ID2209 – Distributed Artificial Intelligence and Intelligent Agents

Assignment 1 – GAMA and Agents

Group
Ilias Merentitis
Chrysoula Dikonimaki

12.11.2021

In this assignment, our task was to simulate a Festival with Stores. The Guests of the Festival get hungry or thirsty and they go to the Information Center to get informed about the available Stores.

How to run

Run Gama.exe and import the .gaml files.Press the green button to execute the experiment called myExperiment (each .gaml file has a separate experiment).

Species

Guest

The Guests are agents that normally wander around the premises of the festival (colour = green, shape = circle). However there is a 1% chance that they will either get hungry or thirsty. That means that there is a 0.1% chance that they will get both hungry and thirsty. (yellow = hungry, orange = thirsty, red = both). There is also a 20% chance that a guest will be evil at the start of the festival. In that case a black border is drawn around them. According to the implementation of Festival2 when a guest is hungry he can choose to visit a shop he has already visited or discover a new one, by asking in the information centre. This is of course a random decision.

Security

The security guard (colour = blue, shape = circle) is patrolling the premises of the festival. When an evil guest enters an information centre, he is reported (by the information centre) to the security guard who moves (with double speed) towards the guest and when he gets into range executes him. It is possible that the security guard will be busy and start off to a new execution if he has received calls about more troublemakers. Only when the festival grounds have been totally cleared can he continue with his patrol.

Information Center

The information centre (colour = blue, shape = square) is a unique building on the map that holds information about the different shops available in the festival. When a guest that is hungry, thirsty or both enters, he can require information about appropriate shops. It is also responsible for reporting evil guests to the security guard.

Store

The is a facility to service the needs of the guests (shape = triangle). They are split into cafes (colour = orange) for the thirsty guests, restaurants (colour = yellow) for the hungry ones and kiosks (colour = red) for the guests that need a bit of both.

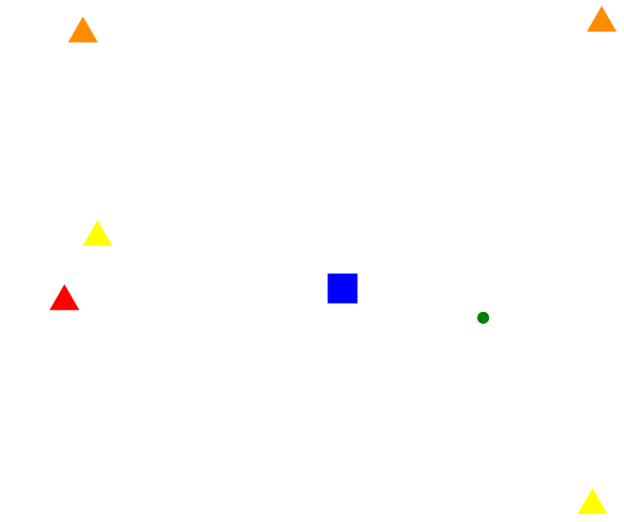
Implementation

Four separate files (Festival1.gaml, Festival2.gaml, Festival3.gaml and Festival4.gaml) have been created. Festival1 is the most basic implementation (guests without brain), Festival2 is an implementation with guests with brain (that means they can chose to explore or go to a facility they already know), Festival3 is an extension of Festival1 to include a security guard capable of removing guests that are evil, and Festival4 enables the information centre to propose the closest facility to a guest, thus reducing the traveling times significantly.

