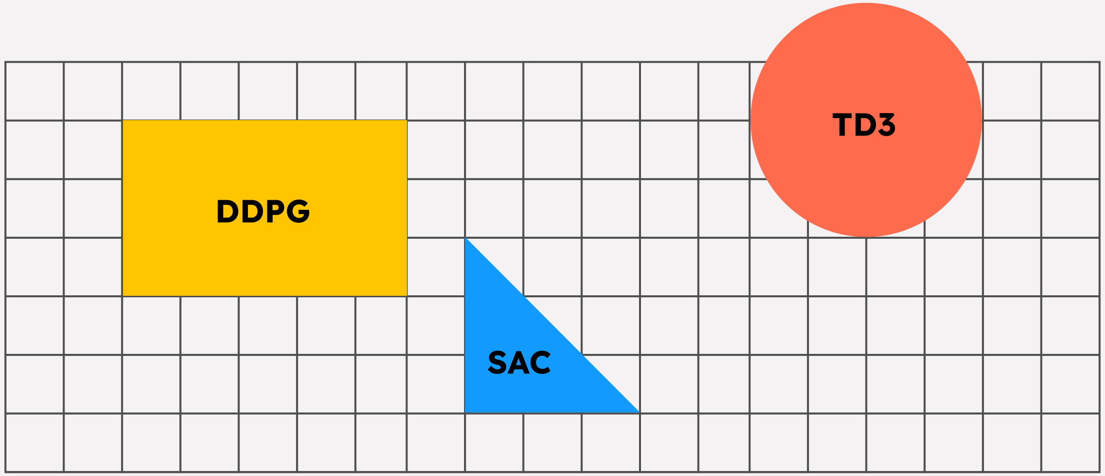
SAC-based hybrid approaches on Mujoco environments



Kexuan Shen - 135001421

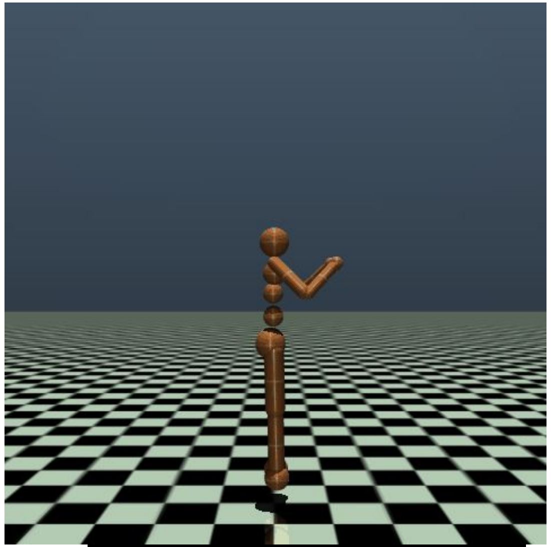
Rohith Yogi - 135001467

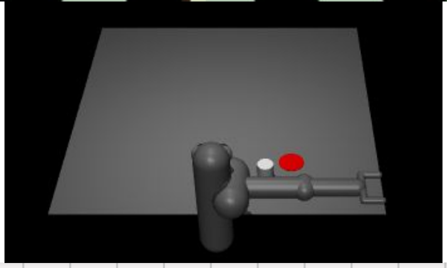
About the Project

This project is based on Aalto University's Mujoco ant model

MuJoCo - Multi Join dynamics with Contact

- A physics engine that simulates articulates structures iterating with their environment
- Although the goal is focused on simulation to reality we just look at how to use PPO and TD3 as baseline to improve the total reward.
- This problem is inherently **sequential**.
- We used SAC to apply to the pusher and humanoid to be the baseline model.



