Spike: Spike_9
Title: Agent Marksmanship

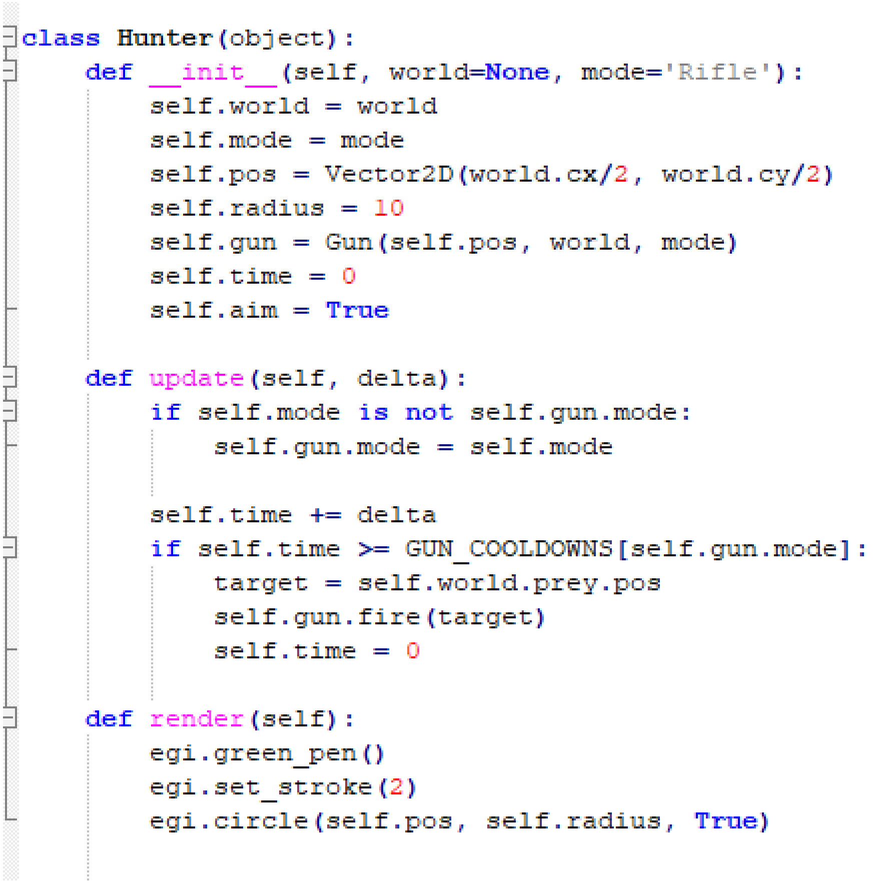
Author: Daniel Newman, 6449921

Goals / deliverables:
Develop a system that implements agent-targeting methods, predicting where the target is headed and using that to their advantage.

