**Step 1:** The game was improved using a simple track that the agent could follow. Track used can be randomised by pressing R.

Agent.py

def follow\_path(self):

self.path.render()

if self.path.is\_finished():

return self.arrive(self.path.current\_pt(), 'slow')

else:

if (self.path.current\_pt() - self.pos).length() < 50:

self.path.inc\_current\_pt()

else:

return self.seek(self.path.current\_pt())

return Vector2D()

def render(self, color=None):

''' Draw the triangle agent with color'''

*# draw the path if it exists and the mode is follow*

if self.mode == 'follow\_path':

self.path.render()

*# draw the ship*

if(color == None):

egi.set\_pen\_color(name=self.color)

else:

egi.set\_pen\_color(name=color)

pts = self.world.transform\_points(self.vehicle\_shape, self.pos,

self.heading, self.side, self.scale)

*# draw it!*

egi.closed\_shape(pts)

Main.py

*## LAB 09 STEP 1: Reset all paths to new random ones*

elif symbol == KEY.R:

for agent in world.agents:

agent.randomise\_path()

A picture containing laptop, computer, star

Description automatically generated

A picture containing laptop, object, computer, screen

Description automatically generated

**Step 2:** The ship is now equipped with wandering mode. Adding shapes to the ship by pressing I.

Agent.py

if self.mode == 'wander':

*## ...*

wnd\_pos = Vector2D(self.wander\_dist, 0)

wld\_pos = self.world.transform\_point(

wnd\_pos, self.pos, self.heading, self.side)

egi.green\_pen()

egi.circle(wld\_pos, self.wander\_radius)

egi.red\_pen()

wnd\_pos = (self.wander\_target + Vector2D(self.wander\_dist, 0))

wld\_pos = self.world.transform\_point(

wnd\_pos, self.pos, self.heading, self.side)

egi.circle(wld\_pos, 3)

def update(self, delta):

''' update vehicle position and orientation '''

*# calculate and set self.force to be applied*

*## force = self.calculate()*

force = self.calculate(delta) *# <-- delta needed for wander*

*## limit force? <-- for wander*

*# ...*

force.truncate(self.max\_force)

*# determin the new accelteration*

self.accel = force / self.mass *# not needed if mass = 1.0*

*# new velocity*

self.vel += self.accel \* 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.length\_sq() > 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)

A picture containing fireworks

Description automatically generated