# CS1527 Object-Oriented Programming

# Mini-Project: Text-Based Maze Game

# 2019-2020

# Name: Darius Dragnea

# ID: 51984000

1. Description of the game

The game I have implemented is a simple 2D maze game presented on the terminal with ASCII characters where the player’s character - the hero -, interacts with monsters and goblins inside the maze with the aim of fighting all the monsters and winning the game.

When the player runs the game, a menu with some commands is displayed:

A close up of a logo

Description automatically generated

Thus, the player has to choose from five options: press ENTER for starting the game, press L for loading a game they have already saved , press T for seeing the league table, press S for seeing the instructions of the game or press E for exiting the game.

If the player decides to start the game, they will be asked to choose a difficulty.A screenshot of a cell phone

Description automatically generated

The difficulty of the game reflects the strength of the abilities of the monsters and goblins which will be displayed latter in the maze . The four difficulties are: easy, medium, hard and very hard, and the player has to press a number between 0 and 3 in order to choose it.

(NOTE: by choosing an option or a difficulty, the player should only press the chosen key once. The character will be not displayed on the screen, but the program will still run the command)

Once the difficulty is chosen, the game itself starts, featuring:

* The difficulty of the game
* The maze with the 3 main characters: 1 hero( H) , 5 monsters(M) and 5 goblins(G), walls (#) and the route(-). The hero can only walk on the route and cannot jump over the walls
* The hero’s initial position which is randomly determined inside the maze, health(100) and coins(1000).
* The creatures’ initial position which is randomly determined inside the maze and their names suggesting their abilities( Thief Monster, Fighter Monster, Gamer Monster, Wealth Goblin, Health Goblin and Gamer Goblin). The abilities of the creatures are randomly determined as well such as there exists at least one type of each possible ability

A close up of a black background

Description automatically generated

The hero will start moving through the maze by pressing the four arrow keys, thus losing a health point. After moving once, the program will display the updated maze featuring hero’s new position, the number of moves he has done so far, the updated health and coins, and the number monsters he has visited so far.

A screenshot of a computer

Description automatically generated

In addition, the player has the choice to press M for seeing the current map of the game featuring the creatures’ positions in the maze in real time or to press H for seeing the instructions of the game.

A screenshot of a cell phone

Description automatically generated

A screenshot of a computer screen

Description automatically generated

By pressing M or H, the player will not see the pressed key, but only the instruction the program has followed automatically. Also, during the game, the player has the choice to save the game by pressing S. Nothing will be displayed, but the game will be automatically saved. Therefore, the player can exit the game and rerun it by choosing the option LOAD GAME.

A screenshot of a cell phone

Description automatically generated

The hero’s purpose is to fight all the monsters( at least once) and if he manages to do it without losing all his health, he wins the game. When he meets a creature, some suggestive messages will be displayed on the screen in order to keep the player updated with the outcome of each fight with monster or meeting with a goblin:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

After meeting a goblin, the creature will disappear and the hero will be displayed on the goblin’s position, whereas the monster will still figure in the maze, regardless of the result of the fight and in this case, the hero will be on the monster’s position, invisible on the screen. After fighting a monster, the number of visited monsters will be updated (If the hero fights the same monster twice, the number will not increase).

A special ability of both monsters and goblins is the gamer. When meeting a gamer monster or goblin, the player is asked to play a rock-paper-scissor game with them. Thus, they have to choose between rock (press R), paper (press P) or scissors (press S). The creature will randomly choose an option and depending on the result, the hero will lose or win coins and health points. In case of draw, both sides are asked to replay the game, until one of them wins.

A screenshot of a computer

Description automatically generated

In case of losing (health <= 0), an appropriate message will be displayed on the screen, and the menu will be displayed and the game will reset. All the game is an infinite loop that can be stopped only by pressing the key E.

A screenshot of a computer

Description automatically generated

In case of winning (the hero manages to visit all the monsters BEFORE losing all his health points), the player will be asked to enter a username that will be saved in the league table along with the coins gained.

A screenshot of a cell phone

Description automatically generated

The league table can be accessed from the menu by pressing T. It is divided in four parts by difficulty , and for each difficulty , only the top 10 players will be displayed. Players are sorted in the league tables ONLY by the number of coins, which means that the players with the same number of coins will not be on the same place.

1. Classes used

When it comes to the classes I have used in designing the maze game, the main one is Game, where almost every instruction of the game happens. This class features, as objects attributes, the hero character and the environment objects which belong to Hero and \_Environment classes and several class attributes such as \_users, \_coins (for saving all the users of the game and their score) or \_files , a 2D array that consists of the 4 league tables saved in 4 different text files and each of the 4 difficulties. Most of the Game class’ methods are static as they design the game ( e.g *update\_league\_table*, *menu*, *choose\_difficulty*), with the exceptions of the main method *play* that runs the whole game , the method *type\_user* that saves in a class variable the name of the player and the number of coins gained, the method *load* which loads the game that has been already saved with *pickle()* or the methods that initializes the hero.

In order to generate the characters of the game, I first created a Character class that generates randomly the coordinates of all characters of the game, sets their values ( method used for loading the saved values by using pickle) and returns them. The two subclasses of Character are Creature with its own subclasses Monster and Goblin, and Hero.

The Creature class consists of a class variable used for the difficulty of the game which is being returned and initialized by two class methods within the class. In addition, a static method for randomly determining the rock-paper-scissor-game has been created in this class, such that both Monster and Goblin classes would inherit it.

The Monster and Goblin classes are relatively similar: they have as class attributes the number of objects created, all the coordinates of the creatures and an array of the possible abilities of the creatures, ranging from 1 to 3. The abilities array is randomly created such that each type of creature is generated at least once. Furthermore, both of the classes feature methods for the first two abilities ( thief&fighter/wealth&health), methods for getting and setting the ability of each creature and methods for loading the creatures’ positions and abilities that have been saved using pickle, for the creatures reset and for the creatures details. The Monster class contains, in addition, a class variable array that saves the monsters which fought with the hero. All the abilities of the creatures are calculated by using probabilities and coins/health points lost/gained which are higher or lower depending on the game difficulty.

The Hero class is a subclass of Character class that contains the main methods that make possible the movements of the hero through the maze, the fights with the creatures, the outcomes displayed on the screen by using messages including the interactional rock-paper-scissors game ( implemented by a static method). Another important method of this class is *save-game* which is used for saving all the necessary elements of the game in a file by using pickle method. These elements will be automatically loaded by using a method of the Game class mentioned above.

Finally, the \_Getch class has been used within Hero and Game classes and its purpose is to allow the player to choose a single key of the keyboard. It makes possible the movements inside the maze by using the arrow keys.

1. UML Class Diagrams

A close up of a piece of paper

Description automatically generated