SPIKE REPORT SPIKE 9

Jake Lucic

2103435 AI for Games

Number: Spike 9
Spike Title: Agent Marksmanship
Personal: Jake Lucic (2103435)

Context:

The method of targeting an agent uses to attack a target with a range weapon should depend on weapon projectile characteristics. In particular speed (infinite or finite), accuracy and range are all worth considering as part of a targeting plan.

Knowledge/Skill Gap:

Developer needs to be able to implement agent-targeting methods, including target prediction, so that the agent can successfully use different projectile weapons.

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:

In this task, the technologies used are listed below:

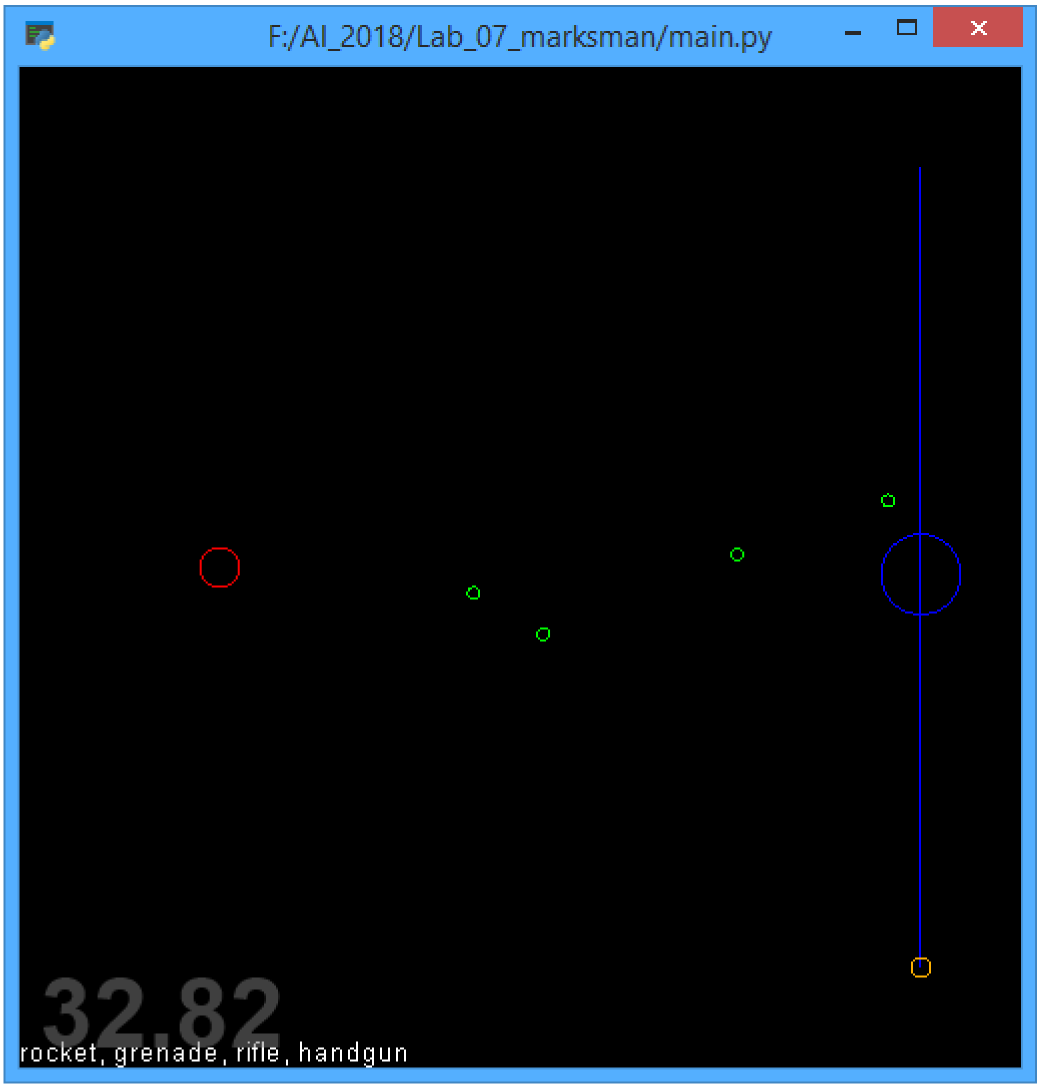
- Simple Code base of the Steering Spikes
- Python IDLE v3.6.4 / Python language / PyCharm
- Path Following, Wandering Lecture
- Tactical Action Selection and Predictive Shooting Lecture
- Planning Notes in Spike 09 sheet

