**Spike:** 14
**Title:** Emergent Group Behaviour
**Author:** Hoang Bao Phuc Chau, 103523966

**Goals / deliverables:**

Create a group agent steering behaviour simulation that is able to demonstrate distinct modes of emergent group behaviour. In particular, the simulation must

- Include cohesion, separation and alignment steering behaviours
- Include basic wandering behaviours
- Use a weighted-sum to combine all steering behaviours
- Support the adjustment of parameters for each steering force while running
- Spike outcome report and working code (with key instructions).

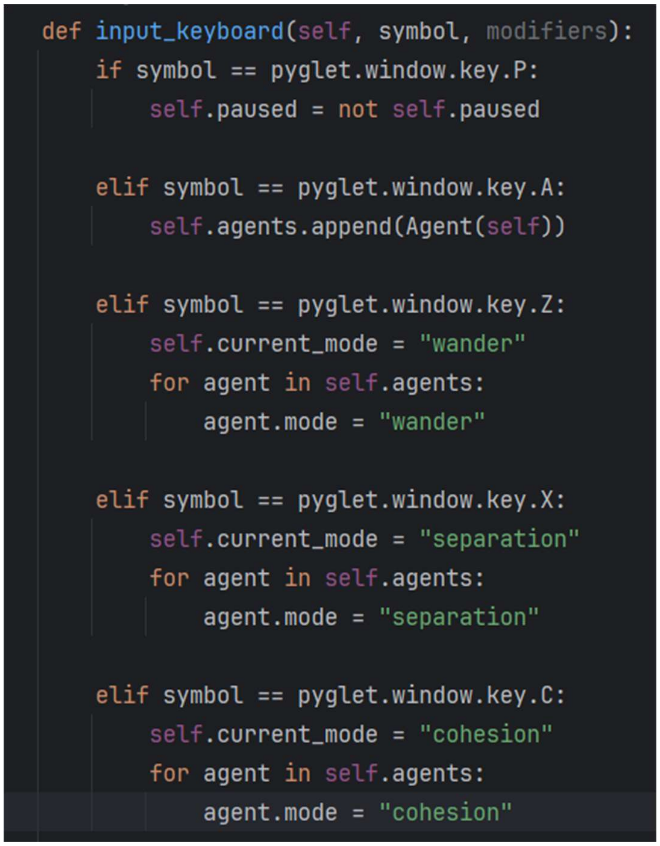
**Technologies, Tools, and Resources used:**

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

**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 Agents
    - Z: Wander
    - X: Separation
    - C: Cohesion
    - 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.A:
        self.agents.append(Agent(self))

    elif symbol == pyglet.window.key.Z:
        self.current_mode = "wander"
        for agent in self.agents:
            agent.mode = "wander"

    elif symbol == pyglet.window.key.X:
        self.current_mode = "separation"
        for agent in self.agents:
            agent.mode = "separation"

