🔍 Обзор MuJoCo

MuJoCo (Multi-Joint dynamics with Contact) — это физический движок высокой точности, разработанный для моделирования сложных взаимодействий между твёрдыми телами, особенно в контексте биомеханики и робототехники.

MuJoCo описывает сцену (робота, окружение) с помощью XML-файлов, в которых задаются тела, сочленения (joints), геометрия, материалы, сенсоры и другие параметры.

📄 Анализ конфигурационного файла

🤖 `ROBOT = "h1"`

- **Назначение**: Указывает имя робота, который будет загружен в симуляцию.
- **Возможные значения**: `"go2"`, `"b2"`, `"b2w"`, `"h1"`, `"go2w"`, `"g1"` — это модели роботов от Unitree:
  - **Go2** — четвероногий робот (аналог Spot от Boston Dynamics)
  - **B2 / B2W** — четвероногий робот (B2W — версия с колёсами)
  - **H1** — высокий гуманоидный робот (аналог Atlas)
  - **G1** — компактный гуманоид
- **Использование**: Имя используется для формирования пути к XML-сцене.

🏗️ `ROBOT_SCENE = "../unitree_robots/" + ROBOT + "/scene.xml"`

- **Назначение**: Путь к XML-файлу с описанием робота в формате MuJoCo.
- **Структура**: Обычно содержит:
  - `<worldbody>` — тела, геометрия, массы
  - `<actuator>` — моторы/приводы
  - `<sensor>` — IMU, энкодеры, силовые датчики
  - `<contact>` — правила столкновений
- **Пример**: Для `ROBOT = "h1"` загружается `../unitree_robots/h1/scene.xml`

🌐 `DOMAIN_ID = 0`

- **Назначение**: Используется при работе с **DDS (Data Distribution Service)** или **ROS 2**, где `domain_id` определяет изолированную сеть обмена сообщениями.
- **Пояснение**: Если симулятор взаимодействует с внешними узлами (например, реальным роботом или контроллером через ROS 2), `DOMAIN_ID` должен совпадать у всех участников.
- **Значение по умолчанию**: `0` — стандартный домен.

📡 `INTERFACE = "ens33"`

- **Назначение**: Указывает сетевой интерфейс, используемый для UDP- или DDS-коммуникации.
- **Зачем нужно**: При управлении роботом через Wi-Fi или Ethernet важно указать правильный интерфейс (например, `wlan0`, `eth0`, `ens33`).

- **Примечание**: На разных машинах имя интерфейса может отличаться. Можно проверить через `ip a` в Linux.

🏷️ `PRINT_SCENE_INFORMATION = True`

- **Назначение**: Выводит в консоль структуру робота при запуске:
  - Список звеньев (links)
  - Сочленений (joints)
  - Сенсоров (sensors)
- **Полезно для**: Отладки, понимания доступных степеней свободы и сенсорных данных.

🧘 `ENABLE_ELASTIC_BAND = False`

- **Назначение**: Активирует виртуальную "резинку" (пружину), которая может, например, помогать поднимать гуманоида H1 в вертикальное положение.
- **Зачем нужно**: В обучении с подкреплением иногда используют такие вспомогательные силы, чтобы избежать падений на ранних этапах.
- **Аналог**: "Помощь тренера" в RL.

⏱️ `SIMULATE_DT = 0.005`

- **Назначение**: Шаг интеграции физики в симуляции (5 мс → 200 Гц).
- **Важно**: Должен быть **меньше или равен** времени, необходимого для одного шага симуляции (включая вычисления управления).
- **Если слишком велик** → нестабильность, "разрыв" физики.
- **Если слишком мал** → избыточная нагрузка на CPU.

👁️ `VIEWER_DT = 0.02`

- **Назначение**: Частота обновления графического интерфейса (viewer) — 0.02 с = 50 кадров в секунду.
- **Примечание**: Это **не влияет на физику**, только на отображение. Физика может работать на 200 Гц, а рендер — на 50 Гц.
- **Совет**: `VIEWER_DT` обычно ≥ `SIMULATE_DT`, иначе viewer не успеет отображать все шаги.