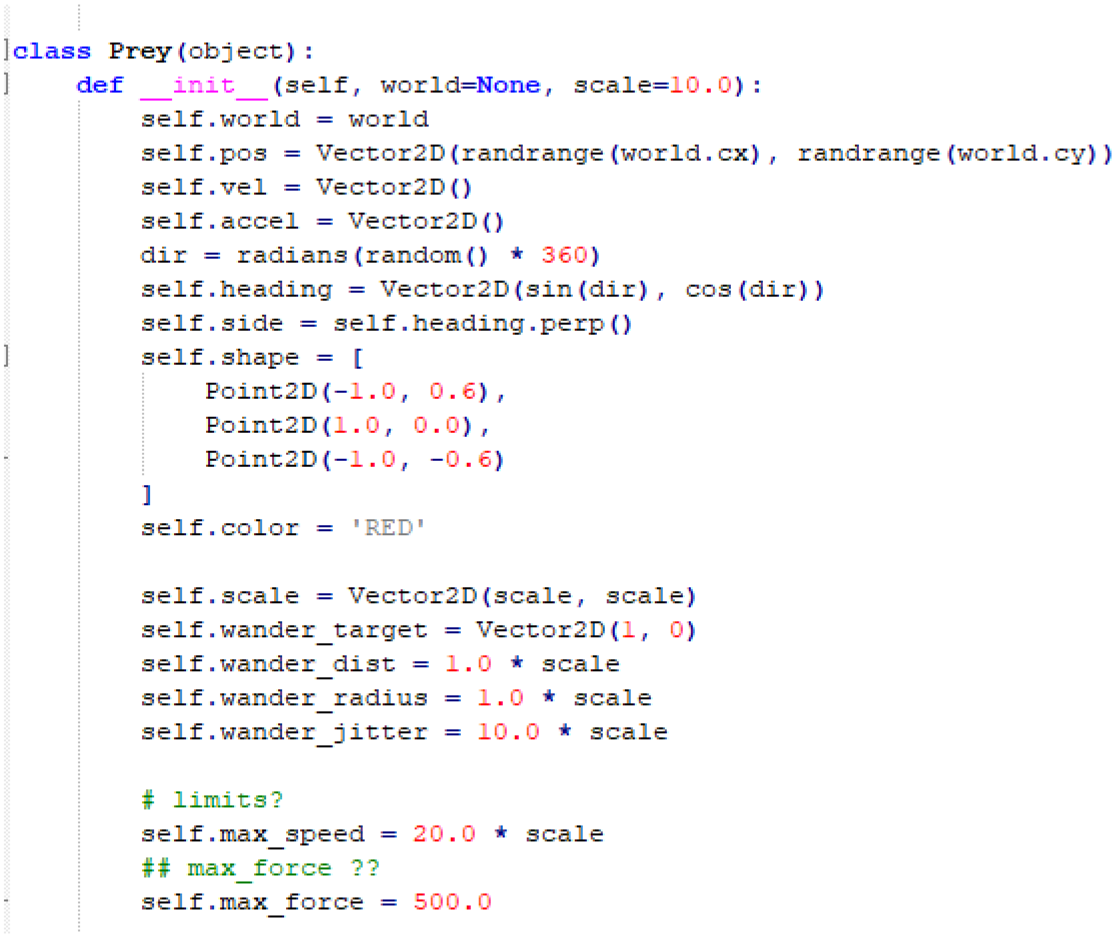
Technologies, Tools, and Resources used:

Python 3.6.4
Python compatible IDE

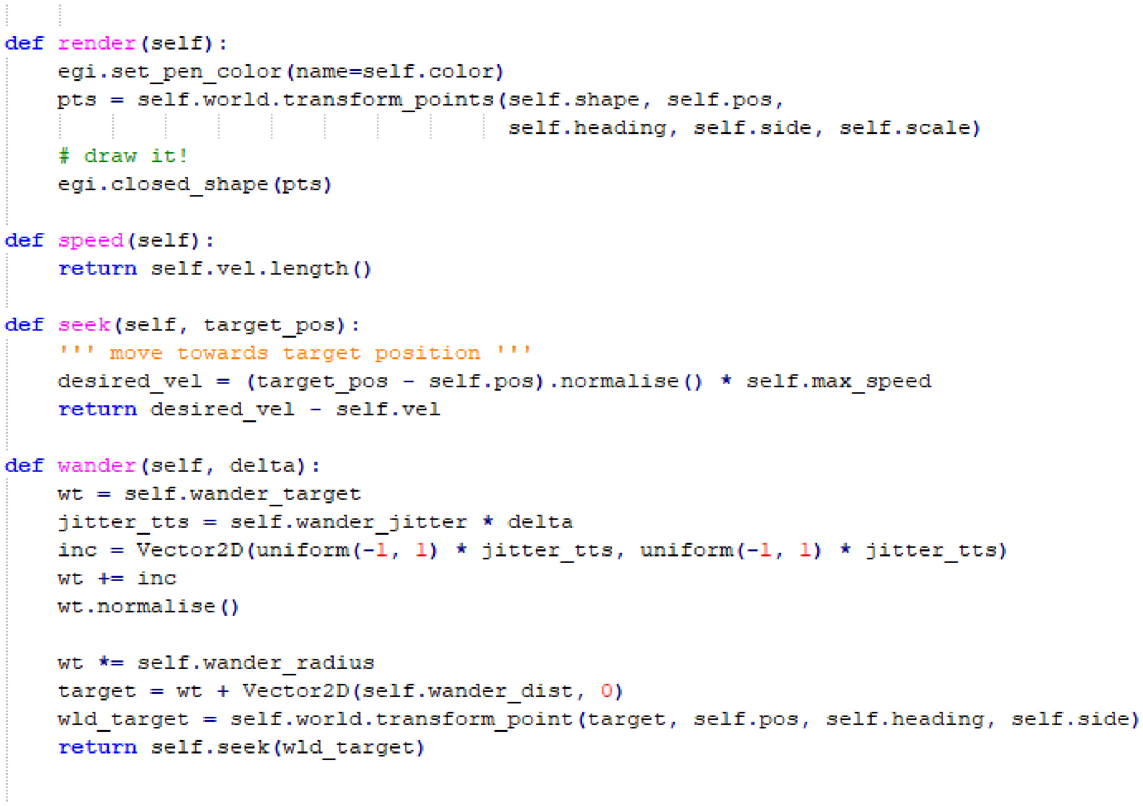
Tasks undertaken:
For this task we had to declare a few new classes to represent the different agent behaviours for this spike. We created a hunter class, who's goal is to fire upon our prey class. The prey class is designed to simply wander around the world and provide a target for the hunter class. This was achieved by using the wander code we had developed earlier to create autonomous boids.



```
class Hunter(object):
    def __init__(self, world=None, mode='Rifle'):
        self.world = world
        self.mode = mode
        self.pos = Vector2D(world.cx/2, world.cy/2)
        self.radius = 10
        self.gun = Gun(self.pos, world, mode)
        self.time = 0
        self.aim = True

    def update(self, delta):
        if self.mode is not self.gun.mode:
            self.gun.mode = self.mode

        self.time += delta
        if self.time >= GUN_COOLDOWNS[self.gun.mode]:
            target = self.world.prey.pos
            self.gun.fire(target)
            self.time = 0

    def render(self):
        egi.green_pen()
        egi.set_stroke(2)
        egi.circle(self.pos, self.radius, True)
```



```
class Prey(object):
    def __init__(self, world=None, scale=10.0):
        self.world = world
        self.pos = Vector2D(randrange(world.cx), randrange(world.cy))
        self.vel = Vector2D()
        self.accel = Vector2D()
        dir = radians(random() * 360)
        self.heading = Vector2D(sin(dir), cos(dir))
        self.side = self.heading.perp()
        self.shape = [
            Point2D(-1.0, 0.6),
            Point2D(1.0, 0.0),
            Point2D(-1.0, -0.6)
        ]
        self.color = 'RED'

        self.scale = Vector2D(scale, scale)
        self.wander_target = Vector2D(1, 0)
        self.wander_dist = 1.0 * scale
        self.wander_radius = 1.0 * scale
        self.wander_jitter = 10.0 * scale

        # limits?
        self.max_speed = 20.0 * scale
        ## max_force ??
        self.max_force = 500.0
```



