深入浅出强化学习

交流群号：1012903309

🚩 学而不思则罔，思而不学则殆。

python语法基础：语法基础

c++语法基础：语法基础

linux语法基础：语法基础

ros基础：ros文档

神经网络基础：《动手学习深度学习》

强化学习基础基础：《动手学习强化学习》

github：介绍

注：本开源项目需python基础，c++基础，linux基础，ros、github的基本使用，以及基本控制电机逻辑。本文以致于简单实现强化学习流程，具体技能流程请自行掌握。

视频推荐：

什么是强化学习、强化学习数学原理、onnx是什么

一：学会使用AI。

注：由于整套项目复杂且庞大，一定要学会提问和使用ai自学。

在这里我以deepseek为例，让大家开始入门：

比如，我想知道如何做一只机械狗：

如果我要做四足机械狗，我应该如何使用mlp来强化学习训练四足

2 / 2

已深度思考（用时 22 秒）

嗯，用户从神经网络基础概念直接跳到了具体的四足机器人控制应用，这个跨度不小啊。看来ta是个行动派，刚理解MLP就想动手实践了。不过用MLP做强化学习控制四足机械狗确实是个有趣的方向。

用户可能是个工科学生或机器人爱好者，从"小白"的自我描述看，应该刚入门机器学习。但ta提问很具体，说明已经查阅过相关资料，只是不确定如何落地。深层需求可能是想了解从理论到实践的完整链路，而不仅仅是概念解释。

上次我详细解释了MLP原理，这次需要侧重工程实现。四足控制涉及运动学、动力学和实时控制，直接用原始MLP可能效率不高。不过既然用户明确问MLP，就先围绕这个展开，后续再补充其他网络结构的对比。

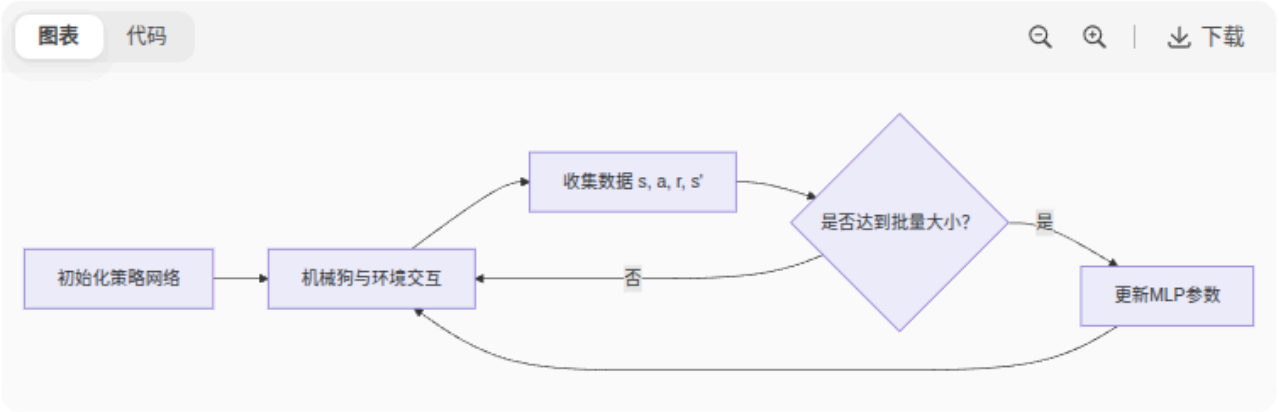
关键要强调两点：一是强化学习框架如何与MLP结合，二是四足控制的特殊性。需要分层次说明：先讲强化学习基本要素（状态、动作、奖励函数），再讲如何用MLP作为策略网络，最后讨论四足特有的动作空间设计问题。

用户可能没意识到机械狗控制的复杂性。比如步态生成、平衡控制这些子问题，是否需要分层训练？单层MLP可能不够。但作为起点，还是建议ta从简化模型开始，比如先训练单腿摆动，再扩展到四足协同。

硬件实现方面得提醒风险。直接在实体机器人训练容易损坏设备，应该先仿真。可以推荐PyBullet或MuJoCo，但得说明MuJoCo现在开源了，不像以前需要付费。

这回答也是蛮有意思

6. 训练流程



7. 关键优化技术

- **课程学习（Curriculum Learning）**：
  先训练简单任务（如站立），再逐步增加难度（行走→小跑→避障）。
- **域随机化（Domain Randomization）**：
  在仿真中随机化地面摩擦、电机噪声等，提升迁移到实机的鲁棒性。
- **提前终止（Early Termination）**：
  当机械狗摔倒时提前结束本轮训练，加速学习。
- **动作平滑（Action Smoothing）**：
  在动作输出层加入低通滤波器，避免电机抖动。

虽然ai不能帮助我们完全解决问题，但是可以帮助我们学习基础内容。

总结

用MLP训练四足机械狗的核心流程：

1. **建模**：定义状态、动作、奖励函数；
2. **构建MLP策略网络**：输入状态 → 输出关节指令；
3. **选择RL算法**：优先PPO或SAC；
4. **仿真训练**：在PyBullet/Isaac Gym中迭代优化；
5. **迁移到实体**：加入域随机化提升鲁棒性；
6. **部署优化**：模型轻量化以满足实时控制。

> ⚠️ **重要提醒**：首次尝试建议从**2D平面仿真**开始（如仅控制俯仰方向），再扩展到全3D环境。可参考开源项目如 legged_gym 快速上手。

这里也提到了我们将要使用的开源代码

二：必要工具。

1.安装conda

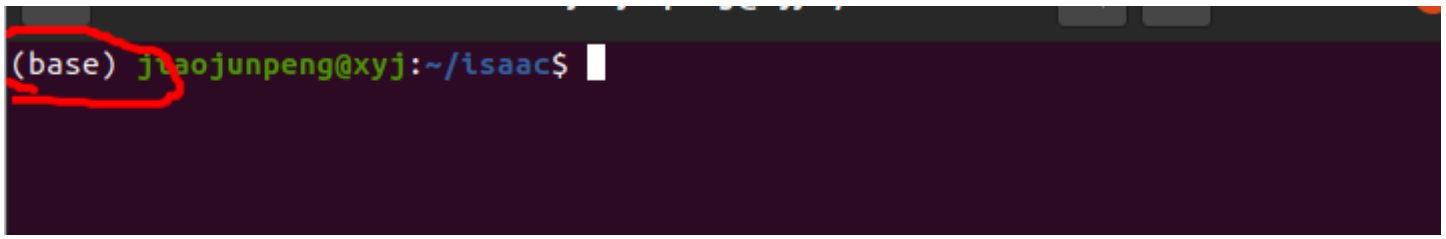
1.安装python虚拟环境miniconda

代码块

```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
chmod +x Miniconda3-latest-Linux-x86_64.sh
./Miniconda3-latest-Linux-x86_64.sh
```

一直按Enter直到输入yes

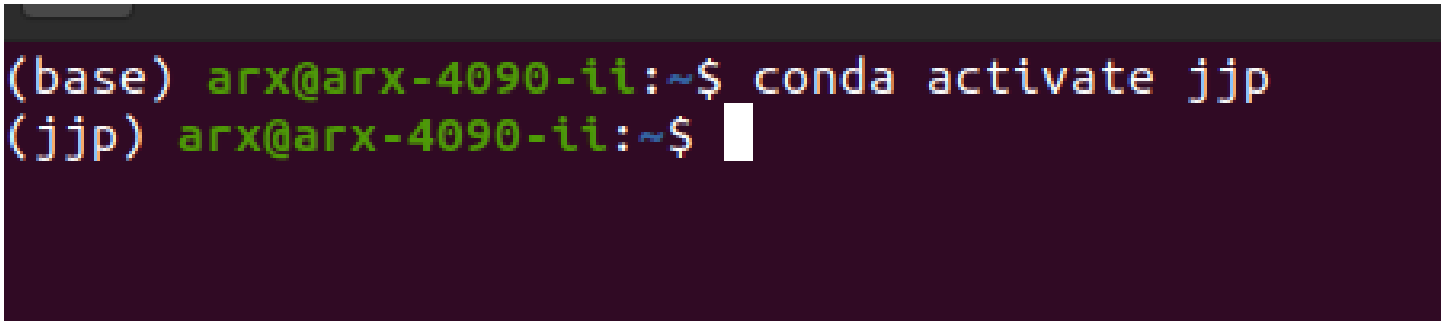
重新打开终端会出现



创建虚拟环境：例如名称为jjp（可以起别的名字，这里博主自己用顺手了）

代码块

```
conda create -n jjp python=3.8.10                #创建虚拟环境
conda activate jjp                               #进入虚拟环境
```



同时下载cuda，附带conda和cuda的区别，避免混淆

特性	Conda	CUDA
是什么	开源包管理和环境管理系统	NVIDIA 推出的并行计算平台和API
作用层级	软件层（管理Python环境、库版本）	硬件层（调用GPU进行计算）
核心目标	解决依赖冲突，创建隔离环境	利用GPU进行高性能并行计算
依赖关系	可管理CUDA工具包	是GPU运行深度学习的基础

2.安装cuda

输入以下命令，查看 GPU 支持的最高 CUDA 版本。这里显示的是 12.2 ，这意味着，安装的 CUDA 版本必须 <= 12.2 。

代码块

```
1  nvidia-smi
```

cunda官方网址

我这里选择了12.1.0

below, and be sure to check www.nvidia.com/drivers for more recent production drivers appropriate for your hardw

Download Latest CUDA Toolkit Learn More about

Latest Release

CUDA Toolkit 12.6.3 (November 2024), Versioned Online Documentation

Archived Releases

CUDA Toolkit 12.6.2 (October 2024), Versioned Online Documentation
CUDA Toolkit 12.6.1 (August 2024), Versioned Online Documentation
CUDA Toolkit 12.6.0 (August 2024), Versioned Online Documentation
CUDA Toolkit 12.5.1 (July 2024), Versioned Online Documentation
CUDA Toolkit 12.5.0 (May 2024), Versioned Online Documentation
CUDA Toolkit 12.4.1 (April 2024), Versioned Online Documentation
CUDA Toolkit 12.4.0 (March 2024), Versioned Online Documentation
CUDA Toolkit 12.3.2 (January 2024), Versioned Online Documentation
CUDA Toolkit 12.3.1 (November 2023), Versioned Online Documentation
CUDA Toolkit 12.3.0 (October 2023), Versioned Online Documentation
CUDA Toolkit 12.2.2 (August 2023), Versioned Online Documentation
CUDA Toolkit 12.2.1 (July 2023), Versioned Online Documentation
CUDA Toolkit 12.2.0 (June 2023), Versioned Online Documentation
CUDA Toolkit 12.1.1 (April 2023), Versioned Online Documentation
CUDA Toolkit 12.1.0 (February 2023), Versioned Online Documentation
CUDA Toolkit 12.0.1 (January 2023), Versioned Online Documentation
CUDA Toolkit 12.0.0 (December 2022), Versioned Online Documentation
CUDA Toolkit 11.8.0 (October 2022), Versioned Online Documentation
CUDA Toolkit 11.7.1 (August 2022), Versioned Online Documentation
CUDA Toolkit 11.7.0 (May 2022), Versioned Online Documentation
CUDA Toolkit 11.6.2 (March 2022), Versioned Online Documentation
CUDA Toolkit 11.6.1 (February 2022), Versioned Online Documentation
CUDA Toolkit 11.6.0 (January 2022), Versioned Online Documentation
CUDA Toolkit 11.5.2 (February 2022), Versioned Online Documentation
CUDA Toolkit 11.5.1 (November 2021), Versioned Online Documentation
CUDA Toolkit 11.5.0 (October 2021), Versioned Online Documentation

接下来选择自己的系统版本

CUDA Toolkit 12.1 Downloads

Select Target Platform

Click on the green buttons that describe your target platform. Only supported platforms will be shown. By downloading and using the software, you agree to fully comply with the terms and conditions of the CUDA EULA.

Operating System	Linux Windows
Architecture	x86_64 ppc64le arm64-sbsa aarch64-jetson
Distribution	CentOS Debian Fedora KylinOS OpenSUSE RHEL Rocky SLES Ubuntu WSL-Ubuntu
Version	18.04 20.04 22.04
Installer Type	deb (local) deb (network) runfile (local)

Download Installer for Linux Ubuntu 20.04 x86_64

The base installer is available for download below.

下面两句就是我们打开终端安装的命令

安装cuda版本pytorch

```
代码块
1   conda activate jjp
2   conda install pytorch torchvision pytorch-cuda=12.1.0 -c pytorch -c nvidia
```

如何验证是否安装cuda，**Ctrl+Alt+T**打开终端

```
代码块
1   conda activate jjp
2   python
```

在进入的python终端中输入

```
代码块
1   import torch
2   print(torch.__version__)
3   print(torch.cuda.is_available())
```

```
(base) jiaojunpeng@xyj:~$ python
Python 3.12.7 | packaged by Anaconda, Inc. | (main, Oct  4 2024, 13:27:36) [GCC
11.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> print(torch.__version__)
2.5.1+cu124
>>> print(torch.cuda.is_available())
True
```

若输出了版本号和True则为安装成功

2.安装仿真

主流仿真软件：gazebo 、isaac gym、isaac sim、mujoco、genesis

由于机器学习需要大量数据集，所以在训练过程中我们一般只使用isaac gym和isaac sim（两者是新旧的区别isaac gym已经停止维护了），genesis是新出的仿真器（看宣传片非常厉害,但是不知道什么可以用上完整版:宣传视频），在本次开源中，我只使用isaac gym作为训练仿真器，mujoco作为仿真验证。

1.安装pytorch(深度学习库)

安装pip

代码块

```
sudo apt install python3-pip
```

安装依赖：

代码块

```
pip install torchvision -i https://pypi.tuna.tsinghua.edu.cn/simple
pip install torch -i https://pypi.tuna.tsinghua.edu.cn/simple
pip install pyquaternion -i https://pypi.tuna.tsinghua.edu.cn/simple
pip install pyyaml -i https://pypi.tuna.tsinghua.edu.cn/simple
pip install pexpect -i https://pypi.tuna.tsinghua.edu.cn/simple
pip install matplotlib -i https://pypi.tuna.tsinghua.edu.cn/simple
pip install einops -i https://pypi.tuna.tsinghua.edu.cn/simple
pip install tqdm -i https://pypi.tuna.tsinghua.edu.cn/simple
pip install packaging -i https://pypi.tuna.tsinghua.edu.cn/simple
pip install h5py -i https://pypi.tuna.tsinghua.edu.cn/simple
pip install ipython -i https://pypi.tuna.tsinghua.edu.cn/simple
pip install getkey -i https://pypi.tuna.tsinghua.edu.cn/simple
pip install wandb -i https://pypi.tuna.tsinghua.edu.cn/simple
pip install chardet -i https://pypi.tuna.tsinghua.edu.cn/simple
pip install matplotlib -i https://pypi.tuna.tsinghua.edu.cn/simple
pip install numpy==1.23.2 -i https://pypi.tuna.tsinghua.edu.cn/simple
pip install h5py_cache -i https://pypi.tuna.tsinghua.edu.cn/simple
pip install opencv-python -i https://pypi.tuna.tsinghua.edu.cn/simple
pip install tensorboard -i https://pypi.tuna.tsinghua.edu.cn/simple
pip install onnxruntime
pip install mujoco-python-viewer
```

2.安装isaacgym

isaacgym.tar.xz
130.62MB

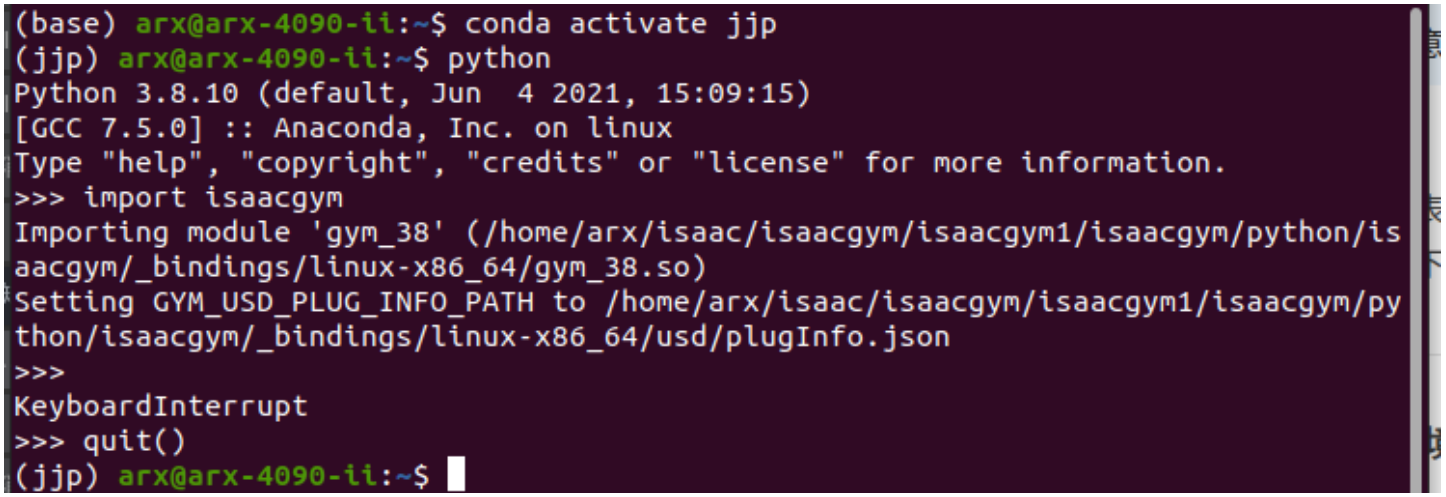
在此目录下打开终端进入虚拟环境并安装库

```
代码块
1   git clone https://github.com/isaac-sim/IsaacGymEnvs.git
2   conda activate jjp
3   pip install -e ./isaacgym1/isaacgym/python
4   pip install -e ./IsaacGymEnvs
```

验证安装库

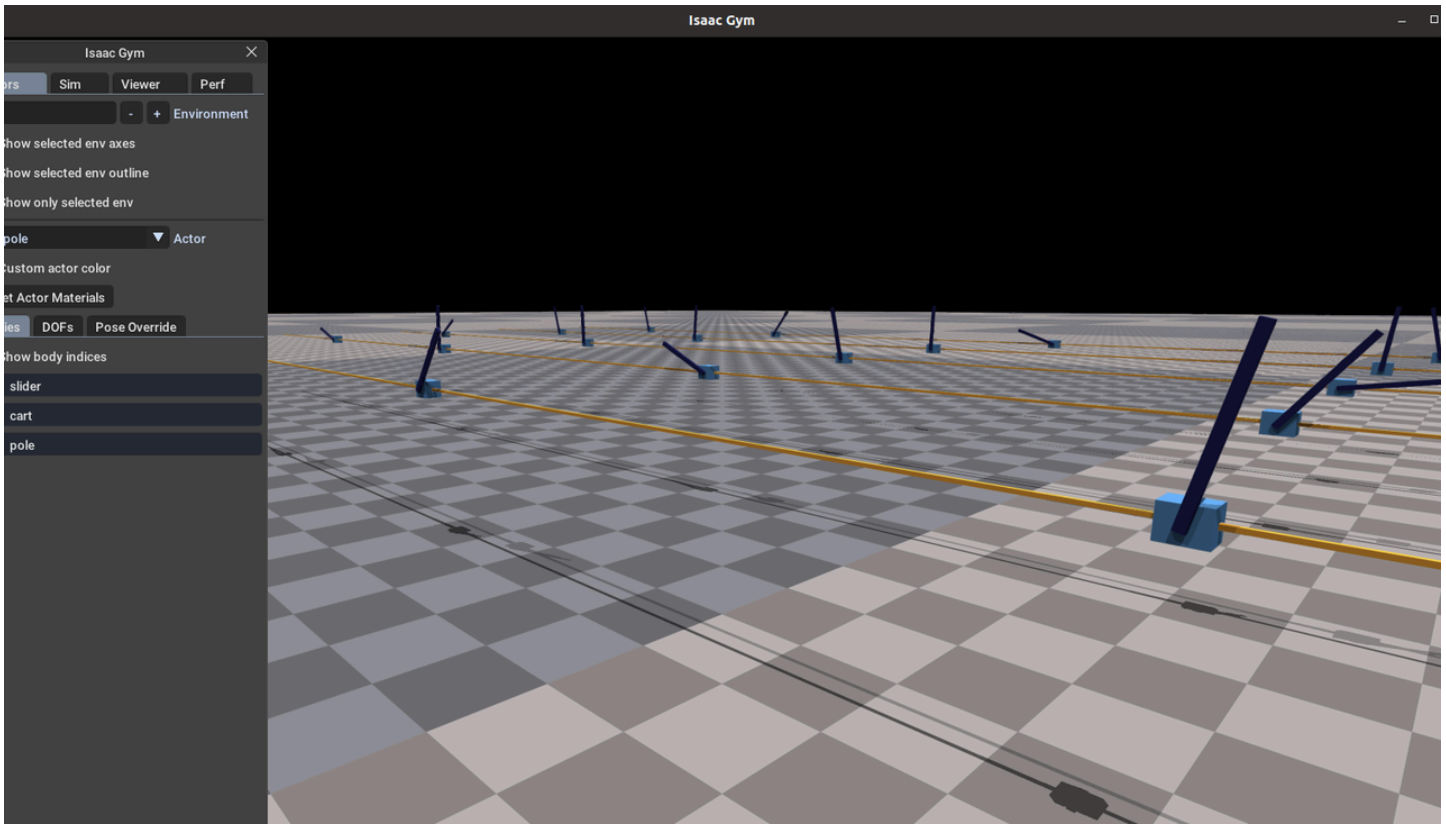
```
代码块
1   python
```

```
代码块
1   import isaacgym
2   quit()
```

