SPIKE REPORT SPIKE 14

Jake Lucic

2103435 AI for Games

Number: Spike 14
Spike Title: Soldier on Patrol
Personal: Jake Lucic (2103435)

Context:

Agents in games can take advantage of layered state-machine designs with high-level modes of behaviour and lower-level step behaviour working together.

Knowledge/Skill Gap:

Developers need to be able to create layered finite state machine models for agents to use.

Goals:

Create a “soldier on patrol” simulation where an agent has two or more high-level FSM modes of behaviour and low-level FSM behaviour.
The model must show (minimum)
• High level "patrol" and "attack" modes
• The "patrol" mode must use a FSM to control low-level states so that the agent will visit (seek/arrive?) a number of patrol-path way points.
• The "attack" mode must use a FSM to control low-level fighting states. (Think “shooting”, “reloading” – the actual states and transition rules are up to you.)

Technologies, Tools, and Resources used:

In this task, the technologies used are listed below:

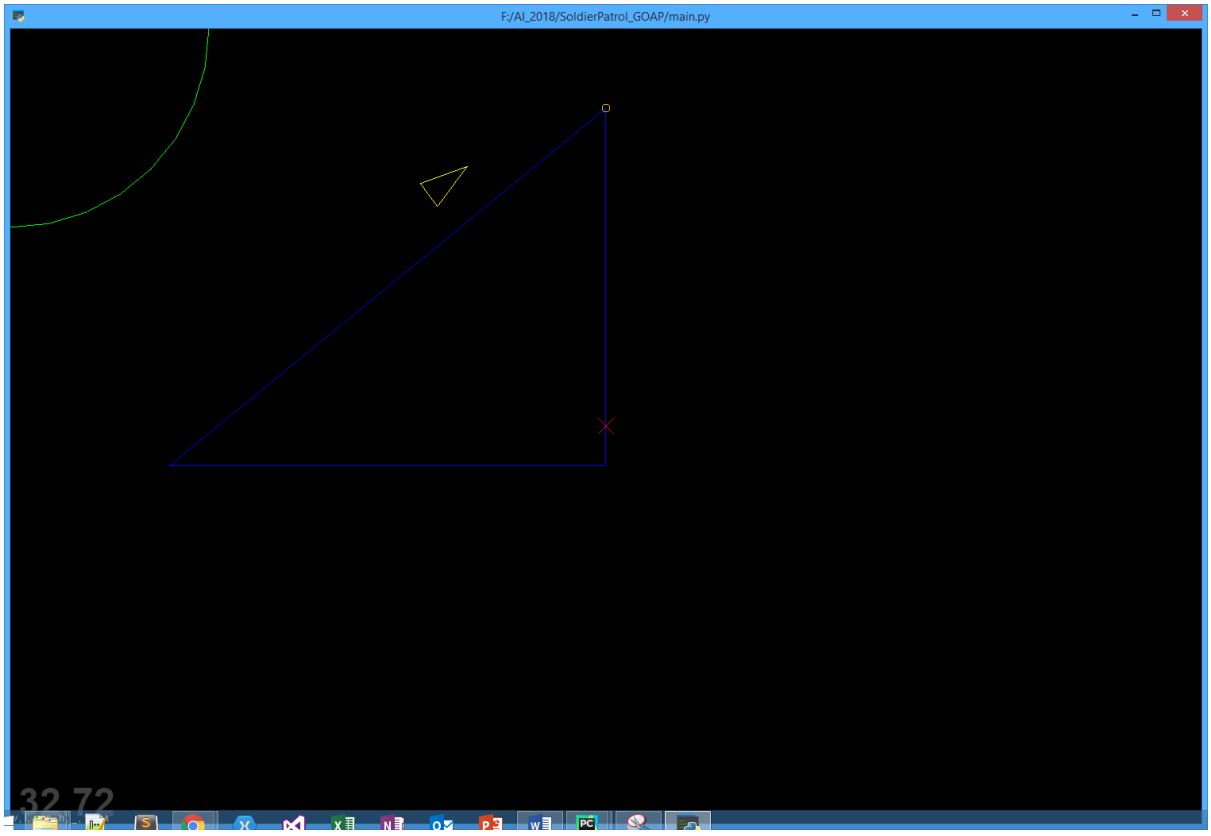
- Simple Code base, the not-yet-functional code that required fixing.
- Python IDLE v3.6.4 / Python language / PyCharm
- Tactical Action and Predictive Shooting Lecture
- Path Following, wandering lecture
- Autonomous Steering Lecture
- FSM Lecture

Tasks undertaken:

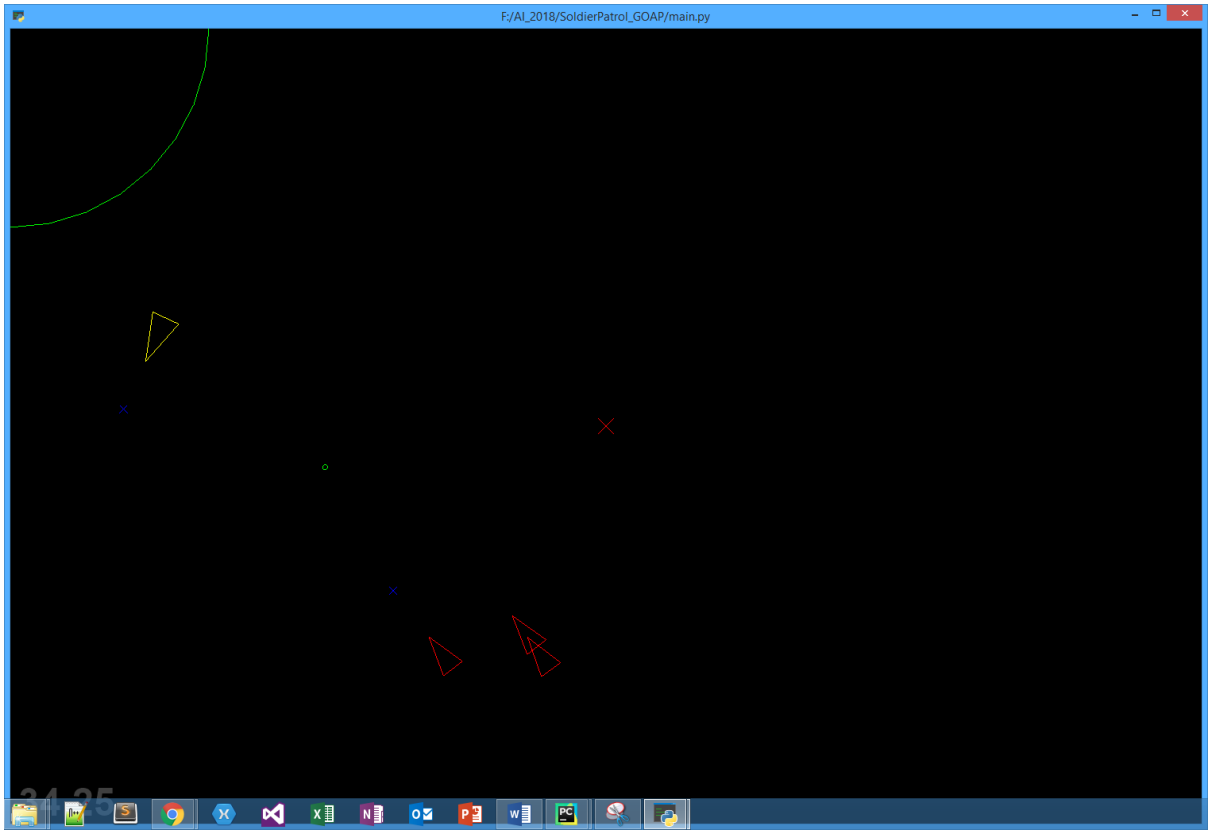
- Reassess the code base and understand what each function does
- Study the different key words and variable names
- Follow the Lectures listed in the resources column
- Plan what my Agent is required to do
- Follow the process below

Process:

I planned this task by imagining what I wished my agent to do. I knew I wanted it to follow a path, so I used my adjusted code from the Marksman spike so I could set a consistent path for the Agent's Patrol mode. After completing this step, I had a beginning state.



