MAS[1] Simulation

Practical Lab for the “Concept Objet” (M2) course

Introduction

You develop a simulation of a multi-agent system (MAS) with 4 different groups of living beings in C++, JAVA or RUST. You can take inspiration from the Heroic Fantasy universe (e.g., Orcs, Goblins, Elves and Humans) or propose another universe (e.g., Harry Potter, Futurama, Star Wars, The Rings of Power, House of the Dragon, Final Fantasy, Mario etc.). You should build on the inheritance between classes and messaging developed during the previous practical works. Thus, the simulation will consist of the exchange of messages between the different instances of individuals in each population, this time taking charge of their location on a two-dimensional map.

From the start of the simulation, the 4 groups are in competition although there are alliances. Victory is won by the wiser group, that is, the group that has the most knowledge, that is, the group that collects the most different messages. There is a special individual for each group who is called “Master”. Here, the more a teacher has listened to different messages, the more likely the group is to be the wiser and therefore to be the winner. The latter is not mobile and lets his companions collect messages and bring them back to him. The exchange of messages that takes place between two individuals is done according to their group and a possible alliance. Such an exchange is possible due to their proximity.

You will thus develop a multi-agent system where all the fundamental object concepts appear. You must document your project with UML diagrams and find the usefulness of a design pattern. Pseudo-randomness will occur at the level of movements on the card and also at the level of the reception of messages. The following elements can be adapted, you must nevertheless respect the points raised in the section “Expectations” at the end of this document.

The Map class

Let us call the environment in which the 4 populations of the simulation evolve “Map”. The properties of this map are as follows:

- Rectangular in shape, each corner has a region called SafeZone. Each population has one and only one SafeZone. In proportion, it is preferable that the sum of the 4 SafeZones is less than half of the surface of the map. Each Master is static in its SafeZone, it could for example be located on a corner of the map.
- The map is discretized into n * m boxes. Each box is identifiable by a horizontal index and a vertical index. A method of the Map class will return the tiles available around a tile given as a parameter: 3 (for a corner of the map), 5 (for a side of the map) or 8 tiles (for the other cases). Another method will return the direction to take (see subsection on the movement of individuals) to reach your own SafeZone,

[1] https://en.wikipedia.org/wiki/Multi-agent_system

samuel.deleplanque@junia.com

- Obstacles (e.g. trees, furniture...anything you will imagine) can be generated randomly at the start of the simulation.
- Each box can contain a maximum of one individual or obstacle; when there is a "confrontation" between 2 individuals, it can only be done on two tiles that touch each other (including diagonals).

Simulation

The simulation is discrete and at constant time step. Triggering each new step in the simulation will mean that the duration of a time step has elapsed. At each stage, all mobile individuals have a method running in order to make them evolve on the Map. The order of manipulation of individuals is calculated randomly at each step. The simulation ends when one of the masters has all the messages in the game. In case the implemented system involves a too difficult calibration to systematically reach this end, the simulation can end after a maximum number of steps and imagine a calculation method based on the final characteristics of the individuals to determine the winners.

Individuals

Orcs, Goblins, Elves and Humans or any other group of 4 populations therefore evolve on the Map. All are living beings, but alliances must be made. For example: Orcs and Goblins generalize as Bad beings while Elves and Humans generalize as Good beings. Consider a class structure reflecting the fundamental object principles by making 2 alliances like in this example. All of these individuals have these attributes in common:

- An array of `strings`
    - for saving first messages as well as those obtained from other groups,
- The last direction taken (North, North-East, East, South-East etc.), in the previous step.
- EP (Energy Points): these points are spent outside the individual's SafeZone and are recovered there. The difficulty is to properly calibrate the starting EPs and their consumption according to the number of tiles traveled in order to obtain a balanced simulation.

The very first design pattern that we will see in class, the singleton, will have to make sense there, with the unique instantiation of MasterTypeGroup' object (eg, for each MasterElf, MasterOrc classes etc.) that we will find as a specialization of a group. If the singleton has not been seen in class yet, it's up to you to document it in the meantime, but you can of course ask the teacher.

Rules for the movement of individuals

- Like the polymorphic `move()` method seen in class and in previous practical work, each step consists of calling this method for all of the individuals. You have to find a way to generate a different order of calling the individuals `move()` method at each step.
- At each step, and for each individual, the `move()` methods consider several possibilities that you have to define. You can for example use, as in chess, the movement of the Bishop, the Rook, the Queen and the King. It is up to you to define these modes of movement, it can be simpler (e.g., direction and number of boxes to

be traversed drawn randomly). The movement can be stopped by an obstacle or an individual standing in the way. Moreover, it is only when the path of a first individual is stopped by a second that there is "Meeting" and the possibility of exchanging messages.

