

ID2209 – Distributed Artificial Intelligence and Intelligent Agents

Assignment 1 – Agents & stuff

Group 13

Adele Robaldo

Karthika Satheesh

17/11/2023

Overview: Basic Festival with Stores

In this assignment, we developed a multi-agent simulation in the GAMA platform, focusing on the dynamics of agent interaction within a party environment. The core objective was to simulate the behaviour of festival attendees (Person agents) as they engage with various points of interest, such as FoodStore and DrinkStore agents. We made sure that the different agents looked different in shape and colour, and we added them to a simulation to make them interact with each other.

A critical aspect of the assignment was to implement the SecurityGuard agent tasked with maintaining order by identifying and removing Person agents exhibiting bad behaviour, as reported by the InfoCenter.

How to run

To run the simulation model developed in this assignment, please follow these steps:

- Open GAMA Platform version 1.7.
- Import the provided project files into GAMA by selecting File > Import, navigating to the 'General' folder, and choosing 'Existing Projects into Workspace'. Select the archive file and complete the import process.
- Locate the Project1_Party.gaml model within the imported project.
- Select the main experiment, which serves as the entry point for running the simulation.
- Press the 'Play' button or 'Start' to initiate the simulation run.

Note: the implementation currently works for 3 InfoCenterPoints, but it can be increased/decreased using add/remove elements from the list *infoCenterPoints*.

Species

Agent Person

The main skill of this agent was “moving”. In fact, it wanders until the boolean values *isHungry* or *isThirsty* goes to true.

In fact, when *isThirsty* or *isHungry*, the agent moves to the target, which can be an InfoCenter or a Store.

If the agent goes to the InfoCenter, he will perform the behaviour *enterInfo*, which consists of asking the agent to suggest a place where to eat or drink and it will be given the point of the Store that satisfies its needs.

The behaviour *enterStore*, consists of asking the store if it still has drinks or food, to understand if it still has stock and to satisfy its need.

Parameter Tweaking:

The simulation behaviour can be influenced by adjusting the following parameters:

- Parameter 1 (e.g., *numberOfPerson*): Modifying this parameter will affect the number of Person agents within the simulation, impacting how crowded the festival appears and the interactions between attendees.
- Parameter 2 (e.g., *personSpeed*): Changing the speed at which Person agents move will influence the dynamics of reaching points of interest and the likelihood of encountering the SecurityGuard.

- Parameter 3 (e.g., prob_h): This parameter controls the probability of a Person becoming hungry or thirsty, thus affecting the demand for FoodStore and DrinkStore services and the overall activity level.

Adjusting these parameters allows for experimentation with different scenarios and observing the resulting behaviour of A (e.g., the movement patterns of attendees), B (e.g., the frequency of visits to stores), and C (e.g., the interaction with the InfoCenter and SecurityGuard).

Agent Store

This agent was responsible for providing food and drink to Persons.

Agent InfoCenterPoint

This agent provides the action checkStores, responsible for asking Stores if they have food or drink. In case the boolean value hasDrink or hasFood goes to true, the InfoCenterPoint asks the location of the Store and returns it.

This action will be called by the agent Person, when it goes to the infoPoint, and provides it with the location of the Store that has a stock of food and/or drink, depending on its needs.

