Distributed Artificial Intelligence and Intelligent Agents
ID2209

Assignment 1

Group 22
Ayushman Khazanchi
Vasigaran Senthilkumar
16.11.2021

Introduction

In assignment 1 we were tasked with creating a basic festival with three types of agents: guests, stores, and an information centre. We were tasked with creating and running a simulation where our guests could interact with the information centre as well as the different types of stores. For our stores we went further and created three types of store agents: pub store for when guests are thirsty, foodstall store for when guests are hungry, and tavern for when guests are both thirsty and hungry together.

Later on as part of our challenges we implement a small brain in each of our guests that helps in navigating to known stores and also implement a security guard to make our festival safer. Both these challenges are explored in detail further ahead.

How to run

Note: This section assumes user knows how to run Gama IDE and how to run experiments within the IDE.

Main task can be run from festival.gaml.
Challenge 1 can be run from brain.gaml.
Challenge 2 can be run from security.gaml.

Species

Guest Agent

The guest agent is a species of type guest store. The guest agent has the skills of moving and they define two important properties of isThirsty and IsHungry. The guest agent has multiple reflexes which allow it to either “wander” or seek information from the information store or go to another store to fulfill its needs. It is represented by a circle of size (1).

Information Centre Agent

Information center agent is a species of type information centre. There is only one information centre in our implementation, and it is responsible for informing guests of other types of stores like pubs, taverns, or foodstalls. The information centre also is responsible for identifying bad apples and alerting security about the presence of bad apples and their locations. It is represented by a square of size (10).

Pub Agent

Pub agent is a species of type store where a guest can go and relieve their thirst. It is represented by a rectangle of size (5,3).

Tavern Agent

Tavern agent is a species of type store where a guest can go and relieve their thirst and hunger together. It is represented by a rectangle of size (7,4).

Foodstall Agent

Foodstall agent is a species of type store where a guest can go and relieve their hunger. It is represented by a rectangle of size (5,3).

Security Agent

The security agent is a species of type security who has the special ability to "kill" a guest that is behaving badly. The security agent is only invoked a guest is identified as a bad apple.

Implementation

Our main festival task is located in festival.gaml. Simply running the festival.gaml file from the IDE should run the basic simulation where guests can interact with the different stores and with the information centre. In the main festival task the guests are initialized at random spots in the festival and can roam around ("wander") as they enjoy the festival. Each guest has two counters for thirst (isThirsty) and hunger (isHungry) that slowly decrement down to zero. When either or both counters reach zero or negative values, the guest proceeds to the information center to find out the location of either the pub (isThirsty<=0) or the foodstall (isHungry<=0) or the tavern (both <= 0). Once it receives the location it uses the goToTarget reflux to proceed to its intended store. Once the guest gets to the store the respective counters reset to 100 and the guest goes back to enjoying the festival by "wandering" around in it through the enjoyTheFestival reflex.

At the information center the guest is sent only to the relevant store. If they're only thirsty they are sent to the pub. If they're only hungry, they're sent to the foodstall. If they're both hungry and thirsty, they're sent to the tavern. In the main implementation the location they're sent to is chosen at random but is always decided after the guest reaches the information center.

Challenge 1

In challenge 1 we implement a small brain in each guest to remember the different locations of the stores that it has been to before. This is implemented simply by using a list data type that stoers the locations for each type of store. Initially when the simulation starts, all guests have an "empty" brain i.e. they're initialized with an empty list. When the brain is empty, naturally, the guests proceed to the information centre for the location of stores they're looking for. When they get the store location, they save it to their memory. This is done by adding the store location index to the relevant list of that agent. In the future if the guest is looking for a store, it first checks whether there are values in its brain, i.e. whether it has been to that particular type of store before. Assuming that a value is present for the particular store, the guest can then directly go to that store instead of going to the information center again. This emulates a "small brain" or "small memory" which the guest can use in its activities.