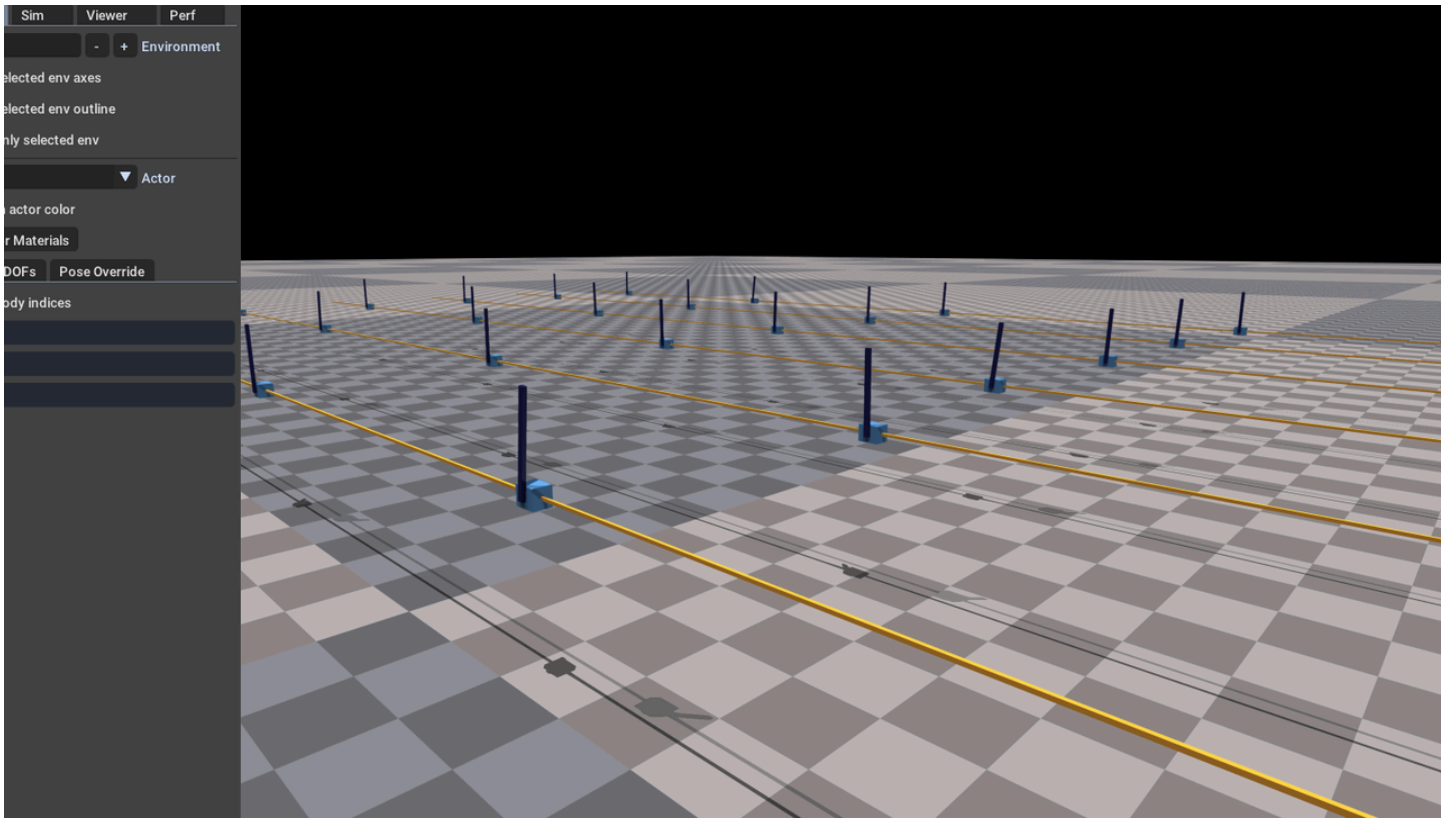


python运行倒立摆强化学习模型

```
代码块
1   python ./IsaacGymEnvs/isaacgymenvs/train.py task=Cartpole
```



可以看到过了一会，即可看见倒立摆趋于稳定



3.安装仿真环境mujoco

打开终端

代码块

```
1   wget https://mujoco.org/download/mujoco210-linux-x86_64.tar.gz
```

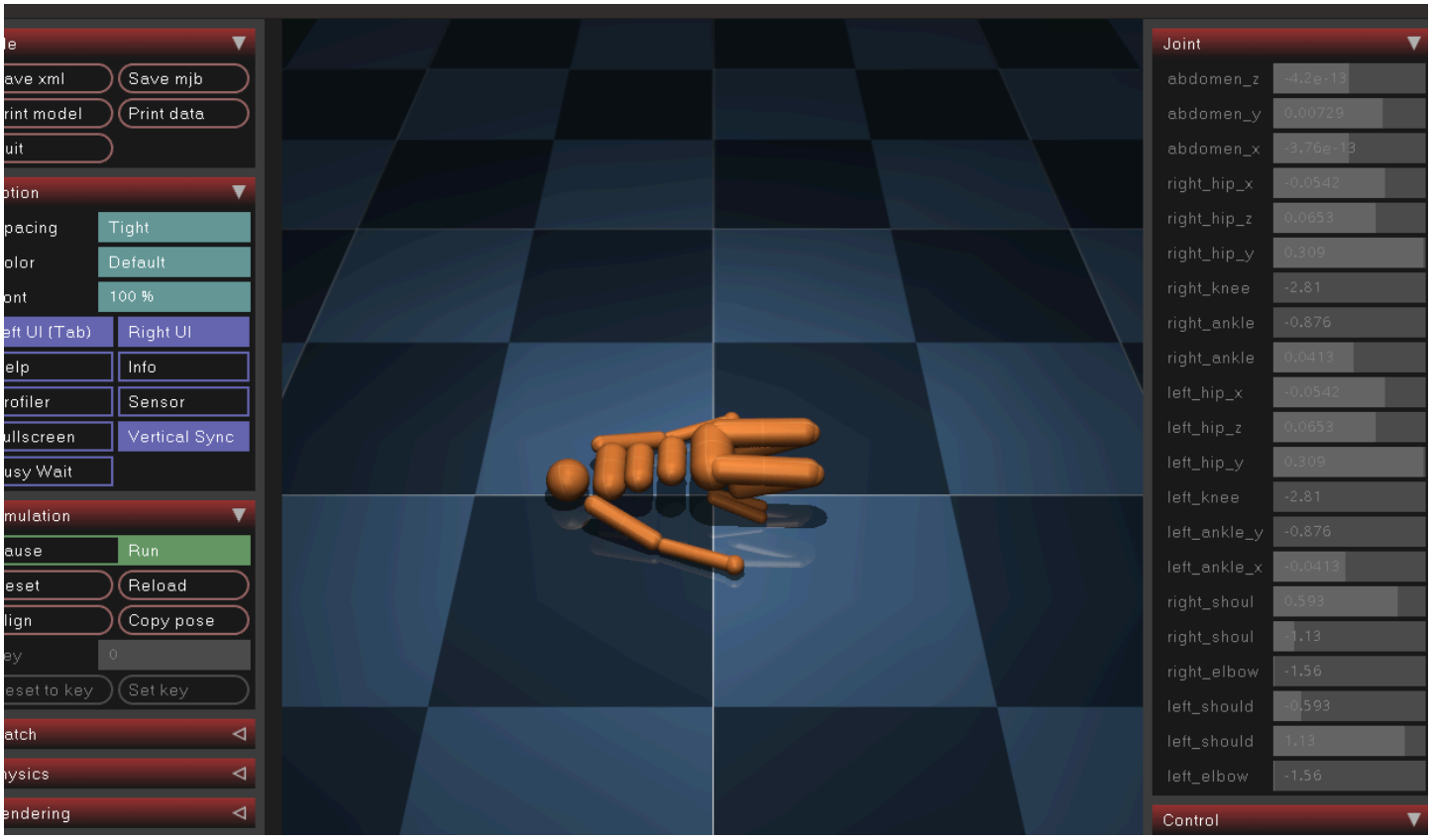
```
tar -zxvf mujoco210-linux-x86_64.tar.gz
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/.mujoco/mujoco210/bin
```

测试

代码块

```
cd mujoco210/bin
./simulate ../model/humanoid.xml
```

你将会看见这样一个界面



二.legged_gym

修改代码：

legged_gym.zip
68.19MB

1代码介绍

代码块

```
https://github.com/leggedrobotics/legged_gym.git        #源代码地址
```

本教程基于此源码改动讲解，**代码隶属于**: Robotic Systems Lab, ETH Zurich

2.代码讲解(注：开源代码并不能部署，需要经过一系列随机化和验证)

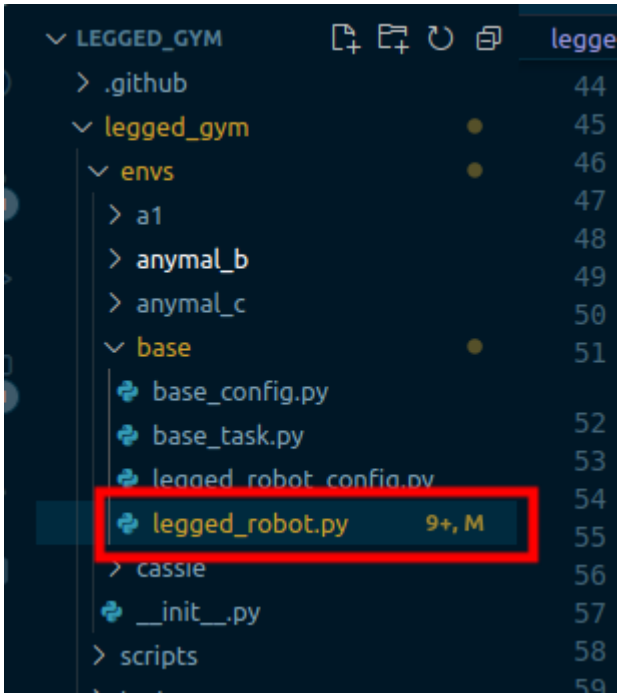
加：rsl_rl(ppo实现)

强化学习：近端策略优化算法（proximal policy optimization，**PPO**）

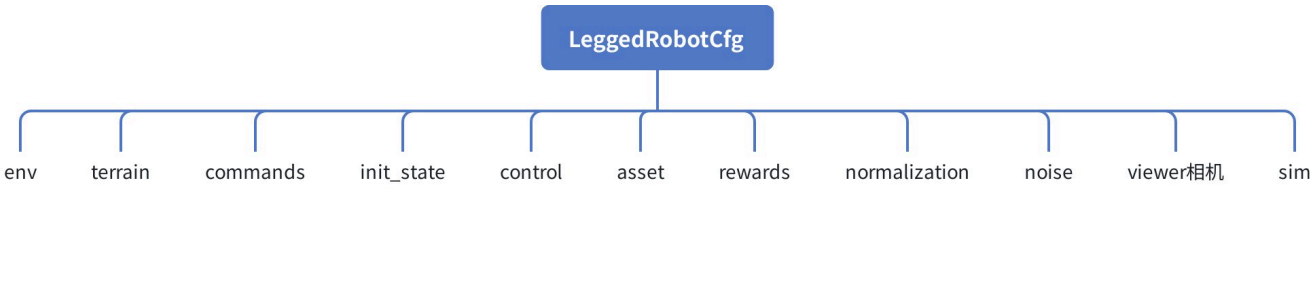
在readme中下载rsl_rl.

```
4. Install rsl_rl (PPO implementation)
   - Clone https://github.com/leggedrobotics/rsl_rl
   -  `cd rsl_rl && git checkout v1.0.2 && pip install -e .`
5. Install legged_gym
    - Clone this repository
   - `cd legged_gym && pip install -e .`
```

仿真搭建重要代码：



1.legged_robot_config文件，包含奖励函数



```
class env:
    num_envs = 4096
    num_observations = 235
    num_privileged_obs = None # if
    num_actions = 12
    env_spacing = 3.  # not used w
    send_timeouts = True # send ti
    episode_length_s = 20 # episod
```

Env

- num_envs---训练智能体数量
- num_observations---观察状态数量（网络输入数据长度）
- num_privileged_obs---如果是None则step返回的privileged_obs是None,如果不是则返回privileged_obs

  （一般在使用非对称网络结构时使用）
- num_actions---模型输出维度（机器人关节自由度）
- env_spacing---智能体间距只在平地（plane）中使用
- send_timeouts---是否显示智能体（感觉这个没啥用）
- episode_length_s---每个智能体存活最高时间（s）

```
class terrain:
    mesh_type = 'trimesh' # "heightfield" # none, plane, heightfield or trimesh
    horizontal_scale = 0.1 # [m]
    vertical_scale = 0.005 # [m]
    border_size = 25 # [m]
    curriculum = True
    static_friction = 1.0
    dynamic_friction = 1.0
    restitution = 0.
    # rough terrain only:
    measure_heights = True
    measured_points_x = [-0.8, -0.7, -0.6, -0.5, -0.4, -0.3, -0.2, -0.1, 0., 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8] #
    measured_points_y = [-0.5, -0.4, -0.3, -0.2, -0.1, 0., 0.1, 0.2, 0.3, 0.4, 0.5]
    selected = False # select a unique terrain type and pass all arguments
    terrain_kwargs = None # Dict of arguments for selected terrain
    max_init_terrain_level = 5 # starting curriculum state
    terrain_length = 8.
    terrain_width = 8.
    num_rows= 10 # number of terrain rows (levels)
    num_cols = 20 # number of terrain cols (types)
    # terrain types: [smooth slope, rough slope, stairs up, stairs down, discrete]
    terrain_proportions = [0.1, 0.1, 0.35, 0.25, 0.2]
    # trimesh only:
    slope_treshold = 0.75 # slopes above this threshold will be corrected to vertical surfaces
```

terrain

- mesh_type---地形none, plane（平地）, heightfield（高度图） or trimesh（崎岖地形）
- horizontal_scale---水平距离（m）相当于照片分辨率
- vertical_scale---垂直距离（m）
- border_size---边缘距离，地形距离边界的距离（m）

- curriculum---课程开关（从低难度到高难度）
- static_friction---静摩擦
- dynamic_friction---动摩擦
- restitution---恢复力
- 仅崎岖地形
  - measure_heights
    - 地形感知
      - 这个影响模型输入维度会在原有的基础上加入感知地形的数据。开启之后还要注意_get_noise_scale_vec函数中对地形感知噪声范围添加
      - 例:unitree a1，默认num_observations=235,其中0-48是传感器，48-235是地形，将此置False则num_observations要调整为48
  - measured_points_x---测量位置X
  - measured_points_y---测量位置Y
  - selected---选择唯一的地形类型并传递所有参数
  - terrain_kwargs---所选地形的参数（字典）
  - max_init_terrain_level---开始课程状态
  - terrain_length---地形长度
  - terrain_width---地形宽度
  - num_rows---地形行数
  - num_cols---地形列数
  - terrain_proportions---地形类型：[Smooth Slope、Rough Slope、Stairs Up、Stairs Down、Discrete]
  - slope_treshold---高于此阈值的坡度将被校正为垂直表面

```
class commands:
    curriculum = False
    max_curriculum = 1.
    num_commands = 4 # default: lin_vel_x, li
    resampling_time = 10. # time before comma
    heading_command = True # if true: compute
    class ranges:
        lin_vel_x = [-1.0, 1.0] # min max [m/
        lin_vel_y = [-1.0, 1.0]   # min max [
        ang_vel_yaw = [-1, 1]     # min max [r
        heading = [-3.14, 3.14]
```

Commands

- curriculum---速度课程
- max_curriculum---课程最大速度
- num_commands=4---默认值：lin_vel_x、lin_vel_y、ang_vel_yaw、heading（在heading模式下ang_vel_yaw根据heading误差重新计算）
- resampling_time---切换命令的时间
- heading_command---如果为 true：根据航向误差计算 ang vel 命令
- ranges
  - lin_vel_x---min max [m/s]
  - lin_vel_y---min max [m/s]
  - ang_vel_yaw---min max [rad/s]
  - heading---rad

```
class init_state:
    pos = [0.0, 0.0, 1.] # x,y,z [m]
    rot = [0.0, 0.0, 0.0, 1.0] # x,y,z,w [quat]
    lin_vel = [0.0, 0.0, 0.0]  # x,y,z [m/s]
    ang_vel = [0.0, 0.0, 0.0]  # x,y,z [rad/s]
    default_joint_angles = { # target angles when
        "joint_a": 0.,
        "joint_b": 0.}
```

init_state

- pos---x,y,z [m]
- rot---x,y,z,w [quat]
- lin_vel---x,y,z [m/s]
- ang_vel---x,y,z [rad/s]
- default_joint_angles={ "joint_a": 0.,"joint_b": 0.}标准站立关节角度

```
class control:
    control_type = 'P' # P: position, V: velocity, T: torques
    # PD Drive parameters:
    stiffness = {'joint_a': 10.0, 'joint_b': 15.}  # [N*m/rad]
    damping = {'joint_a': 1.0, 'joint_b': 1.5}     # [N*m*s/rad]
    # action scale: target angle = actionScale * action + defaultAngle
    action_scale = 0.5
    # decimation: Number of control action updates @ sim DT per policy DT
    decimation = 4
```

control

- control_type---P: pd控制器, V:速度控制, T: 力矩控制

- PD参数
  - stiffness=={'joint_a': 10.0, 'joint_b': 15.}---刚度(P)[N*m/rad]
  - damping= {'joint_a': 1.0, 'joint_b': 1.5}---阻尼(D)[N*m*s/rad]
- action_scale
  - action scale: target angle = actionScale * action + defaultAngle---动作缩放（降低网络输出占比）
- decimation推理执行次数

```
class asset:
    file = ""
    name = "legged_robot"  # actor name
    foot_name = "None" # name of the feet bodies, used to index body state and contact force tensors
    penalize_contacts_on = []
    terminate_after_contacts_on = []
    disable_gravity = False
    collapse_fixed_joints = True # merge bodies connected by fixed joints. Specific fixed joints can be kept by adding " <... dont_collapse="true">
    fix_base_link = False # fixe the base of the robot
    default_dof_drive_mode = 3 # see GymDofDriveModeFlags (0 is none, 1 is pos tgt, 2 is vel tgt, 3 effort)
    self_collisions = 0 # 1 to disable, 0 to enable...bitwise filter
    replace_cylinder_with_capsule = True # replace collision cylinders with capsules, leads to faster/more stable simulation
    flip_visual_attachments = True # Some .obj meshes must be flipped from y-up to z-up

    density = 0.001
    angular_damping = 0.
    linear_damping = 0.
    max_angular_velocity = 1000.
    max_linear_velocity = 1000.
    armature = 0.
    thickness = 0.01
```

asset

- file--urdf路径
- name---机器人名字.
- foot_name---脚体的名称，用于为身体状态和接触力张量编制索引
- penalize_contacts_on---惩罚跟地面有接触的linx
- terminate_after_contacts_on---模型的link接触到地面就终止
- disable_gravity---禁用重力
- collapse_fixed_joints---合并由固定关节连接的实体。可以通过添加 “<... dont_collapse=” true“ 来保留特定的固定关节>
- fix_base_link---固定机器人base_link在空中
- default_dof_drive_mode---请参阅 GymDofDriveModeFlags（0 表示无，1 表示 pos tgt，2 表示 vel tgt，3 表示effort）
- self_collisions---是否开始自身碰撞
- replace_cylinder_with_capsule---用胶囊替换碰撞圆柱体，实现更快/更稳定的模拟
- flip_visual_attachments---某些 .obj 网格必须从 y 轴向上翻转到 z 轴向上
- density---密度
- angular_damping---旋转阻尼

- linear_damping---线性阻尼
- max_angular_velocity---最大角速度
- max_linear_velocity---最大线速度
- armature--关节惯性
- thickness---厚度

