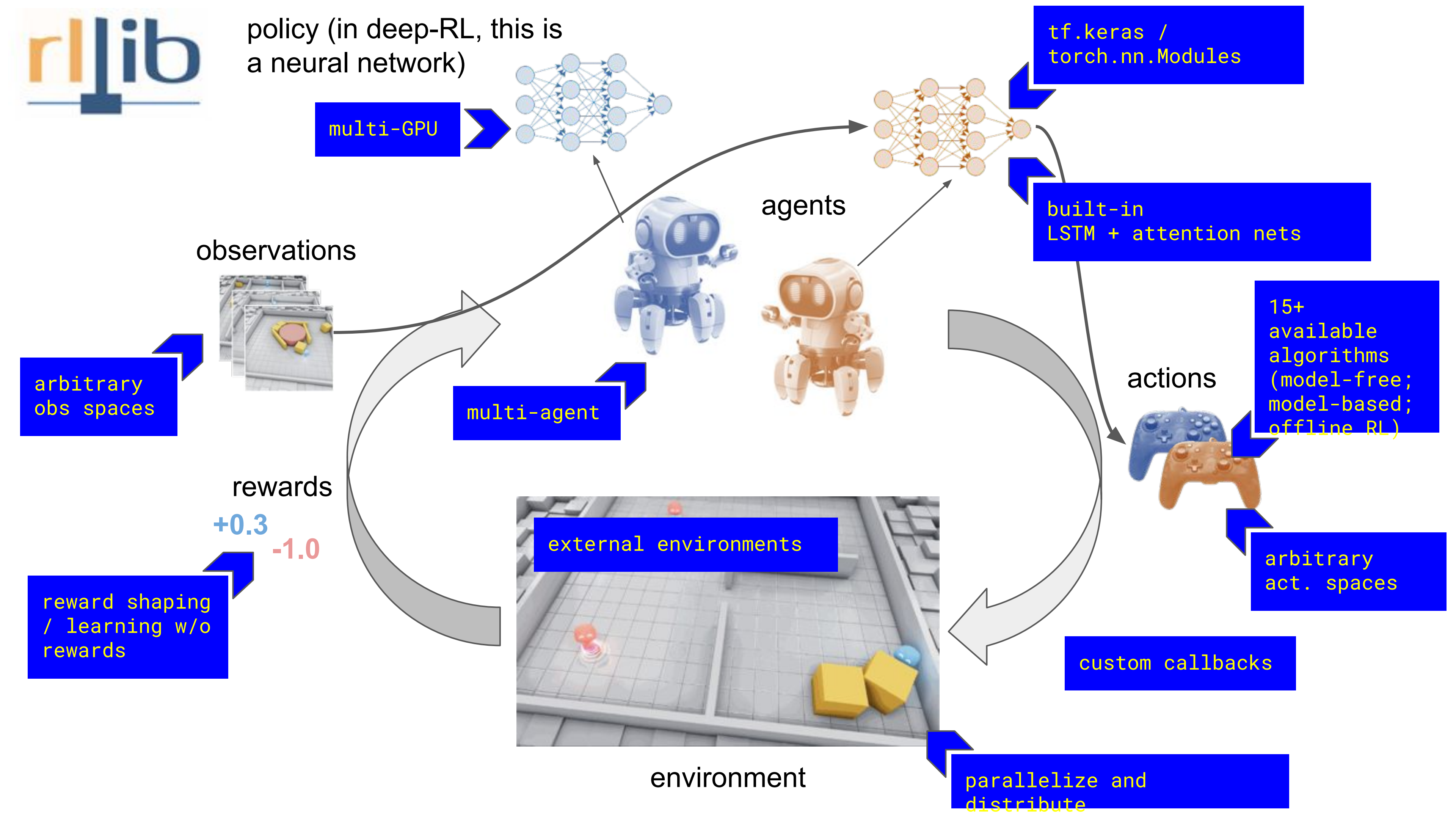
Welcome to:

Hands-on RL with Ray's RLlib

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

Who am I?

