Environment "Find the Goals" Documentation

Author: Shuo Jiang (jiang.shuo@husky.neu.edu)

Introduction

The document introduces a environment for multi agent study as two agents are trying to find their goals in a maze. The code is implemented in python and provided with simple interfaces. The environment is decoupled with learning method so reader can try any algorithms on.

Scene Description

The task of the environment is explained as below

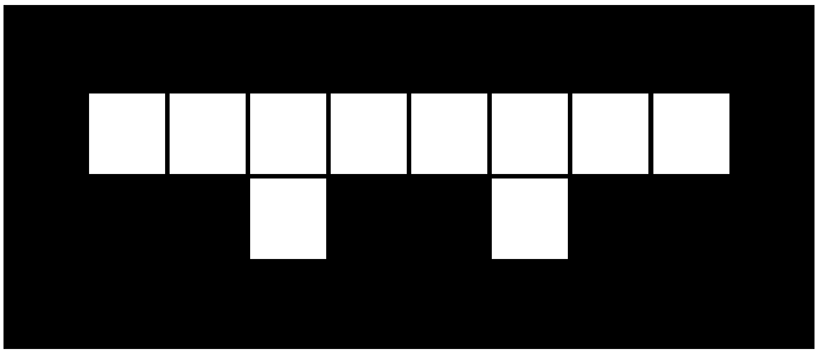


Figure 1

The environment works like a maze, two agents(agent 1 and agent2) will run in the maze and find their targets. The two agents, agent 1 and agent2 will be denoted using color red and blue, and will be shown as red and blue rectangles in the maze like:

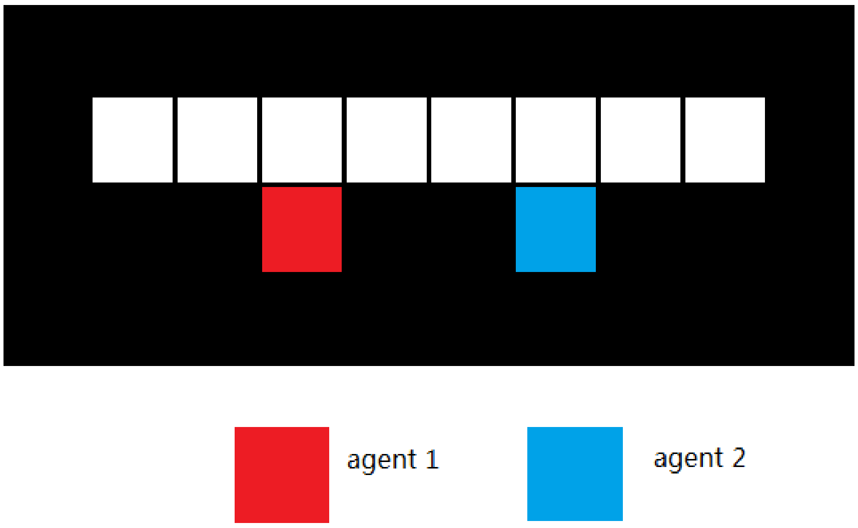


Figure 2

blocks in black are obstacles that agents could not step on and white ones are free spaces. The coordinate system is show as

