AI4GAMES - Lab 2

Report for tasks 1, 2 and 3

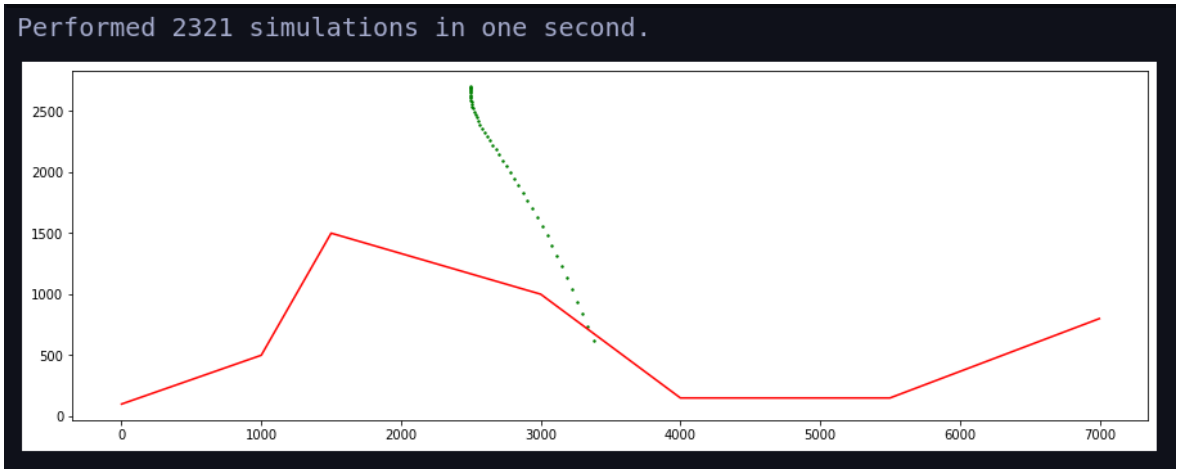
Filip Komorowski

14 listopada 2021

1 Forward simulator

I implemented the forward simulator in python. It averaged about 2 thousand iterations per second on the first map.

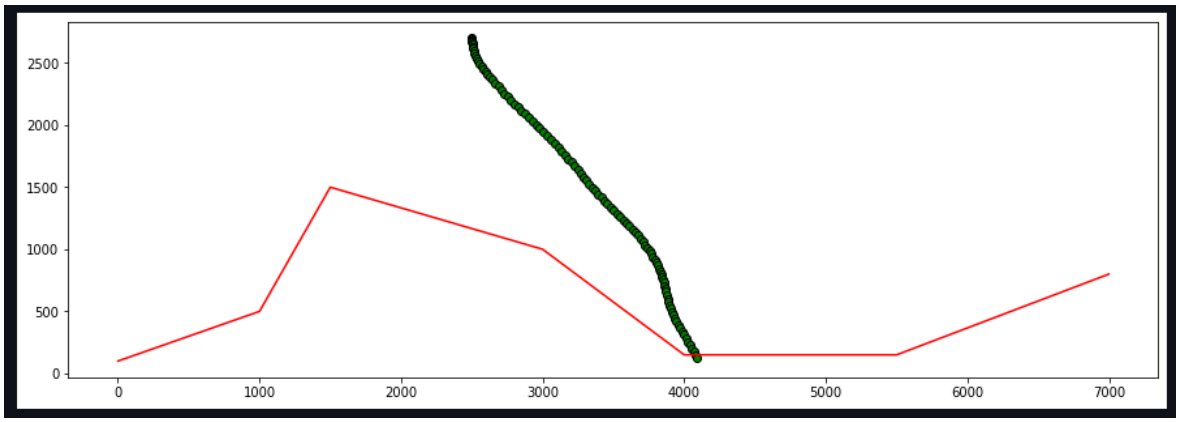
Here you can see one of the random tries:

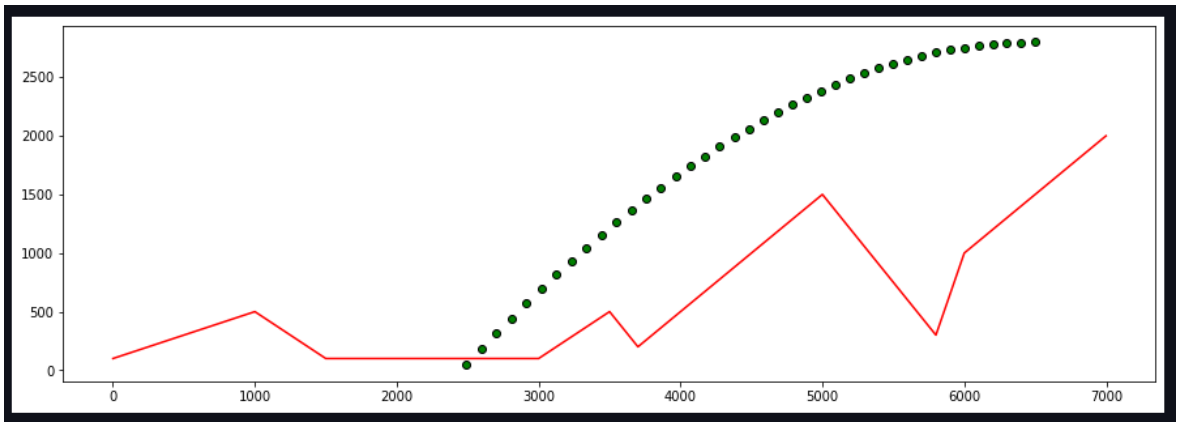


2 RHEA

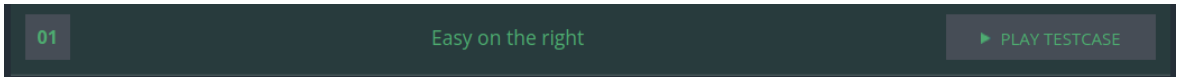
I decided to implement this solution in python as well, since this is what I am used to from Evolutionary Algorithms class, but that was a mistake. Under the tight time constraints that codingame has, my solution can't solve any of the puzzles. I studied the blog that was linked under the assignment and tried implementing the ideas and learnings of the author, but the effects were not as expected.

When running the RHEA algorithm with 1 second per move I managed to solve the first puzzles locally:





Hard-coded moves from local calculations:



3 Finding the best fitness function

So my intuition was to highly penalize crashes and reward small distances to the landing area, angles and velocities that are in the landing limits. The author of the blog came pretty much to same conclusion, so after some trials my function looks like this:

```
        distance_reward = determine_dist_reward(landing_x1, landing_x2, landing_y, x, y
                                                , x_start, y_start)
        angle_reward = determine_angle_reward(angle)
        if abs(v_speed) <= 40:
            v_speed_reward = 1000
        else:
            crash = True
            v_speed_reward = 0

        if abs(h_speed) <= 20:
            h_speed_reward = 1000
        else:
            crash = True
            h_speed_reward = 0

        total_reward = distance_reward*10 + angle_reward + v_speed_reward +
                                                h_speed_reward
        if crash:
            total_reward -= 100000
```

Where dist helper function looks like:

```
        dist_x = 0
        dist_y = 0
        middle = (lx2 + lx1) / 2
        dist_x = abs(middle - x) ** 2

        if ly - 1 <= y <= ly + 1:
            dist_y = 0
        else:
            dist_y = (y - ly) ** 2

        dist = np.sqrt(dist_x + dist_y)

        return -dist
```