Spike: 15
Title: Agent Marksmanship
Author: Hoang Bao Phuc Chau, 103523966

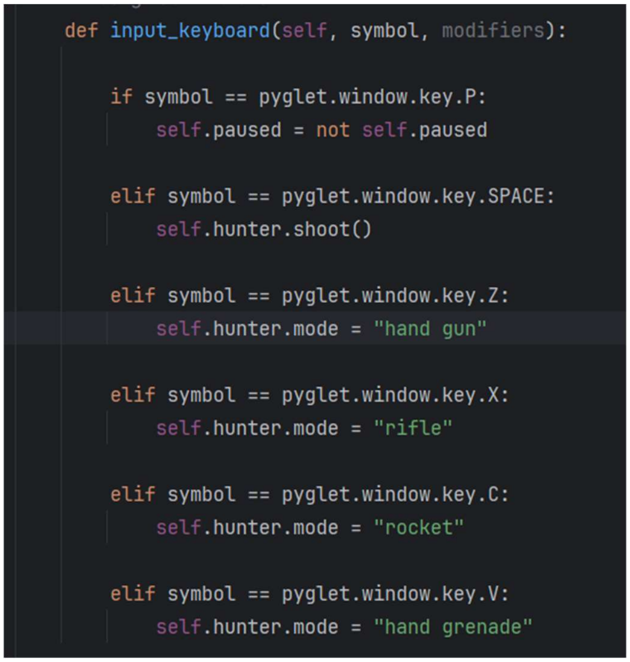
Goals / deliverables:
Create an agent targeting simulation with:
(a) an attacking agent (can be stationary),
(b) a moving target agent (can simply move between two-way points), and
(c) a selection of weapons that can fire projectiles with different properties.

Be able to demonstrate that the attacking agent that can successfully target (hit) with different weapon properties:
(a) Fast moving accurate projectile. (Rifle)
(b) Slow moving accurate projectile. (Rocket)
(c) Fast moving low accuracy projectile (Hand Gun)
(d) Slow moving low accuracy projectile (Hand grenade)**Technologies, Tools, and 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
  - Space: Shoot
  - Z: Hand Gun
  - X: Rifle
  - C: Rocket
  - V: Hand Grenade
  - It looks like this:



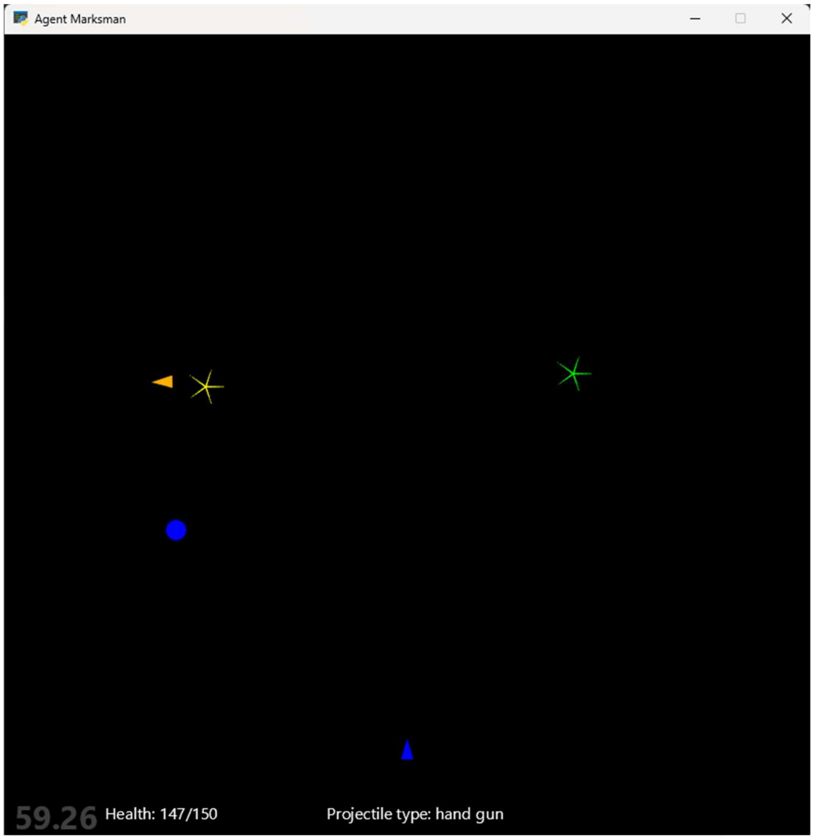
```
def input_keyboard(self, symbol, modifiers):

    if symbol == pyglet.window.key.P:
        self.paused = not self.paused

    elif symbol == pyglet.window.key.SPACE:
        self.hunter.shoot()

    elif symbol == pyglet.window.key.Z:
        self.hunter.mode = "hand gun"

    elif symbol == pyglet.window.key.X:
        self.hunter.mode = "rifle"

    elif symbol == pyglet.window.key.C:
        self.hunter.mode = "rocket"

    elif symbol == pyglet.window.key.V:
        self.hunter.mode = "hand grenade"
```

