Coursera CARLA Setup for Ubuntu

Prerequisites

Hardware

Recommended hardware specifications (from the Unreal Engine 4 Wiki)

- Quad-core Intel or AMD processor, 2.5 GHz or faster
- NVIDIA GeForce 470 GTX or AMD Radeon 6870 HD series card or higher
- 8 GB RAM
- ~10GB of hard drive space for the simulator setup

Note that the above are the **recommended** hardware requirements.

A computer with lower hardware specifications, including systems with integrated graphics, may also be able to run the CARLA simulator with slower performance.

Software

Ubuntu

The CARLA loader requires **Ubuntu 16.04 or later** to run.

Firewall

CARLA requires networking enabled with the firewall allowing access to the CARLA loader, and by default **port 2000, 2001 and 2002** (TCP and UDP) available on the network. Generally Ubuntu's built-in firewall is disabled by default, so you shouldn't need to worry about the firewall access for the CARLA Simulator.

You can also check this by running the command in a terminal:

```
$ sudo ufw status
```

, and confirm that the response is `Status: inactive`

If your network does not provide access to port 2000, you may change which ports are used at a later stage of the setup, just make note of the option here in the FAQ section when trying to run CARLA in server-client mode.

Graphics Card Drivers

Update to the latest video card drivers for your system to avoid graphics issues.

Open GL 3.3 or above is also required for the CARLA binary to work. Don't worry about checking these requirements - they are validated during the loading of the CARLA binary.

Python

The CARLA python client runs on Python 3.5.x or Python 3.6.x (x is any number). **Python 3.7 is not compatible with CARLA**. Note that it is assumed that **pip** is installed along with the installation of Python. In Ubuntu, it is possible to install

multiple versions of Python beside each other, so go ahead and install Python 3.6x, even if you already have Python 3.7x.

The setup guide uses the command `python3` to load all of its python clients. Make sure that the commands `python3` and `pip` exists in `/usr/bin`, so they are readily accessible via terminal.

If you have a more recent version of Python and the `python3` command does not give the correct version (3.5.x or 3.6.x), you can replace all `python3` commands with `python3.5` or `python3.6` for the remainder of the setup, which loads either Python 3.5 or Python 3.6, respectively. The `python3.5` or `python3.6` files should be located in `/usr/bin` through default installations of Python 3.5 or Python 3.6.

To check that `python3` points to the correct version, run the following bash command in terminal:

```
$ python3 --version
```

It should return `Python 3.5.x` or `Python 3.6.x`, as these are the versions supported by this version of CARLA.

Check whether pip is installed for Python 3.5 or Python 3.6:

```
$ python3 -m pip --version
```

It should return with the pip version, as well as the Python version that it points to (in the brackets). For example: `pip 8.1.1 from ... (python 3.5)`, or something similar. Make sure that it points to the correct Python version: `(python 3.5)` or `(python 3.6)`

When both python3 and pip3 are available, try to install a conventional package for your current user (you will need to be connected to the internet for this to work):

```
$ pip3 install numpy --user
```

If you get an error that is similar to

```
pip is configured with locations that require TLS/SSL, however the
ssl module in Python is not available.,
```

, then try to install the dependencies below

```
$ sudo apt-get install libreadline-gplv2-dev libncursesw5-dev
libssl-dev libsqlite3-dev tk-dev libgdbm-dev libc6-dev libbz2-dev
```

and re-install Python. If the issue with pip still exists, please refer to the Discussion forums for help.

Preparing the CARLA Simulator

Download and Extract the CARLA Simulator [1]

1. Download the CARLA simulator (`CarlaUE4Ubuntu.tar.gz`) found in the reading page. Note that this may take a while as the simulator file is several gigabytes in size.

2. Extract the contents of `CarlaUE4Ubuntu.tar.gz` to any working directory. The extraction will create a folder named `CarlaSimulator` in the working directory, which hosts the CARLA server and client files required for the projects.

   The guide assumes the simulator is extracted to `$HOME/opt/CarlaSimulator`.

   Continue with this step for extraction instructions, otherwise skip to the next step.

   Ubuntu GUI method:

   Copy the `CarlaUbuntuUE4.tar.gz` file to the `opt` directory found under the home folder. Create the `opt` directory if it does not exist.

   Right click and extract the file contents into the current (`opt`) directory by clicking `Extract Here...`.

   Terminal method (alternative to using the Ubuntu GUI)

   Run the following commands in a terminal to extract the simulator to `$HOME/opt`:

```
$ cd <path/to/CarlaUE4Ubuntu.tar.gz>
$ mkdir -p $HOME/opt # Ensures the opt directory exists
$ tar -xzf CarlaUE4Ubuntu.tar.gz --directory $HOME/opt
```

[1] The CARLA simulator used here is a **modified binary of the version 0.8.4 CARLA**. There are additional maps included as well as the default vehicle model is modified for the purpose of better demonstrating concepts taught in the courses. While it is encouraged to perform your own compilation and installation of CARLA to learn more about the simulator, the course itself must use the provided binaries to evaluate the course assignments.

Install Python Dependencies for Client

The CARLA Simulator client files requires additional dependencies to be installed, which are detailed inside the `$HOME/opt/CarlaSimulator/requirements.txt` file.

To install these dependencies for your current user, run the following commands in terminal (you will need to be connected to the internet for this to work). Make sure that the version that `python3` points to is the correct version.

```
$ python3 -m pip install -r $HOME/opt/CarlaSimulator/requirements.txt
--user
```

There should be a `Successfully installed ...` or `Requirement already satisfied` message at the end of the installation process when all of the requirements are successfully installed. If there are issues with installing these requirements, please refer to the Discussion forums for help.

Testing the CARLA Simulator

Loading the Simulator with the Default Map

Note: This step must work to continue with the testing process. Please ensure that the Prerequisites section is met and the Preparing the CARLA Simulator steps were followed. If there are still issues, please refer to this section in the FAQ.

CARLA can be launched using the `CarlaUE4.sh` script, and a simple test would be running the following command in terminal. The mouse cursor will disappear when the simulator window is focused on, so press `Alt-Tab` and switch to another application to defocus from the window and reveal the mouse cursor.

```
$ cd $HOME/opt/CarlaSimulator # where the CarlaSimulator is located
$ ./CarlaUE4.sh -windowed -carla-no-networking
```

The `-windowed` argument is used to make sure the simulator is loaded to a window. If the `-windowed` argument is not included, CARLA will run in full-screen mode by default.

The `-carla-no-networking` argument is to ensure that the server-mode is not enabled.

If the CARLA window is too large, you may also add the arguments `-ResX=<width_size_in_pixels>` and `-ResY=<height_size_in_pixels>` to adjust the window width and height, respectively. For example, add `-ResX=640 -ResY=480` to the above `./CarlaUE4.sh` command to resize the window to be 640x480 pixels.

To move the CARLA window, defocus from the CARLA window using `Alt-Tab` and selecting another application to reveal the mouse cursor. Hover the mouse cursor over the top of the window title bar, left-click and drag to move the window around. Another method to move the CARLA window would be to press `Alt-F7` and use the mouse to adjust the window position, then left-click to confirm the new window position. Refer to this site for more information on window resize and moving.

The default map loaded is `Town01` (see below):

To close the CARLA simulation session, press `Alt-F4` in the simulator or `Alt-Tab` to the terminal running CARLA and press `Ctrl-C`.

Use the `A`, `S`, `D`, and `W` keys to manually drive the vehicle around the map.
Use the `Q` key to toggle reverse and forward drive mode.
Use the `P` key to toggle auto-pilot mode (this only works in maps with pre-defined roads, such as the `Town01` and `Town02` maps provided by CARLA)
Refer to the CARLA keyboard input documentation for other commands.

If there are issues with loading the Simulator, please refer to this section in the FAQ.

Loading the Simulator with the Race Track Map

The final project for Course 1 uses a race track map that is pre-packaged with the simulator.

Run the following command in terminal to load the race track map in windowed mode:

```
$ cd $HOME/opt/CarlaSimulator # where the CarlaSimulator is located
$ ./CarlaUE4.sh /Game/Maps/RaceTrack -windowed -carla-no-networking
```

Which should load the race track map (see below).

When loading a map, make sure that it is the **first argument** to follow `./CarlaUE4.sh`, i.e.,

```
$ ./CarlaUE4.sh [ /Game/Maps/<map_name> ] [ <args> ]
```

Use the `A`, `S`, `D`, and `W` keys to manually drive the vehicle around the map.
Use the `Q` key to toggle reverse and forward drive mode.
Refer to the CARLA keyboard input documentation for other commands.

