Spike: 5
Title: Agent Marksmanship

Author: Ryan Chessum - 102564760

Goals / deliverables:
Create an agent targeting simulation with:
(a)an attacking agent (can be stationary),
(b)a moving target agent (can simply move between two-way points), and
(c)a selection of weapons that can fire projectiles with different properties.

Be able to demonstrate that the attacking agent that can successfully target (hit) with different weapon properties:

(a) Fast moving accurate projectile.
(b) (Rifle) (b)Slow moving accurate projectile. (Rocket)
(c) (c)Fast moving low accuracy projectile (Hand Gun)
(d) (d)Slow moving low accuracy projectile (Hand grenade)

Technologies, Tools, and Resources used:

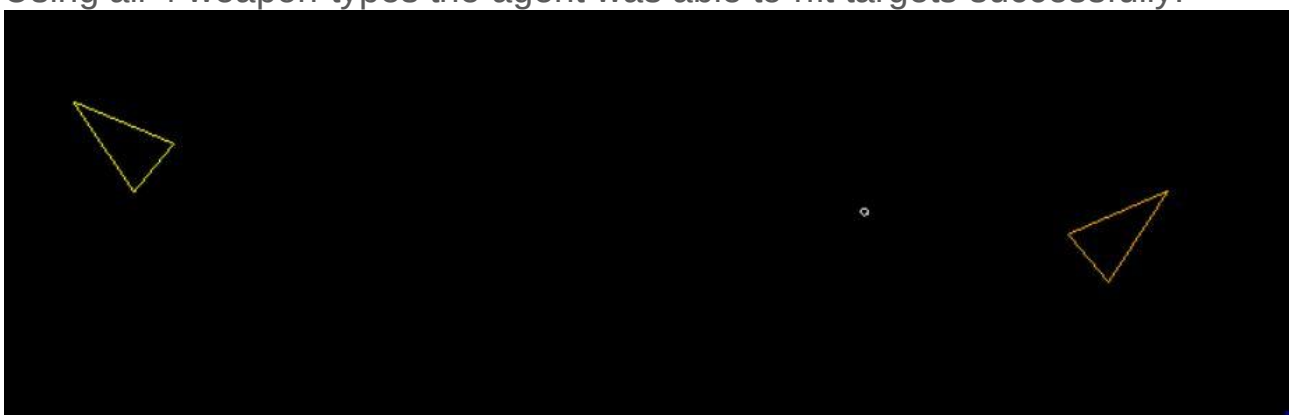
- Visual Studio Code
- Python 3.8.0
- Pyglet
- Lecture Notes
- Code from Labs

Tasks undertaken:

- Create 2 agent subclasses, one for the marksman agents and one for the moving target agents
- Give the moving target agent a simple movement behaviour
- Create a class for projectiles, set it's update to move in a constant direction over time
- Override the marksman's update function so that it remains stationary and instead shoots projectiles
- Add the different weapon names to a dictionary so that the marksman's weapon can be changed with a keypress in main
- Add different weapon statistics to be used in functions
- Add a function for the marksman to shoot bullets
- Add a function to the marksman that allows it to predict the future position of a target to aim at
- Add the update functions for all the new agent types that aren't stored in the agents list in the world update function

What we found out:

Using all 4 weapon types the agent was able to hit targets successfully.



The target behaviour is set to a random path and once it finishes that random path it generates a new one to follow. I found this showed off the aiming predictions well as the targets constantly change speed and directions.

```
def follow_path(self):
    #If heading to final point (is_finished?),
    if self.path.is_finished():
        self.randomise_path()
    # #     Return a slow down force vector (Arrive)
    elif (self.path.current_pt() - self.pos).length() <= self.waypoint_threshold:
    # #   If within threshold distance of current way point, inc to next in path
        self.path.inc_current_pt()
        return self.seek(self.path.current_pt())
       # return self.seek(self.path.current_pt())
    # #   Return a force vector to head to current point at full speed (Seek)
    return self.seek(self.path.current_pt())
```

These are the values used for the different statistics of each weapon.

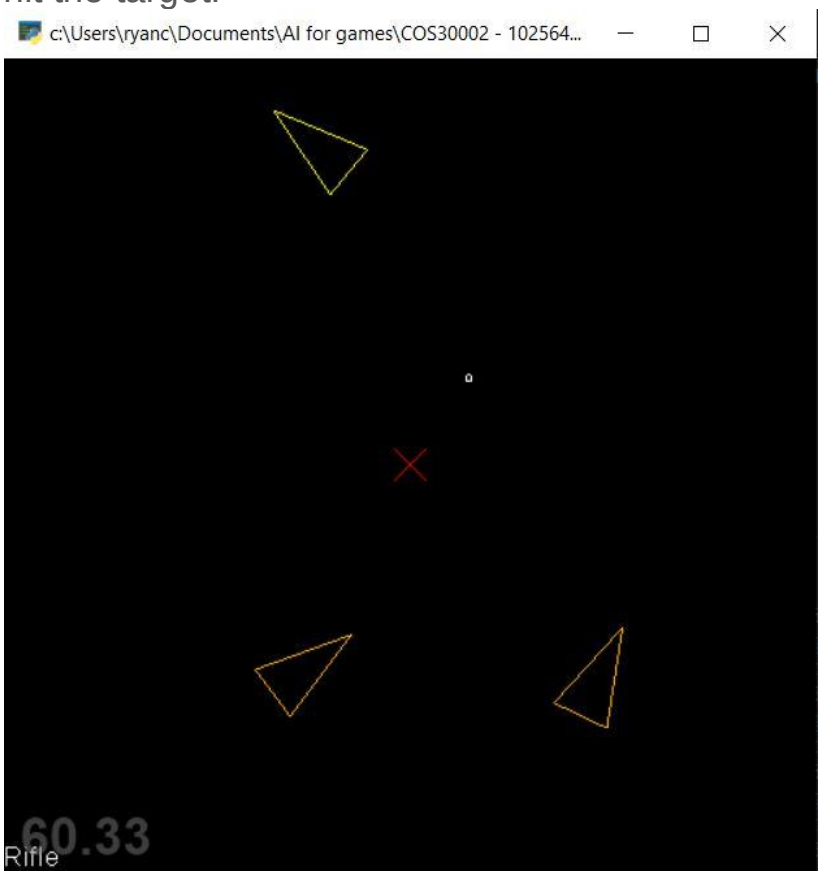
```
WEAPON_TYPES = {
    KEY._1: 'Rifle',
    KEY._2: 'Rocket Launcher',
    KEY._3: 'Hand Gun',
    KEY._4: 'Grenade'
    }

WEAPON_COOLDOWN = {
    'Rifle': 1.0,
    'Rocket Launcher': 1.2,
    'Hand Gun': 0.2,
    'Grenade': 1.0
}

PROJECTILE_SPEED = {
    'Rifle': 500.0,
    'Rocket Launcher': 200.0,
    'Hand Gun': 350.0,
    'Grenade': 150.0
}
```

