ID 2209 - Distributed Artificial Intelligence and Intelligent Agents

Project Report - Festival Simulation

Group 3
Albert Asratyan
Justas Dautaras
12.12.2019

Introduction

This project is a simulation of different behaviours of festival guests (agents) at a festival (environment). These guests possess different traits, such as age or wealth, and the agents themselves can be of different types. Some may prefer to party more than others, whereas the majority are just regular visitors without any concrete preferences. Different agents behave differently in different environments. For example, an agent with preference to party, will end up going to the party area most of the time. However, this does not mean that that agent would not like to enjoy some quiet time in the relaxation area. The agent might wish to go there, but not as often. The simulation has a total of 50 agents running/interacting at the same time. Each of these agents behaves in its own personal interests, thus making the whole simulation resemble an actual festival.

How to run

Run GAMA 1.8 and open the *Main.gaml* file. Press the green button '*my_experiment*' to run the simulation. The simulation consists of two main parts, the simulation itself with a 2D view on the 50 visitors, and a chart that plots the total sum of different activities that the visitors perform. If this approach does not work, try to create an empty project and copy paste the contents of the *Main.gaml* file into the created *Model1.gaml*. Then proceed normally.

When running the main simulation, only the first agent is going to log its actions in the console. This is done to not pollute the console when the simulation is running at high amounts of agents. Some parts of the output are also rendered on the simulation screen, and not only in the console. The requirements specify that the agents should be of at least 5 different types. To change the type of the agent, the variable *agent_type* should be changed. The options are: "*average*", "*party*", "*chill*", "*gambler*", and "*weirdo*". The options will be explained later in greater detail.

Approach

Even though the project could have been built upon the knowledge and code skeleton from the previous homeworks, it was decided that a clean start would be beneficial for the project. This has allowed us to have a completely new approach to the task, as we have tried to not repeat anything that has been done in the previous assignments.

Species And Grid

Grid

The image below shows the grid of the simulation without any spawned agents.

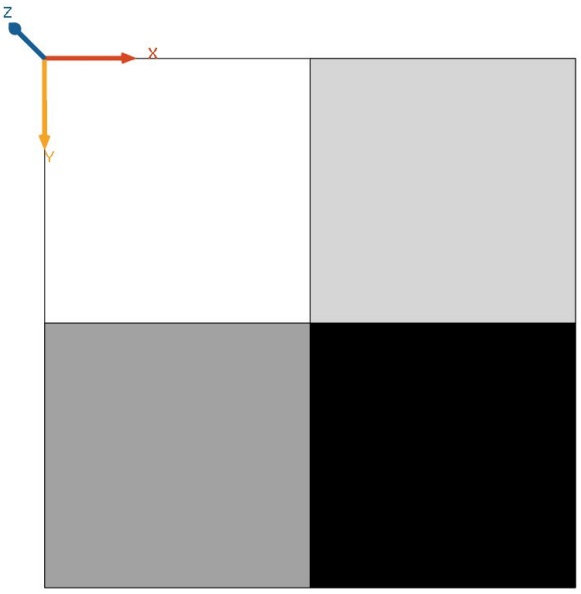

Image 1. The festival grid.

The grid consists of four segments (cells). Each of the segments represents a specific area with a designated role. The areas are: party area (white, top left), chill area (light gray, top right), food area (dark gray, bottom left), and gambling area (black, bottom right). Each of the segments represents an area to which the agents would need to travel in order to fulfill the party/chill/eat/gamble wish. The grid is mainly used by the agents to determine whether they have reached the area corresponding to their actions or not by checking the color of the grid.

Agent *visitor*

This is the only species type in the main simulation. However, the species has 5 main subtypes, as per requirements. The types of "*visitor"* are: "*average*", "*party*", "*chill*", "*gambler*", and "*weirdo*". Each of the agents is able to get a "wish". Wishes are used for determining what the next action of the agents will be. There are four types of wishes: "*party*", "*chill*", "*gamble*", and "*wander*". As it can be noticed from the names, some of the agents will have a preference, thus increasing the chances of getting a specific type of a wish. The agents are color coded accordingly to match their subtype. Their colors are:

- Average - #green (neutral)
- Party - #white (matching the "party" corner of the grid)
- Chill - #darkgray (matching the "chill" corner of the simulation)
- Gambler - #black (matching the "gamble" corner of the simulation)
- Weirdo - #red (to stand out visually due to the low spawn rate)

There is also a passive wish to "eat". Each of the agents has a food counter that goes down with time. When the counter reaches 0, the agent changes its wish to show that it needs to eat and stops doing whatever it was doing to go and fulfill that wish. Once that is done, the agent will return to whatever it was doing before it got hungry.