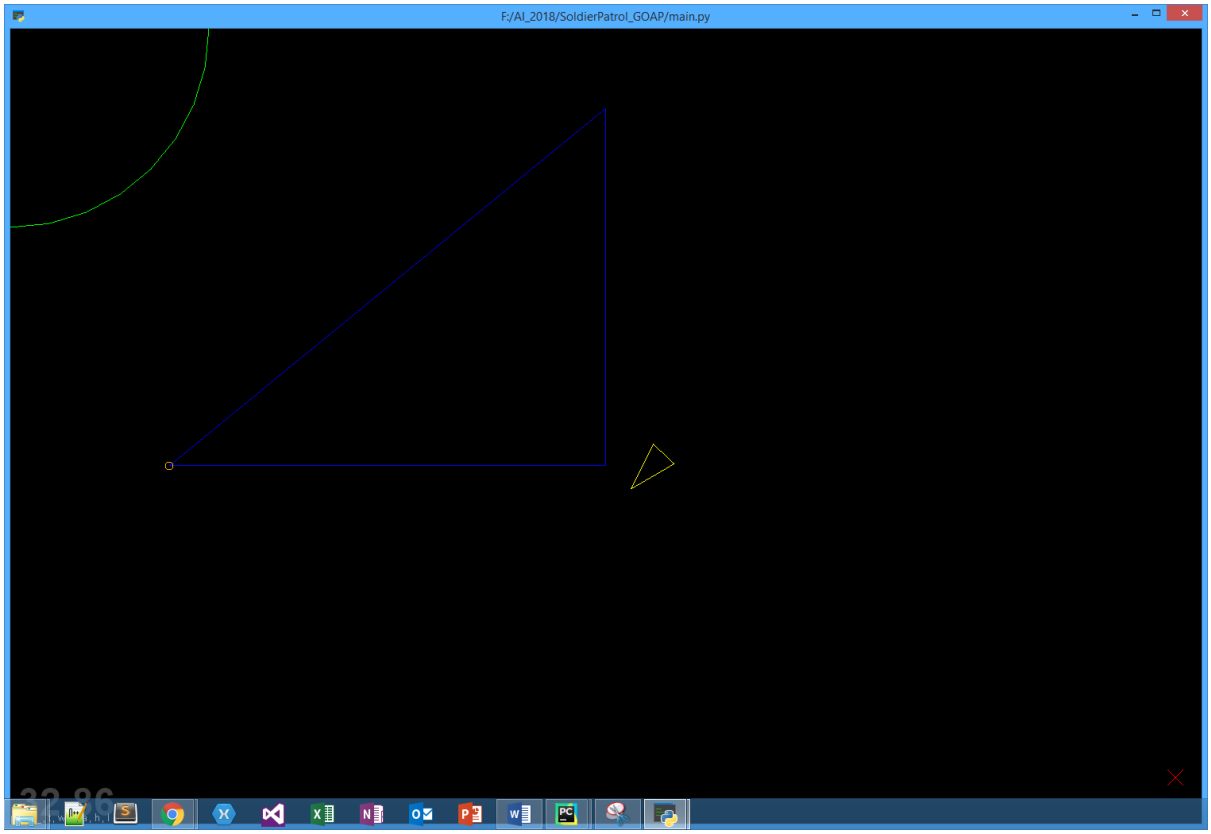
The blue line is the agents patrol path. It remains on this path until it sees an enemy.



Once the agent sees the enemy agents(red), it drops its state of patrol, and then goes into attack mode. The green bullet used is the same as the marksman spike, just applied to a moving agent.

I extended the attack mode to decide that when it is in the attack mode, read how many agents there are and if there are more than 1, flee and attack. If there is only 1 agent, seek and attack. This incorporates knowledge from the FSM labs, Tactical Analysis labs, Marksman and predictive shooting labs, Steering labs.

As you can see, each agent has a blue marker in front of them, this is their predicted position which is calculated in the same manner as the predictive shooting lab.



After the enemy agents have been dealt with, the Main agent returns to its patrol mode to seek its next waypoint.

This behaviour makes use of a Hierarchical FSM, where its behaviour has SubStates.

Patrol Mode > Seek next waypoint

Attack Mode > Flee and shoot/reload

> Seek and shoot/reload

```
#Returns actual target as object, bullet then takes .pos
#Needed to pass object so that uodate could call this method to acquire predicted pos
def lookforenemy(self):

    if self.is_enemy == False:
        if len(self.world.enemies) > 0:
            target = self.world.enemies[0]
            self.mode = 'seek'
            self.my_target = target
            for agent in self.world.enemies:
                if (self.pos - agent.pos).length() < (self.pos - target.pos).length():
                    target = agent
                    self.mode = 'flee'
                    self.my_target = agent

            return target

        else:
            self.mode = 'follow_path'
            target = self.world.targett
            return target

    else:
            target = self.world.agent
            return target


def shoot(self):
    if not self.is_enemy:
        self.world.bullet.append(Bullet(self.world, self.predictedtarget, self.pos.copy(), self.gun))
```

This is the code that is used by the Main Agent. It determines how many enemies there are and chooses a mode of attack. If there are more than one enemy agent, it chooses the closest enemy to its position and it shoots at its predicted position, whilst in the flee mode. If there is only 1 enemy agent in an immediate area, it will seek out that enemy agent and attack. Otherwise, it will return to a patrol mode.

```
    if self.mode == 'seek' or self.mode == 'flee':
        if self.reloadtimer > 0:
            self.reloadtimer -= delta
        else:
            self.shoot()
            self.reloadtimer = 2
```

This is my reload timer, which checks if it is in a current state, and then proceeds to shoot/reload in the update method.