

How to Install MuJoCo on Ubuntu

Ben Kolligs

February 4, 2021

1 What is MuJoCo?

MuJoCo (Multi-Joint Dynamics with Contact) is a general purpose physics engine intended to facilitate research and development in various fields including robotics, machine learning, biomechanics and more [3]. It was developed out of the [Movement Control Laboratory](#) at University of Washington by principal developer Emo Todorov.

MuJoCo (original paper [4]) attracted a lot of attention outside the research community when OpenAI started using it as the physics backend of its “OpenAI Gym” environment.

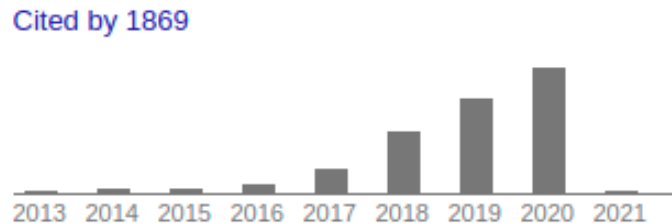


Figure 1: Mujoco [4] citations over time from Google scholar. Note that OpenAI Gym came out in 2016.

MuJoCo hosts several advantages over other physics simulators such as Open Dynamics Engine, PhysX and Bullet. Most notably is that MuJoCo is computationally quicker and more accurate. The interested reader is encouraged to read [1] for a more detailed comparison of these engines.

Originally, MuJoCo was only written in C/C++, and while this offered the user extreme control and speed over the simulation's performance, it made certain functions such as rendering quite complicated. After the advent of Gym, OpenAI in collaboration with UC Berkeley wrote a python wrapper for MuJoCo called [mujoco-py](#). This wrapper simplified the execution of a MuJoCo model and made the engine more accessible to the broader base of python users.

2 Installing MuJoCo

For your assignment, you will be using the python wrapper mujoco-py to play around with the wonderful world of dynamic controls. This means that we will need to install both the C++ library and the python wrapper on your system. For this installation tutorial, we will be using **Ubuntu 18.04 LTS** as our OS. We will also need Python 3.6+ on the system, as well as a corresponding pip installation. At a high level, here's what we need to do:

1. Download the MuJoCo binary library
2. Install mujoco-py
3. Test an example python script

2.1 Download the Library

MuJoCo is a commercial product, which means we need a license file in order for it to work. Luckily the nice people at MuJoCo offer a 30-day trial we shall make use of. The steps are:

1. Go to <https://www.roboti.us/license.html> and fill out the form to request a trial license.

MuJoCo Trial License: 30 days

We invite you to register for a free trial of MuJoCo. Trials are limited to one per user per year. After registration you will receive an email w corresponding to your platform (using the links below) and run it to obtain your Computer id.

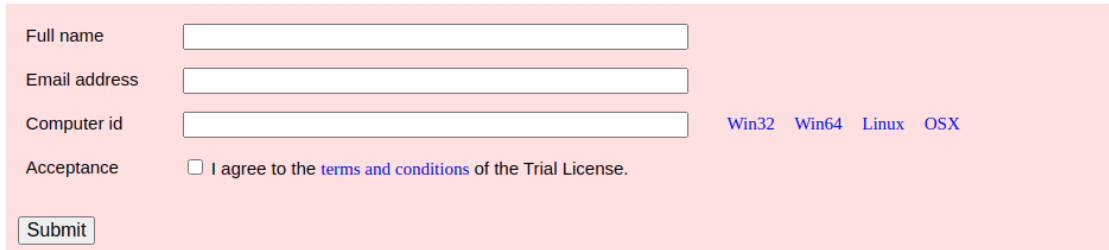


Figure 2: Trial license request page.

2. Download the `getid_linux` executable for Linux and run it on your computer. Note that you may have to add an execute permission with `chmod +x getid_linux`.
3. Download the MuJoCo version 2.0 binaries at <https://www.roboti.us/index.html> and select the `mujoco200 linux` option. Or if you are feeling adventurous here's the direct download link: https://www.roboti.us/download/mujoco200_linux.zip.
4. Unzip this file and place it in the directory `~/.mujoco/mujoco200` and place your license key (`mjkey.txt`) in `~/.mujoco/mujoco200/bin/mjkey.txt`.
5. Test this installation by navigating to `~/.mujoco/mujoco200/bin` and executing `./simulate ../model/humanoid.xml`. You should see an interactive window with a humanoid figure as shown in figure 3.

Assuming the testing command launches, then we are ready to move on to the installation of `mujoco-py`.

