# **Berry Field – gym env**

## Requirements:

gym – version 0.0.1

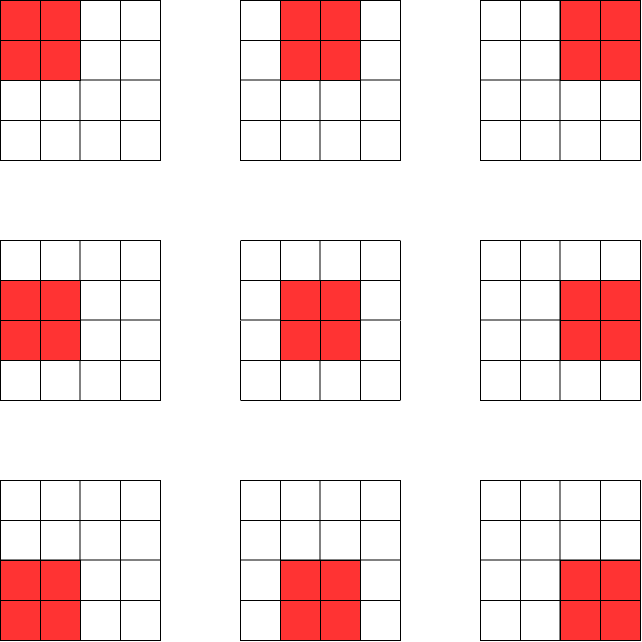
## Installation:

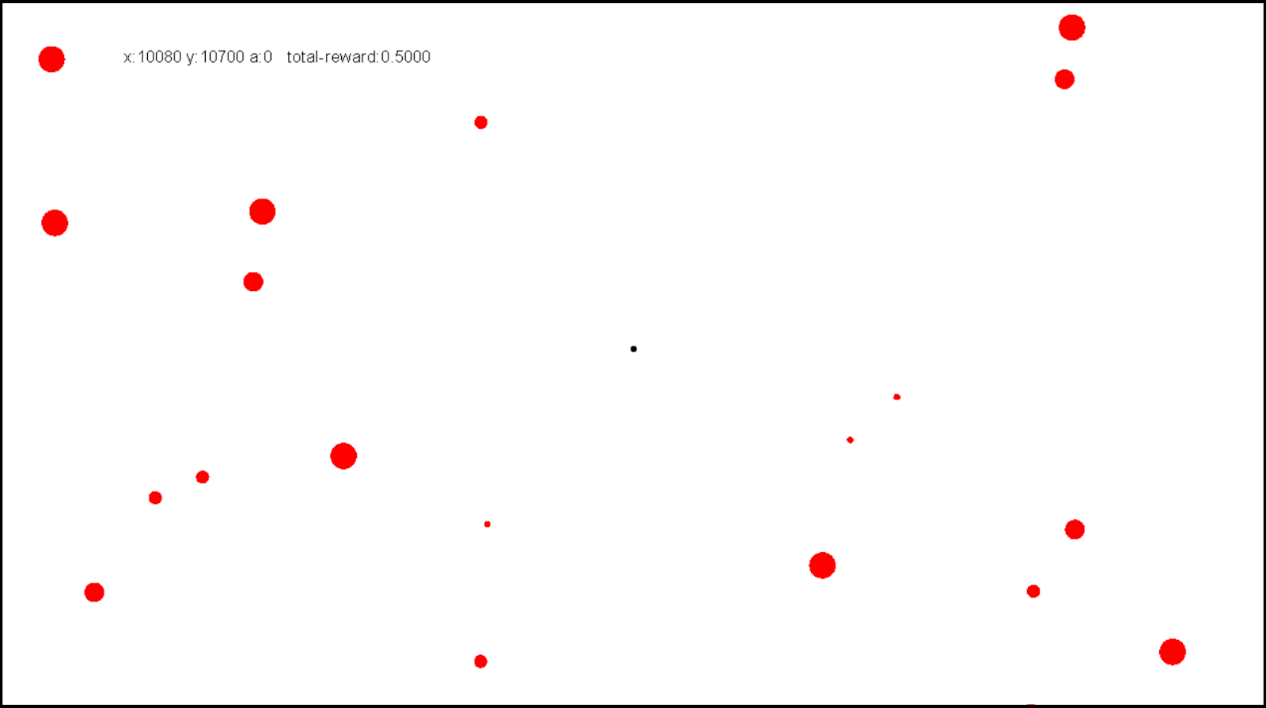
Located in the folder **env**. To install navigate into the **env** folder and use the command:  
 *pip install -e berry-field*

After installation, to use the environment please use the *get\_env(…)* function from  
**get\_env.py**

## Documentation:

The agent can move in 8 directions and also choose to not move at every step.