- Each time you move out of the SafeZone, EPs are lost, depending on the number of tiles covered (you must specify these variations). Traveling according to the EP can follow these rules:
    - If EP ≥ 20%, the movements are completely random (number of boxes and/or types of movement),
    - If EP < 20%, the movements are oriented towards the direction of their SafeZone. Once in SafeZone, individuals see their EP increase. It's up to you to calibrate so that people don't go back and forth too much.
    - If EP == 0, the individual will no longer be able to move and becomes an obstacle and therefore loses all his messages.
    - If an individual is blocked by an obstacle, he will have to stop in front of the tile blocking him, but he will lose as many EP as he has left to go. The individual will have saved the last direction taken so that the next movement is not directly blocking.

Encounters between individuals

An individual can encounter another individual on his way[2]. With the 4 types of population, there are several cases:

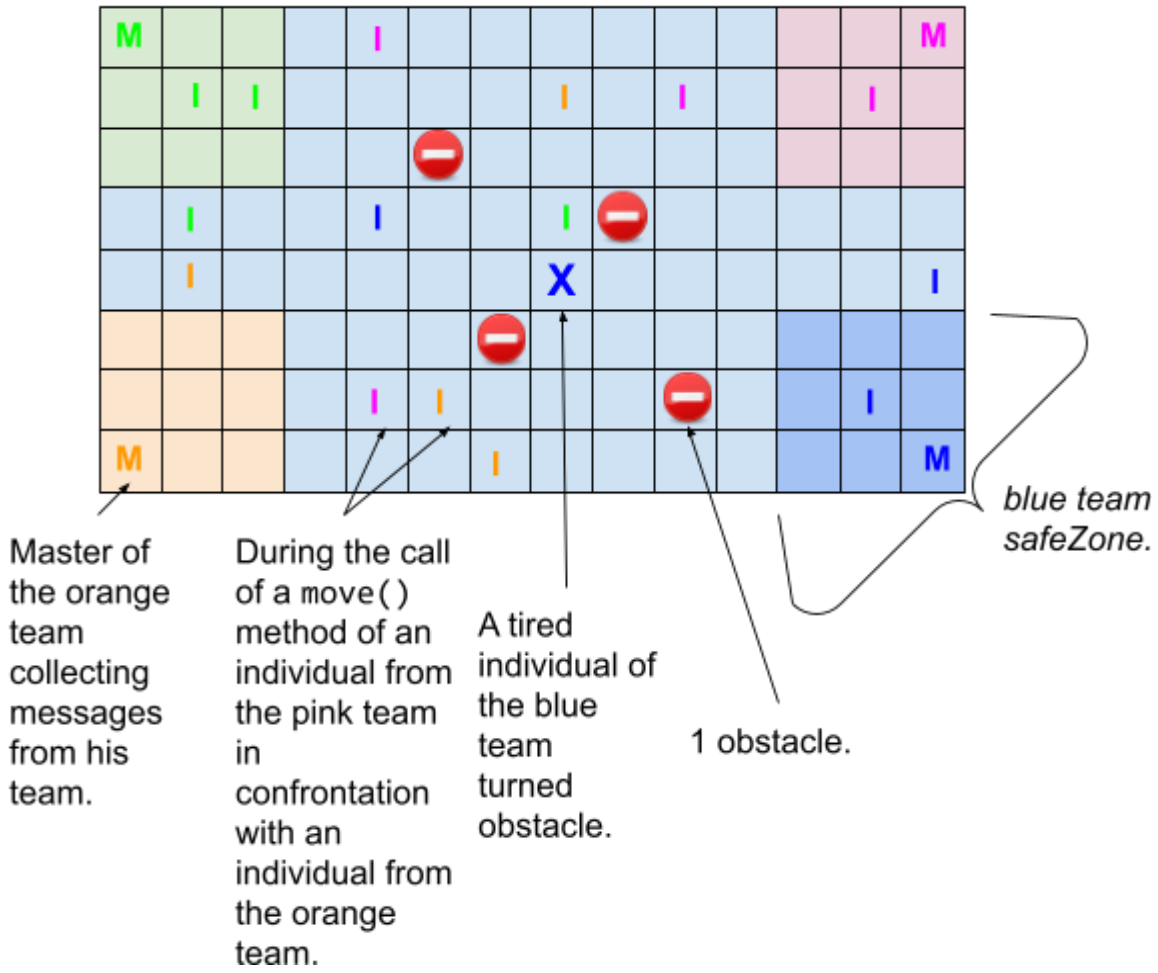
- Same population (class) encounters:
    - Individuals make the union of all their knowledge[3].
- Different population encounters:
    - Between two individuals who share the same alliance:
        - a random number of messages can be exchanged (up to you to calibrate).
    - Between two individuals who do not share the same alliance:
        - There is confrontation. The winner gets a random number of messages from the loser that the winner does not yet have. The loser loses these messages which are thus transferred. You could implement something simple like “Stone-Leaf-Scissors” or go further: for example, you could develop a turn-based system from role-playing games. The generation of pseudo-random numbers will still be of great help to you here. You are free to add the attributes you want to your classes, but this on condition that they are well placed in the class hierarchy (inheritance). Strictly no confrontation can take place in a SafeZone since only the individuals of the group of the master who is there can enter it. For others, the outline of SafeZones is characterized by obstacles.

