Physics Engine on Ubuntu

<Last Updated on July 21st, 2023, at 3:00 pm KST>

Jungjae Lee[1]

[1] Boston University College of Engineering

Table of Contents

1. Physics Engine

1.1 What is Physics Engine?

- Physics Engine provides a mathematical model and algorithms to **compute the behavior of objects** in a virtual environment, considering the principles of classical mechanics, such as gravity, collisions, forces, and motion.
- Physics Engine **incorporates realistic physics simulations into virtual environments**, enhancing the visual and interactive experience for users.

*** Note (1) *** There are various physics engines available, each catering to **specific use cases** such as 2D simulations, 3D simulations, game development, robotics control, and research purposes. The selection of a physics engine depends on the specific requirements and intended application, as different engines excel in different domains.

Well-known Physics Engines:

- **Bullet Physics**
  - Open-source physics engine.
  - Widely used in the gaming industry for physics-based simulations in both 2D and 3D environments.
  - Known for its open-source nature, flexibility, and robustness.
- **NVIDIA PhysX**
  - Physics engine developed by NVIDIA.
  - Predominantly used in the gaming industry and is recognized for its high-performance physics simulation.
  - Offers advanced features like real-time cloth simulation, particle simulation, and fluid simulation.
- **Havok Physics**
  - Physics engine developed by Havok.
  - Widely used in many AAA video games.
  - Offers robust collision detection and response, rigid body dynamics, constraints, and complex character animation support.
  - Known for its high performance and stability.
- **ODE (Open Dynamics Engine)**
  - Open-source physics engine.
  - Provides a versatile set of tools for simulating rigid body dynamics, joints, constraints, and collisions.
  - Widely used in various applications, including games, virtual reality, and robotics.
- **Unity Physics**
  - Physics engine developed by Unity Technologies.
  - Used in the Unity game engine and supports both 2D and 3D simulations.

- Known for its performance, stability, and integration with other Unity features.

Well-known Physics Engines for Robotics Control:

- **MuJoCo (Multi-Joint Dynamics with Contact)**
  - Widely utilized physics engines in the robotics research community.
  - Focuses on simulating articulated rigid bodies and their interactions with contacts and constraints.
  - Offers efficient and accurate simulations of complex robotics systems.
- **Gazebo**
  - Versatile robotics simulation environment.
  - Widely used for robotics development, testing and research.
  - Provides realistic physics simulation, sensor modeling, and integration with Robot Operating System (ROS).

Well-known Physics Engines in medical field:

- **SOFA (Simulation Open Framework Architecture)**
  - Open-source physics simulation framework.
  - Specializes in simulating deformable objects, soft tissues, and biomechanical systems.
  - Advanced collision handling, fluid simulation, and interaction models that make it suitable for medical simulations, surgical planning, and virtual training.

*** Note (2) *** This paper will cover **MuJoCo**, **Gazebo**, and **SOFA** simulations with **ROS**.

*** Note (3) *** Although **Ubuntu is NOT a requirement** for running a Physics Engine, this paper will focus on explaining the steps using Ubuntu.

2. General Installation

2.1 Ubuntu

2.1.1 Notes

- Recommend installing **Ubuntu 22.04 Jammy Jellyfish LTS**, as it ensures support until April 2027 and is compatible with a wide range of software, as of 2023.

2.1.2 Instruction for installing Ubuntu

- Ubuntu can be installed through Ubuntu's official website at https://ubuntu.com/.
- Once Ubuntu has been installed, run the code below to ensure that all the packages and security updates are up to date.

Update packages on Ubuntu

```
sudo apt update

sudo apt upgrade

<ERROR 2.1.2.1>
<ERROR 2.1.2.2>
```

Check the Ubuntu version on Ubuntu

```
lsb_release -a
```

2.1.2 Debugging: Warning & Error

-
- After installing Ubuntu, you might face an error message ('Unable to install updates: cannot refresh "snap-store"') when updating Ubuntu software. If you encounter this error, run the code below.

```
killall Snap-store Snap refresh
```

2.2 Visual Studio Code

- Visual Studio Code can be installed either through the Ubuntu Software app or by visiting the official Visual Studio Code website at https://code.visualstudio.com/#alt-downloads.

2.3 Python

2.3.1 Notes

- Generally, Python comes **pre-installed in Ubuntu,** as Python is part of the essential system components required for Ubuntu's functionality.
    - Many system tools and utilities are written in Python, and they rely on the presence of a Python interpreter.

2.3.2 Instruction for installing Python

- Python can be installed by either visiting the official Python website at https://www.python.org/downloads/ or by following the steps outlined below.

Install Python on Ubuntu [2.3.2.1]
// Check Python's version as most ubuntu come with Python pre-installed python --version
Continue if Python is not installed

// Allows you to easily manage your distribution and independent software sources
sudo apt install software-properties-common
 	
// Allows you to install multiple Python versions on Ubuntu System
sudo add-apt-repository ppa:deadsnakes/ppa
 	
sudo apt update
 	
sudo apt install python3.11

2.4 pip

2.4.1 Notes

- pip is typically **installed automatically when Python is installed**.

2.4.2 What is pip?

- A package manager used to install and manage Python packages.
    - Here, Python packages are software libraries or modules that provide specific functionality.
- Connects to PyPI to install the requested packages onto the local system.
    - Here, PyPI is a Python Package Index, which is an online repository that hosts a vast collection of Python packages.

2.4.3 Instruction for installing pip

- pip can be installed by following the steps outlined below.

Install pip on Ubuntu [2.4.3.1]
// Check pip's version as Python typically come with pip pre-installed pip –version
Continue if pip is not installed
sudo apt install python3-pip

3. Basic Information on Python

3.1 setup.py

3.1.1 Notes

- It is generally recommended **NOT** to call setup.py directly when installing or distributing python packages.

3.1.2 What is setup.py?

- setup.py is a common file used in Python projects to define the metadata and configuration for a python package.
    - In example, pip uses setup.py to install module.
- Typically used in conjunction with the setuptools library.

3.2 setuptools

3.2.1 Notes

- setuptools is included in the standard library starting from Python version 3.4. Therefore, setuptools should be installed automatically when Python is installed.

3.2.2 What is setuptools?

- A package development library that is a collection of enhancements to the distutils module.
- The user can define metadata for their package, such as its name, version, author, and dependencies.

3.3 distutils

3.3.1 Notes

-

3.3.2 What is distutils?

- A module in the Python standard library.
- Provides functionality for packaging and distributing Python modules and extensions.
- Simplifies the process of creating distribution packages for Python projects.

3.4 setuptools vs. distutils

3.4.2 Notes:

- distutils is considered the **predecessor** to setuptools.
- Largely superseded distutils in modern Python development.
    - Although distutils is still included in the Python standard library and can be used for simpler projects or in situations where using setuptools is not feasible or necessary.

3.4.2 Comparison:

- Distutils
  - distutils was introduced in Python 1.6 and provided basic packaging functionality, allowing developers to create distribution packages for their Python projects.
- Setuptools
  - Setuptools, on the other hand, was created as an enhancement to distutils. It builds upon the features provided by distutils and introduces additional functionality such as dependency management, automatic script generation, and plugin support.

3.5 venv

3.5.1 What is venv?

- venv (virtual environment), is a Python module that allows user to create isolated and independent Python environments.
- venv enables you to create separate environments for different projects, each with its own set of installed packages and a Python interpreter.
  - venv creates a self-contained directory that includes a copy of the Python interpreter, along with the necessary standard library files and scripts.
  - venv also creates a separate "site-packages" directory where user can install additional packages specific to that environment.

4. MuJoCo

4.1 MuJoCo

- Some physics simulation software uses “Cartesian” or “Subtractive” to represent degrees of freedom.
- MuJoCo uses a representation known as the “Lagrangian”, “generalized” or “additive” representation, whereby objects have no degree of freedom unless explicitly added using joints.

4.2 Required Installation

- Install MuJoCo [4.2.1.x]

Install MuJoCo on Ubuntu [4.2.2.x]

```
pip install mujoco==2.3.6
```

- Install UR5e

4.3 Notes

4.3.1 mujoco-py (Not Supported)

- **Mujoco-py** allows using MuJoCo from Python 3, supported by the OpenAI Robotics team.
- mujoco-py does not support any version after MuJoCo 2.1.0, according to GitHub [4.3.1.1]. Instead, you can use MuJoCo Python Bindings [4.3.1.2] or DeepMind Control Software [4.3.1.3].
- Install mujoco-py

Install mujoco-py using pip [4.3.1.4]

```
pip install mujoco-py
```

- Additional Sources 1

4.3.2 MuJoCo Gym (Not Supported)

- **MuJoCo Gym** is a standard API for reinforcement learning, and a diverse collection of reference environments. [4.3.2.1]
- The newest OpenAI gym does not work with MuJoCo 2.0, which you will have to install MuJoCo 1.50 binaries. Alternatively, if you need to use MuJoCo 2.0, you can download the MuJoCo 2.0 binaries and install the newest mujoco-py. Then, you can install the latest Gym that supports MuJoCo 2.0 using the command below. [4.3.2.2]

Install MuJoCo Gym using pip [4.3.2.3]

```
pip install -U gym[all]==0.15.3
```

- DeepMind does not support MuJoCo Gym anymore. Alternatively, you can use Gymnasium.

4.3.3 MuJoCo Python Bindings (Tutorial Available)

-

4.3.4 DeepMind Control Software (Tutorial Available)

-

Install dm-control using pip
pip install dm-control

4.3.5 Gymnasium

- **Gymnasium** is a standard API for reinforcement learning, and a diverse collection of reference environments, maintained fork of OpenAI's Gym library. [4.3.5.1]
- Install Gymnasium

Install Gymnasium on Ubuntu [4.3.5.2]
pip install gymnasium[classic-control]

4.4 Codes

4.4.1 mujoco

-

4.4.2 Extension

- Allow user-defined logic to be inserted into various parts of MuJoCo's computational pipeline.
- Designed to overcome the disadvantages of MuJoCo's physics callbacks.

4.4.3 Asset

- A Collection of resources or data files that are used to define the various components of a simulation.
- Can include elements that contribute to the visual and physical aspects of the simulation.

4.4.4 visual (Global)

- This element is in one-to-one correspondence with the low-level structure mjVisual contained in the field mjModel.vis of mjModel.
- Affect the visualizer, or the abstract phase of visualization.

