ID2209 – Distributed Artificial Intelligence and Intelligent Agents

Final Project – GAMA and Agents

Group 14
Ilias Merentitis
Chrysoula Dikonimaki

30.12.2021

How to run

Run Gama.exe and import the .gaml files. Press the green button to execute the experiment called Project (each .gaml file has a separate experiment).

Species

Guest:

The Guests are the actual people that this project is trying to simulate. They have a view distance, within which all places are considered visible, and a mood for a random type of place (as this is defined in the placeTypes list). A guest's stay in a place is influenced by two variables: *happiness* -which is in turned influenced by positive or negative interactions with other agents- and *time* -which is meant to simulate random boredom which will make a guest want to try out a new place-.

With `use_emotions_architecture` we take advantage of built-in properties, which we use to shape a guest's traits at random:

```
float openness <- rnd (0.0, 1.0);
float conscientiousness <- rnd (0.0, 1.0);
float extroversion <- rnd (0.0, 1.0);
float agreeableness <- rnd (0.0, 1.0);
float neurotism <- rnd (0.0, 1.0);
```

The combinations of those characteristics are used to classify the agents (i.e., a politically incorrect "labelling"). Different types of guests have different and non-symmetrical opinions of others. For example, a boring person may like a person who is into deep conversations, but they other way around may not be true.

Place:

The Place species was created to represent all possible places where the guests can go. For the purposes of this assignment, we created "bars" and "concerts". The types of species are defined in a global list, which can be very easily extended to include more:

```
list<string> placeTypes <- ["bar", "concert"];
```

Base concept:

The base concept was to create an environment where guests would interact in different way based on their preferences, character and the place the interactions happen.

The agents of type "guest" can be split into 5 broad types:

1. noisyPartyAnimal
2. quietPartyAnimal
3. deepTopicsChatter
4. boring
5. cringy

These types of guests are affected by their 5 personal traits:

1. openness
2. conscientiousness
3. extroversion
4. agreeableness
5. neurotism

For example:

```
if (extroversion > 0.5) {
	if (neurotism > 0.5) {
		guestType <- "noisyPartyAnimal";
	}
	else {
		guestType <- "quietPartyAnimal";
	}
}
```

In the beginning each guest starts out with the `find_place` desire. That only means that they will randomly wander around. In the meanwhile, they perceive their environment, and when a place of the correct (moodFor) type comes into view, they will go in. (they will not visit a place a 2nd time if they have visited it a while back. The condition for this place to become available again is when they have visited at least 1/3 of the total places again).

When they enter a place, they will scope all the other people inside. Using the `socialize liking` attribute, each guest will create a social network of known people. We consider any positive value of liking as good, and any negative value as bad. Liking > 0.5 is excellent. Depending on the liking value, their character, as well as the other agent's character, agents will interact with each other. Each interaction is handled through FIPA protocol. We assume that a conversation is starting (an agent may propose a dance, offer a drink, ask about something, or even insult another agent or ask them to leave. We use the `accept_proposal` mechanic to demonstrate that the agents are in agreement, and thus happiness is increasing, and `reject_proposal` to show disagreement, and happiness is decreasing. For that, the actual place this interaction is taking place is also considered. For example, cringy agents that often insult others will get a lot of rejects and their happiness will quickly drop, forcing them to look for another place. (Sooner or later, those will find themselves hard to fit in in any place, and will roam aimlessly around the premises like outcasts (which they sort of are – they are cringy after all and life is not fair). To make matters worse, if a guest's happiness drops below a certain level, they see no further point in continuing to live, and promptly end their life.

