Spike: Spike Week 6
Title: Navigation with Graphs

Author: Nguyen Khanh Toan - 104180605

Goals / deliverables:
Enhance the Task 5 navigation graph to demonstrate the following:
• Expanding the game world into a higher number of navigation tiles, with an expanded navigation graph structure.
• Implementing a path-planning system that can generate paths for agents considering the current dynamic environment, utilizing heuristic algorithms that consider at least six distinct 'terrain' types.
• Can utilize multiple moving agent characters (minimum of four), each capable of following their individual paths.
• Illustrating at least two distinct agent types with unique navigation behaviors within the world.

Technologies, Tools, and Resources used:

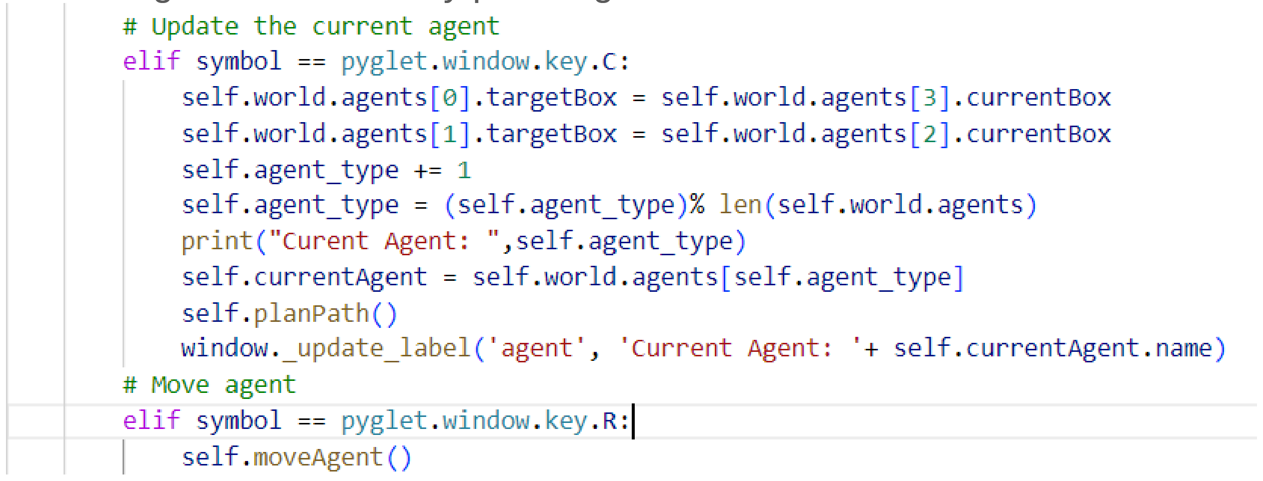
- Visual Studio Code
- Python 3.12

Tasks undertaken:

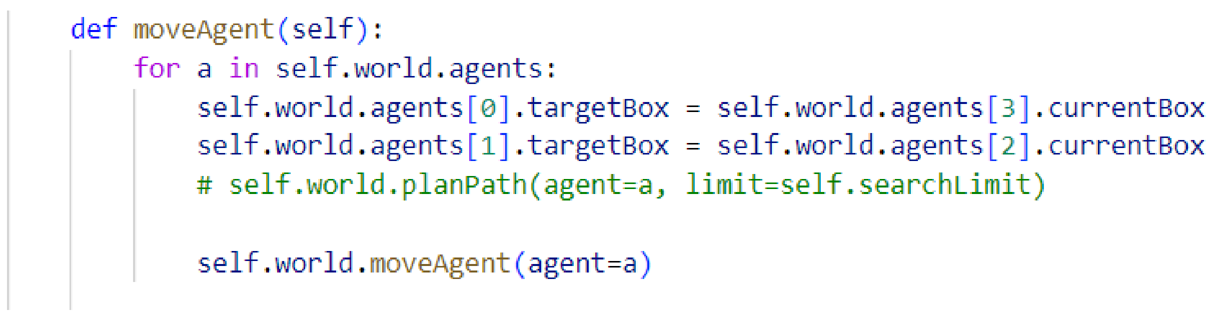
- Install Python 3+
- Install and setup compatible IDE for the language, e.g.: Visual Studio Code
- Pay attention to the comment of how the code work and functionality. Can use debug tool to observe the program more clearly.
- Run the code and observing the output.