4.5 Tutorials

4.5.1 MuJoCo Python Bindings Tutorial

- https://colab.research.google.com/github/deepmind/mujoco/blob/main/python/tutorial.ipynb#scrollTo=IbZxYDxzoz5R

4.5.2 MuJoCo dm_control python bindings Tutorial

- https://colab.research.google.com/github/deepmind/dm_control/blob/main/dm_control/mujoco/tutorial.ipynb

5. Gazebo

- Install Gazebo Sim 7 (Ubuntu Software App or https://gazebosim.org/docs/all/getstarted)

6. SOFA

6.1 What is SOFA (Simulation Open Framework Architecture)

- An open source for multi-physics simulation that provides a platform for modeling and simulating physical phenomena, including mechanical deformation, fluid dynamics, and medical simulations.

6.2 Notes

- SOFA only supports the latest Ubuntu LTS version, according to the official SOFA website.
    - As of 2023, the latest Ubuntu LTS version is **Ubuntu 22.04 LTS**.
- SOFA requires **Python 3.8** version based on official SOFA GitHub information.

6.3 Python version (Required)

6.3.1 Notes

- It is **required to have Python version 3.8** to run SOFA, according to the official SOFA website. If your Linux operating system is using a different Python version, you have the option to either create an alternative Python environment or create the virtual environment to install Python version 3.8 without altering the default version.

6.3.2 Instruction for installing Python 3.8 with alternatives python

- Python 3.8 can be installed by following the steps outlined below.

Install Python 3.8.17 on Ubuntu
1. Install “Python 3.8.17” from Python official website at https://www.python.org/downloads/source/ 2. Extract the file at the desired location *** Note *** The decision of where to install Python 3.8 depends on the specific requirements of the user. If the system's default Python version is different and Python 3.8 is only needed for SOFA or specific software, it is **advisable to install it locally in user-specific directories**. For more detailed information on these topics, please refer to Chapter 6. // Navigate to the extracted Python folder, and execute the following code: ./configure make sudo make install

```
Python3.8 –version
```

Option 1: Create an alternative Python environment

```
// sudo update-alternatives --install <new folder location> python <python location> <priority>
// Here, new "python" folder will install on <new folder location>
sudo update-alternatives --install /home/softrobotics/ python /home/softrobotics/python-3.8.17 1

// Check the alternatives python
sudo update-alternatives –config python
```

Option 2: Create a virtual environment using a venv

```
which python3

// Let's assume the location of python3 is /usr/local/bin/python3.8
// Add the path to the ~/.bashrc file
echo 'export PATH="/usr/local/bin:$PATH"' >> ~/.bashrc

// Apply the changes
source ~/.bashrc

sudo apt update

// Install a virtual environment
sudo apt install python3-venv

// Create a virtual environment
python3.8 -m venv env1

// Activate the virtual environment
source env1/bin/activate
```

6.4 SOFA (Required)

6.4.1 Notes

- Although SOFA 23.06 is available, it is advised to install **SOFA 22.12** as SofaGym does not support the latest version yet, as of July 2023.

6.4.2 Instruction for installing SOFA

- SOFA can be installed by following the steps outlined below.

Install Standard Compilation toolkit on Ubuntu (Required) [6.4.2.1]
sudo apt install build-essential software-properties-common
Install Clang on Ubuntu (Recommended) [6.4.2.1] *** Note *** Users have the option to choose either GCC or Clang for compilation. However, it is **advisable to use Clang** due to its faster compilation speed, which is approximately twice as fast as GCC.

// Search which Clang versions are available for distribution
apt-cache search '^clang-[0-9.]+$'
 	
// Install the latest Clang versions. Let's assume the latest Clang version is clang-12.
sudo apt install clang-12

Install GCC on Ubuntu (Optional) [6.4.2.1] Alternative to Clang. If you have installed Clang, skip to the next step.

// Search which GCC versions are available for distribution
apt-cache search '^gcc-[0-9.]+$'
 	
// Install the latest GCC versions. Let's assume the latest GCC version is gcc-11.
sudo apt install gcc-11

Install CMake on Ubuntu (Recommended) [6.4.2.1]
sudo apt install cmake cmake-gui
Install Ninja (build system) on Ubuntu (Recommended) Alternative to CMake. If you have installed CMake, skip to the next step.
sudo apt install ninja-build
Install CCashe (cashing system) on Ubuntu (Recommended) *** Note *** Installation of this is optional; however, it is strongly recommended as it will enhance compilation time when making changes to SOFA
sudo apt install ccache

Install Qt (>= 5.12.0) on Ubuntu (Required)

1. First, Install "qt-unified-linux-x64-online.run" at the official Qt website at https://download.qt.io/official_releases/online_installers/.

2. While it is installing, find the closest country to mirror the http at mirror list at the official Qt website at https://download.qt.io/static/mirrorlist/. This will prevent the user from waiting 10+ hours to install the Qt.

