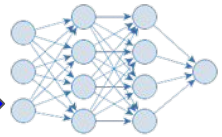
Welcome to:

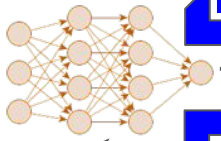
Hands-on RL with Ray's RLlib

A beginner's tutorial for working with environments, models, and algorithms

policy (in deep-RL, this is a neural network)

multi-GPU





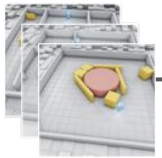
tf.keras / torch.nn.Modules

built-in LSTM + attention nets

agents

multi-agent

observations



arbitrary obs spaces

actions

15+ available algorithms (model-free; model-based; offline RL)

arbitrary act. spaces

rewards

+0.3

-1.0

reward shaping / learning w/o rewards

environment

custom callbacks

parallelize and distribute

Key differences: RL vs Supervised Learning

- Data collection loop is essential part of the RL algorithm, especially in the distributed setting.

- RL is forced to use unstable (bootstrapped) loss functions due to missing labels (rewards cannot cover absence of SL-style labels).

$$\mathbb{E}_{(s,a,r,s')}\left[\frac{1}{2}\left(\underbrace{R(s,a,s')+\gamma\max_{a'}Q(s',a';\theta)}_{\text{“labels” (bootstrapped using reward AND network output)}}-\underbrace{Q(s,a;\theta)}_{\text{predictions (actual network output)}}\right)^2\right]$$

- Exploration vs exploitation. An (online) RL algo can try new things in the environment.

- By default, RL tries to optimize over a time axis.

Overview of RLlib's Industry Users

Many Great Reasons for Using RLlib

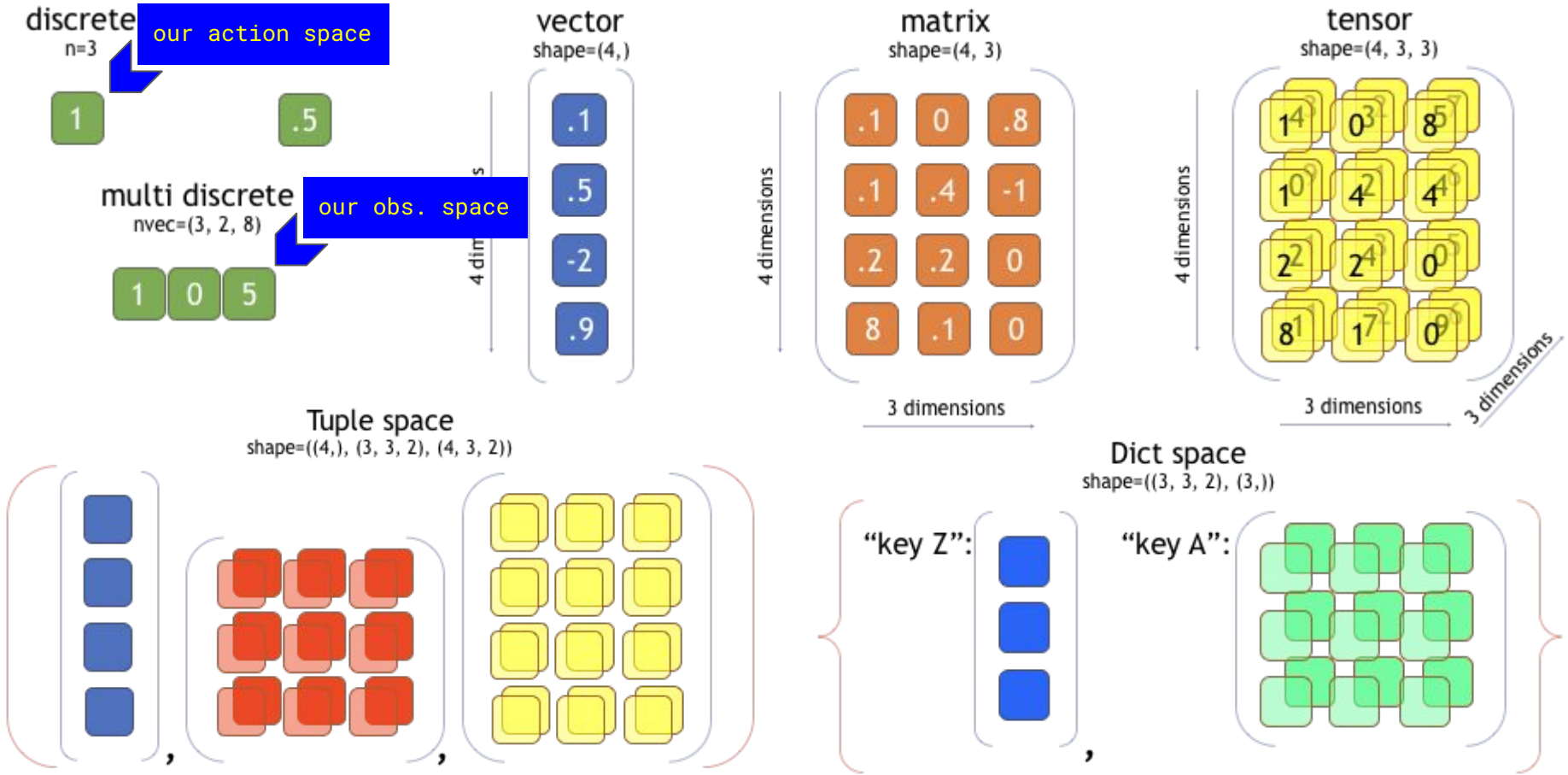
- RLlib is based on Ray, so it benefits from all its improvements on **performance** and **scalability**.

- RLlib is backed by Anyscale, the fast-growing, well-funded company behind Ray, offering strong **OSS support**.

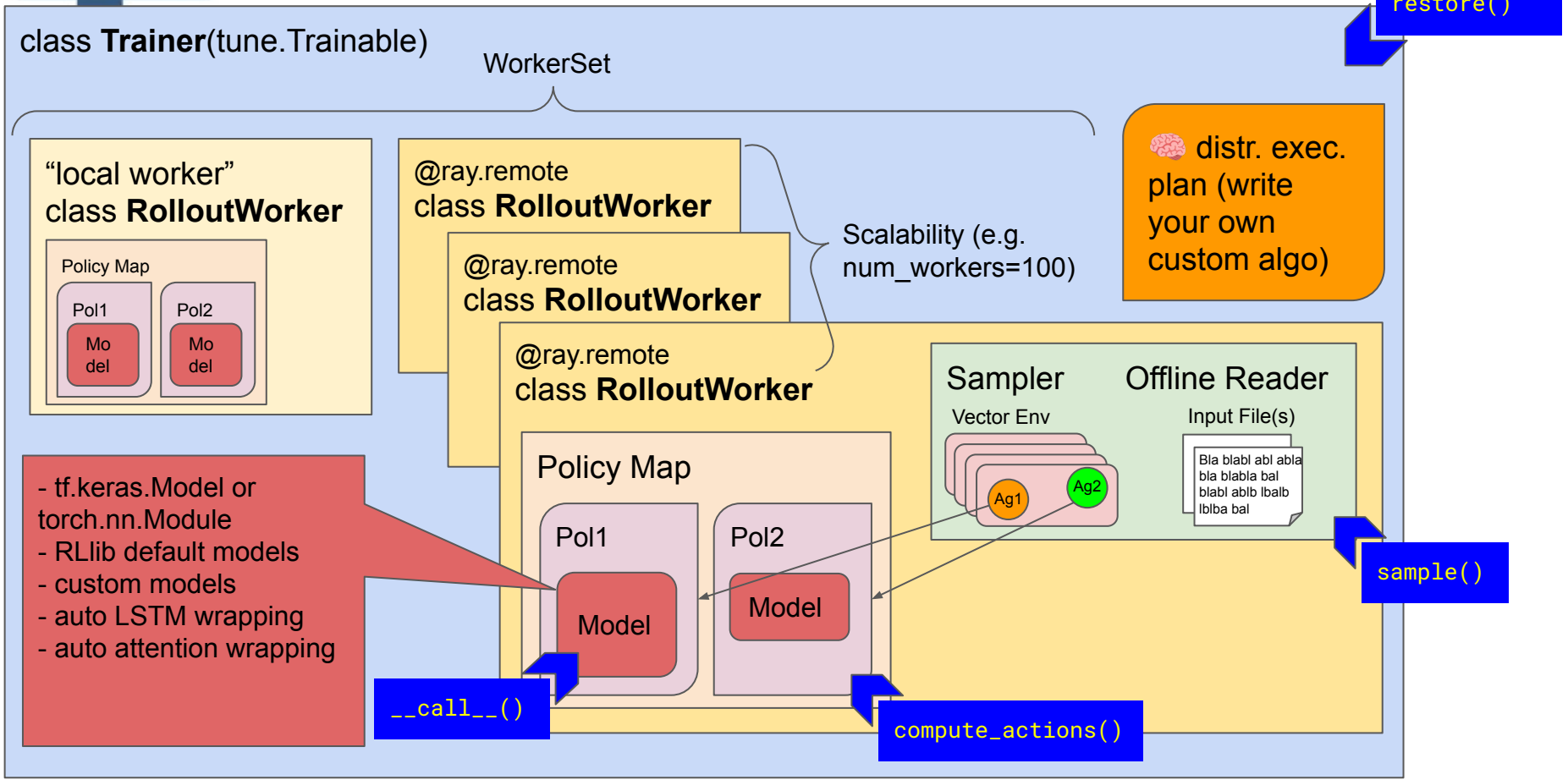
- RLlib is extremely **flexible**, allowing you to **customize every aspect** of the RL cycle and workflows.

- RLlib is good at solving **real-world** problems, supporting **offline RL**, **multi-agent** setups, **external simulators**, and more.

What's a Space?

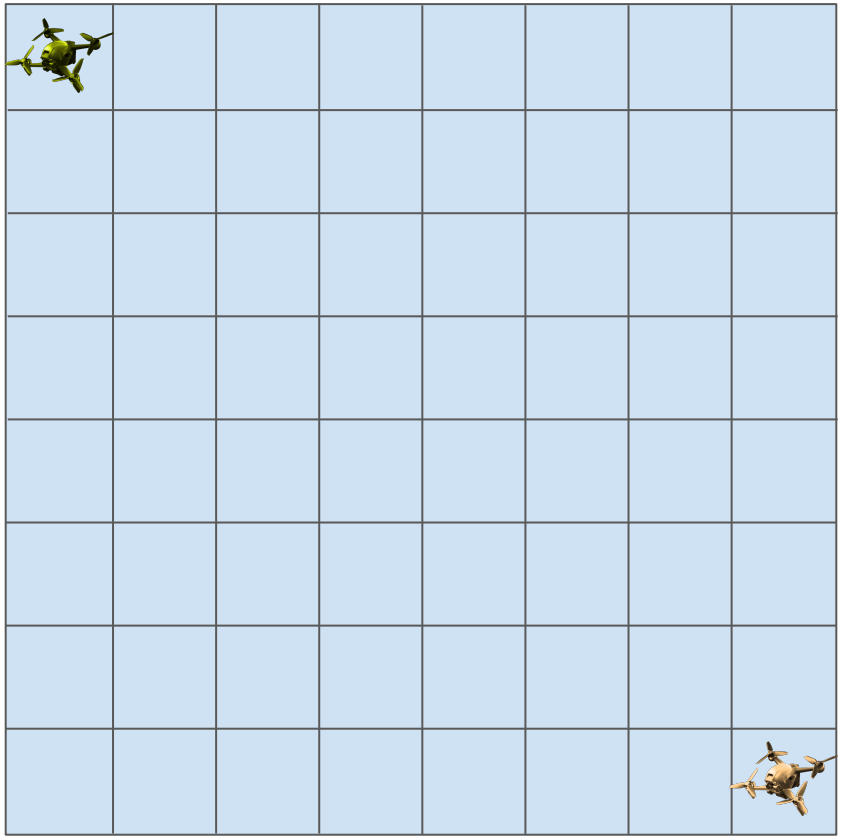


A Closer Look at RLlib



Our Environment - The “MultiAgentArena” 😶

action space: Discrete(4)
0 - up
1 - right
2 - down
