Spike: 16
Title: Soldier On Patrol
Author: Hoang Bao Phuc Chau, 103523966

Goals / deliverables:
Create a “soldier on patrol” simulation where an agent has two or more high-level FSM modes of behaviour and low-
level FSM behaviour. The model must show (minimum)
(a) High level "patrol" and "attack" modes
(b) The "patrol" mode must use a FSM to control low-level states so that the agent will visit (seek/arrive?) a
number of patrol-path way points.
(c) The "attack" mode must use a FSM to control low-level fighting states. (Think “shooting”, “reloading” - the
actual states and transition rules are up to you.)

Resources used:
List of information needed by someone trying to reproduce this work

- Pycharm
- Python 3.12
- Pyglet 2.0.15
- ChatGPT4.0 AI
- Microsoft Copilot AI
- https://www.youtube.com/watch?v=m9jNpzk71ow

Tasks undertaken:

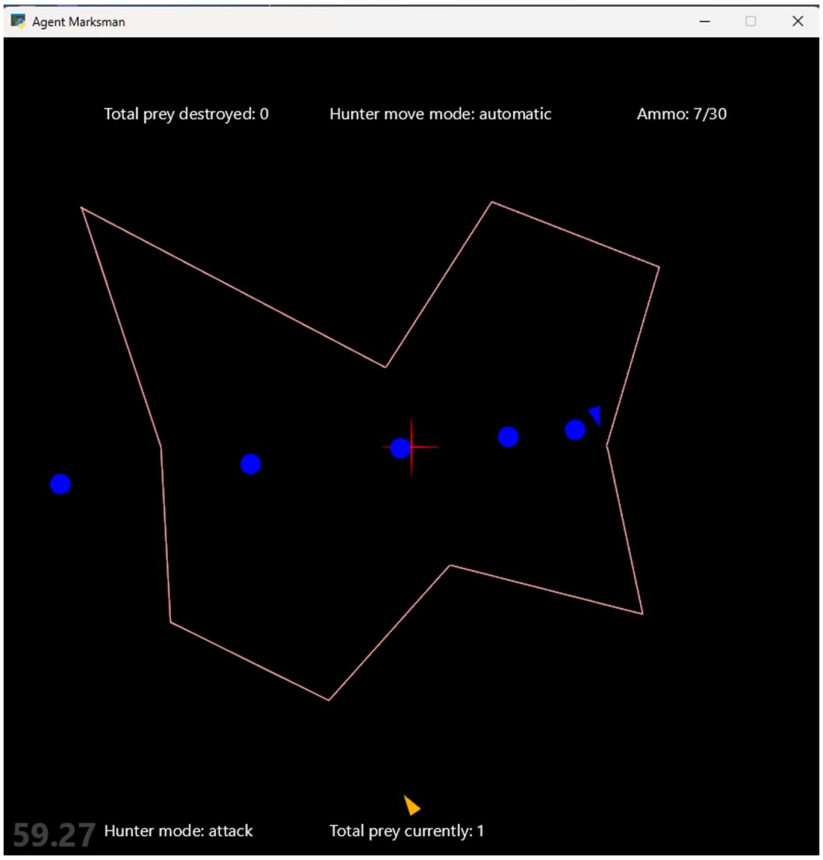
- Copy codebase from task 12
- Inside World class, modify input_keyboard() method so it supports these modes and functions:
  - P: Pause
  - A: Add more prey
  - Z: Patrol Mode
  - X: Attack Mode
  - R: Randomise Path
  - It looks like this

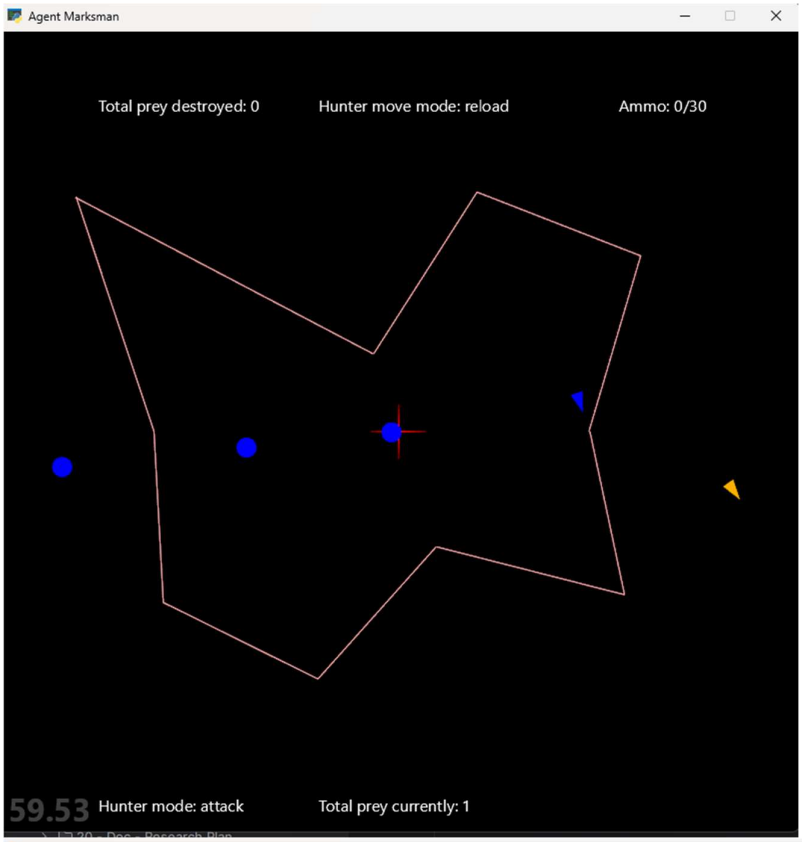
```
Hoang Bao Phuc Chau
def input_keyboard(self, symbol, modifiers):
    if symbol == pyglet.window.key.P:
        self.paused = not self.paused

    elif symbol == pyglet.window.key.A:
        self.preys.append(Prey(self))
        self.total_target_left += 1

    elif symbol == pyglet.window.key.Z:
        self.hunter.mode = 'patrol'
        self.hunter.FSM.mode = self.hunter.mode

    elif symbol == pyglet.window.key.X:
        self.hunter.mode = 'attack'
        self.hunter.FSM.mode = self.hunter.mode

    elif symbol == pyglet.window.key.R:
        self.hunter.randomise_path()
```