When an agent is fulfilling a wish, it will take some time to do so. This is done to represent a real life situation, because it would be rather weird if an agent decided to "party" or "chill", but

took only 1 time frame to do so. For example, an agent will spend 60 frames to wander or 30 frames to party.

The first visitor agent is always being tracked and his information is displayed and debugged to present all the functionality in an easy to understand way.

Image 2. Tracking of an agent.

Agent Probabilities

Each of the subtypes of the '*visitor'* agents has a certain predefined behaviour patterns. On top of this, to make the simulation more interesting, the program initializes the agent types based on the following percentages:

- Average - 41%
- Party - 18%
- Chill - 18%
- Gambler - 18%
- Weirdo 5%

There is no particular reason as to why these and the following numbers were chosen as such, apart from an attempt to resemble reality (at least roughly anyway).

Agent Type Description

Average - the most common agent type in the simulation. Understanding that it is at a festival, this type likes to party a good amount, but does not forget that it needs to rest and occasionally gamble a little for the fun of it. Therefore, an average visitor would have the following "*wish*" distribution:

- Wander - 60%
- Party - 25%
- Chill - 10%
- Gamble - 5%

Party - an agent type that prefers to party more than the rest. Being completely occupied with partying and only partying, this type does not gamble at all because it doesn't have time for such silliness. Therefore, this type has the following "wish" behaviour:

- Wander - 45%
- Party - 40%
- Chill - 15%
- Gamble - 0%

Chill - a type that prefers to hang out in the chill area, but doesn't mind performing other actions as well. When bored, this agent might gamble a little bit more than average for some peculiar reason. Therefore, the "chill" agent has the following "wishes":

- Wander - 40%
- Party - 15%
- Chill - 35%
- Gamble - 35%

Gambler - a type that prefers to be in the gambling area of the simulation. Obviously, loves nothing more than losing some money, but is up for occasional break or some partying, when getting weary of its financial losses. Therefore, the "gambler" agent has the following "wishes":

- Wander - 40%
- Party - 10%
- Chill - 10%
- Gamble - 30%

Weirdo - a type that prefers to wander around the festival rather than do anything in particular. However, even weirdos do get bored sometimes, so this type can occasionally choose to do something more meaningful than just walking around and doing nothing. Therefore, this type has the following "wish" distribution:

- Wander - 85%
- Party - 5%
- Chill - 5%
- Gamble - 5%

All of the agent types with their respective spawn rate are shown on the Image 3 below. As expected, the red (weirdo) agents are rare and green (average) agents are prevalent.

