moos-ivp>