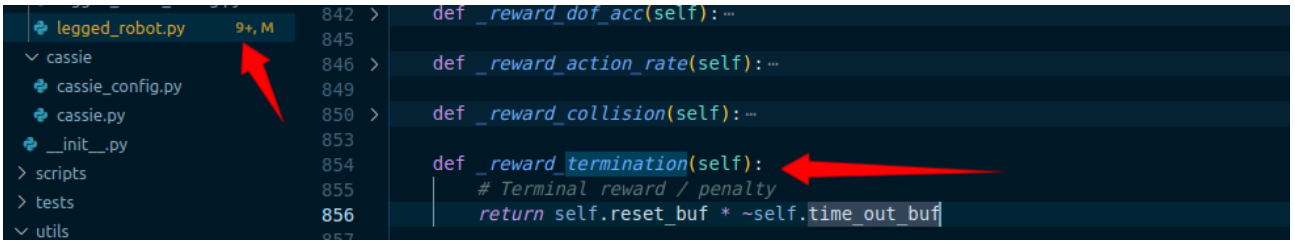
```
class rewards:
    class scales:
        termination = -0.0
        tracking_lin_vel = 1.0
        tracking_ang_vel = 0.5
        lin_vel_z = -2.0
        ang_vel_xy = -0.05
        orientation = -0.
        torques = -0.00001
        dof_vel = -0.
        dof_acc = -2.5e-7
        base_height = -0.
        feet_air_time =  1.0
        collision = -1.
        feet_stumble = -0.0
        action_rate = -0.01
        stand_still = -0.

    only_positive_rewards = True # if true negative total rewards are
    tracking_sigma = 0.25 # tracking reward = exp(-error^2/sigma)
    soft_dof_pos_limit = 1. # percentage of urdf limits, values above
    soft_dof_vel_limit = 1.
    soft_torque_limit = 1.
    base_height_target = 1.
    max_contact_force = 100. # forces above this value are penalized
```

rewards

以第一个奖励函数为例子，相关奖励函数写在主要文件中如下图。

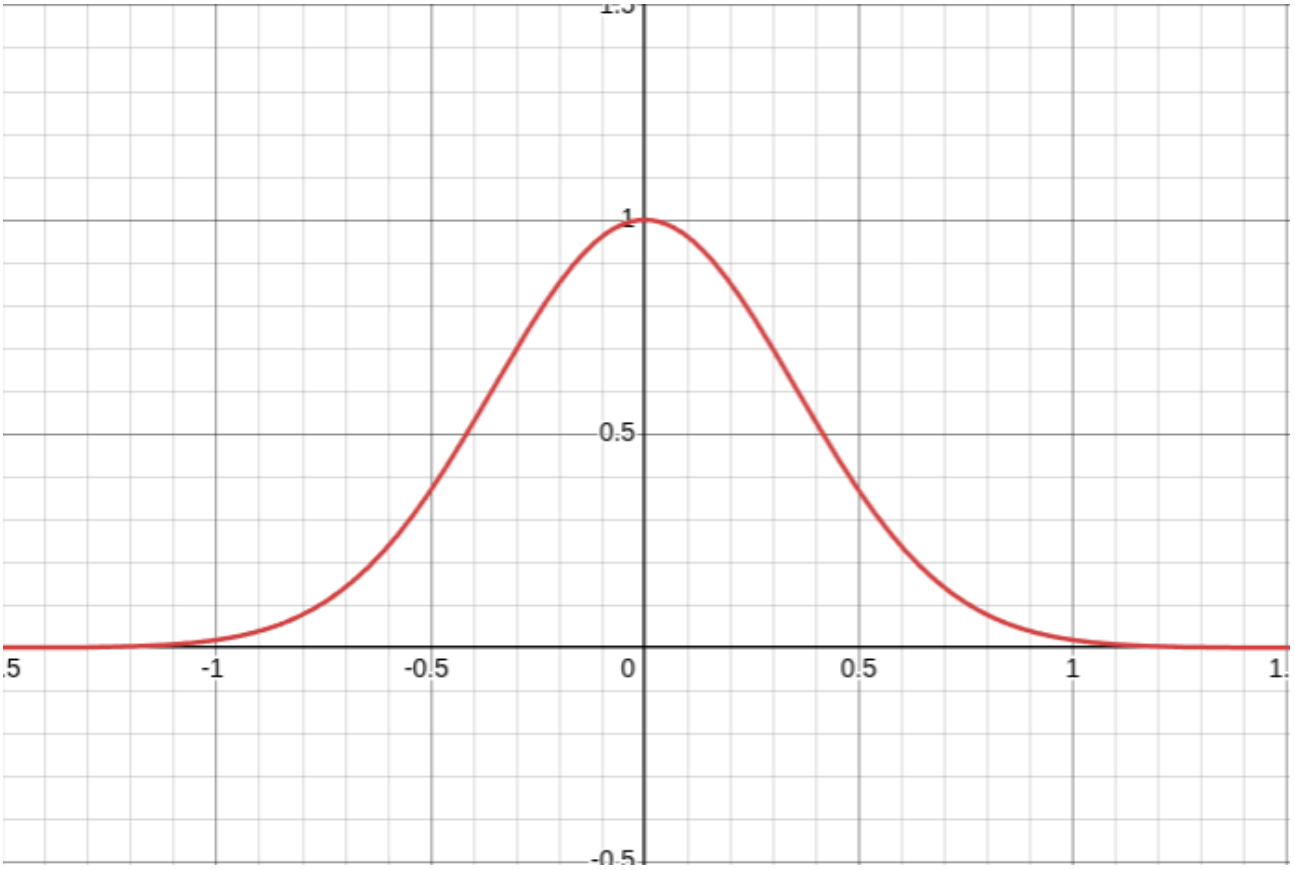
缩放scales（**权重**）,和对应的奖励函数相乘

- termination
  - 终止处罚1*termination



- tracking_lin_vel（**重要函数**）

- $exp(\frac{-x^2}{0.25})$，这里误差用x表示,非常完美的正态分布，标准差这里对应的速度为[-1,1]



- tracking_ang_vel（**重要函数**）
  - 同上 $exp(\frac{-x^2}{0.25})$，但权重为0.5
- lin_vel_z（**重要函数**）
  - z轴速度平方,在这里权重是-2，用来抑制z轴速度，同时帮助了身高奖励收敛

```python
def _reward_lin_vel_z(self):
    # Penalize z axis base linear velocity
    return torch.square(self.base_lin_vel[:, 2])
```

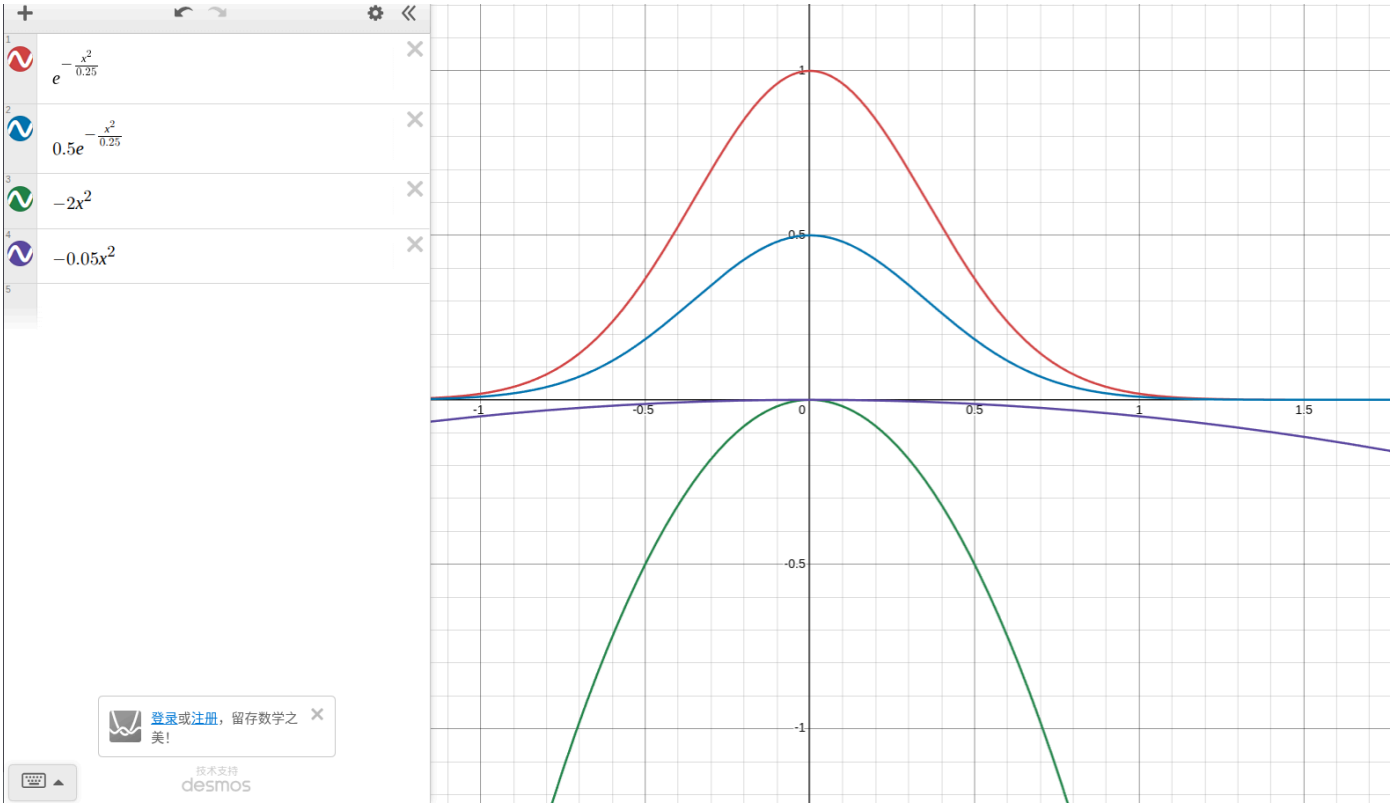
- ang_vel_xy（**重要函数**）
  - 对x,y轴角速度的惩罚
- orientation
  - 奖励
- torques
  - 力矩惩罚
- dof_vel
  - 关节速度惩罚

        - dof_acc
            - 关节加速度惩罚
        - base_height
            - 身高奖励
        - feet_air_time
            - 脚离开地面时间惩罚
        - collision
            - 惩罚选定刚体上的碰撞
        - feet_stumble(官方开源删除了奖励函数)
            - 惩罚脚撞击垂直表面，和下面函数类似

```
def _reward_stumble(self):
    # Penalize feet hitting vertical surfaces
    return torch.any(torch.norm(self.contact_forces[:, self.feet_indices, :2], dim=2) >\
        5 *torch.abs(self.contact_forces[:, self.feet_indices, 2]), dim=1)
```

        - action_rate
            - 动作变化率，让动作更连续，不突变
        - stand_still
            - 惩罚距离标准位置的差距（在速度小的时候）
    - only_positive_rewards
        - 如果 true negative 总奖励被削减为零（避免提前终止问题）
    - tracking_sigma
        - 跟踪奖励 = exp（-error^2/sigma）
    - soft_dof_pos_limit
        - Urdf 限制的百分比，高于此限制的值将受到惩罚
    - soft_dof_vel_limit
        - 关节速度限制
    - soft_torque_limit
        - 扭矩限制
    - base_height_target
        - base_link目标高度
    - max_contact_force
        - 高于此值的力会受到惩罚（配合*_reward_feet_contact_forces奖励使用*）

主要奖励函数图像：




```
class normalization:
    class obs_scales:
        lin_vel = 2.0
        ang_vel = 0.25
        dof_pos = 1.0
        dof_vel = 0.05
        height_measurements = 5.0
    clip_observations = 100.
    clip_actions = 100.
```

normalization

- obs_scales观测缩放
  - lin_vel 线速度缩放
  - ang_vel 角速度缩放
  - dof_pos关节位置缩放
  - dof_vel 关节速度缩放
  - height_measurements
    - 测量的高度
- clip_observations
  - 观测裁减

- clip_actions
  - 动作裁减

```
class noise:
    add_noise = True
    noise_level = 1.0 # scales other values
    class noise_scales:
        dof_pos = 0.01
        dof_vel = 1.5
        lin_vel = 0.1
        ang_vel = 0.2
        gravity = 0.05
        height_measurements = 0.1
```

noise

- add_noise添加噪声
- noise_level噪声等级
- noise_scales噪声缩放
  - dof_pos 关节位置噪声
  - dof_vel 关节速度噪声
  - lin_vel 线速度噪声
  - ang_vel 角速度噪声
  - gravity imu噪声
  - height_measurements

```
# viewer camera:
class viewer:
    ref_env = 0
    pos = [10, 0, 6]  # [m]
    lookat = [11., 5, 3.]  # [m]
```

viewer相机

- ref_env（无用）
- pos 相机位置
- lookat 相机看向位置

```
class sim:
    dt =  0.005
    substeps = 1
    gravity = [0., 0. ,-9.81]  # [m/s^2]
    up_axis = 1  # 0 is y, 1 is z

    class physx:
        num_threads = 10
        solver_type = 1  # 0: pgs, 1: tgs
        num_position_iterations = 4
        num_velocity_iterations = 0
        contact_offset = 0.01  # [m]
        rest_offset = 0.0   # [m]
        bounce_threshold_velocity = 0.5 #0.5 [m/s]
        max_depenetration_velocity = 1.0
        max_gpu_contact_pairs = 2**23 #2**24 -> needed for 8000 envs and more
        default_buffer_size_multiplier = 5
        contact_collection = 2 # 0: never, 1: last sub-step, 2: all sub-steps (default=2)
```

sim

- dt---真运行步长
- substeps---多少步一次反馈计算
- gravity---重力场
- up_axis---0 is y, 1 is z
- physx（isaac使用的物理引擎参数）
  - num_threads---线程数
  - solver_type---求解器
  - num_position_iterations---位置迭代次数，影响仿真精度
  - num_velocity_iterations---速度迭代次数，影响仿真精度
  - contact_offset---触点偏移
  - rest_offset---重置偏移
  - bounce_threshold_velocity---弹跳阈值速度
  - max_depenetration_velocity---最大穿透速度?
  - max_gpu_contact_pairs---最大 GPU 接触对数
  - default_buffer_size_multiplier---默认缓冲区大小
  - contact_collection---联系人集合

```python
class LeggedRobotCfgPPO(BaseConfig):
    seed = 1
    runner_class_name = 'OnPolicyRunner'
    class policy:
        init_noise_std = 1.0
        actor_hidden_dims = [512, 256, 128]
        critic_hidden_dims = [512, 256, 128]
        activation = 'elu' # can be elu, relu, selu, crelu, lrelu, tanh, sigmoid
        # only for 'ActorCriticRecurrent':
        # rnn_type = 'lstm'
        # rnn_hidden_size = 512
        # rnn_num_layers = 1
```

2.PPO

- speed:随机种子
- init_noise_std:噪声标准差
- actor_hidden_dims：策略网络隐藏层
- critic_hidden_dims：价值网络隐藏层
- activation：激活函数

```python
class algorithm:
    # training params
    value_loss_coef = 1.0
    use_clipped_value_loss = True
    clip_param = 0.2
    entropy_coef = 0.01
    num_learning_epochs = 5
    num_mini_batches = 4 # mini batch size = num_envs*nsteps / nminibatches
    learning_rate = 1.e-3 #5.e-4
    schedule = 'adaptive' # could be adaptive, fixed
    gamma = 0.99
    lam = 0.95
    desired_kl = 0.01
    max_grad_norm = 1.
```

这里主要是ppo算法，不需要更改，所以这里不再赘述，有兴趣可以自行了解ppo算法

```
class runner:
    policy_class_name = 'ActorCritic'
    algorithm_class_name = 'PPO'
    num_steps_per_env = 24 # per iteration
    max_iterations = 1500 # number of policy updates

    # logging
    save_interval = 50 # check for potential saves every this many iterations
    experiment_name = 'test'
    run_name = ''
    # load and resume
    resume = False
    load_run = -1 # -1 = last run
    checkpoint = -1 # -1 = last saved model
    resume_path = None # updated from load_run and chkpt
```

- runner
  - policy_class_name:网络结构名称
  - algorithm_class_name：算法名字
  - num_steps_per_env：网络输入步数
  - max_iterations：最大迭代次数
  - save_interval：每隔多少轮保存一次
  - run_name：自定义路径名字（不好用）
  - resume ：运行模型
  - 下面的不好用，我就不赘述参数作用了

三.开始训练

1.认识基本参数

注：！！！！为了防止显卡的显存崩溃，建议在复杂地形中使用2048个agent（智能体）

即num_envs。

num_observations参数为模型输入：

235=3(线速度)+3（角速度）+3（重力投影）+3（命令控制）+12（关节位置）+12（关节速度）

+12（上次网络输出）+187（地形高度观测）

这里我们如果是训练盲狗的话，线速度和地形高度是无法获取的。所以，num_observations应该改为235-187-3

```python
class LeggedRobotCfg(BaseConfig):
    class env:
        num_envs = 4096
        num_observations = 235
        num_privileged_obs = None # if not None a priviledge_obs_buf will be returne
        num_actions = 12
        env_spacing = 3.  # not used with heightfields/trimeshes
        send_timeouts = True # send time out information to the algorithm
        episode_length_s = 20 # episode length in seconds
```

观测值的顺序和长度

```python
def compute_observations(self):
    """ Computes observations
    """
    self.obs_buf = torch.cat((  self.base_lin_vel * self.obs_scales.lin_vel,
                                self.base_ang_vel  * self.obs_scales.ang_vel,
                                self.projected_gravity,
                                self.commands[:, :3] * self.commands_scale,
                                (self.dof_pos - self.default_dof_pos) * self.obs_scales.dof_pos,
                                self.dof_vel * self.obs_scales.dof_vel,
                                self.actions
                                ),dim=-1)
    # add perceptive inputs if not blind
    if self.cfg.terrain.measure_heights:
        heights = torch.clip(self.root_states[:, 2].unsqueeze(1) - 0.5 - self.measured_heights, -1, 1.) * self.obs_scales.height_measu
        self.obs_buf = torch.cat((self.obs_buf, heights), dim=-1)
    # add noise if needed
    if self.add_noise:
        self.obs_buf += (2 * torch.rand_like(self.obs_buf) - 1) * self.noise_scale_vec
```

这里的a1观测数据维度是235,因为默认开启了地形高度感知(measure_heights)，关闭则是红框标出的观测。

```python
def compute_observations(self):
    """ Computes observations
    """
    self.obs_buf = torch.cat((
        # self.base_lin_vel * self.obs_scales.lin_vel,
```

同时删除第一个线速度观测、关闭地形检测：measure_heights = False

```python
class LeggedRobotCfg(BaseConfig):
    class env:
        num_envs = 2048
        num_observations = 235-187-3
        num_privileged_obs = None # if not None a priviledge_obs_buf will be returned
        num_actions = 12
        env_spacing = 3.  # not used with heightfields/trimeshes
        send_timeouts = True # send time out information to the algorithm
        episode_length_s = 20 # episode length in seconds

    class terrain:
        mesh_type = 'plane' # "heightfield" # none, plane, heightfield or trimesh
        horizontal_scale = 0.1 # [m]
        vertical_scale = 0.005 # [m]
        border_size = 25 # [m]
        curriculum = True
```

将legged_robot_config.py文件中的terrain的mesh_type修改为*plane*，开始训练

2.开始训练

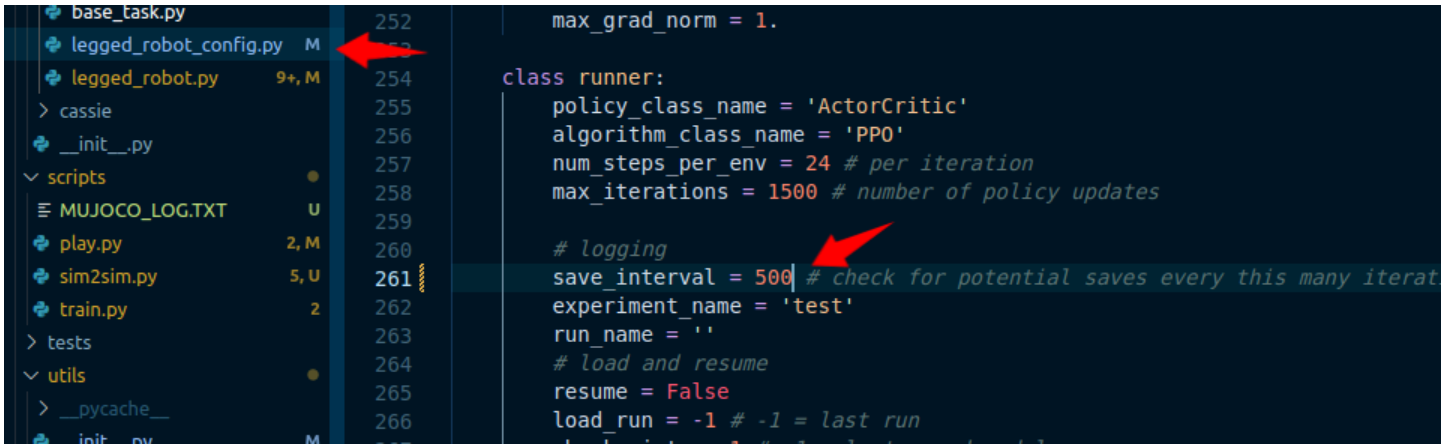
在scripts文件夹下打开终端进入虚拟环境jjp，开始训练a1

让我们来看看这个继承类干了些什么

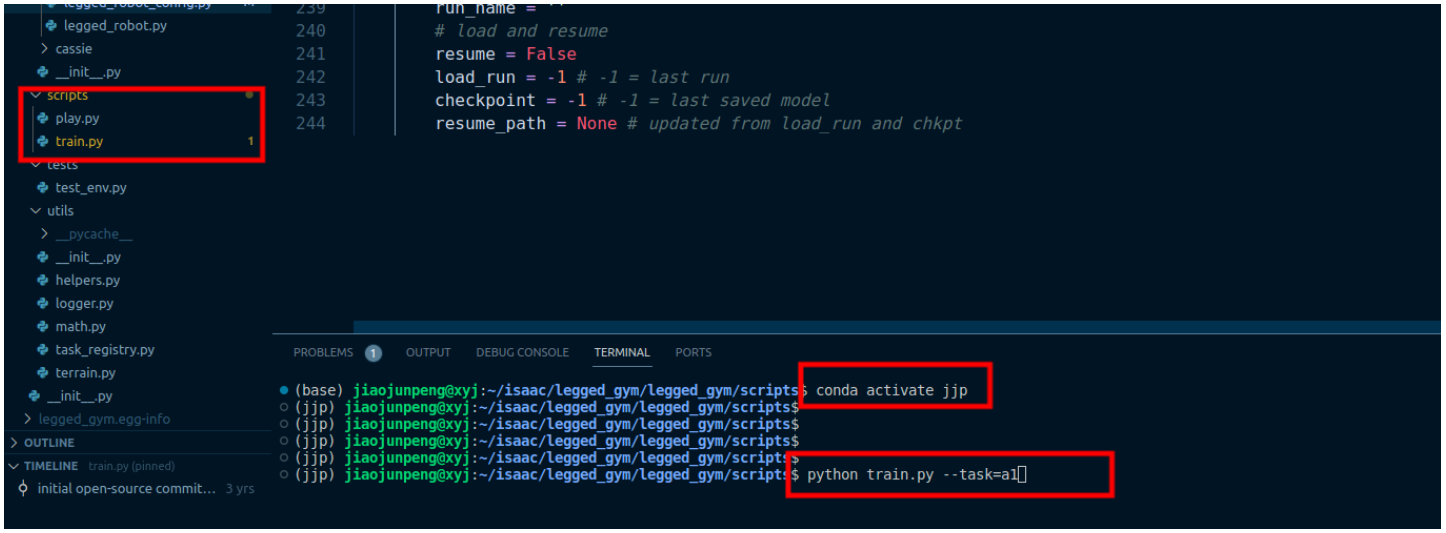
设置了初始位置，以及初始的默认关节位置和kp kd等.....