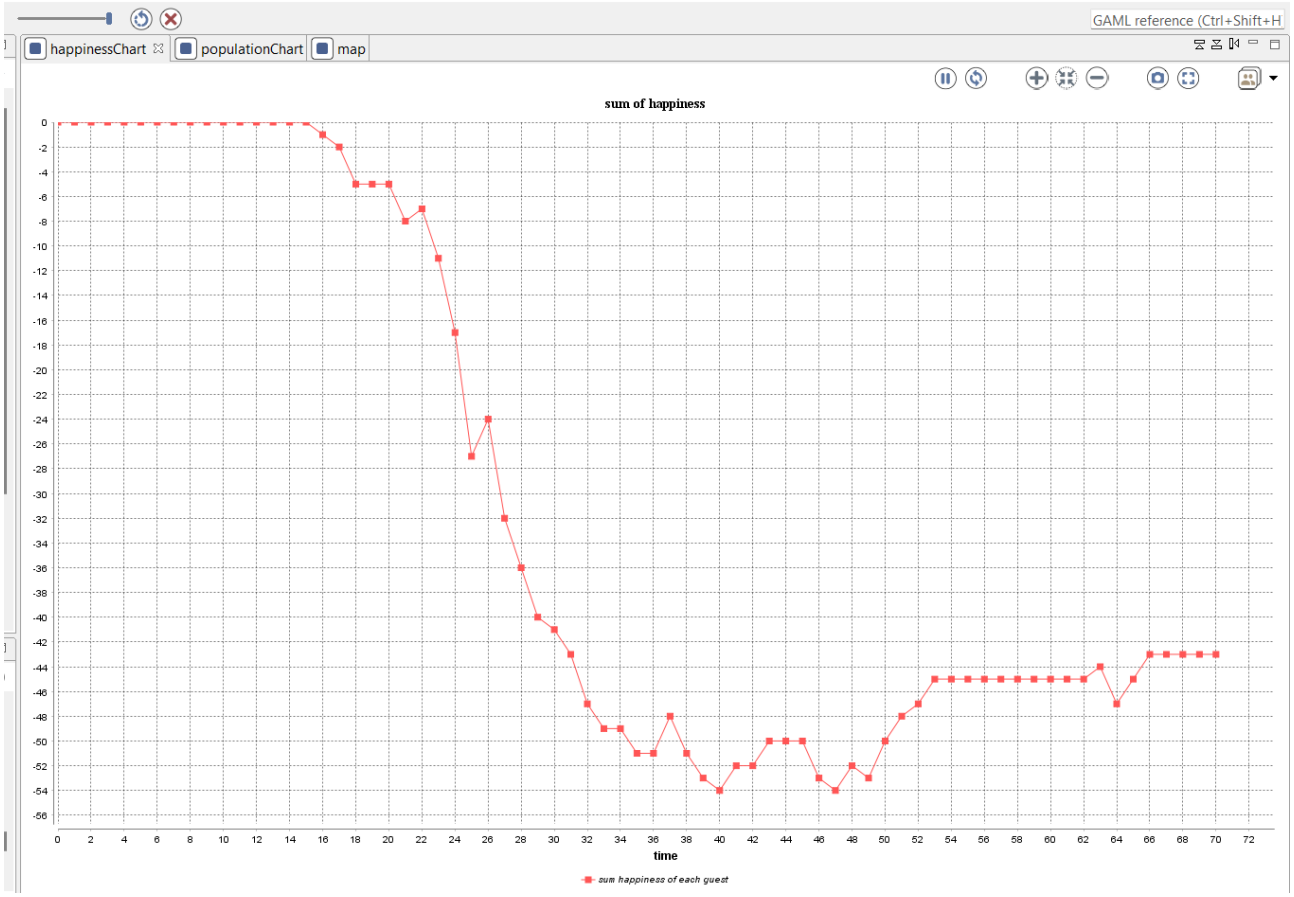
Challenge 1: BDI

In this challenge, we were supposed to implement the project using the BDI architecture. For the shake of convenience, we have implemented it with the BDI architecture as a basis from the very beginning, so completion of this task was included in the basic version above.

