Goals:

Create an agent targeting simulation with:
- an attacking agent (can be stationary),
- a moving target agent (can simply move between two-way points), and
- 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:
- Fast moving accurate projectile. (Rifle)
- Slow moving accurate projectile. (Rocket)
- Fast moving low accuracy projectile (Hand Gun)
- Slow moving low accuracy projectile (Hand grenade)

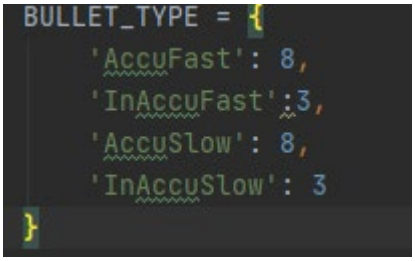
Technologies, Tools, and Resources used:

- Python IDE(PyCharm) with python 3 installed
- Piglet Documentation http://pyglet.readthedocs.io/en/pyglet-1-3-maintenance/
- Python 3 Documentation http://docs.python.org/
- The code from previous lab task
- Help from peers.

Tasks Completed:

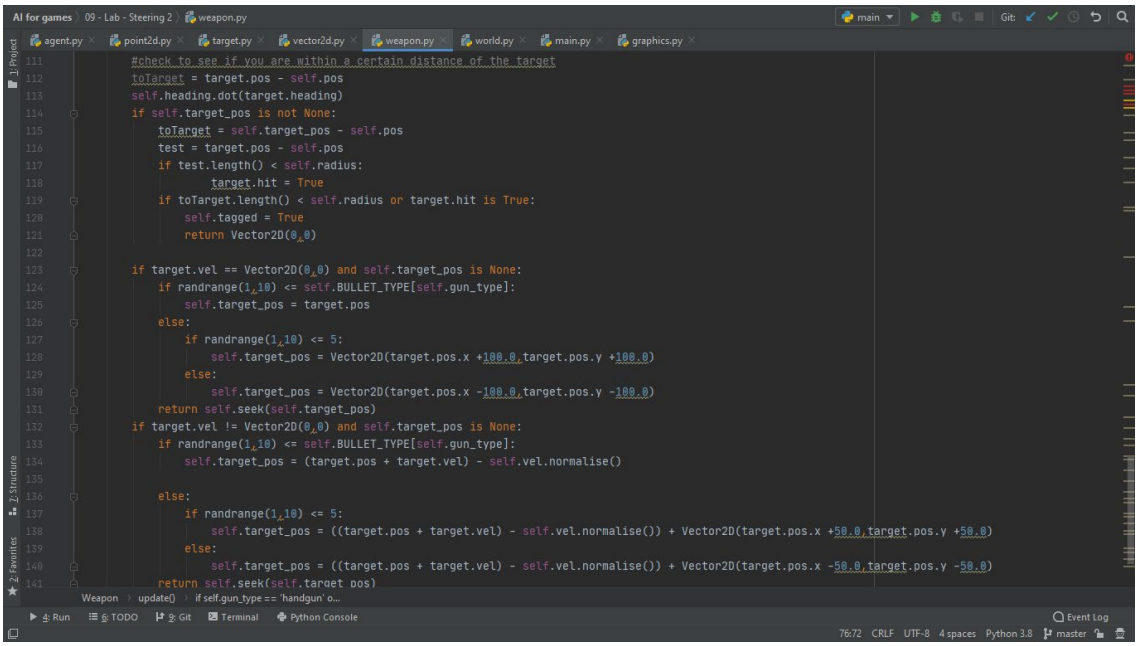
By utilizing the world.py from the Tactical Analysis;

1. Created a Target agent in Target.py which can:
    a. stay still or move between two random points.
    b. Start off with white color and then changes its color to red when hit.
    c. Target loses speed.
2. Made a new agent which targets the other agent in agent.py, to which I decided to restrict his x-axis movement while allowed Y position to be randomized within the vertical axis of the window, this was done so that agent can predict target locations from different points on the window.
3. Adding a weapon(bullet) in weapon.py, this was a bit tricky, I wasn't sure of what to do here so I made it this way:
    a. that the bullet would purse the target (which was quite unreasonable) I use the code from pursuit for the bullet movement.
    b. Check for the target to be stationary for the accuracy
    c. Different bullet types added like slow inaccurate (hand grenade), slow accurate(handgun), fast inaccurate(rocket) and fast accurate(rifle).