1.开始训练

为了避免大家储存空间占用过大，我们在config文件中，找到对应变量并修改



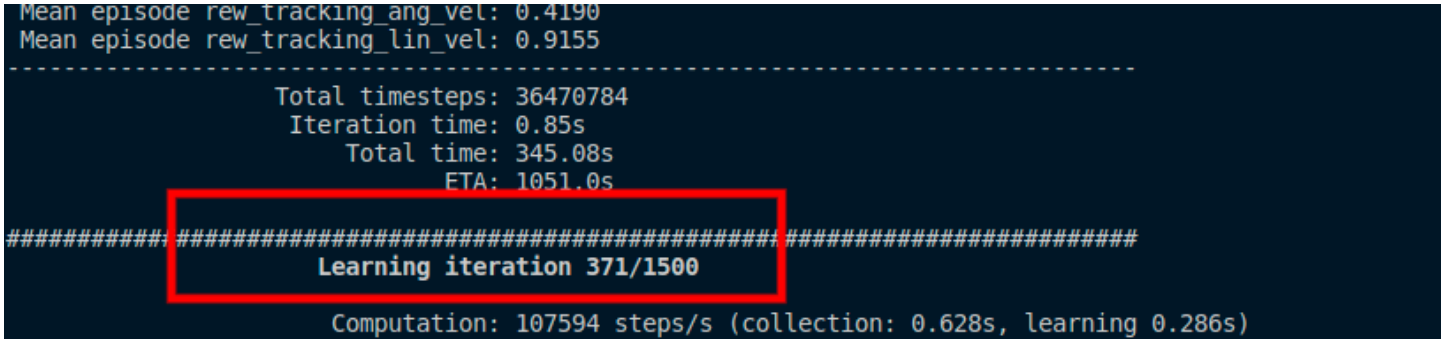
之后便可以开始训练：

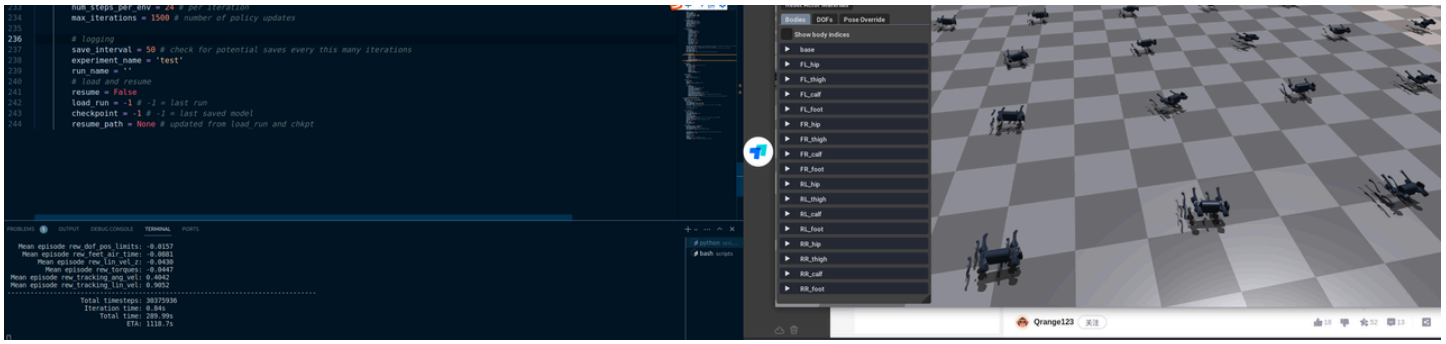


代码块

```
conda activate jjp
python train.py --task=a1
```

运行即可看到开始训练轮次以及gym训练界面，**按V停止渲染加速训练**





如果看到模型的腿不正确,修改设置。

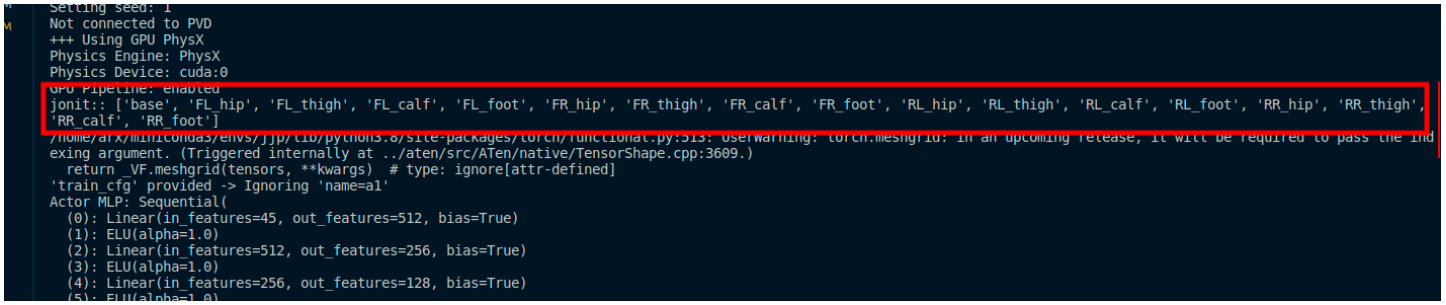
robot_config文件下

flip_visual_attachments =True或False

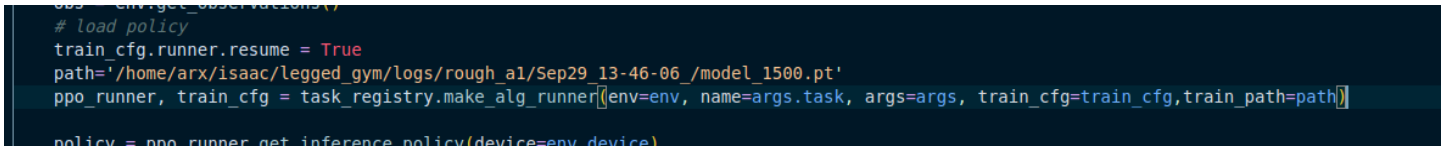
查看urdf关节顺序：

```python
def _create_envs(self):
    """ Creates environments:
        1. loads the robot URDF/MJCF asset,
        2. For each environment
           2.1 creates the environment,
           2.2 calls DOF and Rigid shape properties callbacks,
           2.3 create actor with these properties and add them to the env
        3. Store indices of different bodies of the robot
    """
    asset_path = self.cfg.asset.file.format(LEGGED_GYM_ROOT_DIR=LEGGED_GYM_ROOT_DIR)
    asset_root = os.path.dirname(asset_path)
    asset_file = os.path.basename(asset_path)

    asset_options = gymapi.AssetOptions()
    asset_options.default_dof_drive_mode = self.cfg.asset.default_dof_drive_mode
    asset_options.collapse_fixed_joints = self.cfg.asset.collapse_fixed_joints
    asset_options.replace_cylinder_with_capsule = self.cfg.asset.replace_cylinder_with_capsule
    asset_options.flip_visual_attachments = self.cfg.asset.flip_visual_attachments
    asset_options.fix_base_link = self.cfg.asset.fix_base_link
    asset_options.density = self.cfg.asset.density
    asset_options.angular_damping = self.cfg.asset.angular_damping
    asset_options.linear_damping = self.cfg.asset.linear_damping
    asset_options.max_angular_velocity = self.cfg.asset.max_angular_velocity
    asset_options.max_linear_velocity = self.cfg.asset.max_linear_velocity
    asset_options.armature = self.cfg.asset.armature
    asset_options.thickness = self.cfg.asset.thickness
    asset_options.disable_gravity = self.cfg.asset.disable_gravity

    robot_asset = self.gym.load_asset(self.sim, asset_root, asset_file, asset_options)
    self.num_dof = self.gym.get_asset_dof_count(robot_asset)
    self.num_bodies = self.gym.get_asset_rigid_body_count(robot_asset)
    dof_props_asset = self.gym.get_asset_dof_properties(robot_asset)
    rigid_shape_props_asset = self.gym.get_asset_rigid_shape_properties(robot_asset)

    # save body names from the asset
    body_names = self.gym.get_asset_rigid_body_names(robot_asset)
    self.dof_names = self.gym.get_asset_dof_names(robot_asset)
    print("jonit::",body_names)
    self.num_bodies = len(body_names)
    self.num_dofs = len(self.dof_names)
```

即可看到：



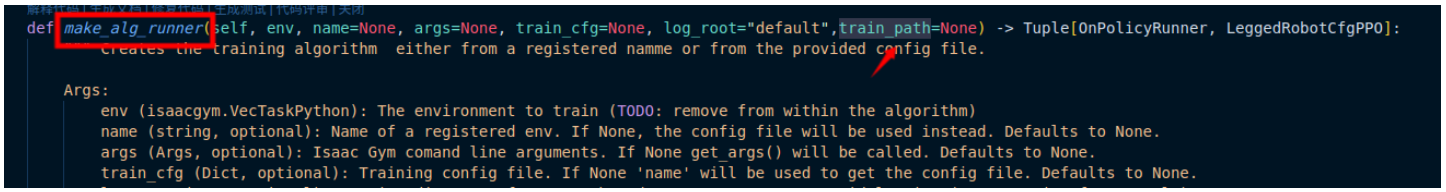
为了使得查看之前训练的效果，我们修改play.py中为：

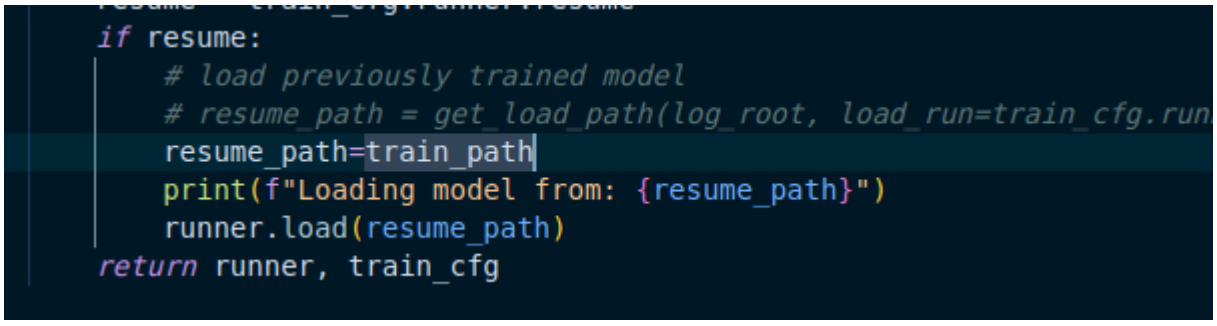


代码块

```
1       path='/home/arx/isaac/legged_gym/logs/rough_a1/Sep29_13-46-
    06_/model_1500.pt'
2       ppo_runner, train_cfg = task_registry.make_alg_runner(env=env,
    name=args.task, args=args, train_cfg=train_cfg,train_path=path)
```

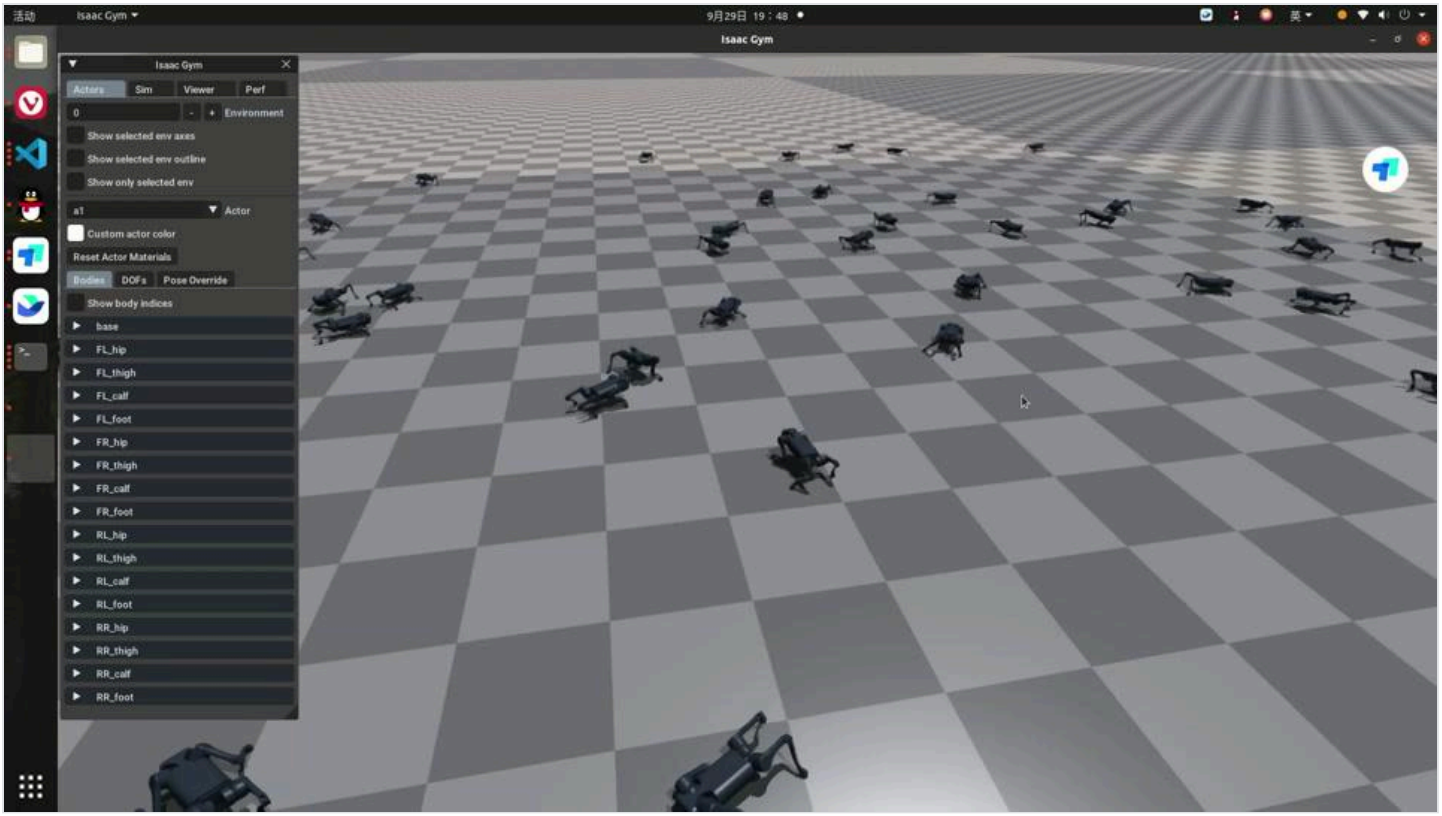
然后修改这个函数



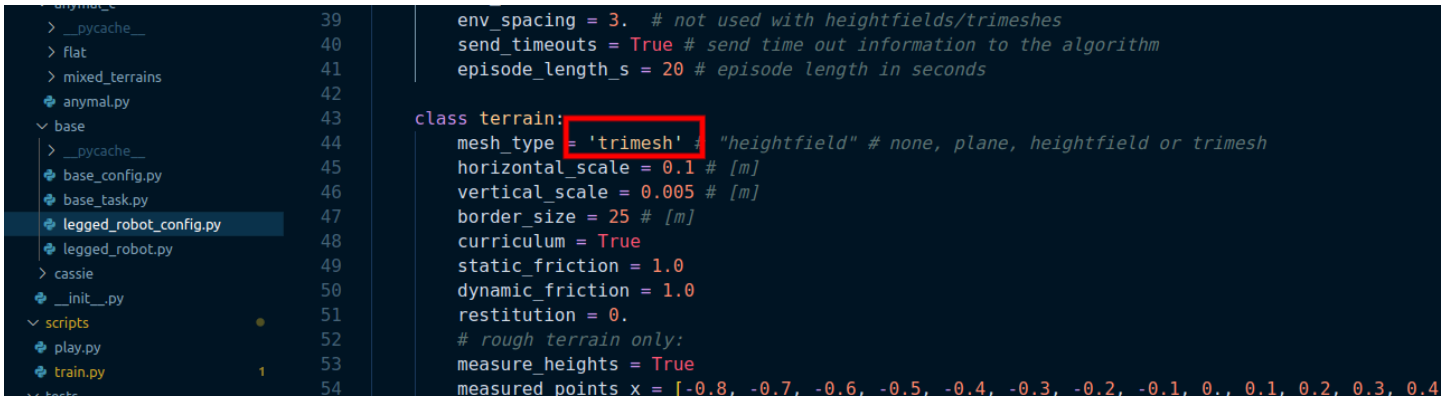


代码块

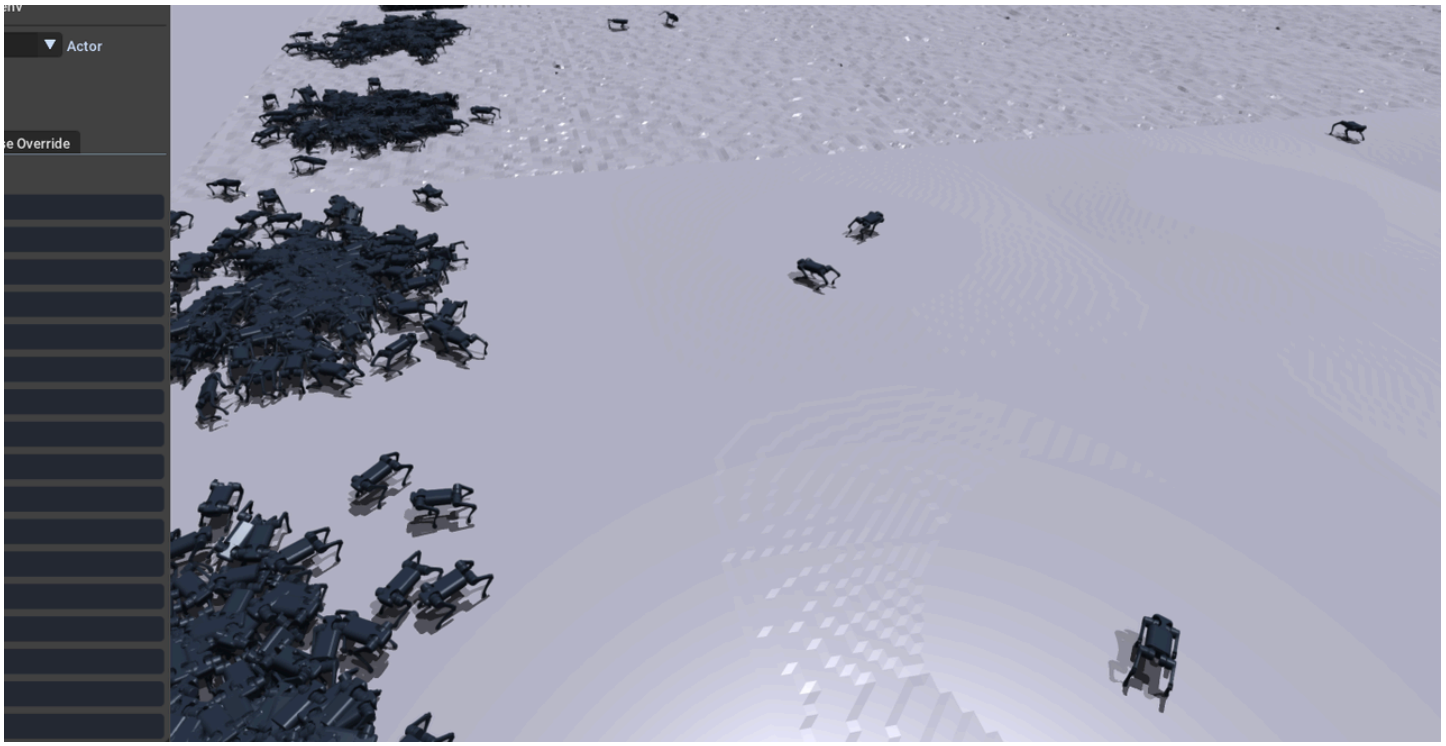
```
1   python play.py --task=a1
```



3.增加地形训练



将legged_robot_config.py文件中的terrain的mesh_type修改为*trimesh*，开始训练



3.查看训练结果

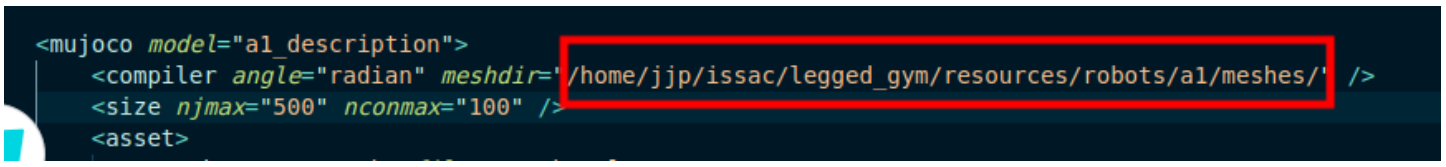
代码块

```
1    conda activate jjp
2    python play.py --task=a1
```

