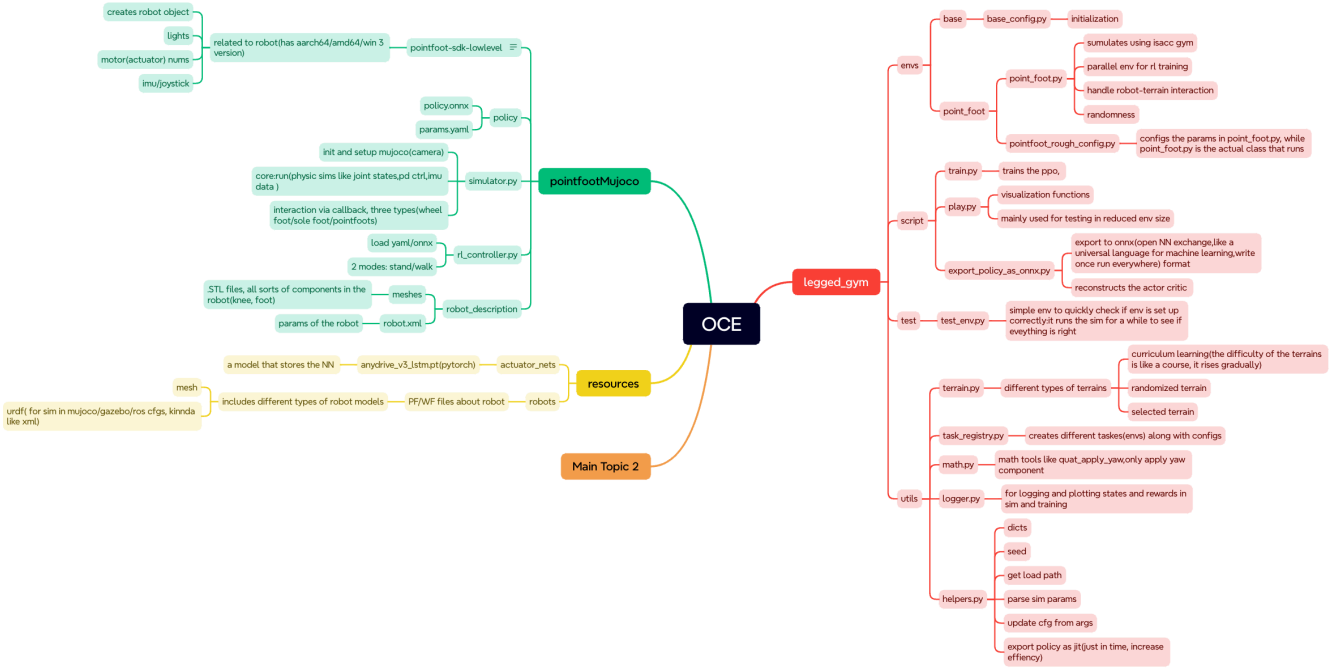
Code explanation for Pointfoot robot

Yitao Zeng, Luo Yizhou, Qianli Hao

1 Full code structure



2 Detailed explanation

2.0.1 legged_gym

This is the core folder of the reinforcement learning project.

`Envs` :

this is where the training and simulation related parameters and functions lie. Most of the code we add is also located in this section. It includes `pointfoot.py` and `pointfoot_rough_config.py`.

`pointfoot.py` :

1. Related to simulation and control:

   steps/reset/physical steps（using decimation to control the stepping frequency)
2. Related to observations and rewards:

   There are two types of observations:

   1.Proprioceptive observations are **internal body-state measurements** that describe the robot's own physical configuration and motion, *without* external sensing (like cameras or terrain scans).

2.Privileged observations typically include simulation-only or hard-to-measure states that wouldn't be available in the real world, such as:

- Exact terrain heightmaps
- Ground truth contact forces
- External disturbance parameters (e.g., friction coefficients)
- Perfect robot state estimates (no sensor noise)

After obtaining the observations, rewards and punishments are set to let the robot walk in a desired way, below are key patterns of reward functions(here we do not list an exhaustive list of rewards functions):

- **Tracking rewards** (7): Exponential functions for commanded states
- **Safety penalties** (9): Hard limits on physical quantities
- **Gait rewards** (7): Focus on foot synchronization/placement, like the walking steps of the robot.
- **Smoothness rewards** (3): Temporal consistency in actions

3. Related to terrains and environmenst:

   Creates different terrain types
4. Related to robot control:

   Computing the torques and feet air time.

`pointfoot_rough_config.py`:

This file contains all the configuration related to `pointfoot.py` , including environments(obs, episodes), terrains, giving commands to the robot(linear and angular velocity), robotic assets, control, states, and property(like stiffness and damping),normalizations(for the neural network),reward parameters; noise, running policy and so on.

From our perspective, the following factors are less critical:

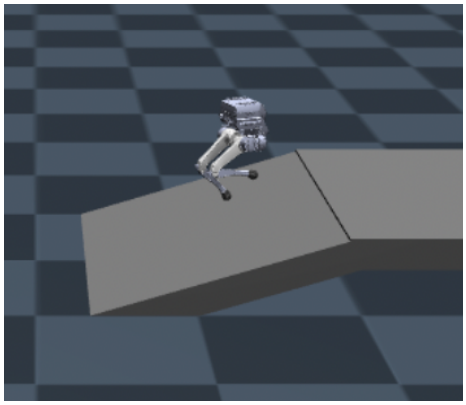
`Script`:

This contains the useful codes of running the simulation and training.

`util`:

As its name indicates, this folder contains the utilities of supporting the code structure.

2.0.2 pointfootMujoco



The environment of the above codes is isaacgym, here Mujoco is another simulation environment. We take the policy(`policy.onnx`) trained by `train.py` and use it on the robot in Mujoco to test the generalization of the robotic policy.

2.0.3 Resources

Contains the actuator nets and robotic structures.