Spike: 3
Title: Tactical Steering

Author: Ryan Chessum - 102564760

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).

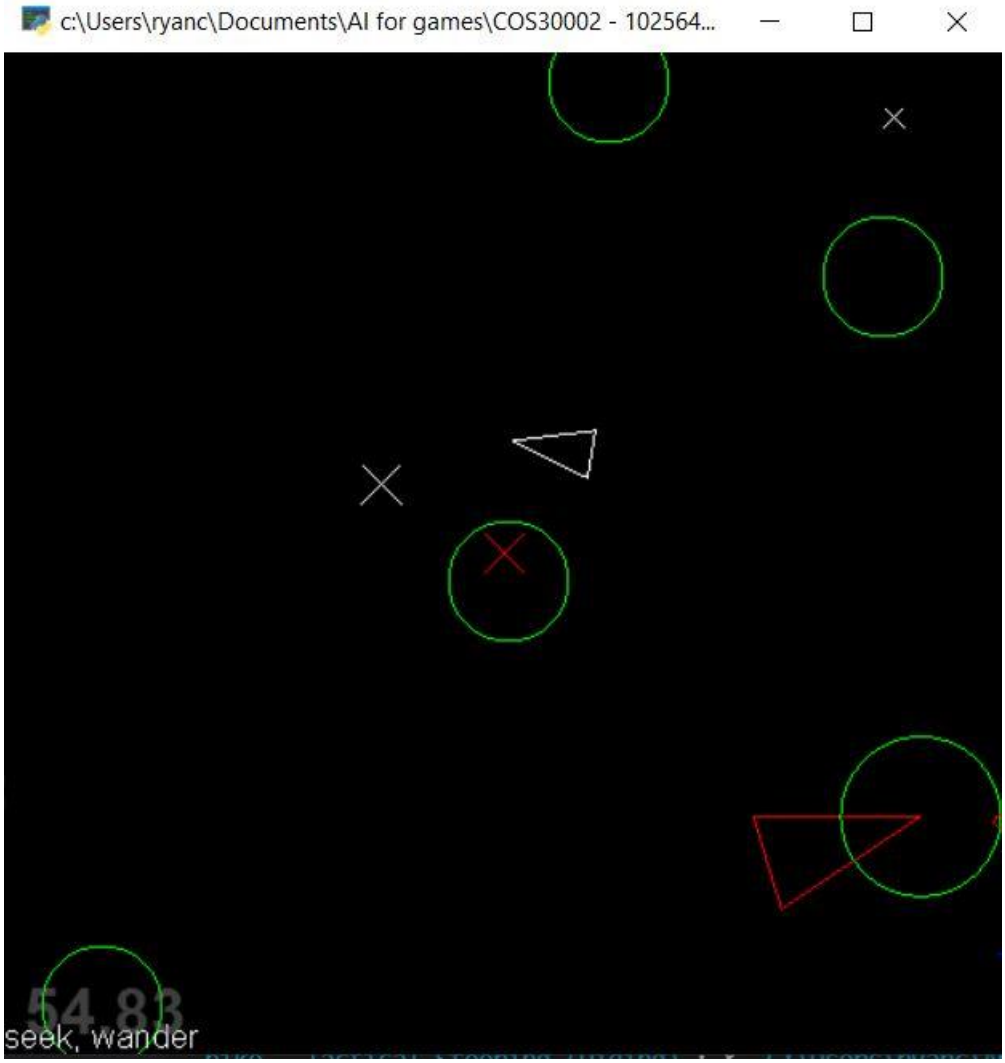
Technologies, Tools, and Resources used:

- Visual Studio Code
- Python 3.8.0
- Pyglet
- Lecture Notes

Tasks undertaken:

- Create an agent class that can move around on the window screen or use the same class from Lab 9
- Create 2 agent sub classes one for the Hunter and one for the Prey
- Edit hunter and prey properties to make them distinct
- Create a class for objects that the prey can hide behind
- Add functionality to add these objects to the scene
- Create a function that searches for hiding spots based on the location of the hunters position, returns a location behind the object relative to the hunter
- Create a hide function in the prey class that searches for the best hiding location and returns the velocity needed to travel there (uses the inherited arrive() function
- Give the hunter basic movement using wandering

What we found out:



The hiding spots the prey use are determined by this function.