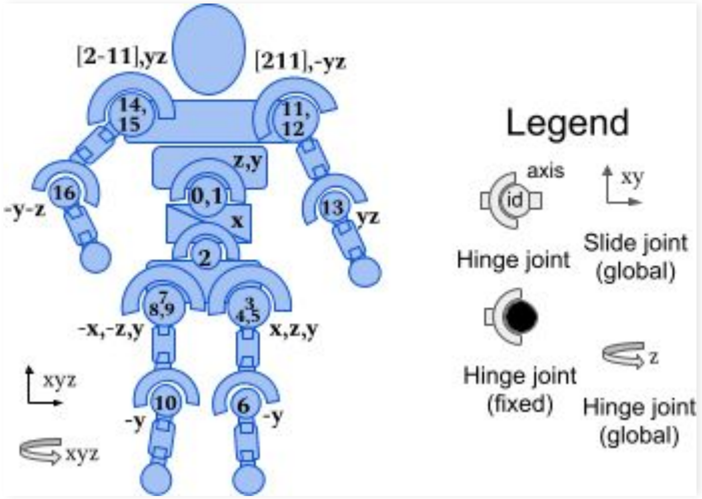


Twin-delayed deep deterministic (TD3)
Proximal policy optimization (PPO)
Soft Actor-Critic (SAC)

Environments

Taking the ant robot as example

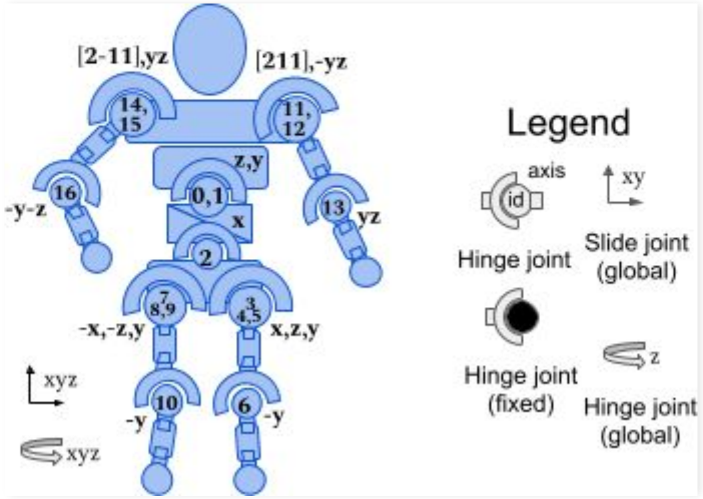
- State Space
  - Angular velocity of different parts of robot
  - Coordinates of various body parts centers
- Action Space
  - Torque applied to 8 hinge points/rotatable joints
  - Motors in the physical robot
- Transition function and reward functions are predefined in Mujoco environment
- Original plan : Focus more on algorithms, running baseline PPO and TD3 with modifications to optimization process and introduce new regularization techniques.

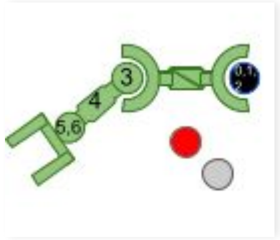


Related work

But after reviewing more papers related to continuous control

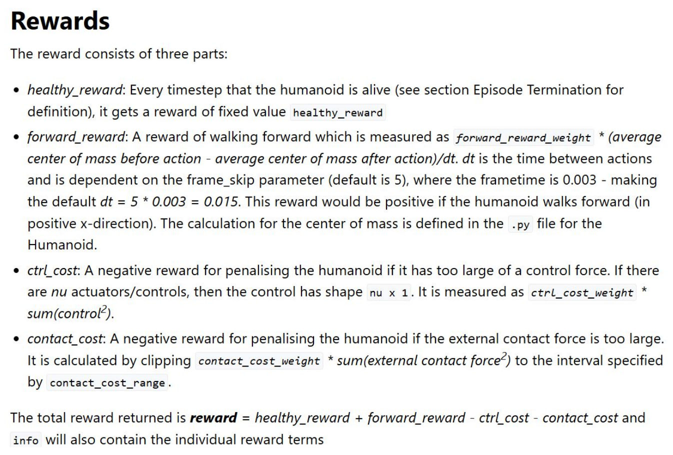
- We still need to benchmark DDPG on Humanoid and pusher since the paper was 8 years as of 2024.
- We have a deeper understanding of TD3 and SAC in continuous control about how they improved the performance.
- We take inspiration from Monte Carlo and TRPO to develop hybrid method based on SAC.
- We also wish to add constraints in the surroundings to conserve energy but is not done.
- We found some algorithms that are proven to work in such constraint environments.





