Introduction to the MuJoCo Simulator

Will C. Forte
(will.c.forte@rutgers.edu)

Department of Mechanical Engineering
Rutgers University–New Brunswick

February 11, 2025
N2E Robotics

About Me

I am an undergraduate freshman in the Engineering Honors Academy and an aspiring roboticist. Currently, I work on legged robots and some simulations.

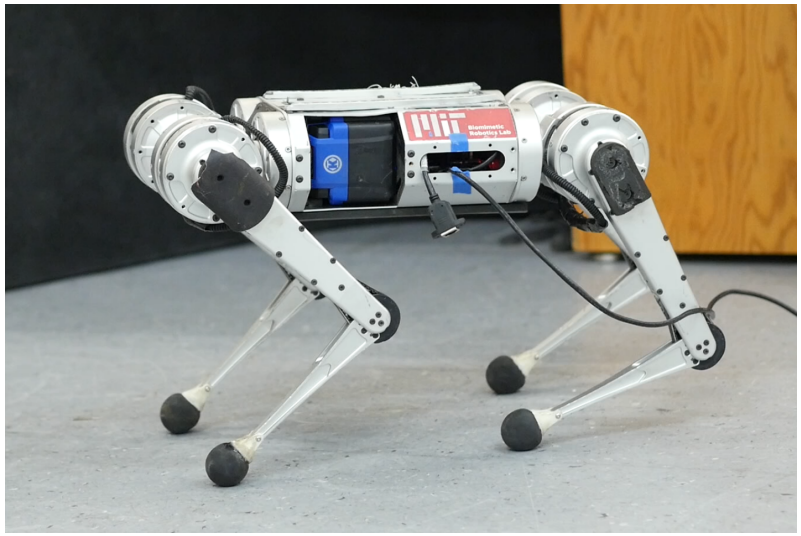
- 2023 – Intern at an NJIT robotics lab
- 2024 – Graduated high school
- 2024 – Intern at Prof. Burlion's UAV lab
- 2025 – Intern at Prof. Yi's legged robot lab

My personal project site is `willcforte.com`.

My Research Interests

- Bipedal and quadrupedal design/control
- UAV control
- Dynamical systems analysis
- Biomimetic design

Ben Katz' MIT Mini Cheetah inspired me to do robotics.