3. When "qt- unified-linux-x64-online.run is installed, go to the terminal, and navigate to the directory that this file is located. Then, run the code below.

```
// Give execute permission
Chmod +x qt-unified-linux-x64-online.run

// Execute the file
// Let's assume the closest country on mirror (step 2) was South Korea. Copy the Japan's Http address, which is https://ftp.jaist.ac.jp/pub/qtproject/
./qt-unified-linux-x64-online.run –mirror https://ftp.jaist.ac.jp/pub/qtproject/
```

Setup Qt

1. Complete from Login to Contribute to Qt Development section.

2. Check "Custom installation"

*** Note *** You are more than welcome to enable any necessary components; however, these three are strongly recommended

3. Expand "Qt" -> Expand "Qt 6.4.3" (or wanted version) -> Check "Desktop gcc 64-bit"

4. In Qt 6.4.3, expand "Additional Libraries" -> Check "Qt Charts" and "Qt WebEngine"

Install required libraries on Ubuntu

```
// Install OpenGL
sudo apt install libopengl0

// Install Boost (>= 1.65.1)
sudo apt install libboost-all-dev

// install Python3.8 + pip + numpy + scipy
sudo apt install python3.8-dev

sudo apt-get install python3-distutils \ && curl -L https://bootstrap.pypa.io/pip/get-pip.py --output /tmp/get-pip3.py \ && python3.8 /tmp/get-pip3.py \ && python3.8 -m pip install --upgrade pip \ && python3.8 -m pip install numpy scipy

// Install additional libraries (libPNG, libJPEG, libTIFF, Glew, Zlib)
sudo apt install libpng-dev libjpeg-dev libtiff-dev libglew-dev zlib1g-dev

Install Eigen (>=  3.2.10)
sudo apt install libeigen3-dev
```

Install SOFA Plugins on Ubuntu (Recommended) [6.4.2.1]

```
// CGALPlugin
sudo apt install libcgal-dev libcgal-qt5-dev

// MeshSTEPLoader
sudo apt install liboce-ocaf-dev

// SofaAssimp
sudo apt install libassimp-dev

// SofaCUDA
sudo apt install nvidia-cuda-toolkit

// SofaHeadlessRecorder
sudo apt install libavcodec-dev libavformat-dev libavutil-dev libswscale-dev

// SofaPardisoSolver
sudo apt install libblas-dev liblapack-dev
```

Build SOFA on Ubuntu (Required) [6.4.2.1]

```
// Install development unstable version on the master branch
```

git clone -b master https://github.com/sofa-framework/sofa.git sofa/src

Generate a Makefile with CMake on Ubuntu [6.4.2.1]

cd sofa/

mkdir build/

cd build/

cmake-gui

// A popup will ask to specify the generator for the project.

1. Specify the generator
   - If you installed Ninja, select "CodeBlocks – Ninja"
   - If you have NOT installed Ninja, select "CodeBlocks – Unix Makefile"

2. Choose "Specify native compilers" and press "Next"

3. Set the C compiler to "/usr/bin/clang" OR "/usr/bin/clang"

4. Set the C++ compiler to "/usr/bin/clang++" OR "/usr/bin/g++"

5. Run Configuration

6. Customize SOFA via Cmake variables
   * Let's assume python 3.8 path is "/home/softrobotics/Python-3.8.17"
   - Set **PLUGIN_SOFAPYTHON3** to "/home/softrobotics/Python-3.8.17"
   - Set **PYTHON_EXECUTABLE** to "/home/softrobotics/Python-3.8.17"
   - Set **PYTHON_INCLUDE_DIRS** to "/home/softrobotics/Python-3.8.1/include"
   - Set **PYTHON_LIBRARIES** to "/home/softrobotics/Python-3.8.17"
   - Set **SP3_PYTHON_PACKAGES_DIRECTORY** to "python3/site-packages"
   - Set **SP3_PYTHON_PACKAGES_LINK_DIRECTORY**
     To "/home/softrobotics/Python-3.8.17/Lib/site-packages"
   - Set **SofaPyton3Tools** to "SofaPython3Tools-NOTFOUND"
   - Set **SofaPython3_DIR** to "/home/softrobotics/sofa/build/lib/cmake"
   - Set **CMAKE_BUILD_TYPE** to "RelWithDebInfo"

7. Run Configuration

8. Run Generate

Run SOFA software

```
./bin/runSofa
```

6.5 SofaGym

6.5.1 Notes

- SofaGym **requires SOFA version of 22.12**.
- SofaGym also **requires SofaPython3**, which will be introduced below in ***Chapter 6.6***.
    - Outside of SofaPython3, there are **several optional plugins available** that can be installed if needed. These plugins will be introduced in ***Chapter 6.7 - 6.9***.

6.5.2 What is SofaGym?

- An open-source Python library that provides a reinforcement learning environment for simulating and training agents in physics-based scenarios in the SOFA framework.
- Enables researchers and developers to apply reinforcement learning algorithms to control and interact with deformable objects and their environments.
- Users can define reinforcement learning tasks, such as object manipulation or locomotion, and train reinforcement learning agents to learn effective control policies within these simulations.

6.6 SofaGym: SofaPython3 Plugin (Required)

6.6.1 Notes

- The following **three components are required** to install the SofaPython3 plugin: [6.2.3.1]
    - pybind11 (minimal 2.3)
    - cmake (minimal 3.11)
    - python3-dev
- This section assumes that users have already installed cmake, as explained in ***Chapter 6.4***, during the installation of the SOFA framework.

6.6.2 What is SofaPython3 Plugin?

- A module that provides support for integrating Python 3 into the SOFA framework.
- Allows developers to write and execute Python 3 scripts within the SOFA environment.

6.6.3 Instruction for installing SofaPython3 Plugin

- SofaPython3 Plugin can be installed by following the steps outlined below.