    elif symbol == pyglet.window.key.C:
        self.current_mode = "cohesion"
        for agent in self.agents:
            agent.mode = "cohesion"
```

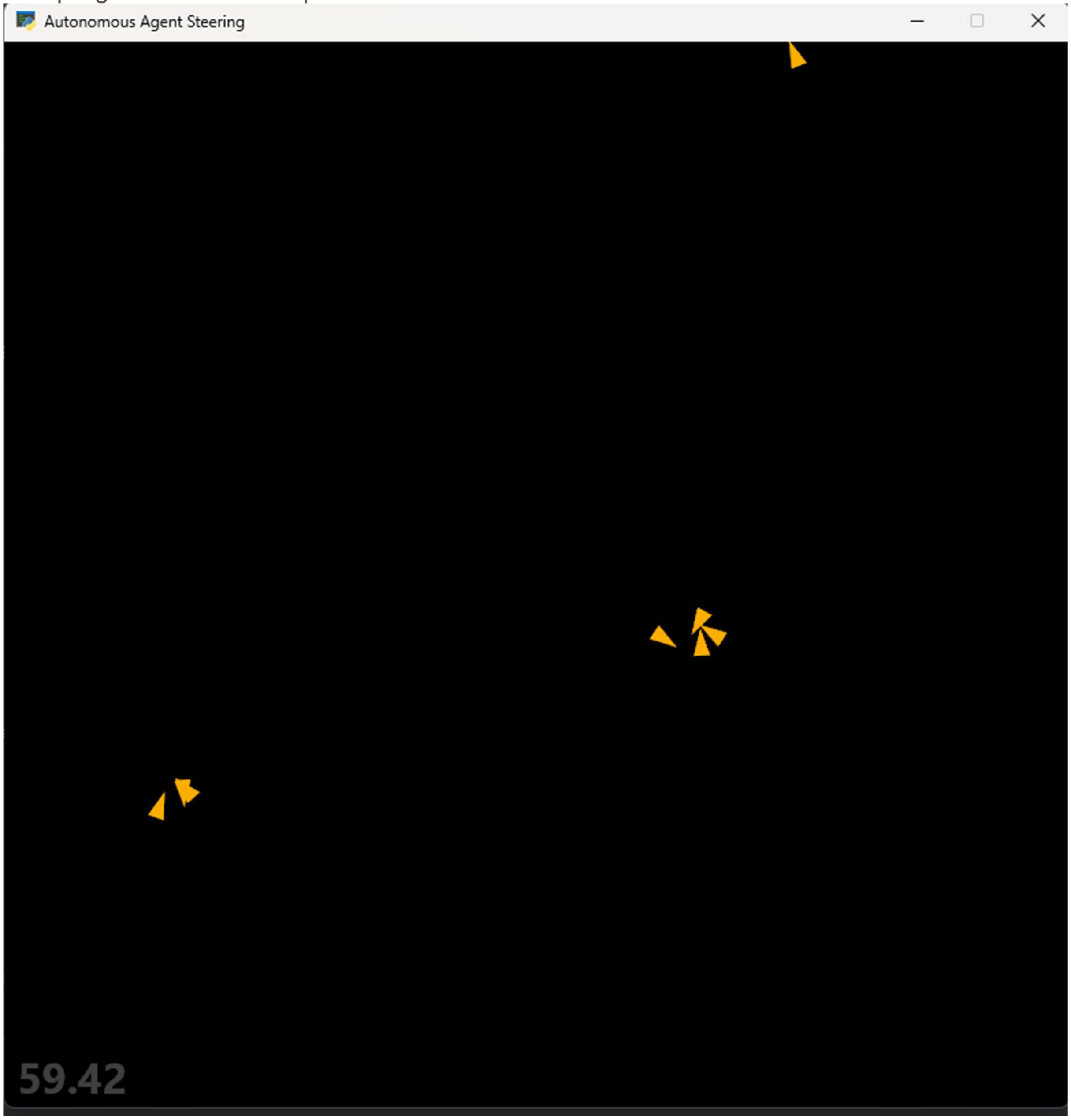
- Inside Agent class, perform these steps:
    - Declare a method named get_near_range_agents() to get all of agents nearby
        - Loop through the list of agents in the game
        - Get the vector between ourselves and the agent at the iteration.
        - Get the length of the vector.
