Assignment 1 - GAMA and Agents

Group 40

Sara Moazez Gharebagh saramg@kth.se

Sebastian Lihammer lihammer@kth.se

16 November 2021

In this assignment, we were tasked with creating different types of agents; Stores, Guests and an Information Center. Each type of agent has different behaviors and together they simulate a festival.

1. *How to run*

Run Gama 1.8.1 and import the Festival folder as a new project. To run the basic simulation, use *Festival.gaml*. Press main to run the simulation. We recommend changing the speed to be slightly lower than maximum to better see what is happening. To run the challenge 2 version, use *FestivalChallenge.gaml* instead.

2. Species

2.1. Guest

Guest agents move around (wander) until they get either hungry or thirsty. Each guest starts with a set value for their hunger and thirst, which is then incremented at random until it reaches a threshold, whereafter the guest makes their way to the Information Center. After getting the location of a store, the guests then move there. When they reach the store, their hunger or thirst is reset to 0.

The important variables involved in this process are the floats *thirst* and *hunger* (which keep track of their respective values) and the boolean *knowledgeOfStore* (which determines if they have been informed of the store's location). The most important reflexes are *moveToTarget* (which makes them move to a selected location), *moveToInfoCenter* (moving to the InfoCenter if hungry/thirsty), *goToStore* (go to the store) as well as *getDrink* and *getFood* (which reset the hunger and thirst values).

2.2. Store

There are two types of store agents: food stores or drink stores. There is very little code for these agents (only their location and colors are set) since the store-associated functionality is handled by either the Information Center (giving the location of the store) or the Guests (getting a drink/getting food).

2.3. Information Center

Information Center agents are responsible for informing guests of the location for the (nearest) food store when a guest is hungry or the (nearest) drink store when a guest is thirsty. The important variables of the Information Centers are *drinkstoreloc* (the location of the nearest drink store), *foodstore1loc* and *foodstore2loc* (the locations of the nearest food stores). The Information Center does not have any reflexes of its own, but is involved in the *moveToInfoCenter* reflex of the guests. Here, guests ask the Information Center for the nearest store and the Information Center, depending on what is asked for, provides the location of the nearest drink store or one of the nearest food stores.

3. Implementation

We started developing the store agents (4 food stores and 2 drink stores) and the information center agents (2). Firstly we decided the shape, color and the location for these agents. We decided to place two food stores and one drink store near each information center. The food stores are black squares and the drink stores are grey squares. The information centers are orange triangles. Then we developed the guests (40) by first deciding their shape (circle) and color. Guests can have 4 different colors; their default color is green, if they are hungry they will turn purple, if they are thirsty they will turn blue and if they are both thirsty and hungry they will turn pink. All guests have a randomly determined fixed information center that they will go to when they are hungry, thirsty or both.

4. Results

The text below is an excerpt from the log of the simulation, which shows the messages printed when guests are moving to the information center and when they receive the information.

```
Guest moving to info center
Guest moving to info center
Guest at info center 2
Guest asks info center 2 for drink store
Guest moving to drink store 2
Guest asks info center 2 for food store
Guest moving to food store 4
```

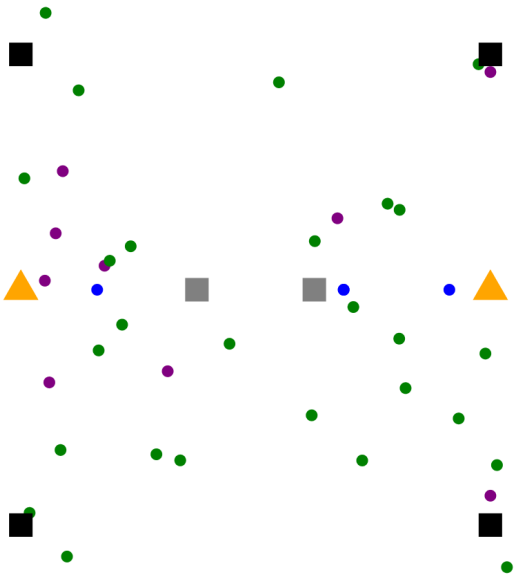

Figure 1: Screenshot of the simulation

5. Challenge 2

In the implementation for challenge 2 there are two new types of agents: bad guests and security guards. Bad guests are red circles and security guards are larger black circles. The bad guests have a random chance of going to an Information Center and behaving badly, whereafter the security guard will be alerted of their location, travel to them and kill them. A function was added to the Information Center that adds a bad guest to a list in the security guard. When a bad guest is killed (through the *die* function), the security guard removes that guest from its list.

The text below is an excerpt from the log of the simulation, which shows the messages printed when the Information Center detects a bad guest and informs the security guard, and when the security guard is tracking down the bad guest and then kills them.

```
Info center 1 informing security guard of bad guest
Bad guest behaving badly at info center 1
Tracking down bad guest
Tracking down bad guest
Tracking down bad guest
Tracking down bad guest
Tracking down bad guest
Tracking down bad guest
Bad guest killed
```

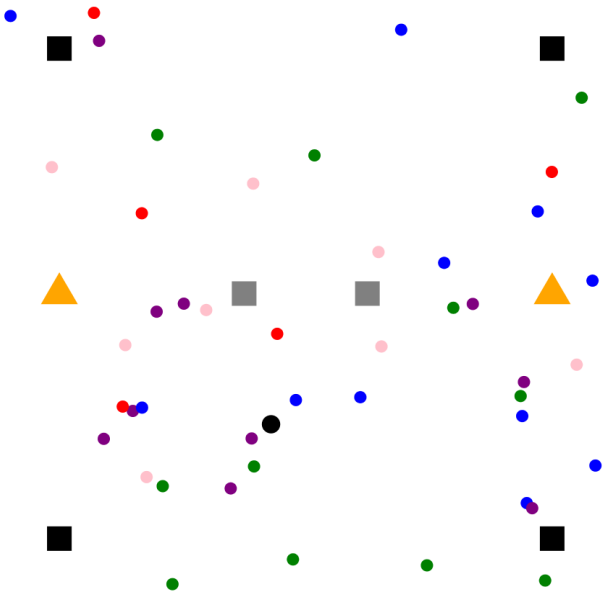

Figure 2: Screenshot of the simulation with Challenge 2 implemented

6. Discussion/Conclusion

The implementation of the assignment went well overall. Since none of the group members have worked with GAML before, it was a bit tricky to figure out how to implement some functions, mainly because we had trouble finding much documentation on the Internet. The security guard agent for challenge 2 was a bit hard to implement, since it had to move to a moving target and thus had to know where its target was at all times. However, both members agreed that the assignment was both fun and a good introduction to GAML.