SINGLE AGENT

Environment:

1. In environment, we first created class ‘drone’. We defined ‘init’ function and ‘reset’ function in it that will initialize and restart the game when game is over.

```
class Drone(gym.Env):
    def __init__(self):
        super(Drone, self).__init__()
        self.action_space = spaces.Discrete(4)
        self.observation_space = spaces.MultiDiscrete([610]*13)

    def reset(self,seed=None):

        self.weeds={1:[300,230],2:[450,230],3:[310,210],4:[320,230],5:[300,420]}
        self.observation=np.array([300,300,np.random.randint(100, 500),np.random.rand
        self.charge = 100
        self.run=1
        self.rewards=0
        self.charger=[25,300]
        pygame.init()
        pygame.display.set_caption("CATCH THE WEEDS")
        icon = pygame.image.load("drone.png")
        pygame.display.set_icon(icon)
        self.display = pygame.display.set_mode((600,600))
        self.clock = pygame.time.Clock()
```

Here self.observation stores the positions of the drone and the weeds.

2. In step function, firstly we described the movement of the drone so that it does not cross borders and also continuous to move left, right, up or down.
We also described how to reduce battery while moving and to terminate game if battery reaches **0**.
Battery will also decrease due to movement of drone at the rate of **.0055**..
We gave rewards if drone position collides with weed position and drone battery will be reduced by **5**.
To encourage the drones to go to charging point when their battery is low, we are giving reward to it if it moves in direction of charging point whenever its battery power is low.

```
def step(self,action):
    terminated=False

    a= self.observation[0]
    b= self.observation[1]

    if action==0 and a > 2 and self.run==1:
        a=a-1
    if action ==1 and a < 598 and self.run==1:
        a=a+1
    if action==2 and b > 2 and self.run==1:
        b=b-1
    if action ==1 and b < 598 and self.run==1:
        b=b+1
    if self.run==1:
     self.charge -= 0.055

    if self.charge<0 :
        terminated=True
        self.rewards -= 100
```

```
    def isCollision(a, b, bul_x, bul_y):
       distance = math.sqrt(math.pow(a - bul_x, 2) + (math.pow(b - bul_y, 2)))
       if distance < 3:
          return True
       else:
          return False
    def isCollisionw(a, b, bul_x, bul_y):
       distance = math.sqrt(math.pow(a - bul_x, 2) + (math.pow(b - bul_y, 2)))
       if distance < 25:
          return True
       else:
          return False
    for weed in self.weeds:
        p=self.weeds[weed][0]
        q=self.weeds[weed][1]

        if isCollision(a,b,p,q):
            self.rewards += 10
            self.charge -= 5
            self.weeds[weed][0]=np.random.randint(100,500)
            self.weeds[weed][1]=np.random.randint(100,500)

    if isCollisionw(a,b,self.charger[0],self.charger[1]) and self.run==1:
        self.run=0
        self.charge += 1
        if self.charge<20:
            self.rewards+=60
```

```
    if isCollisionw(a,b,self.charger[0],self.charger[1]) and self.run==1:
        self.run=0
        self.charge += 1
        if self.charge<20:
            self.rewards+=60
        if self.charge==100:
            self.run==1
    reward=self.rewards
    truncated=False
    self.render()
    self.observation=[a,b,self.weeds[1][0],self.weeds[1][1],self.weeds[2][0],self

    return self.observation, reward, terminated, truncated, {}
```

3. In last we defined render function that will display drone, weed, charging point and drones movement on screen.

```
def render(self):
    a= self.observation[0]
    b= self.observation[1]


    pygame.display.set_caption("Drone game")
    icon = pygame.image.load("drone.png")
    pygame.display.set_icon(icon)

    screen = pygame.display.set_mode((600,600))
    screen=self.display
    screen.fill((0, 255, 0))


    def circle(a,b):
     circle_color = (34, 139, 34) # GREEN color in RGB
     circle_radius = 10
     pygame.draw.circle(screen, circle_color, (int(a),int(b)), circle_radius)

    def weeds(i):
     v=self.weeds[i][0]
     f=self.weeds[i][1]

     circle(v,f)

    weeds(1)
```

```
        weeds(1)
        weeds(2)
        weeds(3)
        weeds(4)
        weeds(5)

        charge_wall1 = pygame.Rect(0, 250, 50, 50)
        screen.blit(playerImg,(a,b))
        pygame.draw.rect(screen, (0,0,0), charge_wall1)

        pygame.display.update()
        self.clock.tick(10)


    def close(self):
        pygame.quit()
```

ALL CODES ARE IN DRONE FOLDER

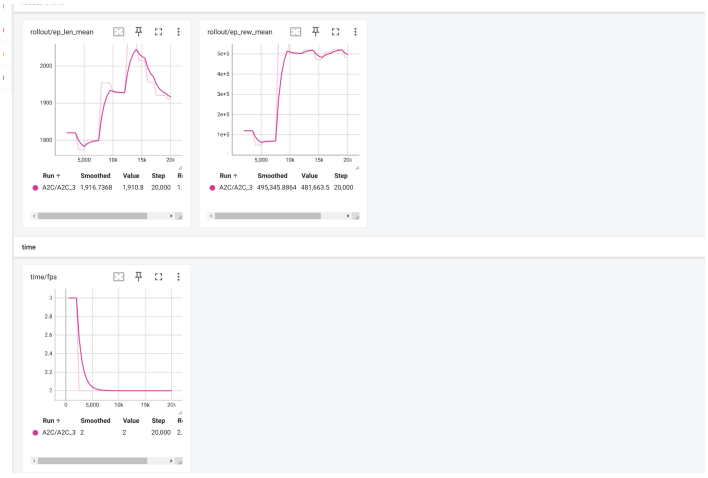
THERE ARE SEVERAL ENVIRONMENTS DEVELOPED FROM EXPERIENCE IN fin.py ,fin1.py ,fin2.py etc which we would like to describe in presentation

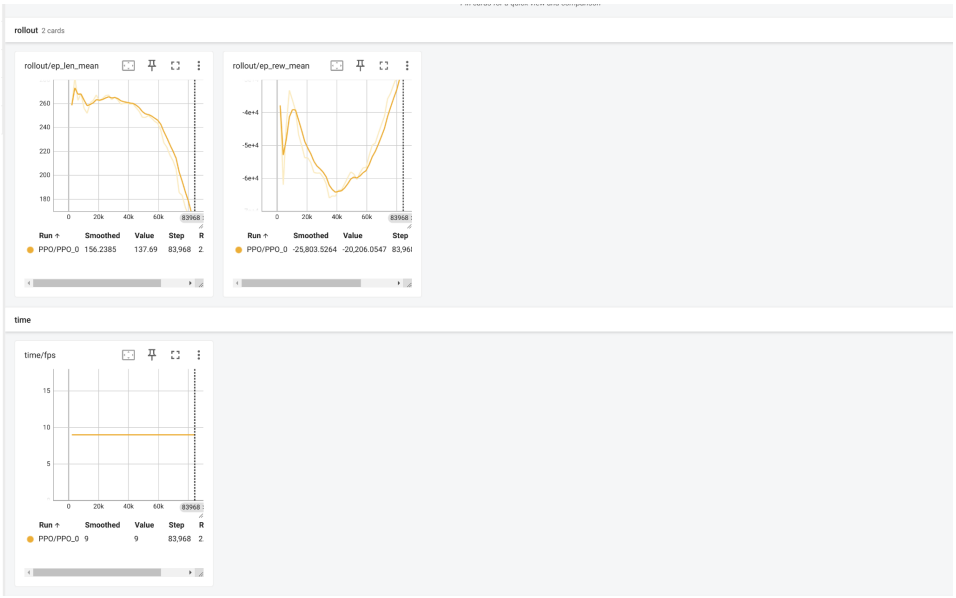
Logs and models are in the same folder

Demo game can be run from test.py and various model codes are in names of model.py