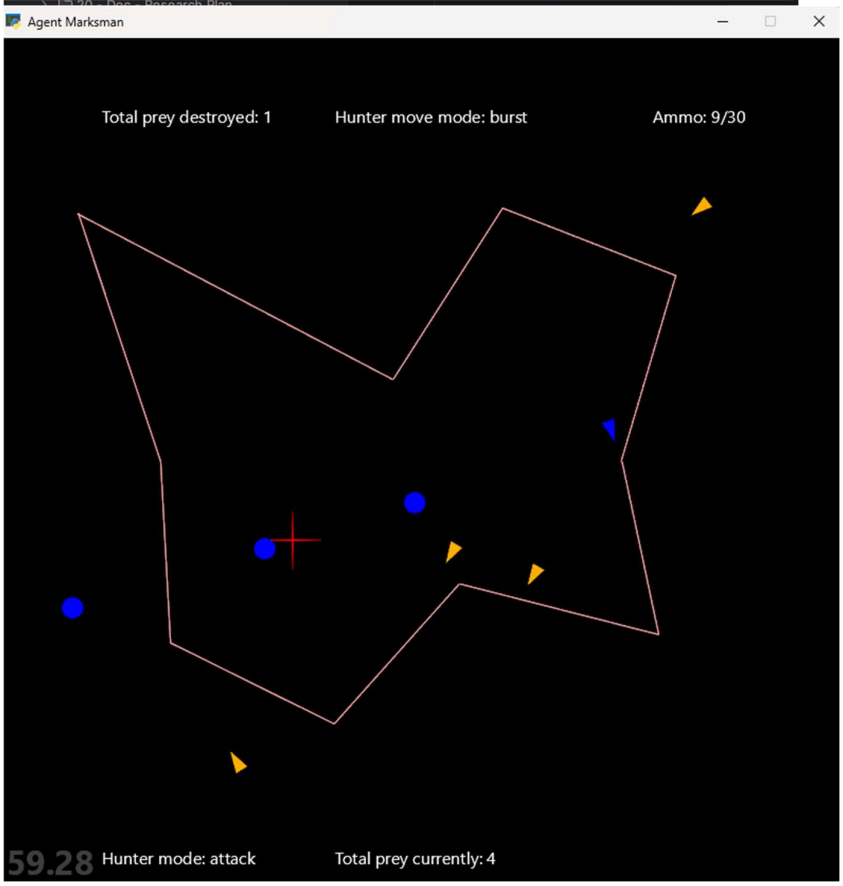
- Inside graphics.py, add 5 labels to observe hunter mode, prey count, number of destroyed prey, hunter state and total ammo
- Inside World class, use window.update_label() method in update() to get the stats in real time.
- Inside agent.py, perform these steps:
  - Create a Hunter class, initialise it with blue colour, 10 scale, 1.0 mass, selected mode as “patrol”. Also, initialise it with Path, max ammo, ammo, FSM and array of projectile (I set max ammo to 30)
    - Copy randomise_path(), follow_path(), current_waypoint_distance() and arrive() from Agent class and pass to Hunter class. However, follow_path() now has an external parameter move_mode to handle correctly
    - Inside calculate() method, firstly set up the target position and acceleration.
      - If mode is patrol, call FSM.exec and assign it to accel (FSM class will be demonstrated later)
      - If mode is attack, set acceleration and velocity to Vector2D() (for it to stand still), then call FSM.exec
      - Assign accel to self.accel then return accel.
    - Inside update() method, it’s mostly the same as the original version, except for having something extra:
      - Loop through all the projectiles to find the nearest prey compare to each projectile, the logic is somewhat similar to what have been done in the last assignments, If a prey is found to be closer than any previously checked prey, it updates nearest_target and nearest_target_distance.
      - If the projectile collides with the prey or gets out of the map, it will get deleted
  - Copy Projectile class and all of its implementation from task 15

