# D2209 – Distributed Artificial Intelligence and Intelligent Agents

# Assignment 1 – GAMA and Agents

Group 29

Muhammad Jahangir Zafar
Javier Ron

13.11.2019

# Introduction

In this assignment we were task to make a small experiment in GAMA that resembles a music festival. The agents involved in the experiment were of types: Guest, Information Center and Food/Drinks stand, and they had to interact according to the rules defined in the assignment description file.

# How to run

Run GAMA 1.8 and import file:

- `hw1-1.gaml` for implementation of mandatory task.
- `hw1-4.gaml` for implementation of mandatory task plus the two challenges
- `hw1-5.gaml` for implementation of mandatory task plus the two challenges plus the creative addition.

# Species

Guest: The guest attending the music festival; It has several states which include: dancing, hungry, thirsty, looking_for_guard, back_to_dancing, misbehaving; some of these states have sub-states to implement the functionality where the Guest first goes to the Information Center.

Information Center: It is an agent responsible for providing coordinates for the guests, since they may want to know where the food stands, drink stands or security guards are.

Food Stand: This agent will provide a means to change the state of the guests from hungry to back_to_dancing, by modifying the agents' parameter.

Drinks Stand: This agent will provide a means to change the state of the guests from thirsty to back_to_dancing, by modifying the agents' parameter.

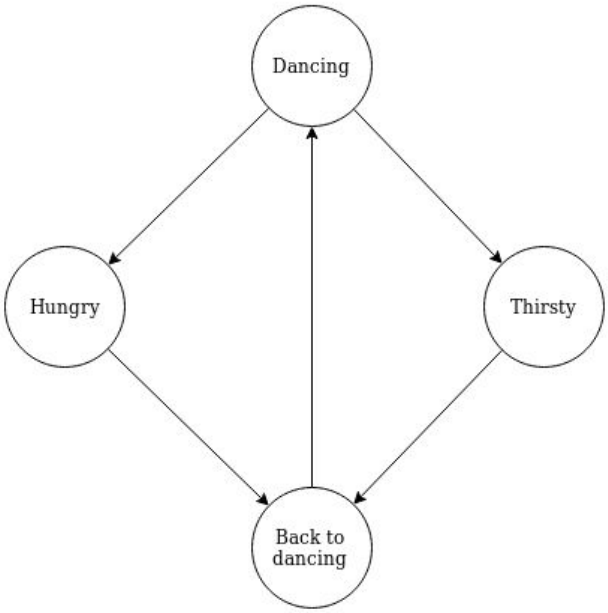
Security Guard (challenge): This agent will receive reports of misbehaving Guests, and follow them and remove them. It is also implemented with an internal state machine.

# Implementation

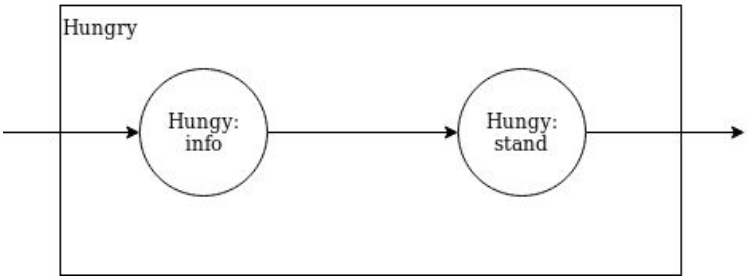
The most of the implementation was inside the Guest agent. This agent has an internal state machine that defines the behavior of the agent. E.g. on the dancing state it will wander, but when the hungry variable reaches a certain threshold it changes state, and now moves toward

the Information Center. The random behaviours are implemented with the help of the ‘flip’ function.

The Guest state machine is defined as follows:




The hungry and thirsty states have sub-states that interact as follows:




# Challenges

## Challenge 1: Memory

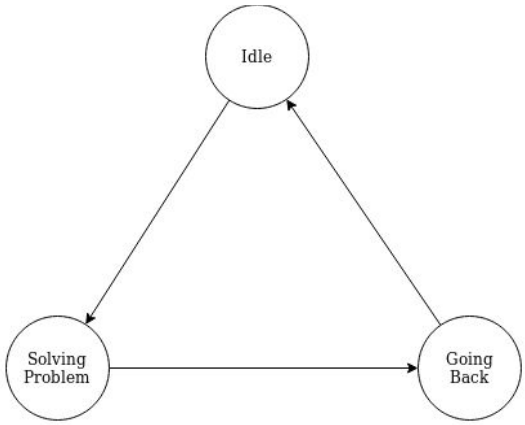
We implemented challenge 1 by storing the positions of the food/drink stands in the Guest after it visited any of them, and then if the agent got hungry/thirsty again it will first try to go to the

stored position if there is any, the trick is that we also implemented a random ‘forget’ reflex, which erases the saved positions, and forces the Guest to go back to the Information Center, making it possibly discover a new location.

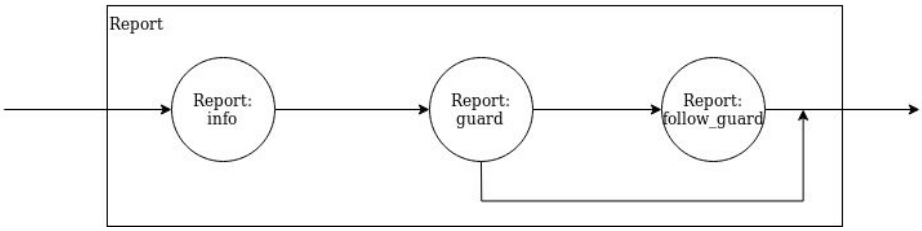
We also implemented inter-guest communication, this means that when a Guest is in need of information, and on it way to the information center, it will ask all nearby Guests if they have the information that it needs.

# Challenge 2: Security Guard

We implemented a new agent and a two new states for the Guest (report) that follows Guests that are misbehaving after another Guest reports them. It is implemented with a state machine.




Also the reporting Guest was added a new state with corresponding sub-states




# Creative challenge:

For the creative challenge we implemented two fun features related to the security guard and Guest interaction:

- The misbehaving Guest will now run away from the Guard.
- The Guard will now tackle the misbehaving Guest when it is in range.

We also looked around on how to use textures and modify the camera view for more visually appealing experiment.

# Results

Here we make a comparison between the experiment with Guests with memory and the experiment without memory. One important value is the amount of interactions with the information center, in a real-life system this central agent can be considered a bottleneck and thus reducing the amount of necessary interactions with it can be beneficial.

| | Without Memory | With Memory | Difference |
|---|---|---|---|
| 50 Guests | 2853 | 1265 | -55.66% |
| 200 Guests | 11411 | 1544 | -86.46% |

*Table 1: snapshot after ~10000 cycles.*

In Table 1 we can clearly observe not only that the difference is significant, but that the number of interaction in the implementation with memory does not grow as fast as the other implementation, even if the number of Guests is grows at the same rate.

# Conclusion

This assignment was an excellent way to learn about GAMA and how to use it, this includes species, reflexes, the ‘ask’ statement, and some moving functions.
Now that we have learned the basics we believe that we are ready to more complex experiments such as the negotiation, and coordination approaches seen in the lectures.