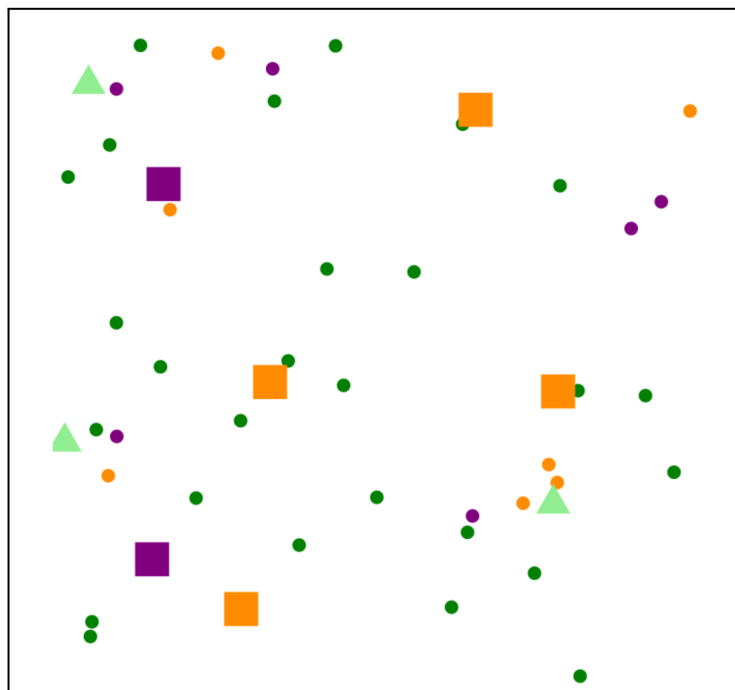


Figure 1: The info centres are represented as light green triangles. The stores are represented as squares, with food stores being purple and drink stores being orange. The Persons (guests) who are dancing are represented as green circles. Hungry people are purple circles, and thirsty people are orange circles.

Implementation

- We started developing Agent *Person* defining its aspect (colour and shape). The colour changes depending on its state: Hungry/Thirsty, or not Hungry and not Thirsty.
- Then we made its simple behaviour of wandering and going to the targetPoint (store or info point), depending on the state.
- When we had created Agent *Person*, the next logical step was to create the other two agents, *InfoCenterPoint* and *Store*.
- Both were defined in aspect: an unchanging black triangle for the *InfoCenterPoint* and a square for the *Store*, the last one with changing colours. The colour changed with a certain rate depending on the stocks of food and drinks of the *Store*.
- The next step was to create a behaviour to make the *Person* ask the *InfoCenter* the new target point (Store with food or drinks to satisfy its needs).
- Then, we created an action to make the *InfoCenter* ask *Stores* if they had food and or drinks depending on what the *Person* has asked.
- A similar implementation was done to make *Person* ask *Stores* if they had food/drink, in order to change their state from Hungry/Thirsty to notHungry/notThirsty.

Results

The results of our work can be seen by running the program on GAMA.

As you can see, messages have been printed to understand the different stages of agent interaction.

```
Person 29: Can you suggest me where to eat / drink Info 2
Person 29 can drink at store {95.69402472524841,75.32724544472612,0.0}
Person 29 can drink at store {68.64224790144624,92.03170068689242,0.0}
Person 29 can drink at store {95.69402472524841,75.32724544472612,0.0}
Person 29 can drink at store {68.64224790144624,92.03170068689242,0.0}
Person 29 can drink at store {95.69402472524841,75.32724544472612,0.0}
Person 29 can drink at store {68.64224790144624,92.03170068689242,0.0}
Person 29 can drink at store {95.69402472524841,75.32724544472612,0.0}
Person 29 can drink at store {68.64224790144624,92.03170068689242,0.0}
Person 29 can drink at store {95.69402472524841,75.32724544472612,0.0}
Person 29 can drink at store {68.64224790144624,92.03170068689242,0.0}
Person 29 can drink at store {95.69402472524841,75.32724544472612,0.0}
Person 29 can drink at store {68.64224790144624,92.03170068689242,0.0}
targetPoint is :{68.64224790144624,92.03170068689242,0.0}
Person 29 drank at store Drink Store 3
```

Figure 2: Example of sentences printed. The agent *Person 29* asks the *Info Center 2* where it can eat or drink. Then, all the different store locations are printed, and the store closer to the *InfoCenter* will be the *targetPoint*. At the end, it informs you that the *Person* drank at *Store 3*.

Challenge 1

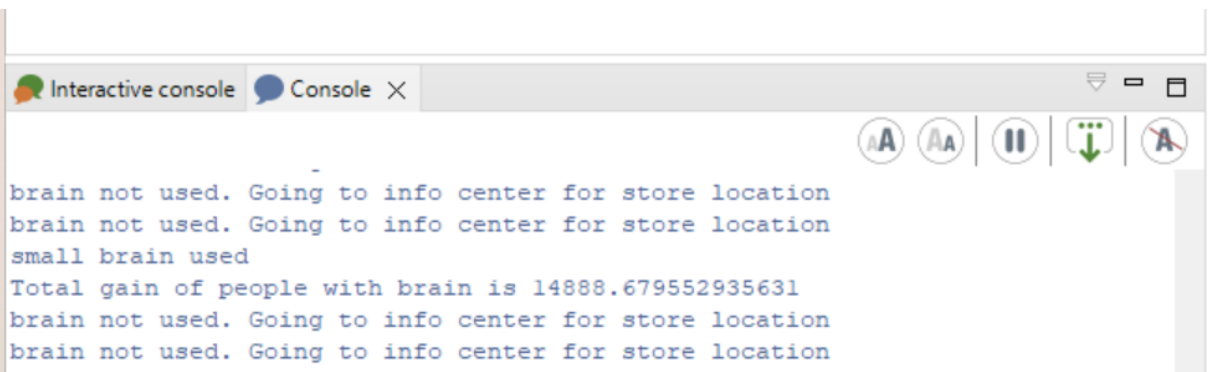
To complete Challenge 1, we enhanced Agent Person with a memory capability, represented by the visitedFoodStores and visitedDrinkStores lists. These lists store the coordinates of the FoodStore and DrinkStore agents that the Agent Person has visited. Through this modification, the Agent Person can recall previous visits and decide whether to return to a known store or explore new options. This behaviour is controlled by a probability switch, where there is a 50% chance of the Agent Person choosing to use their memory to revisit past locations instead of seeking out new ones.

Furthermore, in addressing Challenge 1, we not only enhanced Agent Person with a rudimentary memory function but also implemented a mechanism to quantify the efficiency gain achieved through this memory utilisation. Specifically, the memory function is simulated via the visitedFoodStores and visitedDrinkStores lists within Agent Person, which store the coordinates of FoodStore and DrinkStore agents visited previously.

To quantify the efficiency gained from using memory (referred to as 'small brain'), we introduced a variable named gain. This variable calculates the potential distance saved when an Agent Person opts to use its memory to revisit known locations instead of heading to the InfoCenter for new recommendations. The gain is computed as follows:

- When Agent Person decides to use its memory, it calculates the distance it would have travelled to the InfoCenter and then to a new store (infoCenterLocation to targetPoint) and subtracts the direct distance to the remembered store (self.location to targetPoint).

This calculation is conducted for both food and drink needs, depending on whether Agent Person is hungry or thirsty, and the remembered store locations

A screenshot of a web-based interactive console. The window has a title bar with 'Interactive console' and a close button. Below the title bar is a toolbar with icons for font size (A, A+), a pause button, a download button, and a refresh button. The console area displays several lines of text in a monospaced font, with some lines highlighted in blue. The text includes: 'brain not used. Going to info center for store location', 'small brain used', 'Total gain of people with brain is 14888.679552935631', and another 'brain not used. Going to info center for store location'.

Challenge 2

To complete this challenge, we added a new agent called SecurityGuard

Agent SecurityGuard

Agent SecurityGuard fulfils the role of maintaining order within the simulation. Its primary function, executed through the catchBadGuest reflex, is to respond to reports of misconduct of Agent Person individuals. When Agent Person exhibits bad behaviour (defined by the isBadGuest variable) and visits the InfoCenter, Agent SecurityGuard is alerted to intervene and remove them from the party.

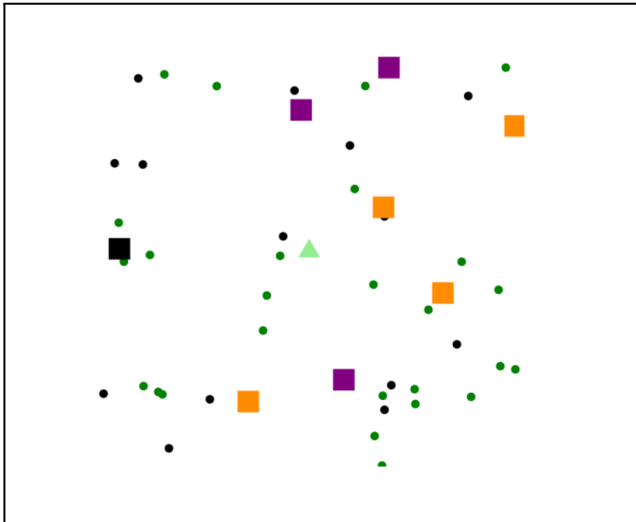


Figure 1 Challenge 2: The security guard is represented by the black cube, and the badGuests are represented as black circles.

The action for Agent SecurityGuard is triggered when the Infocenter reports a Person and the Agent SecurityGuard navigates towards the reported Person using the goto action, at a speed governed by the securitySpeed variable, which is notably faster than the Agent Person's speed. Upon reaching the offending Person, Agent SecurityGuard employs the killBadGuest reflex, removing the misbehaving agent from the simulation. This interaction highlights the responsive and corrective capabilities programmed into Agent SecurityGuard to ensure the party's orderly progression. It demonstrates the die function in Gaml.

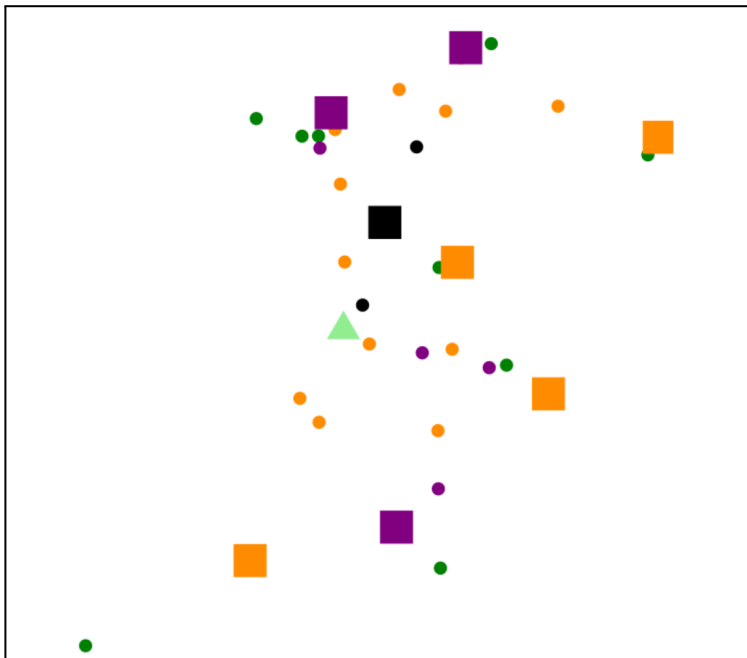


Figure 2 Challenge 2: You can see that the number of Person(guests) are less due to the Agent killing bad guests.

Creative implementation

- Multiple info centre points
- Name for log messages

<i>Qualitative/Quantitative questions</i>	<i>Answer</i>
Time spent on finding and developing the creative part	2 hours
In what area is your idea mostly related to...	Info centre points and log messages
On the scale of 1-5, how much did the extra feature add to the assignment?	2
On the scale of 1-5, how much did you learn from implementing your feature?	3

Discussion / Conclusion

Positively, the assignment was beneficial in imparting knowledge on creating agents and facilitating their communication. The most engaging aspect was the implementation of the challenges and in particular the creation of a memory mechanism for the agents.

This segment highlighted the immense potential such functionalities hold in practical real-world applications.

Overall, the assignment provided a practical understanding of agent communication and we enjoyed implementing it.