Install pybind11 on Ubuntu [6.2.3.2]

```
Sudo apt install pybind11-dev
```

Install python3-dev on Ubuntu [6.2.3.3]

```
Sudo apt install python3-dev
```

Install SofaPython3 Plugin

```
cd home/softrobotics/sofa/build/plugins

git clone https://github.com/sofa-framework/SofaPython3.git sofa/src/plugins

CMAKE_EXTERNAL_DIRECTORIES=/home/softrobotics/sofa/build/plugins

// Find the location of local python3
which python3

// Go to the directory that contains CMakeLists.txt
cd ~/home/softrobotics/sofa/src

// Let's assume the location of local python3 is /usr/bin/python3
cmake -DPython_EXECUTABLE=/usr/bin/python3
```

Install the python 3 bindings

6.7 SofaGym: SoftRobots Plugin (Optional)

6.7.1 Notes

-

6.7.2 What is SoftRobots Plugin

-

6.7.3 Instruction for installing SoftRobots Plugin [6.2.4.1]

- SoftRobots Plugin can be installed by following the steps outlined below.

Install SoftRobots Plugins [6.2.4.2]

```
git clone https://github.com/SofaDefrost/SoftRobots

// In the CMake gui, add the plugin path to SOFA_EXTERNAL_DIRECTORIES.
```

6.8 SofaGym: STLIB (Optional)

STLIB (Sofa Template Library)

Install STLIB [6.2.5.1]

```
git clone https://github.com/SofaDefrost/STLIB.git

// In the CMake gui, set PLUGIN_SOFAPYTHON on

// In the CMake gui, add the path of STLIB (such as "your_path/STLIB) to SOFA_EXTERNAL_DIRECTORIES

// Then build SOFA

// You should be able to use "import stlib in python from inside SOFA
```

6.2.6 Install REALSENSE Camera Module

6 REALSENSE Camera

Install REALSENSE Camera module [6.2.6.1]

```
// Register the server's public key
sudo mkdir -p /etc/apt/keyrings

curl -sSf https://librealsense.intel.com/Debian/librealsense.pgp | sudo tee /etc/apt/keyrings/librealsense.pgp > /dev/null

// Install apt HTTPS support
sudo apt-get install apt-transport-https

// Add the server to the list of repositories
echo "deb [signed-by=/etc/apt/keyrings/librealsense.pgp] https://librealsense.intel.com/Debian/apt-repo `lsb_release -cs` main" | \

sudo tee /etc/apt/sources.list.d/librealsense.list

sudo apt-get update

// Install the libraries – deploy librealsense2 udev rules, build, and activate kernel modules, runtime library, and executable demos and tools
sudo apt-get install librealsense2-dkms
sudo apt-get install librealsense2-utils

// Install the developer and debug packages (Optional)
```

```
sudo apt-get install librealsense2-dev
sudo apt-get install librealsense2-dbg
```

6.2.6.1: https://github.com/SofaDefrost/SoftRobots

https://www.sofa-framework.org/community/doc/plugins/build-a-plugin-from-sources/#in-tree-build

https://github.com/SofaDefrost/SofaGym

https://github.com/sofa-framework/SofaPython3

7. ROS 2 (Humble) with MoveIt 2 Motion Planning Framework

7.1 ROS 2

- There are several ROS 2 versions.
  - Humble Hawksbill
    - Released in May 2022, EOL in May 2027
    - Support Ubuntu 20.04, Ubuntu 22.04 LTS, Windows 10, and macOS
  - Iron Irwini
    - Released in May 2023, EOL in November 2024
    - Support Ubuntu 22.04 LTS, Windows 10, and macOS
  - Rolling Ridley
    - Ongoing development
    - Support Ubuntu 22.04 LTS, Windows 10, macOS 10.14

7.2 General Installation

-

7.3 Notes

7.3.1 Robot Operating System

- **ROS 2 (Robot Operating System 2)** is an open-source software development kit for robotics applications. [7.3.1.1]
- Install ROS 2 (Humble)

Install ROS 2 (Humble) on Ubuntu [7.3.1.2]
locale // Check for UTF-8
Continue if it is not set as UTF-8. If it is already set, skip to the next step
sudo apt update && sudo apt install locales sudo locale-gen en_US en_US.UTF-8 sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8 export LANG=en_US.UTF-8 locale // Verify the setting
sudo apt install software-properties-common

```
sudo add-apt-repository universe

sudo apt update && sudo apt install curl -y

sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o /usr/share/keyrings/ros-archive-keyring.gpg

echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-keyring.gpg] http://packages.ros.org/ros2/ubuntu $(. /etc/os-release && echo $UBUNTU_CODENAME) main" | sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null

sudo apt update

sudo apt upgrade

sudo apt install ros-humble-desktop

source /opt/ros/humble/setup.bash
```

To run ROS, you must run "source /opt/ros/humble/setup.bash" to configurate the environment. Instead, you can do this to skip this step

```
gedit ~/.bashrc                    // This would open the ".bashrc" file

source /opt/ros/humble/setup.bash
// Type this code at the very bottom of the file, which will be on line 119
```

7.3.2 MoveIt Motion Planning Framework

- **MoveIt 2 Motion Planning Framework** is a robotic manipulation platform for ROS 2, and incorporates the latest advances in motion planning, manipulation, etc. [7.3.2.1]
- Install MoveIt 2 Motion Planning Framework