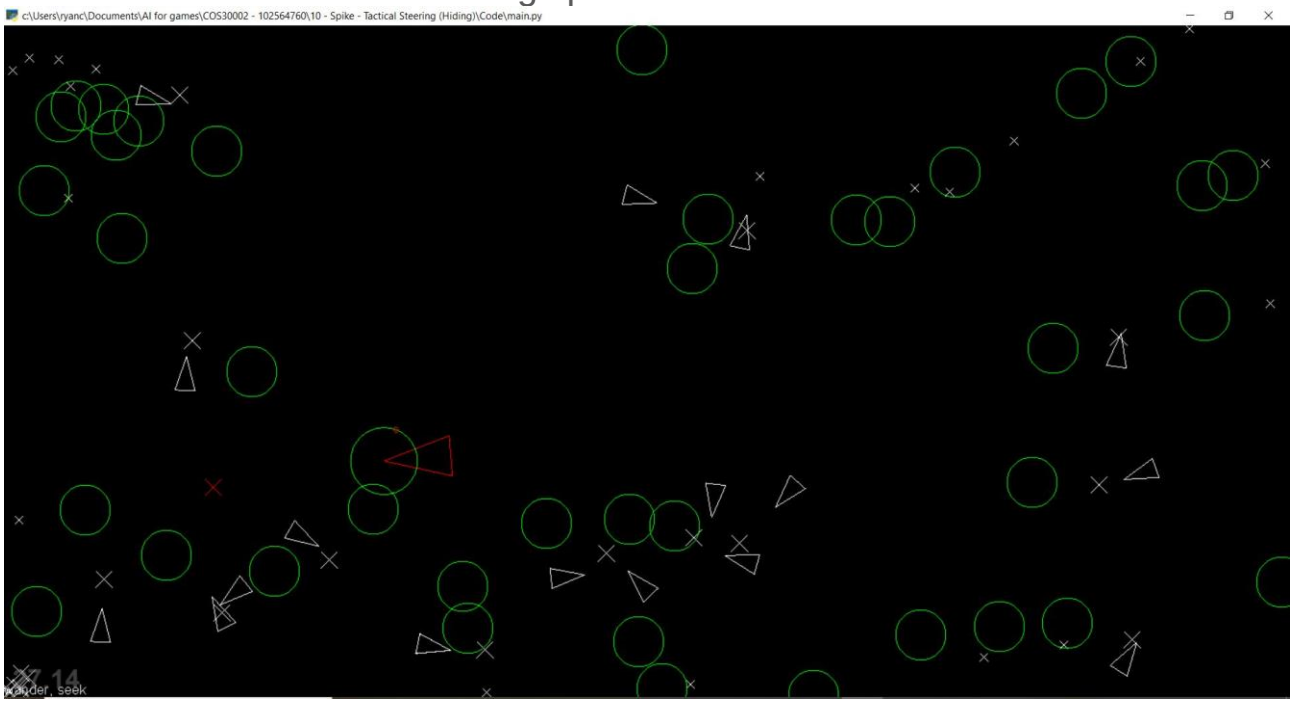
```
def get_hiding_spot(self, hunter, obj):
    # set the distance between the object and the hiding point
    DistFromBoundary = 50.0

    distAway = obj.radius + DistFromBoundary
    toObj = (obj.pos - hunter.pos).normalise()

    return (toObj*distAway)+obj.pos
```

Originally I had the DistFromBoundary variable as 10 but the Prey moved too close to the circle so I moved it back to 50 and it worked much more nicely.

The Hide function the prey use cycles through all of the found hiding spots then picks the best one based on it's proximity to them.

```
def Hide(self, hunter, objs):
    DistToClosest = 100000000.0
    BestHidingSpot = None
    for obj in objs:
        HidingSpot = self.get_hiding_spot(hunter, obj)
        HidingDist = self.pos.distanceSq(HidingSpot)
        egi.white_pen
        egi.cross(HidingSpot, 5.0)
        if HidingDist < DistToClosest:
            DistToClosest = HidingDist
            BestHidingSpot = HidingSpot
    if BestHidingSpot:
        egi.green_pen
        egi.cross(BestHidingSpot, 10.0)
        return self.arrive(BestHidingSpot, 'fast')
    return Vector2D()
```

In the Image the hunter is the Large red triangle and the prey are the small white ones. The green circles are the cover objects. The x's are where hiding spots have been found and the larger x's are x's that a Prey agent has determined to be the best hiding spot.



Keys:

- Create new prey agent: a
- Create new hiding object: o
- Randomise hiding object locations: r

Risks and Issues:

- Because I didn’t implement agent collisions, often multiple Prey agents will eventually become stacked on top of one another as they are all going for the same spot