- Late 90s: Biochemistry (undergrad)
- Early-mid 2000s: Bioinformatics & NLP (PhD)
  "Why are you using NNs? SVMs are so much better and robust!"
- 2010s: Wall Street (Quant. Dev./Data Scientist)
- 2015+: Contributor to other OSS RL libraries (TensorForce, RLgraph, Surreal); Self-taught RL: Books, papers, online courses.
- 2019+: Anyscale Inc. (lead-dev RLlib)



rllib
policy (in deep-RL, this is a neural network)
multi-GPU
tf.keras / torch.nn.Modules
built-in LSTM + attention nets
agents
observations
arbitrary obs spaces
multi-agent
actions
15+ available algorithms (model-free; model-based; offline RL)
arbitrary act. spaces
rewards
+0.3
-1.0
reward shaping / learning w/o rewards
external environments
custom callbacks
environment
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.

rllib

Overview of RLlib's Industry Users

GM
GENERAL MOTORS

WILD LIFE

Microsoft

pathmind

UBISOFT

amazon web services

THIRD WAVE
AUTOMATION

2σ
TWO SIGMA

QUANTUMBLACK
A MCKINSEY COMPANY

ERICSSON

RIOT GAMES

DOW®

REPSOL

InstaDeep™

REWE
Besser leben.

Many Great Reasons for Using RLlib

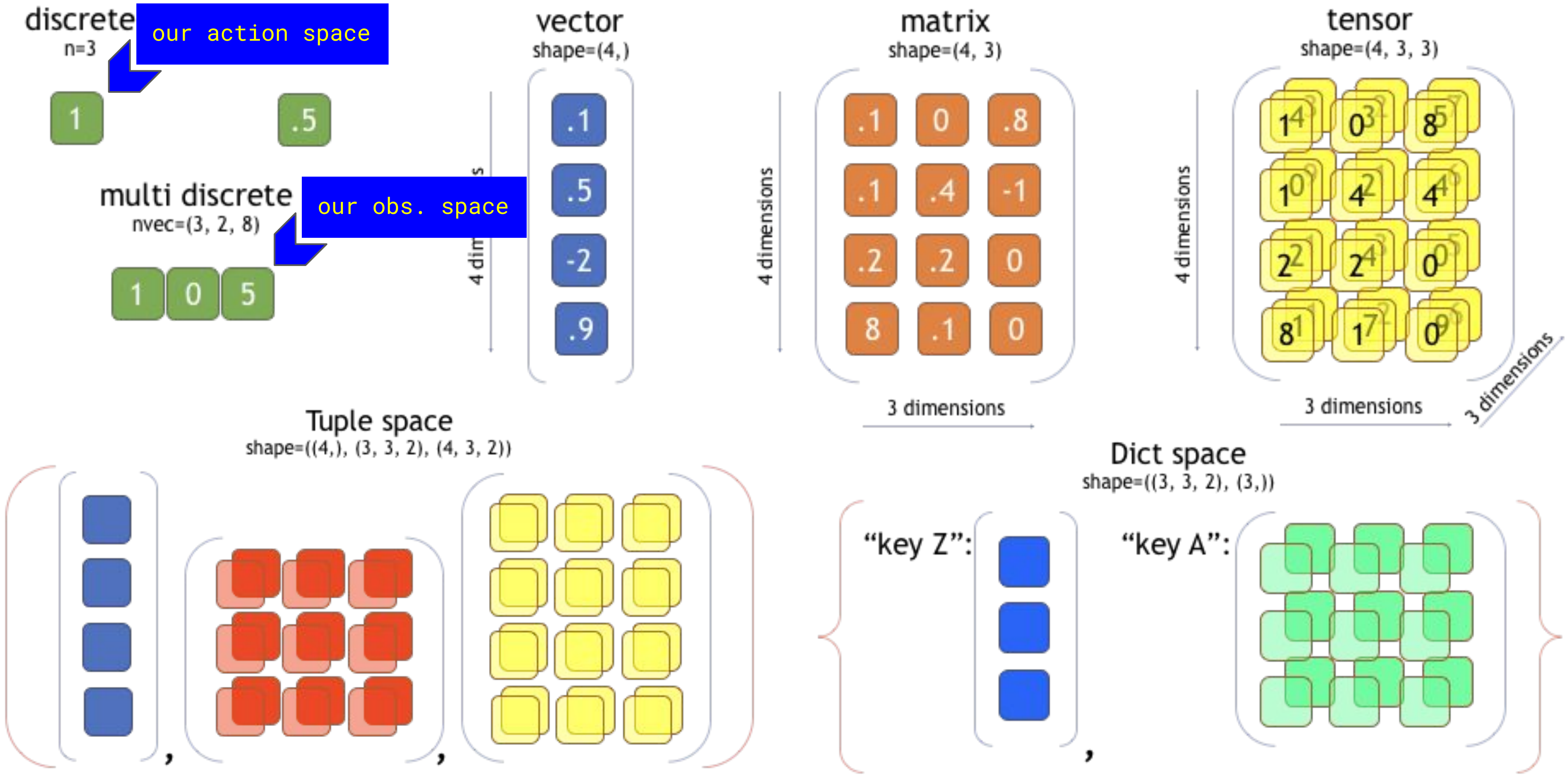
- RLlib is based on Ray, so it benefits from all its improvements on **performance** and **scalability**.