三.Mujoco仿真迁移

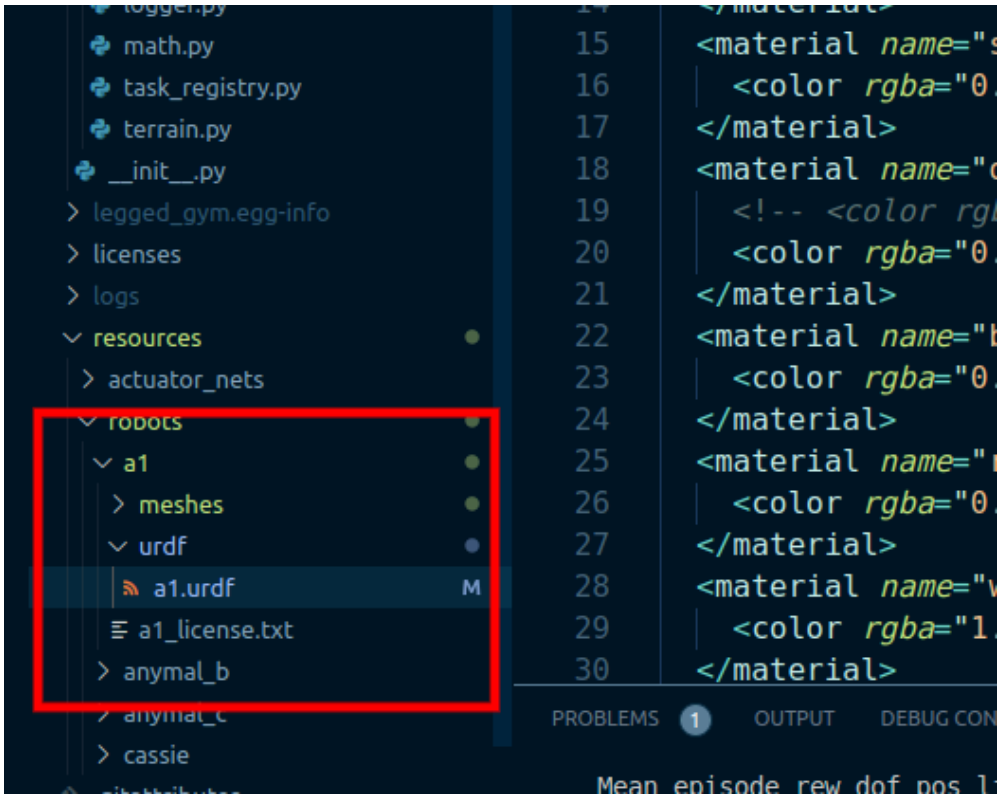
这里是我导好的文件，记得修改路径：

a1.zip
4.33MB



我们需要一个描述文件xml来在mujoco仿真器下进行仿真迁移：

1.urdf转mujoco模型



在文件robots下有a1.urdf模型，我们在a1文件下创建xml文件夹用来存放mujoco所需的描述文件

代码块

```
cd robots/a1
mkdir xml
```

在robot名称下面加入mujoco基座代码，注意这里路径改为自己的

代码块

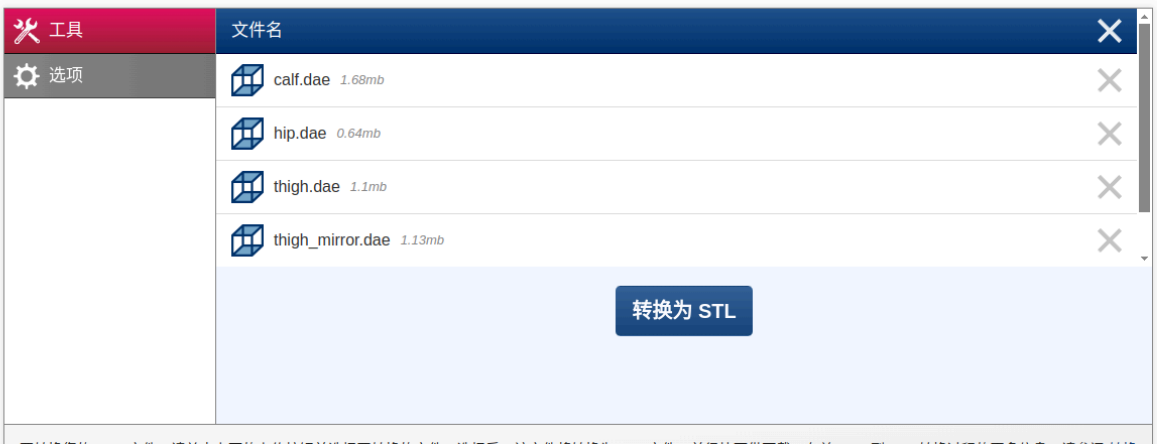
```
<mujoco>
  <compiler
    meshdir="/home/jjp/issac/legged_gym/resources/robots/a1/meshes"
    balanceinertia="true"
    discardvisual="false" />
</mujoco>
```

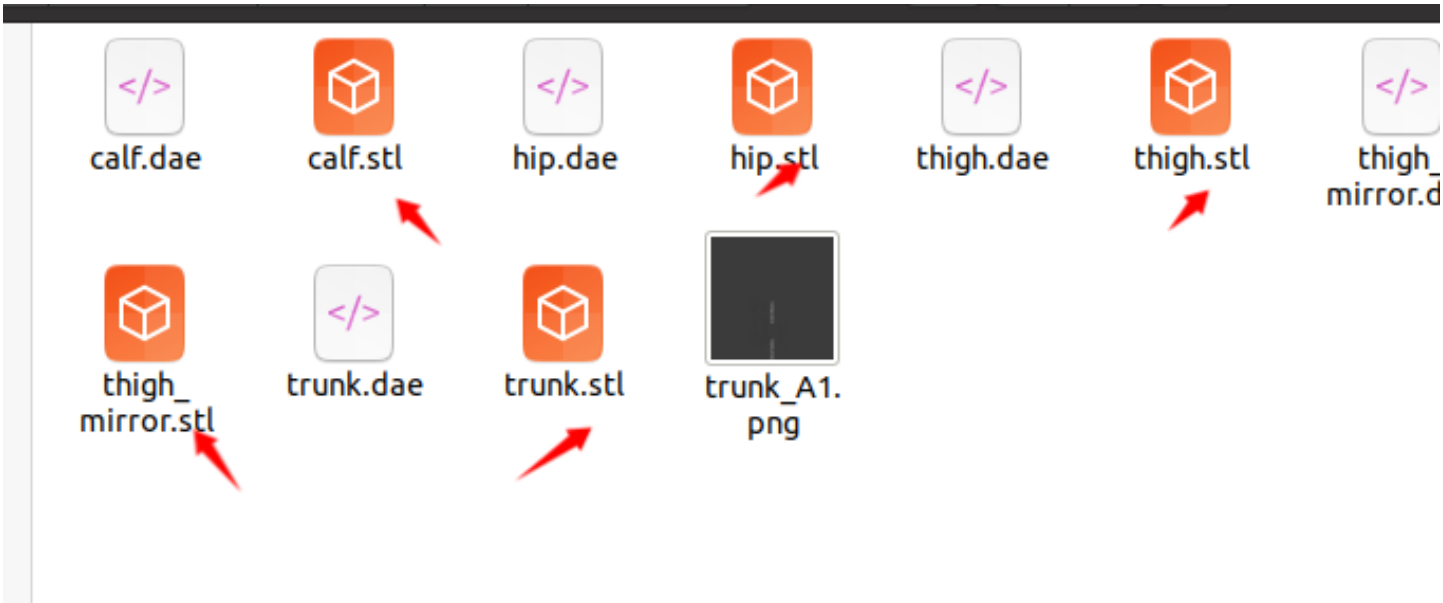
```xml
<?xml version="1.0" ?>
<robot name="a1_description" xmlns:xacro="http://www.ros.org/wiki/xacro">
  <mujoco>
  <compiler
    meshdir="/home/pi/Downloads/LocomotionWithNP3O-master/resources/tinymal/meshes"
    balanceinertia="true"
    discardvisual="false" />
</mujoco>
  <material name="black">
    <color rgba="0.0 0.0 0.0 1.0"/>
  </material>
  <material name="blue">
    <color rgba="0.0 0.0 0.8 1.0"/>
  </material>
  <material name="green">
    <color rgba="0.0 0.8 0.0 1.0"/>
  </material>
  <material name="grey">
    <color rgba="0.2 0.2 0.2 1.0"/>
```

将文件中的.dae文件变为.stl文件，并同时修改路径

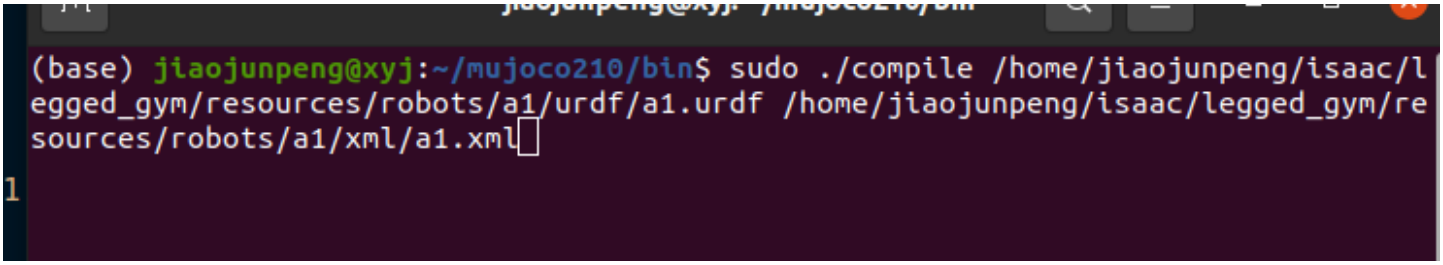
```xml
  <child link="trunk"/>
</joint>
<link name="trunk">
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="/home/jiaojunpeng/isaac/legged_gym/resources/robots/a1/meshes/trunk.stl" scale="1 1 1"/>
    </geometry>
    <material name="orange"/>
  </visual>
  <collision>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
```

将dae文件转化为stl文件stl转化，并下载压缩改为同样名称放入同一文件夹下,这里我放了链接





复制a1.urdf文件路径，打开mujoco文件夹



注意：这里的路径要自己的路径

代码块

```
sudo ./compile
/home/jiaojunpeng/isaac/legged_gym/resources/robots/a1/urdf/a1.urdf
/home/jiaojunpeng/isaac/legged_gym/resources/robots/a1/xml/a1.xml
```

相同目录下创建scene.xml文件如下

代码块

```
<mujoco model="a1 scene">
  <include file="a1.xml"/>

  <statistic center="0 0 0.1" extent="0.8"/>

  <visual>
    <headlight diffuse="0.6 0.6 0.6" ambient="0.3 0.3 0.3" specular="0 0 0"/>
    <rgba haze="0.15 0.25 0.35 1"/>
  </visual>

```

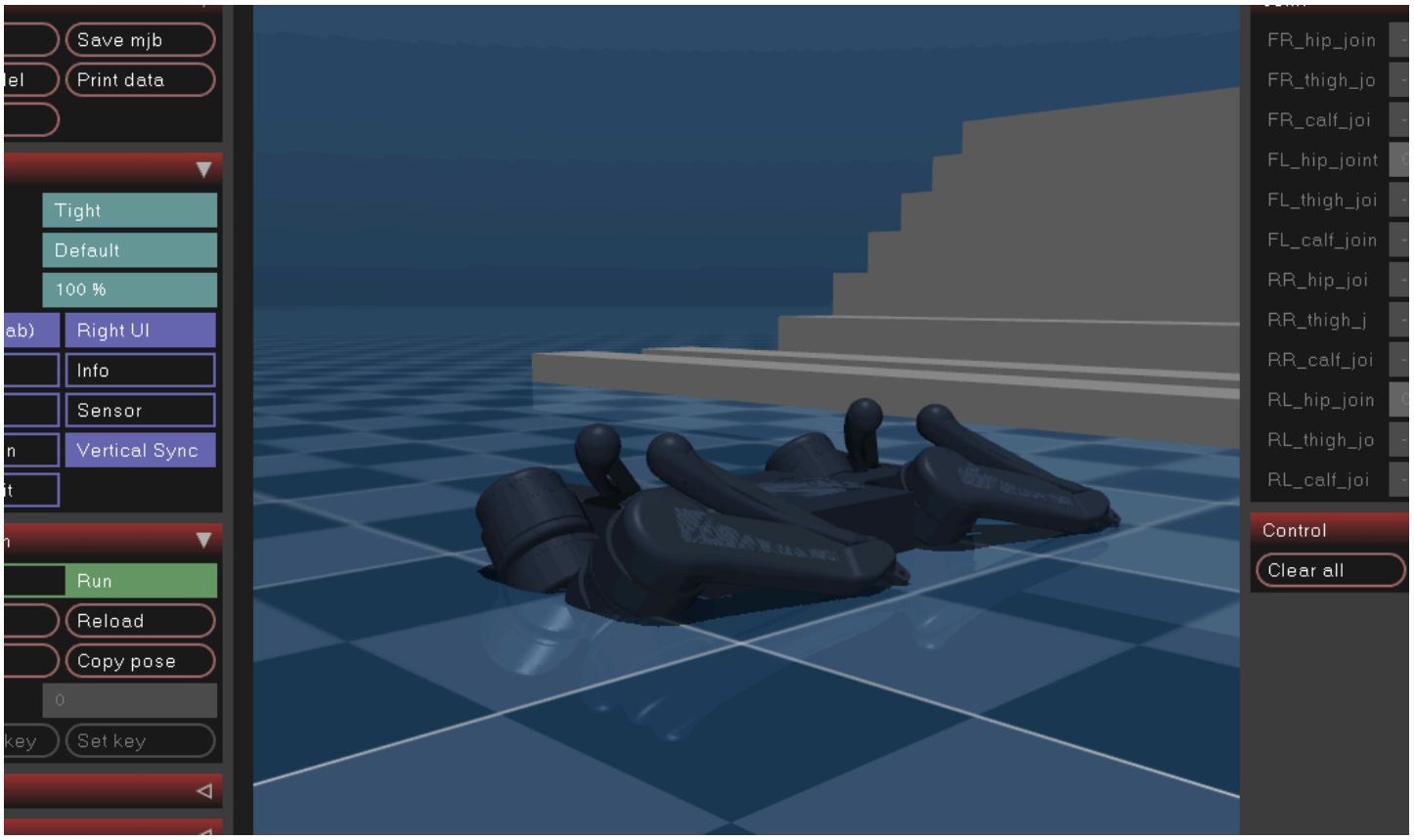
```
  <asset>
    <texture type="skybox" builtin="gradient" rgb1="0.3 0.5 0.7" rgb2="0 0 0" width="512" height="3072"/>
    <texture type="2d" name="groundplane" builtin="checker" mark="edge" rgb1="0.2 0.3 0.4" rgb2="0.1 0.2 0.3"
      markrgb="0.8 0.8 0.8" width="300" height="300"/>
    <material name="groundplane" texture="groundplane" texuniform="true" texrepeat="5 5" reflectance="0.2"/>
  </asset>

  <worldbody>
    <light pos="0 0 1.5" dir="0 0 -1" directional="true"/>
    <geom name="floor" size="0 0 0.05" type="plane" material="groundplane"/>

    <geom pos="1.2 0 0.04" type="box"  size="0.1 2 0.04" quat="1.0 0 0 0"/>
    <geom pos="1.6 0 0.04" type="box"  size="0.1 2 0.04" quat="1.0 0 0 0"/>

    <geom pos="2.3 0 0.02" type="box"  size="0.2 2 0.15" quat="1.0 0 0 0"/>
    <geom pos="2.6 0 0.02" type="box"  size="0.22 2 0.3" quat="1.0 0 0 0"/>
    <geom pos="2.8 0 0.02" type="box"  size="0.23 2 0.45" quat="1.0 0 0 0"/>
    <geom pos="3 0 0.02" type="box"  size="0.24 2 0.6" quat="1.0 0 0 0"/>
    <geom pos="3.2 0 0.02" type="box"  size="0.25 2 0.75" quat="1.0 0 0 0"/>
    <geom pos="3.4 0 0.02" type="box"  size="0.26 2 0.9" quat="1.0 0 0 0"/>
  </worldbody>
</mujoco>

```

mujoco命令行输入

代码块

```
./simulate
```

将scene.xml文件拖入



2.修改a1.xml文件

```
<body name="robot" pos="0 0 0.415" quat="1 0 0 0">
    <joint name="float_base" type="free"/>
  <inertial pos="0 0 0" quat="1 0 0 0" mass="4.713" diaginertia="0.0168 0.0566 0.0647"/>
   <site name="imu" pos="-0.02557 0 0.04232" />
```

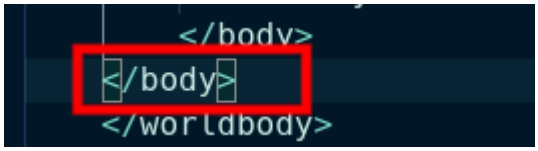
增加身体位置和惯性，以及imu

代码块

```
1    <body name="robot" pos="0 0 0.415" quat="1 0 0 0">
2        <joint name="float_base" type="free"/>
3        <inertial pos="0 0 0" quat="1 0 0 0" mass="4.713" diaginertia="0.0168
    0.0566 0.0647"/>
4        <site name="imu" pos="-0.02557 0 0.04232" />
5        .....
6    </body>
```

```
<worldbody>
    <body name="robot" pos="0 0 0.415" quat="1 0 0 0">
    <joint name="float_base" type="free"/>
    <inertial pos="0 0 0" quat="1 0 0 0" mass="4.713" diaginertia="0.0168 0.0566 0.0647"/>
    <site name="imu" pos="-0.02557 0 0.04232" />
    <geom size="0.0005 0.0005 0.0005" type="box" contype="0" conaffinity="0" group="1" />
```