What we found out:

- Program contains 4 type of agents, and agent can be change by pressing “C”
- All of the Agent can be run by pressing “R”

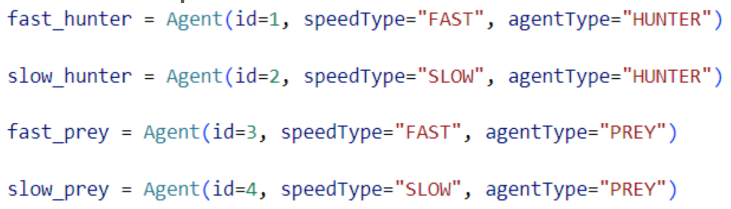


```
        # Update the current agent
        elif symbol == pyglet.window.key.C:
            self.world.agents[0].targetBox = self.world.agents[3].currentBox
            self.world.agents[1].targetBox = self.world.agents[2].currentBox
            self.agent_type += 1
            self.agent_type = (self.agent_type)% len(self.world.agents)
            print("Curent Agent: ",self.agent_type)
            self.currentAgent = self.world.agents[self.agent_type]
            self.planPath()
            window._update_label('agent', 'Current Agent: '+ self.currentAgent.name)
        # Move agent
        elif symbol == pyglet.window.key.R:
            self.moveAgent()
```

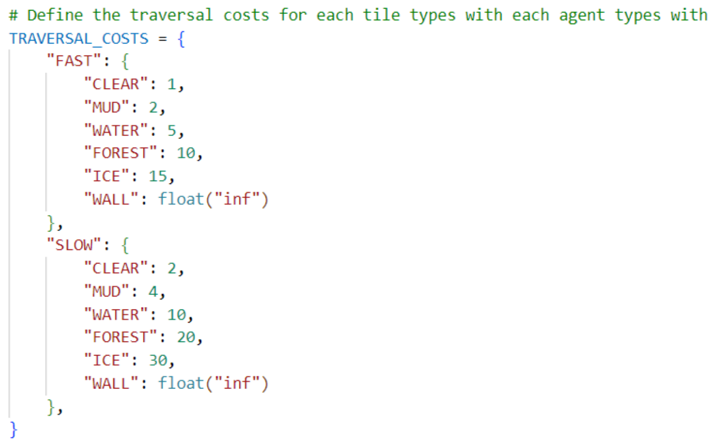


```
def moveAgent(self):
    for a in self.world.agents:
        self.world.agents[0].targetBox = self.world.agents[3].currentBox
        self.world.agents[1].targetBox = self.world.agents[2].currentBox
        # self.world.planPath(agent=a, limit=self.searchLimit)

        self.world.moveAgent(agent=a)
```

- The program can integrate with 4 types of agent, first two agents are Prey, with different speed variable, and the last two are prey, with different of speed.

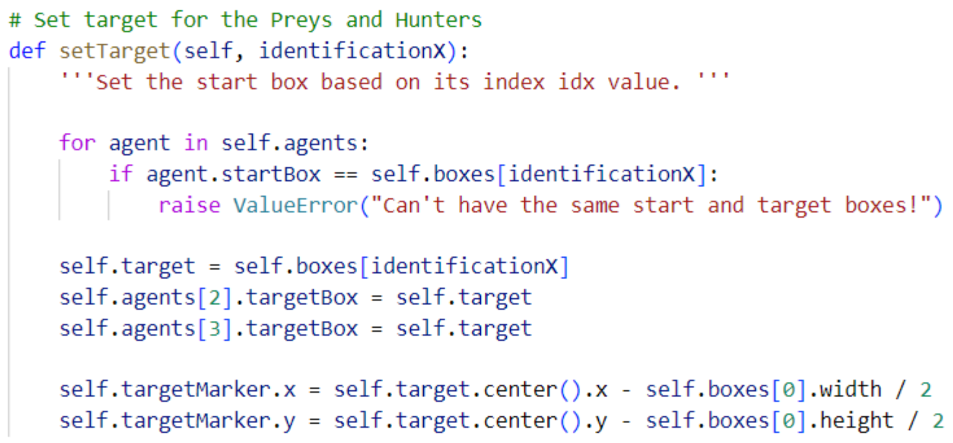


```
fast_hunter = Agent(id=1, speedType="FAST", agentType="HUNTER")

slow_hunter = Agent(id=2, speedType="SLOW", agentType="HUNTER")

fast_prey = Agent(id=3, speedType="FAST", agentType="PREY")

slow_prey = Agent(id=4, speedType="SLOW", agentType="PREY")
```



```
# Define the traversal costs for each tile types with each agent types with
TRAVERSAL_COSTS = {
    "FAST": {
        "CLEAR": 1,
        "MUD": 2,
        "WATER": 5,
        "FOREST": 10,
        "ICE": 15,
        "WALL": float("inf")
    },
    "SLOW": {
        "CLEAR": 2,
        "MUD": 4,
        "WATER": 10,
        "FOREST": 20,
        "ICE": 30,
        "WALL": float("inf")
    },
}
```