3 - left



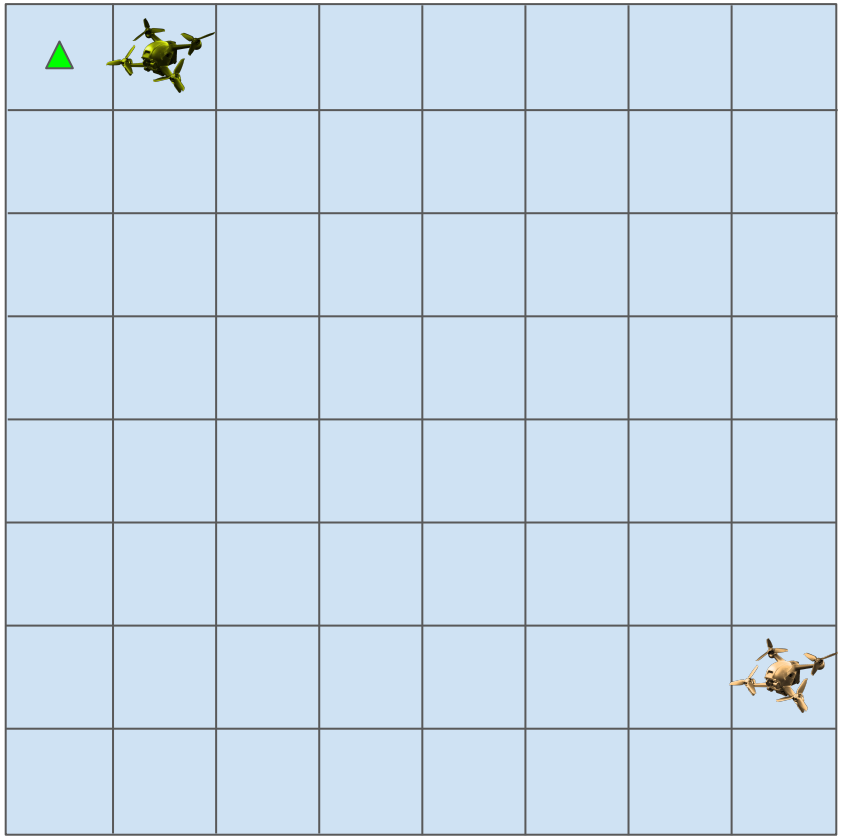
observation space:
MultiDiscrete([64, 64])

pos. agent 1

pos. agent 2

Our Environment - The “MultiAgentArena” 😶

action space: Discrete(4)
0 - up
1 - right
2 - down
3 - left



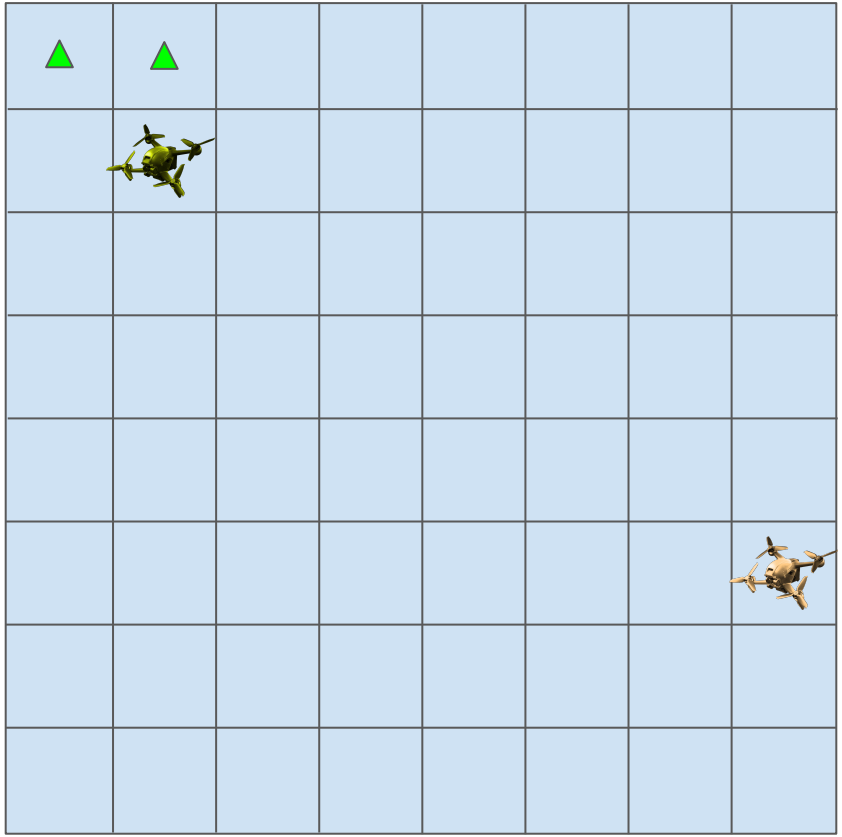
observation space:
MultiDiscrete([64, 64])

