

# Environment "Find the Treasure" Documentation

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

License: Northeastern University, Lab for Learning and Planning in Robotics(LLPR)

## Introduction

The document introduces an environment for multi agent study as two agents are trying to find the treasure in a room. 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

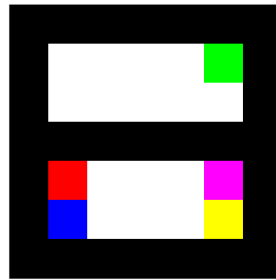


Figure 1

The environment consists of two rooms, two agents (agent 1 and agent2) will run in one room and the treasure is in the other room. The two agents, agent 1 and agent2 will be denoted using color **red** and **blue**, and the treasure will be marked with **green**.

Blocks in black are obstacles that agents could not step on and white ones are free spaces. We can see that the agents and the treasure are in separated rooms. There is a secret door on the wall in the middle. In the larger room, there will be a lever, marked by **yellow**. If any of the agent is standing on the lever position, the secret door will open and the other agent gets the access to the smaller room to get treasure. If any agent stands on the treasure position, the system resets, which means the two agents are relocated in the bottom room, as shown in Figure 1.

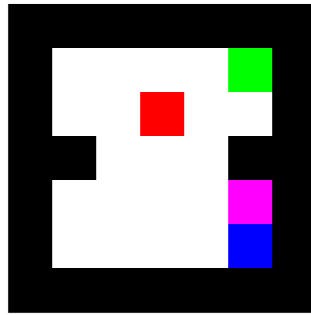


Figure 2. When an agent stands on the lever, the secret door opens

There is a sub optimal solution that when agents stand both on the pink block and yellow block, the episode finishes, but the agents get a small reward.

## Coordinate System

The coordinate system is show as

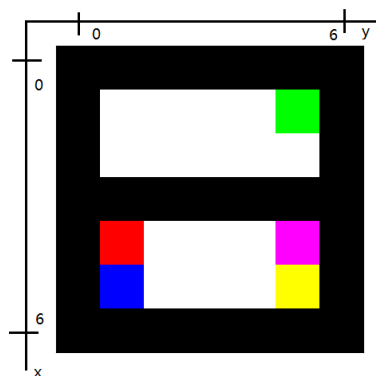


Figure. 3 Coordinate System

## Reward

Each time any agent finds the treasure will be rewarded 100 for both agents. Bump to the wall will be rewarded -0.1. The sub optimum rewards 3 for both agents.

## Action Space

There are 5 possible action for the two agents


	go up	0
	go down	1
	go left	2
	go right	3
	wait	4

Figure. 4 Action space

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. The five actions are indexed by an integer 0 - 4.

## Observation

There are two ways to get the observation depending on if it is fully observable or not. When it is fully observable, the agents get a joint observation as image as Figure.1. When it is partially observable, each agent receive a local self-centered observation as bottom figures in Figure 5

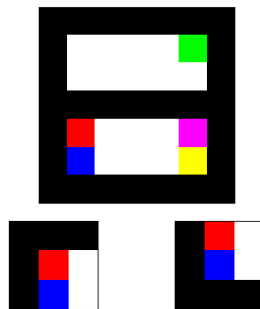


Figure. 5

## Class Organization

The environment is programmed in python 3.6 and has very simple interfaces. The file contains the environment is "env\_FindTreasure.py" and a class " EnvFindTreasure" is defined.

The class has the following interfaces:

```
__init__(map_size)
obs_list = get_obs()
obs = get_global_obs()
state = get_state()
reward_list, done = step(action_list)
reset()
```

```
__init__(map_size)
initialize the object, important variables are
map_size
agent1_pos
agent2_pos
```

`get_obs()`  
This function returns a list contains current views of agent and agent2.  
one example is the view of agent 1 is shown as  
Each returned observation is a numpy array with size (3, 3, 3) as an image. `obs_list = [obs1, obs2]`

`get_global_obs()`  
return global view with no fog, a numpy array with size (map\_size, map\_size, 3)

`get_state()`  
returns a numpy array of size [1, 4], which are the positions of two agents.

`step(action_list)`  
The function updates the environment by feeding the actions for agent1 and agent2. The size of valid action is 5. The returned values of the function are reward, done, which are the rewards of agent 1 and agent 2 in the given step, and whether the episode is done. Reward is a float number

`reset()`  
This function relocates the two agents in the bottom room

## Example

Here is an example using test function, and it is in "test\_FindTreasure.py". The actions for agents are random chosen, click and run.

```
from env_FindTreasure import EnvFindTreasure
import random
```

```
if __name__ == '__main__':  
    env = EnvFindTreasure(7)  
    max_iter = 1000  
    for i in range(max_iter):  
        print("iter= ", i)  
        env.render()  
        env.plot_scene()  
        action_list = [random.randint(0, 3), random.randint(0, 3)]  
        print()  
        reward, done = env.step(action_list)  
        if done:  
            print('find goal, reward', reward)  
            env.reset()
```