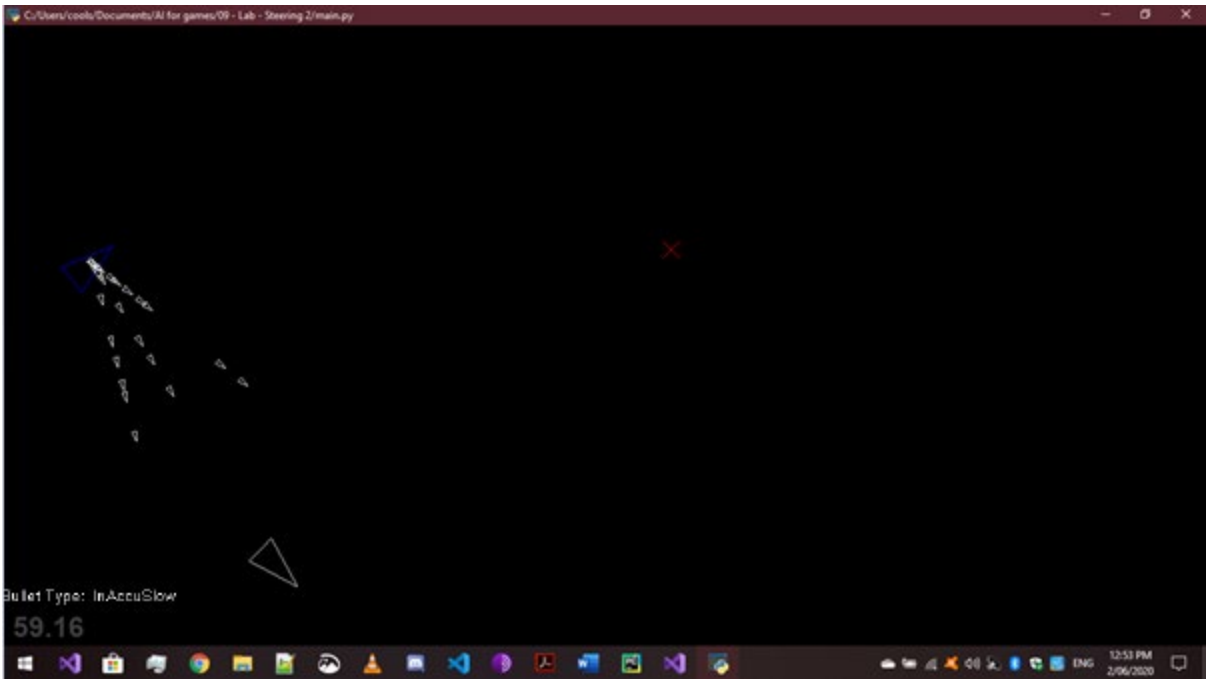
```
def update(self, delta):
    ''' update Bullet position and orientation '''
    force = self.calculate()
    # new velocity
    if self.gun_type == 'handgun' or self.gun_type == 'hand_grenade':
        self.vel += force * (delta/4)
    if self.gun_type == 'rifle' or self.gun_type == 'rocket':
        self.vel += force * delta
    # check for limits of new velocity
    self.vel.truncate(self.max_speed)
    # update position
    self.pos += self.vel * delta
    # update heading is non-zero velocity (moving)
    if self.vel.lengthSq() > 0.00000001:
        self.heading = self.vel.get_normalised()
        self.side = self.heading.perp()
    # treat world as continuous space - wrap new position if needed
    self.world.wrap_around(self.pos)
```

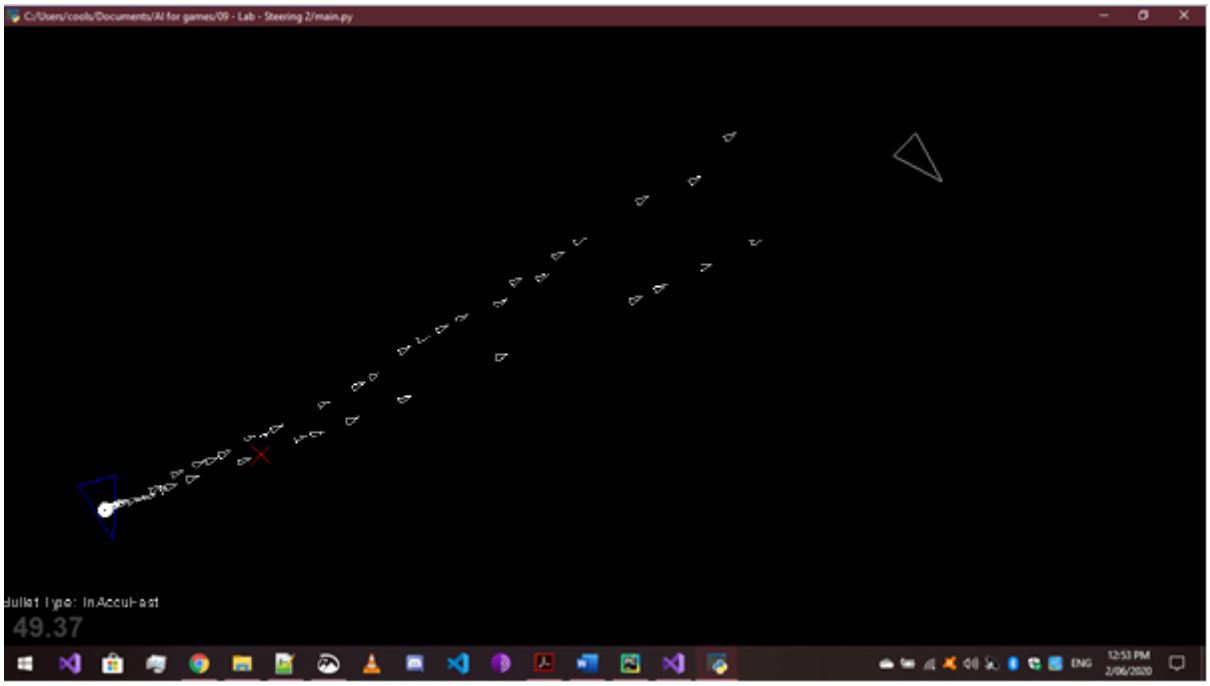
d. Used a rando numbers to decide on the value and type of shot fired, like if lower than j than accurate otherwise inaccurate (used the logic that whenever a fast shot is fired it will always miss a target hence inaccurate and vice versa).

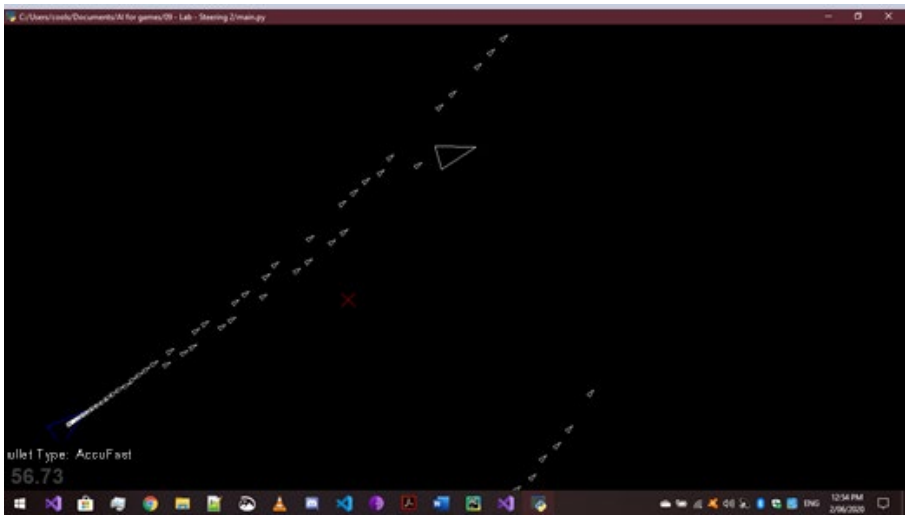


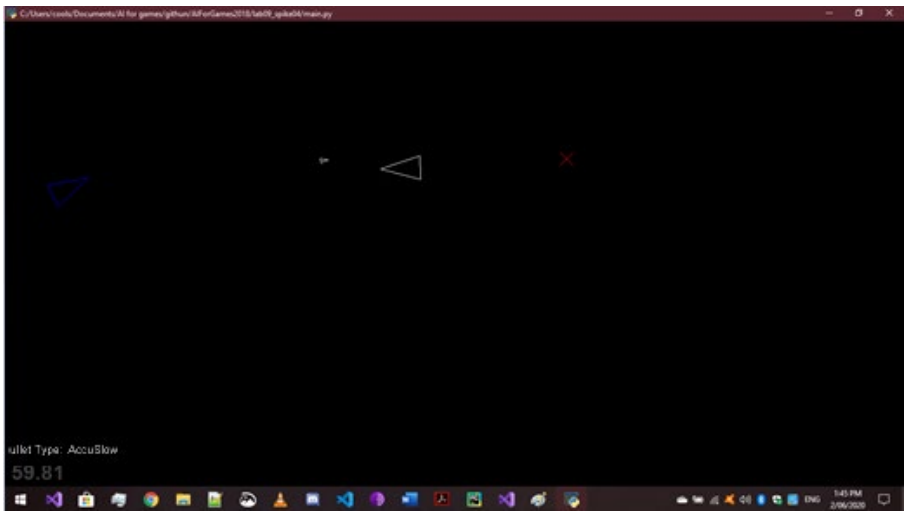
What we Observed:

Based on the below screen captures it can be observed that how each shot falls based on the modes









What we found out:

- ✓ Estimates for aiming positions.
- ✓ Inaccuracy and accuracy with shots.
- ✓ Projectiles and these all features can be combined to produce an assassination game, like Assassin's creed, I have to implement this in my Custom Project.
- ✓ Using Autonomous steering behavior to create predictive behavior and adapt to new situations/events, this sort of thing depicts realistic behavior which can suited for FPS non player characters.

Notes on Key notes used and what for:

1) Control Keys:
   a) F – Freeze target or make it still
   b) M – Allow target to move between two points
   c) S – Fire a single shot at target
   d) R – Fire at will (auto firing enabled)
2) Modes:
   a) Q – fast accurate(rifle)
   b) W - fast inaccurate(rocket)
   c) E – slow accurate(handgun)
   d) R – slow inaccurate (hand grenade)