- The target of prey will located in the random location, or pressing “8” cho change location for Prey. Prey will automatically assign target of preys’ start

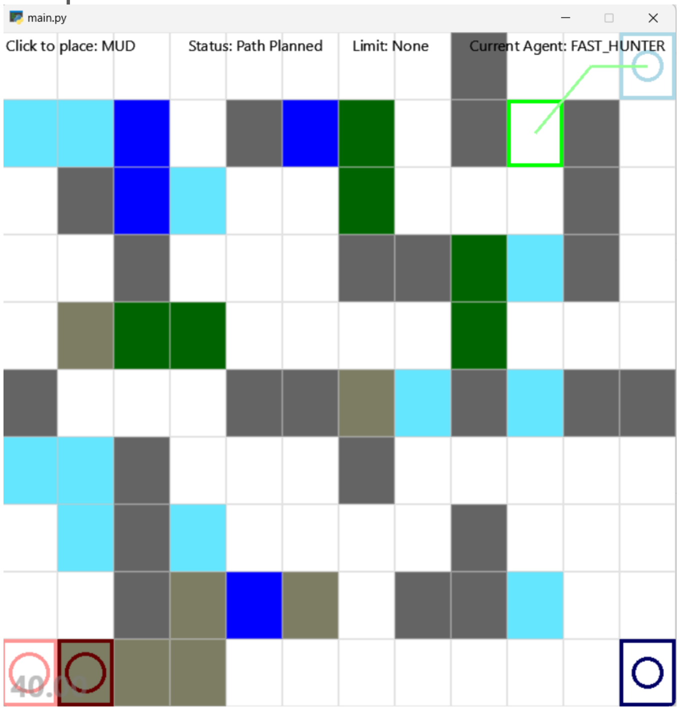


```
# Set target for the Preys and Hunters
def setTarget(self, identificationX):
    '''Set the start box based on its index idx value. '''

    for agent in self.agents:
        if agent.startBox == self.boxes[identificationX]:
            raise ValueError("Can't have the same start and target boxes!")

    self.target = self.boxes[identificationX]
    self.agents[2].targetBox = self.target
    self.agents[3].targetBox = self.target

    self.targetMarker.x = self.target.center().x - self.boxes[0].width / 2
    self.targetMarker.y = self.target.center().y - self.boxes[0].height / 2
```

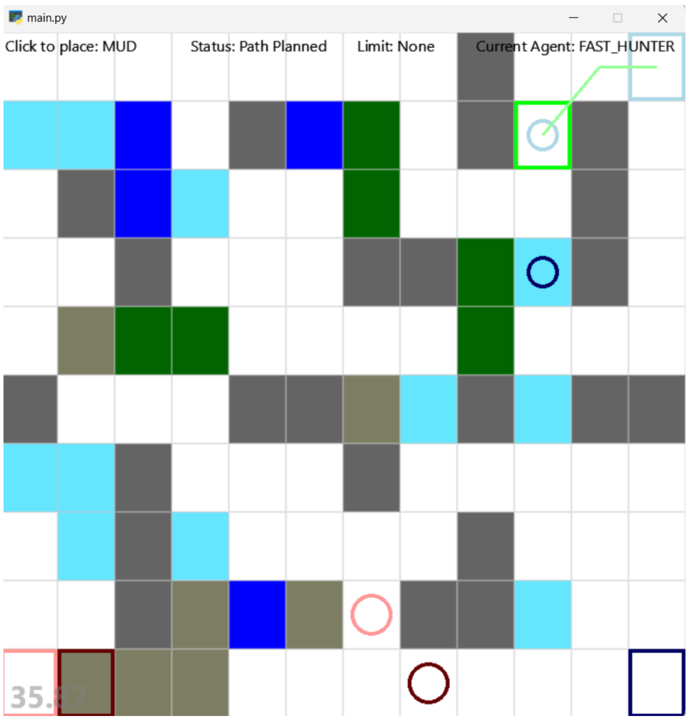
- Start of the agent can will automatically set by the program, or pressing “7” to change the location of each Agent. If the prey changed location, hunter will update their target to the new start of prey

```
    def setStart(self, agent, identificationX):
        '''Set the start box based on its index idx value. '''
        # for agent in self.agents:
        if self.target == self.boxes[identificationX]:
            raise ValueError("Can't have the same start and target boxes!")
        self.start = self.boxes[identificationX]
        agent.startBox = self.start
        self.startMarkers[agent.id].x = agent.startBox.center().x - self.boxes[0].width / 2
        self.startMarkers[agent.id].y = agent.startBox.center().y - self.boxes[0].height / 2

        self.agents[0].targetBox = self.agents[3].currentBox
        self.agents[1].targetBox = self.agents[2].currentBox
```

