Goals:

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:

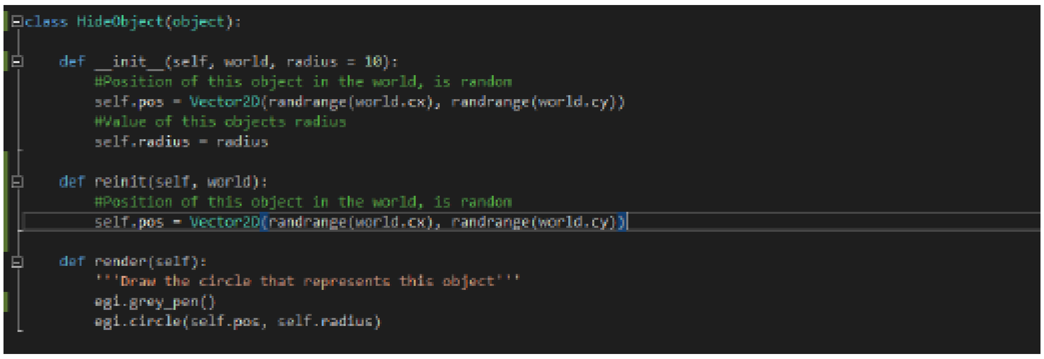
- Include several "objects" that prey can hide behind (simple circles).
- Show a distinction between the "hunter" and "prey" agent appearance and abilities.
- Show an indicator ("x" or similar) to indicate suitable "hide" locations for prey to select from
- Prey agents must select a “good” location, and head to it, based on tactical evaluation.

Technologies, Tools, and Resources used:

- Python IDE(PyCharm) with python 3 installed
- Piglet Documentation http://pyglet.readthedocs.io/en/pyglet-1-3-maintenance/
- Python 3 Documentation http://docs.python.org/
- The code from previous lab task
- Help from peers.

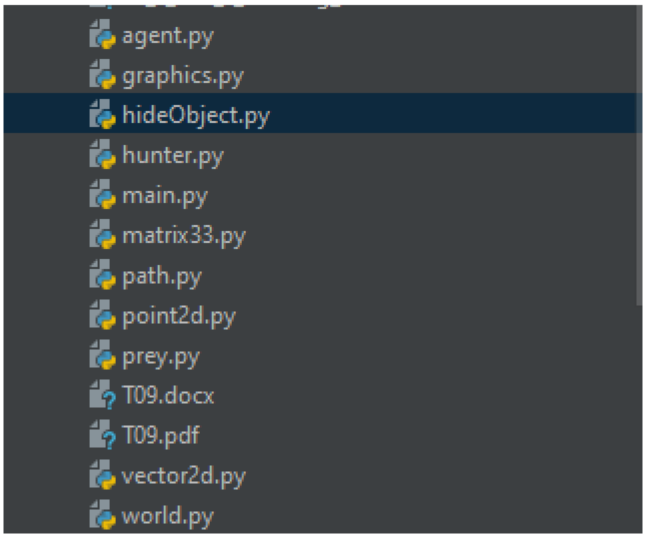
Tasks done:

1. Created a class for an object for the prey agent to hide behind (hideobject.py)



```
class HideObject(object):

    def __init__(self, world, radius = 10):
        #Position of this object in the world, is random
        self.pos = Vector2D(randrange(world.cx), randrange(world.cy))
        #Value of this objects radius
        self.radius = radius

    def reinit(self, world):
        #Position of this object in the world, is random
        self.pos = Vector2D(randrange(world.cx), randrange(world.cy))

    def render(self):
        '''Draw the circle that represents this object'''
        egi.grey_pen()
        egi.circle(self.pos, self.radius)
```