- Modify input_mouse() method to:
    - Press Right Click to change first destination position
    - Press Left Click to change second destination position
- Inside graphics.py, add 2 labels to observe health of prey and type of projectile the hunter is carrying.
- 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, 0 scale, 1.0 mass, selected mode as “rifle” and position at x = world.cx/2, y = 100
        - Write a method to change the projectile speed, where ‘rifle’ and ‘hand gun’ get 3, ‘rocket’ gets 1.5 and ‘hand grenade’ gets 1
        - Write a method to change the projectile power, where ‘rifle’ gets 3, ‘hand gun’ get 2, ‘rocket’ gets 10 and ‘hand grenade’ gets 20
        - Write a method to change the projectile inaccuracy rate, where ‘rifle’ gets 0 (always accurate), ‘hand gun’ get 1.5, ‘rocket’ gets 0 and ‘hand grenade’ gets 1.7
        - Write an attack() method to calculate the acceleration of the projectile using the formula demonstrated inside the youtube video, then solve the equation where pos(projectile) = pos(prey) to get the acceleration of the projectile
        - Write a shoot() method to shoot the projectile (using the Projectile class, which will be demonstrated later)
        - Write a calulate() method, which will assign the value of attack() to accel, then return it
        - Copy update() from Agent class, then made some modification
            - Delete code lines used for updating the position of the agent. Instead, call calculate() method only
            - If the projectile is shot, check if it collide with the prey. If yes, delete the projectile. This also applies to the case where the projectile gets out of the screen.

- Create a Projectile class, initialise it with blue colour, 0 scale, 1.0 mass, and add the inaccuracy_property. Also, make it to have circle shape
  - Update the velocity with Acceleration
  - Next, adjust the velocity for inaccuracy, using the rotation matrix [cos(θ)sin(θ)−sin(θ)cos(θ)] where θ is self.inaccuracy_rate * delta
  - Update the position, the rest of the code is similar to the original update()
- Create a Prey class, initialise it with orange colour, 40 scale, 1.0 mass.
  - Initialise it with collision range, max health and destination (here I set collision range = 20 and max health = 150)
  - Setup position for first and second destination. Set position for the prey at the center of 2 destinations
  - Write 2 methods for updating position of the first and second destination (only update when the distance from desired point to the destination is larger than 100)
  - Inside calculate() method, determine the acceleration based on the prey destination. Also, if the prey is in the collision range of the destination, switch the destination.
  - Update() will be called from Agent class

