

Market in NetLogo

*Art galleries, paintings, and collectors*

Jordi Bonet Valiente

Clara Rivadulla Duró

Index

**Introduction** 1

**Analysis** 1

**Tests and results** 7

**Conclusions** 7

## INTRODUCTION

In this final delivery, we’ve decided to model a MAS Market which consists mainly of the **transaction** between **art galleries** and **collectors**, where the specific products that are sold and bought are **paintings**. Communication is done through message passing (which we saw in the last practice).

Although its process has involved the definition of interaction protocols among agents and the design of the interface in NetLogo, the following document explains in detail how the project was coded, and which were the experiments we did to test it.

## CODE ANALYSIS

We’ve declared the following global and agents’ (turtles) variables:

Texto

Descripción generada automáticamente

Like we can see, we’ve declared as global variables which ones that needs to be access to all agents (all the paintings, his authors, his buyers). The attribute *paintings-sold-to-process* is a list for update the market’s interface when a painting be sold.

By the other way, we’ve the turtles’ attributes for galleries, collectors and paintings (the paintings aren’t agents, only is a turtle to see them in the market interface).

In the setup function, we setup the paintings, the galleries and the collectors.

Texto

Descripción generada automáticamente

In the paintings’ setup, we set all paintings attributes and load them in the market’s interface. We set them considering that always have the same two art galleries.

Imagen que contiene Texto

Descripción generada automáticamente

In the following picture, we’ve the galleries’ setup. Here we create two art galleries: the art gallery of Albacete and the art gallery of Madrid. We initialize their attributes and put them in market’s screen.

Interfaz de usuario gráfica, Texto, Aplicación, Teams

Descripción generada automáticamente

In the collectors’ setup, we create N collectors and initialize their attributes. As we can see, the collector’s preferences, which are his money and his preference author, are random.

Texto

Descripción generada automáticamente con confianza media

The ‘go’ function is like the last delivery. First, we swap messages and process them. Then send advertisements and see if a painting was sold to update it (process-paintings-sold).

Texto

Descripción generada automáticamente

In ‘send-ads’, we send an advertisement to a collector, checking if the collector hasn’t already received an ad of this gallery. When the gallery has sent each collector (when the number of ad receivers is equal to the number of collectors) the gallery doesn’t send more advertisements.

The next functions are very similar to the functions of message passing in the last delivery. The difference is that we add a list of values to pass with the messages that are necessary to an operation when we process it. Only the agents, a gallery or a collector can swap messages because the paintings aren’t real agents. The different kinds of messages are *AD, INFORM, REQUEST, RESPONSE, BUY, SELL* and *SOLD,* with a function similar like the illocutionary particles that we saw in theory.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

When a collector, the receiver, process an advertisement message, first chech if the attributes are compatible (the same), and, if is the case, check if the author that is for his interest (index 1 of the preferences’ list) is the same that the advertisement. If is true, sends a message confirming to the gallery that is interested. If is false, sends one saying that it’s not interesting.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

When the gallery, the receiver, receives the feedback of the collector, if the collector is interested, tries to sell him the painting. Else, asks for collector’s author preference. If the message of inform is that doesn’t have paintings of this artist, doesn’t do nothing and the communication finish.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

If the gallery sends a *REQUEST* message, the collector (the receiver) sends his preference author to them.

When the gallery processes the *RESPONSE* message, if has the author tries to sell a painting to the collector. Else, only informs that doesn’ have paintings by this artist.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

To process a *SELL* message, the collector (receiver) compares his preference price with the price of the painting and, if is equal or less than his preference price, sends a *BUY* message. Else, the collector negotiates the price: make the medium of the price and sends a *BUY* petition with this new price. In the interface, is here when the collector goes to the gallery.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

The gallery processes the *BUY* message sending a *SOLD* message if hasn’t aleready sold the painting (if is the case, sends an *INFORM* message).

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

The process of a *SOLD* message consists of updating the gallery's and the collector's money, updating the collector's paintings and setting this painting as sold. In the interface, this is when the collector returns to his initial position.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Each iteration calls this function which has the responsibility to change the colour of painting if it is sold.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

## TESTS AND RESULTS

The influence of the value of the number of collectors

Changing the number of

## CONCLUSIONS

In conclusion, a better performance