2.2 Installing `mujoco-py`

Note that we need to move our `mjkey.txt` file now that we have tested the raw C/C++ library. The easiest way to do this is to just copy the file to the `~/.mujoco` directory. Figure 4 shows what `~/.mujoco` should look like afterwards.

```
1 cp ~/.mujoco/mujoco200/bin/mjkey.txt ~/.mujoco/mjkey.txt
```

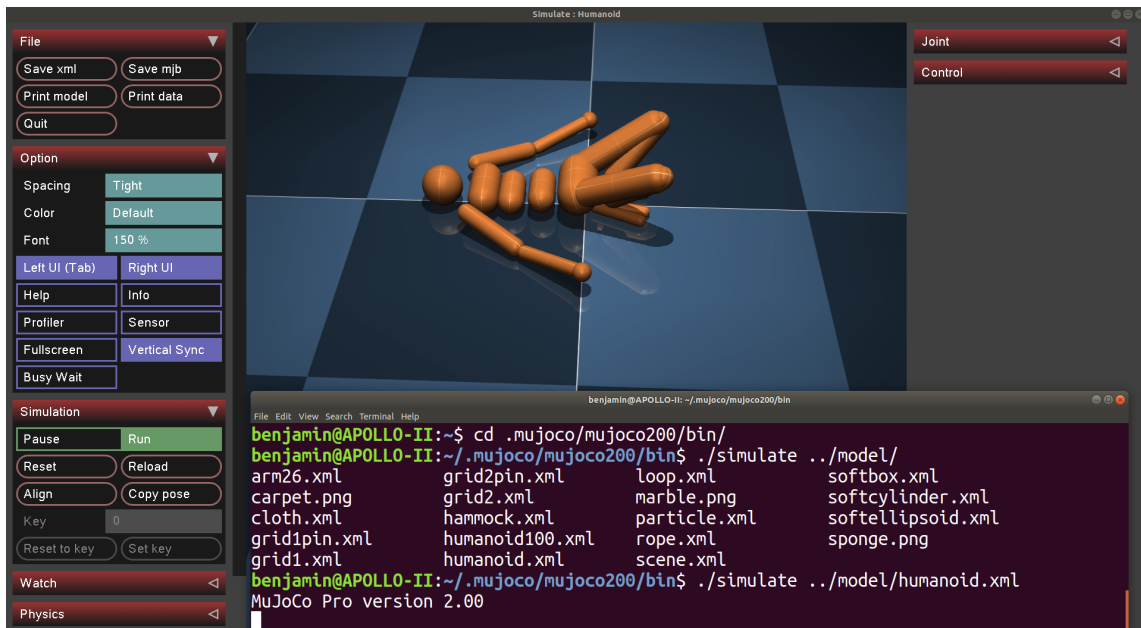


Figure 3: The testing command.

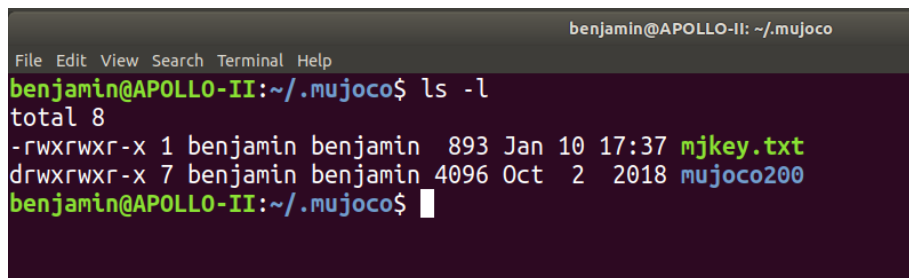


Figure 4: The .mujoco directory

There are several packages we need to make sure are installed on the computer before we can install mujoco-py. First, we need to make sure we have Python 3.6+. You can check this easily by typing `python3` in the command line and looking at the version.

```

benjamin@APOLLO-II:~/mujoco/mujoco200/bin$ python3
Python 3.6.9 (default, Oct 8 2020, 12:12:24)
[GCC 8.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> quit()
benjamin@APOLLO-II:~/mujoco/mujoco200/bin$ pip3

Usage:
  pip <command> [options]

Commands:
  install           Install packages.
  download          Download packages.
  uninstall         Uninstall packages.
  freeze           Output installed packages in requirements format.
  list              List installed packages.
  show              Show information about installed packages.

```

Figure 5: We can see that we have both Python 3.6.9 and a corresponding pip3 installation

Before we can actually install mujoco-py with pip3 (or pip with Anaconda) we need various other libraries we can install like so:

```

1 sudo apt install libosmesa6-dev libgl1-mesa-glx libglfw3 libglew-dev

```

Once these are installed, you should be able to install mujoco-py and then test it like so:

```

1 $ pip3 install -U 'mujoco-py<2.1,>=2.0'
2 $ python3
3 >>> import mujoco_py
4 >>> import os
5 >>> mj_path, _ = mujoco_py.utils.discover_mujoco()
6 >>> xml_path = os.path.join(mj_path, 'model', 'humanoid.xml')
7 >>> model = mujoco_py.load_model_from_path(xml_path)
8 >>> sim = mujoco_py.MjSim(model)
9
10 >>> print(sim.data.qpos)
11 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
12
13 >>> sim.step()
14 >>> print(sim.data.qpos)
15 [-2.09531783e-19  2.72130735e-05  6.14480786e-22 -3.45474715e-06

