Gymnasium

Overview

- Open Source lib for building RL environments
  - Builds on OpenAI Gym
- Interoperability between environments/algorithms
- Tools for customization
- Compatible with
  - Different RL Libs
  - Vectorized environments
  - Interface for functional environments

Overview

Benchmarks

- Toy text: Blackjack, Taxi, Cliff Walking, Frozen Lake
  - Base Benchmarks, discrete environments
- Classic control: Cartpole, Acrobot, Mountain Car, Pendulum
  - Simple physic, continuous observation space
- Box2D: Bipedal Walker, Car Racing, Lunar Lander
  - Physic sim, collision detection
- Mujoco: Eleven physics based robot control environments
  - More realistic
- Several external environments: flappy bird, snake, …

Overview

Environment

- Reset, step, method
- Observation and action space

Step

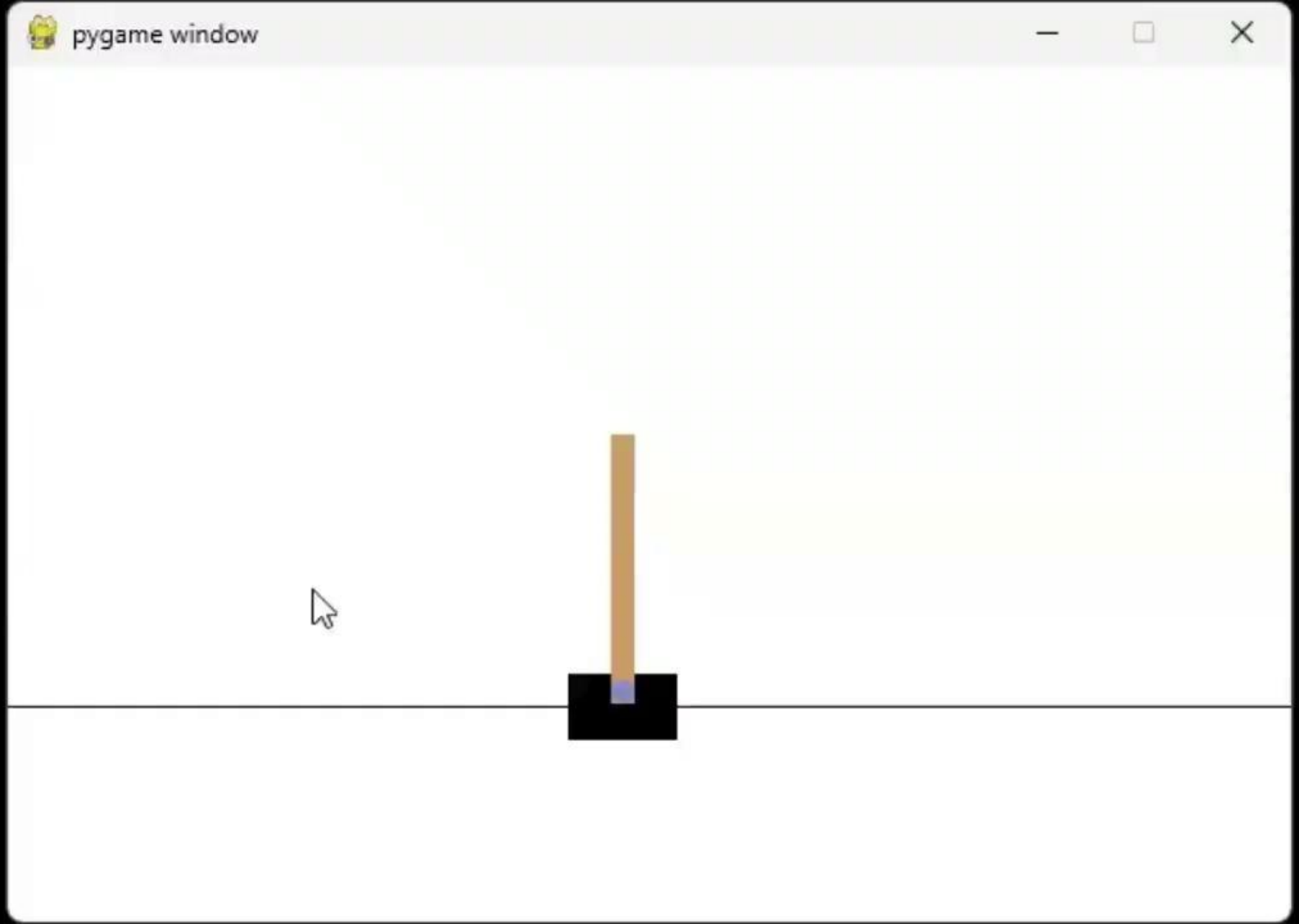
- Executes action -> Moves sim forward

VectorEnv

- Batch of identical independently running RL envs
- Same as env but batched -> Performance to parallizing
- Signals:
  - Termination: End of episode -> Failure or success -> State based
  - Truncation same but time based

Demo: CartPole

```python
from stable_baselines3 import PPO
import gymnasium as gym

