Spike: Task 14

Title: Agent Marksmanship

Author: Sam Huffer, 101633177

# Goals / Deliverables

* Basic deliverables:
  + An agent targeting simulation with:
    - An attacking agent (stationary or otherwise)
    - A moving target agent (can move between two points) that shows when it has been hit
    - Several weapons that can successfully hit the target, including:
      * Rifle (fast, accurate)
      * Rocket (slow, accurate)
      * Hand gun (fast, inaccurate)
      * Hand grenade (slow, inaccurate)
* Extensions:
  + Account for rate of fire and effect range (specifically, make a shotgun with a low rate of fire and short but deadly effective range; move the attacker closer before shooting).
  + Splash damage (projectile that explodes on the ground; attacker should take this into account when aiming).
  + A target that can avoid slow projectiles and / or move to hiding spots when attacked.

# Technologies, Tools, and Resources Used

* SublimeText (for editing, executing and testing the code)
* Learning materials on Canvas (for instructions and sample code)

# Tasks Undertaken

* I copied the Tactical Steering project from Task 10 into the Task 14: Agent Marksmanship folder, then cleaned up some of the existing code to be better suited for the current scenario, and started putting together a skeleton for the methods required by this task. I also added the walls from Task 11: Emergent Group Behaviour, tweaking existing methods and adding the appropriate calls in main, world, agent, etc. as needed to adjust them for the current task, and added keyboard controls for toggling the obstacles and walls on and off.
* I set up the target and shooter to start in stationary positions and for the latter to shoot the former from those stationary positions. Projectiles die on contact with the target or when they pass outside the bounds of the screen, and are pooled inside the originating agent rather than being instantiated when fired and destroyed on contact to improve program efficiency. I then implemented inaccuracy (within a specified margin of error) and tweaked the agents to show when they’ve been shot by going red for 0.1 seconds.
* I set up “moving back and forth” and “wander” movement types for the target, and predictive shooting for the shooter. I tried using a fancy method that uses sine and cosine functions, but that failed miserably. Currently, the shooter calculates time it’d take the projectile to get to the current / currently-calculated-future position, then predicts the target’s position at that new time using d = ut + ½ at2, and wraps position if the target is wandering. The shooter iterates through this process until it thinks the projectile will be able to get within an acceptable distance of the target.
* I tweaked hiding spots, fleeing and hunting behaviours, and field of view updating to work with a single hunter and evader, and to fit with this task’s logic for when to move and what movement to take.
* I set up explosions for rockets and hand grenades, and automatic firing, rate of fire, and effective range (while the target is evading) for all weapons, using raw rounds per second and scaled effective range values from equivalent *Halo: Combat Evolved* weapons a baseline where possible, and estimating from there how fast I’d realistically fire them if I was trying to be accurate. While doing so, I also set up the shotgun configuring it to fire a scattershot of 5 pellets at once.
* Lastly, I implemented projectile avoidance for the target when it is not stationary, adapting the obstacle avoidance code and adding a method that gets the agent to move perpendicularly to the vector it would take if moving towards the projectile.

# Instructions for Operating the Code

* A: toggle the display of agents’ obstacle avoidance range if it would otherwise not be displayed.
* B: toggle walls (i.e. boundaries of the simulation space) on and off
* I: toggle the display of agents’ force, velocity and net desired change in position.
* N: create a new obstacle in a random but valid position.
* O: toggle obstacles and hiding spots on and off.
* P: pause or un-pause the game.
* R: reposition all obstacles in random but valid positions. Obstacles are automatically repositioned when the window changes size.
* T: scroll through target movement types (stationary, moving between two points, evading).
* W: scroll through shooter weapons.
* Escape: exit the game.

# Code Snippets

Figure : the code for determining how each agent should move, and if the shooter is able to shoot again.

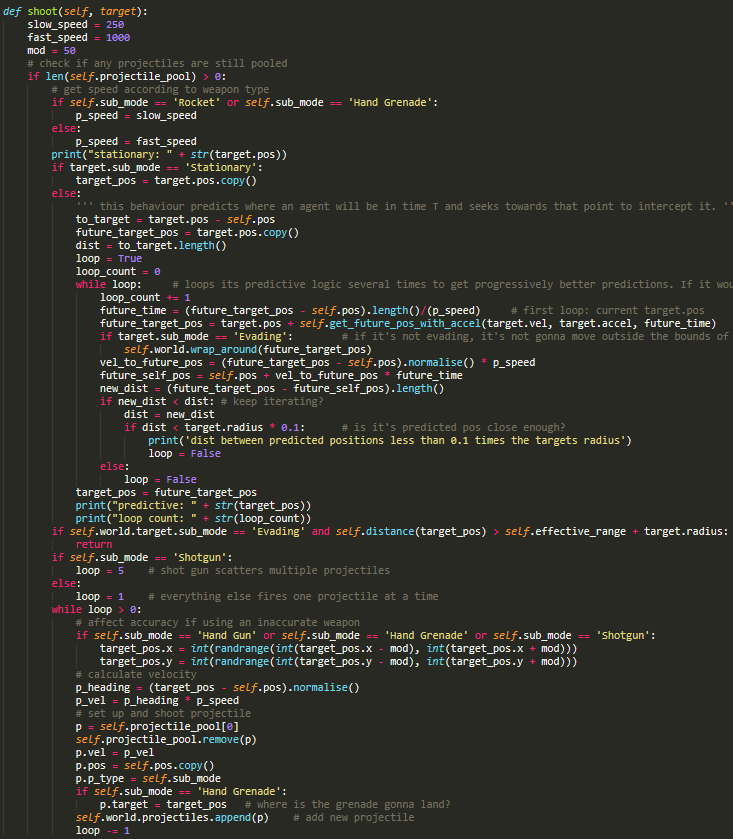


Figure : the logic for what properties the projectile will have, and calculating where the shooter should aim its shot.

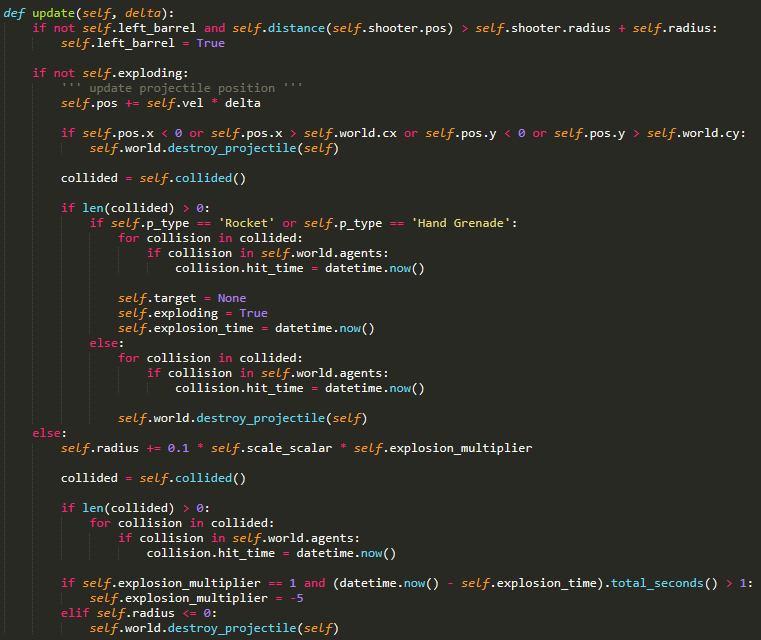
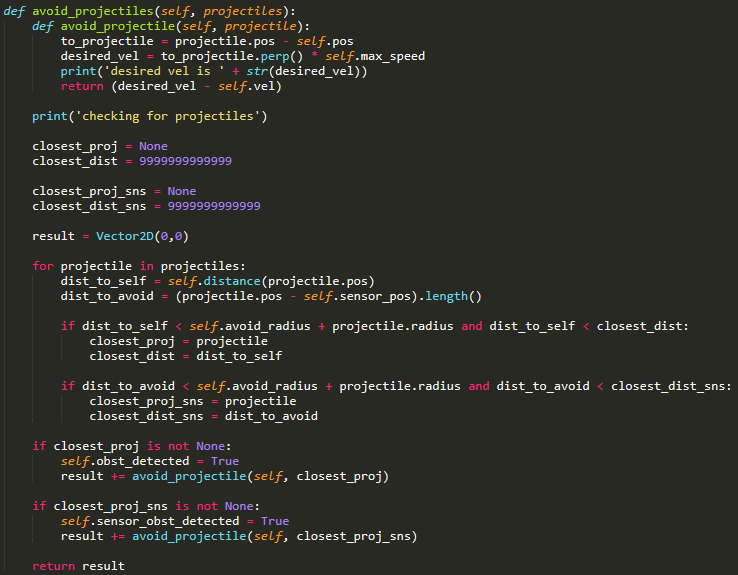


Figure 4: the code for the target to check if there are projectiles to dodge, and how to go about dodging them.

Figure 3: the projectile’s update method, which handles what to do if it goes outside the screen, collides with another object, or is currently exploding.

# In-Simulation Screenshots

Figure : a grenade detonating after colliding with the target.

Figure 6: the shooter firing its shotgun at the target.

# What I Found Out

* Setting up the different weapon / projectile types was easy enough. Setting up the movement settings and writing the predictive logic, even with the pursuit method as a basis for the latter, was the tricky bit.
* If target changes its velocity or acceleration (either their vector or magnitude), that can render the predicted position inaccurate proportionate to the change in vector or magnitude, particularly if the projectile isn’t moving fast enough or shot from a close enough position for the inaccuracy to be negligible.
* Once effective range while the target was wandering was implemented, if obstacles were disabled and the shooter thought target’s predicted position, after wrapping, would be out of range, the shooter would not shoot; this can look a bit weird if that position is wrapped, as it seems like the target should be fired upon as it’s current position is within the weapon’s range.
* With projectile avoidance, when projectiles are moving at a fast speed, the target will try to avoid them but almost certainly fail; any effort to avoid them will at most only register as a slight recoil from the projectile as if the projectile pushed the agent away. However, when projectiles are moving at a slower speed, the target will have more success trying to avoid them, doing best when the projectile would have glanced their side if it did hit. Even in an unsuccessful dodging attempt, the agent will have more visibly attempted to dodge the projectile.