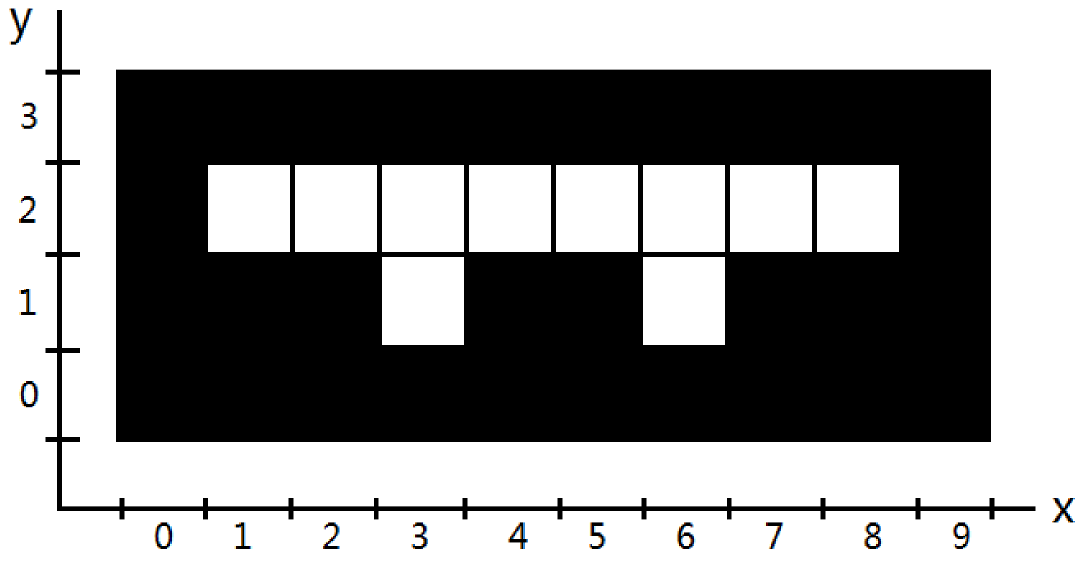


Figure 3

For agent 1, it starts at block (x, y)=(3, 1), agent2 starts at (6, 1) as shown in figure 1. The target of agent 1 is (8, 2) and target of agent 2 is(1, 2). Each time the agent reaches its goal will be immediately transported back to its start position, but the other agent will not be affected, as:

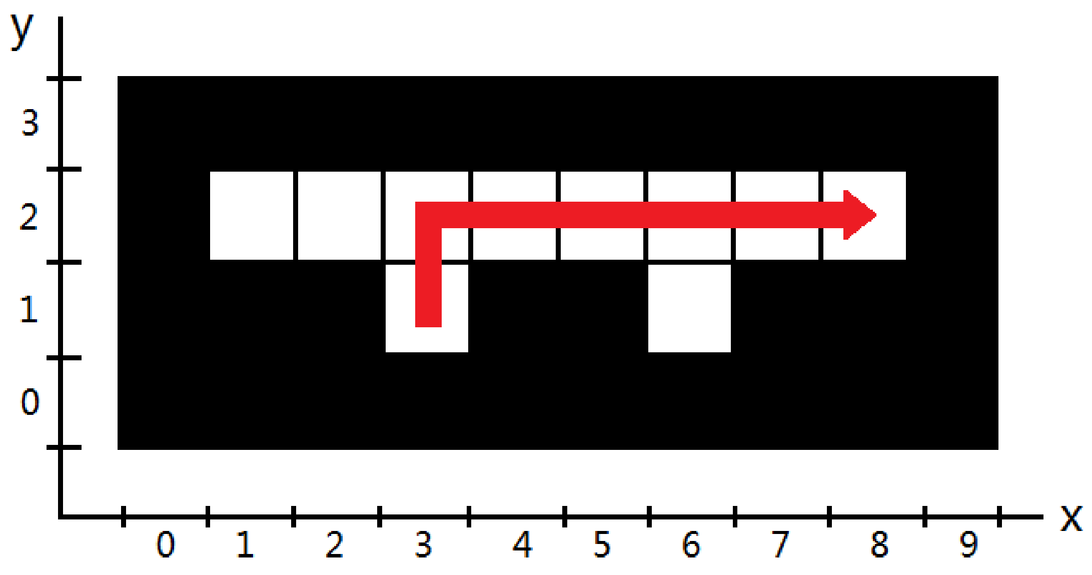


Figure 4

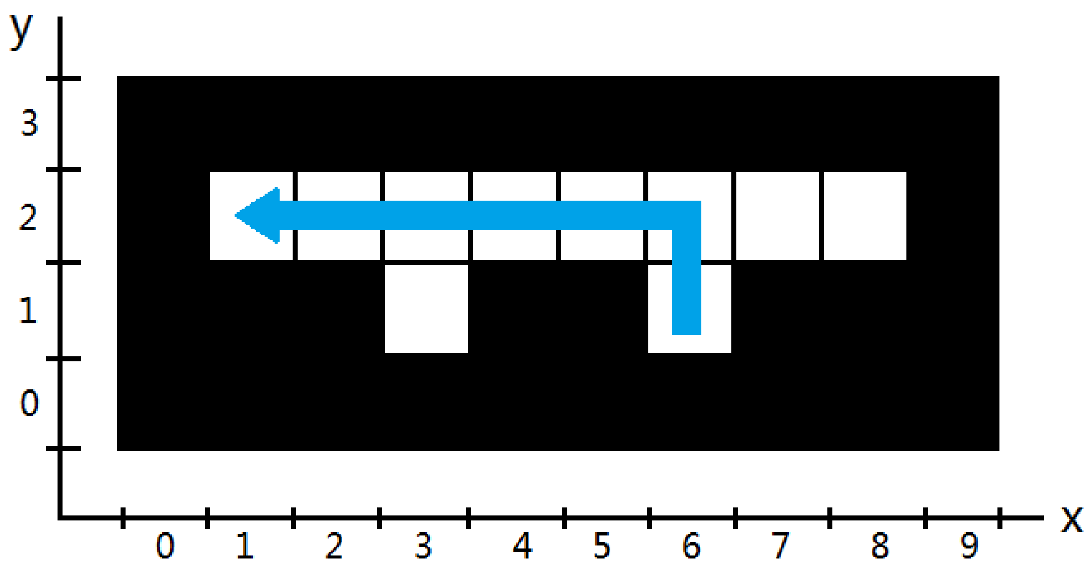


Figure 5

There are 5 possible action for the two agents

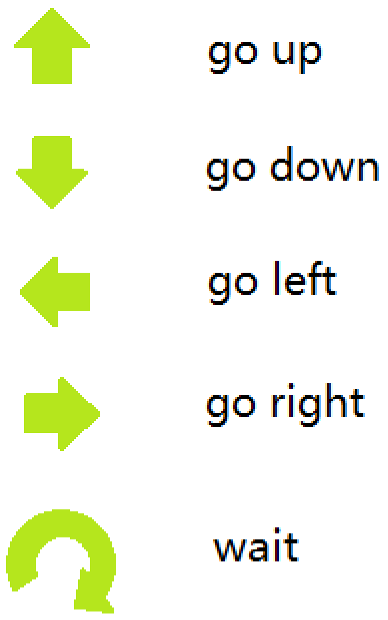


Figure 6

when the agent moves, it will move to the free space of 1 distance near it. However, when there is a block or the other agent is in its way, the action won't work. One can understand the this by an example as

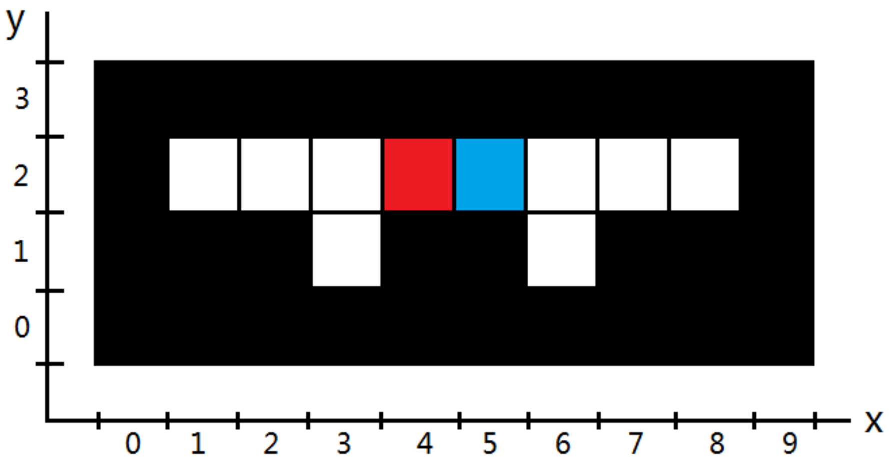


Figure 7

When agent 1(red) is at (4,2) and agent 2 is at (5,2). As agent 1 wants to move right and agent2 wants to move left, the result will be no one moving. This interference increases the challenge of the work as the maze is narrow, only when two agents are programmed in a very synchronized and coordinated way, the interference can be avoided.

The reward for each action taken is the same for both agents. Action "go up", "go down", "go left" and "go right" will return a -1 reward and agent reaches its corresponding target will be rewarded for 100.

Agent 1 and 2 will only observe their local surroundings as 9 nearest positions as an image.