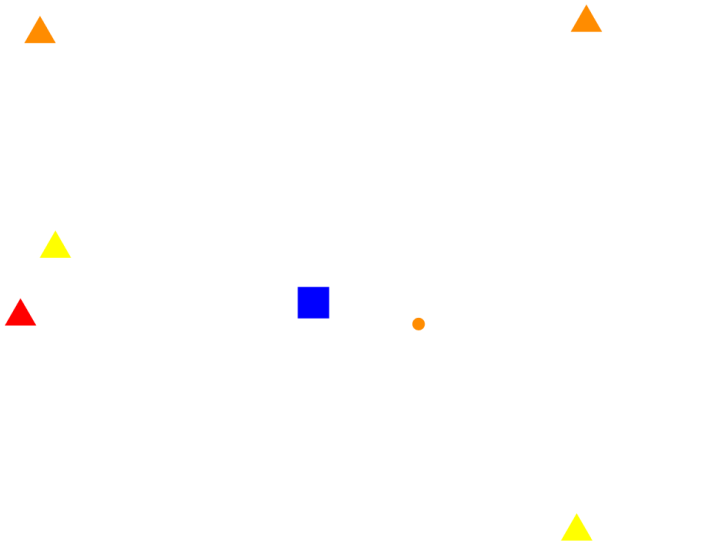
In order to calculate the number of facilities (cafes, restaurants and kiosks) on the map we decided upon a “threshold” solution. Let me explain:
We consider N to be the number of agents of type Store. We can then picture a list like this: [Cafe1, Cafe2, Restaurant1, Restaurant2, Kiosk1]. Threshold 1 (thr1) would then point to Restaurant1 (basically where we change from Cafes to Restaurants, and threshold 2 would point to Kiosk 1 similarly.
Therefore, when we traverse this list we can go through all the Cafes by using the indexes from 0 to thr1-1. For Restaurants: thr1 to thr2-1. And for Kiosks from thr2 to N-1.

Results

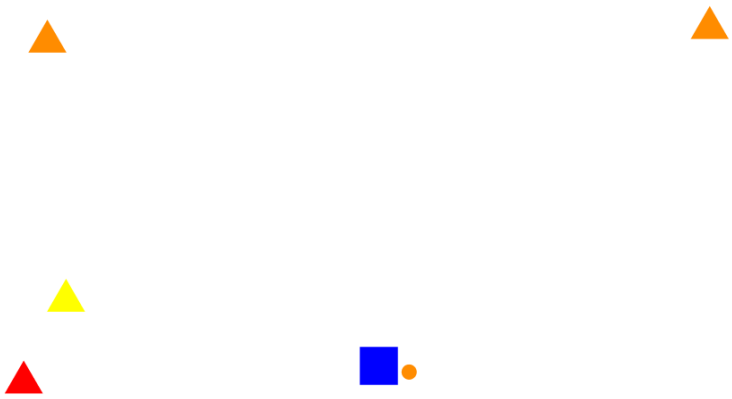
In order to demonstrate that our solution works properly we will set the number of guests = 1 and see what this guest does.
Initial state:



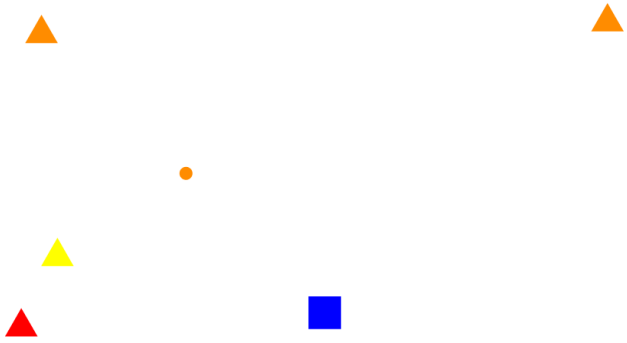
After flipping to thirsty:



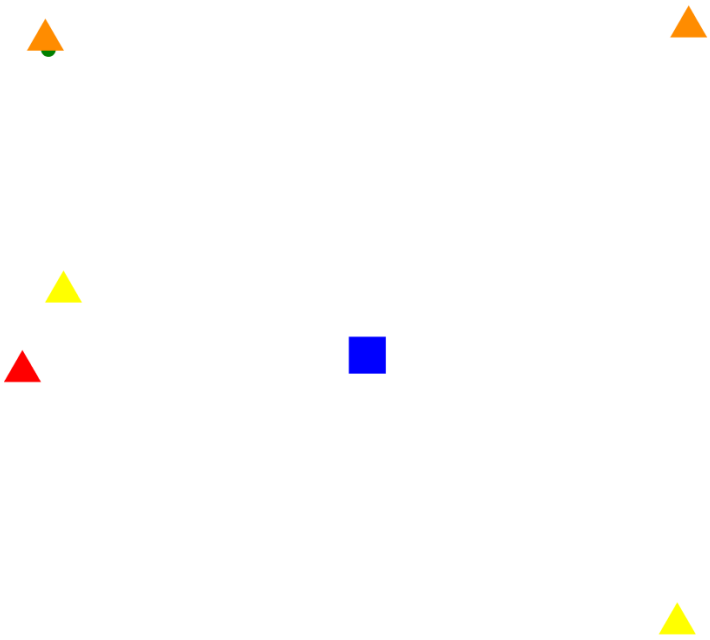
Travelling to the information center:



Travelling to a Cafe:



After drinking:



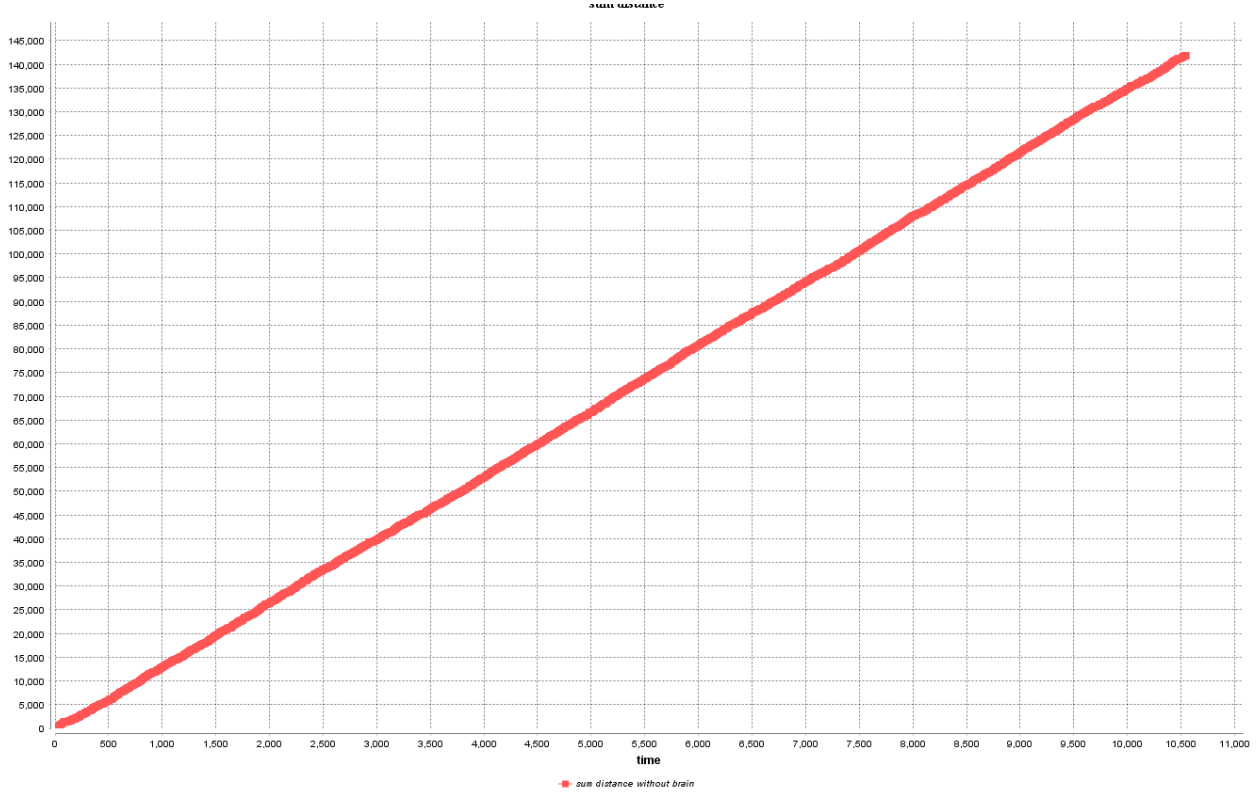
Challenge 1: Memory of Agents - Small brain

To complete the first challenge, we added 3 lists with points at the Guest agents. One that keeps the visited restaurants, one that keeps the visited kiosks and one that keeps the visited coffees. Every time a guest visits one of these shops, the guest will save its point at the corresponding list.