```

```

16 | 7.42993721e-06 -1.40711141e-04 -3.04253586e-04 -2.07559344e-04
17 | 8.50646247e-05 -3.45474715e-06 7.42993721e-06 -1.40711141e-04
18 | -3.04253586e-04 -2.07559344e-04 -8.50646247e-05 1.11317030e-04
19 | -7.03465386e-05 -2.22862221e-05 -1.11317030e-04 7.03465386e-05
20 | -2.22862221e-05]

```

Now if we were to try and use mujoco-py to launch a simulation right now, we'd probably see an error like this:

```

benjamin@APOLLO-II:~/kantor_lab/python_mujoco_simulation/examples$ conda activate
(base) benjamin@APOLLO-II:~/kantor_lab/python_mujoco_simulation/examples$ python markers_demo.py
Creating window glfw
ERROR: GLEW initialization error: Missing GL version
Press Enter to exit ...

```

Figure 6: Linker issue with shared libraries

So the last thing we need to do is tell the GNU/Linux GCC linker to preload the libGLEW shared library, otherwise it can't find it.

```

1 | export LD_PRELOAD = /usr/lib/x86_64-linux-gnu/libGLEW.so

```

I put this line in my `~/.bashrc` file so that I don't have to set this environment variable everytime I start a new shell. You can do this like so:

```

1 | echo "export LD_PRELOAD = /usr/lib/x86_64-linux-gnu/libGLEW.so" >> ~
   | ↪ ~/.bashrc

```

This is what we call a "hacky" way to solve a problem.

2.3 Testing Graphical Interface

Now we need to test the graphical interface setup. Create a directory called `~/mujoco_sims` and download the example scripts `body_interaction.py`, `markers_demo.py`, `setting_state.py` from OpenAI; <https://github.com/openai/mujoco-py/tree/master/examples>.

Then run them with your chosen python interpreter.

```

1 | :~/mujoco_sims$ python3 markers_demo.py

```

You should see text saying "Creating window glfw", and then following screen:

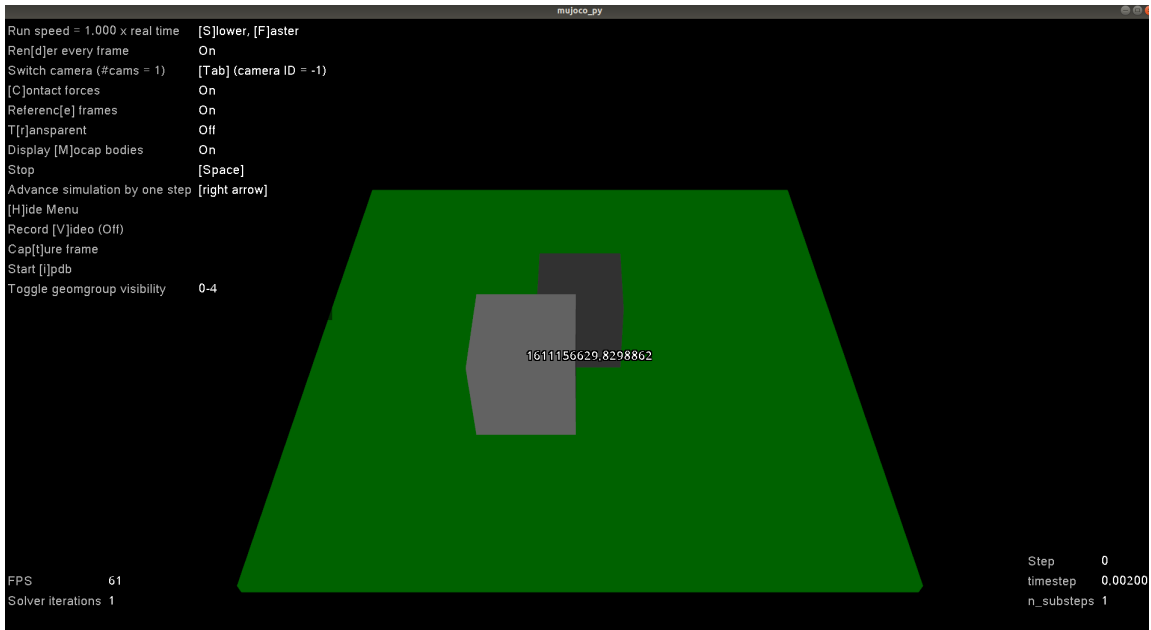


Figure 7: Testing markers demo. There should be a video of a cube orbiting another cube.

Rejoice! You’ve successfully set up MuJoCo and mujoco-py on your computer. Now you can get to work...

References

- [1] T. Erez, Y. Tassa, and E. Todorov. Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4397–4404, 2015.
- [2] OpenAI. mujoco-py github. <https://github.com/openai/mujoco-py>.
- [3] E. Todorov. Chapter 1: Overview. <http://www.mujoco.org/book/index.html>.
- [4] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012.