# Create the environment
env = gym.make("CartPole-v1", render_mode=None)

# Initialize the environment
obs, info = env.reset()

model = PPO("MlpPolicy", env, verbose=1)


# Train the agent

model.learn(total_timesteps=100000, progress_bar=True)
model.save("pole")

env = gym.make("CartPole-v1", render_mode="human")

# Test the trained agent
obs, info = env.reset()
while True:
    action, _ = model.predict(obs, deterministic=True)
    obs, reward, done, truncated, info = env.step(action)
    env.render()
    if done or truncated:
        obs, info = env.reset()
```


pygame window

Demo: Donkey Kong

```python
from stable_baselines3 import PPO
import gymnasium as gym
import ale_py

# Create the environment
env = gym.make("ALE/DonkeyKong-v5", render_mode=None)

# Initialize the environment
obs, info = env.reset()

model = PPO("MlpPolicy", env, verbose=1)


# Train the agent
model.learn(total_timesteps=100000, progress_bar=True)
model.save("kong")

env = gym.make("ALE/DonkeyKong-v5", render_mode="human")

# Test the trained agent
obs, info = env.reset()
while True:
    action, _ = model.predict(obs, deterministic=True)
    obs, reward, done, truncated, info = env.step(action)
    env.render()
    if done or truncated:
        obs, info = env.reset()
```