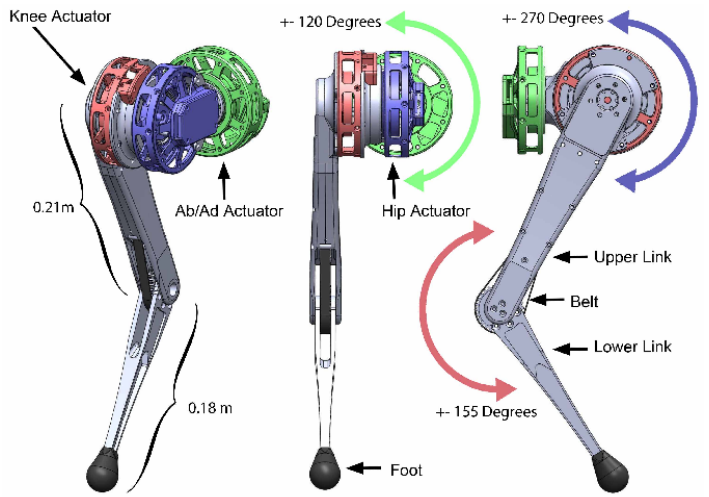
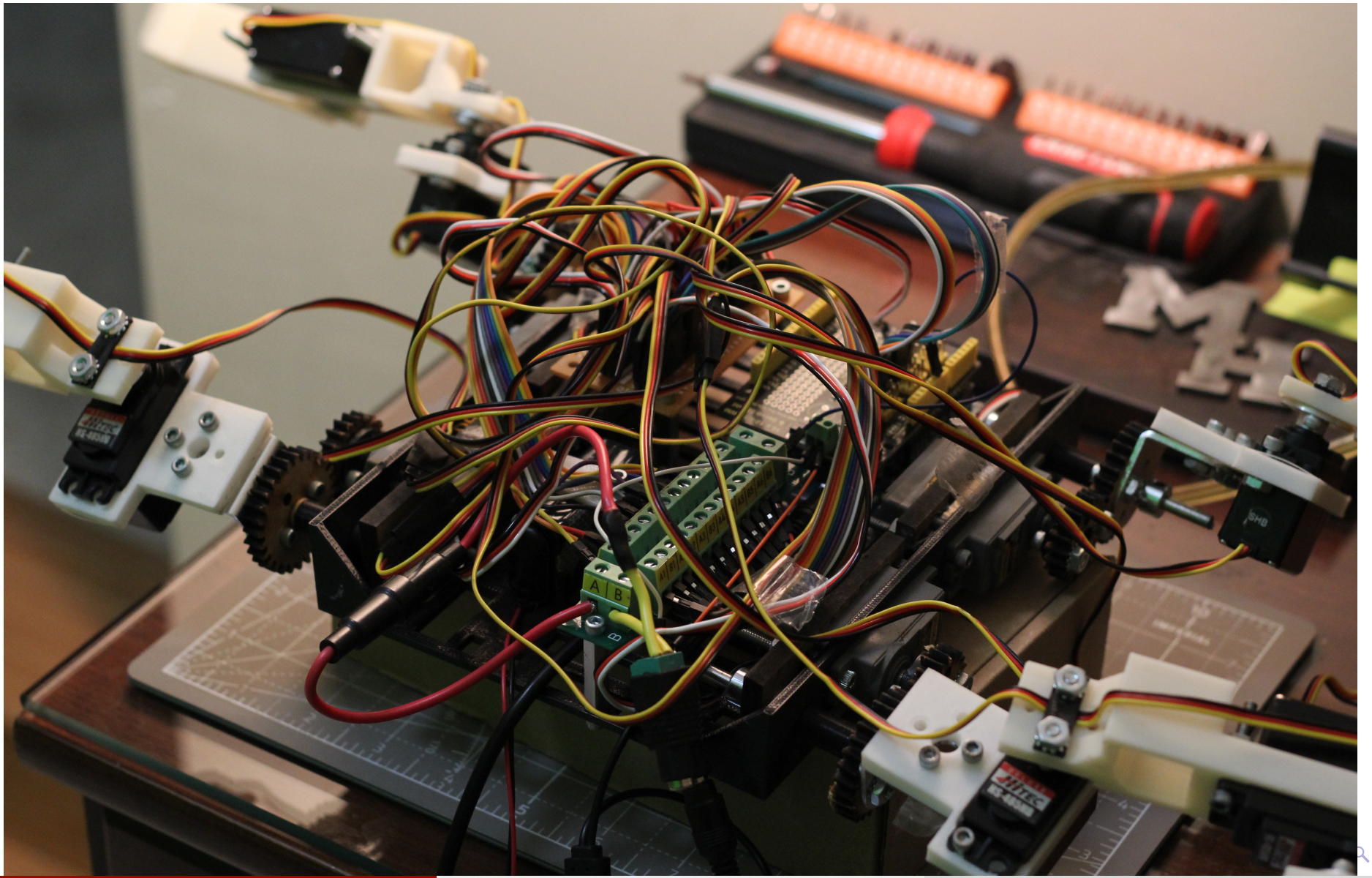