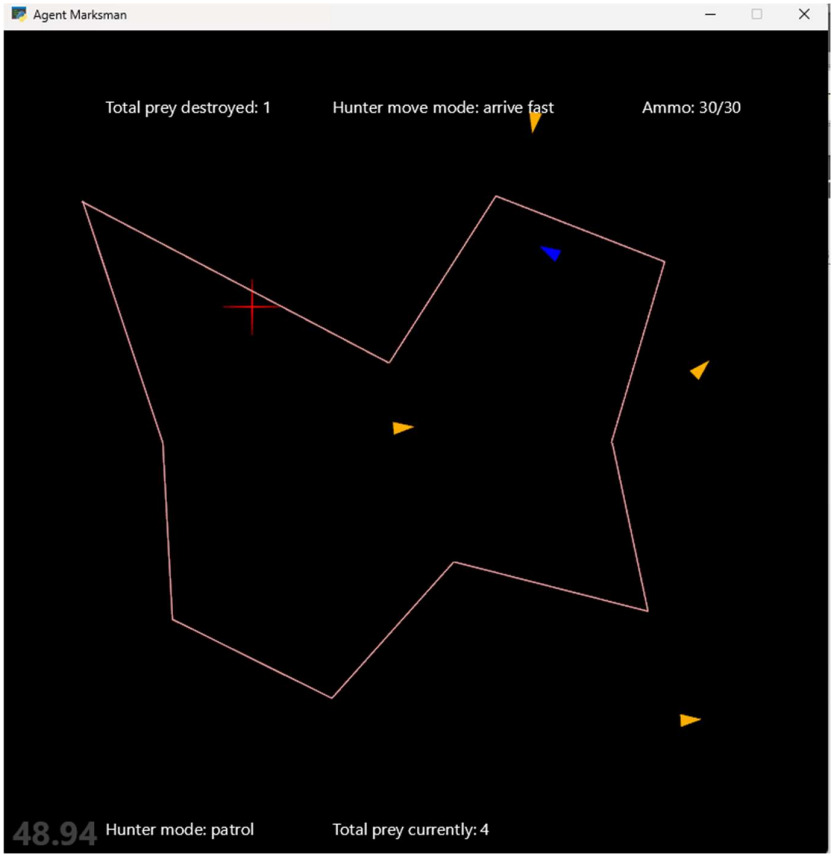
- Create a Prey class, initialise it with orange colour, 10 scale, 1.0 mass, max health as 150 and collision range as 20
  - Implementation is somewhat simple, copy wander() method from Agent class, then assign it to accel in calculate() method
  - Inside update() function, call the update() from Agent class first. If the health reaches 0, change the collision range to 0 and colour to invisible. Also, decrease the total target left by 1 and increase the total target destroyed by 1
- Create an FSM class inherit from Agent
  - Initialise it with properties such as patrol_modes, attack_modes, firing interval, mode index, hunter.
  - Implementation of firing mechanism:
    - Fire_projectile: fire a single projectile by append a projectile to projectiles array, then decrease ammo by 1
    - Burst_fire: burst fire mode which shoot 3 rounds in a short period of time. Use pyglet.clock.schedule_one to add some delay between each shot
    - Automatic_fire: full auto fire mode which will shoot until the magazine is empty. Use pyglet.clock.schedule_one to add some delay between each shot
    - Reload_ammo: reload ammo into the magazine.
  - State management:
    - If the mode is patrol:
      - If the number of target destroyed is fully divisible by 4, set the state to seek
      - If the remainder is 1, set the state to arrive fast.
      - If the remainder is 1, set the state to arrive normal.
      - If the remainder is 1, set the state to arrive slowly.
    - If the mode is attack
      - If the number of prey is larger than 6 and ammo is not empty, set the attack mode to single
      - If the number of prey is larger than 2 and less than 7 and ammo is not empty, set the attack mode to burst
      - If the number of prey is less than 3 and ammo is not empty, set the attack mode to automatic
      - Else, reload ammo.
  - Exec() method implementation:
    - If the mode is set to patrol, based on the condition, return hunter.followpath()
    - If the mode is set to attack, based on the condition, call the suitable attack method such as burst_fire() or automatic_fire().

What we found out:
The program works as expected:


Agent Marksman
Total prey destroyed: 0
Hunter move mode: automatic
Ammo: 7/30
59.27
Hunter mode: attack
Total prey currently: 1


Agent Marksman
Total prey destroyed: 0
Hunter move mode: reload
Ammo: 0/30
59.53
Hunter mode: attack
Total prey currently: 1


Agent Marksman
Total prey destroyed: 1
Hunter move mode: burst
Ammo: 9/30
59.28
Hunter mode: attack
Total prey currently: 4



I also learnt the way how to use FSM effectively to control both high-level and low-level state.

Open issues/risks

- Risk: The hunter will continually shoot until the mag in empty when changing the state suddenly. It also can not change the firing directing while firing. If the user want to change, they must wait for it to finish shooting.

Recommendations