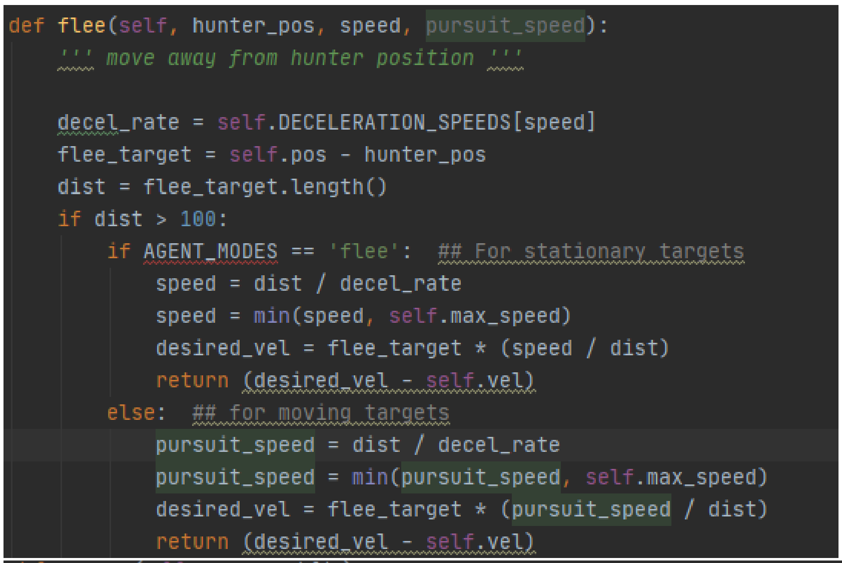
2. Used agent class (as a parent class) to create prey and hunter agents’ classes(child)



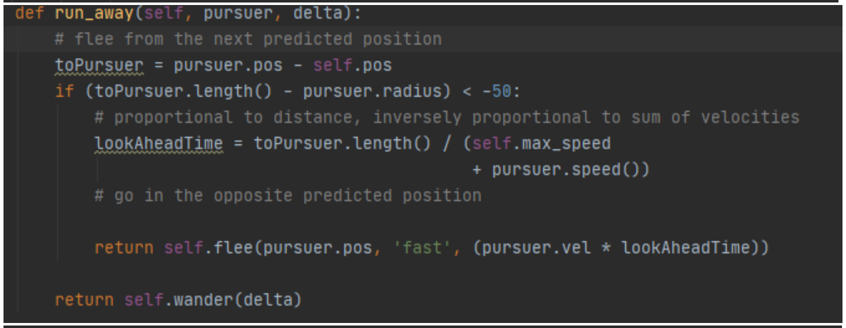
```
agent.py
graphics.py
hideObject.py
hunter.py
main.py
matrix33.py
path.py
point2d.py
prey.py
T09.docx
T09.pdf
vector2d.py
world.py
```

.

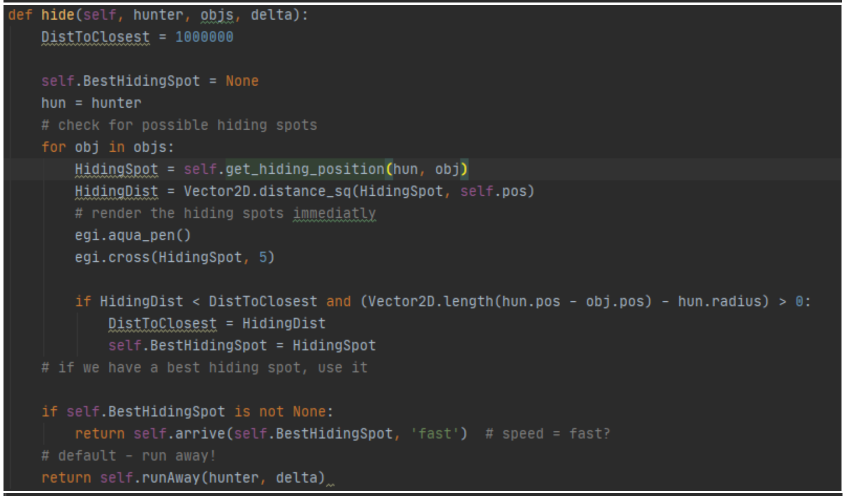
3. Sounded by the word ‘Prey’ I used Flee, runaway, hide and get_hiding_postion functions as the major roles for the class.



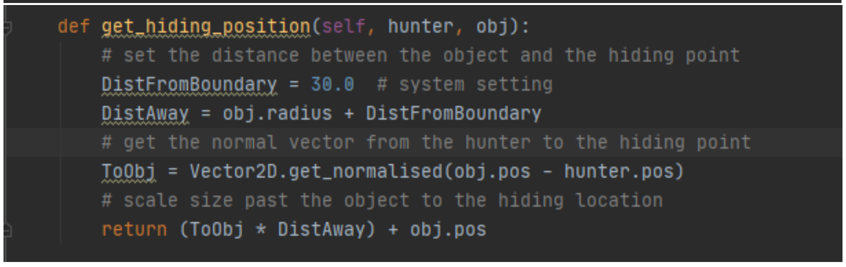
```python
def flee(self, hunter_pos, speed, pursuit_speed):
    ''' move away from hunter position '''

    decel_rate = self.DECELERATION_SPEEDS[speed]
    flee_target = self.pos - hunter_pos
    dist = flee_target.length()
    if dist > 100:
        if AGENT_MODES == 'flee':  ## For stationary targets
            speed = dist / decel_rate
            speed = min(speed, self.max_speed)
            desired_vel = flee_target * (speed / dist)
            return (desired_vel - self.vel)
        else:  ## for moving targets
            pursuit_speed = dist / decel_rate
            pursuit_speed = min(pursuit_speed, self.max_speed)
            desired_vel = flee_target * (pursuit_speed / dist)
            return (desired_vel - self.vel)
```



```python
def run_away(self, pursuer, delta):
    # flee from the next predicted position
    toPursuer = pursuer.pos - self.pos
    if (toPursuer.length() - pursuer.radius) < -50:
        # proportional to distance, inversely proportional to sum of velocities
        lookAheadTime = toPursuer.length() / (self.max_speed
                                              + pursuer.speed())
        # go in the opposite predicted position

        return self.flee(pursuer.pos, 'fast', (pursuer.vel * lookAheadTime))

    return self.wander(delta)
```

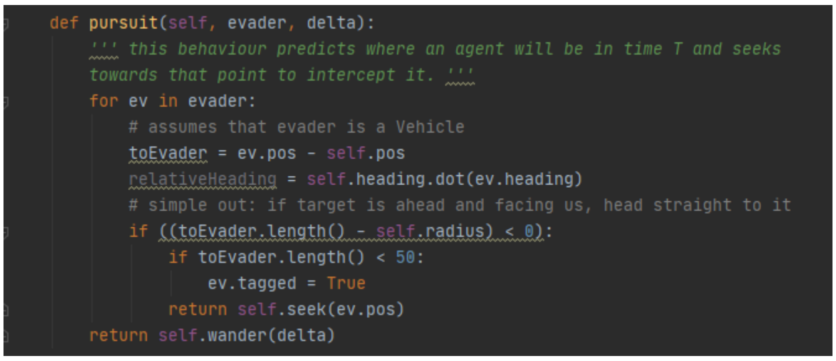


```python
def hide(self, hunter, objs, delta):
    DistToClosest = 1000000

    self.BestHidingSpot = None
    hun = hunter
    # check for possible hiding spots
    for obj in objs:
        HidingSpot = self.get_hiding_position(hun, obj)
        HidingDist = Vector2D.distance_sq(HidingSpot, self.pos)
        # render the hiding spots immediatly
        egi.aqua_pen()
        egi.cross(HidingSpot, 5)

        if HidingDist < DistToClosest and (Vector2D.length(hun.pos - obj.pos) - hun.radius) > 0:
            DistToClosest = HidingDist
            self.BestHidingSpot = HidingSpot
    # if we have a best hiding spot, use it

    if self.BestHidingSpot is not None:
        return self.arrive(self.BestHidingSpot, 'fast')  # speed = fast?
    # default - run away!
    return self.runAway(hunter, delta)
```



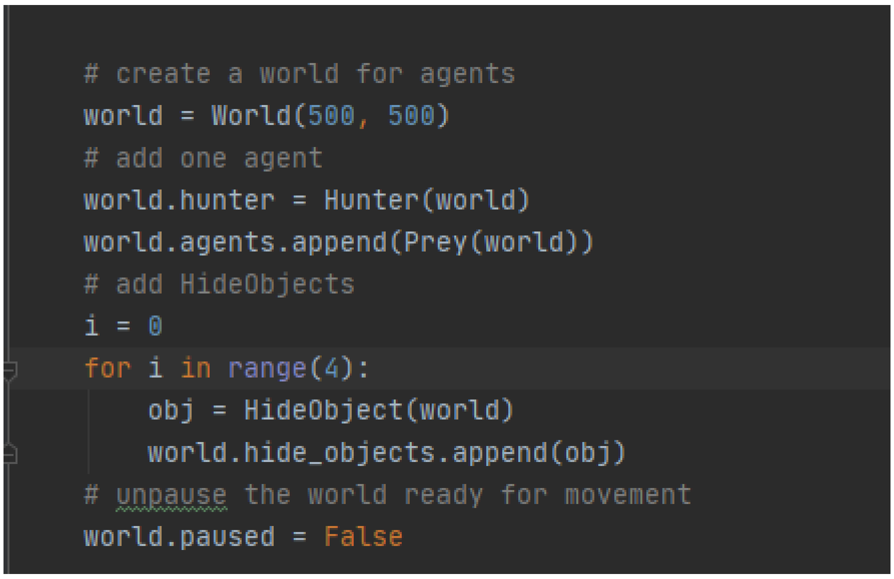
```python
def get_hiding_position(self, hunter, obj):
    # set the distance between the object and the hiding point
    DistFromBoundary = 30.0  # system setting
    DistAway = obj.radius + DistFromBoundary
    # get the normal vector from the hunter to the hiding point
    ToObj = Vector2D.get_normalised(obj.pos - hunter.pos)
    # scale size past the object to the hiding location
    return (ToObj * DistAway) + obj.pos
```

4. The hunter sounded by the word itself, its nature is to pursue the prey:



```
def pursuit(self, evader, delta):
    ''' this behaviour predicts where an agent will be in time T and seeks
    towards that point to intercept it. '''
    for ev in evader:
        # assumes that evader is a Vehicle
        toEvader = ev.pos - self.pos
        relativeHeading = self.heading.dot(ev.heading)
        # simple out: if target is ahead and facing us, head straight to it
        if ((toEvader.length() - self.radius) < 0):
            if toEvader.length() < 50:
                ev.tagged = True
            return self.seek(ev.pos)
    return self.wander(delta)
```

5. Modifying the main.py to create prey agent in the world agents list and hunter in the world hunter, then adding 4 hiding objects for the prey to hide behind in world hide objects.



```
# create a world for agents
world = World(500, 500)
# add one agent
world.hunter = Hunter(world)
world.agents.append(Prey(world))
# add HideObjects
i = 0
for i in range(4):
    obj = HideObject(world)
    world.hide_objects.append(obj)
# unpause the world ready for movement
world.paused = False
```

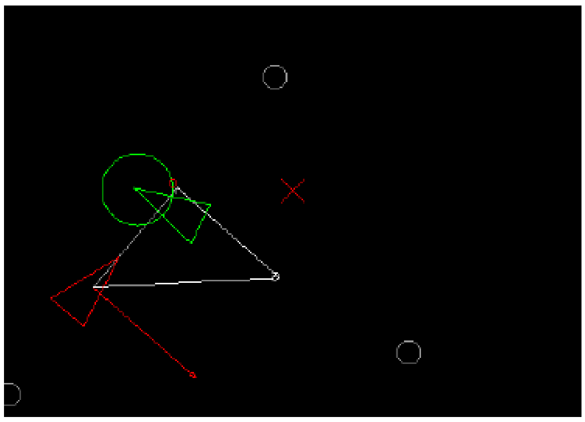
Output we found out:

I tried running it multiple times but for the hunter wasn't chasing after the prey, at first I was confused then I tried changing the radius from 10 to 100 of the hunter multiple times until I got it chasing after the prey but the hiding object got in between them allowing the prey to escape, check below for outputs captures.