Tasks undertaken:

- Read the code base and understand what each function does
- Study the different key words and variable names
- Follow the lectures on Path Following, and Tactical Action Selection
- Approach Task using similar process below
- Read about predictive shooting
    - https://stackoverflow.com/questions/2248876/2d-game-fire-at-a-moving-target-by-predicting-intersection-of-projectile-and-u

Process:

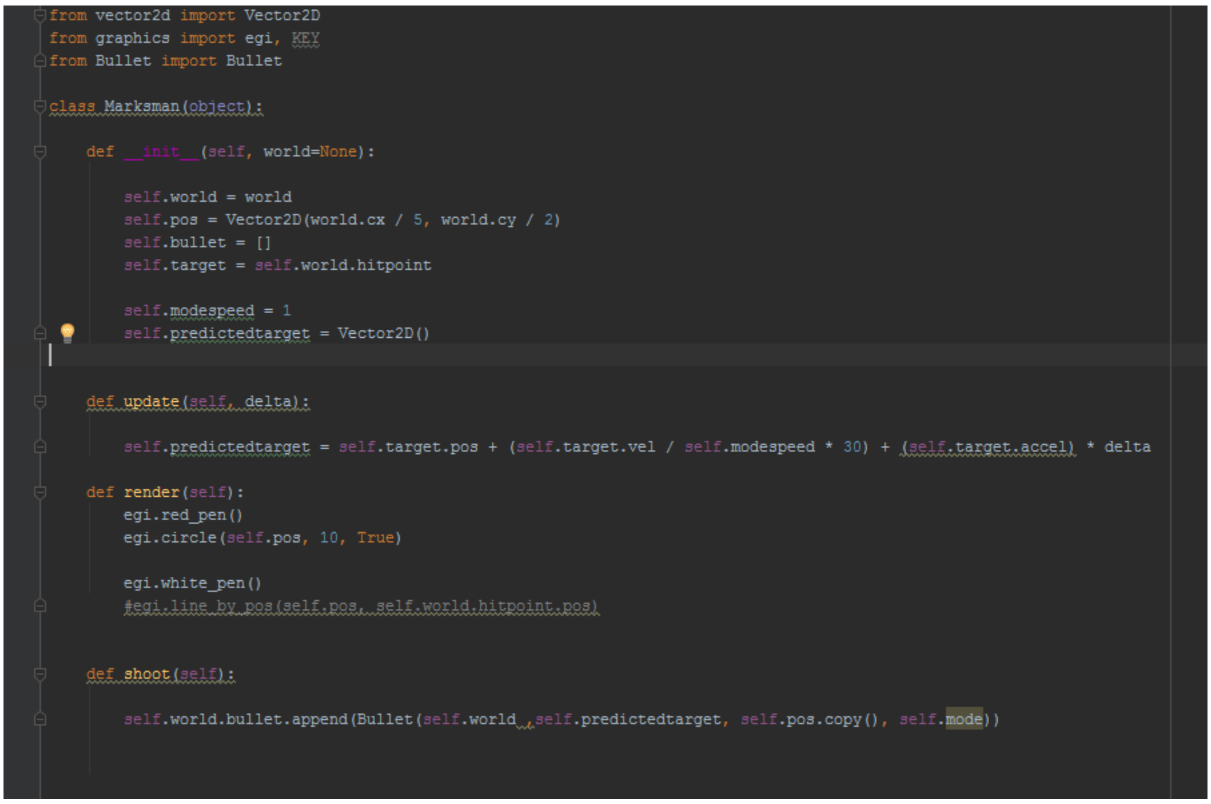
I approached this task by first creating a Hitpoint agent, that functions like an agent, however its only movement mode is to follow path. To get this to follow a strict vertical path, I read the Path() class, and then instead of using the randomize path, I created a path using hard coded points, just so it was the same every start up.


F:/AI_2018/Lab_07_marksman/main.py
32.82
rocket, grenade, rifle, handgun

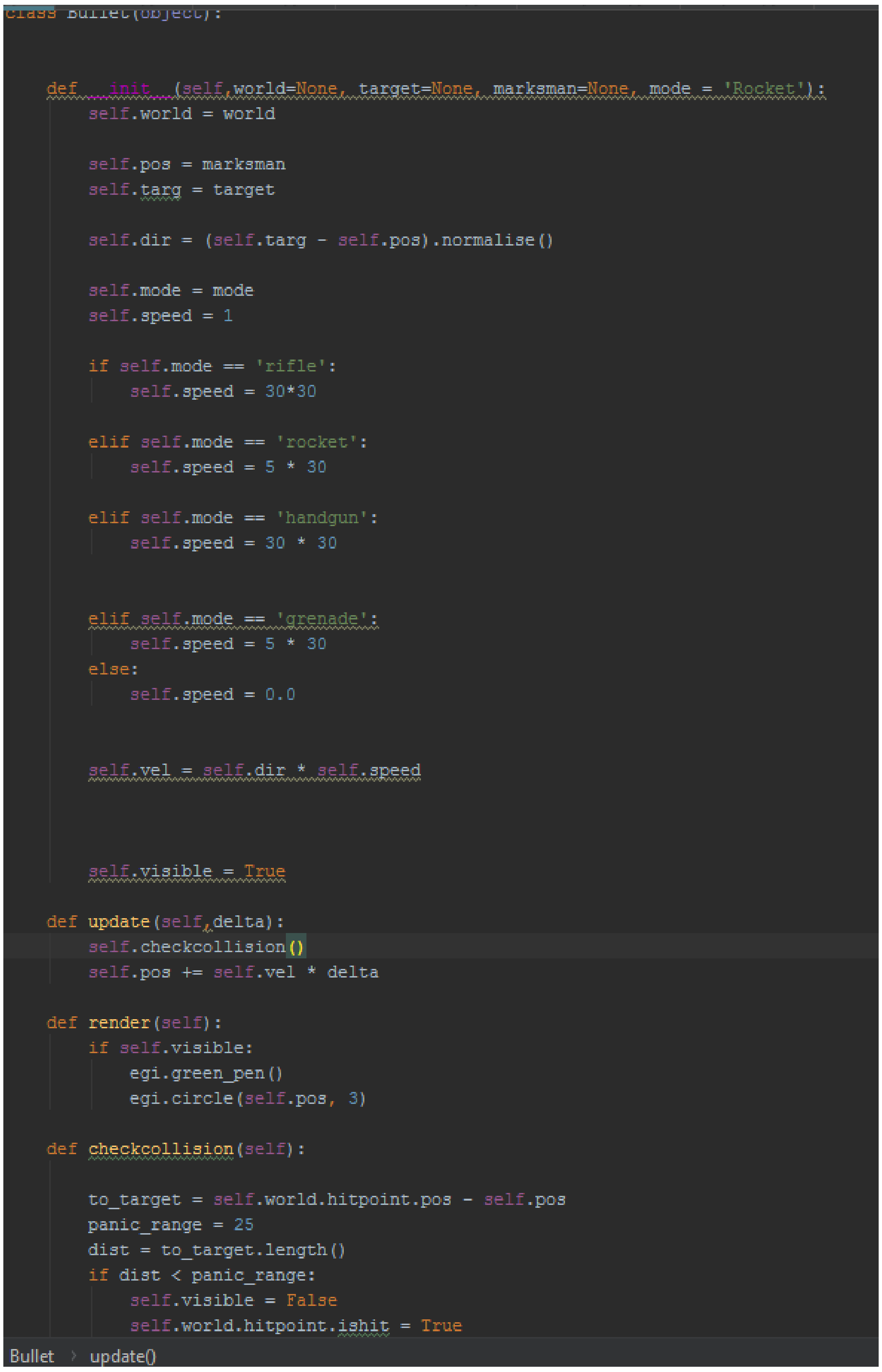
The Blue circle is the Hitpoint, and the Red circle is the marksman. The little green circles are my bullets that have altering speeds and accuracy.