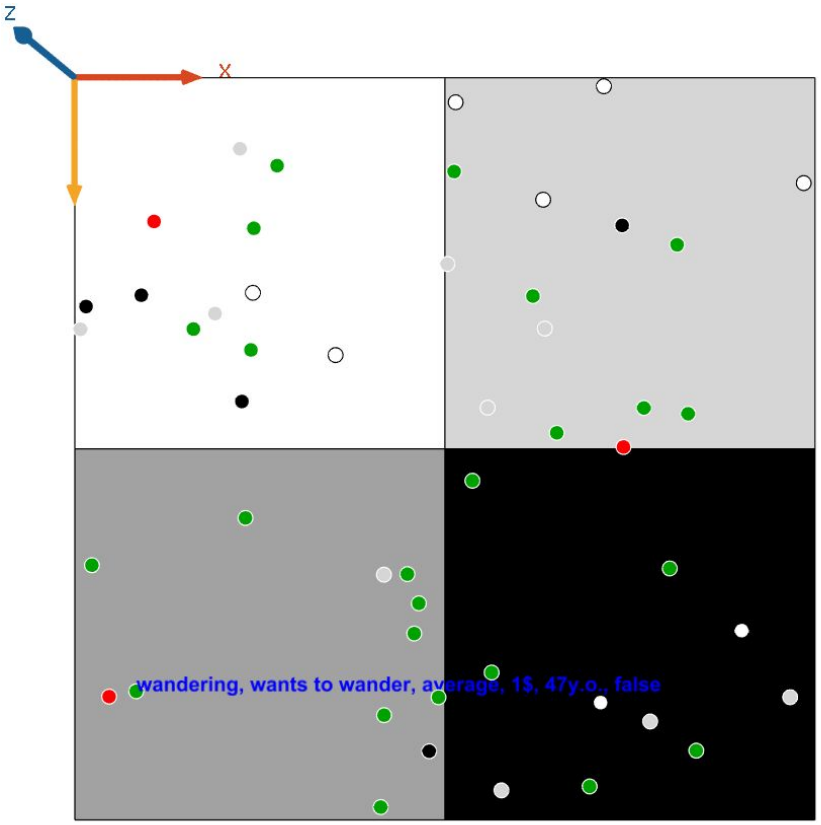

Image 3. The festival grid with agents spawned in.

Agent Traits

As stated previously, agents can interact with one another and depending on their personalities the agents might get along and do an activity together if their personalities match. At the same time, they might dismiss each other if the personalities do not match. The way this works is an agent interacts with nearby agents with the FIPA protocol by sending the other agent his traits. Three traits used in this simulation are *talkative* (boolean), *age* (integer from 18 to 50), *wealth* (integer from 0 to 9).

The agent who receives the FIPA message, depending on his own traits and the location it is at, decides the outcome and responds positively or negatively. Each of the traits has a specific bond to a specific location. The correlation is:

- *Age*. Just for the sake of simplicity, the age is the main attribute linked to an agent's wish to party with another agent, based on the assumption that people like to party within a certain age group. Therefore, if the agents are too apart in their age, they will not interact in the party area.
- *Wealth*. Wealth is the main attribute used for determining whether the agents are suitable for gambling in the gambling area. If one of the agent's wealth parameters does not reach a certain minimum (which is based on the agent type), then the interaction will fail.
- *Talkative*. This attribute is used for determining whether the agents would be compatible to "chill" together. Since this parameter is a boolean, both of the agents in the interaction should have it set to the same value, be it true or false. If the attributes

match, then the interaction is successful, otherwise, it is cancelled. However, it must be noted that different agent types might have different presets. For example, an agent of *chill* type will always want to "chill" with anyone, no matter what their talkative attribute is set to, because it is in its preferred area and is more accepting of other agents.

Therefore, agents will have completely different encounters with each other, all depending on their type, their age, their wealth and their talkative parameter. For example, an old rich gambler might never interact with a young poor party person because their characters are just too different. On top of this, it must be also taken into account that these two agents will have different preferences in what areas they like to visit, thus decreasing the chance of them interacting further.

Agent Reflexes

Visitor agent has the following behaviour reflexes:

- *go_wander*
  When an agents wish is to wander and it has no target point, the agent goes to wander until his wish is satisfied. This reflex does not make use of the in-built wander function, but rather implements a pattern that would resemble random movement that would allow to get to a completely new location quickly. This is done because wander function does not allow for a quick relocation and visually just represents some minor directional movement.
- *eat*
  Since the food level is decreasing over time, once it hits 0, the agent has to go to eat something to replenish this parameter. So, a new target is set - eating location. Once the target point for eating has been reached, the parameter is refreshed with a random value within a certain range.
- *get_a_wish*
  This reflex is used to set a wish of an agent when it is equal to nil. A wish is randomly selected according to the type of the agent and the probabilities it has set. Note that eating is not a part of the wish list.
- *party, chill, gamble*
  Same as the eat reflex, these are used to guide the agent to the corresponding location of what he wants to do and stay there until the wish is satisfied. However, one main difference is that eat reflex is instant once the target reaches the food destination. In the case of party, chill, gamble, these actions take some time to finish, usually 30 frames.
- *ask_visitor*
  If there are other visitors nearby, the agent will ask them if they want to do something together.
- *answer_visitor*
  When an agent receives a request, it decides what to do based on his traits. However, apart from the traits, the location of interaction matters as well. For example, if two agents are communicating in the gambling area, their interaction will depend on the wealth of both of the agents. On the other side, if the agents are

communicating in the chill area, their interaction will depend on whether the characters are talkative or not.