Install MoveIt 2 Motion Planning Framework [7.3.2.2]

```
sudo apt update

sudo apt dist-upgrade

rosdep update
```

```
sudo apt install -y \
build-essential \
cmake \
git \
libbullet-dev \
python3-colcon-common-extensions \
python3-flake8 \
python3-pip \
python3-pytest-cov \
python3-rosdep \
python3-setuptools \
python3-vcstool \
wget && \
```

```
python3 -m pip install -U \
argcomplete \
flake8-blind-except \
flake8-builtins \
flake8-class-newline \
flake8-comprehensions \
flake8-deprecated \
flake8-docstrings \
flake8-import-order \
flake8-quotes \
pytest-repeat \
pytest-rerunfailures \
pytest
```

```
sudo apt remove ros-$ROS_DISTRO-moveit*
// remove any pre-existing MoveIt debians

export COLCON_WS=~/ws_moveit2/

mkdir -p $COLCON_WS/src

cd $COLCON_WS/src

git clone https://github.com/ros-planning/moveit2.git -b $ROS_DISTRO
```

```
for repo in moveit2/moveit2.repos $(f="moveit2/moveit2_$ROS_DISTRO.repos"; test -r $f && echo $f); do vcs import < "$repo"; done

rosdep install -r --from-paths . --ignore-src --rosdistro $ROS_DISTRO -y

sudo apt install ros-$ROS_DISTRO-rmw-cyclonedds-cpp

export RMW_IMPLEMENTATION=rmw_cyclonedds_cpp

cd $COLCON_WS

colcon build --event-handlers desktop_notification- status- --cmake-args -DCMAKE_BUILD_TYPE=Release
<ERROR 7.3.1>

source $COLCON_WS/install/setup.bash
```

7.3.3 Colcon

- **Colcon** is a command line tool to improve the workflow of building, testing, and using multiple software packages.
- Install Colcon

Install Colcon

```
sudo apt install python3-colcon-common-extensions

sudo apt install python3-colcon-mixin

colcon mixin add default https://raw.githubusercontent.com/colcon/colcon-mixin-repository/master/index.yaml

colcon mixin update default

sudo apt install python3-vcstool                // install vcstool

mkdir -p ~/ws_moveit/src

cd ~/ws_moveit/src
git clone https://github.com/ros-planning/moveit2_tutorials -b main --depth 1

vcs import < moveit2_tutorials/moveit2_tutorials.repos

sudo apt update && rosdep install -r --from-paths . --ignore-src --rosdistro $ROS_DISTRO -y
```

```
cd ~/ws_moveit
colcon build --mixin release
<WARNING 7.3.2>
```

8. Warnings/Errors

ERROR 2.1.2.1

Error Message

E: Failed to fetch http://security.ubuntu.com/ubuntu/pool/main/f/fo2zjs/printer-driver-foo2zjs-common_20200505dfsg0-2ubuntu2.22.04.1_all.deb Connection timed out [IP:]

E: Unable to fetch some archives, maybe run apt-get update or try with –fix-missing?

Solution

```
sudo apt-get update --fix-missing

sudo apt-get upgrade --fix-missing
```

ERROR 2.1.2.2

Error Message

Can’t refresh snap-store

Solution

```
sudo killall snap-store
sudo snap refresh snap-store
```

ERROR 7.3.1

- Error Message:

```
--- stderr: moveit_setup_controllers

c++: fatal error: Killed signal terminated program cc1plus
```

```
compilation terminated.

gmake[2]: *** [CMakeFiles/moveit_setup_controllers.dir/build.make:298: CMakeFiles/moveit_setup_controllers.dir/include/moveit_setup_controllers/moc_urdf_modifications_widget.cpp.o] Error 1

gmake[2]: *** Waiting for unfinished jobs....

gmake[1]: *** [CMakeFiles/Makefile2:161: CMakeFiles/moveit_setup_controllers.dir/all] Error 2

gmake: *** [Makefile:146: all] Error 2

---
Failed   <<< moveit_setup_controllers [3min 43s, exited with code 2]

Aborted  <<< moveit_setup_core_plugins [3min 43s]

Aborted  <<< moveit_setup_app_plugins [3min 48s]

Aborted  <<< moveit_setup_srdf_plugins [3min 48s]

Summary: 48 packages finished [12min 27s]

1 package failed: moveit_setup_controllers
3 packages aborted: moveit_setup_app_plugins moveit_setup_core_plugins moveit_setup_srdf_plugins
5 packages had stderr output: controller_manager moveit_configs_utils moveit_setup_controllers moveit_setup_srdf_plugins ros2controlcli
2 packages not processed
```

- Solution:
  - Simply just re-run the code: “colcon build --event-handlers desktop_notification- status- --cmake-args -DCMAKE_BUILD_TYPE=Release”

WARNING 7.3.2

- Warning Message:

```
--- stderr: moveit_configs_utils
```

```
/usr/lib/python3/dist-packages/setuptools/command/install.py:34:
SetuptoolsDeprecationWarning: setup.py install is deprecated. Use build and pip and other standards-based tools.

  warnings.warn(
---
```

- Solution:

```
python3                    // check the python version

import setuptools
print(setuptools.__version__)          // check the setuptools version
```

if the setup tools version is above 58.2.0, follow the instructions below

```
pip install setuptools==58.2.0          // my version was 59.6.0 (updated to 68.0.0 due to "DeprecationWarning: The distutils package is deprecated and slated for removal in Python 3.12. Use setuptools or check PEP 632 for potential alternatives")
```