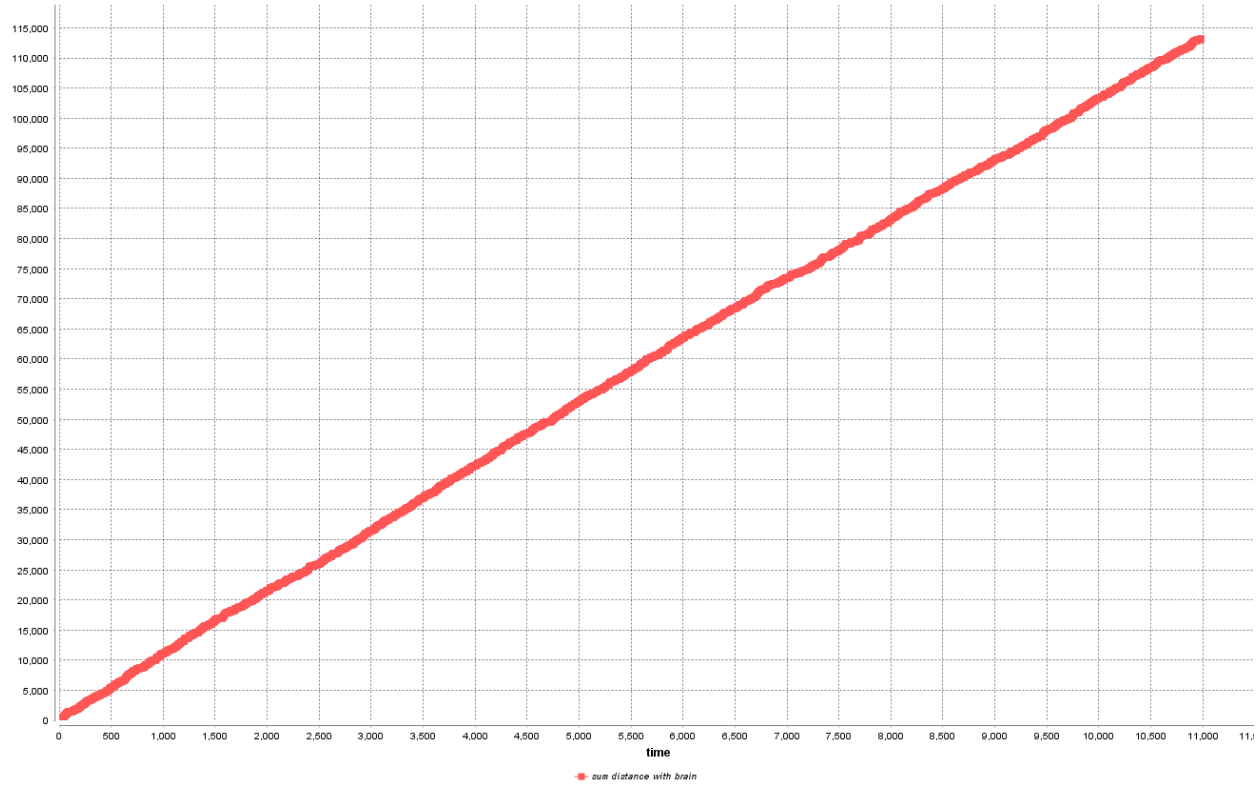
We also add extra functionality at the timePass procedure. When a guest is hungry or thirsty and has visited at least one shop and hasn't visited all the shops, it will choose to discover a new shop with probability 50% else it will choose to visit again a shop from the corresponding list. When a guest wants to discover a new place, it sends the list it keeps to the Info Centre and the InfoCentre returns one shop which hasn't been visited before by this guest.

In order to compare the distances of the implementation with the brain and the implementation without the brain, we create a new display called chartWithDistances. We also add a variable at the Guest agents called distance which keeps the distance each specie has been traveled. So the graph shows the sum of the distances of all guests.