If there are issues with loading the Simulator, please refer to this section in the FAQ.

Loading the Simulator with a Fixed Time-Step

To run the simulator with a fixed time-step of 20 frames-per-second (fps), or 50 milliseconds per simulation frame, use the following command in a terminal:

```
$ cd $HOME/opt/CarlaSimulator # where the CarlaSimulator is located
$ ./CarlaUE4.sh /Game/Maps/RaceTrack -windowed -carla-no-networking
-benchmark -fps=20
```

Which loads the race track map at 20 fps (see below):

Take note that the Simulation Step is now fixed to **50ms (50 millisecond)** as intended.

The `-benchmark -fps=<frames_per_second>` argument is used to fix the simulator to a given frame-per-second rate. Make sure that `-benchmark` always precedes `-fps` and with a space in between the two.

Also note that the simulation flows much slower at this rate, this is because **there is a shorter in-game time interval in between each frame**. By increasing the frames-per-second value (e.x. from `-fps=20` to `-fps=40`) the frames per second is doubled and the simulation should slow down about twice as much. Please refer to this documentation for more information between variable (default) and fixed time-steps.

If there are issues with loading the Simulator, please refer to this section in the FAQ.

Some Other Useful CARLA Simulator Information

The `CarlaUE4.sh` script supports several other command-line options. The following table shows some useful arguments (in addition to the ones already shown in the previous examples):

Option	Description
`/Game/Maps/<map_name>`	**Must be the first argument.** Loads a prepackaged map inside the simulator. The useful maps to replace `<map_name>` are: • `Town01` - CARLA's first town map (default) • `Town02` - CARLA's second town map • `FlatEarth` - A large flat area bounded by walls • `RaceTrack` - A race track course The `Town01` and `Town02` maps are very detailed environments with other vehicles and pedestrians, and are good for seeing the full environment that CARLA has to offer. The `FlatEarth` and `RaceTrack` maps are useful for testing and evaluating algorithms.
`-windowed`	Loads the CARLA simulator in a window (without this argument, the CARLA simulator will load to full-screen).
`-ResX=<pixel_width>`	Sets the rendered image width to be `<pixel_width>` (in pixels). Decreasing this will increase rendering speed.
`-ResY=<pixel_height>`	Sets the rendered image height to be `<pixel_height>` (in pixels). Decreasing this will increase rendering speed.
`-benchmark -fps=<fps>`	Sets a fixed time-step for the simulator, by changing the value of `<fps>` to a desired frame-per-second in game-time for the simulator (min = 10). Ensure that `-benchmark` always precedes `-fps`.
`-carla-no-networking`	Disables networking (server-mode) for the CARLA simulator.
`-carla-server`	Enable server mode for the CARLA simulator, which will wait until a client connects to it. **Carla will need access to the network and bypass any firewall.**
`-carla-no-hud`	Do not display the HUD by default in the simulator.
`-carla-world-port=N`	**(only active during server mode)** Sets the world-port to be `N` (default: `2000`). This allows the client to connect to the server from a different port.
`-quality-level=<level>`	Sets the rendering quality of the simulator, which is `Low` or `Epic`. **Only use `Epic` for higher end computers**.

Testing CARLA in Server-Client Mode

If you have made it this far - great work! Now we will use a client to pass commands and receive data from the CARLA simulator. This server-client interaction is used to develop and test algorithms used in the Coursera projects.

Note: Enabling the server mode forces the simulator to wait for a Python client to connect before the simulator starts. This means that nothing will appear on the window or the simulator will appear frozen, which is expected until the Python client connects. Make sure to press `Alt-Tab` to defocus from the simulator (as it is still idling until a client connects), so you can run the Python client in a separate terminal.

First, lets enable server mode for the CARLA simulator by running the following command in a terminal (**Make sure there are no firewalls that are blocking CARLA UE4 from the network - by default the built-in Ubuntu firewall is disabled so you shouldn't need to worry about this**):

```
$ cd $HOME/opt/CarlaSimulator # where the CarlaSimulator is located
$ ./CarlaUE4.sh /Game/Maps/RaceTrack -windowed -carla-server -benchmark
-fps=20
```

Use any other arguments if desired (for example, `-ResX` and `-ResY` to resize the window). Since there is no client that is running yet, the terminal should show that the server is waiting for the client to connect:

```
9.48:454][  2]LogCarlaServer: Waiting for the client to connect...
```

Then, in another terminal, run the example Python client (`manual_control.py`):

```
$ cd $HOME/opt/CarlaSimulator/PythonClient
$ python3 manual_control.py
```

If the Python client successfully connects, a new **pygame window** should appear (see below). If there is an issue with running the client, refer to this section for troubleshooting.