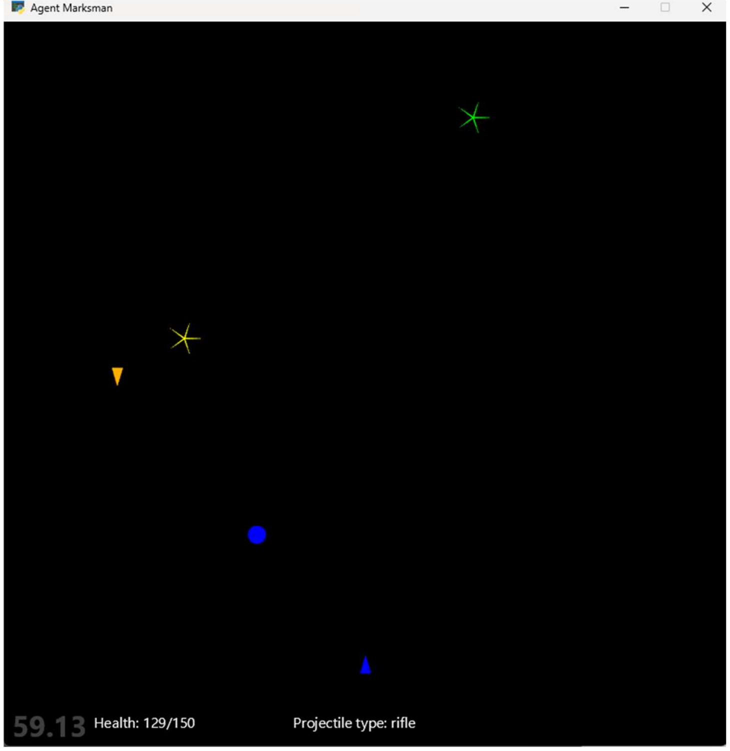
What we found out:

The program works as expected:



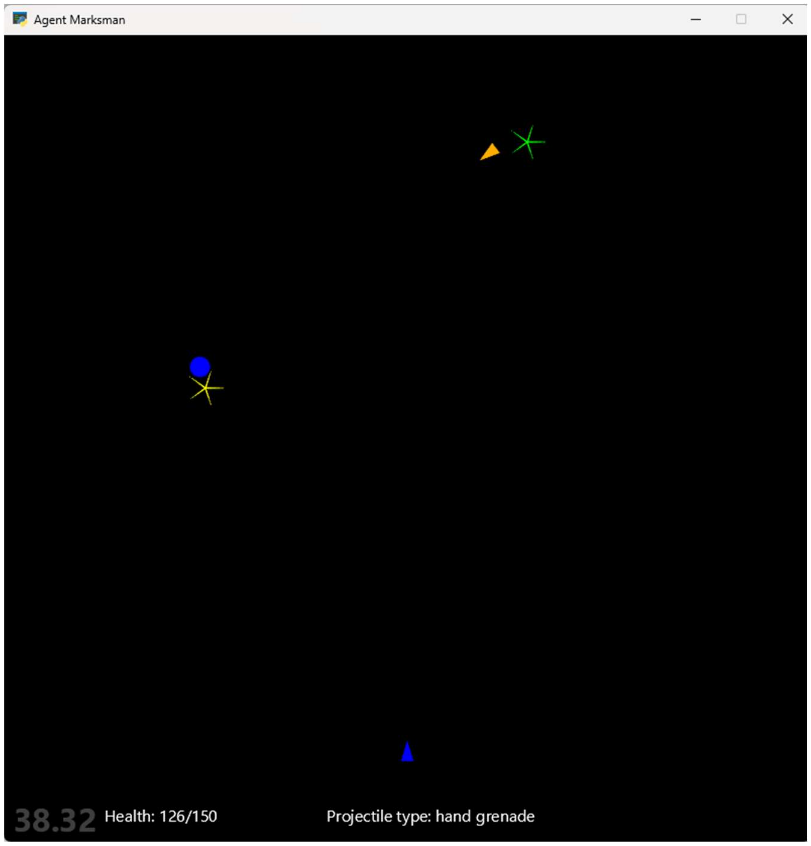
59.26 Health: 147/150

Projectile type: hand gun



59.13 Health: 129/150

Projectile type: rifle



Open issues/risks [Optional – **remove** heading/section if not used!]**:**
List out the issues and risks that you have been unable to resolve at the end of the spike. You may have uncovered a whole range of new risks as well.

- eg. Risk xyz (new)

Recommendations [Optional – **remove** heading/section if not used!]**:**
Often based on any open issues/risks identified. You may state that another spike is required to resolve new issues identified (or) indicate that this spike has increased your confidence in XYZ and should move on.