- Output 1: Initial state



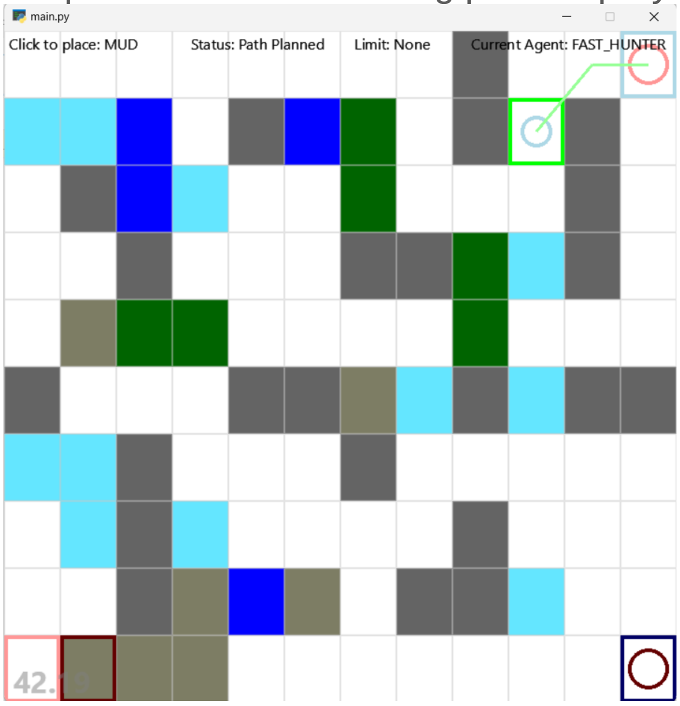
Output 2: Hunter chasing



main.py

Click to place: MUD Status: Path Planned Limit: None Current Agent: FAST_HUNTER

- Output 3: It find the starting point of prey:



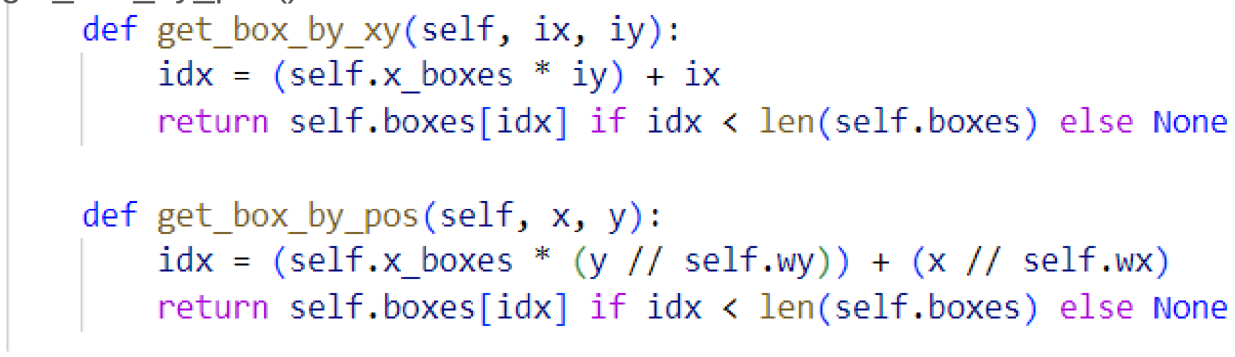
main.py

Click to place: MUD Status: Path Planned Limit: None Current Agent: FAST_HUNTER

Program details:

- Each node can only move to the other node with constant jump speed by using get_neighbor() (which choosing the linked node with the node idx)

- Graph can be added or customized in txt file with width, height and each type of cell.
- Each agent can be added to “game.py” file to interact with the game.
- The map construct by creating each “Box” and add to the “Box World” with each box containing box_type.
- Heuristic function to indicate how far from the current node to the goal, however since the world constructed is node added by node so there need a function to indicate position of the node, which use get_box_by_pos().



```python
def get_box_by_xy(self, ix, iy):
    idx = (self.x_boxes * iy) + ix
    return self.boxes[idx] if idx < len(self.boxes) else None

def get_box_by_pos(self, x, y):
    idx = (self.x_boxes * (y // self.wy)) + (x // self.wx)
    return self.boxes[idx] if idx < len(self.boxes) else None
```