Source: https://answers.ros.org/question/396439/setuptoolsdeprecationwarning-setuppy-install-is-deprecated-use-build-and-pip-and-other-standards-based-tools/

9. Computer Term

9.1 General

9.1.1 Deserialized

- The process of reconstructing a data structure or object from a series of bytes or a string to instantiate the object for consumption.
- The reverse process of serialization (converting a data structure or object into a series of bytes for storage or transmission across devices)

9.1.2 Rendering

- Process of generating or creating a final image from a three-dimensional (3D) scene or mode.
- Involves calculating the color, shading, and other visual properties of objects in the scene and transforming them into a 2D image
- “Visual representation of contact points”

9.2 Directories

9.2.1 System-wide vs. User-specific directories

- **System-wide directories**
  - Directories that contain packages that are installed using the system’s package manager (‘apt’)
  - Accessible to all users on the system.
  - Typically used for packages that are required by the operating system or shared by multiple users.
    - Considered part of the operating system
  - Path example: /home/username/folder
- **User-specific directories**
  - Directories that contain packages that are installed by individual users using tools like ‘pip.’
  - Limited to the user who installed the packages.
  - Used for individual projects or personal use.
- **How to differentiate?** ‘ls -l’ command
  - **System-wide directories:** If the owner is a system-related user or group such as ‘root’, ‘sys’, ‘adm’, or ‘staff,’ and the group is something like ‘root’ or ‘admin,’ it suggests that the file or directory is system-wide directories.
  - **User-specific directories:** If the owner matches user’s username, and the group is the same as the owner’s username or a user-specific group, it suggests that the directory is user-specific directories.

9.3 Library & Framework

9.3.1 Wrapper Library

- Also known as a binding or interface library.
- A software component that provides an interface or bridge between two different programming languages or software systems.
    - It acts as an intermediary, handling the communication and translation between the two languages.
    - This allows developers to access the functionalities of a library or software written in one programming language from another language.
- Commonly used when there is a need to use a library or framework written in a specific language in a different programming language.

9.3.2 Open Graphic Library (OpenGL)

- A cross-platform API (Application Programming Interface) that provides a standardized set of functions for rendering 2D and 3D graphics.
- Operates as a state machine, where you set various parameters and issue rendering commands to generate graphics.

9.2.3 Graphics Library Framework (GLFW)

- A lightweight, cross-platform library for creating and managing windows, handling user input, and managing OpenGL or Vulkan contexts in graphical applications.

9.4 Programming

9.4.1 High vs. Low Level Programming

- **High-level Programming**
    - Designed to be more abstract and human-readable, providing a higher level of abstraction from the hardware and low-level details of the computer system.
    - Often have built-in functions, libraries, and syntax that simplify complex tasks and promote code reusability.
    - Examples: Python, Java, C#, JavaScript, Ruby, PHP, Swift, MATLAB, etc.
    - Advantages
        - Productivity: Faster development and prototyping due to their simplicity and expressive syntax. Often have extensive libraries and frameworks that provide pre-built functionality, saving development time.
        - Portability: Generally designed to be platform-independent, allowing code to be easily run on different operating systems with minimal modifications.
        - Readability: Closer to natural language, making code easier to understand, maintain, and debug. Usually offer clearer abstractions and higher-level constructs, enhancing code readability.

- Trade-offs
    - Performance: May introduce some overhead due to their abstractions and automatic memory management. Might not offer fine-grained control over system resources, which can impact performance in computationally intensive or time-critical applications.
    - Limited-low-level access: Often restrict direct access to hardware and low-level system operations, which can be a limitation for certain tasks that require low-level control.

- **Low-level Programming**
    - Writing code that is closer to the hardware and specific to the computer architecture.
    - Often requires a deeper understanding of the underlying hardware and system operations.
    - Examples: Assembly language, Machine code, C, C++, Rust, Ada, Fortran, etc.
    - Advantages
        - Control: Provide direct control over hardware and system resources, allowing fine-grained optimizations for performance-critical tasks.
        - Efficiency: Result in highly efficient and optimized programs as developers have full control over memory management, data structures, and system-level operations.
        - Embedded System: Crucial in developing software for embedded systems, microcontrollers, and devices with limited resources, where efficiency and direct hardware access are paramount.
    - Trade-offs
        - Complexity: Requires a deeper understanding of hardware architecture and system-level operations. Often involves manual memory management, dealing with pointers, and working at a lower level of abstraction, making it more complex and error prone.
        - Portability: Tends to be less portable as it relies heavily on the specific hardware and system architecture. Porting low-level code to different platforms may require significant modifications.

9.4.2 Errors (Few Common Errors)

- **Syntax Errors**
    - Occur when the code violates the rules and structure of the programming language.
    - These errors prevent the code from being compiled or executed.
    - Examples
        - Missing parenthesis
        - Missing semicolons
        - Using incorrect keywords
- **Runtime Errors**
    - Occurs during the execution of a program. Often result from logical mistakes or unexpected conditions encountered while the program is running.