Note that in the server-client mode, the CARLA simulator window **will not receive** keyboard inputs (like in the previous examples without `-carla-server` enabled). Try clicking on the **pygame window** and manually drive the vehicle around, then try to do the same for the simulator.

In this particular example, since the Python Client (`manual_control.py`, which loads the **pygame window**) is actively listening for the keyboard commands and passing control commands to the CARLA simulator server. It is also receiving the CARLA server information and relaying information (such as the depth map on the bottom left corner and the semantic segmentation on the bottom right).

To close the Python Client, press `Ctrl-C` on the terminal which the Python Client was ran from. You can keep the CARLA server up and load other Python Clients as well as long as it is one client at a time - the server will run whenever a new client connects to it.

Try to run the same exercise using the `Town01` and `Town02` maps. What did you notice that was different?

Side note: If your computer has a lower performance specification, try starting the client script first and then the server may make the execution faster.

If there are issues with loading the Simulator, please refer to this section in the FAQ. If there are issues with running any Python client, please follow this section in the FAQs for troubleshooting.

Testing Other CARLA Python Client Examples

CARLA Python clients can be found in `$HOME/opt/CarlaSimulator/PythonClient`.

CARLA by default provides several example clients that are detailed in the link. These example client scripts are useful for demonstrating various features that CARLA has to offer. More on these features can be found under the Measurements and Cameras and Sensors sections. Feel free to examine the client script code to learn more about how to program client scripts to communicate with the CARLA server (simulator).

Frequently Asked Questions (FAQ)

Can I run the simulator in a virtual machine (VM)?

The simulator will not run on a VM using VirtualBox as VirtualBox only supports OpenGL 2, whereas CARLA requires Open GL 3.3 or above. Other issues for VMs might be due to the lack of the necessary hardware virtualizations required to run the Unreal Engine. We have not tested the simulator with a different virtual machine platform.

The simulator freezes or crashes as soon as I start it! What do I do?

Make sure you have all of the prerequisites prepared!

First, make sure that you have the proper hardware and software requirements for the simulator to work, and also followed the simulator setup instructions. In particular, make sure you are using a **most recent driver for your video card and have OpenGL 3.3 or above installed**.

Try to run CARLA without server mode enabled

Once the above prerequisites are completed, and run the simulator with this command:

```
$ cd $HOME/opt/CarlaSimulator # where the CarlaSimulator is located
$ ./CarlaUE4.sh -carla-no-networking
```

If you are able to run the CARLA without server mode enabled, then proceed to the next section.

If you are **unable** to run CARLA without server mode enabled, first double check to see if it is not a permissions issue.

If CARLA is still unable to execute, please navigate to: `$HOME/opt/CarlaSimulator/CarlaUE4/Saved/Logs` and examine the most recent `CarlaUE4.log` file (should be time-stamped at the time of the latest crash or stall).

If you are experienced with the Unreal Engine, you may be able to debug the simulator yourself; however, most people should refer or post to the Discussion Forums at this stage so that the course team can help you.

Make sure you include the following in the forum post:

- `CarlaUE4.log` file (`$HOME/opt/CarlaSimulator/CarlaUE4/Saved/Logs`)
- Hardware Specifications
- Operating System

It freezes when I am trying the run CARLA in server mode!

Be sure you are able to run CARLA without server mode first.

As mentioned in the Testing CARLA in server-client mode section, running the simulator with the argument `-carla-server` will keep the CARLA simulator idle (frozen) until a client connects to it. If you run CARLA in server mode and things seem to be stuck on screen, make sure to press `Alt-Tab` and defocus from the idling CARLA simulator window to gain access to the mouse cursor. Load the Python Client from another window to connect to the CARLA server and the simulation should commence.