However, we interpreted the assignment to add some excitement to the guest where the guest may occasionally want to find some new store to go to instead of the ones it already

knows of. We do this by randomizing the times that a guest “uses” its brain or chooses to “disregard” its brain and go to the information centre again to maybe find something new. We output this in our code as well by showing that the guest is aware that it has a value in its brain but is still choosing to disregard it to maybe find some new locations.
Example output statement of the above case:

“write guestName+ " brain is available but i'm not using it - going to info centre";

Challenge 2

Challenge task 2 creates a safer environment in our festival by introducing a security guard agent who ensures that bad guests (“bad apples”) are removed from the festival as and when the security guards are informed to do so by the information center. The information center checks each guest when they visit it whether they are misbehaving characters or not. If the guest is identified as a misbehaving character then the information center runs an “ask” interaction to the security agent and provides it the identifying index of the guest as well as the guest’s next location. In our implementation, guests are randomly assigned to become misbehaving characters when they visit the information center.

The security agent using a reflex action keeps a tab on whether any guests have been identified as bad agents and if they are then the security agent sends out an alert and starts moving towards the target location of the misbehaving guest. Since security is important, the security agent moves at a speed much faster than the guest. When the security agent reaches the location and the guest also reaches the location, the security agent “kills” the bad guest by invoking a die interaction on the guest agent.

Sometimes the guest can actually escape being killed if they do not arrive in the target location in time and in the meantime another bad actor has appeared in the festival.

Creative implementation

We feel that we deserve an extra bonus point for making our festival implementation a little bit more well-rounded and beyond what is expected of the base assignment and challenges.

In our implementation we have taken the liberty to be a little creative with how our guest agents are setup. Each guest has a counter that decreases from 100 for their thirst and hunger. We felt that thirst and hunger are something that happen gradually and not “instantly” and as a result went with a depleting counter as opposed to a Boolean value. This also allows some of our guests to enjoy the festival longer.

For our brain challenge we had three different memory lists in each guest for the three different types of stores in our festival (pub, foodstall, tavern). A guest that has been to a pub before will only know of its earlier pub and not of any foodstalls or taverns (and thus those memory lists remain empty).

For security we added a custom high-speed to our security agent which allows security to act fast on each threat as it is the case usually in real-life situations.

Results

Our basic implementation of festival works as expected. Both our challenges also satisfy the basic requirements of functionality and expectation.

For challenge 1 we went a bit further to explore how having a brain might help in the distance covered by each guest. We tested the scenarios where a guest

1. Never refers to its brain
2. has a brain that it first builds and then uses all the time
3. has a brain but still randomly may choose not to use it

Theoretically, if a guest were to build a brain map of all locations at the very beginning and then rely only on the brain map to access the shortest paths the distance travelled would be a lot less than the distance travelled in all other scenarios.

However, in our implementation, there are many variables that we intentionally do not control such as the initialization of each guest in random spots with random values of thirst and hunger. These randomizations along with the initial brain-building efforts that require going to the information centre for each type of store end up creating a lot of distance that we don't necessarily control.

Keeping these assumptions in mind, we ran simulations of 500 cycles for each test case above by toggling the brain on and off as well as toggling the randomization of choosing or not choosing the brain.

Scenario 1: Never refers to its brain. The guest always goes to the information center whenever it needs information on the stores. This test case produced a total distance of 178 steps travelled in 500 cycles as shown in Figure 1 below.

```
person 1 my small pub brain [0,0]
person 1 my small tavern brain []
person 2 brain is available but i'm not using it - going to info centre
Distance traveled so far - 176
person 1 on the way
person 1 my small food brain [1]
person 1 my small pub brain [0,0]
person 1 my small tavern brain []
person 2 brain is available but i'm not using it - going to info centre
Distance traveled so far - 177
person 1 on the way
person 1 my small food brain [1]
person 1 my small pub brain [0,0]
person 1 my small tavern brain []
person 2 brain is available but i'm not using it - going to info centre
Distance traveled so far - 178
```