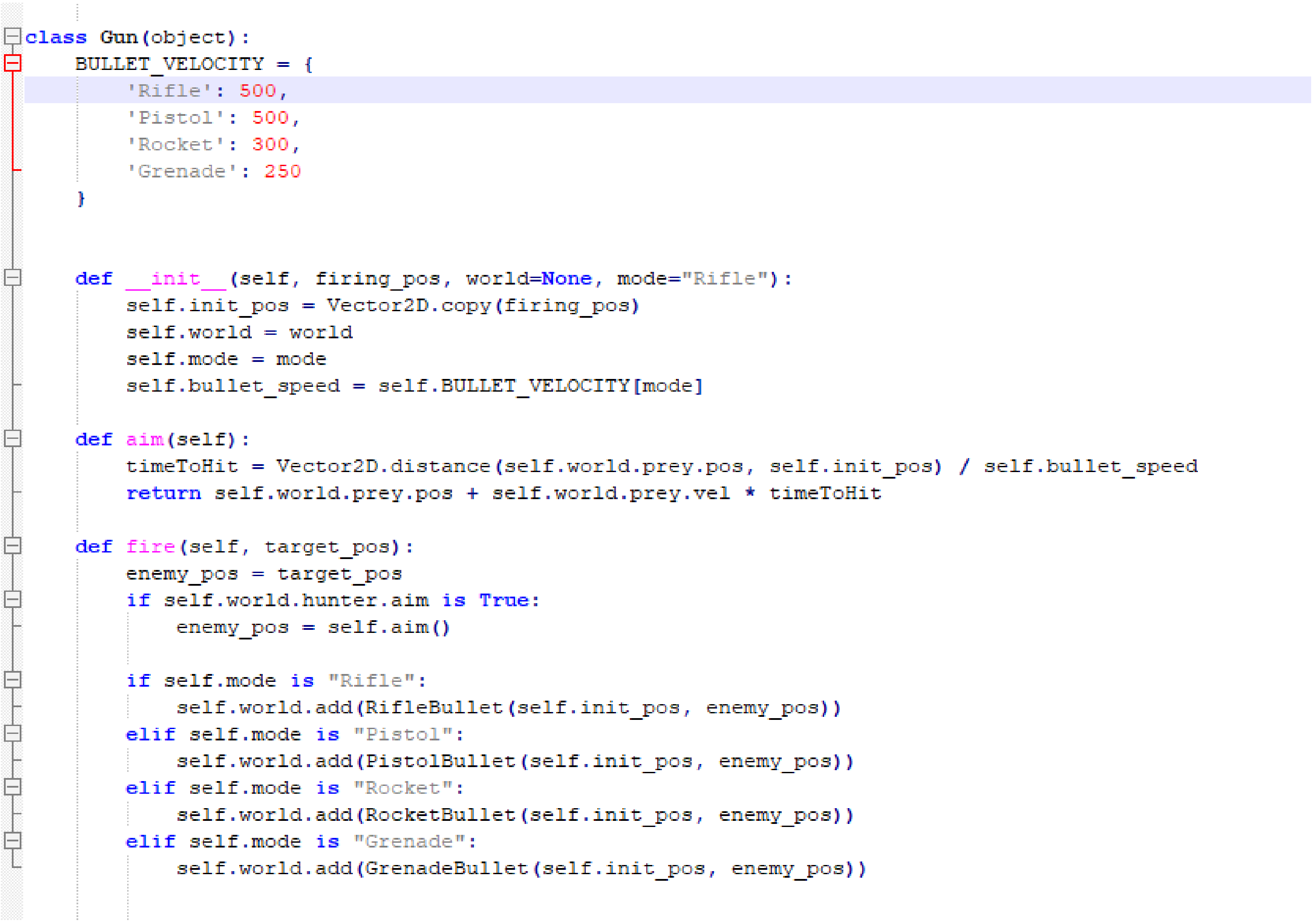
```
def render(self):
    egi.set_pen_color(name=self.color)
    pts = self.world.transform_points(self.shape, self.pos,
                                      self.heading, self.side, self.scale)
    # draw it!
    egi.closed_shape(pts)

def speed(self):
    return self.vel.length()

def seek(self, target_pos):
    ''' move towards target position '''
    desired_vel = (target_pos - self.pos).normalise() * self.max_speed
    return desired_vel - self.vel

def wander(self, delta):
    wt = self.wander_target
    jitter_tts = self.wander_jitter * delta
    inc = Vector2D(uniform(-1, 1) * jitter_tts, uniform(-1, 1) * jitter_tts)
    wt += inc
    wt.normalise()

    wt *= self.wander_radius
    target = wt + Vector2D(self.wander_dist, 0)
    wld_target = self.world.transform_point(target, self.pos, self.heading, self.side)
    return self.seek(wld_target)
```

