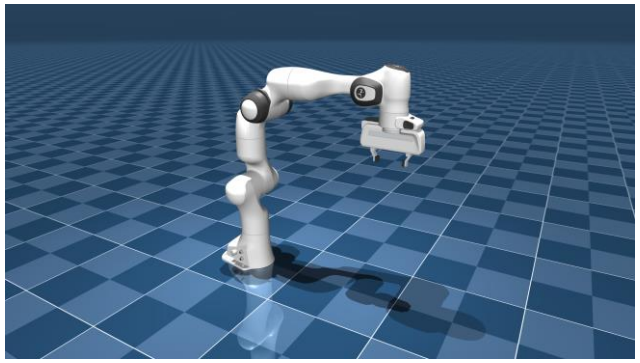
Mujoco + Metaworld Tutorial:

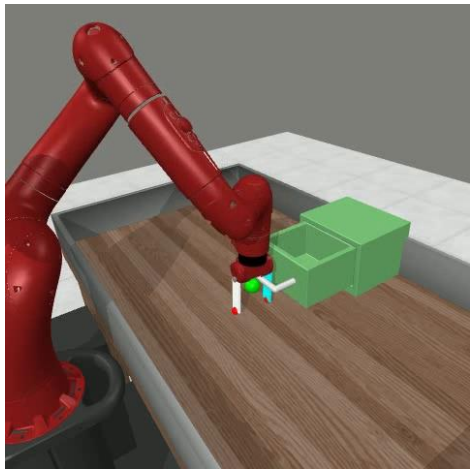
From Physics Simulation to Manipulation Tasks

Embodied AI: Perception, Representation and Action

Outline

MuJoCo → Metaworld → Your Code





- physics engine
- dynamics, contacts, friction, rendering

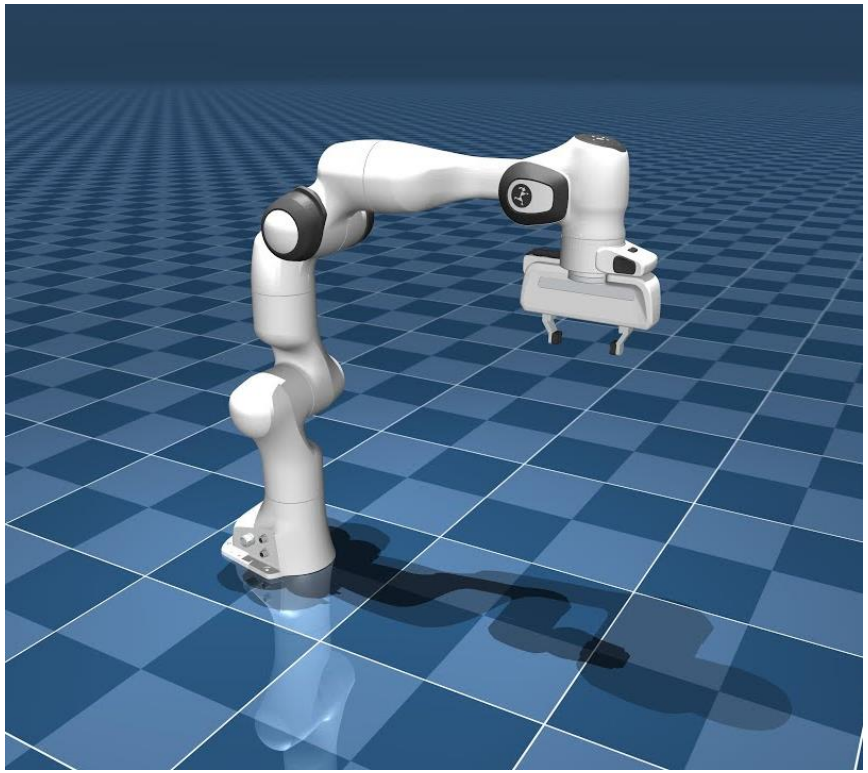
- task suite
- goals, rewards, success metrics, benchmarks

- controller / policy
- chooses actions from observations

MuJoCo

What is it? → **What can MuJoCo be used for?** → **How to use MuJoCo**

- **Multi-Joint dynamics with Contact**
- Real-time simulator for rigid-body dynamics (articulated robots)
- **Strong at contacts**: collision, friction, constraints
- **Input**: model file (MJCF XML) + control signals
- **Output**: states (qpos/qvel), end-effector pose, contacts, rendered frames



Example MuJoCo scene (robot + floor + lighting)

MuJoCo

What is it? → **What can MuJoCo be used for?** → **How to use MuJoCo**

Fast iteration

• Try ideas without risking hardware.

Repeatable experiments

• Same seed & reset → comparable results.
• Easy ablations and baselines.