        - Compare it to the declared near_range property. If smaller, push the agent into the array of result
        - Return the result
    - For getting far range agents, just copy the get_near_range_agents() method and compare the vector length with the far_range property.
    - Inside update() method, perform these modifications:
        - If current mode is wander, use existing wander() method
        - If current mode is separation:
            - Get all the near range agents
            - If there is no one nearby, continue using wander() method
            - If there is at least 1 agent, take the first agent in the list and use the flee() method
        - If current mode is Cohesion
            - Get all the near range agents
            - If there is no one nearby, continue using wander() method
            - If there is at least 1 agent, sum up all of the agents' position (including the current one), then divide them by total number of agents nearby (including the current one). The apply seek()

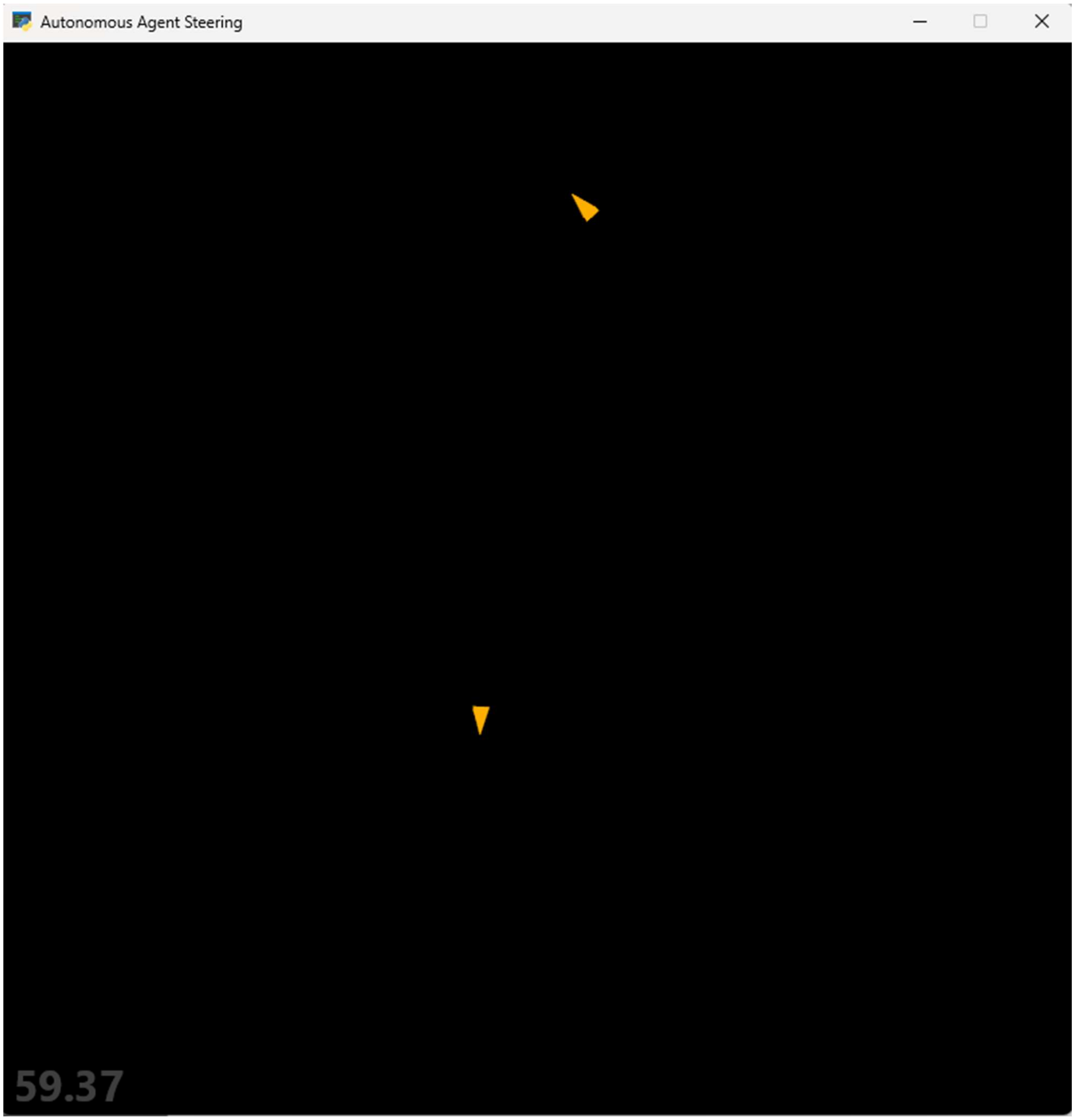
method to them. After that, they will all move to the same point with the same direction.