Training

With stablebaseline3 both PPO AND A2C ALGORITHMS





MULTI AGENT

All codes are in drone3 folder
Log files are in 1 folder inside

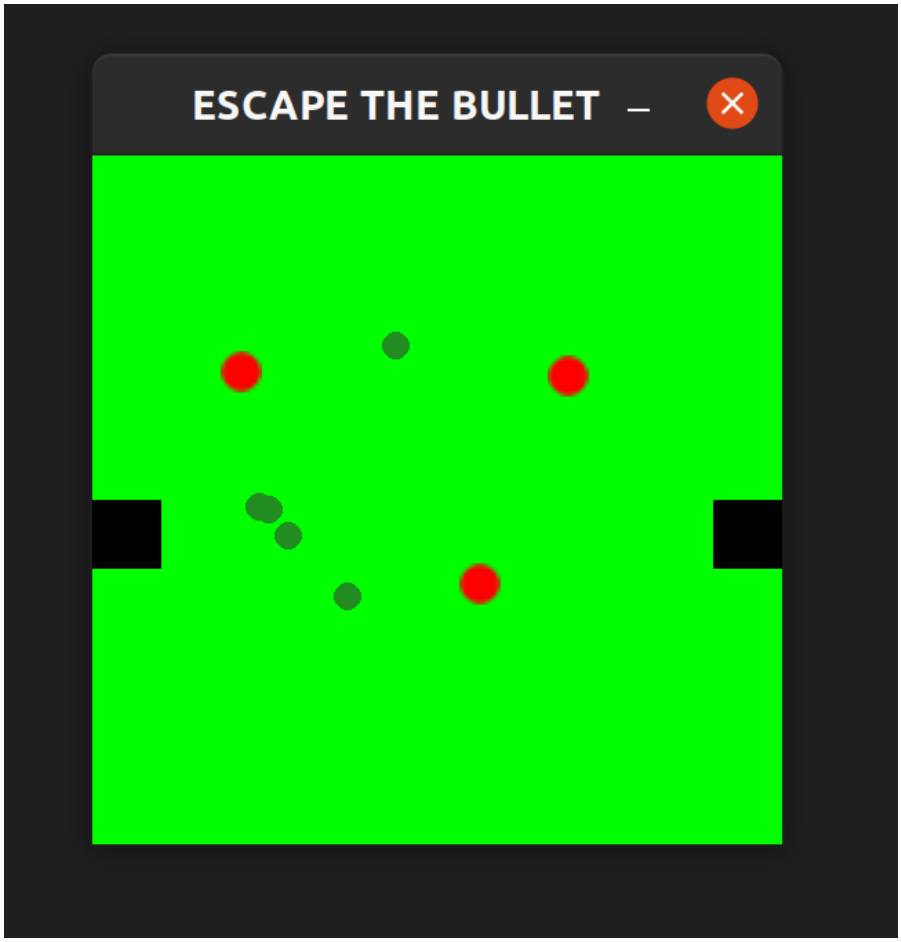
Game

Drone Simulation Game

This Python program uses the Pygame library to create a simple drone simulation game. The goal of the game is to control drones efficiently to kill more weeds without dying out of battery

Prerequisites

- Python installed (version 3.x recommended)
- Pygame library installed (`pip install pygame`)



Controls

- Left mouse button: Drag and move drones
- Charging Points: Drones can recharge their batteries at designated charging points.

Game Rules

- Each drone has a battery, and the game ends when a drone's battery is depleted.
- Drones can interact with objects represented by circles on the screen.
- Weeds decrease the drone's battery when in proximity.
- Drones can recharge their batteries at charging points.

Additional Notes

- The game includes 3 drones, each with its battery and charging/discharge rate.
- Weeds randomly spawn on the screen, affecting drone battery levels.

Acknowledgments

- This game was created using the Pygame library.

ENVIRONMENT

Custom Drone Environment

This project implements a custom environment for drone interactions using Pygame, OpenAI Gym, and PettingZoo.

The code is in drone3 folder with name drone.py

Environment Details

- Observation Space: MultiDiscrete with 16 dimensions (drone (x,y) , weeds(x,y))
- Action Space: Discrete with 4 possible actions for each drone.
- Rewards

  Local :

  +ve rewards are given when drone kills weeds or get gets closer to weeds when it has more charge greater than 40 percent to encourage exploration

  +ve rewards are given when drone goes to chase power grid when its battery is less than 30 percent to encourage conservation

  -ve rewards are given when drone dries out of battery

  Global:-

  -ve rewards are given when drone collides with another or tries to cross the grid

  +ve rewards are given when a weed is killed

  Game ends when any of the drone dries out or colloids

Training

Petting zoo and Ray libraries are use and PPO ALGORITHM IS TRAINED on both single policy and multiple policy modes

MODEL FILES FOR MULTIAGENT ARE TOO HEAVY WE MAY SHOW IN LAPTOP