在最后增加



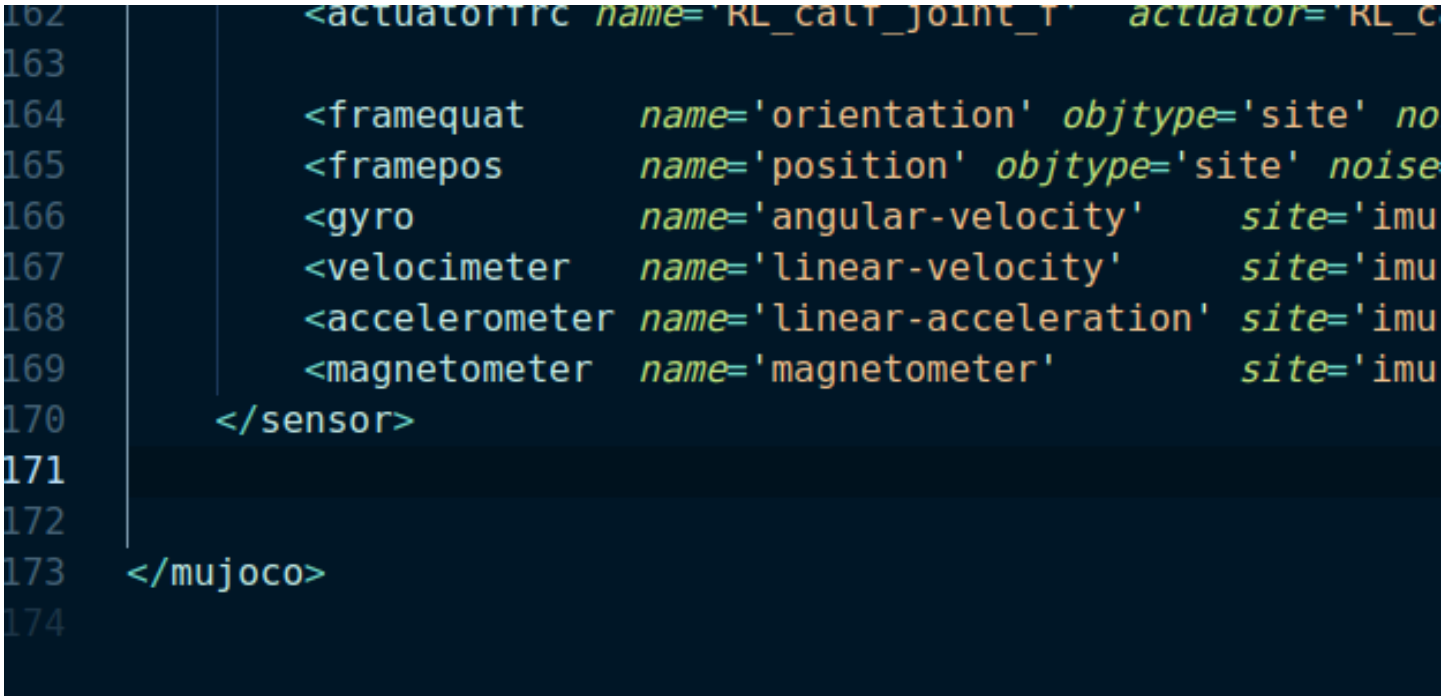
添加电机传感器等

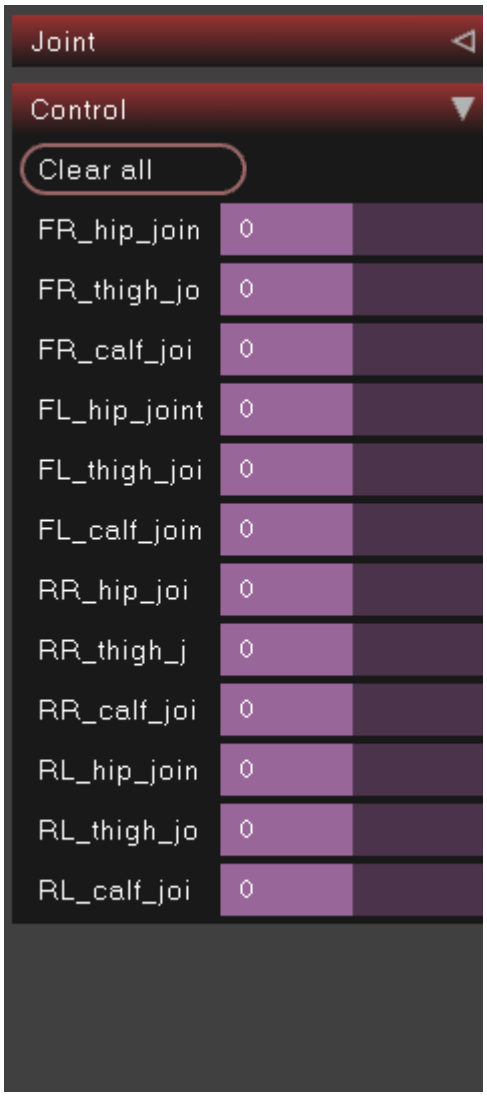
代码块

```
    <actuator>
        <motor name="FL_hip_joint" joint="FL_hip_joint" gear="1" ctrllimited="true" ctrlrange="-12 12"/>
        <motor name="FL_thigh_joint" joint="FL_thigh_joint" gear="1" ctrllimited="true" ctrlrange="-12 12"/>
        <motor name="FL_calf_joint" joint="FL_calf_joint" gear="1" ctrllimited="true" ctrlrange="-12 12"/>

        <motor name="FR_hip_joint" joint="FR_hip_joint" gear="1" ctrllimited="true" ctrlrange="-12 12"/>
        <motor name="FR_thigh_joint" joint="FR_thigh_joint" gear="1" ctrllimited="true" ctrlrange="-12 12"/>
        <motor name="FR_calf_joint" joint="FR_calf_joint" gear="1" ctrllimited="true" ctrlrange="-12 12"/>

        <motor name="RL_hip_joint" joint="RL_hip_joint" gear="1" ctrllimited="true" ctrlrange="-12 12"/>
        <motor name="RL_thigh_joint" joint="RL_thigh_joint" gear="1" ctrllimited="true" ctrlrange="-12 12"/>
        <motor name="RL_calf_joint" joint="RL_calf_joint" gear="1" ctrllimited="true" ctrlrange="-12 12"/>

        <motor name="RR_hip_joint" joint="RR_hip_joint" gear="1" ctrllimited="true" ctrlrange="-12 12"/>
        <motor name="RR_thigh_joint" joint="RR_thigh_joint" gear="1" ctrllimited="true" ctrlrange="-12 12"/>
        <motor name="RR_calf_joint" joint="RR_calf_joint" gear="1" ctrllimited="true" ctrlrange="-12 12"/>
    </actuator>
 <sensor>
        <actuatorpos name='FR_hip_joint_p'  actuator='FR_hip_joint' user='13'/>
        <actuatorpos name='FR_thigh_joint_p'  actuator='FR_thigh_joint' user='13'/>
        <actuatorpos name='FR_calf_joint_p'  actuator='FR_calf_joint' user='13'/>
        <actuatorpos name='FL_hip_joint_p'  actuator='FL_hip_joint' user='13'/>
        <actuatorpos name='FL_thigh_joint_p'  actuator='FL_thigh_joint' user='13'/>
```

```
24          <actuatorpos name='FL_calf_joint_p'  actuator='FL_calf_joint'
    user='13'/>
25          <actuatorpos name='RR_hip_joint_p'  actuator='RR_hip_joint' user='13'/>
26          <actuatorpos name='RR_thigh_joint_p'  actuator='RR_thigh_joint'
    user='13'/>
27          <actuatorpos name='RR_calf_joint_p'  actuator='RR_calf_joint'
    user='13'/>
28          <actuatorpos name='RL_hip_joint_p'  actuator='RL_hip_joint' user='13'/>
29          <actuatorpos name='RL_thigh_joint_p'  actuator='RL_thigh_joint'
    user='13'/>
30          <actuatorpos name='RL_calf_joint_p'  actuator='RL_calf_joint'
    user='13'/>
31
32          <actuatorvel name='FR_hip_joint_v'  actuator='FR_hip_joint' user='13'/>
33          <actuatorvel name='FR_thigh_joint_v'  actuator='FR_thigh_joint'
    user='13'/>
34          <actuatorvel name='FR_calf_joint_v'  actuator='FR_calf_joint'
    user='13'/>
35          <actuatorvel name='FL_hip_joint_v'  actuator='FL_hip_joint' user='13'/>
36          <actuatorvel name='FL_thigh_joint_v'  actuator='FL_thigh_joint'
    user='13'/>
37          <actuatorvel name='FL_calf_joint_v'  actuator='FL_calf_joint'
    user='13'/>
38          <actuatorvel name='RR_hip_joint_v'  actuator='RR_hip_joint' user='13'/>
39          <actuatorvel name='RR_thigh_joint_v'  actuator='RR_thigh_joint'
    user='13'/>
40          <actuatorvel name='RR_calf_joint_v'  actuator='RR_calf_joint'
    user='13'/>
41          <actuatorvel name='RL_hip_joint_v'  actuator='RL_hip_joint' user='13'/>
42          <actuatorvel name='RL_thigh_joint_v'  actuator='RL_thigh_joint'
    user='13'/>
43          <actuatorvel name='RL_calf_joint_v'  actuator='RL_calf_joint'
    user='13'/>
44
45          <actuatorfrc name='FR_hip_joint_f'  actuator='FR_hip_joint'  user='13'
    noise='1e-3'/>
46          <actuatorfrc name='FR_thigh_joint_f'  actuator='FR_thigh_joint'
    user='13' noise='1e-3'/>
47          <actuatorfrc name='FR_calf_joint_f'  actuator='FR_calf_joint'
    user='13' noise='1e-3'/>
48          <actuatorfrc name='FL_hip_joint_f'  actuator='FL_hip_joint'  user='13'
    noise='1e-3'/>
49          <actuatorfrc name='FL_thigh_joint_f'  actuator='FL_thigh_joint'
    user='13' noise='1e-3'/>
50          <actuatorfrc name='FL_calf_joint_f'  actuator='FL_calf_joint'
    user='13' noise='1e-3'/>
```

```
51          <actuatorfrc name='RR_hip_joint_f'  actuator='RR_hip_joint'  user='13'
    noise='1e-3'/>
52          <actuatorfrc name='RR_thigh_joint_f'  actuator='RR_thigh_joint'
    user='13' noise='1e-3'/>
53          <actuatorfrc name='RR_calf_joint_f'  actuator='RR_calf_joint'
    user='13' noise='1e-3'/>
54          <actuatorfrc name='RL_hip_joint_f'  actuator='RL_hip_joint'  user='13'
    noise='1e-3'/>
55          <actuatorfrc name='RL_thigh_joint_f'  actuator='RL_thigh_joint'
    user='13' noise='1e-3'/>
56          <actuatorfrc name='RL_calf_joint_f'  actuator='RL_calf_joint'
    user='13' noise='1e-3'/>
57
58          <framequat     name='orientation' objtype='site' noise='0.001'
    objname='imu'/>
59          <framepos      name='position' objtype='site' noise='0.001'
    objname='imu'/>
60          <gyro          name='angular-velocity'    site='imu' noise='0.005'
    cutoff='34.9'/>
61          <velocimeter   name='linear-velocity'     site='imu' noise='0.001'
    cutoff='30'/>
62          <accelerometer name='linear-acceleration' site='imu' noise='0.005'
    cutoff='157'/>
63          <magnetometer  name='magnetometer'        site='imu'/>
64      </sensor>
```



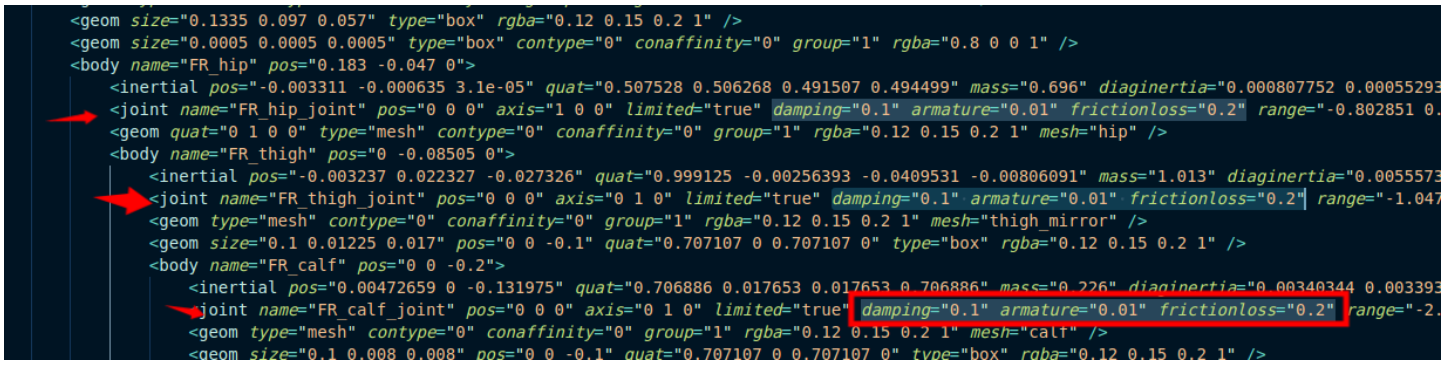


重新载入Control为这样则为成功添加

代码块

```
1   damping="0.1" armature="0.01" frictionloss="0.2"
```

然后我们加入关节摩擦：在每个joint后加入



然后，更改关节顺序，同训练时一样：

```
    <body name="FL_hip" pos="0.183 0.047 0">
        <inertial pos="-0.003311 0.000635 3.1e-05" quat="0.494499 0.491507 0.506268 0.507528" mass="0.696" diaginertia="0.00080775
        <joint name="FL_hip_joint" pos="0 0 0" axis="1 0 0" limited="true" damping="0.1" armature="0.01" frictionloss="0.2" range=
        <geom type="mesh" contype="0" conaffinity="0" group="1" rgba="0.12 0.15 0.2 1" mesh="hip" />
        <body name="FL_thigh" pos="0 0.08505 0">
            <inertial pos="-0.003237 -0.022327 -0.027326" quat="0.999125 0.00256393 -0.0409531 0.00806091" mass="1.013" diaginerti
            <joint name="FL_thigh_joint" pos="0 0 0" axis="0 1 0" limited="true" damping="0.1" armature="0.01" frictionloss="0.2"
            <geom type="mesh" contype="0" conaffinity="0" group="1" rgba="0.12 0.15 0.2 1" mesh="thigh" />
            <geom size="0.1 0.01225 0.017" pos="0 0 -0.1" quat="0.707107 0 0.707107 0" type="box" rgba="0.12 0.15 0.2 1" />
            <body name="FL_calf" pos="0 0 -0.2">
                <inertial pos="0.00472659 0 -0.131975" quat="0.706886 0.017653 0.017653 0.706886" mass="0.226" diaginertia="0.0034
                <joint name="FL_calf_joint" pos="0 0 0" axis="0 1 0" limited="true" damping="0.1" armature="0.01" frictionloss="0.
                <geom type="mesh" contype="0" conaffinity="0" group="1" rgba="0.12 0.15 0.2 1" mesh="calf" />
                <geom size="0.1 0.008 0.008" pos="0 0 -0.1" quat="0.707107 0 0.707107 0" type="box" rgba="0.12 0.15 0.2 1" />
                <geom size="0.01" pos="0 0 -0.2" contype="0" conaffinity="0" group="1" rgba="0.12 0.15 0.2 1" />
                <geom size="0.02" pos="0 0 -0.2" rgba="0.12 0.15 0.2 1" friction="1.4 0.02 0.01"/>
            </body>
        </body>
    </body>
    <body name="FR_hip" pos="0.183 -0.047 0">
        <inertial pos="-0.003311 -0.000635 3.1e-05" quat="0.507528 0.506268 0.491507 0.494499" mass="0.696" diaginertia="0.0008077
        <joint name="FR_hip_joint" pos="0 0 0" axis="1 0 0" limited="true" damping="0.1" armature="0.01" frictionloss="0.2" range=
        <geom quat="0 1 0 0" type="mesh" contype="0" conaffinity="0" group="1" rgba="0.12 0.15 0.2 1" mesh="hip" />
        <body name="FR_thigh" pos="0 -0.08505 0">
            <inertial pos="-0.003237 0.022327 -0.027326" quat="0.999125 -0.00256393 -0.0409531 -0.00806091" mass="1.013" diaginert
            <joint name="FR_thigh_joint" pos="0 0 0" axis="0 1 0" limited="true" damping="0.1" armature="0.01" frictionloss="0.2"
            <geom type="mesh" contype="0" conaffinity="0" group="1" rgba="0.12 0.15 0.2 1" mesh="thigh_mirror" />
            <geom size="0.1 0.01225 0.017" pos="0 0 -0.1" quat="0.707107 0 0.707107 0" type="box" rgba="0.12 0.15 0.2 1" />
            <body name="FR_calf" pos="0 0 -0.2">
                <inertial pos="0.00472659 0 -0.131975" quat="0.706886 0.017653 0.017653 0.706886" mass="0.226" diaginertia="0.0034
                <joint name="FR_calf_joint" pos="0 0 0" axis="0 1 0" limited="true" damping="0.1" armature="0.01" frictionloss="0.
                <geom type="mesh" contype="0" conaffinity="0" group="1" rgba="0.12 0.15 0.2 1" mesh="calf" />
                <geom size="0.1 0.008 0.008" pos="0 0 -0.1" quat="0.707107 0 0.707107 0" type="box" rgba="0.12 0.15 0.2 1" />
                <geom size="0.01" pos="0 0 -0.2" contype="0" conaffinity="0" group="1" rgba="0.12 0.15 0.2 1" />
                <geom size="0.02" pos="0 0 -0.2" rgba="0.12 0.15 0.2 1" friction="1.4 0.02 0.01"/>
            </body>
        </body>
    </body>
```

3.导出模型onnx

在log同级下创建onxx文件夹

在play.py中找到jit模型函数：

```
# export policy as a jit module (used to run it from C++)
if EXPORT_POLICY:
    path = os.path.join(LEGGED_GYM_ROOT_DIR, 'logs', train_cfg.runner.experiment_name, 'exported', 'policies')
    export_policy_as_jit(ppo_runner.alg.actor_critic, path)
    print('Exported policy as jit script to: ', path)
```

进入这个函数的定义文件创建下面函数：helpers.py

下面的路径修改为自己的

代码块

```
def export_policy_as_onnx(actor_critic):
    model = copy.deepcopy(actor_critic.actor).to('cpu')
    actor_input = torch.randn(1, 45)  # 根据实际情况调整形状

    body_onnx_path = '/home/jjp/issac/legged_gym/onnx/' + 'legged.onnx'
    paths = [body_onnx_path]
    for path in paths:
        if os.path.exists(path):
            os.remove(path)
            print(f"已删除: {path}")
```

```
        else:
            print(f"文件不存在: {path}")
    print("path:",body_onnx_path)
    torch.onnx.export(model, actor_input, body_onnx_path, opset_version=11)
```

然后，play导入函数,__init__.py中同理

```
from legged_gym.utils import  get_args, export_policy_as_jit, task_registry, Logger,export_policy_as_onnx
```

```
from .helpers import class_to_dict, get_load_path, get_args, export_policy_as_jit, export_policy_as_onnx, set_seed, update_class_
from .task_registry import task_registry
```

修改导出函数：

```
if EXPORT_POLICY:
    # path = os.path.join(LEGGED_GYM_ROOT_DIR, 'logs', train_cfg.runner.experiment_name, 'exported', 'policies')
    # export_policy_as_jit(ppo_runner.alg.actor_critic, path)
    export_policy_as_onnx(ppo_runner.alg.actor_critic)
    # print('Exported policy as jit script to: ', path)
```

导出onnx模型

4.创建mujoco验证python文件

在legged_gym文件夹下新建sim2sim.py文件

代码块

```
import math
import numpy as np
import mujoco, mujoco_viewer
from tqdm import tqdm
from collections import deque
import onnxruntime as ort
from legged_gym.envs import *
import torch

import rospy
from sensor_msgs.msg import Joy
import os

import time

default_dof_pos=[0.1,0.8,-1.5 ,-0.1,0.8,-1.5, 0.1,1,-1.5, -0.1,1,-1.5]#默认角度需要与isacc一致
```