In order for the hunter to fire upon the prey, we had to give the agent a gun and in order to do that we had to first declare the gun class and assign the hunter an instance of the gun class.

The gun class is responsible for the different fire modes, aiming at the prey and firing upon it.

Our aim function is designed similarly to the earlier boids pursuit code, taking into account the prey's position and velocity as well as the bullets velocity to predict where its future heading would be if the prey continued in its current direction.

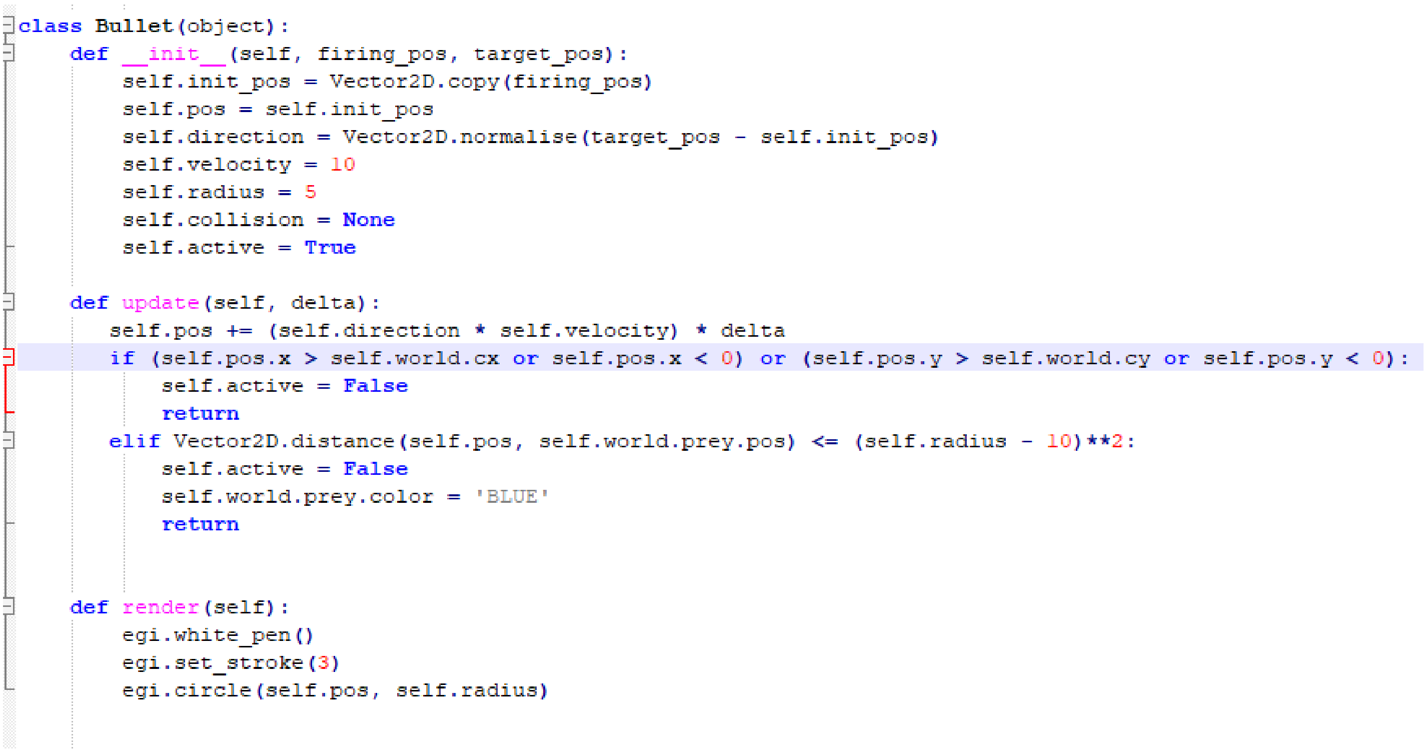
The fire function simply instantiates the appropriate bullet based on the guns mode, passing the appropriate trajectory over to the bullet class to handle traversal and collision detection.



```
class Gun(object):
    BULLET_VELOCITY = {
        'Rifle': 500,
        'Pistol': 500,
        'Rocket': 300,
        'Grenade': 250
    }


    def __init__(self, firing_pos, world=None, mode="Rifle"):
        self.init_pos = Vector2D.copy(firing_pos)
        self.world = world
        self.mode = mode
        self.bullet_speed = self.BULLET_VELOCITY[mode]

    def aim(self):
        timeToHit = Vector2D.distance(self.world.prey.pos, self.init_pos) / self.bullet_speed
        return self.world.prey.pos + self.world.prey.vel * timeToHit

    def fire(self, target_pos):
        enemy_pos = target_pos
        if self.world.hunter.aim is True:
            enemy_pos = self.aim()

        if self.mode is "Rifle":
            self.world.add(RifleBullet(self.init_pos, enemy_pos))
        elif self.mode is "Pistol":
            self.world.add(PistolBullet(self.init_pos, enemy_pos))
        elif self.mode is "Rocket":
            self.world.add(RocketBullet(self.init_pos, enemy_pos))
        elif self.mode is "Grenade":
            self.world.add(GrenadeBullet(self.init_pos, enemy_pos))
```