Deep Deterministic Policy Gradient (DDPG)

Reward Function

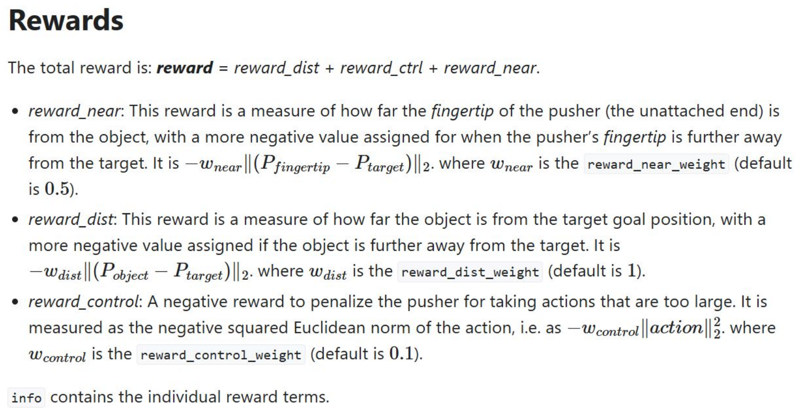


Rewards

The reward consists of three parts:

- *healthy_reward*: Every timestep that the humanoid is alive (see section Episode Termination for definition), it gets a reward of fixed value `healthy_reward`
- *forward_reward*: A reward of walking forward which is measured as `forward_reward_weight` * *(average center of mass before action - average center of mass after action)/dt*. *dt* is the time between actions and is dependent on the frame_skip parameter (default is 5), where the frametime is 0.003 - making the default *dt = 5 * 0.003 = 0.015*. This reward would be positive if the humanoid walks forward (in positive x-direction). The calculation for the center of mass is defined in the `.py` file for the Humanoid.
- *ctrl_cost*: A negative reward for penalising the humanoid if it has too large of a control force. If there are *nu* actuators/controls, then the control has shape `nu x 1`. It is measured as `ctrl_cost_weight` * *sum(control^2)*.
- *contact_cost*: A negative reward for penalising the humanoid if the external contact force is too large. It is calculated by clipping `contact_cost_weight` * *sum(external contact force^2)* to the interval specified by `contact_cost_range`.

The total reward returned is ***reward*** = *healthy_reward* + *forward_reward* - *ctrl_cost* - *contact_cost* and `info` will also contain the individual reward terms



Rewards

The total reward is: ***reward*** = *reward_dist* + *reward_ctrl* + *reward_near*.

- *reward_near*: This reward is a measure of how far the *fingertip* of the pusher (the unattached end) is from the object, with a more negative value assigned for when the pusher's *fingertip* is further away from the target. It is $-w_{near}\|(P_{fingertip} - P_{target})\|_2$. where w_{near} is the `reward_near_weight` (default is 0.5).
- *reward_dist*: This reward is a measure of how far the object is from the target goal position, with a more negative value assigned if the object is further away from the target. It is $-w_{dist}\|(P_{object} - P_{target})\|_2$. where w_{dist} is the `reward_dist_weight` (default is 1).
- *reward_control*: A negative reward to penalize the pusher for taking actions that are too large. It is measured as the negative squared Euclidean norm of the action, i.e. as $-w_{control}\|action\|_2^2$. where $w_{control}$ is the `reward_control_weight` (default is 0.1).

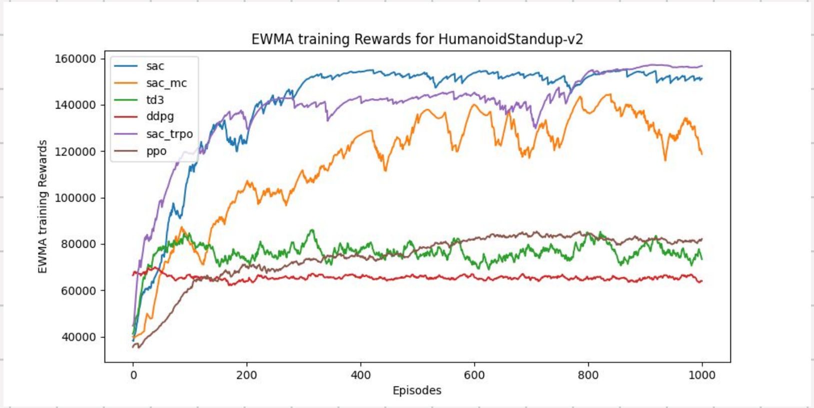
`info` contains the individual reward terms.

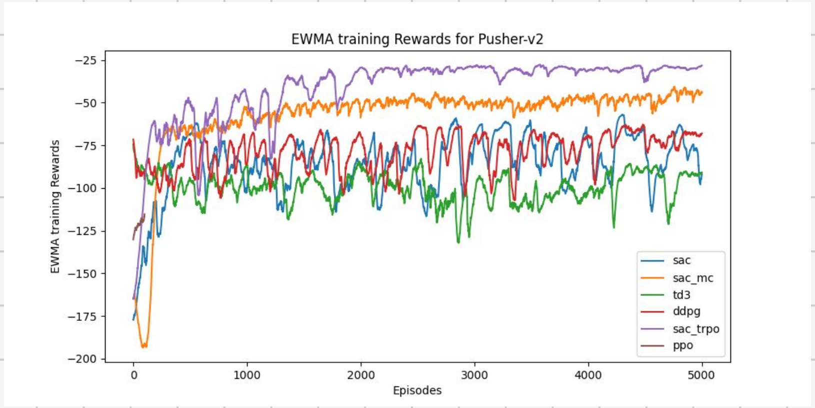
Implementation

The first hybrid model introduced an enhancement to the SAC algorithm by incorporating **Monte Carlo (MC) returns** for updating the critic network. Monte Carlo methods involve estimating the value of a state by averaging the returns obtained from multiple complete episodes starting from that state.
Unlike **Temporal Difference (TD)** methods, which rely on bootstrapping and update estimates based on the immediate reward and the value of the next state, Monte Carlo returns provide an unbiased and accurate estimation by considering the entire sequence of rewards.

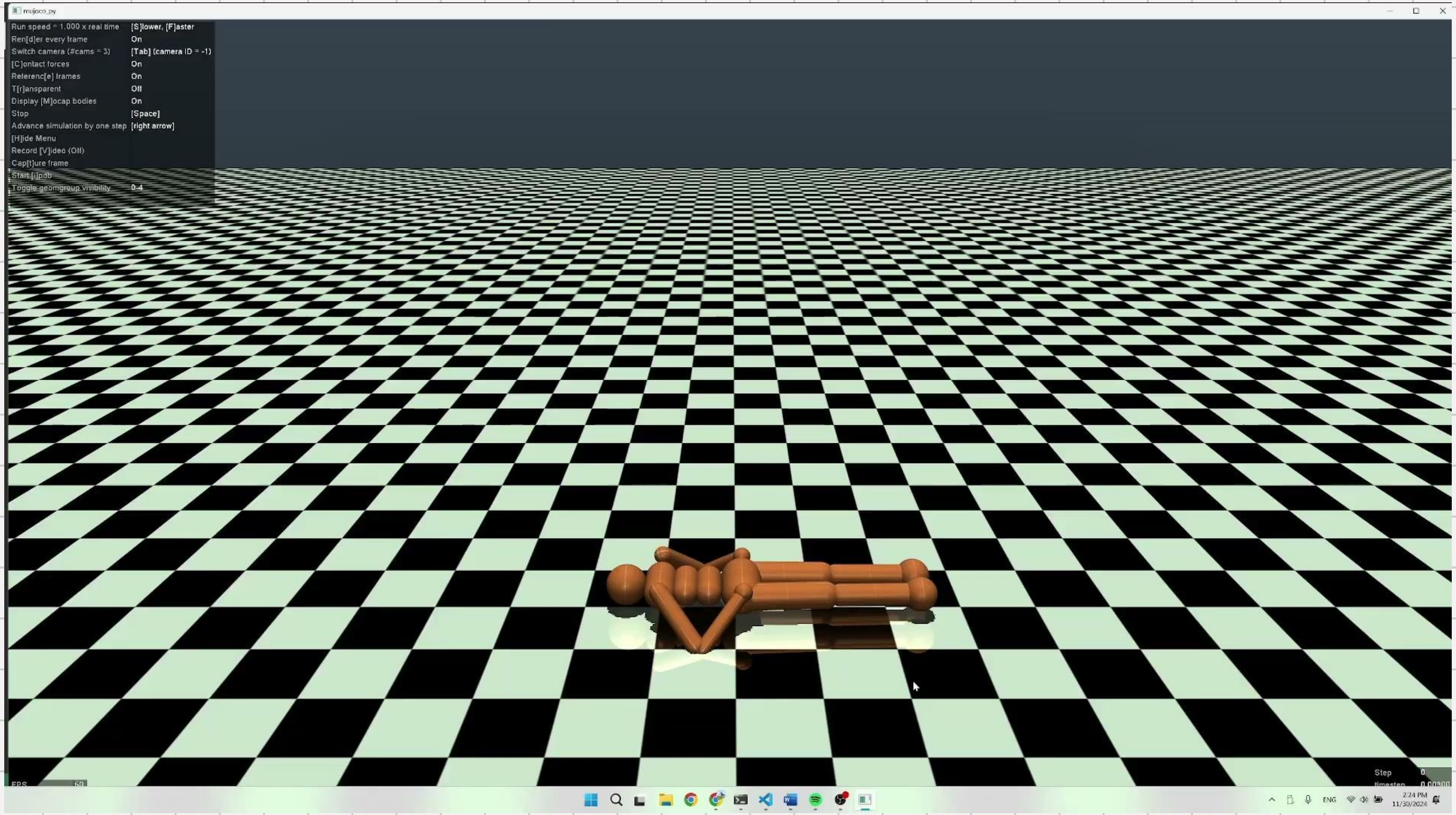
Building upon the first hybrid model, the second hybrid model introduced two key enhancements: the use of **n-step returns** and the incorporation of **Kullback-Leibler (KL) divergence** checks during parameter updates, inspired by the Trust Region Policy Optimization (TRPO) algorithm.

