Spike: 13
Title: Tactical Steering (Hide!)

Author: Ben Holmes, 103024841

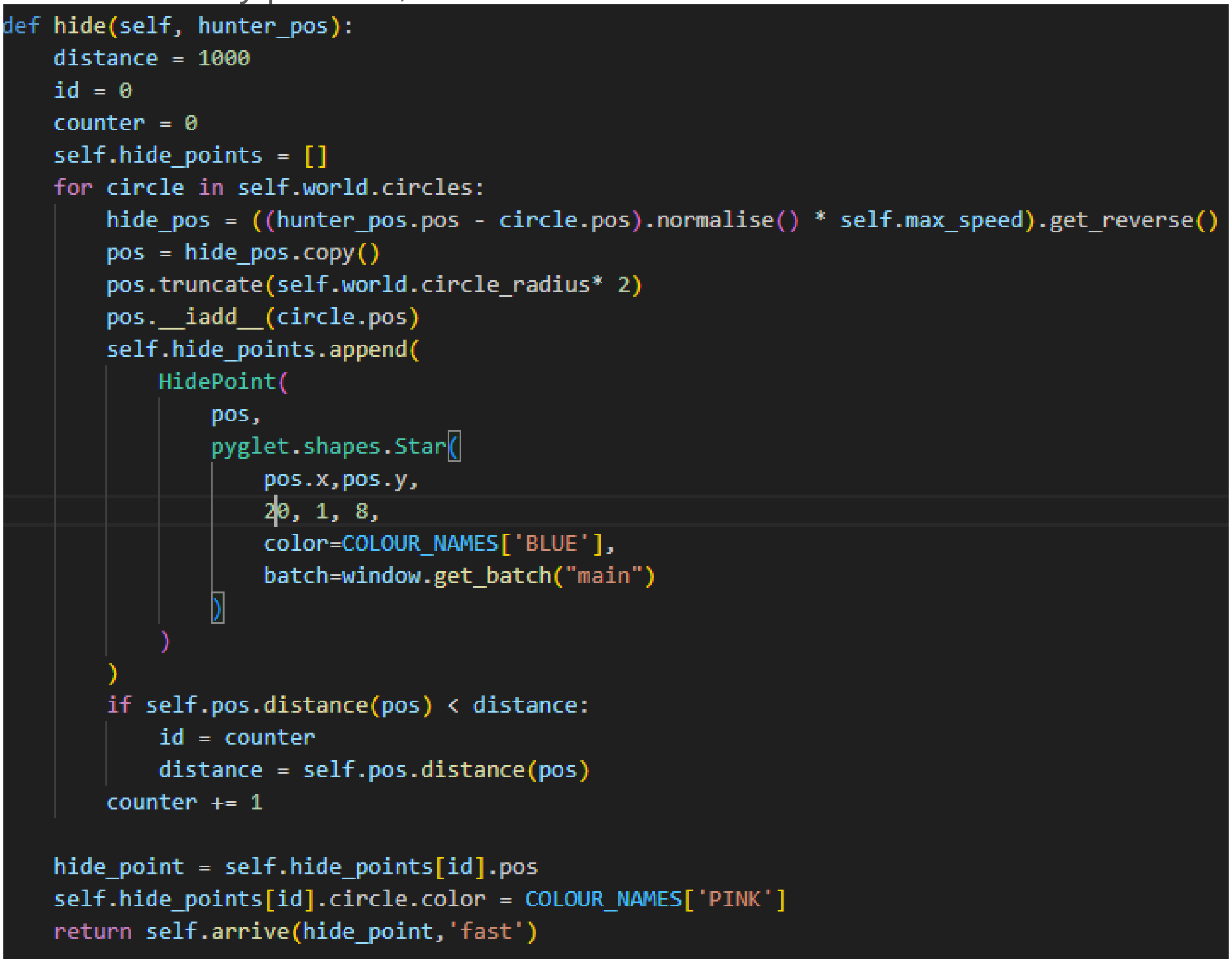
Goals / deliverables:
Create a hunter-prey agent simulation for two or more agents, in which "prey" agents avoid "hunter" agents by
concealing themselves behind objects in the environment. The simulation must:
• Include several "objects" that prey can hide behind (simple circles).
• Show a distinction between the "hunter" and "prey" agent appearance and abilities.
• Show an indicator ("x" or similar) to indicate suitable "hide" locations for prey to select from
• Prey agents must select a “good” location, and head to it, based on tactical evaluation.
• Do NOT hide “inside” objects - rather find a location outside (behind).

Technologies, Tools, and Resources used:
- Visual Studio Code
- Python 3.12.2
- Pyglet

Tasks undertaken:

- Copied code from lab 12
- Added a hunter, coloured purple
- Added 2 classes, HidePoint and Obstacle in the HideClasses.py file, these were to store the position and shape of the circles (Obstacle) and hide points (HidePoint) in the same object so they could be added to lists easily
- Added 5 randomly spawning circles (using Obstacle), coloured green (world stored a circle_radius variable which was used later in hide code)
- Added basic hide functionality (as in the layout and temp code)
- Added figuring out of hide position by subtracting circle position from hunter position, multiplying that by max speed and then reversing it, then truncating (limiting value) to circle_Radius times 2, and then adding circle position. This creates a blue star (closest point to one of the agents is pink) on the other side of the circle from the hunter, that moves around the circle at twice the circles radius

- Added a check for the agent (non-hunter) to move to the closest point to it, this is done in the same loop as the figuring out each circles hide position using a counter, and an initial distance set to a value higher than realistically possible, HIDE CODE:



```python
def hide(self, hunter_pos):
    distance = 1000
    id = 0
    counter = 0
    self.hide_points = []
    for circle in self.world.circles:
        hide_pos = ((hunter_pos.pos - circle.pos).normalise() * self.max_speed).get_reverse()
        pos = hide_pos.copy()
        pos.truncate(self.world.circle_radius* 2)
        pos.__iadd__(circle.pos)
        self.hide_points.append(
            HidePoint(
                pos,
                pyglet.shapes.Star(
                    pos.x,pos.y,
                    20, 1, 8,
                    color=COLOUR_NAMES['BLUE'],
                    batch=window.get_batch("main")
                )
            )
        )
        if self.pos.distance(pos) < distance:
            id = counter
            distance = self.pos.distance(pos)
        counter += 1

    hide_point = self.hide_points[id].pos
    self.hide_points[id].circle.color = COLOUR_NAMES['PINK']
    return self.arrive(hide_point,'fast')
```

What we found out:

The first and second deliverables were necessary to allow for hiding to be done.

The third deliverable was needed to show where the hiding spots were when faced with a single hunter so that the agent could go to the hiding spots

The fourth deliverable was the tactical analysis, where picking the closest hiding spot for the agent was best as it needed to hide more quickly.

The fifth deliverable was mainly so that the agent was not clipping inside the object constantly so it was actually hiding from the hunter correctly

The main issue I had was figuring out how to get the vector of the hide position in the first place, and then it was figuring out how to translate that to the correct point near the circle. Once I had figured that out the rest was very simple

Testing buttons:

I added some extra buttons that make testing much easier.

First was changing between whether other buttons changed agents or the hunter, it starts in agent mode, “A” changes it to agent and “H” changes it to hunter. This applies to the number keys and the force change not the path randomiser or the agent add (as there is only one hunter currently implemented in code)

Second was adding max force and max speed increase and decrease.

“S” changes the mode to speed and “F” for force. Up arrow key increases by 100 and down arrow key decreases by 100