Chart

To complement the solution, a chart is added to display some interesting information besides what we can already see on the screen.

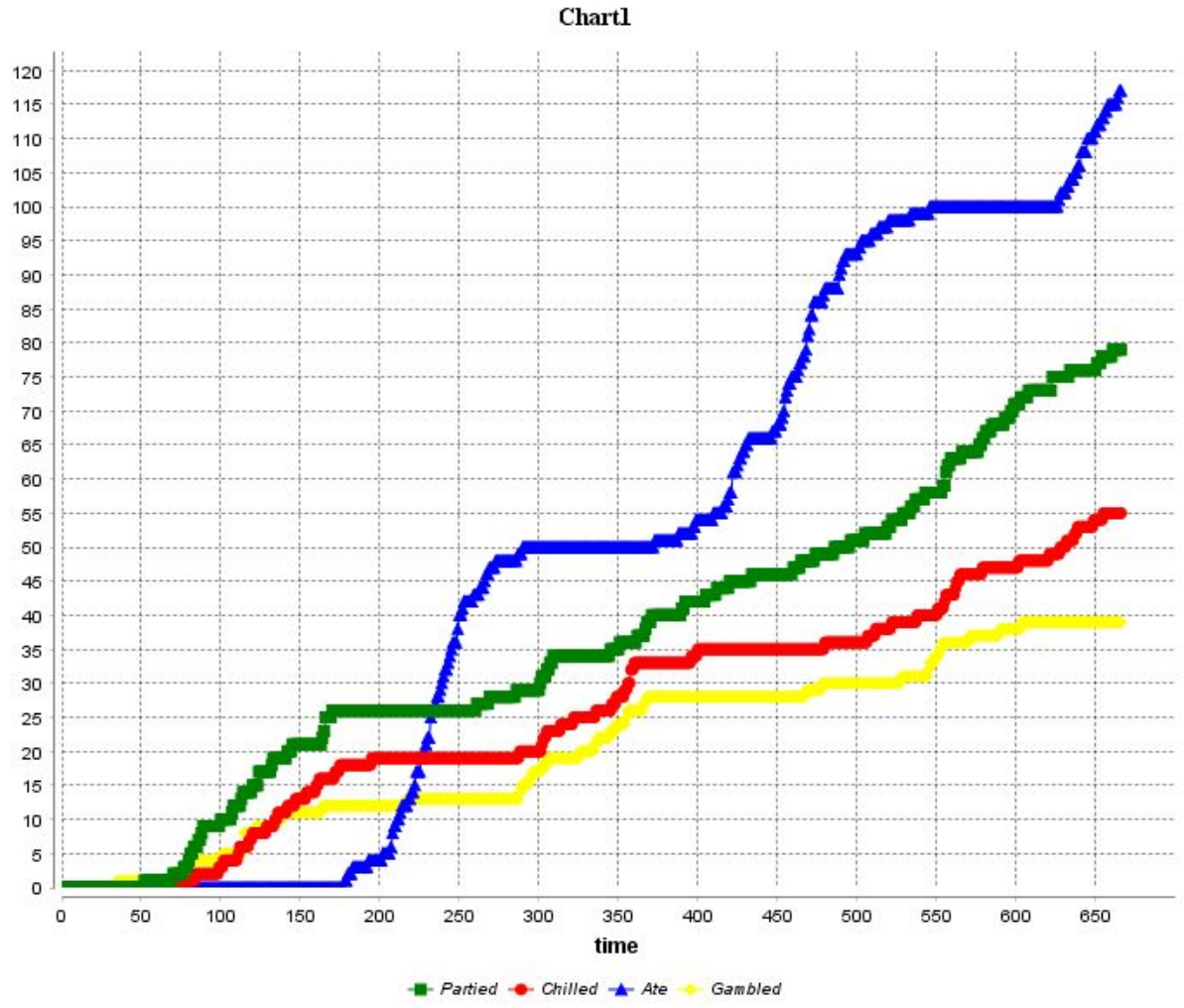

Image 4. Chart showing total number of different actions.