Results

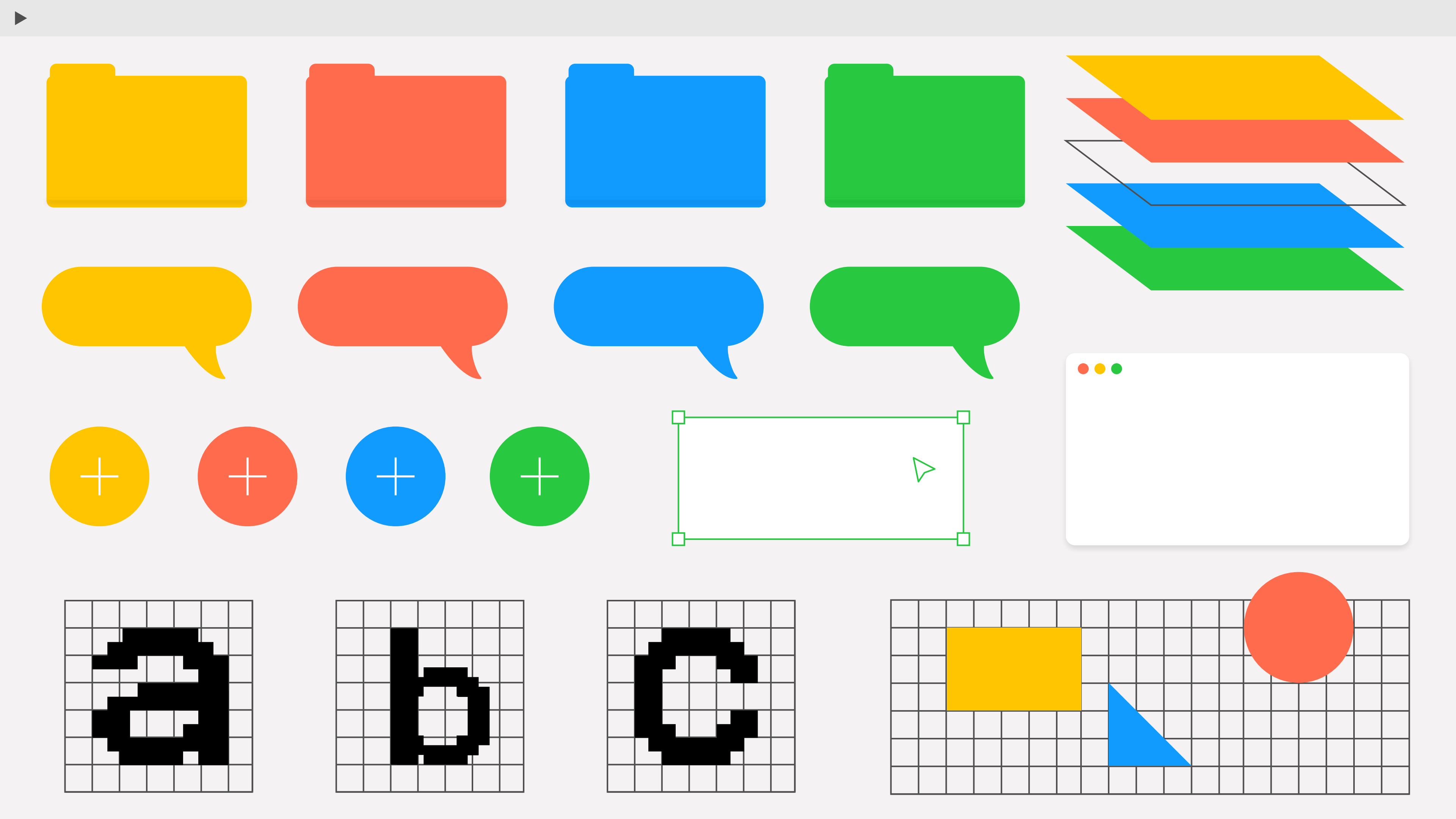

EWMA training Rewards for HumanoidStandup-v2
EWMA training Rewards
Episodes
sac
sac_mc
td3
ddpg
sac_trpo
ppo
160000
140000
120000
100000
80000
60000
40000
0
200
400
600
800
1000


EWMA training Rewards for Pusher-v2
EWMA training Rewards
Episodes
sac
sac_mc
td3
ddpg
sac_trpo
ppo
−25
−50
−75
−100
−125
−150
−175
−200
0
1000
2000
3000
4000
5000

Demo of humanoid



```
Run speed = 1.000 x real time       [S]lower, [F]aster
Ren[d]er every frame                On
Switch camera (#cams = 3)           [Tab] (camera ID = -1)
[C]ontact forces                    On
Referenc[e] frames                  On
T[r]ansparent                       Off
Display [M]ocap bodies              On
Stop                                [Space]
Advance simulation by one step      [right arrow]
[H]ide Menu
Record [V]ideo (Off)
Cap[t]ure frame
Start [i]pdb
Toggle geomgroup visibility         0-4
```


a
b
c