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

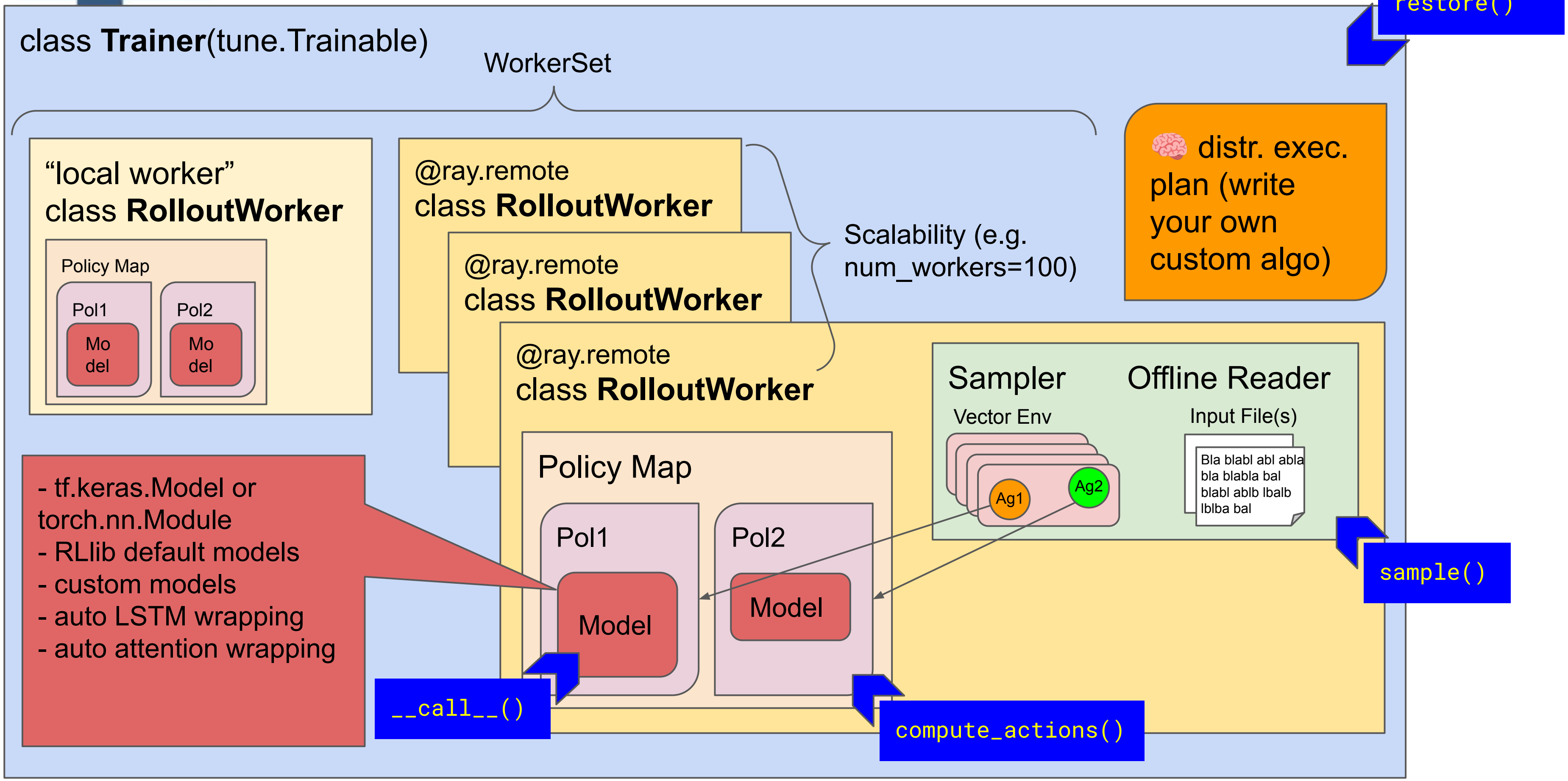
- RLlib is extremely **flexible**, allowing you to **customize every aspect** of the RL cycle and workflows.