Basic server-client mode test:

Server terminal command:

```
$ cd $HOME/opt/CarlaSimulator # where the CarlaSimulator is located
$ ./CarlaUE4.sh /Game/Maps/RaceTrack -windowed -carla-server
```

Client terminal command:

```
$ cd $HOME/opt/CarlaSimulator/PythonClient
$ python3 manual_control.py
```

If the Carla simulator remains frozen as soon as you enable server mode and try to connect a client to it, then maybe your issue can be resolved by using another port in your network, or maybe double check to ensure the built-in firewall is disabled.

If the Python Client doesn't load properly, refer here for help.

If there is still an issue with connecting the client to the server, please refer or post to the Discussion Forums so that the course team can help you.

Make sure you include the following in the forum post:

- `CarlaUE4.log` file (`$HOME/opt/CarlaSimulator/CarlaUE4/Saved/Logs`)
- Python Client output
- Hardware Specifications
- Operating System

Running a Python client produces issues.

Please double check the Python version check as well as the dependencies install guide and ensure that these are properly followed.

You may have accidentally installed the dependencies for a different version of Python. The simulator uses Python 3.5 or Python 3.6; if you installed the dependencies for a later version, it will be unable to find them. Please make sure that the dependencies are installed with the required version of Python.

If the dependencies are installed with the proper version of Python, the dependency may be corrupted and might need a reinstall. Try the following commands:

```
$ python3 -m pip uninstall <dependency_name>
$ cd $HOME/opt/CarlaSimulator # where the CarlaSimulator is located
$ python3 -m pip install -r requirements.txt --user
```

, and see if the issue is fixed.

If both Python client and CARLA Simulator both seems to run successfully, try to use another port from the network for your server-client connection or determine that the default Ubuntu firewall is disabled via the command in terminal:

```
$ sudo ufw status
```

, and confirm that the response is `Status: inactive`

If the issue still remains, please search the Discussion Forums for a possible solution, or post there if the solution is not found.

Make sure you include the following in the forum post:

- Hardware Specifications
- Operating System
- Output from `python3 --version`
- Output from the Python client crash (from terminal)

How do I use another port for the server-client mode.

To use another port (besides the default port of 2000), use the argument `-carla-world-port=N` for the CARLA Simulator script and `--port=N` for the Python client. Make sure to set the port number `N` to be the same for both the server and client.

Example:

Server side:

```
$ cd $HOME/opt/CarlaSimulator # where the CarlaSimulator is located
$ ./CarlaUE4.sh -carla-server -carla-world-port=2003
```

Client side:

```
$ cd $HOME/opt/CarlaSimulator/PythonClient
$ python3 manual_control.py --port=2003
```

The `python3` command did not give the correct version, what should I do?

This issue is likely due to the Python version installed in your computer is a more recent one (3.7+). First ensure that one of Python 3.5.x or Python 3.6.x is installed to your Windows computer. Make sure to check the option "Add Python to environment variables" during the custom installation process.

If you installed Python 3.5, replace all of the `python3` commands to `python3.5` for the setup. First, determine that `python3.5` is located in `/usr/bin` and version check by using the commands

```
$ python3.5 --version
$ python3.5 -m pip --version
```

, and making sure that both commands points to Python 3.5.

If you installed Python 3.6, replace all of the `python3` commands to `python3.6` for the setup. First, determine that `python3.6` is located in `/usr/bin` and version check by using the commands

```
$ python3.6 --version
$ python3.6 -m pip --version
```

, and making sure that both commands points to Python 3.6.

Similarly, the `pythonX.X` command can be used to point to other Python versions by typing:

```
$ python<version_number>
```

I am getting permissions issues when trying to run the CARLA simulator, how do I go about fixing this?

If you are trying to run `CarlaUE4.sh` and it produces `-bash: ./CarlaUE4.sh: Permission denied`, try to add executable permissions to `CarlaUE4.sh` and `CarlaUE4/Binaries/Linux/CarlaUE4`:

```
$ cd $HOME/opt/CarlaSimulator
$ sudo chmod +x CarlaUE4.sh
$ sudo chmod +x CarlaUE4/Binaries/Linux/CarlaUE4
```

And see if the CARLA simulator runs properly.

If it still produces permissions issues, ensure that the ownership of `CarlaSimulator` (and its subfolders) are assigned to your current user:

```
$ cd $HOME/opt
$ sudo chown -R $USER:$USER CarlaSimulator
```

See if this resolves the issue.

If there are still issues, please refer or post in the Discussion forums for further assistance.