**Spike:** Spike\_4

**Title:** Agent Marksmanship

**Author:** Le Bao Duy Nguyen 102449993

**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:**

* Visual Studio Code
* Python 3.0+

**Tasks undertaken:**

* Download and install Visual Studio Code
* Download and install Python 3.0 & above
* Download and install Python extension within Visual Studio Code
* Read the codes and guidelines. Researching Canvas materials as well as Google, YouTube, etc.

Gun modes and cooldowns of each gun are declared.

GUN\_MODES = {

KEY.\_1: 'Rifle',

KEY.\_2: 'Rocket',

KEY.\_3: 'HandGun',

KEY.\_4: 'HandGrenade',

}

GUN\_COOLDOWNS = {

'Rifle': 0.8,

'Rocket': 2,

'HandGun': 0.5,

'HandGrenade': 1.4

}

Velocity/ speed of the bullets of each gun is declared.

Aim and fire function in order to make the agent fire and aim against its target.

Our aim function is designed similarly to the pursuit code, taking into account the target’s position and velocity as well as the bullets velocity to predict its future heading.

class Gun(object):

BULLET\_VELOCITY = {

'Rifle': 500,

'HandGun': 500,

'Rocket': 300,

'HandGrenade': 250

}

def \_\_init\_\_(self, firing\_pos, world=None, mode="HandGun"):

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 == "Rifle":

self.world.add(RifleBullet(self.init\_pos, enemy\_pos))

elif self.mode == "HandGun":

self.world.add(HandGunBullet(self.init\_pos, enemy\_pos))

elif self.mode == "Rocket":

self.world.add(RocketBullet(self.init\_pos, enemy\_pos))

elif self.mode == "HandGrenade":

self.world.add(HandGrenadeBullet(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 = 'WHITE'

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.

Radius of each bullet and speed varies for each gun.

class RifleBullet(Bullet):

def \_\_init\_\_(self, firing\_pos, target\_pos):

Bullet.\_\_init\_\_(self, firing\_pos, target\_pos)

self.radius = 12

self.velocity = 500

class RocketBullet(Bullet):

def \_\_init\_\_(self, firing\_pos, target\_pos):

Bullet.\_\_init\_\_(self, firing\_pos, target\_pos)

self.radius = 15

self.velocity = 300

class HandGunBullet(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

class HandGrenadeBullet(Bullet):

def \_\_init\_\_(self, firing\_pos, target\_pos):

Bullet.\_\_init\_\_(self,firing\_pos, target\_pos + Vector2D(randrange(-50,50),randrange(-50,50)))

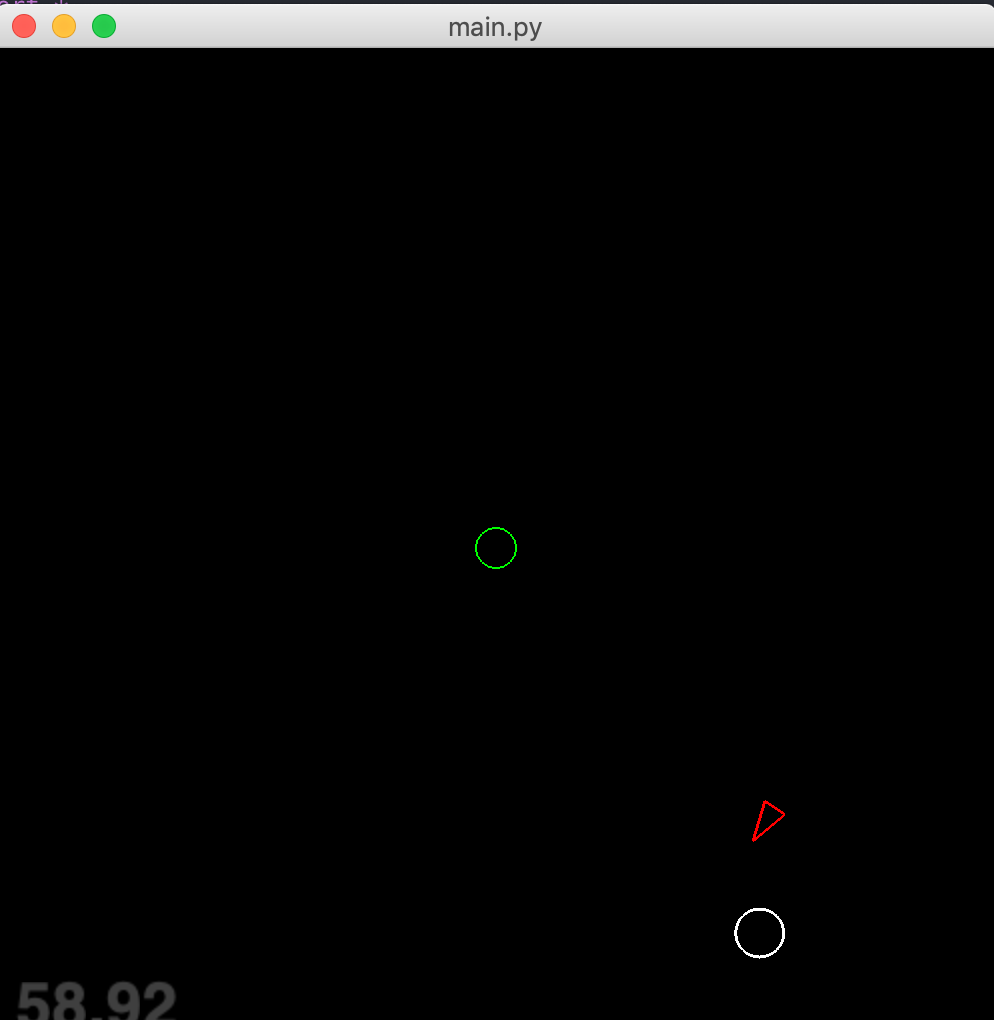
self.radius = 15

self.velocity = 250

**Output:**

E.g. HandGun

Agent is green, bullet is white, target is red.

****

Upon getting hit, target changed its colour to white to indicate death.

**A picture containing electronics

Description automatically generated**

**What we found out**

It is possible to recreate predictive behaviour using autonomous steering behaviours that has been developed earlier and adapt them to new situations.

In this case the predictive behaviours allowed the hunter agent to behave in a realistic way and would suit non-player characters in FPS games.

**EXTENSION**

if self.world.hunter.aim is True:

bullet\_speed = 20 if self.mode in ['Shotgun'] else 10

target\_pos = self.aim()

elif self.mode == 'Shotgun':

for i in range(5):

self.world.add(ShotgunBullet(self.init\_pos, enemy\_pos))

class ShotgunBullet(Bullet):

def \_\_init\_\_(self, firing\_pos, target\_pos):

Bullet.\_\_init\_\_(self, firing\_pos, target\_pos +

Vector2D(randrange(-100, 100), randrange(-100, 100)))

self.radius = 5

self.velocity = randrange(18, 22)

**A close up of a logo

Description automatically generated**

**Shotgun mode is added where it sent out waves of controllable number of bullets out. Slow but deadly ( guaranteed to hit ).**