Spike Outcome Report

Number: 8
Spike Title: Tactical Steering (Hiding)
Personal: Michael Williams (7668481)

Goals:

- 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
- Sample Lab Code
- Lecture Material

Tasks undertaken:

1. The first step for this spike was to study the lecture material and work out what was required for the spike.

2. The next step was to come up with a rough plan on how the system would work and what each agent would be required to do (ie. one hunts and one runs away).

3. After determining the plan, we started of with the code from the previous lab and duplicated the agent code so that we had a hunter and prey. From this point, we refactored the classes to strip out behaviours not required.
   We made the hunter very simplistic so it’s only remaining behaviour was “seek”, we reduced the speed to make it slower than the prey and made it red so that it was visually easier to see.
   For the prey, a similar thing was done, but the behaviours it was left with were “seek” and “arrive” and the colour and speed were unchanged.

4. The next step was to determine the best location to hide, this was done simply based on the closest hiding spot to the prey. That location was then put into the following function to determine the exact spot to “arrive” at.

```
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
    #use the perpendicular vector for better behaviour
    ToObj = PointToVector2D(obj.pos - hunter.pos).normalise().perp()
    # scale size past the object to the hiding location
    return (ToObj * DistAway) + obj.pos
```

5. The next step was to test the implementation and work out were it failed. It fell short mainly when the hunter was between the prey and the closest location. In order to deal with this the second last line in the function above was added.

What we found out:
We found out that there are a multitude of ways to determine the “best” location for hiding, through tactical analysis. The one that we decided to implement was simply the closest location. This worked pretty well most of the time but we found certain situations where this would fail miserably. After fixing this we achieved much more favourable results which ultimately looked less scripted and more realistic. After completing this, we believe that depending on the situation there could be more applicable tactical analysis implemented. For example, if there were multiple hunters, selecting the location where the least number of hunters were may prove more effective.

Screenshots