    - Examples
        - Division by zero
        - Accessing an out-of-bounds array index
        - Calling a function with incorrect arguments
- **Logic Errors**
    - Bugs in the program that cause it to produce incorrect or unexpected results. May not generate any error messages or exceptions but cause the program to function incorrectly.
- **Type Errors**
    - Occur when incompatible data types are used in operations or assignments.
    - Examples
        - Adding a string to a numeric variable
        - Attempting to access a method that doesn't exit for a particular data type
- **Name Errors**
    - Occurs when a variable or function name is not recognized or not defined in the current scope.
    - Typically happens when a variable is referenced before it is declared or if there is a typo in the name.
- **Index Errors**
    - Occur when attempting to access an array or list using an invalid index.
    - Happens when the index is out of range, either too large or negative.
- Null Pointer Exceptions (or Null Reference Errors)
    - Occurs when a program tries to access or dereference a null object or variable.
    - Often happens when a reference has not been properly initialized or when a function returns null unexpectedly.
- **File I/O Errors**
    - Occur when there are issues reading from or writing to files
    - Examples
        - Attempting to access a file that does not exist
        - Lacking the necessary permissions
        - Encountering disk-related problems
- **Network Errors**
    - Occur when there are issues with network connectivity or when a program fails to establish or maintain a connection with remote servers or resources.
- **Exception Handling Errors**
    - Occur when exceptions are not properly caught, handled, or propagated in the code.
    - Can lead to unexpected program termination or incorrect behavior.
- **Boundary Errors**
    - Occur when the code does not handle boundary or edge cases properly.
    - Example
        - Not accounting for the minimum or maximum values of variables

        - Failing to handle empty arrays or strings
        - Overlooking special conditions that can lead to incorrect behavior or crashes
- **Memory Errors**
    - Occur when there are issues with memory allocation, deallocation, or management.
    - Example
        - Memory leaks (failing to release memory after use)
        - Dangling pointers (pointers pointing to deallocated memory)
        - Accessing memory that has already been freed

10. Sources

2. General Installation

2.3.2.1 Install Python on Ubuntu

- "How to install Python 3 on Ubuntu" (PhoenixNAP, 2019)
- https://phoenixnap.com/kb/how-to-install-python-3-ubuntu

2.4.3.1 Install pip on Ubuntu

- "How to install pip in Python 3 on Ubuntu" (Odoo, 2020)
- https://www.odoo.com/forum/help-1/how-to-install-pip-in-python-3-on-ubuntu-18-04-167715

4. MuJoCo

3.2.1.x Install MuJoCo

-
- https://github.com/deepmind/mujoco/releases

3.2.2.x Install MuJoCo using pip

-
- https://pypi.org/project/mujoco/

3.3.1.1 mujoco-py" (GitHub, 2022)

-
- https://github.com/openai/mujoco-py

3.3.1.2 "MuJoCo Python Bindings" (GitHub, 2023)

-
- https://github.com/deepmind/mujoco/blob/main/python/README.md

3.3.1.3 "DeepMind Control Software" (GitHub, 2023)

-
- https://github.com/deepmind/dm_control

3.3.1.4 Install mujoco-py

-
- https://pypi.org/project/mujoco-py/

3.3.2.1 "MuJoCo" (Gym Documentation)

-
- https://www.gymlibrary.dev/environments/mujoco/index.html

3.3.2.2 "Installing MuJoCo to Work with OpenAI Gym Environments" (Neptune.ai, 2023)

-
- https://neptune.ai/blog/installing-mujoco-to-work-with-openai-gym-environments

3.3.2.3 Install MuJoCo Gym

-
- https://neptune.ai/blog/installing-mujoco-to-work-with-openai-gym-environments

3.3.5.1 “Gymnasium” (Farama Foundation - Gymnasium Documentation)

-
- https://gymnasium.farama.org/

3.3.5.2 Install Gymnasium on Ubuntu

-
- https://gymnasium.farama.org/environments/classic_control/

4. Gazebo

6. SOFA

6.2.1.1

-
- https://brain2life.hashnode.dev/how-to-install-pyenv-python-version-manager-on-ubuntu-2004

6.2.3.1

-
- https://github.com/sofa-framework/SofaPython3

6.2.3.2

-
- https://sofapython3.readthedocs.io/en/latest/menu/Compilation.html#in-tree-build

6.2.3.3

-
- https://sofapython3.readthedocs.io/en/latest/menu/Compilation.html#in-tree-build

6.2.4.1

-
- https://project.inria.fr/softrobot/install-get-started-2/

6.2.4.2

-
- https://project.inria.fr/softrobot/install-get-started-2/download/

6.2.5.1

-
- https://github.com/SofaDefrost/STLIB

6.4.2.1

-
- https://www.sofa-framework.org/community/doc/getting-started/build/linux/

7. ROS 2 (Humble) with MoveIt2 Motion Planning Framework

7.3.1.1 “ROS2” (Robot Operating System)

-
- https://docs.ros.org/en/foxy/_downloads/2a9c64e08982f3709e23d20e5dc9f294/ros2-brochure-ltr-web.pdf

7.3.1.2 Install ROS 2 (Humble) on Ubuntu

-
- https://docs.ros.org/en/humble/Installation/Ubuntu-Install-Debians.html

7.3.2.1 “MoveIt 2 Documentation (MoveIt)

-
- https://moveit.picknik.ai/main/index.html

7.3.2.2 Install MoveIt 2 Motion Planning Framework

-
- https://moveit.ros.org/install-moveit2/source/

11. Additional Sources

1. Installing & Using MuJoCo 2.1.5 with OpenAI Gym (Reddit, 2022)

-
- https://www.reddit.com/r/reinforcementlearning/comments/usaigw/installing_using_mujoco_215_with_openai_gym/