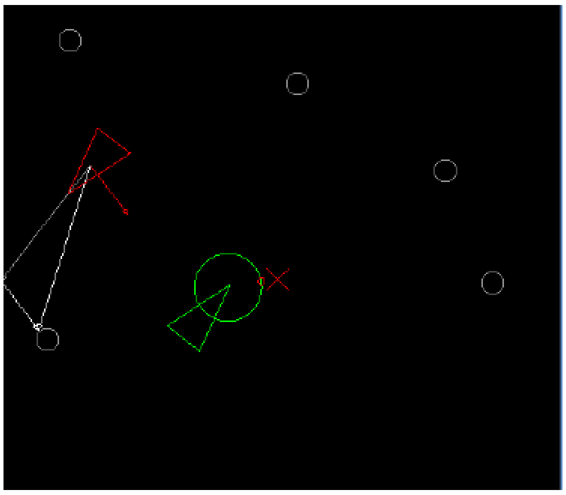
Problems I discovered:

I noticed because the hunter was designed to avoid hiding objects it was doing sharp turns and sometimes it even went out of screen to avoid the hide objects, which shouldn't happen, so that was one of the problems I found in the game.

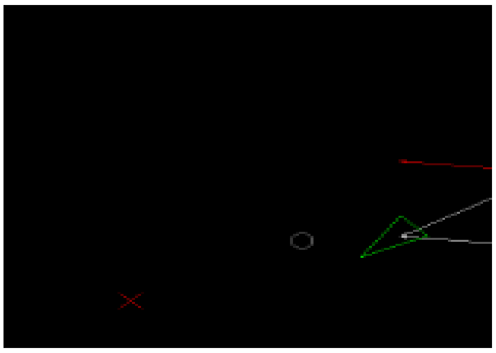
1. This shows that the hunter was wandering around the prey without interacting with it



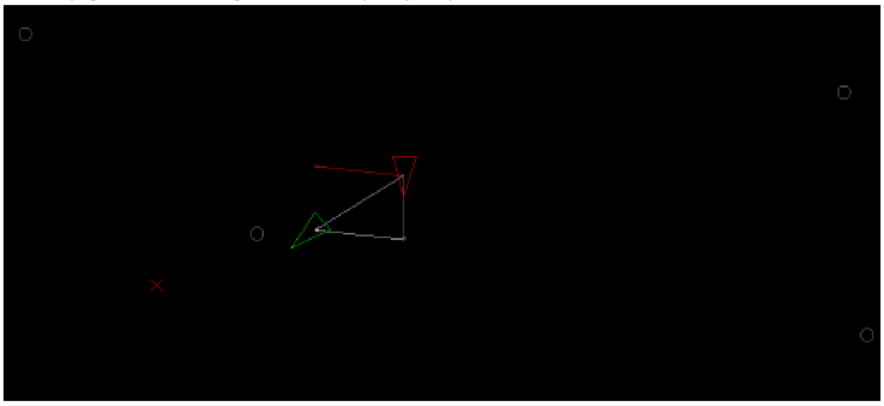
2. Still wandering around but a bit far now.



3. Here the hunter and prey couldn't be together because of the screen wrapping over



4. Finally got it chasing after the prey (..yeaaaa).



The things I learnt, when attempting this task:

- With the use of Emergent behavior I felt that the AI was marking more of complex patterns than the one's assigned which in turned make the AI more creative than the target player and hence with the use of Hiding behavior, we use it both ways around like for AI player the advantage could be to close in on enemy using the Hiding Objects which are the objects between it and the target, which can make the target player patrolling fell confused with AI adaptive hiding approach to close in, plus without choosing the same hiding spots the AI builds up more and more complex patterns for it to move behind inanimate objects.
- The advantage of this behavior can be quite apparent as for sneak assassination attempts made by the AI player on target without being in sight while the disadvantage is that the more complex the hiding pattern the difficult it gets for target player to counter it, which in turn would want the player to give up since it might seem impossible to adapt and counter it.