Visual debugging

•See contacts, failures, and trajectories.
• Tune mass/friction/damping

Important reminder:

Simulation ≠ reality. Differences in contacts, sensors, and dynamics create a sim-to-real gap.
For today, we use simulation as a fast, safe learning and prototyping sandbox.

MuJoCo

What is it? → **What can MuJoCo be used for?** → **How to use MuJoCo**

Model files

- MuJoCo's native format is MJCF — an XML scene description language
- You describe: bodies, joints, geoms (collision/visual), actuators
- MuJoCo can also load URDF (popular in ROS), but it is more limited
- Key intuition: edit XML → the simulated world changes immediately

MJCF is human-editable (change size/mass/friction, then reload)

```
<!-- MJCF (XML) example -->
<mujoco>
  <worldbody>
    <body name="arm" pos="0 0 0">
      <joint name="j0" type="hinge" axis="0 0 1"/>
      <geom type="capsule" size="0.03 0.20"/>
    </body>
  </worldbody>
  <actuator>
    <motor joint="j0" ctrlrange="-1 1"/>
  </actuator>
</mujoco>
```

MuJoCo

What is it? → **What can MuJoCo be used for?** → **How to use MuJoCo**

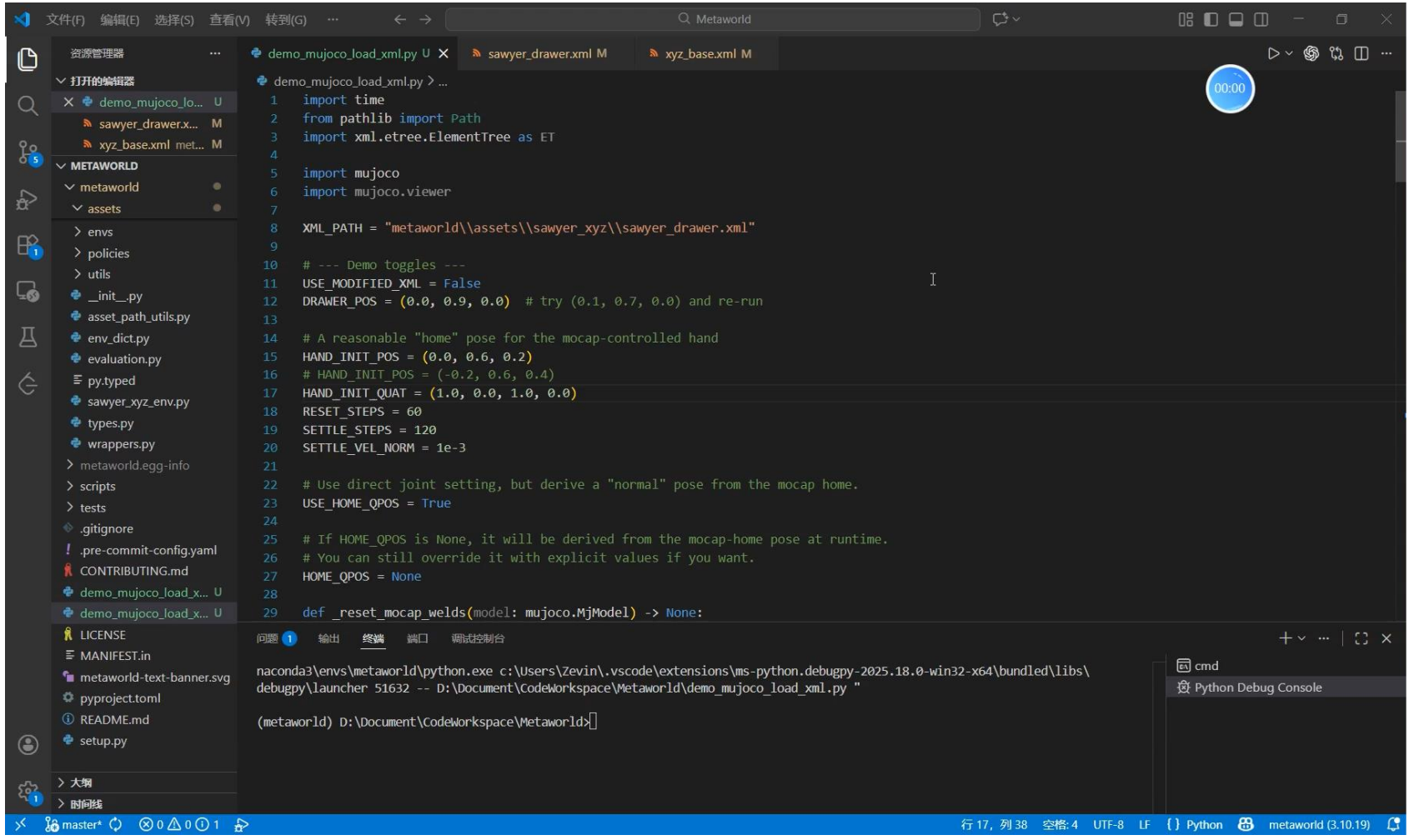
Load Sawyer in MuJoCo

- Load Sawyer MJCF (XML) with MuJoCo: MjModel.from_xml_path → MjData
- Open viewer and step physics in a loop
- Change one XML parameter → reload → see change