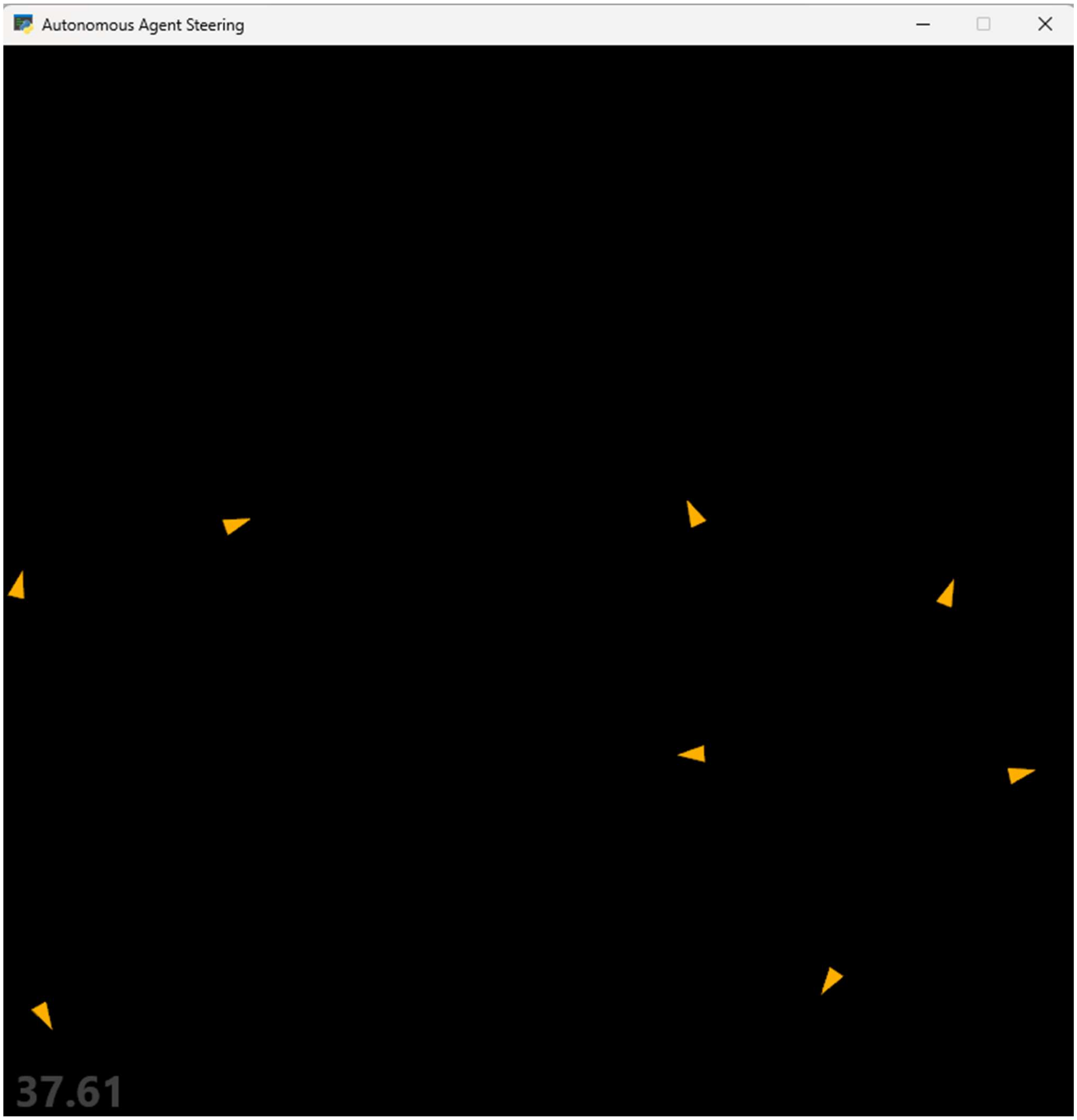
- If the mode is alignment:
  - Get all the near range agents
  - If there is no one nearby, continue using wander() method
  - If there is at least 1 agent, calculate the total force of all the agents in far range.
- Return force

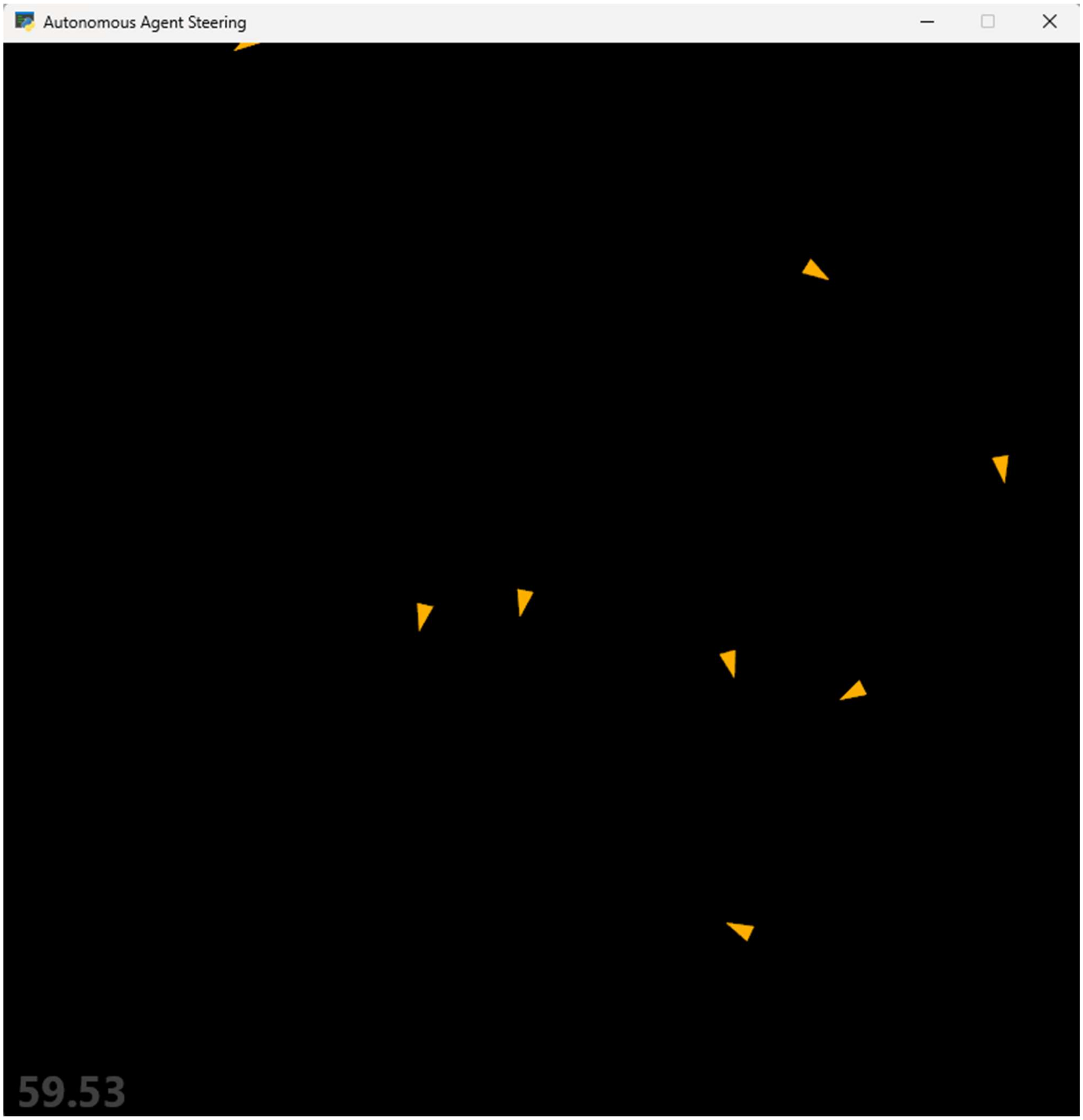
**What we found out:**

The program works as expected:




Autonomous Agent Steering
59.37


Autonomous Agent Steering
37.61



**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.