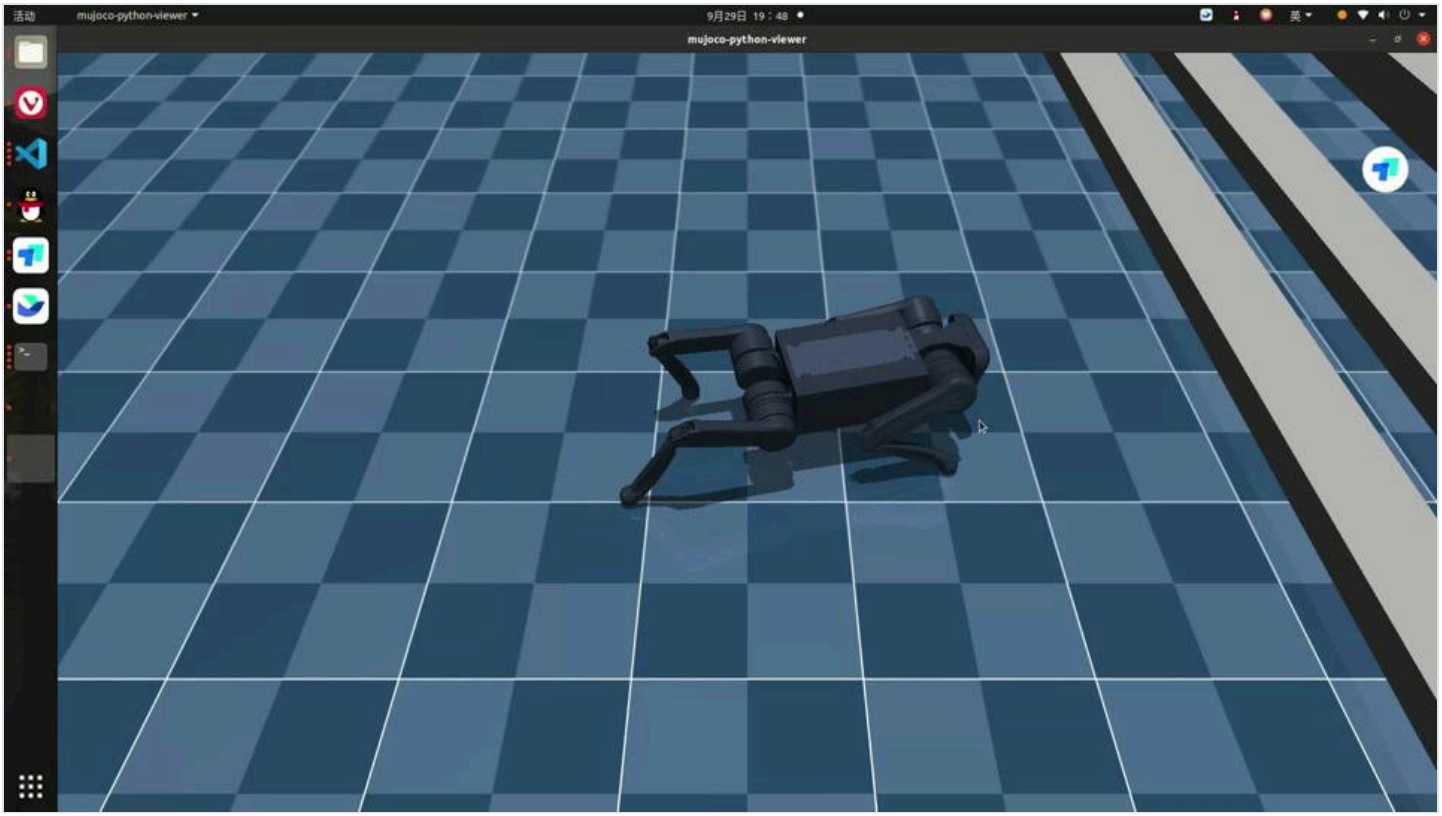
```
17
18  joy_cmd = [0.0, 0.0, 0.0]
19
20  def joy_callback(joy_msg):
21      global joy_cmd
22      joy_cmd[0] =  joy_msg.axes[1]
23      joy_cmd[1] =  joy_msg.axes[0]
24      joy_cmd[2] =  joy_msg.axes[3]  # 横向操作
25
26  def quat_rotate_inverse(q, v):
27      # 确保输入为numpy数组
28      q_w = q[-1]
29      q_vec = q[:3]
30      a = v * (2.0 * q_w**2 - 1.0)
31      b = np.cross(q_vec, v) * q_w * 2.0
32      c = q_vec * np.dot(q_vec, v) * 2.0
33      return a - b + c
34
35  def get_obs(data):
36      '''Extracts an observation from the mujoco data structure
37      '''
38      q = data.qpos.astype(np.double)
39      dq = data.qvel.astype(np.double)
40      quat = data.sensor('orientation').data[[1, 2, 3, 0]].astype(np.double)
41      omega = data.sensor('angular-velocity').data.astype(np.double)
42      vel = data.sensor('linear-velocity').data.astype(np.double)
43      return (q, dq, quat,omega,vel)
44
45  def pd_control(target_q, q, kp, target_dq, dq, kd):
46      '''Calculates torques from position commands
47      '''
48      return (target_q - q) * kp + (target_dq - dq) * kd
49
50  class Sim2simCfg(A1RoughCfg):
51
52      class sim_config:
53          # print("{LEGGED_GYM_ROOT_DIR}",{LEGGED_GYM_ROOT_DIR})
54
55          mujoco_model_path =
    '/home/arx/isaac/legged_gym/resources/robots/a1/xml/scene.xml'
56
57          sim_duration = 60.0
58          dt = 0.005 #1Khz底层
59          decimation = 4 # 50Hz
60
61      class robot_config:
62
```

```
63          kps = np.array(20, dtype=np.double)#PD和isacc内部一致
64          kds = np.array(0.5, dtype=np.double)
65          tau_limit = 20. * np.ones(12, dtype=np.double)#nm
66
67  if __name__ == '__main__':
68      rospy.init_node('play')
69      rospy.Subscriber('/joy', Joy, joy_callback, queue_size=10)
70
71      policy_model_path = "/home/arx/isaac/legged_gym/onnx/legged1.onnx"
72
73
74      policy = ort.InferenceSession(policy_model_path,
75                                    providers=['CPUExecutionProvider'])
76      model =
    mujoco.MjModel.from_xml_path(Sim2simCfg.sim_config.mujoco_model_path)#载入初始化
    位置由XML决定
77      model.opt.timestep = Sim2simCfg.sim_config.dt
78      data = mujoco.MjData(model)
79      mujoco.mj_step(model, data)
80      model.opt.gravity = (0, 0, -9.81)
81      viewer = mujoco_viewer.MujocoViewer(model, data)
82
83      target_q = np.zeros((12), dtype=np.double)
84      action = np.zeros((12), dtype=np.double)
85      action_flt = np.zeros((12), dtype=np.double)
86      last_actions = np.zeros((12), dtype=np.double)
87      lag_buffer = [np.zeros_like(action) for i in range(2+1)]
88
89      hist_obs = deque()
90      for _ in range(15):
91          hist_obs.append(np.zeros([1,45], dtype=np.double))
92      count_lowlevel = 0
93
94      for _ in tqdm(range(int(Sim2simCfg.sim_config.sim_duration*10/
    Sim2simCfg.sim_config.dt)), desc="Simulating..."):
95
96          # Obtain an observation
97          q, dq, quat,omega,vel= get_obs(data)#从mujoco获取仿真数据
98          q = q[-12:]
99          dq = dq[-12:]
100         if 1:
101             # 1000hz ->50hz
102             if count_lowlevel % Sim2simCfg.sim_config.decimation == 0:
103
104                 obs = np.zeros([1, 45], dtype=np.float32) #1,45
105                 gravity_vec =  np.array([0., 0., -1.], dtype=np.float32)
106                 proj_gravity = quat_rotate_inverse(quat,gravity_vec)
```

```
                obs[0, 0] = omega[0]
*Sim2simCfg.normalization.obs_scales.ang_vel
                obs[0, 1] = omega[1]
*Sim2simCfg.normalization.obs_scales.ang_vel
                obs[0, 2] = omega[2]
*Sim2simCfg.normalization.obs_scales.ang_vel
                obs[0, 3] = proj_gravity[0]
                obs[0, 4] = proj_gravity[1]
                obs[0, 5] = proj_gravity[2]
                obs[0, 6] = (joy_cmd[0])*
Sim2simCfg.normalization.obs_scales.lin_vel
                obs[0, 7] = (joy_cmd[1])*
Sim2simCfg.normalization.obs_scales.lin_vel
                obs[0, 8] = (joy_cmd[2])*
Sim2simCfg.normalization.obs_scales.ang_vel
                obs[0, 9:21] = (q-default_dof_pos) *
Sim2simCfg.normalization.obs_scales.dof_pos #g关节角度顺序依据修改为样机
                obs[0, 21:33] = dq *
Sim2simCfg.normalization.obs_scales.dof_vel
                obs[0, 33:45] = last_actions#上次控制指令
                obs = np.clip(obs, -
Sim2simCfg.normalization.clip_observations,
Sim2simCfg.normalization.clip_observations)

                n_proprio=45
                history_len=15
                num_z_encoder = 32

                policy_input = np.zeros([1, n_proprio], dtype=np.float32)

                for i in range(n_proprio):
                    policy_input[0, i] = obs[0][i]

                policy_output_name = policy.get_outputs()[0].name
                policy_input_name = policy.get_inputs()[0].name

                action[:] = policy.run([policy_output_name],
{policy_input_name: policy_input})[0]

                action = np.clip(action, -10,10)

                last_actions=action

                action_flt = action *0.25

                joint_pos_target = action_flt + default_dof_pos
                target_q=joint_pos_target
```

```
143
144                 target_dq = np.zeros((12), dtype=np.double)
145                 # Generate PD control
146                 tau = pd_control(target_q, q, Sim2simCfg.robot_config.kps,
147                                  target_dq, dq, Sim2simCfg.robot_config.kds)  # Calc torques
148                 tau = np.clip(tau, -Sim2simCfg.robot_config.tau_limit, Sim2simCfg.robot_config.tau_limit)  # Clamp torques
149                 # torques = np.clip(tau, -torque_limits, torque_limits)
150                 data.ctrl = tau
151             time.sleep(0.003)
152             mujoco.mj_step(model, data)
153
154         else:
155             target_q = default_dof_pos
156             target_dq = np.zeros((12), dtype=np.double)
157             # Generate PD control
158             tau = pd_control(target_q, q, Sim2simCfg.robot_config.kps,
159                              target_dq, dq, Sim2simCfg.robot_config.kds)  # Calc torques
160             tau = np.clip(tau, -Sim2simCfg.robot_config.tau_limit, Sim2simCfg.robot_config.tau_limit)  # Clamp torques
161             data.ctrl = tau
162
163             mujoco.mj_step(model, data)
164
165         viewer.render()
166         count_lowlevel += 1
167
168     viewer.close()
169
170
```

迁移成功：



四.增加随机化训练

注：由于仿真环境过于理想，我们需要增加一定的随机化

1.增加电机偏执、kpkd 以及力矩的随机化。

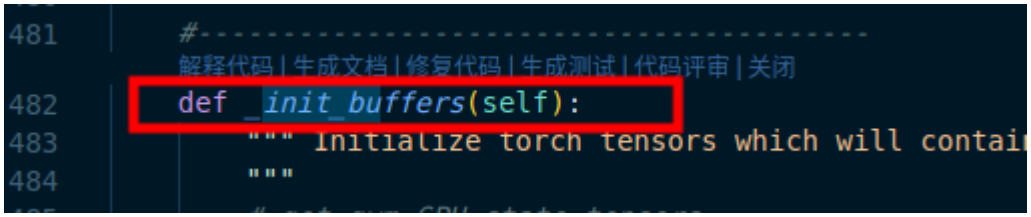
首先，我们在domain_rand中增加随机化变量：

```
class domain_rand:
    randomize_friction = True
    friction_range = [0.5, 1.25]
    randomize_base_mass = False
    added_mass_range = [-1., 1.]
    push_robots = True
    push_interval_s = 15
    max_push_vel_xy = 1.

    randomize_Kp_factor = True
    Kp_factor_range = [0.8, 1.2]

    randomize_Kd_factor = True
    Kd_factor_range = [0.8, 1.2]

    randomize_motor_offset = True
    motor_offset_range = [-0.05, 0.05]

    randomize_motor_strength = True
    motor_strength_range = [0.8, 1.2]
```

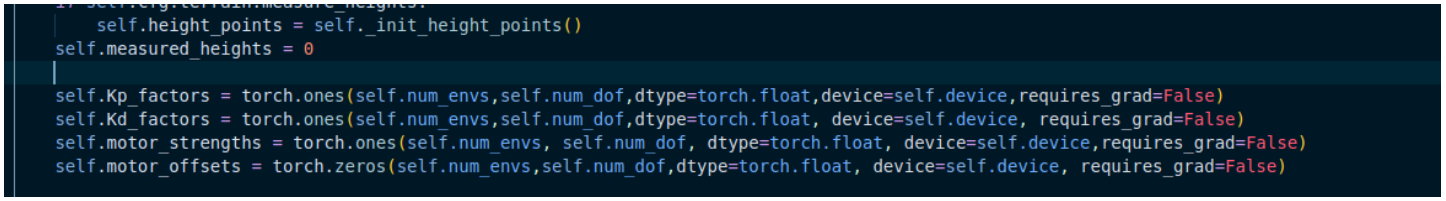
代码块

```
randomize_Kp_factor = True
Kp_factor_range = [0.8, 1.2]

randomize_Kd_factor = True
Kd_factor_range = [0.8, 1.2]

randomize_motor_offset = True
motor_offset_range = [-0.05, 0.05]

randomize_motor_strength = True
motor_strength_range = [0.8, 1.2]
```

其次在legged_robot.py中添加相应配置，首先添加对应名字张量：



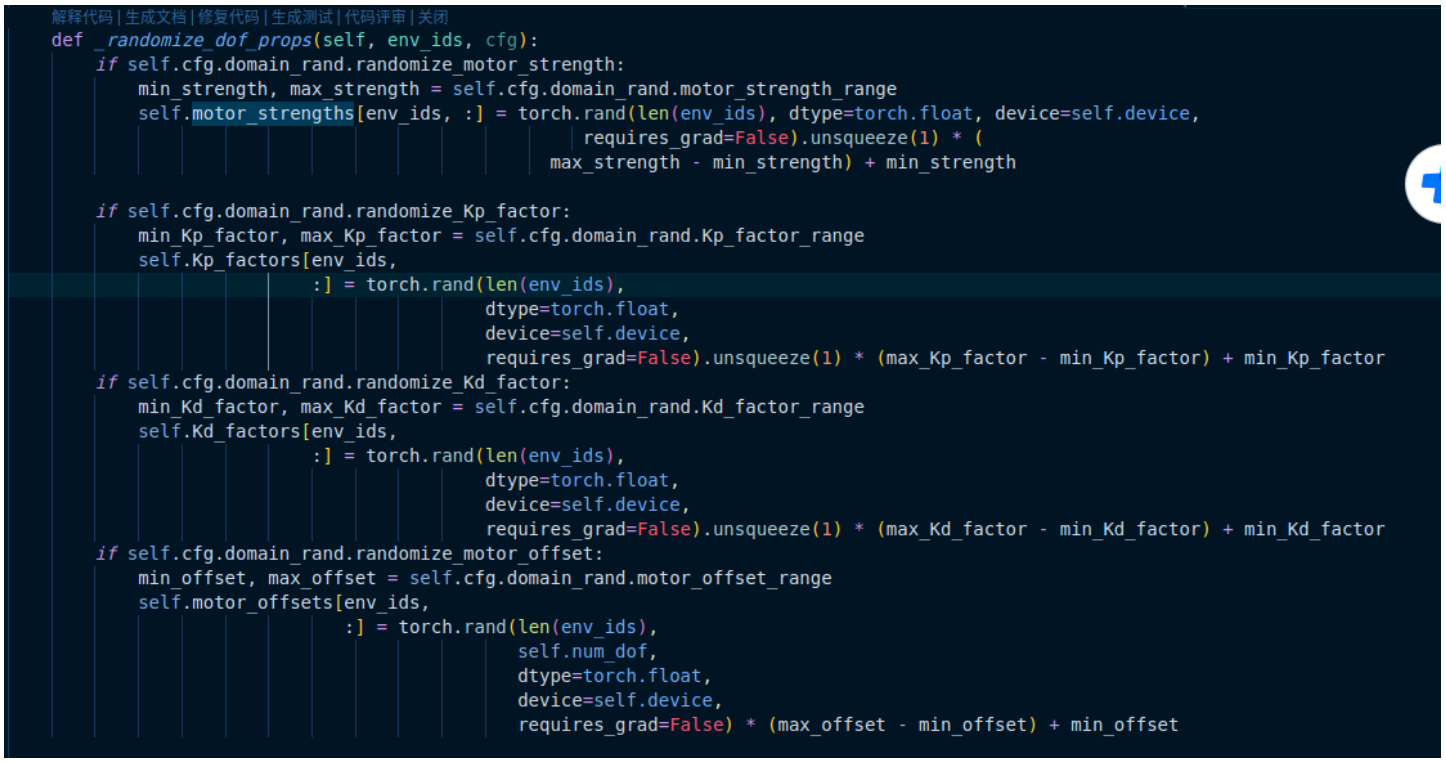
在这个函数中添加



代码块

```
self.Kp_factors = torch.ones(self.num_envs,self.num_dof,dtype=torch.float,device=self.device,requires_grad=False)
self.Kd_factors = torch.ones(self.num_envs,self.num_dof,dtype=torch.float, device=self.device, requires_grad=False)
self.motor_strengths = torch.ones(self.num_envs, self.num_dof, dtype=torch.float, device=self.device,requires_grad=False)
self.motor_offsets = torch.zeros(self.num_envs,self.num_dof,dtype=torch.float, device=self.device, requires_grad=False)
```

然后加入随机化定义函数**_randomize_dof_props**



代码块

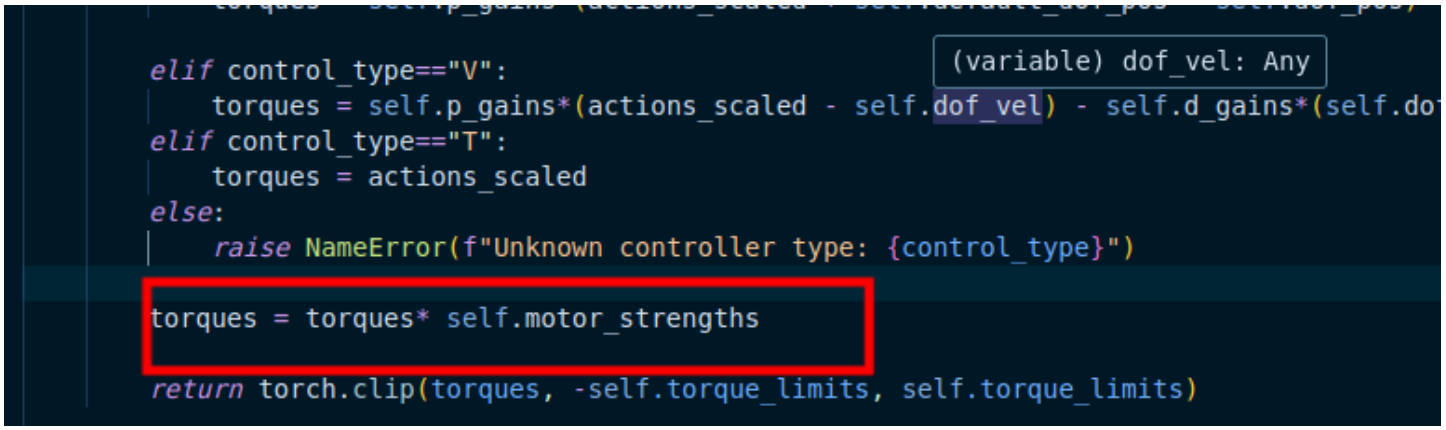
```
    def _randomize_dof_props(self, env_ids, cfg):
        if self.cfg.domain_rand.randomize_motor_strength:
            min_strength, max_strength =
self.cfg.domain_rand.motor_strength_range
            self.motor_strengths[env_ids, :] = torch.rand(len(env_ids),
dtype=torch.float, device=self.device,

requires_grad=False).unsqueeze(1) * (
                                                max_strength - min_strength)
+ min_strength

        if self.cfg.domain_rand.randomize_Kp_factor:
            min_Kp_factor, max_Kp_factor = self.cfg.domain_rand.Kp_factor_range
            self.Kp_factors[env_ids,
                            :] = torch.rand(len(env_ids),
                                            dtype=torch.float,
                                            device=self.device,
                                            requires_grad=False).unsqueeze(1)
* (max_Kp_factor - min_Kp_factor) + min_Kp_factor
        if self.cfg.domain_rand.randomize_Kd_factor:
            min_Kd_factor, max_Kd_factor = self.cfg.domain_rand.Kd_factor_range
            self.Kd_factors[env_ids,
                            :] = torch.rand(len(env_ids),
                                            dtype=torch.float,
                                            device=self.device,
```

```
                                            requires_grad=False).unsqueeze(1)
    * (max_Kd_factor - min_Kd_factor) + min_Kd_factor
        if self.cfg.domain_rand.randomize_motor_offset:
            min_offset, max_offset = self.cfg.domain_rand.motor_offset_range
            self.motor_offsets[env_ids,
                               :] = torch.rand(len(env_ids),
                                               self.num_dof,
                                               dtype=torch.float,
                                               device=self.device,
                                               requires_grad=False) *
    (max_offset - min_offset) + min_offset
```

然后修改*_compute_torques函数改变计算方式：*

加入力矩随机化：

```
代码块
1   torques = torques* self.motor_strengths
```



然后改变P控制表达式：


```
代码块
1   torques = self.p_gains*self.Kp_factors*(actions_scaled + self.default_dof_pos
    - self.dof_pos+ self.motor_offsets) - self.d_gains*self.dof_vel*self.Kd_factors
```

然后将**_randomize_dof_props**函数加入训练过程来不断更新：

reset_idx函数中：

```python
def reset_idx(self, env_ids):
    """ Reset some environments.
        Calls self._reset_dofs(env_ids), self._reset_root_states(env_ids),
        [Optional] calls self._update_terrain_curriculum(env_ids), self.up
        Logs episode info
        Resets some buffers

    Args:
        env_ids (list[int]): List of environment ids which must be reset
    """
    if len(env_ids) == 0:
        return
    # update curriculum
    if self.cfg.terrain.curriculum:
        self._update_terrain_curriculum(env_ids)
    # avoid updating command curriculum at each step since the maximum com
    if self.cfg.commands.curriculum and (self.common_step_counter % self.m
        self.update_command_curriculum(env_ids)

    # reset robot states
    self._reset_dofs(env_ids)
    self._reset_root_states(env_ids)

    self._resample_commands(env_ids)
    self._randomize_dof_props(env_ids)
    # reset buffers
```

_post_physics_step_callback函数中

```python
def _post_physics_step_callback(self):
    """ Callback called before computing terminations, rewards, and observations
        Default behaviour: Compute ang vel command based on target and heading, compute measured terrain heights and randomly
    """
    #
    env_ids = (self.episode_length_buf % int(self.cfg.commands.resampling_time / self.dt)==0).nonzero(as_tuple=False).flatten()
    self._randomize_dof_props(env_ids)
```

开始训练：

```
1   python train.py --task=a1
```

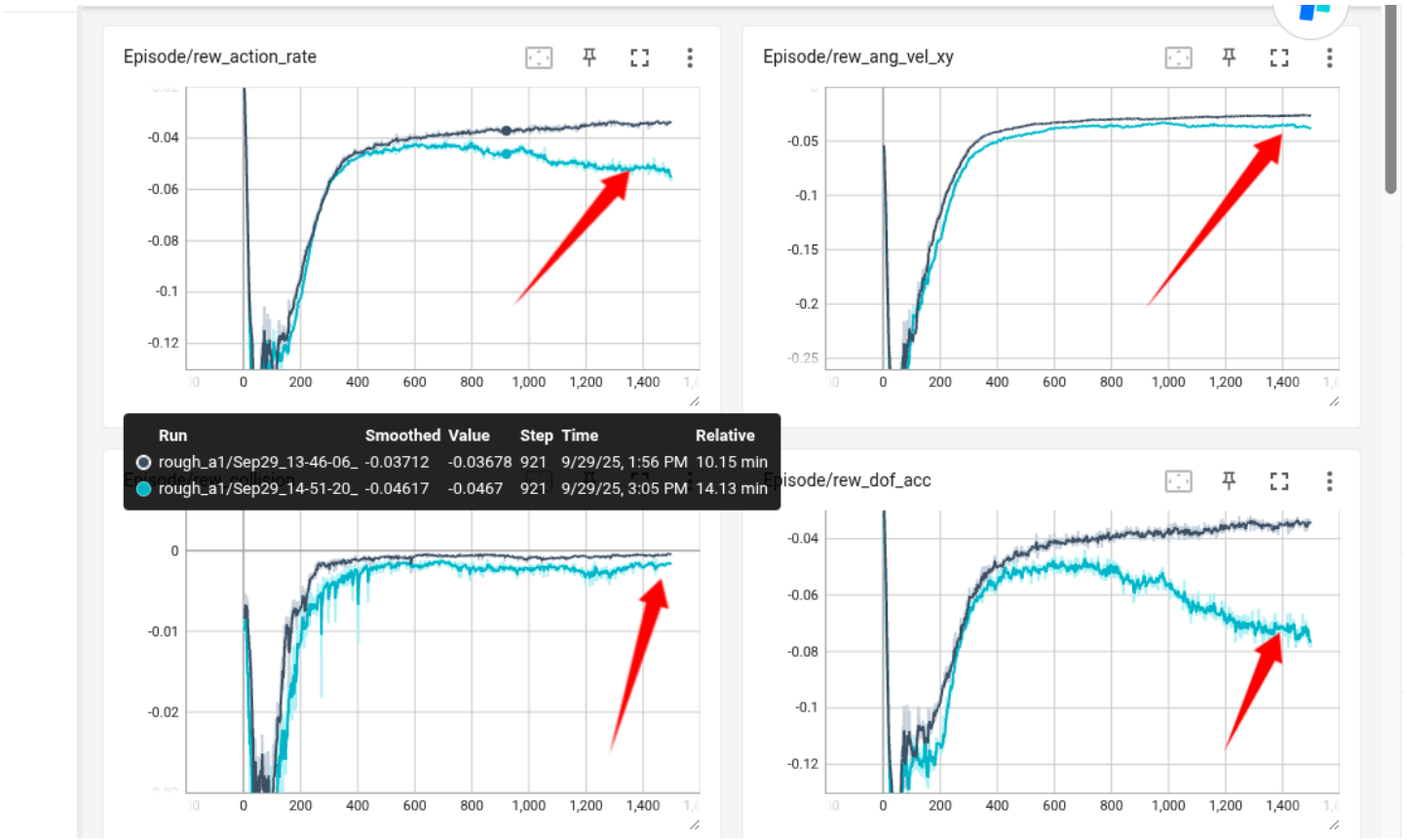
使用tensorboard查看训练曲线：

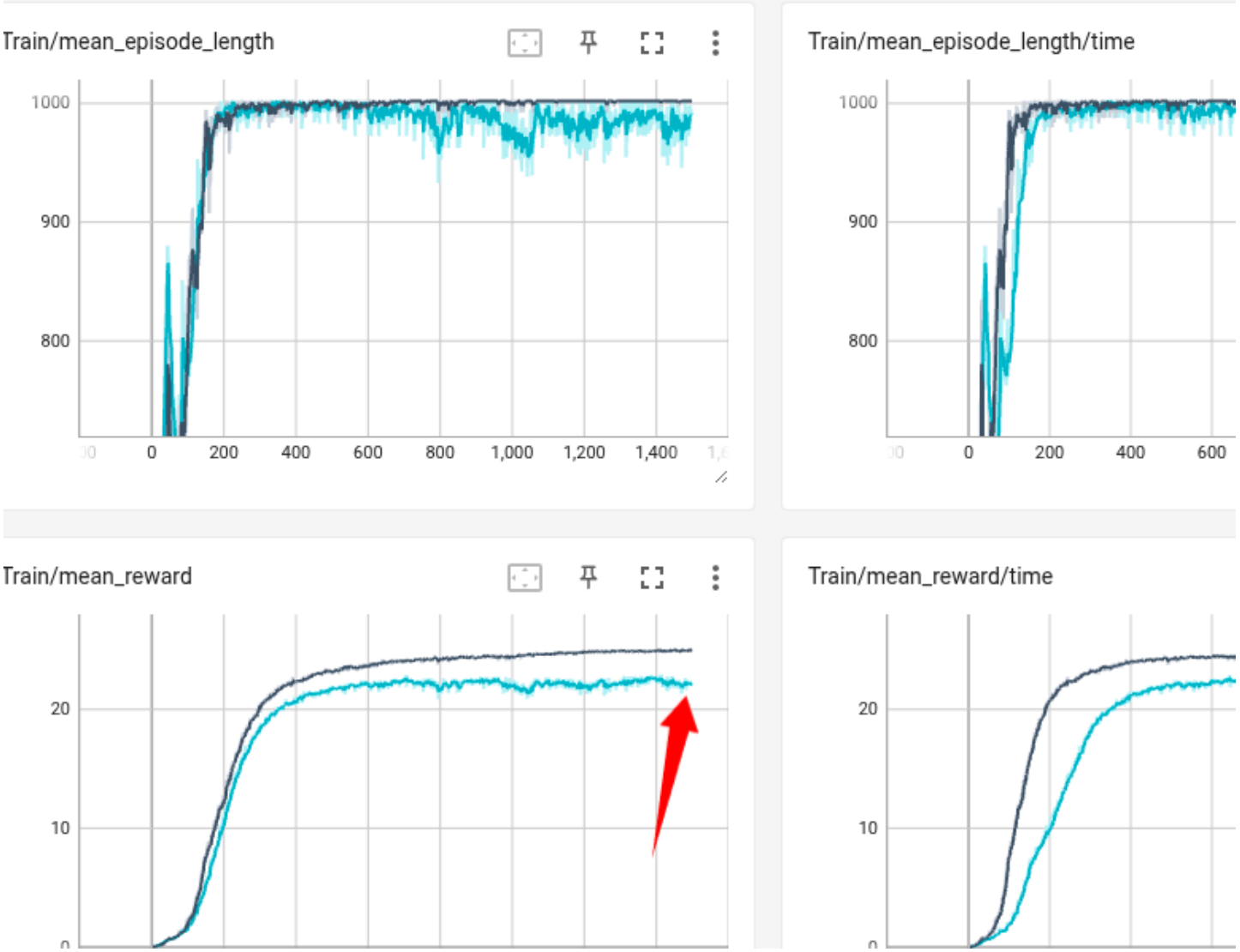
```
1   tensorboard --logdir=/home/arx/isaac/legged_gym/logs/
```

```
^C(jjp) arx@arx-4090-ii:~/isaac/legged_gym$ tensorboard --logdir=/home/arx/isaac/legged_gym/logs/
TensorFlow installation not found - running with reduced feature set.
I0929 15:12:26.415194 140125478405888 plugin.py:429] Monitor runs begin
Serving TensorBoard on localhost; to expose to the network, use a proxy or pass --bind_all
TensorBoard 2.14.0 at http://localhost:6006/ (Press CTRL+C to quit)
```

进入此网站：

惩罚变高，总奖励变差。





对比我们发现增加随机化以后，同样奖励函数比例下，训练效果变差。

2.增加质心摩擦恢复力随机

```
class domain_rand:
    randomize_friction = True
    friction_range = [0.05, 2.25]
    restitution_range = [0.0, 0.5]
    randomize_base_mass = True
    added_mass_range = [-1., 3.]
    randomize_base_com = True
    added_com_range = [-0.05, 0.05]
```

代码块

```
randomize_friction = True
friction_range = [0.05, 2.25]
restitution_range = [0.0, 0.5]
randomize_base_mass = True
```

```
5   added_mass_range = [-1., 3.]
6   randomize_base_com = True
7   added_com_range = [-0.05, 0.05]
```

在_process_rigid_shape_props函数中

```
def _process_rigid_shape_props(self, props, env_id):
    """ Callback allowing to store/change/randomize the rigid shape properties of each environment.
        Called During environment creation.
        Base behavior: randomizes the friction of each environment

    Args:
        props (List[gymapi.RigidShapeProperties]): Properties of each shape of the asset
        env_id (int): Environment id

    Returns:
        [List[gymapi.RigidShapeProperties]]: Modified rigid shape properties
    """
    if self.cfg.domain_rand.randomize_friction:
        if env_id==0:
            # prepare friction randomization
            friction_range = self.cfg.domain_rand.friction_range
            restitution_range = self.cfg.domain_rand.restitution_range

            num_buckets = 64
            bucket_ids = torch.randint(0, num_buckets, (self.num_envs, 1))
            friction_buckets = torch_rand_float(friction_range[0], friction_range[1], (num_buckets,1), device='cpu')
            restitution_buckets = torch_rand_float(restitution_range[0], restitution_range[1], (num_buckets,1), device='cpu')

            self.friction_coeffs = friction_buckets[bucket_ids]
            self.restitutions = restitution_buckets[bucket_ids]

        for s in range(len(props)):
            props[s].friction = self.friction_coeffs[env_id]
            props[s].friction = self.friction_coeffs[env_id]

    return props
```

代码块

```
1           if self.cfg.domain_rand.randomize_friction:
2               if env_id==0:
3                   # prepare friction randomization
4                   friction_range = self.cfg.domain_rand.friction_range
5                   restitution_range = self.cfg.domain_rand.restitution_range
6
7                   num_buckets = 64
8                   bucket_ids = torch.randint(0, num_buckets, (self.num_envs, 1))
9                   friction_buckets = torch_rand_float(friction_range[0],
    friction_range[1], (num_buckets,1), device='cpu')
10                  restitution_buckets = torch_rand_float(restitution_range[0],
    restitution_range[1], (num_buckets,1), device='cpu')
11
12                  self.friction_coeffs = friction_buckets[bucket_ids]
13                  self.restitutions = restitution_buckets[bucket_ids]
14
15              for s in range(len(props)):
```

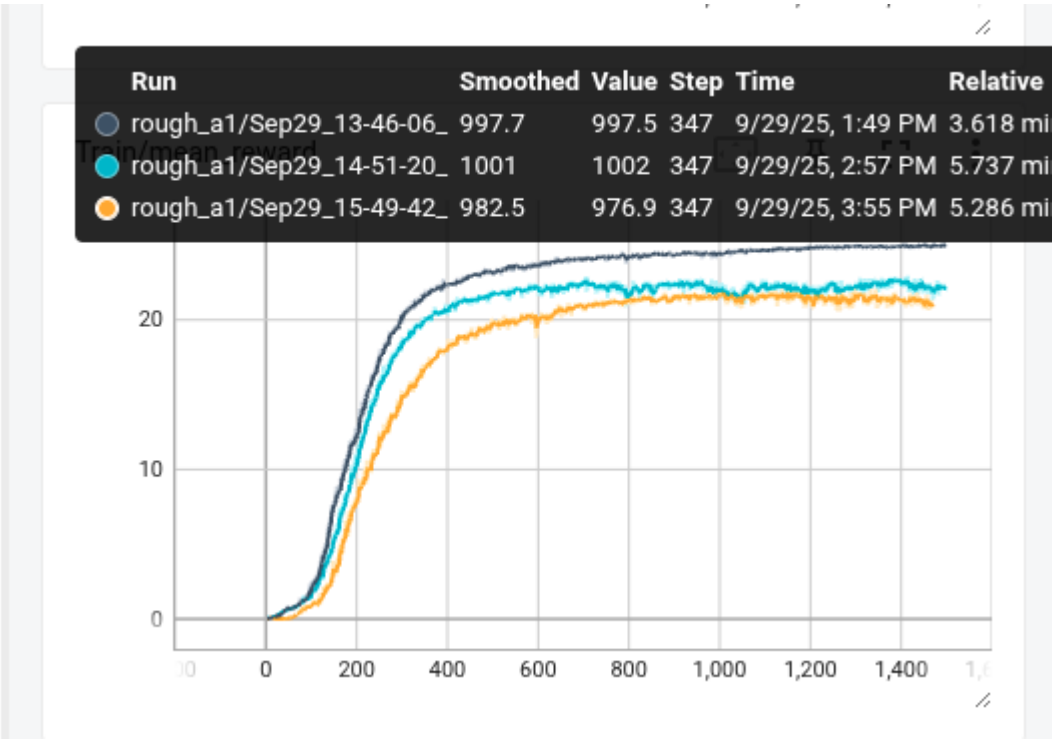
```
                props[s].friction = self.friction_coeffs[env_id]
                props[s].friction = self.friction_coeffs[env_id]

```

_process_rigid_body_props中添加代码

```python
    def _process_rigid_body_props(self, props, env_id):
        # if env_id==0:
        #     sum = 0
        #     for i, p in enumerate(props):
        #         sum += p.mass
        #         print(f"Mass of body {i}: {p.mass} (before randomization)")
        #     print(f"Total mass {sum} (before randomization)")
        # randomize base mass
        if self.cfg.domain_rand.randomize_base_mass:
            rng = self.cfg.domain_rand.added_mass_range
            props[0].mass += np.random.uniform(rng[0], rng[1])
        if self.cfg.domain_rand.randomize_base_com:
            rng_com = self.cfg.domain_rand.added_com_range
            rand_com = np.random.uniform(rng_com[0], rng_com[1], size=(3, ))
            props[0].com += gymapi.Vec3(*rand_com)

        return props
```

代码块

```
          if self.cfg.domain_rand.randomize_base_com:
              rng_com = self.cfg.domain_rand.added_com_range
              rand_com = np.random.uniform(rng_com[0], rng_com[1], size=(3, ))
              props[0].com += gymapi.Vec3(*rand_com)
```



总奖励还在下降

3.增加全身质量随机和质心随机

```
class domain_rand:

    randomize_leg_mass = True
    factor_leg_mass_range = [0.85,1.15]

    randomize_leg_com = True
    added_leg_com_range = [-0.015, 0.015]
```

代码块

```
1   randomize_leg_mass = True
2   factor_leg_mass_range = [0.85,1.15]
3
4   randomize_leg_com = True
5   added_leg_com_range = [-0.015, 0.015]
6
```

***_process_rigid_body_props*中添加：**

代码块

```
1           if self.cfg.domain_rand.randomize_leg_mass:
2               factor_leg_mass = self.cfg.domain_rand.factor_leg_mass_range
3               for i in range(len(props)):
```

```
                mess = np.random.uniform(factor_leg_mass[0], factor_leg_mass[1])
                props[i].mass *= mess

        if self.cfg.domain_rand.randomize_leg_com:
            rng_leg_com = self.cfg.domain_rand.added_leg_com_range
            rand_leg_com = np.random.uniform(rng_leg_com[0], rng_leg_com[1], size=(3, ))
            for i in range(len(props)):
                props[i].com += gymapi.Vec3(*rand_leg_com)

```

```
def _process_rigid_body_props(self, props, env_id):
    # if env_id==0:
    #     sum = 0
    #     for i, p in enumerate(props):
    #         sum += p.mass
    #         print(f"Mass of body {i}: {p.mass} (before randomization)")
    #     print(f"Total mass {sum} (before randomization)")
    # randomize base mass
    if self.cfg.domain_rand.randomize_base_mass:
        rng = self.cfg.domain_rand.added_mass_range
        props[0].mass += np.random.uniform(rng[0], rng[1])

    if self.cfg.domain_rand.randomize_base_com:
        rng_com = self.cfg.domain_rand.added_com_range
        rand_com = np.random.uniform(rng_com[0], rng_com[1], size=(3, ))
        props[0].com += gymapi.Vec3(*rand_com)

    if self.cfg.domain_rand.randomize_leg_mass:
        factor_leg_mass = self.cfg.domain_rand.factor_leg_mass_range
        for i in range(len(props)):
            mess = np.random.uniform(factor_leg_mass[0], factor_leg_mass[1])
            props[i].mass *= mess

    if self.cfg.domain_rand.randomize_leg_com:
        rng_leg_com = self.cfg.domain_rand.added_leg_com_range
        rand_leg_com = np.random.uniform(rng_leg_com[0], rng_leg_com[1], size=(3, ))
        for i in range(len(props)):
            props[i].com += gymapi.Vec3(*rand_leg_com)

    return props
```

五.修改奖励函数

由于开源代码训练出来的效果不好，随对a1修改奖励函数

1.加入身高惩罚：

```
        self_collisions = 1 # 1 to disable, 0 to enable...bit

    class rewards( LeggedRobotCfg.rewards ):
        soft_dof_pos_limit = 0.9
        base_height_target = 0.30
        class scales( LeggedRobotCfg.rewards.scales ):
            torques = -0.0002
```

(class) scales

```
    dof_acc = -2.5e-7
    base_height = -20.
    feet_air_time =  1.0
    collision = -1
```

2.加入站姿惩罚

```
action_rate = -0.01
stand_still = -1
```

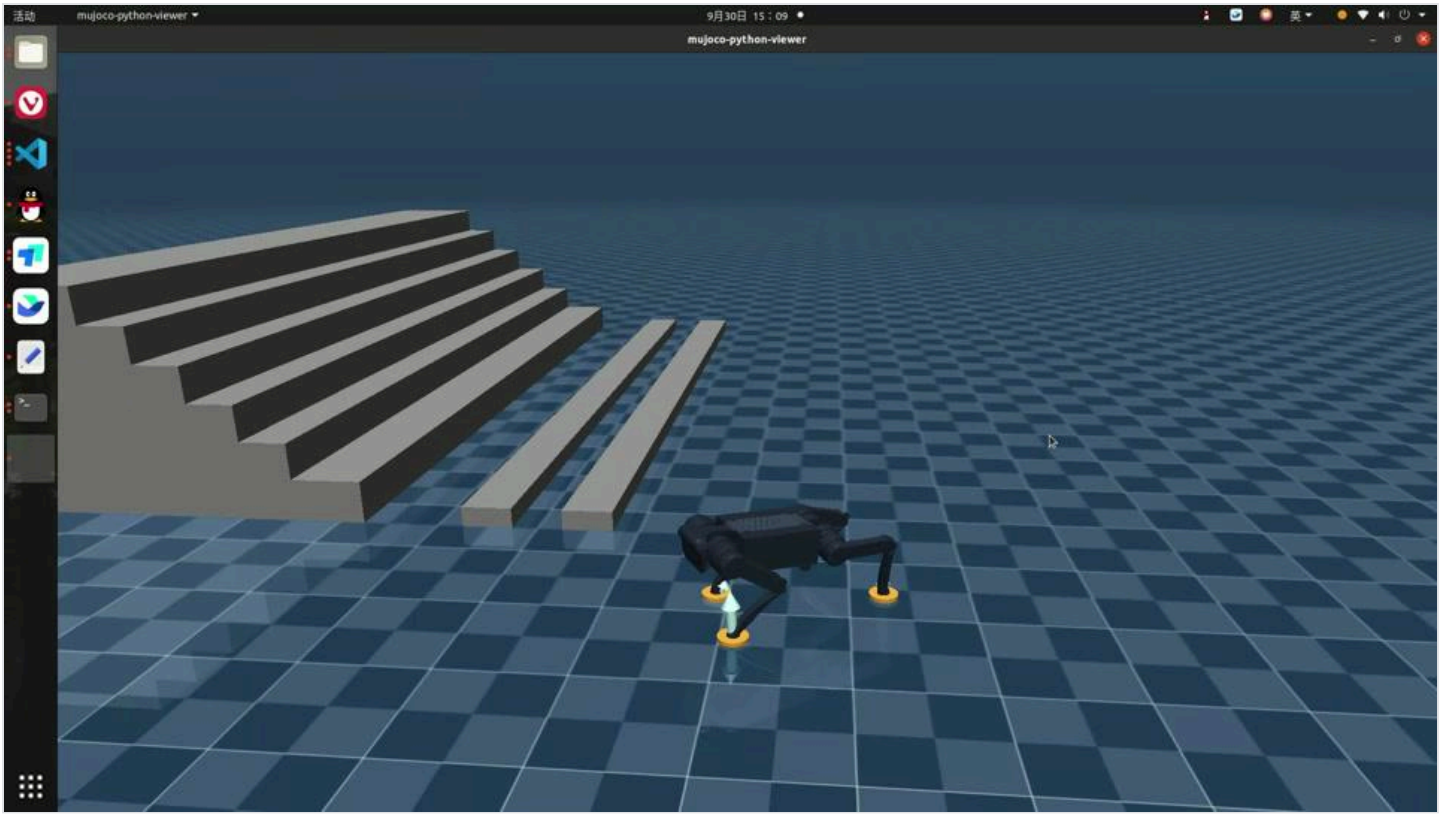
3.对髋关节缩放进行调整

```
解释代码 | 生成文档 | 修复代码 | 生成测试 | 代码评审 | 关闭
def _compute_torques(self, actions):
    """ Compute torques from actions.
        Actions can be interpreted as position or velocity targets given to a
        [NOTE]: torques must have the same dimension as the number of DOFs, e

    Args:
        actions (torch.Tensor): Actions

    Returns:
        [torch.Tensor]: Torques sent to the simulation
    """
    #pd controller
    actions_scaled = actions * self.cfg.control.action_scale
    control_type = self.cfg.control.control_type
    actions_scaled[:,[0,3,6,9]]*=0.5
```

sim2sim中也同样如此：

```
    action_flt = action *0.25
    action[0] *= 0.5
    action[3] *= 0.5
    action[6] *= 0.5
    action[9] *= 0.5
    joint_pos_target = action_flt + default_dof_pos
    target_q=joint_pos_target
```

最后效果：



结束语：本文只帮助大家快速实现sim2sim，并不是说看完以后就能跑出很好的效果，无论何事都需要细细打磨不断学习，祝大家学有所成。