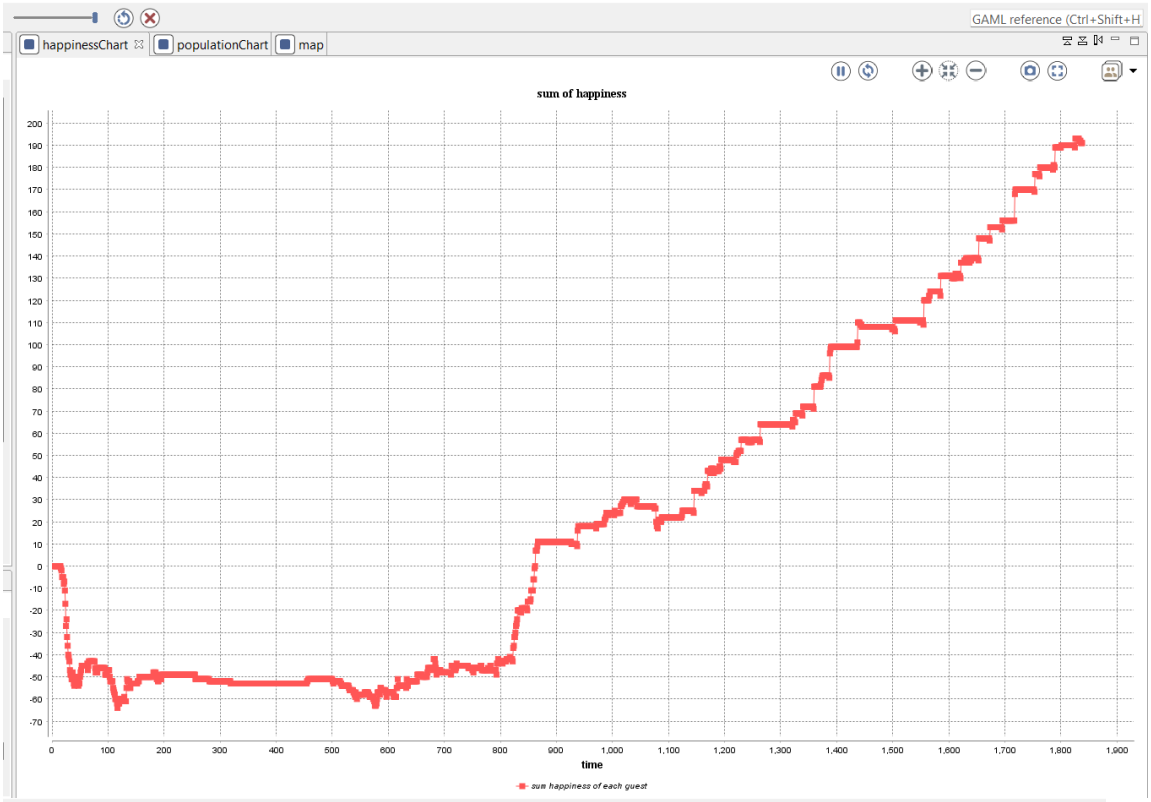
Global and interesting value to monitor on a chart: Happiness

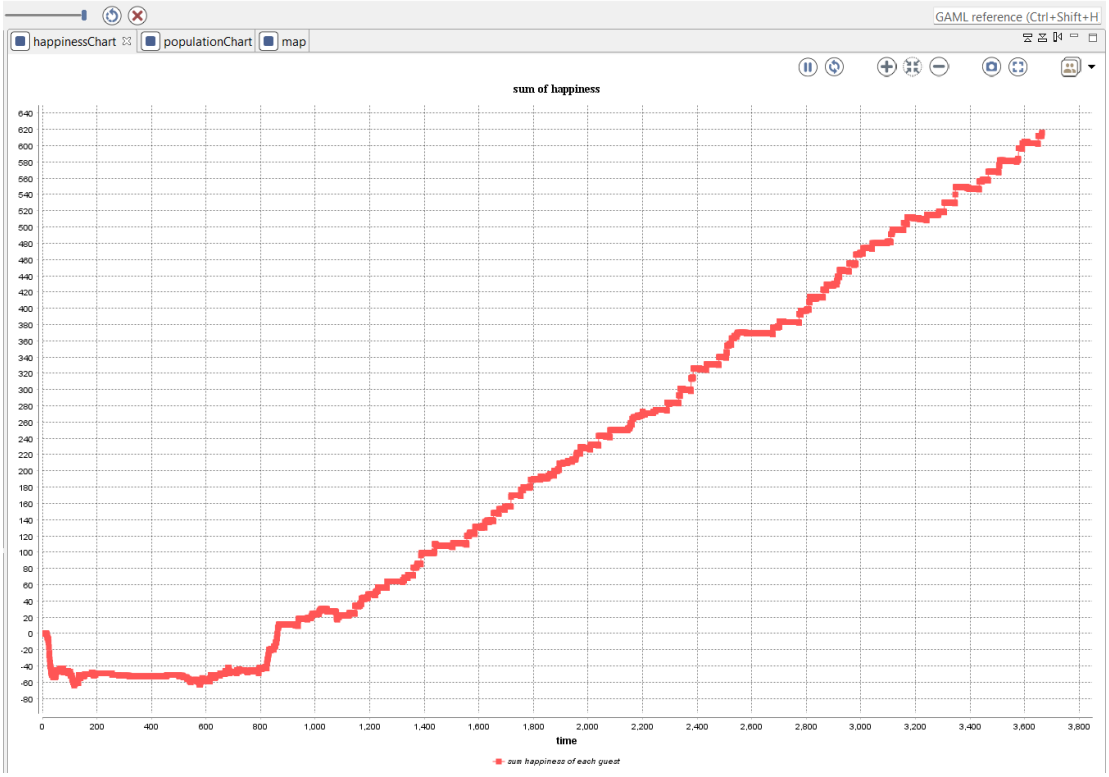
The most fundamental value of this project, where all other calculation converge, is of course *happiness.* In the beginning all guests begin with a default happiness value of 0.

Something interesting to observe is that when the first guests interact with their peers in a place, the happiness value as a whole will drop sharply. That is to be expected as each guest type has strong sympathies to only a minority (maybe 1/3) of the rest of guest types, including themselves.

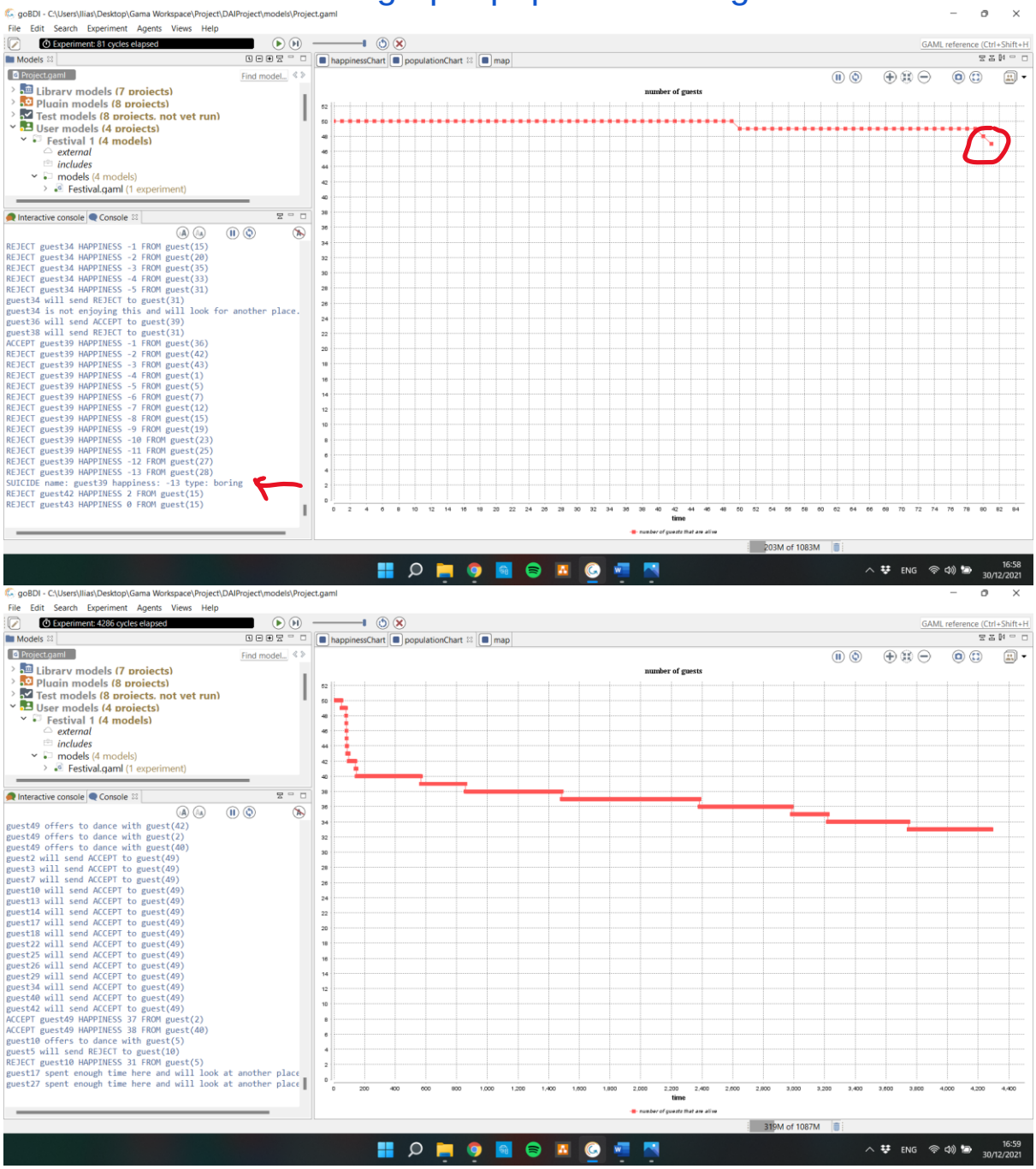


This will prompt guests to leave their current place in search of new ones. In time, a certain balance will prevail, aided by the fact that annoying people are very quickly being forced to leave the place they ruin, and sooner or later commit suicide and get out of the picture anyway. Thus, the overall happiness will increase as the experiment progresses.

