Spike Outcome Report

Number: 03

Spike Title: Tactical Steering

Personal: Peter Argent (7649991)

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:

- The code completed in Spike 2
- Visual Studio 2017 with Python 3 installed
- Pyglet Documentation here: http://pyglet.readthedocs.io/en/pyglet-1.3-maintenance/
- Help from peers.

Tasks undertaken:

- Create a class for an object for the prey agents 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)
```

- Create child classes from the agent class for the prey and hunter agents. prey.py and hunter.py

- For the Prey Class, we need Hide, Runaway, Flee, and getHidingPosition functions.

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

    self.BestHidingSpot = None
    hun = hunter
    # check for possible hiding spots
    for obj in objs:
        HidingSpot = self.getHidingPosition(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)
```

```
def runAway(self, pursuer, delta):
    # flee from the estimated future 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)
```

```
    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 is '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)
        return Vector2D()

    def getHidingPosition(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
```

- Also make sure to have a overridden calculate method with the following: super().calculate is calling the base classes calculate method

```
def calculate(self, delta):
    if self.mode == 'flee':
        force = self.runAway(self.world.hunter, delta)
    elif self.mode == 'hide':
        force = self.hide(self.world.hunter, self.world.hideObjects, delta)
    else:
        force = super().calculate(delta)
    return force
```

- For the Hunter Class we require a pursuit method and an overridden calculate method:

```
def calculate(self, delta):
    if self.mode == "pursuit":
        force = self.pursuit(self.world.agents, delta)
        force.truncate(self.max_force)
        accel = Vector2D(force.x / self.mass, force.y / self.mass)
        self.acceleration = accel
        return accel
    else:
        return super().calculate(delta)
    return super().calculate(delta)

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)
```

The next thing to do is to modify the main method in main.py to create prey in the world.agents list and a hunter in the world.hunter, and then add 5 hideObjects for the prey to hide behind in world.hideObjects

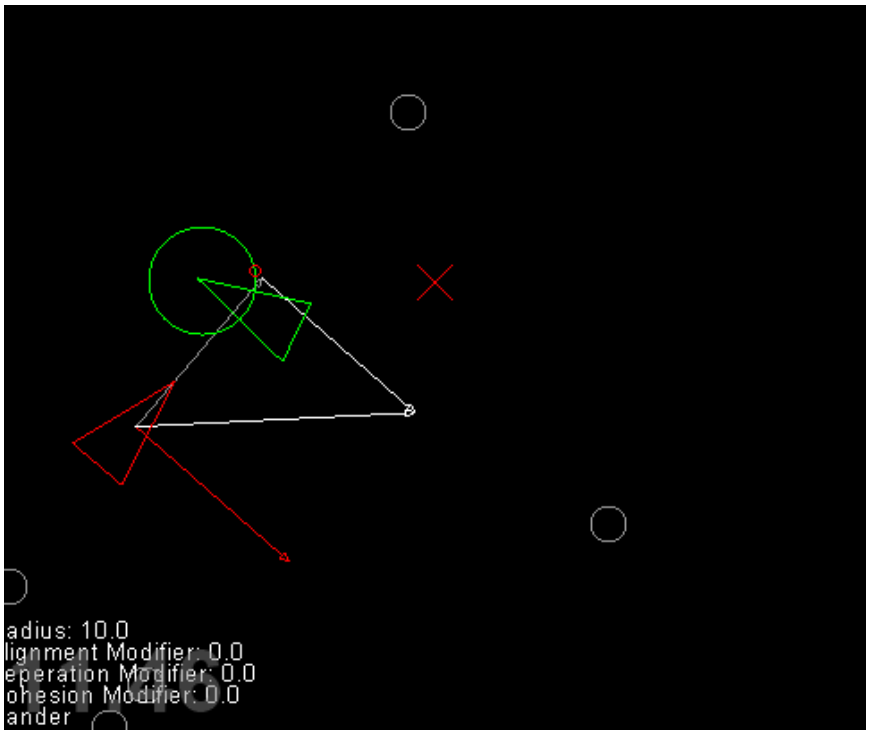
```
world.hunter = Hunter(world)
world.agents.append(Prey(world))
# add HideObjects
for _ in range(5):
    obj = HideObject(world)
    world.hideObjects.append(obj)
# unpause the world ready for movement
```

What we found out:

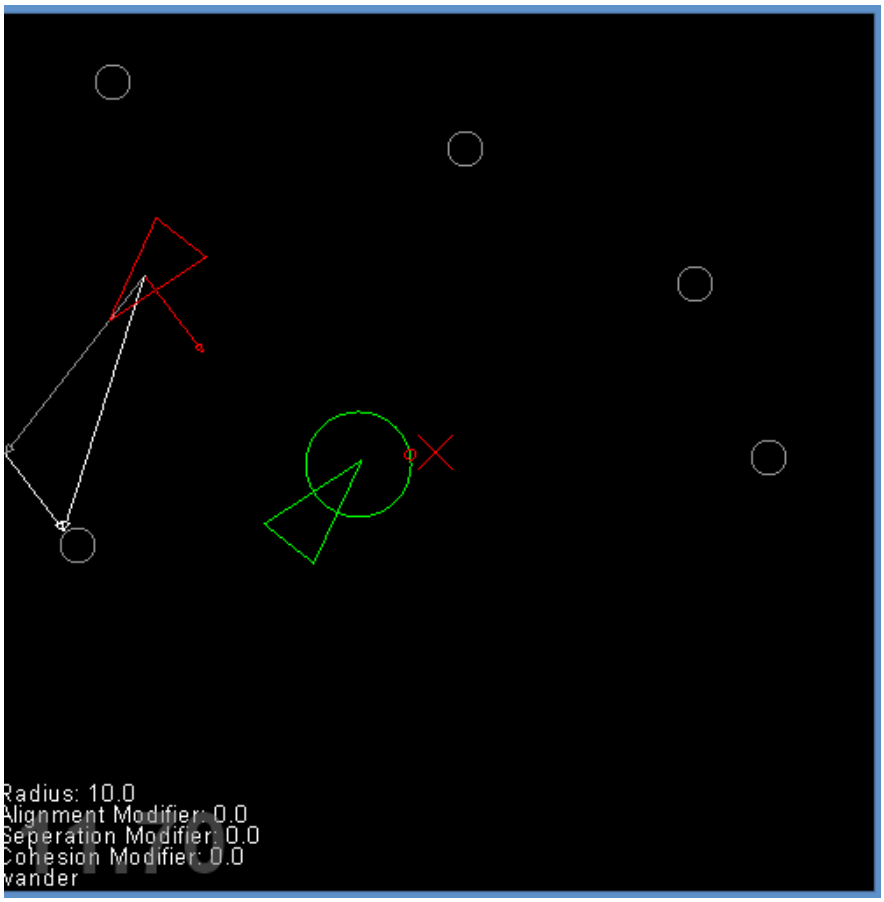
* #### Graph/Screen shot/outcome list/notes

At first, I had the Radius on the Hunter at 10 and the hunter would generally wander without really interacting with the Prey, I changed it to 100 and it would tend towards the prey but still mostly just wander around, at 200 it really chased after the prey, usually losing the prey when it did a sharp turn around a hidingObject or when it went to the other side of the screen with a wraparound.

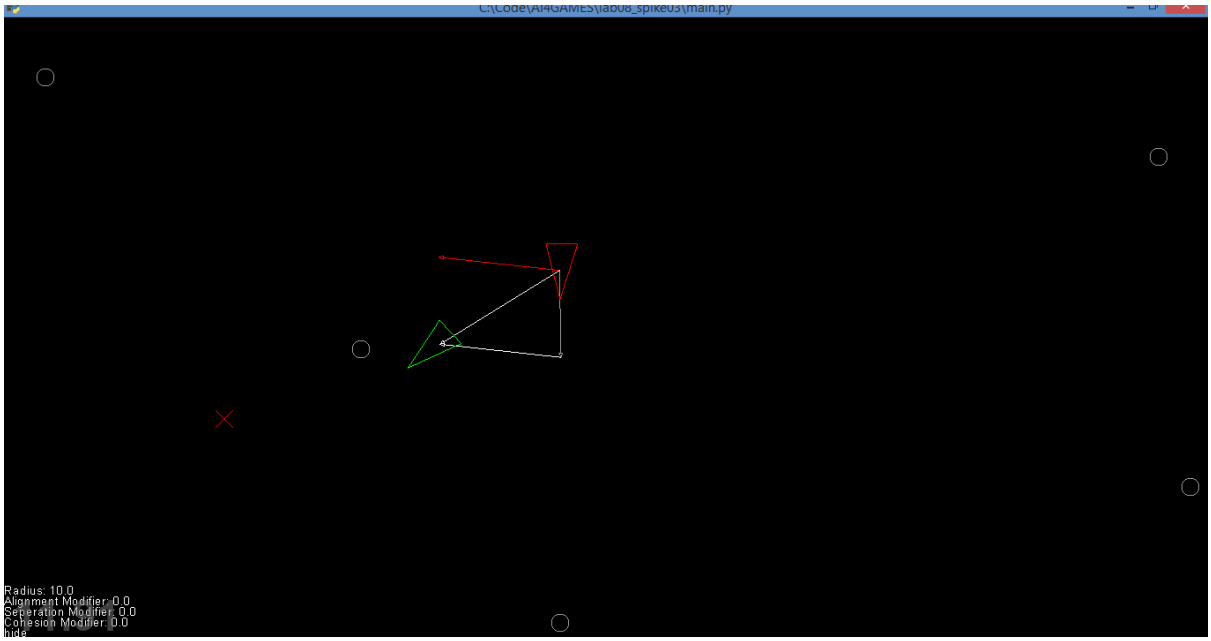
Radius 10



Radius 100



Radius 200



Open issues/risks [Optional]:

- Issue: Agents (Both Hunter and Prey) phase through the hidingObjects.

Recommendations [Optional]:

- Spike should be done to give all agents a steering force away from objects after all other forces are calculated. This will relieve the phasing through hidingObjects Issue