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

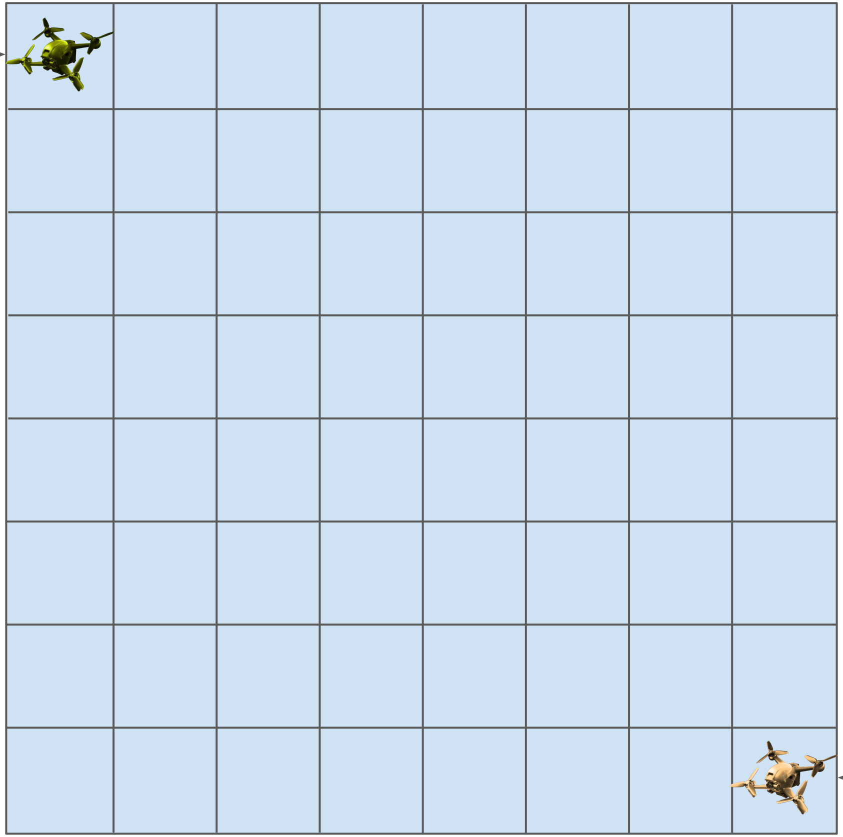
What's a Space?



A Closer Look at RLlib



Our Environment - The “MultiAgentArena” 😶



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

rewards:

agent1 (“cover as much ground as possible”):

- +1 if new field
- -1 if colliding with agent 2
- -0.5 otherwise

agent2 (“defend: bump into agent1 as often as possible”):

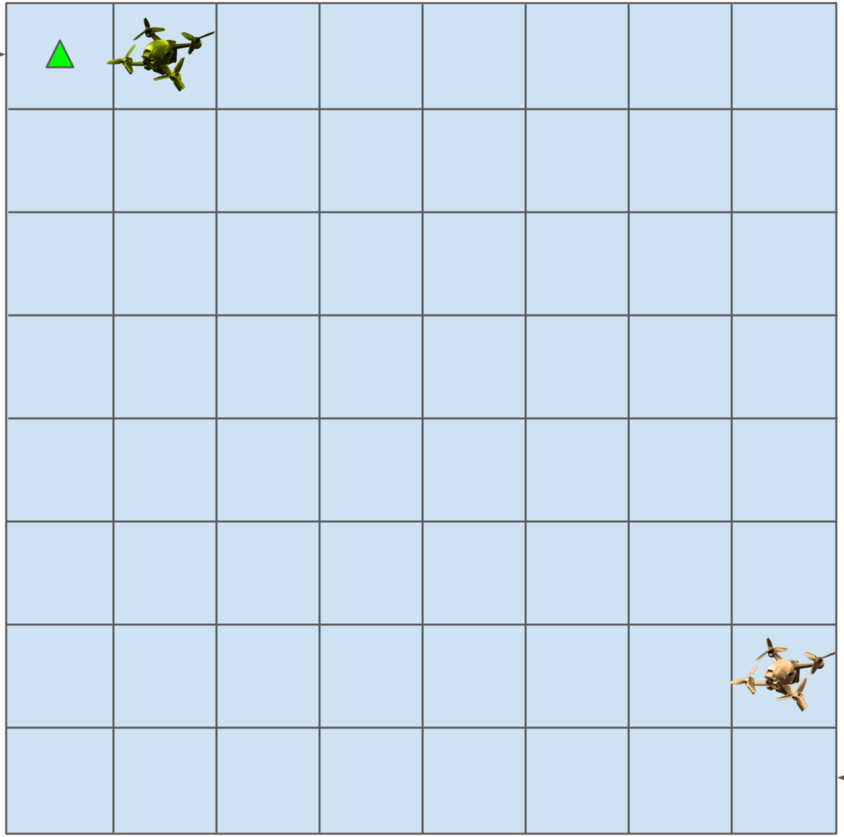
- +1 if colliding with agent 1
- -0.1 otherwise

observation space:
MultiDiscrete([64, 64])

pos. agent 1
pos. agent 2

Our Environment - The “MultiAgentArena” 😶

agent 1



agent 2

action space: Discrete(4)

0 - up

1 - right

2 - down

3 - left

rewards:

agent1 (“cover as much ground as possible”):

- +1 if new field
- -1 if colliding with agent 2
- -0.5 otherwise

agent2 (“defend: bump into agent1 as often as possible”):

- +1 if colliding with agent 1
- -0.1 otherwise

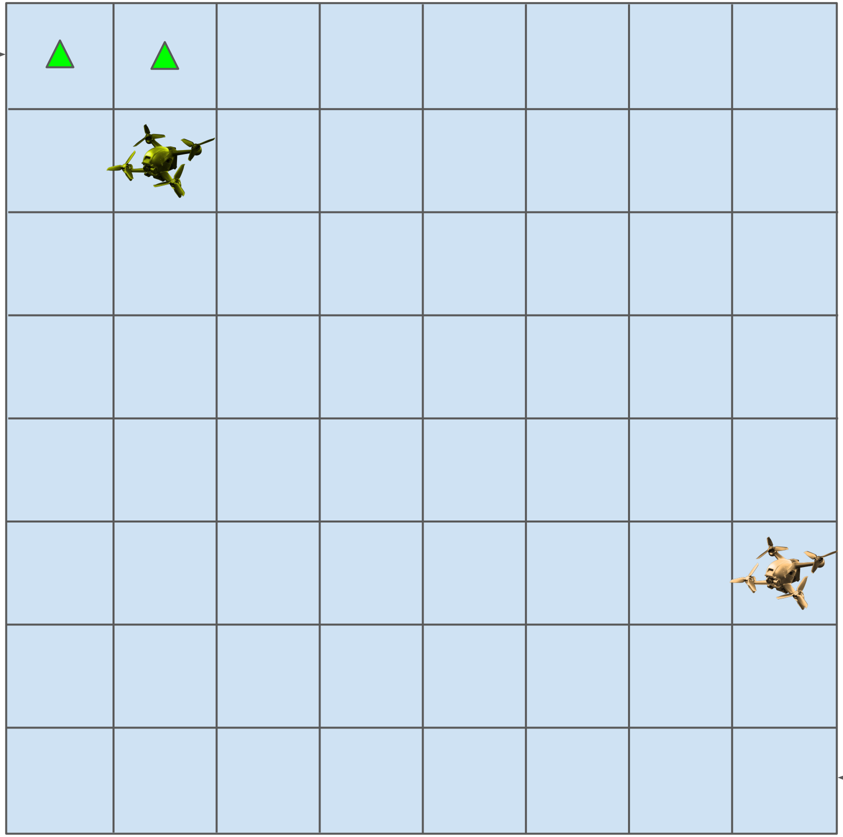
observation space:
MultiDiscrete([64, 64])

pos. agent 1

pos. agent 2

Our Environment - The “MultiAgentArena” 😶

agent 1



agent 2

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

rewards:

agent1 (“cover as much ground as possible”):

- +1 if new field
- -1 if colliding with agent 2
- -0.5 otherwise

agent2 (“defend: bump into agent1 as often as possible”):

- +1 if colliding with agent 1
- -0.1 otherwise

observation space:
MultiDiscrete([64, 64])

pos. agent 1

pos. agent 2

Our Environment - The “MultiAgentArena” 😶

agent 1

action space: Discrete(4)