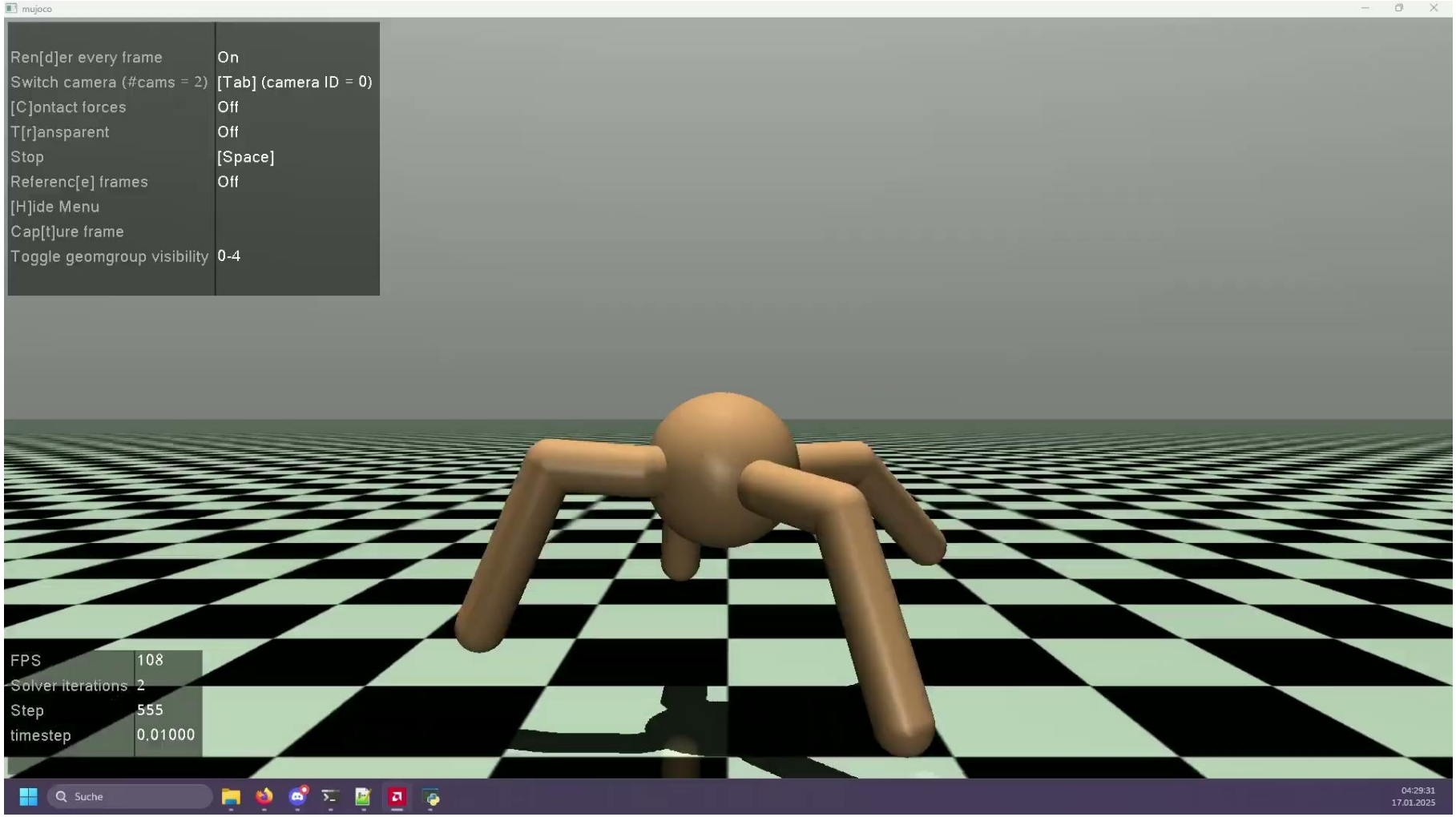

The Arcade Learning Environment
4400

Problems

- Loading of models failed due (apparently 93 GIB) large size
- Donkey Kong environment not running after training with too many steps

Demo: Ant

```
env = gym.make('Ant-v5', render_mode='human')

for episode in range(num_episodes):
    state, _ = env.reset()
    episode_reward = 0
    done = False

    while not done:
        env.render()  # Render each frame

        # Get action from the agent
        state_tensor = torch.FloatTensor(state).unsqueeze(0)
        with torch.no_grad():
            action, _ = agent.actor_critic.get_action(state_tensor)
        action_np = action.detach().numpy()

        # Take step in environment
        state, reward, terminated, truncated, _ = env.step(action_np)
        done = terminated or truncated
        episode_reward += reward

        # Add a small delay to make visualization viewable
        time.sleep(0.01)

    print(f"Episode {episode + 1} Reward: {episode_reward:.2f}")

env.close()
```


mujoco
Ren[d]er every frame
On
Switch camera (#cams = 2)
[Tab] (camera ID = 0)
[C]ontact forces
Off
T[r]ansparent
Off
Stop
[Space]
Referenc[e] frames
Off
[H]ide Menu
Cap[t]ure frame
Toggle geomgroup visibility
0-4
FPS
108
Solver iterations
2
Step
555
timestep
0,01000
Suche
04:29:31
17.01.2025