pos. agent 1

pos. agent 2

Our Environment - The “MultiAgentArena” 😶

action space: Discrete(4)
0 - up
1 - right
2 - down
3 - left



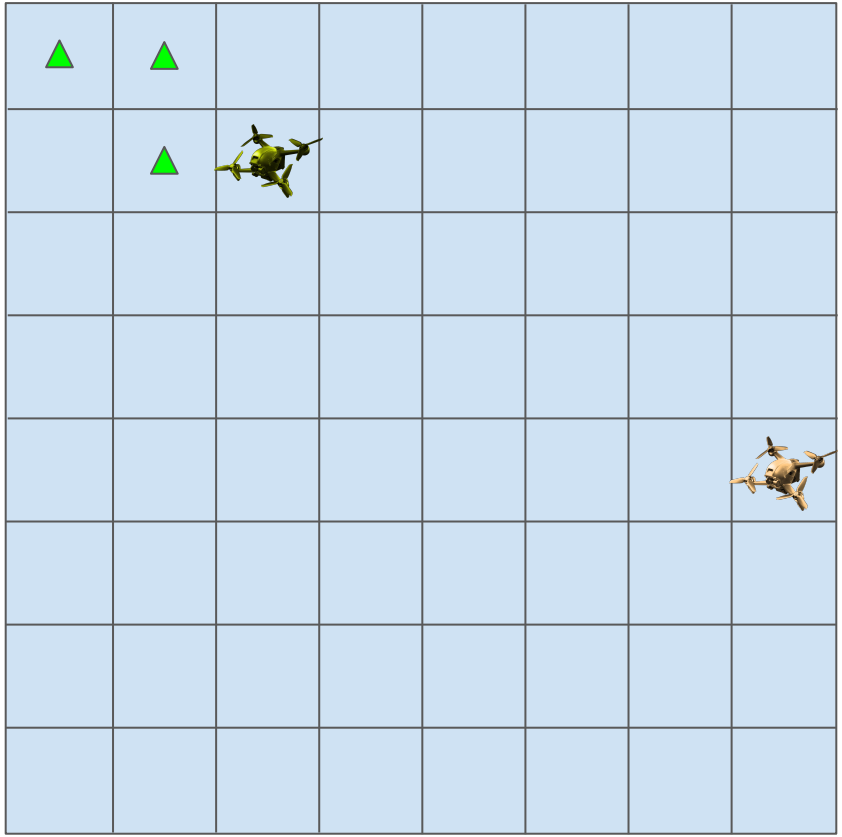
observation space:
MultiDiscrete([64, 64])

pos. agent 1

pos. agent 2

Our Environment - The “MultiAgentArena” 😶

action space: Discrete(4)
0 - up
1 - right
2 - down
3 - left



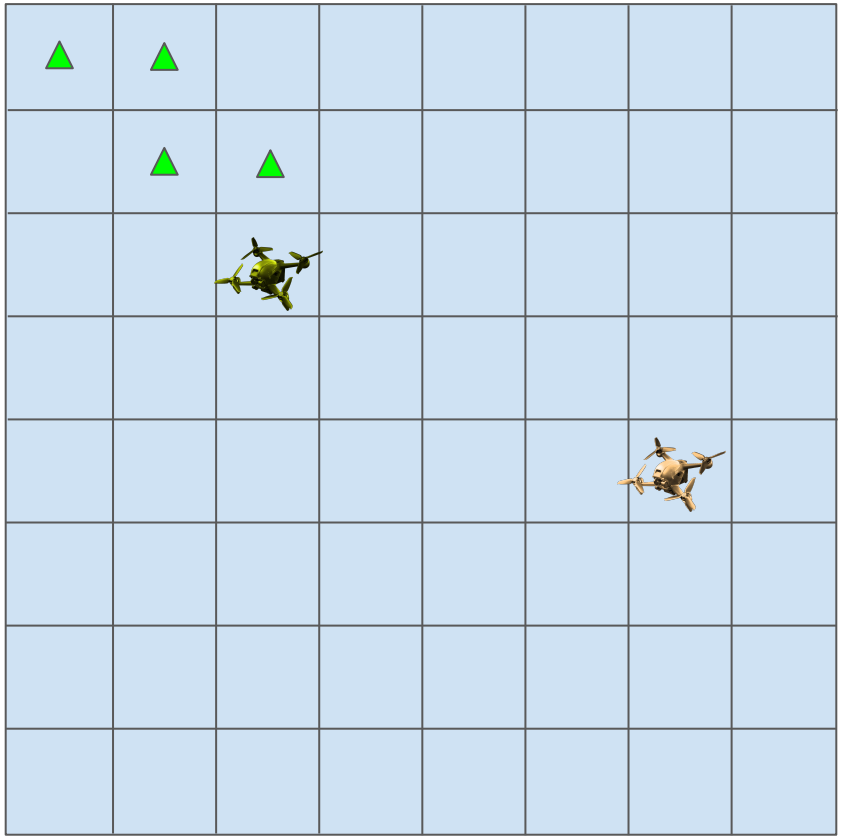
observation space:
MultiDiscrete([64, 64])

