I_Like_trains: Sara Cevallos & Alicia Durand

One Pager

The first thing we did was to try and brainstorm together our ideas for this code. We tried to imagine what functions we were going to need and how they should work. At first, we didn't try to think about them working together, but just what each function should do.

- We knew we would need a function that would avoid walls. We wanted to be able to look ahead of us and see whether we were going towards a wall. For that we needed to create a buffer that would look a few steps ahead of us. That way, if the train is going towards a wall, the buffer will "see" that and will force the train to change its direction. Our first problem with that was that we implemented it to consider every safe direction (putting them into a list) and pick randomly between those. It worked but it wasn't good once the train had an objective (go to passengers) because it would pick a random direction and maybe move away from the goal. We fixed this problem later by adding a system of scores; the train considers the outcome of the direction (getting closer to a passenger, to the delivery zone or just not dying) and picks the direction with the best outcome.
- We then implemented a function that would fetch passengers. We calculate our current position and the position of the closest passenger by looking at the x and y axes, we then go to the passenger. We also direct it towards clusters of passengers.
- We then made a function to avoid other trains; it works kind of like the avoiding walls function but it tries to predict the position of the other trains. It works well but not so much with head on collisions. We tried to make it better, but it still dies a few times like that. We think that against some bots that are a bit more random it doesn't predict the moves well enough to avoid the train.
- We then made it go to the delivery zone every time it had 4 passengers or more and empty itself completely before continuing.
- At that point our code worked well but it wasn't optimized; sometimes it passed next to a passenger without taking it because it already had 4 and was going to the delivery zone. We coded it to be "opportunistic", meaning it checks if a passenger is close and will do a little detour to take it. Same thing with the delivery zone; it will go through it if it passes by.
- Once we were satisfied with the code (it was very long) we needed to "put together" what we could. The system of scores, the safety checks, the calculations of Manhattan distances etc… A lot of the code was very repetitive, so this allowed us to make it much shorter.

Throughout the entire process, we added many debuggers to see what worked and what didn't. We also coded everything function by function (always calling them in the get move) so if something didn't work, we knew exactly where the problem was.

Due to our lack of basic python skills, we did have to use AI a lot, but we always knew exactly what we wanted. Meaning the structure and the ideas of the code are entirely ours. We think that if we had had a lot more time we might have been able to do everything by ourselves, but AI was a good shortcut.

We still had a few ideas for the code that would have been complicated to code but that we still want to mention; we realized that all the bots have very different strategies and that our agent isn't always the best against some of them. We wanted our agent to adapt. If the opponent is fast and goes to the delivery zone a lot, our agent should also be fast, but if the opponent gets very long and takes up a lot of space, our agent shouldn't go to the delivery zone too many times because it's risky. Also with the map, when the delivery zone is close to a wall, it's dangerous to go there a lot so it would be good to make our agent adapt to those things. It might have been good to also make it attack other trains, but we really focused on defense because it gets us more points. To conclude we found this project interesting and the idea very fun, but it was probably a little bit too complicated for us. It also took us a huge amount of time, even with AI. But with everything, we learned a way to code that works for us and familiarized ourselves with the concept of servers.