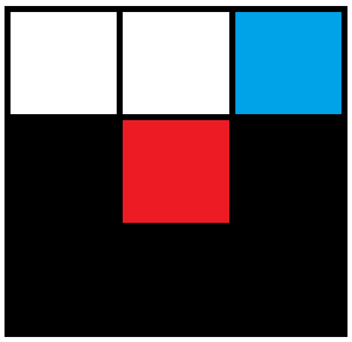


Figure 8

Or using a more global view as

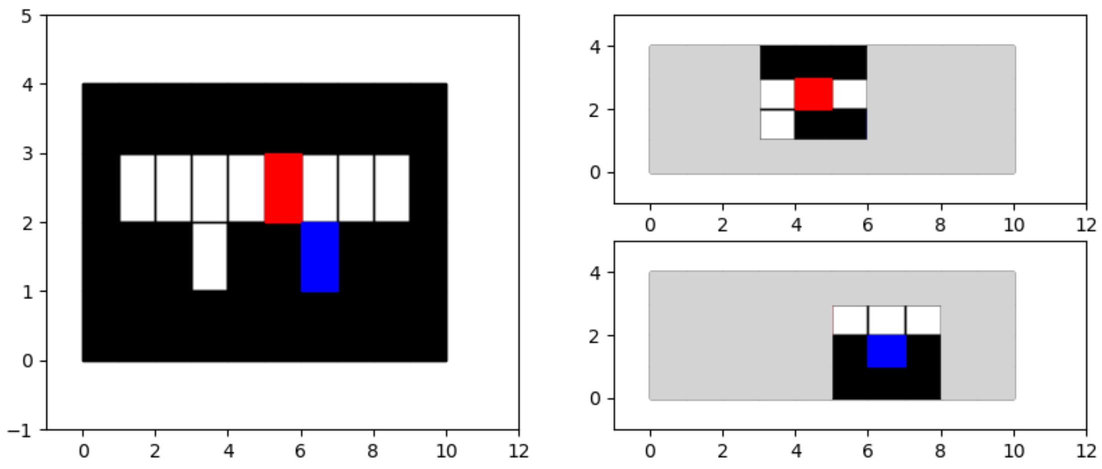


Figure 9

The global positions of agent1 (red) and 2 (blue) is shown as figure 9 left. The vision of agent 1 is shown in figure 9 up right, the light gray area is invisible to agent1. Also the figure 9 down right is the vision of agents 2.

Class Organization

The environment is programmed in python 3.6 and has very simple interfaces. The file contains the environment is "env_FindGoals.py" and a class " EnvFindGoals" is defined.

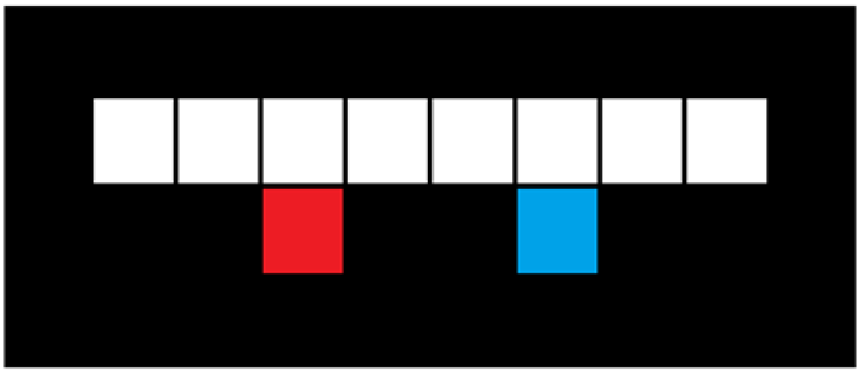
The class has the following interfaces:

```
__init__(self):
get_agt1_obs(self):
get_agt2_obs(self):
step(self, action1, action2):
```

reset(self):
plot_scene(self)

__init__(self):
initialize the object, important variables are
`agent1_pos`
`agent2_pos`
These are programmed by a list as agent_pos=[3, 1]

`get_full_obs(self)`
These function return the current vision of agent 1 by returning a array of size (4,10, rbg channel).
as.



get_agt1_obs(self):
These function return the current vision of agent 1 by returning a array of size (3,3, rbg channel) or (3,3,3) as a image. agent 1 will be centered as at position 4.

0	1	2
3	4	5
6	7	8

Figure 10

one example is the view of agent 1 is shown as

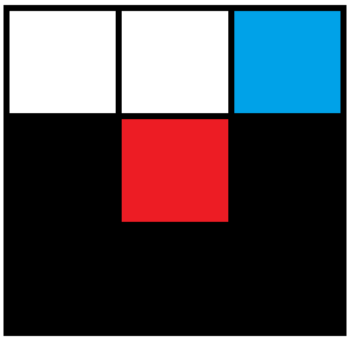
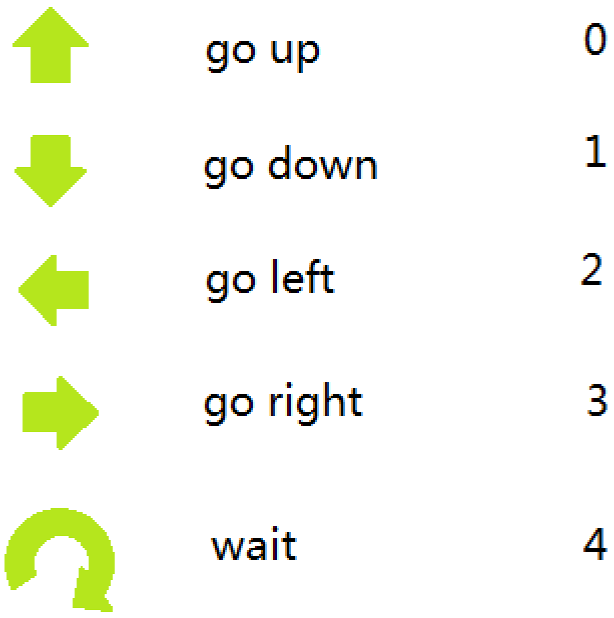