Sawyer robot arm (used across many MetaWorld tasks)

MuJoCo

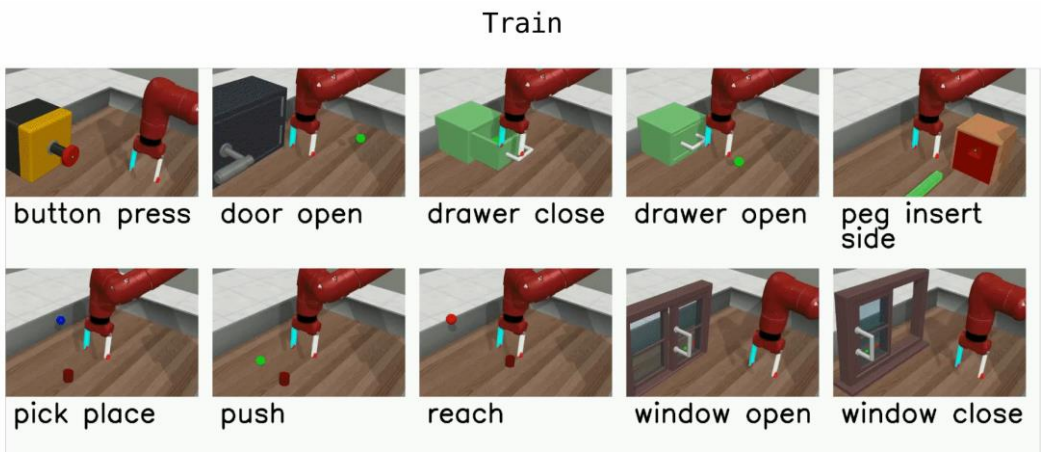
What is it? → What can MuJoCo be used for? → How to use MuJoCo



MetaWorld

What is it? → **What can MuJoCo be used for?** → **How to use MuJoCo**

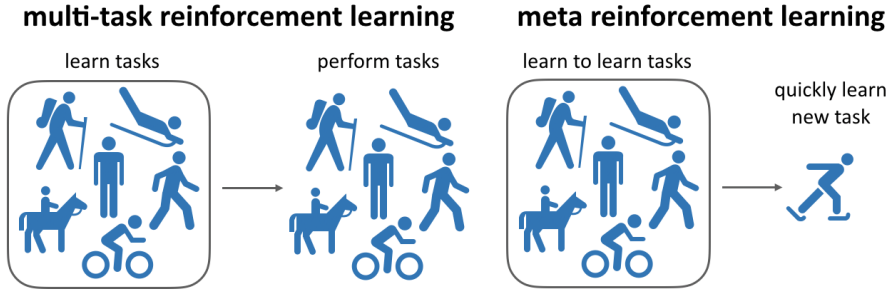
- Open-source benchmark suite for robot manipulation (built on MuJoCo)
- Designed for multi-task RL and meta-RL evaluation
- Includes ~50 distinct manipulation tasks (Reach, Push, Pick-Place, Door, …)
- Standardized success metrics (e.g., success rate) for fair comparisons



MetaWorld

What is it? → **What can MetaWorld be used for?** → **How to use MetaWorld**

- **Action**: learn controllers / policies that move the end-effector to goals
- **Representation**: learn task-agnostic features from diverse interactions
- **Evaluation**: compare algorithms on standardized tasks and success rates
- **Generalization**: train on tasks/goals and test on unseen variations



Multi-task RL vs Meta-RL (intuition)

MetaWorld

What is it? → **What can MetaWorld be used for?** → **How to use MetaWorld**

API intuition

- Create environment
- Call reset() to get an observation
- Each step: send a 4D action → get next obs + reward + success info
- Action meaning: [Δx, Δy, Δz, gripper]

Minimal usage (from official README)

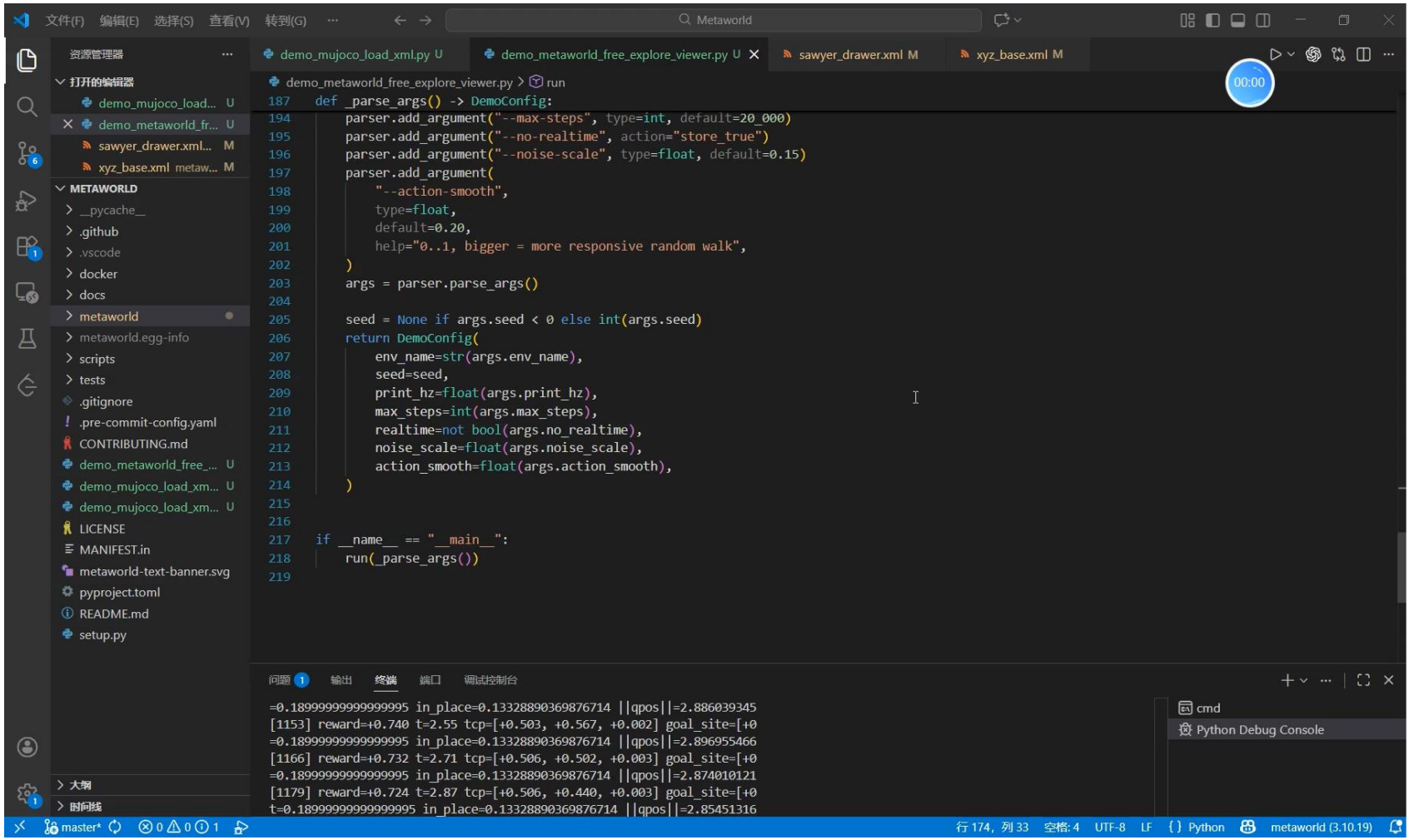
```
import gymnasium as gym
import metaworld

env = gym.make('Meta-World/MT1',
               env_name='reach-v3', seed=42)
obs, info = env.reset()

for t in range(200):
  action = [dx, dy, dz, g]
  obs, r, term, trunc, info = env.step(action)
```

Action = [Δx , Δy , Δz , gripper]

Small Cartesian moves each step → behavior becomes very intuitive to watch.

MetaWorld

What is it? → **What can MetaWorld be used for?** → **How to use MetaWorld**



Takeaways

Summary

- MuJoCo: MJCF/URDF → simulate with mj_step → visualize with viewer
- MetaWorld: MuJoCo + tasks + rewards + success metrics (benchmarks)
- You can control Reach with simple 4D actions: ΔxΔyΔz + gripper
- Next step: replace greedy control with a learned policy (RL / imitation)

Resources

- **MuJoCo docs**: mujoco.readthedocs.io
- **MetaWorld repo**: github.com/Farama-Foundation/Metaworld
- **Meta-World paper**: arxiv.org/abs/1910.10897
- **Easy extension**: try Push / Pick-Place

Q&A prompts

- Installation / common pitfalls
- How to switch tasks
- Using images instead of state
- Sim-to-real considerations

Thanks for listening
Q & A