SPIKE REPORT SPIKE 8

Jake Lucic

2103435 AI for Games

Number: Spike 8
Spike Title: Tactical Steering
Personal: Jake Lucic (2103435)

Context:

Tactical analysis by agents can be used to build on basic steering behaviours to create intelligent behaviour. An agent that is being pursued by another can make use of tactical analysis to identify an object in the game environment with which to hide behind and out-of-sight.

Knowledge/Skill Gap:

The developer needs to know how to give an agent the ability to find and seek good hiding locations using tactical analysis.

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:

In this task, the technologies used are listed below:

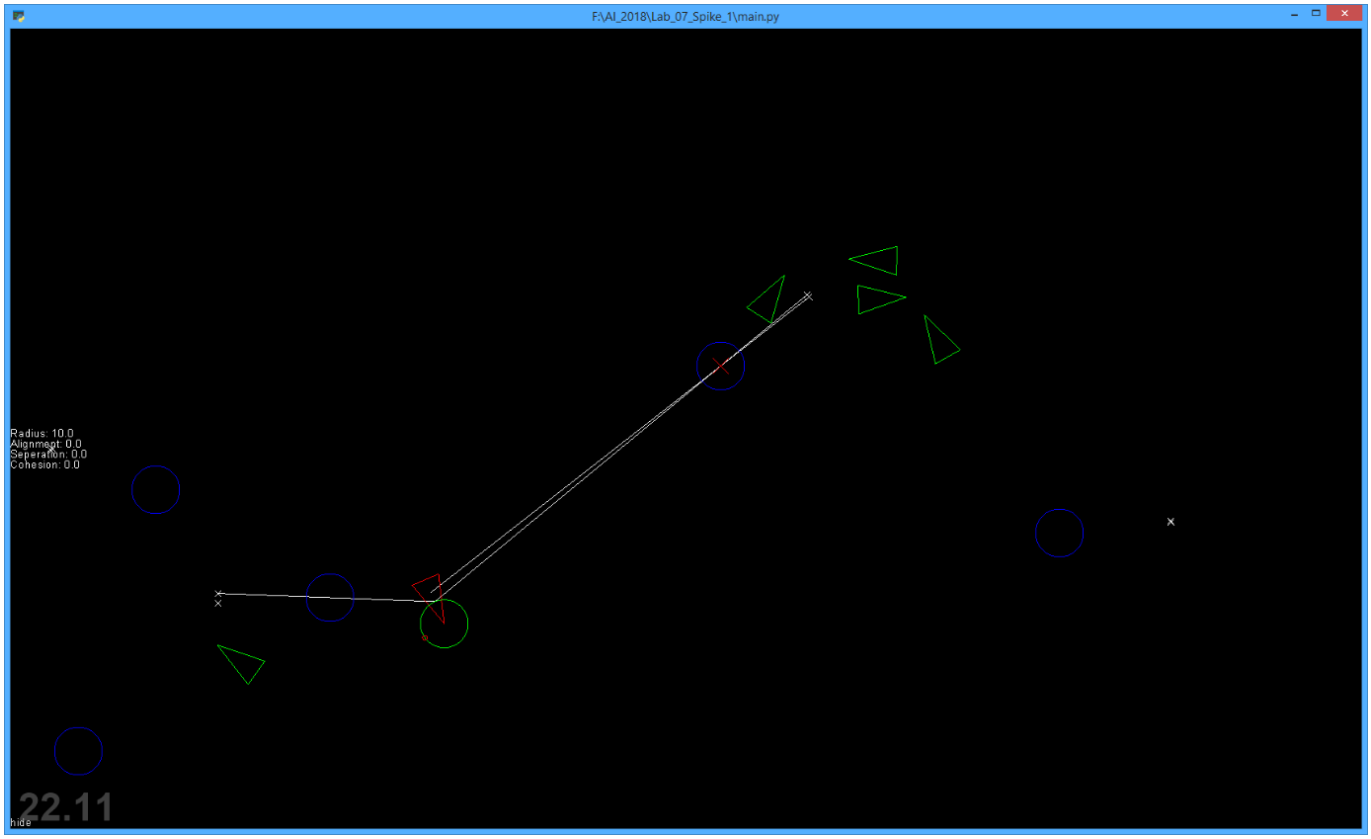
- Simple Code base, the not-yet-functional code that required fixing.
- Python IDLE v3.6.4 / Python language / PyCharm
- Path Following, Wandering Lecture
- Tactical Action Selection and Predictive Shooting Lecture

Tasks undertaken:

- Read the code base and understand what each function does
- Study the different key words and variable names
- Follow the lectures on Path Following, and Tactical Action Selection
- Approach Task using similar process below

Process:

I approached this task by creating objects that the Agent could hide behind. I followed that they should be made as circles. The reason as to why it should be a circle is the fact that to get your Agent to hide behind it, having a radius and a centre point, makes the math to hide behind the obstacle a lot easier.



In this image, you can see the dark blue circles, these are the obstacles that the hiding Agents (Green) hide behind. The hunter agent is the red agent, and it is simply on wander mode.

The white lines stemming from the Hunter, points to target hiding positions, there is one for each agent.

As the Hunter and the Hiders move, the hiding position also moves.

The Hiders look for the closest obstacle to themselves, and then by using the function getHidingPosition, this sets a distance from the radius of the circle. This is then updated every update cycle so it constantly returns the “hidden” side of the obstacle from the Hunter Agent.

Using Slight Tactical Evaluation as an approach to this task, my selection was for each of the Hiders to select the closest obstacle to them. Although, I also found a way to get all of the Hider Agents to hide from the furthest obstacle away from the Hunter Agent. This is explained in full in the image on the next page

```
def GetHidingPosition(self, hunter_pos, obst):
    distFromBoundary = 110
    distAway = obst.radius + distFromBoundary


    toObj = (obst.pos - hunter_pos).normalise()

    return (toObj * distAway) + obst.pos

def hide(self, hunter):

    DistToClosest = float('inf')
    BestHidingSpot = Vector2D()

    to_target = hunter.pos - self.pos
    panic_range = 60
    dist = to_target.length()
    if dist < panic_range:
        self.visible = False


    for obst in self.world.obstacle:
        hidingSpot = self.GetHidingPosition(hunter.pos, obst)
        hidingDist = Vector2D.distance_sq(hidingSpot, self.pos) #change to self.pos for closest obst search, and change vector to positive
        egi.cross(hidingSpot, 4)
        if hidingDist < DistToClosest:
            DistToClosest = hidingDist
            BestHidingSpot = hidingSpot

    if BestHidingSpot:
        egi.line_by_pos(hunter.pos, BestHidingSpot)
        return self.arrive(BestHidingSpot, 'fast')
    return self.flee(hunter.pos)
```

Here is the code for GetHidingPosition and Hide Agent mode

Extension

I also was able to get the Hiding agents be able to be swallowed by the Hunter Agent simply by creating a collision circle, comparing if the Agent type was a Hunter or a Hider, then if they entered that circle, they would simply not be visible (This is also visible in the hide mode function. Although if I had to approach this task again, especially to optimise – I would just remove them from the list, so that they would not update/render in the next cycle.

Added Controls:

H: Adds the hunter into the world. Please place only one.

i: Adds Hider agents.

0: Sets the Hider Agents into Hide mode. Please have a hunter (H) before selecting this option