After creating the Hitpoint Agent, I created the static Agent which would be my Marksman. All the bullets fired would originate at the Marksman's position. This would also hold the equation for predicting it's targets future position, using the targets position, and adding (target.velocity / bulletspeed * scale) + acceleration * time. This is demonstrated in the picture of code below.

When the bullet is appended to the list by using a firing button, it gives its predicted target to the bullet so it knew where to shoot to.



```python
from vector2d import Vector2D
from graphics import egi, KEY
from Bullet import Bullet

class Marksman(object):

    def __init__(self, world=None):

        self.world = world
        self.pos = Vector2D(world.cx / 5, world.cy / 2)
        self.bullet = []
        self.target = self.world.hitpoint

        self.modespeed = 1
        self.predictedtarget = Vector2D()


    def update(self, delta):

        self.predictedtarget = self.target.pos + (self.target.vel / self.modespeed * 30) + (self.target.accel) * delta

    def render(self):
        egi.red_pen()
        egi.circle(self.pos, 10, True)

        egi.white_pen()
        #egi.line_by_pos(self.pos, self.world.hitpoint.pos)


    def shoot(self):

        self.world.bullet.append(Bullet(self.world ,self.predictedtarget, self.pos.copy(), self.mode))
```



```
class Bullet(object):


    def __init__(self,world=None, target=None, marksman=None, mode = 'Rocket'):
        self.world = world

        self.pos = marksman
        self.targ = target

        self.dir = (self.targ - self.pos).normalise()

        self.mode = mode
        self.speed = 1

        if self.mode == 'rifle':
            self.speed = 30*30

        elif self.mode == 'rocket':
            self.speed = 5 * 30

        elif self.mode == 'handgun':
            self.speed = 30 * 30


        elif self.mode == 'grenade':
            self.speed = 5 * 30
        else:
            self.speed = 0.0


        self.vel = self.dir * self.speed



        self.visible = True

    def update(self,delta):
        self.checkcollision()
        self.pos += self.vel * delta

    def render(self):
        if self.visible:
            egi.green_pen()
            egi.circle(self.pos, 3)

    def checkcollision(self):

        to_target = self.world.hitpoint.pos - self.pos
        panic_range = 25
        dist = to_target.length()
        if dist < panic_range:
            self.visible = False
            self.world.hitpoint.ishit = True
```