Figure 1

Scenario 2: The guest first spends time building a brain. This is done by going to the information centre for getting the location of different types of stores it needs. Once it has built the brain map of different stores it then only relies on the brain to go to those stores in the future when they're needed. This implementation relies a lot on the speed of the guests counters depleting and is also affected quite a bit by the randomization of building the brain map (for example, it may not be for a long time that the guest is hungry and thus the

foodstall map may not be built for a long time). This test case produced a total distance of 183 steps travelled in 500 cycles as shown in Figure 2 below.

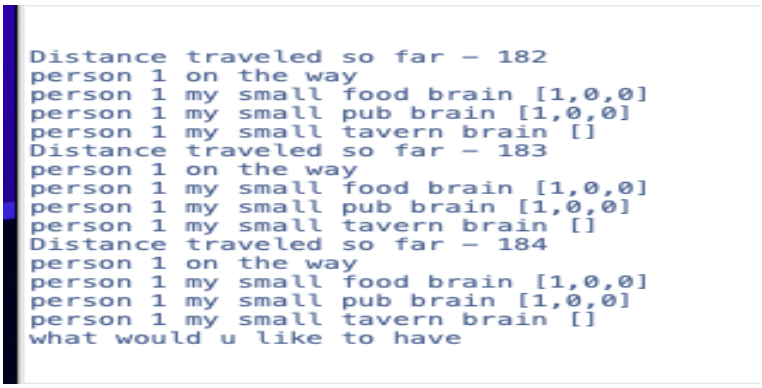

Figure 2

Scenario 3: This is a bonus test case we did to replicate our implementation scenario where a guest may have a brain map that it can use of prior locations it has been to. However it may still choose to disregard the prior locations in its brain in order to find something new. We did a test case for this to see how the distance travelled might change when a guest randomly chooses between the information center and its brain. This test case produced a total distance of 183 steps travelled in 500 cycles as shown in Figure 3 below.

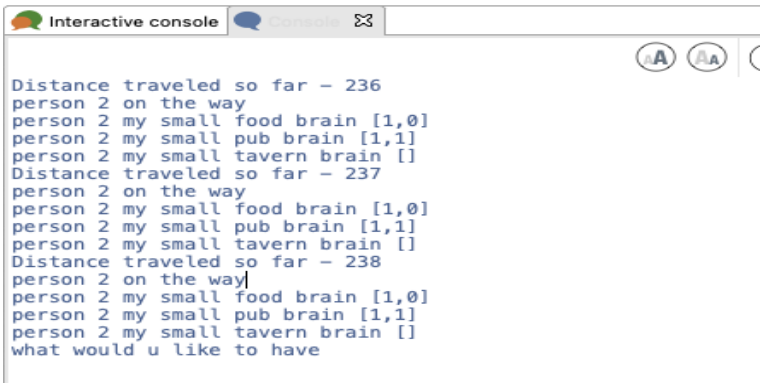

Figure 3

Overall we cannot conclude on what is more effective approach in such a short cycle of test cases. However, over a much longer test cycle we feel that our approach to implementing a small brain will produce less distance travelled by each agent. This can of course be influenced by the randomization in our initialization but with sufficient test cycles and some control variables we can effectively reduce the distance travelled for each guest. In this way we have to end up sacrificing the guest discovering “new” locations since they will always choose to use their brain and never go to the information centre after the brain is available.

Discussion / Conclusion

Overall, we enjoyed working on the simulation. We were able to understand the assignment well and had room for creativity in our interpretation of the brain and distance challenges as mentioned above. However, we struggled quite a bit with the lack of documentation around the GAML language and in some aspects of the IDE. We were unable to implement things in what we felt might have been a clearer and more modular code structure because we were unable to find good documentation around some function and api calls in the language.