ID2209 – Distributed Artificial Intelligence and Intelligent Agents

Lab Assignment 2 Report

Group 28

Enrique Perez Soler — Daniel Conde Ortiz

December 15, 2019

1 Introduction

Improving upon the Festival simulation developed in *Lab 1*, we now proceed to design an auction simulation based on the *Dutch auction* and we introduce a new agent: Auctioneers. We will provide participants to the auction whom will propose prices to buy a certain product provided by the auctioneer. The first participant that offers a reasonable price for the figure the auctioneer announces will obtain the product.

2 How to run

Run GAMA 1.8 and import folder Lab2 as a new project. Press *Lab2* to run the simulation. The simulation runs quite quickly so we recommend the speed is reduced in the upper-left corner of the simulation near the "play" button in order to follow better the steps of the auction and the exchange of the protocols. At the beginning of the file you can change how many agents of each species are there. The default values are: 1 auctioneer and 3 festival guests.

3 Species

3.1 Auctioneer

It's represented as a blue box but it's not important or relevant for the simulation. In this version of the code there's only one of it and it first initializes variables to establish the initial state of the simulation like a Boolean *sold*, selects a random initial *price*, etc. Then, it proceeds to send an inform message to all festival guests that an auction is starting and also informs of the initial price which the product is been sold with through the *send request* reflex.

After which, it checks through reflex *read proposes* the prices proposed by the festival guests and rejects or accepts depending on the value. Should more than one festival guest be suitable to buy the product, the simulation applies the "first come first served" policy as shown in the Dutch auction.

When looping to decrease the initial amount provided by the auctioneer, the state of the *sold* variable will be tested. Should the product had been sold the auctioneer will proceed to *start again* and the loop will be broken. Nevertheless, while the product has not been sold it would loop again decreasing the price amount until a previously stablished *minimum.*

3.2 FestivalGuest

It's represented as a red sphere but it won't be used for this simulation. A festival guest would be provided a random figure to propose as price with which it'll buy the product. Then, through the *reply* reflex it will send a message to the auctioneer agent and formalize a proposal. Its price then would be accepted or rejected by the auctioneer depending on the price it establishes at that particular moment

4 Implementation

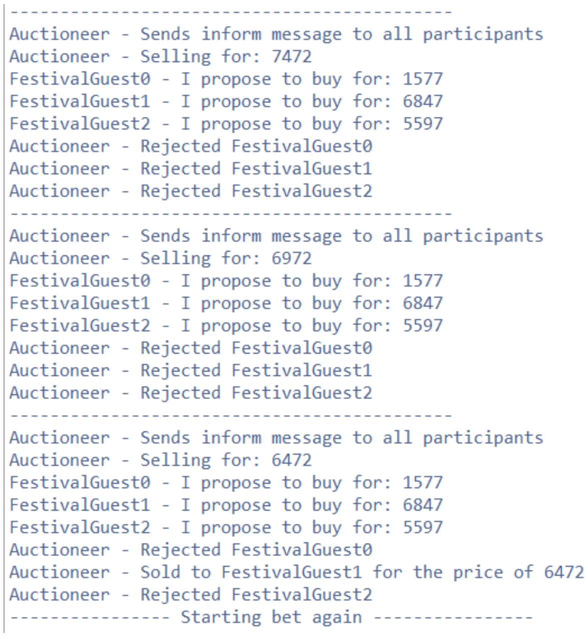
We started with last lab's code as template. We reduced the number of species to 2 and the number of agents of each species. By looking at GAMA's documentation and the lab's presentation, we tried to understand how the FIPA protocol works.

We implemented a pair of reflexes to communicate from the Auctioneer to the FestivalGuest and get information from one side to another. We looked for how to implement CFP and it's different messages. We tried different methods to answer the Auctioneer, but finally settled for *propose*.

Then, in the Auctioneer class, we implemented the accepting or rejecting of these proposals, depending on the price. Finally, we made the whole process loop each 10 cycles, decreasing the price or starting a new auction.

5 Results and Conclusion

This lab's implementation was very slow and at times confusing as we have never dealt with CFP messaging nor were we that experienced with GAMA's language to fully understand the sample code given in the documentation. When running the simulation we can't appreciate any interactions in the graphical window but rather find the sequential "conversation" taking place in the console. We have organised each *write* result to manage a dialogue portraying the different steps taken. With it we can process every decision made and understand what the code is doing at all times (figure 1)



```
--------------------------------------------
Auctioneer - Sends inform message to all participants
Auctioneer - Selling for: 7472
FestivalGuest0 - I propose to buy for: 1577
FestivalGuest1 - I propose to buy for: 6847
FestivalGuest2 - I propose to buy for: 5597
Auctioneer - Rejected FestivalGuest0
Auctioneer - Rejected FestivalGuest1
Auctioneer - Rejected FestivalGuest2
--------------------------------------------
Auctioneer - Sends inform message to all participants
Auctioneer - Selling for: 6972
FestivalGuest0 - I propose to buy for: 1577
FestivalGuest1 - I propose to buy for: 6847
FestivalGuest2 - I propose to buy for: 5597
Auctioneer - Rejected FestivalGuest0
Auctioneer - Rejected FestivalGuest1
Auctioneer - Rejected FestivalGuest2
--------------------------------------------
Auctioneer - Sends inform message to all participants
Auctioneer - Selling for: 6472
FestivalGuest0 - I propose to buy for: 1577
FestivalGuest1 - I propose to buy for: 6847
FestivalGuest2 - I propose to buy for: 5597
Auctioneer - Rejected FestivalGuest0
Auctioneer - Sold to FestivalGuest1 for the price of 6472
Auctioneer - Rejected FestivalGuest2
--------------- Starting bet again ---------------
```

Figure 1: Logs where we can see the conversation between Auctioneer and Guests