Whenever the agent collides with a berry, it collects the berry and gets a reward. The agent gets a negative reward for every movement through the environment. The agent can view only a limited section of the environment. It cannot move outside the boundary of the environment.

The environment supports both square and circular agent, square and circular berries.

The observations are of three types – unordered, ordered, and buckets. This is explained in the next section.

### Class Parameters and Definition:

BerryFieldEnv\_MatInput(gym.Env):

    def \_\_init\_\_(self,

                 file\_paths,

                 field\_size, agent\_size, observation\_space\_size,

                 drain\_rate, reward\_rate,

                 max\_steps,

                 initial\_state, circular\_berries=True,

circular\_agent=True,

                 observation\_type = "unordered",

                 bucket\_angle = 45,

                 reward\_curiosity = True, reward\_curiosity\_beta=0.25,

                 reward\_grid\_size = (100,100)

                 ):

Parameters:

1. file\_paths – the paths to the files to build the field of berries
2. field\_size – tuple of int, the size of the field.
3. agent\_size – int, the size of agent.
4. observation\_space\_size – the size of the space that is visible to the agent.
5. drain\_rate – the living cost of the agent, given only when the agent is moving
6. reward\_rate – the size of the berry is scaled by this to get the reward for picking a berry
7. max\_steps – the maximum number of steps in an episode
8. initial\_state – the initial location of the agent at the start of the episode
9. circular\_berries – True if berries are circles, False for square berries
10. circular\_agent – True if agent is circular, False for a square agent
11. observation\_type – the type of observation output the environment returns:
    1. unordered: (40,5) numpy array is returned. The columns are interpreted as “is-berry”, “unit-vector-x”, “unit-vector-y”, “distance”, “size”. If a row does not describe a berry, then it contains all zeros.
       1. Is-berry: 1 if the row describes a berry, 0 otherwise.
       2. Unit-vector-x: the x coordinate of the tip of the unit-vector towards the berry.
       3. Unit-vector-y: the y coordinate of the tip of the unit-vector towards the berry.
       4. Distance: Euclidean distance to the berries center.
       5. Size: the size of the berry.
    2. ordered: unordered observation but the entries are sorted in clockwise order.
    3. buckets:
       1. divides the observation space into num\_buckets number of equal-angular non-overlapping segments.
       2. Outputs a numpy array of shape (numbuckets,2) with the columns as “average-size-of-berry”, “average-distance-to-berry”
       3. Each row in the array represents the corresponding segment in clockwise order.
       4. If there is no berry in a angular segment, the corresponding row contains all zeros.
12. bucket\_angle – the angular width of each bucket for “buckets” observation type
13. reward\_curiosity – True if agent be given curiosity reward for exploration
14. reward\_curiosity\_beta – the beta parameter in the below equation
15. reward\_grid\_size – the size of each grid where the agent is given the curiosity reward. The size must divide the environment’s size.

**Example:**

file\_paths =['data/berry\_coordinates.csv','data/patch\_coordinates.csv']

num\_berries = 800

num\_patches = 10

field\_size = (20000, 20000) # (width, height)

patch\_size = (2600, 2600) # (width, height)

agent\_size = 10

observation\_space\_size = (1920, 1080) # (width, height)

observation\_type = "ordered"

drain\_rate = 1/(2\*120\*400)

reward\_rate = 1e-4

speed = 400

time = 300

max\_steps = speed\*time

initial\_state = (10960, 11270) # (width, height)

observation\_type = "ordered"

bucket\_angle = 45

reward\_curiosity = True

reward\_curiosity\_beta=0.25

reward\_grid\_size = (100,100) # (width, height)

### Creating the environment using get\_env.py:

**get\_env.py** has a function get\_env(…) as defined below:

def get\_env(observation\_type = "ordered",

bucket\_angle = 45, reward\_curiosity = True,

            reward\_curiosity\_beta=0.25, reward\_grid\_size = (100,100)):

    env = gym.make('berry\_field:berry\_field\_mat\_input-v0',

                   file\_paths=file\_paths,

                   field\_size=field\_size, agent\_size=agent\_size,

                   observation\_space\_size=observation\_space\_size,

                   drain\_rate=drain\_rate, reward\_rate=reward\_rate,

                   max\_steps=max\_steps,

                   initial\_state=initial\_state,

                   observation\_type = observation\_type,

                   reward\_curiosity = reward\_curiosity,

                   reward\_curiosity\_beta=reward\_curiosity\_beta,

                   reward\_grid\_size = reward\_grid\_size, # should divide respective dimention of field\_size

                   bucket\_angle = bucket\_angle

                   )

    return env

The parameters have already been discussed above. You may use the arguments in this function as well as those in the file **constants.py** to setup your own environment.

Import the file **get\_env.py** in your workspace and call the function **get\_env(…)** with the required parameters. This will return a gym environment.