This chart displays the number of times a wish has been granted. As we can see in image 4, people have overall eaten the most times, then partied, chilled and lastly gambled. Although, this chart could be modified to see other kinds of information, such as the average number of activities people are doing and so on.

As it can be seen from the chart, the agents flock together to eat because in the beginning of the simulation they have roughly the same level of food. However, as time passes, some of the agents will eat more and some will eat less but more often, thus lineairizing the curve after a certain amount of time cycles has passed.

Experiments

The experiments we had were based on the number of people with different traits in each experiment. So, for example we tried running the solution, with more gamblers than in other experiments, then we tried running it with an equal amount of people with different traits. And the results displayed on the chart were as we would have expected - more the people of a single trait - more times it happens, since each type of personality has its own highest probability. Although, what didn't changed based on the numbers of people, was the amount of times people went to eat. It is this way, because every single person, not based on its traits have the same hunger attribute and it reduces at the same speed.

Experimenting was a great part of developing this simulation. Fine-tuning some of the behaviours of the agents was a lengthy process. Initially, when the very first iteration of the simulation was created, agents did not have any subtypes and could only interact with the static environment and not other agents. Then, there was a decision to categorize different agent types and their attributes based on the environment, with average and weirdo being two exceptions. Thus, there is a clear correlation between what agent type prefers to do, and what attribute allows/forbids to interact in a specific manner. The table below shows this correlation clearly:

Party Subsystem	Chill Subsystem	Gambling Subsystem
Party background	Chill background	Gambling background
Party agent type	Chill agent type	Gambler agent type
Age (int)	Talkative (bool)	Wealth (int)

Table 1. Correlation between specific environments, agent types and attributes

Results

The implementation of this solution was successful and experiment results are as we would have imagined. The outcome is that it is possible to imagine this software being used to simulate real world simulations.

Challenge

For the challenge part we decided to implement the BDI architecture. To be more specific, we changed the current implementation with belief, desire and intention architecture. What this means is that the agent now has things he believes in, intentions to solve things and desires that makes him to them. This was a whole new architecture and taught us another way to look at a problem and how the beliefs can combine to get different outcomes.
Even though the architecture was new, but the program is still working in the same way and from the outside we do not see any changes besides some minor ones.

Creative implementation

Our creative solution is combined together with the base task. The assignment did not specify anything about the agent's behaviour, so a simple wandering agent would be

sufficient for setting up a skeleton for the FIPA communication. However, in our implementation each of the agents has its own wish system, where the agent will randomly get a wish based on its preferences and go and fulfill that wish. As stated previously, the wishes can be active or passive. Active wishes are partying, chilling, or gambling. A passive wish is a wish to eat, when a food counter goes down to 0. Also, a custom wandering function was written to allow agents to move much faster and in a more human movement resembling manner.

Qualitative/Quantitative questions	*Answer*
Time spent on finding and developing the creative part	5 hours
In what area is your idea mostly related to...	Agent behaviour, agent movement, agent interaction
On the scale of 1-5, how much did the extra feature add to the assignment?	4
On the scale of 1-5, how much did you learn from implementing your feature?	5

Discussion

It can be argued that our simulation contains too many configurable parameters, such as individual percentages for each agent type. Some of the choices that we made when developing the behaviours might not have a concrete explanation behind them. For example, we have assumed that if an agent does not meet a certain wealth attribute level, it will not be eligible to gamble together with another agent. These sort of decisions are an artistic touch to the development of the simulation. These decisions may not carry any concrete benefits, but they make the simulation more visually pleasing to follow. Otherwise, the agent's behaviours would be too predictable and there would be no incentive to run the simulation for prolonged periods of time.

The main learning outcome of this assignment was the understanding of how multi agent systems can be used in real life to simulate situations that provide feedback which is useful and could be used professionally.

Conclusion

This type of a solution could be used in real life, for example, to see the number of times an action has been done (using the chart), so that the supplier of a festival could prepare the food stands. This might be just one of many use cases that a single application like this could solve. However, as stated previously, quite a few variables in this specific simulation have been predefined. Therefore, a careful study and approximation of these values is required if this is to be used purposefully.