pos. agent 1

pos. agent 2

Our Environment - The “MultiAgentArena” 😶

action space: Discrete(4)
0 - up
1 - right
2 - down
3 - left



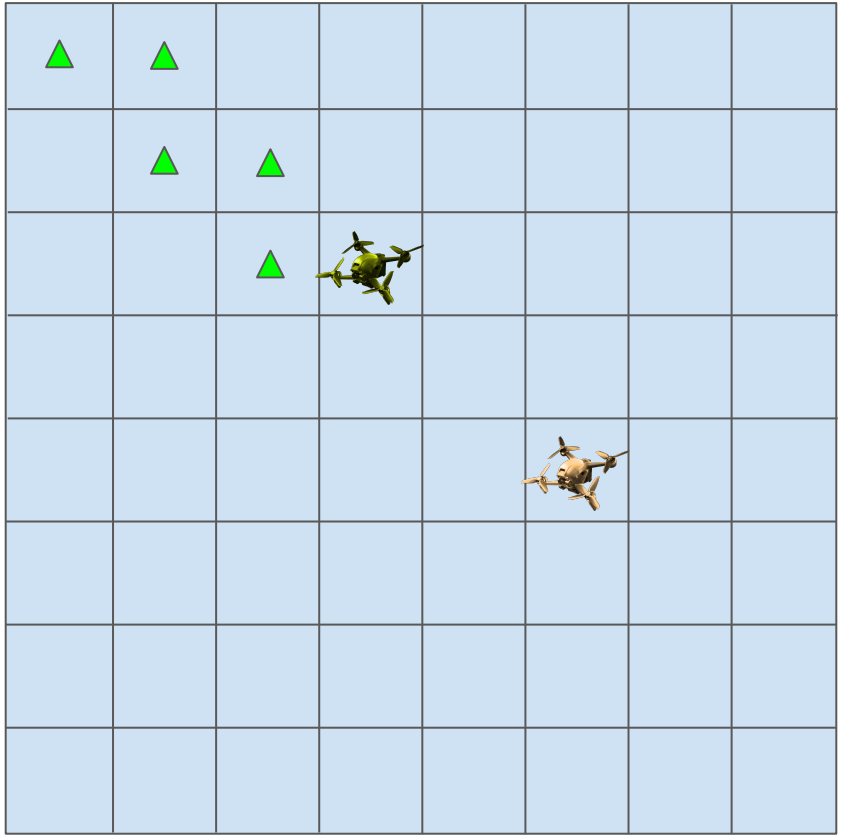
observation space:
MultiDiscrete([64, 64])