The graph without brain:



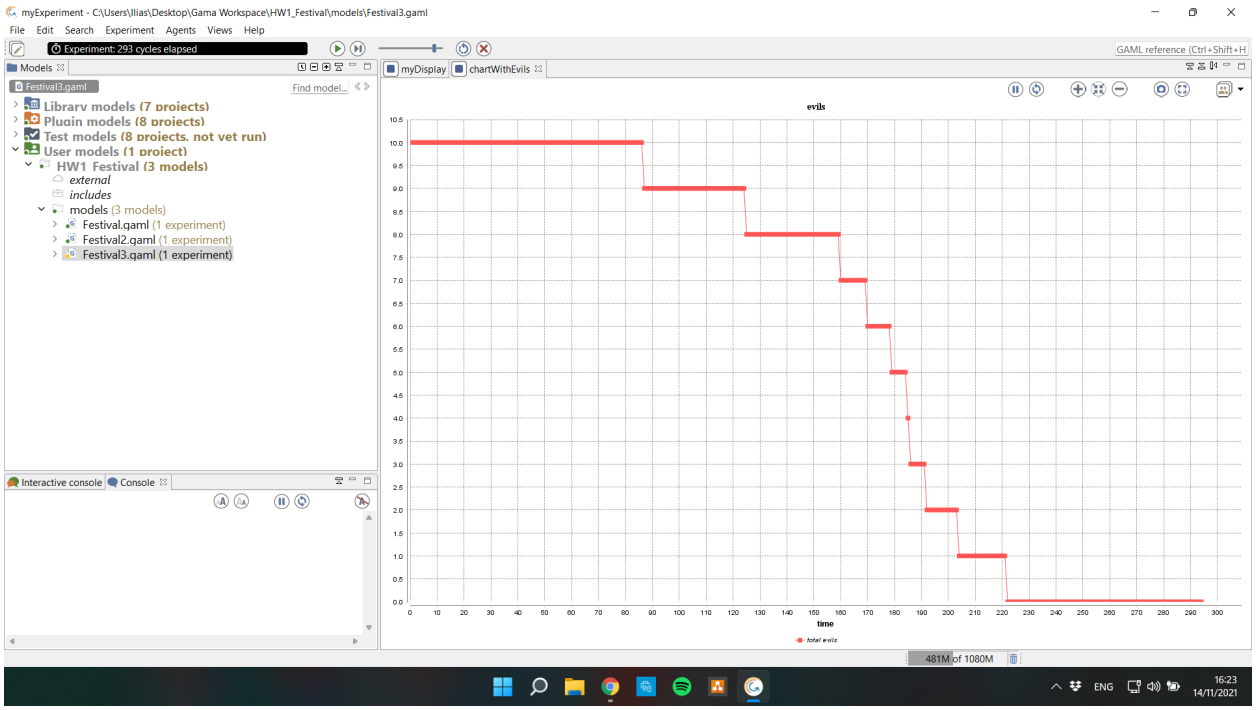
The graph with brain:



We notice that at the cycle=10.000 the total distance traveled without brain is around 135.000 steps while with brain is around 100.000 steps.

Challenge 2: Removing Bad behavior Agents

We added an extra variable at the guest agent called isEvil. When it is true the guest is evil. The infoCenter has a reflex called findEvils which runs when evils come close to the infoCentre. We call security for each of the new evils. CallSecurity is an action which calls the security agent and adds to its list the evil guest. We have implemented a security agent which keeps a list with the evils and moves towards the first evil in the list (followEvil reflex). When the security comes close enough to the first evil, the security kills the evil and removes it from the list of evils. Then, the security starts following the next evil in the list.