[2] For this example, we consider that if the movement ends just in front of a tile with an individual, there is no “share”. For there to be an Encounter, the box of the individual encountered must be part of the planned path.

[3] Be careful not to make any duplicates and check that the messages do not already exist in the table.

- It is up to you to define the transfer of messages from individuals to the master. For example, you can keep it simple by considering that the master collects them as soon as one of his individuals passes one tile away from him. You are responsible for defining the different behaviors linked to SafeZones.

The following graphic is a representation of a map with the elements described above. Here, 4 teams of 4 different colors compete against each other.



Expectations

- **A discrete simulation with constant time step is expected.** It must set up a multi-agent system (MAS). It is considered here that, during a time step, each individual (except master and those who have become obstacles) tries to move. The order of manipulation of individuals is calculated randomly at each step.
- The **3 fundamental object concepts** must have a central place here. They will also be highlighted in the report. In particular, the methods employed should be **polymorphic** as soon as it is appropriate.
- Use **static attributes** to be able to display the number of instances of each of them.
- The **Singleton design pattern** must find its place in your simulation by reminding you of its usefulness and knowing that it will intervene in 4 different classes.
- The programming language used must be either **Java**, **C++,** or **RUST**. A graphic representation is requested. It can be basic like a console display of the Map with the

different protagonists, all with ASCII characters. Display the maximum amount of information per step of simulation, you can resume the method seen in progress to slow down the execution time so that you can understand what is happening at each step. A representation of message transfers will be appreciated. For lack of space, you can simply use letters for these.

- Set up a specific class generating **pseudo-random numbers** for all the needs in the code.
- For the whole group, a **report** in .pdf format, of exactly **6 pages** (“Cover” included), must contain at least one complete **UML class diagram** (with all the attributes and all the methods and reporting their visibility), a **state-machine diagram** (e.g., about the states taken by individuals in a moment of the simulation) and, finally, an **activity diagram** which will represent the important instructions of the “global” algorithm of the simulation (this includes the main loop handling the time step and the loop that manages the movement of individuals).
- Finally, anonymously, explain the roles that have been assigned in your group to develop this project (architecture, code, etc.). Try to define these roles from the first session.

All these expectations must be explained in the report and/or well commented in the code.

To go further (Bonus)

- Events, e.g., meteorites or a virus, can occur randomly on the map. Such events add obstacles until the end of the simulation, or even reach individuals who then turn into obstacles (as if they no longer had EPs outside of SafeZones).
- You can add more complicated movements based on a Shorter Path algorithm in order to avoid obstacles and not stop in front until the next step. For this, we can use a graph model where each node represents a box and each edge represents the possibility of moving from one box to another. Several graphs can be developed depending on the type of move.
- A second Design Pattern can be implemented in addition to the Singleton if it is justified.
- Once the simulation program has been developed, it can be replicated several times and statistics such as the victory percentage of each group can emerge. Be careful not to reset the pseudo-random number generator or at least to do it with a different seed.
- You consider a 3D world and the altitude is then included in the encounter calculation. An individual in a box (x = 1, y = 1, z = 1) will face another individual if the latter encounters these three coordinates during his movement.

The code as well as the report must be sent to `samuel.deleplanque@junia.com` by each group referee. The deadline will be one week after the exam day for the same course. Your .zip file should be called CO.SimuTitle.LASTNAME.Firstname.zip (with the LAST NAME and First names of the referrer). The whole team will be named at the start of the report.

Good luck!