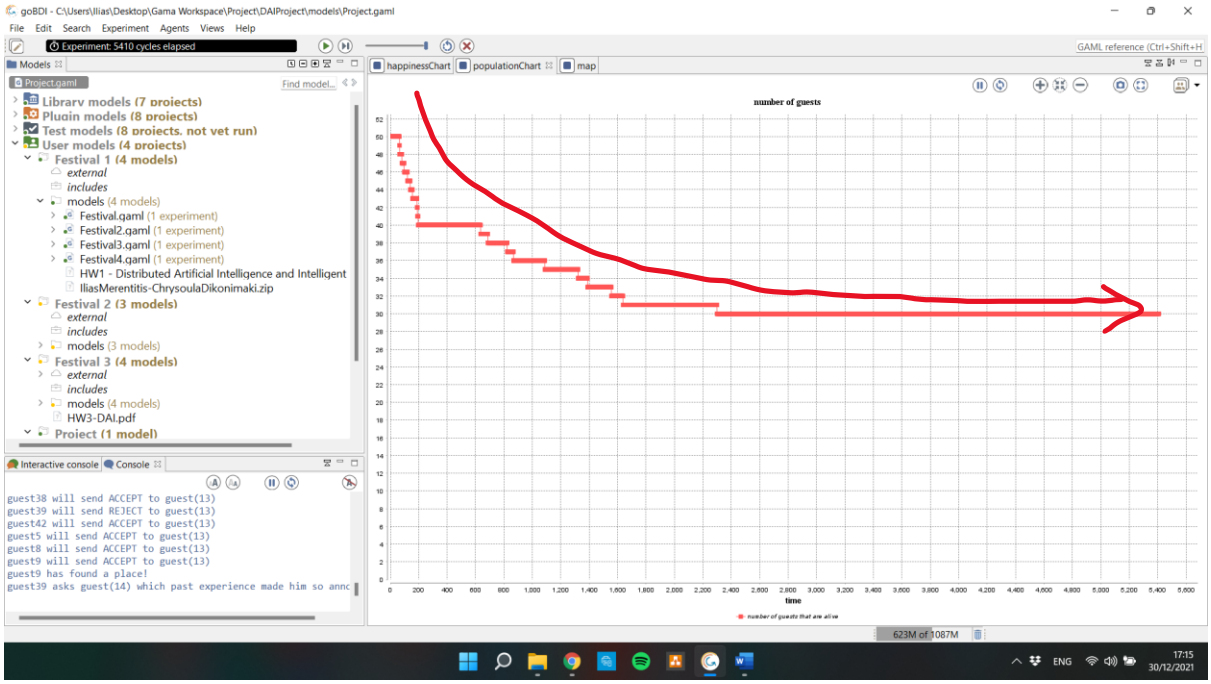

GAML reference (Ctrl+Shift+H
happinessChart
populationChart
map
sum of happiness
time
sum happiness of each guest


GAML reference (Ctrl+Shift+H
happinessChart
populationChart
map
sum of happiness
time
sum happiness of each guest

Useful and informative graph: population of agents



As we can see, the agents that cause most of the trouble will quickly get a very negative happiness and commit suicide. Therefore, the rate of suicides is lowered and flattened as the experiment progresses.



Number of guests = e^(-x), x = time