The marksman is able to predict the future position of a target and aim at it using this function.

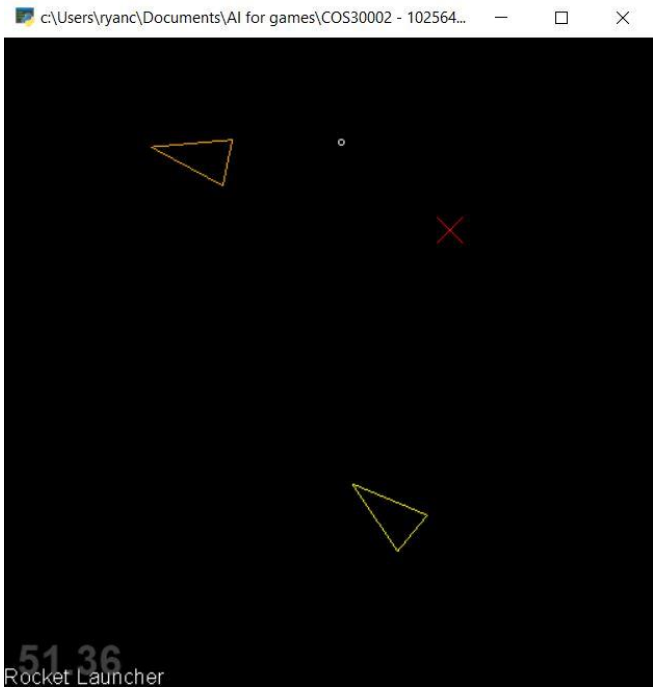
```
def predict(self, rp, rv):
    #solve using quadtatics
    a = rv.dot(rv) - PROJECTILE_SPEED[self.weapon] * PROJECTILE_SPEED[self.weapon]
    b = 2.0 * rv.dot(rp)
    c = rp.dot(rp)

    d = b * b - 4.0 * a * c

    if (d > 0.0):
        return 2 * c / (sqrt(d) - b)
    else:
        return - 1
```

It finds a target point using a quadratic equation made from the projectile speed, relative position and velocity of the target.

The sniper was the fastest and most accurate weapon but has a lower cooldown. Due to the speed of the projectile the marksman is usually able to hit the target.



The rocket launcher fires a slow projectile but has great accuracy like the sniper rifle. The marksman can usually hit a target with it but it struggles when the target is too far away as often the target has changed directions by the time the projectile reaches the target point.



The hand gun has low accuracy but low cooldown and fast bullets. The shots sometimes miss due to the inaccuracy but the marksman is still usually able to hit the target. The fast cooldown also usually means that the target gets hit with the next shot if it misses anyway.

The hand grenade has a low cooldown, low projectile speed and is inaccurate. Despite this, the marksman can still usually hit targets with it in close range.

This is the function the agent uses to shoot it’s targets.

```
def shoot(self, target):
    #copies
    mksmn = copy.deepcopy(self)
    trgt = copy.deepcopy(target)
    #relative position
    rp = trgt.pos - mksmn.pos
    #relative velocity
    rv = trgt.vel - mksmn.vel
    dt = self.predict(rp, rv)

    if (dt > 0):
        crosshair = Vector2D()
        crosshair = trgt.pos + (trgt.vel * dt)
        crosshair = (crosshair - mksmn.pos).normalise()

        #startinf position
        sp = Vector2D()
        sp = mksmn.pos
        #speed
        spd = (PROJECTILE_SPEED[self.weapon])

        if self.weapon == 'Rifle' or 'Rocket Laucher':
            p = Projectile(sp, crosshair, spd, self.world)
            self.world.projectiles.append(p)
        else:
            #add inaccuracy
            crosshair + Vector2D(randrange(-100.0, 100.0), randrange(-20.0, 20.0))
            p = Projectile(sp, crosshair, spd, self.world)
            self.world.projectiles.append(p)
```

If a bullet comes in contact with a target it will destroy it.

```
if (self.pos - target.pos).length() < self.detection_radius:
    #for testing when it didn't delete
    #target.color = 'BLUE'
    self.world.targets.remove(target)
    if self in self.world.projectiles:
        self.world.projectiles.remove(self)
```

Keys:
A – spawn new target
R – Move marksman to random position
1 – Set Weapon to Sniper Rifle
2 – Set Weapon to Rocket Launcher
3 – Set Weapon to Hand gun
4 – Set Weapon to Hand grenade
P – Pause