0 - up
1 - right
2 - down
3 - left

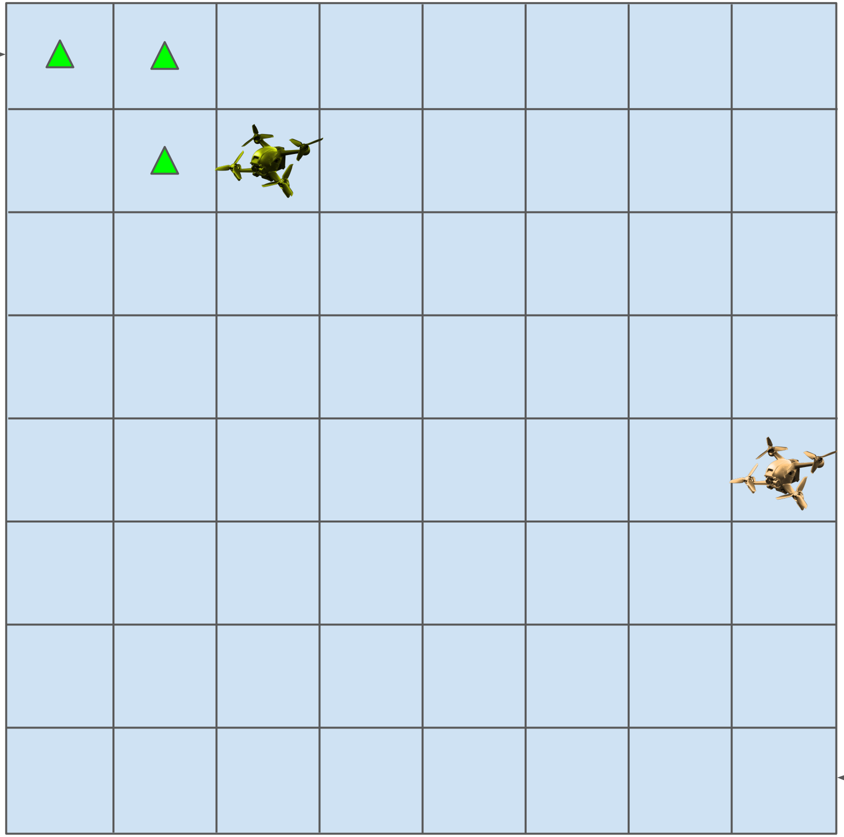
rewards:

agent1 (“cover as much ground as possible”):

- +1 if new field
- -1 if colliding with agent 2
- -0.5 otherwise

agent2 (“defend: bump into agent1 as often as possible”):

- +1 if colliding with agent 1
- -0.1 otherwise



agent 2

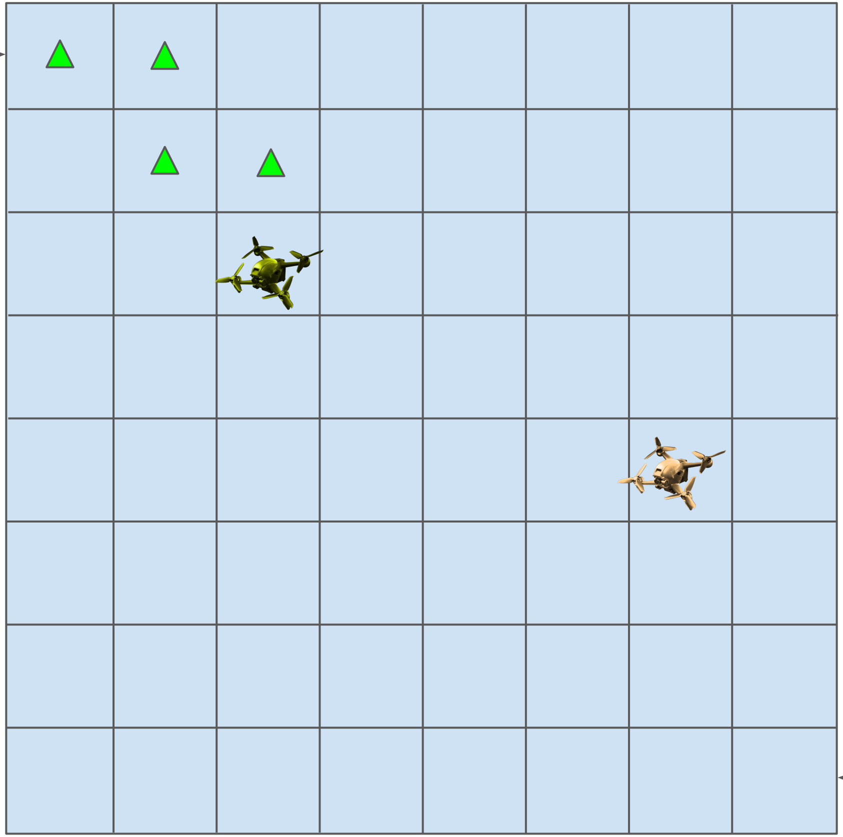
observation space:
MultiDiscrete([64, 64])

pos. agent 1

pos. agent 2

Our Environment - The “MultiAgentArena” 😶

agent 1



agent 2

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

observation space:
MultiDiscrete([64, 64])

pos. agent 1
pos. agent 2

rewards:

agent1 (“cover as much ground as possible”):