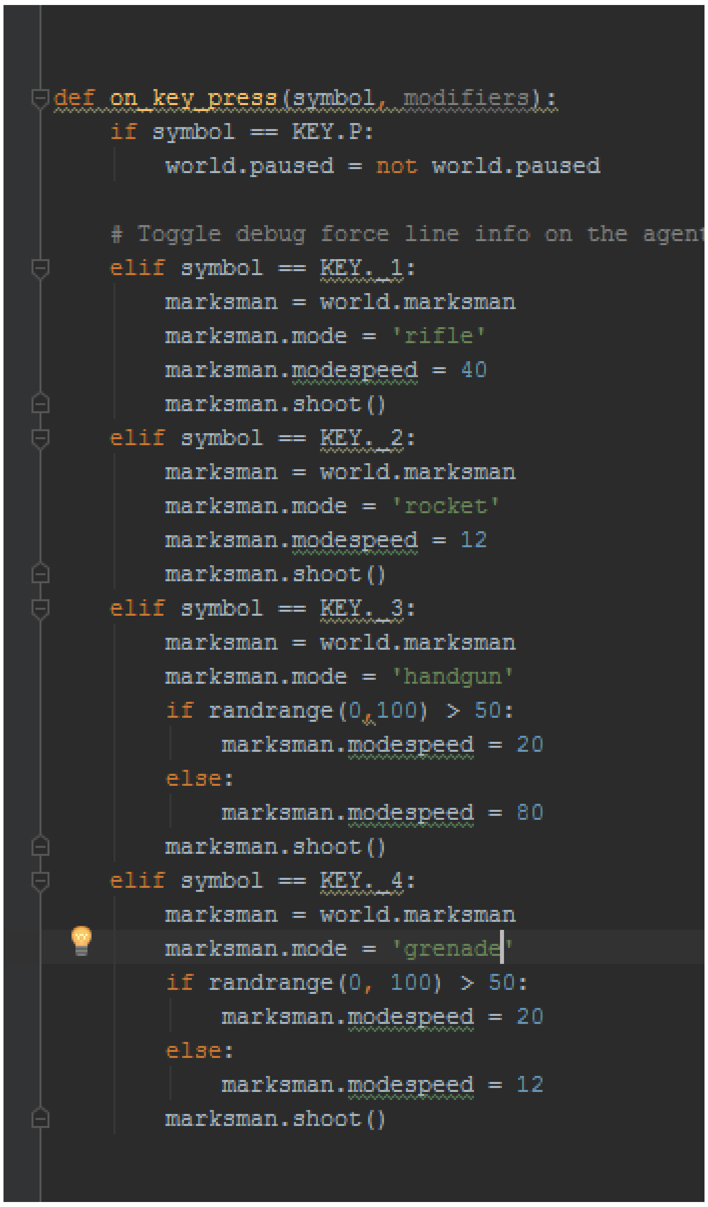
The Bullet needs to know its originating position and its target position, both provided by the marksman class. All the bullet needs to do is move from one place to the next, and check for collision.

The Bullet knows its mode, and will adjust its speed accordingly. I handled the accuracy in the controls section, which isn't elegant, but it did the job. I simply rolled a randrange on key press, and if it was greater than a percentage, then the marksman would alter its predicted position slightly off.

All of the bullets were quite similar, other than the 2 differing speeds.

In my mind, I thought that the marksman should be the class to handle accuracy.

This is shown on the next page.



```
def on_key_press(symbol, modifiers):
    if symbol == KEY.P:
        world.paused = not world.paused

    # Toggle debug force line info on the agent
    elif symbol == KEY._1:
        marksman = world.marksman
        marksman.mode = 'rifle'
        marksman.modespeed = 40
        marksman.shoot()
    elif symbol == KEY._2:
        marksman = world.marksman
        marksman.mode = 'rocket'
        marksman.modespeed = 12
        marksman.shoot()
    elif symbol == KEY._3:
        marksman = world.marksman
        marksman.mode = 'handgun'
        if randrange(0,100) > 50:
            marksman.modespeed = 20
        else:
            marksman.modespeed = 80
        marksman.shoot()
    elif symbol == KEY._4:
        marksman = world.marksman
        marksman.mode = 'grenade'
        if randrange(0, 100) > 50:
            marksman.modespeed = 20
        else:
            marksman.modespeed = 12
        marksman.shoot()
```

I handled the accuracy by adjusting modespeed (which is variable used by Marksman to predict its targets future position)

Controls

1:	Shoots the Rifle
2:	Shoots the rocket launcher
3:	Shoots the Handgun
4:	Throws a Grenade

I found that as the target slowed down or increased its acceleration, the marksman's predictive position would be quite inaccurate, which provides some realism I suppose. Although it was a happy accident, I understand that if the marksman was moving in a consistent fashion, the accuracy would be spot on.