Figure 11

step(self, action1, action2):
The function update the environment by feeding the actions for agent1 and agent2. The actions are expressed by integers as



⬆	go up	0
⬇	go down	1
⬅	go left	2
➡	go right	3
↻	wait	4

Figure 12

So the size of valid action is 5. The returned values of the function are reward, obs_1 and obs_2, which are the joint reward (sum of reward of agent 1 and agent 2) in the given step, the observation vector of agent 1 and agent 2.

reset(self):
This function move the two agent back to their start points as (3, 1) and (6, 1)

plot_scene(self)

This function plot the current state of the whole environment, and uses three subplots to show the views of each agent

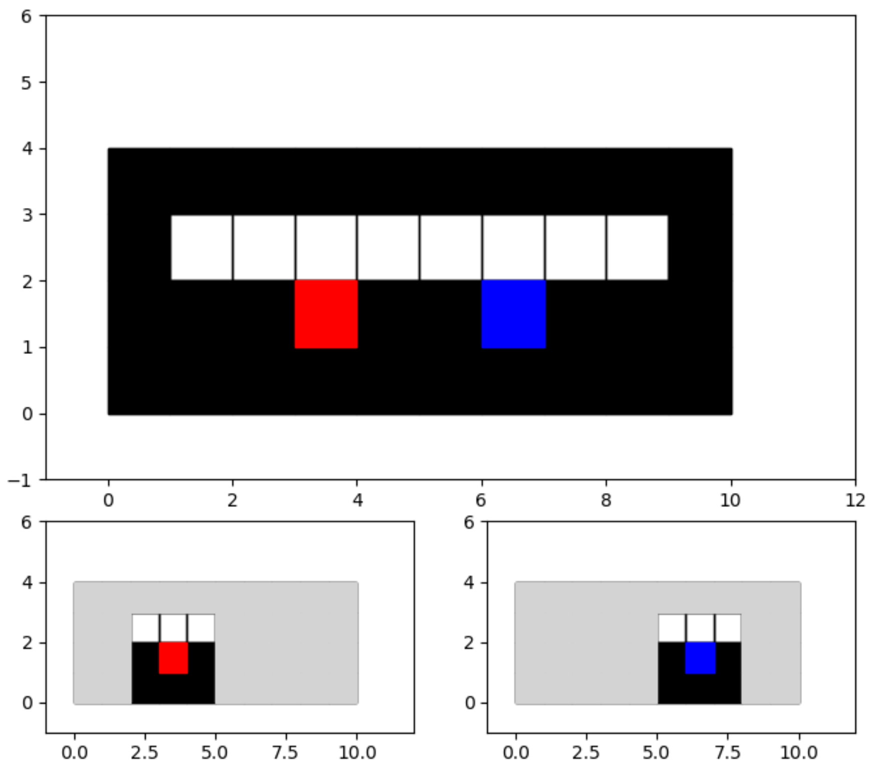


Figure 13

The global positions of agent1 (red) and 2 (blue) is shown as figure 13 top. The vision of agent 1 is shown in figure 13 bottom left, the light gray area is invisible to agent1. Also the figure 13 bottom right is the vision of agents 2.

Example

Here is an example using test function, and it is in "test_FindGoals.py"

```
from env_FindGoals import EnvFindGoals
import random

env = EnvFindGoals()
max_iter = 100
for i in range(max_iter):
    print("iter= ", i)
    a_1 = random.randint(0, 4)
    a_2 = random.randint(0, 4)
    reward_1, reward_2, obs_1, obs_2 = env.step(a_1, a_2)
    env.plot_scene()
```

```python
if reward_1 > 0:
    print("agent 1 finds goal")
if reward_2 > 0:
    print("agent 2 finds goal")
```