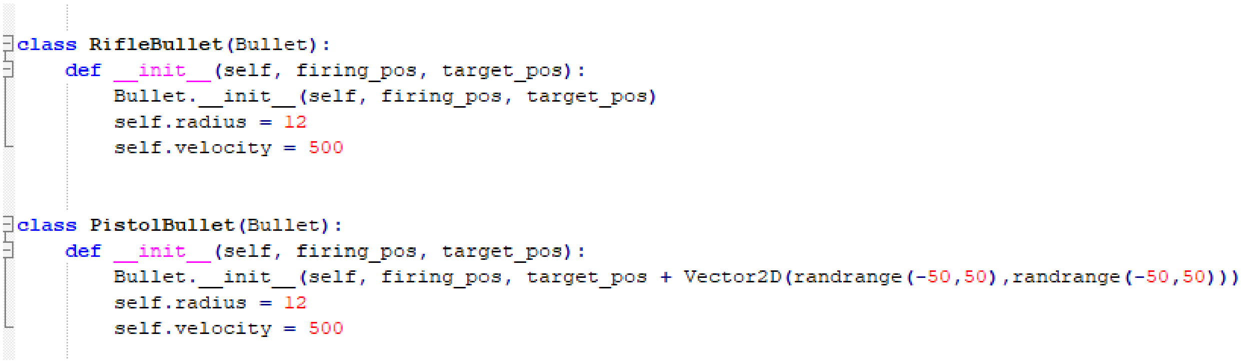
The Bullet class handles updating the position and rendering of the bullet.

In the update class, we check to see if there is any overlap between the bullet and the prey and if so, we remove the bullet from the world and change the prey's colour to illustrate being hit



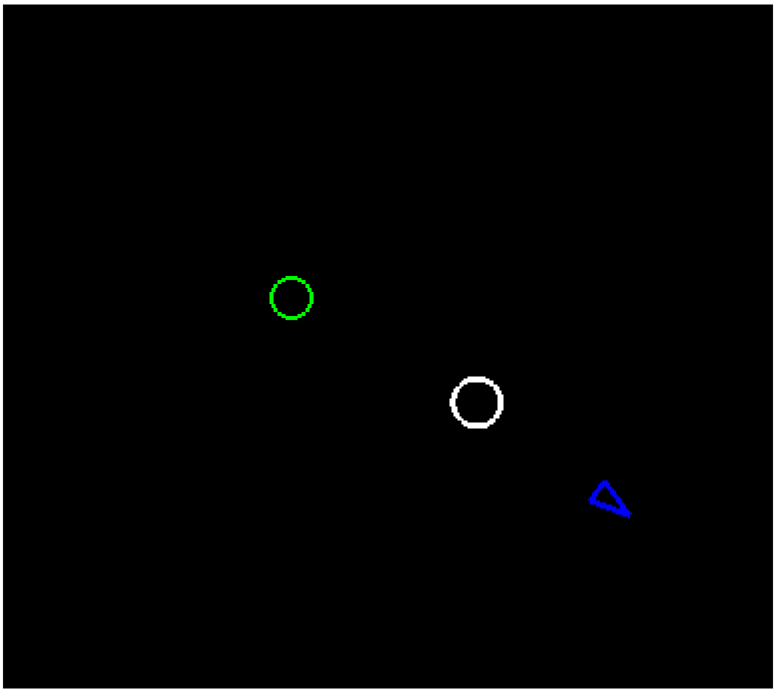
```
class Bullet(object):
    def __init__(self, firing_pos, target_pos):
        self.init_pos = Vector2D.copy(firing_pos)
        self.pos = self.init_pos
        self.direction = Vector2D.normalise(target_pos - self.init_pos)
        self.velocity = 10
        self.radius = 5
        self.collision = None
        self.active = True

    def update(self, delta):
        self.pos += (self.direction * self.velocity) * delta
        if (self.pos.x > self.world.cx or self.pos.x < 0) or (self.pos.y > self.world.cy or self.pos.y < 0):
            self.active = False
            return
        elif Vector2D.distance(self.pos, self.world.prey.pos) <= (self.radius - 10)**2:
            self.active = False
            self.world.prey.color = 'BLUE'
            return


    def render(self):
        egi.white_pen()
        egi.set_stroke(3)
        egi.circle(self.pos, self.radius)
```

To demonstrate inaccuracies with certain weapons, we introduced some randomness into the target position so that not all bullets will hit the target if even if they are aiming for the prey.



```
class RifleBullet(Bullet):
    def __init__(self, firing_pos, target_pos):
        Bullet.__init__(self, firing_pos, target_pos)
        self.radius = 12
        self.velocity = 500


class PistolBullet(Bullet):
    def __init__(self, firing_pos, target_pos):
        Bullet.__init__(self, firing_pos, target_pos + Vector2D(randrange(-50,50),randrange(-50,50)))
        self.radius = 12
        self.velocity = 500
```

What we found out:



From this spike we found out that we can create predictive behaviour using autonomous steering behaviours we developed earlier and adapt them to new situations. In this case our predictive behaviours allowed the hunter agent to behave in a realistic way and would be suitable for FPS non-player characters.