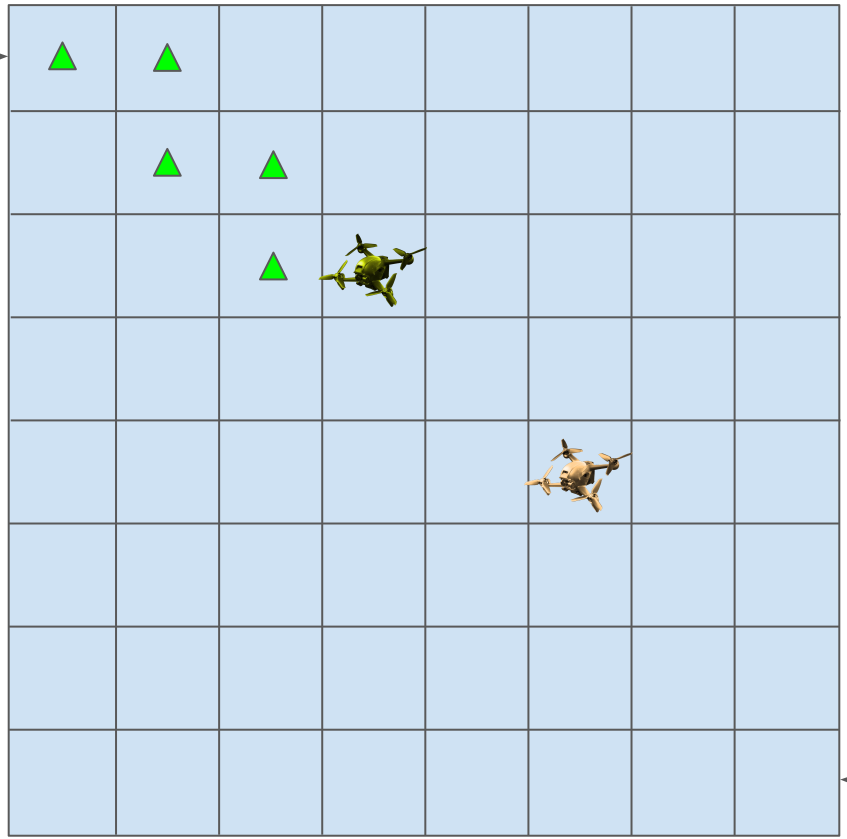
- +1 if new field
- -1 if colliding with agent 2
- -0.5 otherwise

agent2 (“defend: bump into agent1 as often as possible”):

- +1 if colliding with agent 1
- -0.1 otherwise

Our Environment - The “MultiAgentArena” 😶

agent 1



agent 2

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

rewards:

agent1 (“cover as much ground as possible”):

- +1 if new field
- -1 if colliding with agent 2
- -0.5 otherwise

agent2 (“defend: bump into agent1 as often as possible”):

- +1 if colliding with agent 1
- -0.1 otherwise

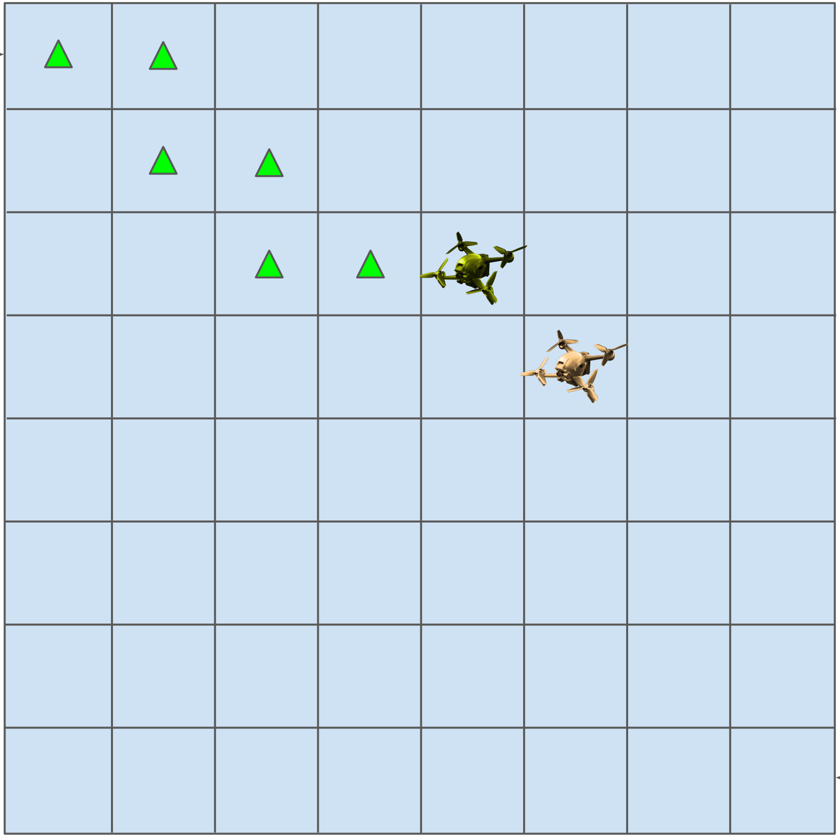
observation space:
MultiDiscrete([64, 64])

