

2020

# Mirrros and Lasers: The Game

DOCUMENTATION

BY: JUAN ESTEBAN CAICEDO A.

## **Functional requirements**

Besides displaying a main menu of 3 main options that the program implements (1- Play the game | 2- Show the leaderboard | 3- Exit the program), the system must be able to:

**RF1:** Allow the creation of a grid using recursion. So, firstly, it is asked to the user and in a same line, his nickname, the number of rows (n) and columns (m) of this grid, and the number of mirrors (k) that he will be randomly placed in some of its cells, inclined to the left (\) or right (/). However, it is always verified that the number of columns is not greater than 26 since for the columns a letter of the English alphabet is assigned; On the other hand, in the case of the number of rows, each one is identified with a number, so there isn't a limit as such; and in the case of the number of mirrors, it is verified that the number of these is not greater than the number of cells of the grid (size of this: n X m).

**RF2:** Allow the shooting of a laser beam in any cell located on an edge or corner of the grid. For this, first of all, the user's nickname and the mirrors that user has yet to guess are printed on one same line. Then, below, the grid of size n X m that the user chose is displayed and he/she is asked to enter a command, which will represent the nomenclature of the cell where he/she wants the shot to start, and if the cell is a corner, he/she must also type and without spaces the direction of the shot, be it horizontal ('H') or vertical ('V'). After entering the command correctly, it is verified that this cell exists through its nomenclature, and if yes, it proceeds with the shot that will take a direction (to the left, to the right, up or down) according to the cell location of the user choice. Finally, a temporary grid is shown to the user with the letter 'S' in the cell where the laser beam shot started, and with the letter 'E' in the cell where the laser beam ended, after eventually bouncing through the mirrors.

**RF3:** Allow the display of a sub-menu with the following four options: 1) Shoot the laser beam again: see explanation of **RF2**. 2) Guess the location of a mirror on the

grid: see explanation of **RF4**. 3) **Cheat mode**: a grid is displayed where the user can see in advance the location of the mirrors and their inclination, without having won the game yet. 4) Return to the main menu: the user writes the word "menu".

*(If the user indicates a different option than the previous ones, he will get an error message.)*

**RF4:** Allow the user to be able to guess the location of a mirror in a cell of the grid. For this, it will type a command composed as follows and without spaces: the first character is 'L' (Locate) followed by the cell nomenclature, followed by the inclination of the mirror: 'L' (Left) or 'R' (Right). Now, if the location and inclination of the mirror is correct, the grid is displayed to the user with the mirror revealed in that cell (like this / or like this \), from now on. In the event of failure, the grid is displayed to the user with an 'X' in that cell, from now on.

**RF5:** Calculate a score for the user based on his/her performance in the game; in other words, according to the number of shots fired, the number of failed attempts to locate a mirror, the size of the grid, and the number of mirrors to be found in it. This score is calculated and always displayed to the user at the time of winning the game (by guessing all the locations [and inclinations] of the mirrors present in the grid) or at the time of returning to the main menu without necessarily having won the game. Also, this score is stored in a binary search tree ordered by score, where each node of said tree has stored the user's nickname, his/her score, the number of rows he/she chose for the grid, the number of columns he/she chose for the grid, and the number of mirrors to find on the grid.

**RF6:** Display the content information of the nodes of the binary search tree that stores the players with their nicknames, scores, number of rows, columns and mirrors that each one chose for the grid. All this information is displayed in order of score from lowest to highest, in a vertical enumerated list, the result of traversing this binary search tree in inorder.