pos. agent 1

pos. agent 2

Our Environment - The “MultiAgentArena” 😶

action space: Discrete(4)
0 - up
1 - right
2 - down
3 - left



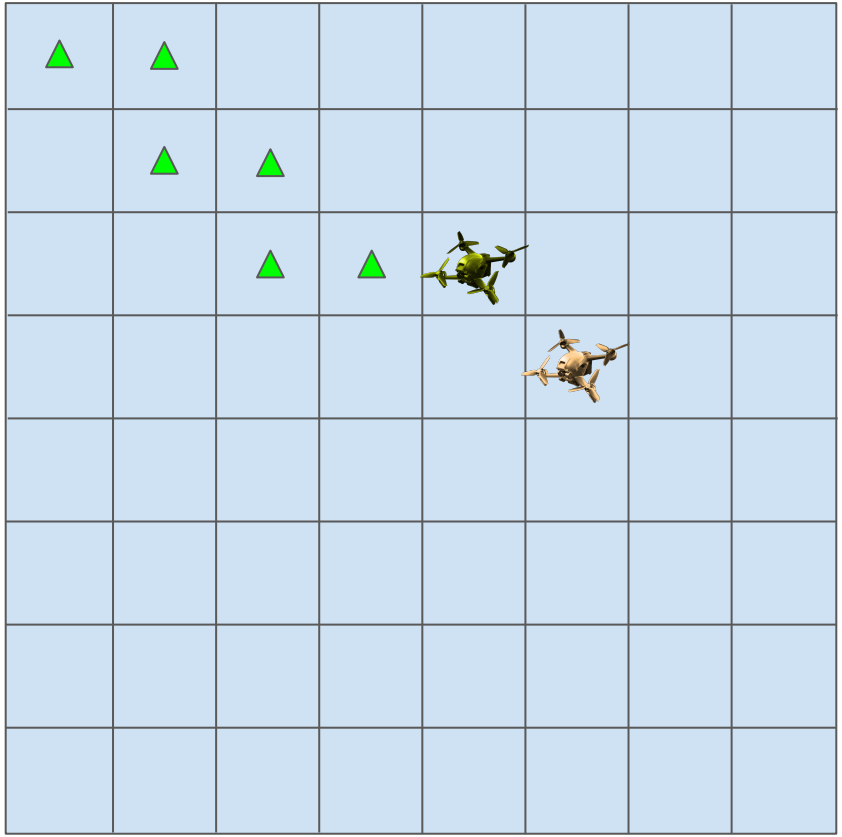
observation space:
MultiDiscrete([64, 64])

pos. agent 1

pos. agent 2

Our Environment - The “MultiAgentArena” 😶

action space: Discrete(4)
0 - up
1 - right
2 - down
3 - left



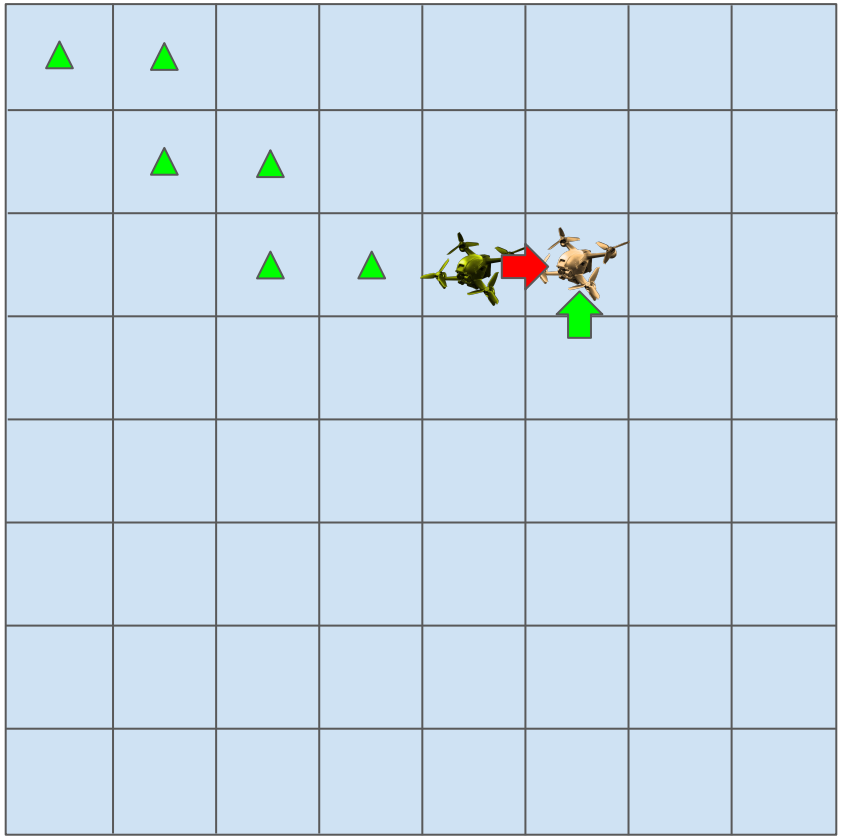
observation space:
MultiDiscrete([64, 64])

pos. agent 1

pos. agent 2

Our Environment - The “MultiAgentArena” 😶

action space: Discrete(4)
0 - up
1 - right
2 - down
3 - left



observation space:
MultiDiscrete([64, 64])

pos. agent 1

pos. agent 2