pos. agent 1

pos. agent 2

Our Environment - The “MultiAgentArena” 😶

agent 1



agent 2

action space: Discrete(4)

0 - up
1 - right
2 - down
3 - left

rewards:

agent1 (“cover as much ground as possible”):

- +1 if new field
- -1 if colliding with agent 2
- -0.5 otherwise

agent2 (“defend: bump into agent1 as often as possible”):

- +1 if colliding with agent 1
- -0.1 otherwise

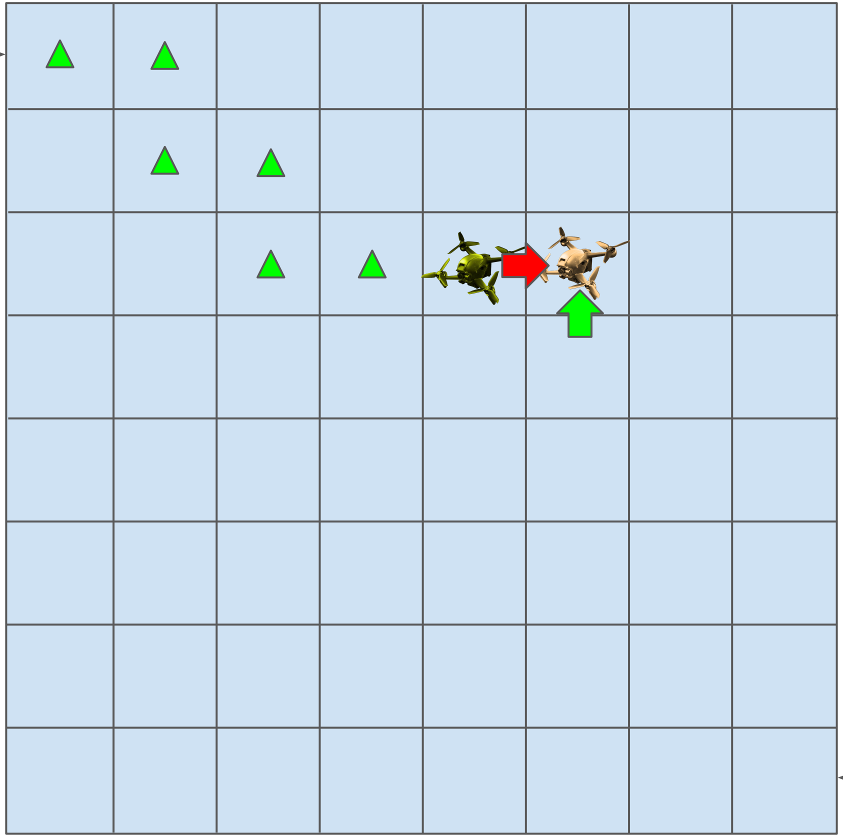
observation space:
MultiDiscrete([64, 64])

pos. agent 1

pos. agent 2

Our Environment - The “MultiAgentArena” 😶

agent 1



agent 2

action space: Discrete(4)

0 - up
1 - right
2 - down
3 - left

observation space:
MultiDiscrete([64, 64])

pos. agent 1

pos. agent 2

rewards:

agent1 (“cover as much ground as possible”):

- +1 if new field
- -1 if colliding with agent 2
- -0.5 otherwise

agent2 (“defend: bump into agent1 as often as possible”):

- +1 if colliding with agent 1
- -0.1 otherwise