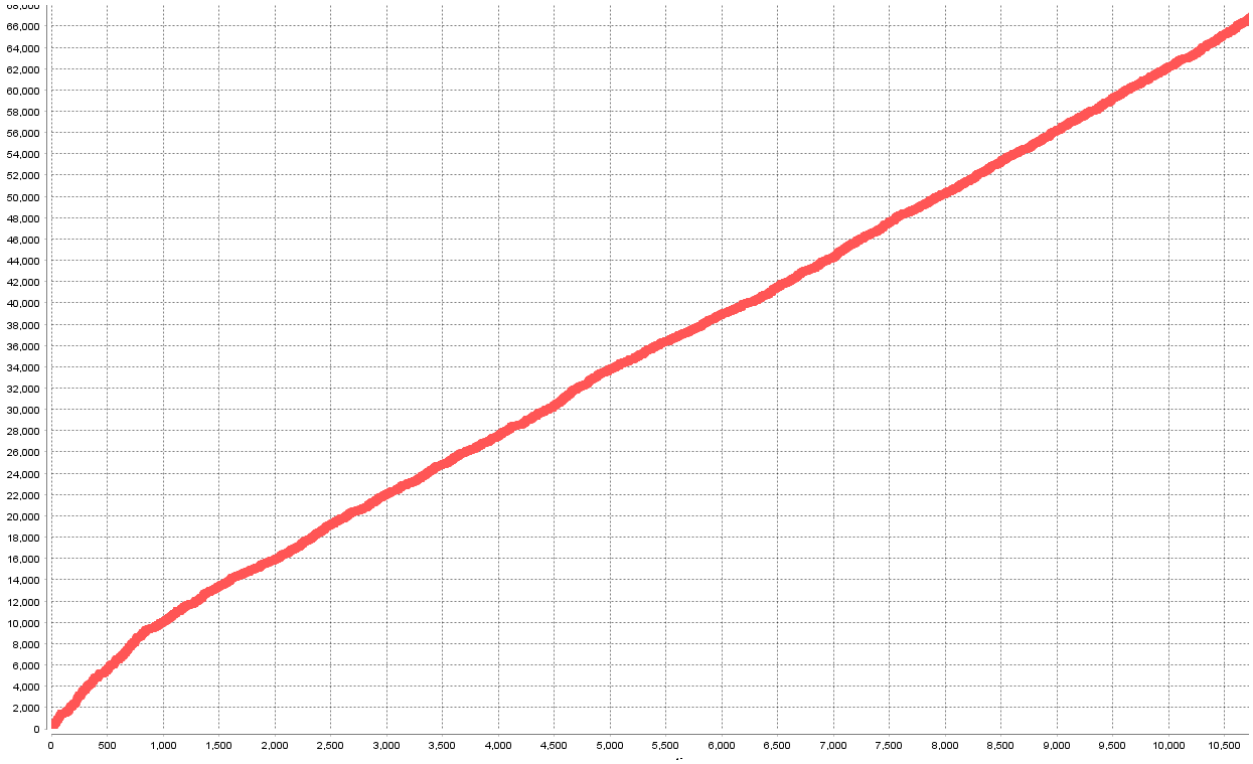


Creative Implementation

For our creative solution, we added a new procedure called findTheBestPoint which takes as input a list of points and returns the best one. The best one for us is the nearest point but someone could change this procedure and decide the best point in a different way.
We use this procedure in order to reduce the time an agent travels. We copied the code from the first challenge (Festival2) and added the necessary functionality. We tried to improve even more the distance traveled by not choosing a random store but when an agent chooses to visit a previously visited store, the agent will choose the nearest store.

This idea can be extended even more. For example, after a guest visits a store he should rate it. So next time we could have a function which will compare the distance and the rating of each store and decide the best store in a more complex way based on the guest's preference.

The result is shown below:



At cycle=10.000, the guests have traveled only around 66.000 steps.

Qualitative/Quantitative questions	**Answer**
Time spent on finding and developing the creative part	1hr
In what area is your idea mostly related to..	optimisation
On the scale of 1-5, how much did the extra feature add to the assignment?	2/5 (guests' travel distance reduced by 40%)
On the scale of 1-5, how much did you learn from implementing your feature?	1/5

Discussion / Conclusion

Initially the lack of extensive documentation of the Gama platform on the internet (which does not compare to, say Java) was a problem as the differences in syntax were many. However the relatively reasonable and intuitive structure of modules, which comes very close to how a natural human would think of (reflexes, actions, etc), was easy to catch up to.

Seeing how this platform can be used to simulate other behaviours beyond the scope of this project, it was a very interesting experience that added to our skillset greatly. We are keen on working on it further.