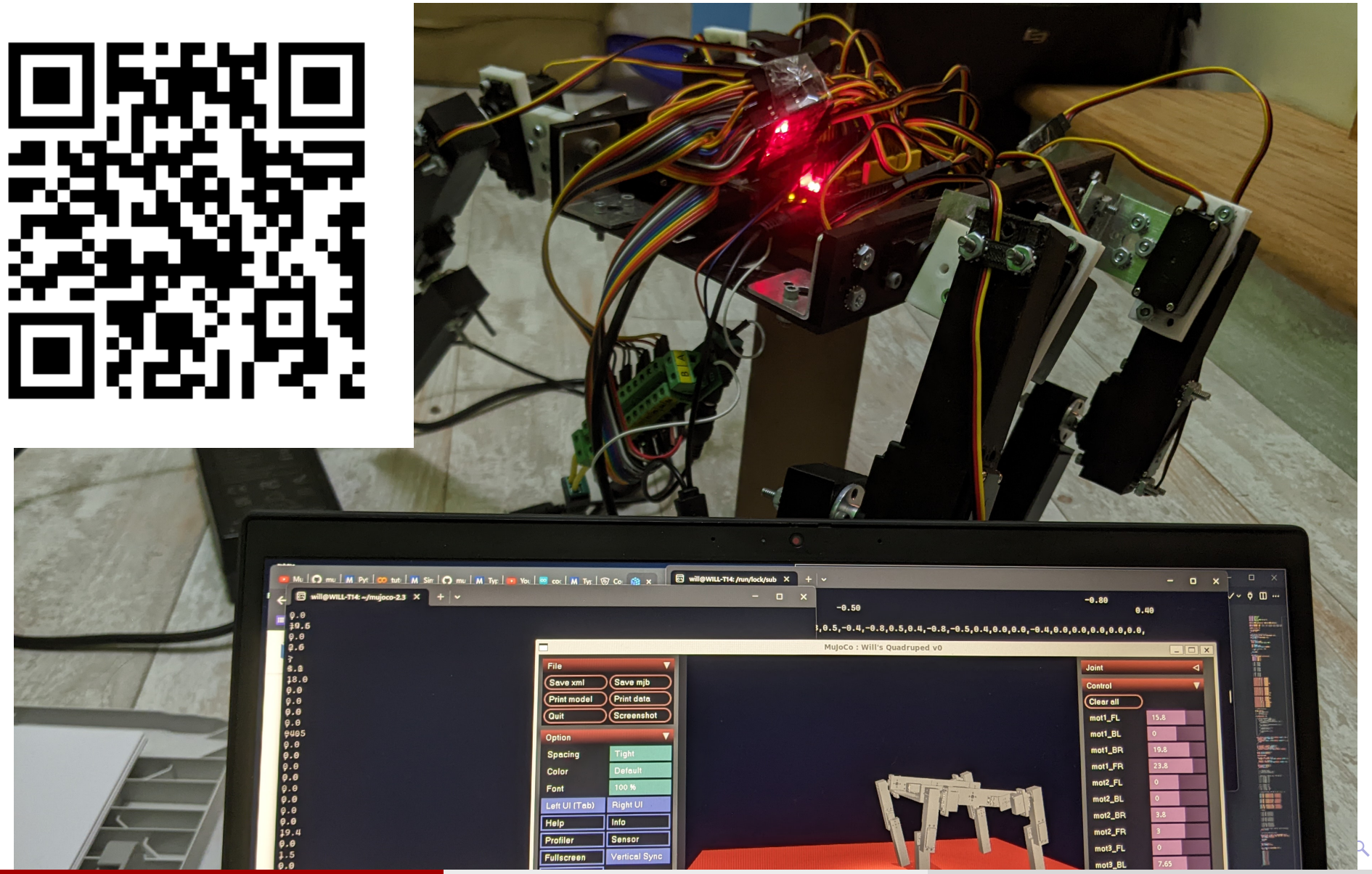


Fig. 3: CAD Diagram of a Mini Cheetah Leg. Ab/ad actuator is highlighted

DIY 12-Motor Quadruped (Summer of 2023)



My MuJoCo Simulation w/ QR



Why Simulate?

- You don't have to buy a robot
- Break things with no consequences
- Easier than a real experiment
- Accurate sensors & data collection
- It's fun!

What is MuJoCo?

MuJoCo is a robotics simulator developed by Emo Todorov of the University of Washington. It is maintained by Yuval Tassa's team at Google DeepMind. Its name is an acronym:

MuJoCo $\equiv$ **Mu**lti **Jo**int (dynamics with) **Co**ntact

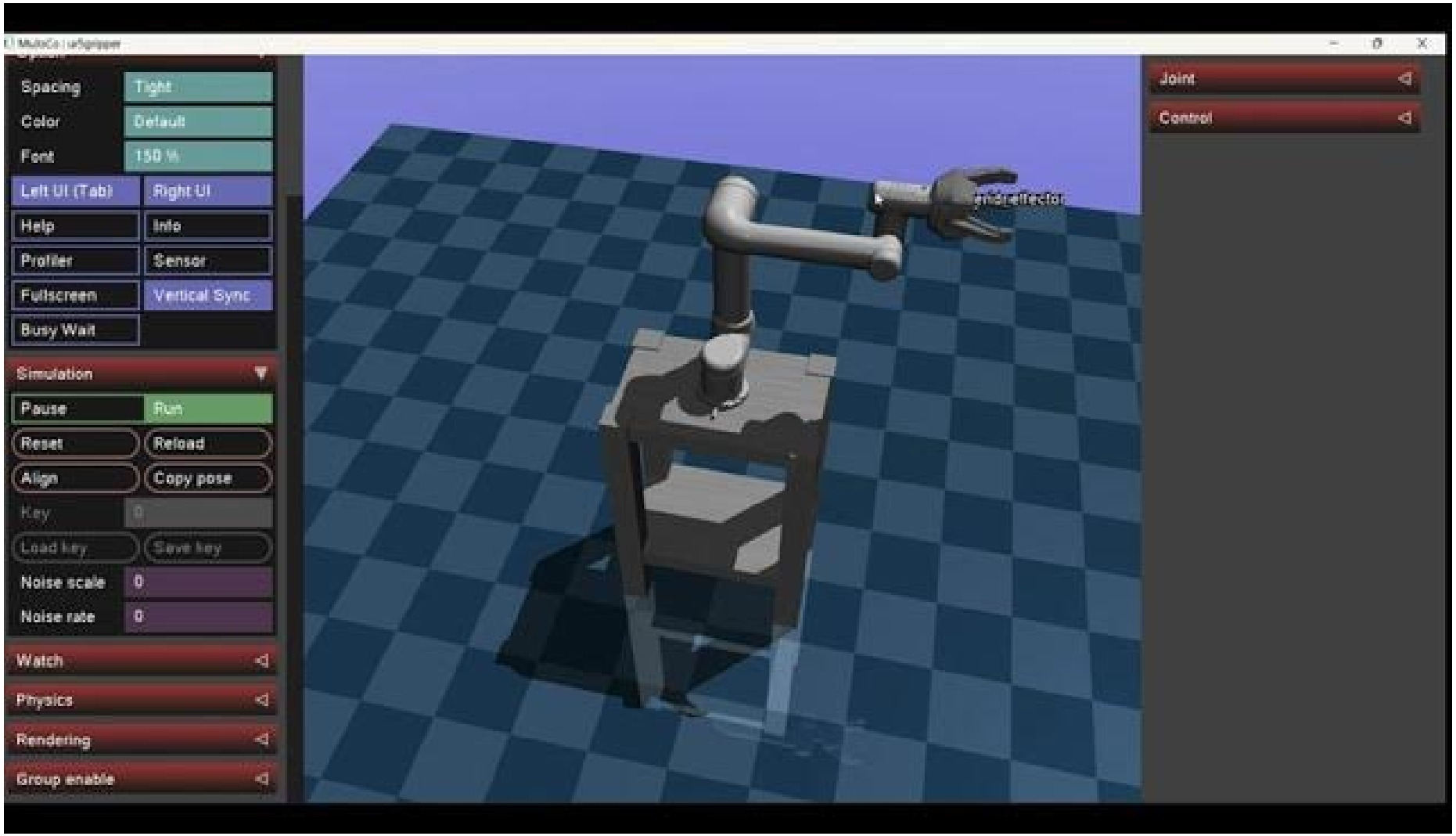
Of the current simulation options, I've found that MuJoCo is the easiest to use and demands the least computational power.

