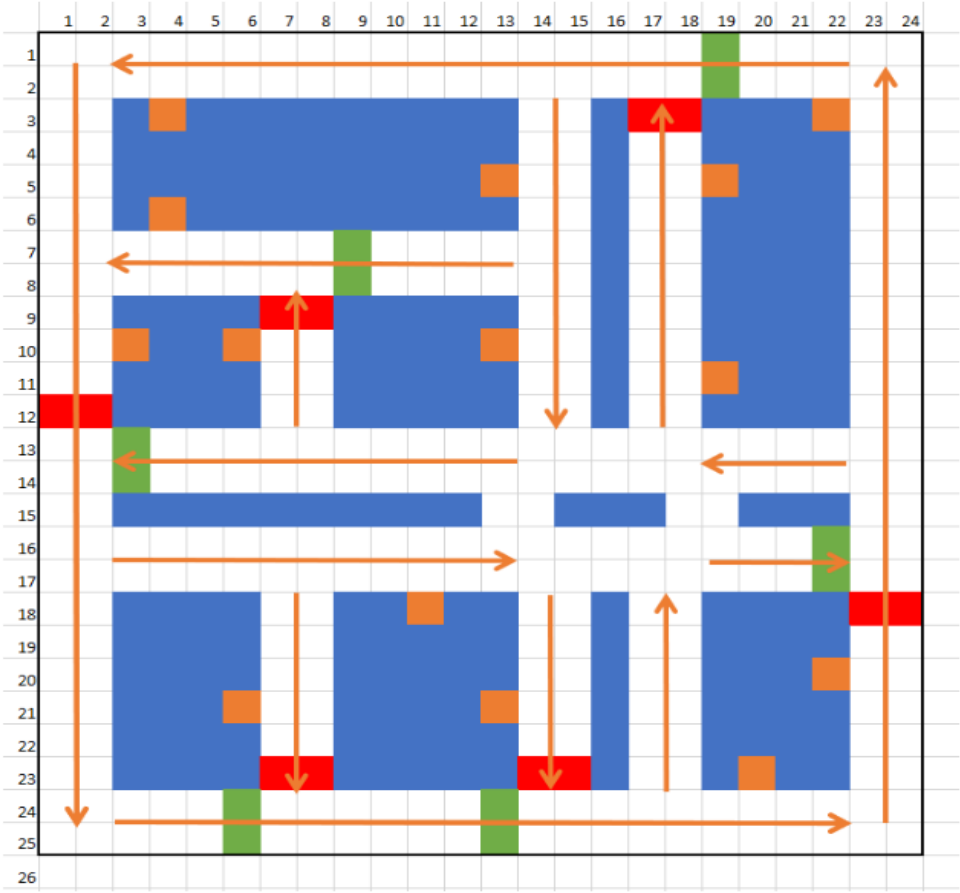
Multi-agent system with Mesa simulated in 3D project of Unity

Authors: José Luis Madrigal Sánchez, Christian Parrish Gutiérrez Arrieta, Jorge Isidro Blanco Martínez, César Emiliano Palome Luna

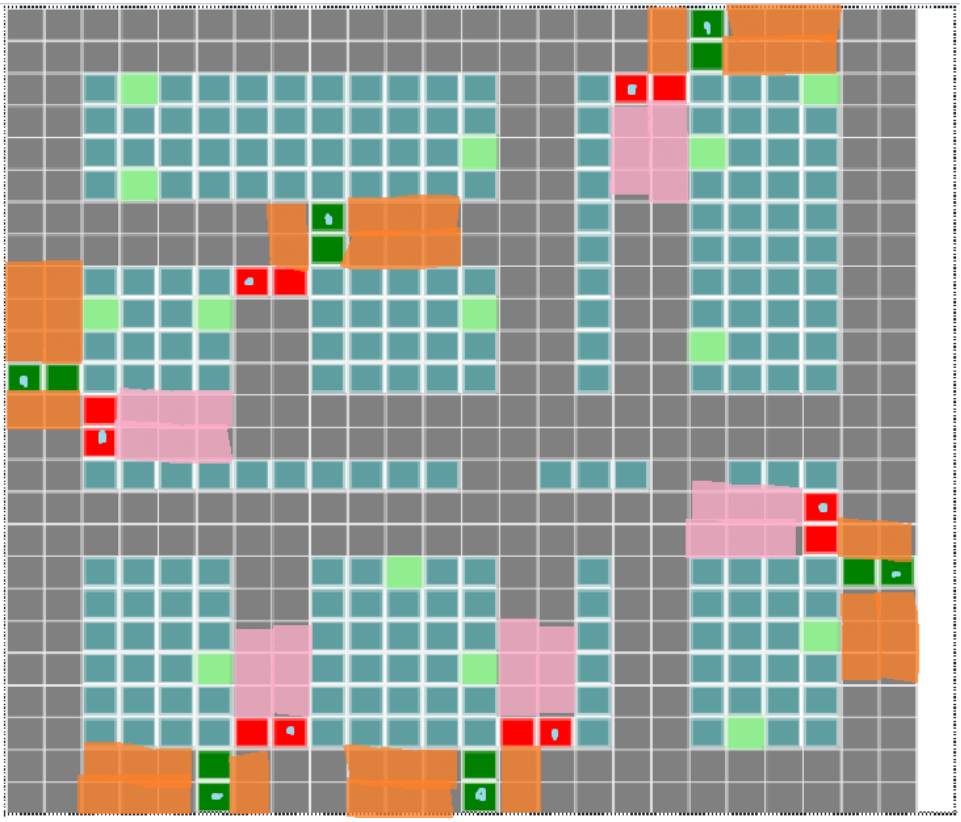
Here you can see the environment, where we have a city with the following elements:

- Buildings (blue)
- Streets (white)
- Destinies (orange)
- Traffic lights (green and red)
- Cars (started in random positions to get to a random destiny)

The city was made with an txt file, which had the position of the constant elements of objects, which are buildings, destinies and streets. The only true agents are the cars and traffic lights. The txt had too the sense of the streets:

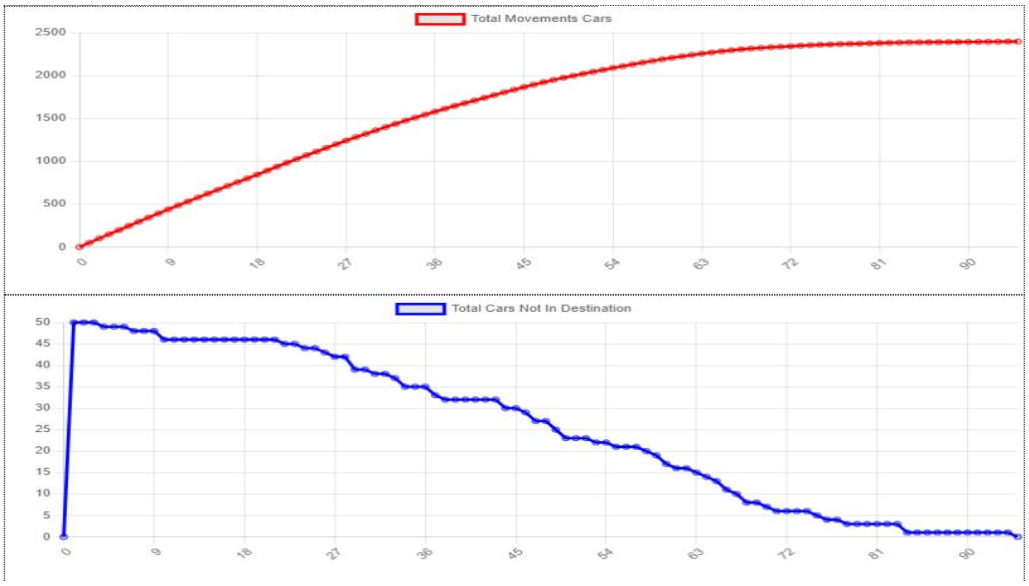


Here is the same environment, made with the Mesa library:



Since each complete traffic light is made up of 2 different traffic light agents, only one of them was chosen to count the cars on your street (range of 3 cells in both lanes, total of 6), and that same one would be the one that alerts you At the traffic light next to him (his brother) what color to be in, in the image they are the ones with the blue dot. In the same way, there is the possibility of having the same number of cars in the 2 streets of an intersection, so the priority traffic lights were designated (those in a street that continues to go straight, without turning), which in the image are the ones that are green, and in fact they have 2 more cells to sense, to maintain priority when the cars have already passed. It is also worth mentioning that the priority ones are the only ones that make the comparison with their opponent, so as not to repeat the same process twice, in this case, as mentioned, each of the green ones, being priority, is the one that sends the signal to the other to make it red.

To get an idea there is a data collector with the code execution, a test with 50 cars is included (it is important to mention that when they arrive at their destination, they are removed from the grid and scheduler):



The 50 cars reached their destinations in 95 steps, making a total of 2,396 movements.

Here you can see the implementation in Unity: