PROGRAM DEVELOPMENT
Programming Project

2017/18 Course

Computers Science Degree:

Software Engineering

Computers Engineering

(Original idea by Roberto Rodríguez Echeverría)

Deliverable 3 (Wednesday, 20th December)

This document specifies the main actions that must be carried out in order to implement the third stage of the project. The main goal is to extend the characters with the calculation of the routes that they should follow and to distribute the weapons in the squares with a higher frequency of appearances in the paths between square 0 and DailyPlanet. Optionally, the project may include a Graphical User Interface. The functionalities to be implemented are the next:

1. Distributing the weapons in the different squares of the map according to their appearance frequency in the different paths.
2. Generation of the characters' routes.
3. Logging the results into a text file with a particular format (if it was not implemented in the second stage).
4. Complete the unit tests set for the system.
5. External documentation for the project.
6. Graphical User Interface implementation (**optional**)

1.- Distributing the weapons in the squares with higher appearance frequency

The weapons that must be distributed over the map will be generated in the same way than in the previous stage: 60 weapons will be generated.

However, unlike in the second stage, in this stage the weapons will be distributed through the squares according to their appearance frequency in the paths existing between square 0 and DailyPlanet square. In order to calculate these squares, the student must calculate all the existing paths from square 0 to Daily Planet and all the squares should be sorted according (descending) according to their appearance frequency in these paths. As an example, if a square belongs to 3 different paths, its frequency is 3 whilst a square that appears in 1 path has a frequency of 1 and a square that do not belong to any path has a frequency of 0. For two squares with the same frequency, they will be sorted according to their identifier (ascending). Then, the weapons will be distributed over the squares that have a higher frequency (those that appear in a higher number of paths). As it was specified in the second stage, each square will receive 5 different weapons so that the distribution will finish when all the weapons have been distributed. Since the number of weapons to be distributed is 60, only the 12 squares with a higher frequency will usually receive weapons (we will not consider maps with less than 12 squares).

2.- Characters' routes

Both the simulation algorithm and the end of the game must be the same that those specified in the previous stages. However, there will be an important difference regarding the characters' behaviour. They will generate their routes by using a particular algorithm.

SuperHero with Physical Powers

Before starting the simulation, each SHPhysical must calculate its route from the initial square to the square that contains the DailyPlanet. This route will be implemented by following a depth-first search in the graph that represents the maze generated in the map. Therefore, the route will be one of the paths that exists in the Map from the initial square to the DailyPlanet square. Based on this algorithm, the SHPhysical will generate a sequence of directions (N, S, E, W) that represents the path. This sequence of directions will be later followed to reach the Daily Planet (in the simulation). Thus, in this stage, the character's route will not be manually introduced (like in the previous stages).

In the depth-first search that the character must implement, **if a square has two possible neighbours, the algorithm will select the neighbour squares by ascending order of the identifiers** (first the neighbour with the lower identifier, second the next ones, and so on…).

The route calculated by the character should not contain movements that drive to squares without an exit (these squares would not be part of the right path). As an example, in the map shown in Figure 1, the route for a SHPhysical character could be:

S, S, E, E, N, E, N, E, S, E, S, S, O, S, S, E

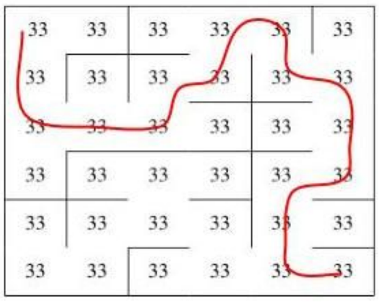

Figure 1: Path for a SHPhysical character

Once the simulation has started, the SHPhysical character will perform the actions that were specified in D2 document (second stage of the project) in each turn.

SuperHero with Extra Sensorial Powers

Before starting the simulation, each SHExtraSensorial must calculate its route from the initial square to the square that contains the DailyPlanet. In order to calculate this route, a SHExtraSensorial must use a well-known algorithm to solve mazes: the **Wall Follower** with right-hand rule. This algorithm is based on the idea that the character is touching a wall in the square with the right hand. Then, the character must just move over the different squares always touching the wall with the right hand. As an example, taking into account that the initial square of the SHExtraSensorial is 0, Figure 2 shows an example of the path that the character would generate (considering that the DailyPlanet is located in the square 35). Initially, it must be assumed that the character is touching the West wall of the square 0 (it would be looking at South direction).

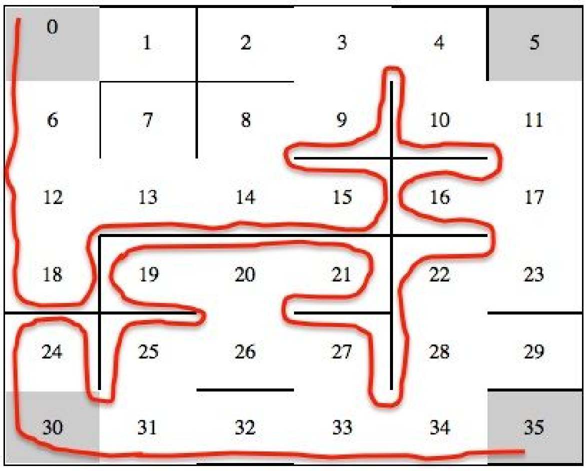

Figure 2: SHExtraSensorial movement according to right-hand rule movement

Once the simulation has started, the SHExtraSensorial will perform the actions that were specified in D2 document (second stage of the project) in each turn.

SuperHero with Flying Powers

Before starting the simulation, each SHFlight must calculate its route through the map. The route for a SHFlight is based on moving from its initial square to the DailyPlanet square by following the shortest path between these two different squares.

Once the simulation has started, the SHFlight will perform the actions that were specified in D2 document (second stage of the project) in each turn.

Villains

Before starting the simulation, each Villain must calculate its route through the map. The route for a Villain is also based on the Wall Follower algorithm. However, in this case, this type character will use the left-hand rule, instead of the right one (like the SHExtraSensorial). We will assume that this character is initially touching the East wall of the initial square (it would be looking at South direction).

Once the simulation has started, the Villain will perform the actions that were specified in D2 document (second stage of the project) in each turn.

3.- Logging the results into the record.

The results must be recorded into a log file according to the specifications described in the document for the second stage (D2).

4.- Complete the unit tests set

The unit and integration tests for the systems must be completed, according to the new classes and functionalities added in this third stage.

5.- External documentation

For the official exam of the subject (in January), the students must present the whole external documentation of the system, namely the user and programmer guides, according to the template described in the appendix of the Unit 2 "*Unit02_OOAD_AppendixDocumentation*". The template for the documentation is available in the theory folder of week 03.

6.- Graphical User Interface (GUI)

Optionally, a Graphical User Interface may be implemented for the project. This interface has not a specified format so that the students may choose its appearance and options. Anyway, the interface must allow executing the system without the user intervention from the beginning to the end of the simulation. The implementation of the GUI does not exempt the students from generating the log file with the final results.

In order to create the GUI, an example of interface will be provided in the lab dedicated to this unit so that the students may extend the interface if they wish.

THIRD DELIVERABLE (D3)

Considerations:

- The name of the log file must be **record.log** and it must **strictly follow the format** specified in the document for the D2.
- The program must execute from beginning to the end **without requiring the user intervention**.
- The **output** on the screen **has not a specified format** so that the students may do it as they wish.
- The students must make a right use of the **inheritance and polymorphism** concepts.
- It is recommended that the project includes the implementation of the **Graph** data structure and the **design patterns** that have been explained in the subject.
- The name of the init file to be loaded will be specified as a parameter of the project, according to the syntax (this will be explained in a lab session): <name_of_the_program> <file_name>
  Example: **java Map init.txt**
- Additionally, the students must follow all the same restrictions and requirements specified in the previous stages.

The deadline for this stage will be: **Wednesday, 20th December.**