Today's Objective

We are going to create a 2-joint manipulator that can be commanded to a certain position or follow a trajectory using MuJoCo and its Python library.

Let's get started! Feel free to ask any questions you may have as we go along.

MuJoCo Simulation Example



Downloading MuJoCo

MuJoCo can be installed from `mujoco.org`.

Opening the Simulate Environment

Open `simulate.sh` to start the environment.

We need an XML file to describe our robot's structure.

Workshop Repository

https://github.com/willcforte/Introduction-to-MuJoCo-Workshop/

Must be downloaded to your computer:

```
git clone https://github.com/willcforte/Introduction-to-MuJoCo-Workshop/
```

Workshop Files

You can follow along using the instructions in the `exercises` folder.
You can find the finished demo in the `reference` folder.
A copy of this presentation is in the `docs folder`.

Background: Degrees of Freedom (DoFs)

What is a degree of freedom (DoF)?

Creating Our Simulated World in XML

Start with `./exercises/2R_robotic_arm.xml`
To describe the hierarchical structure of a robot, MuJoCo uses XML, a format that uses angular brackets to nest elements within each other.

Exercise 1: Joints and Bodies in MuJoCo

The <worldbody> contains everything in our world. Within it, we can place invisible <body> elements.

In MuJoCo, a body can contain any of three things:

1. A <geometry>, i.e. the meshes that you can see 2. A <joint> which brings a degree of freedom to the body
3. A child <body>, e.g. the next joint in the arm

Background: PID Control

$$\text{PID Signal} \equiv P + I + D$$

$$\equiv \text{Proportional} + \text{Integral} + \text{Derivative}$$

$$= k_p e(t) + k_i \int e(t)dt + k_d \frac{d}{dt} e(t)$$

Exercise 2: Actuating Our Joints (Positional)

Positional servos are easy to use if you want to command a specific angle.
Notice in the XML file that they even have a `kp` attribute. This is because the positional servo contains its own controller that keeps it at a certain angle.

Demo: Controlling Actuators by GUI (Positional)

By uncommenting the `<position>` actuators, we can now control them in the GUI.

Exercise 3: Actuating Our Joints (Motor)

Oftentimes in robotics, we use motors instead of positional servos.
Motors apply accelerations, so we cannot control the particular angle.

Demo: Controlling Actuators by GUI (Motor)

By uncommenting the `<motor>` actuators and commenting the `<position>` ones, we can control them in the GUI.

Exercise 4: Using MuJoCo with Python

Make sure you have these dependencies: `wget`, `python3`, `git`.

Exercise 5: Controlling a Positional Trajectory Using PID

We will be using `position_PID.py`.

Try to experiment with different PID gains!

Exercise 6: Controlling a Motor Trajectory Using PID

We will be using `motor_PID.py`

Try to experiment with different PID gains!

How can We Improve This Program?

Consider how humans command robots to go to a certain *position*.

Robots, on the other hand, work best when given *angles*.

The Robot's Perspective

Current instruction method: we provide angles, which the robot uses to control itself.

State vector:

$$\vec{x} = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

The Human's Perspective

It would be more convenient for us to give a trajectory of positions.

$$\vec{r} = \begin{bmatrix} r_x \\ r_y \end{bmatrix}$$

Background: Inverse Kinematics

Going from positions to angles ($x \rightarrow \vec{r}$) is known as Inverse Kinematics.