Let's travel the world

Techno Dreamers

May 2022

Team Members

- Nevina Dërvishi
- Elisa Gjuraj
- Marko Maksuti
- Glen Teneqexhiu

Contents

1 Executive Summary

1.1 Project Overview

This project is an educative game that will enable everyone, especially tourists of different ages to learn about traditions and customs of Medvegja, Presheva and Bujanovac. They will have the opportunity to be entertained by playing with turtles, doing quizzes and puzzles.

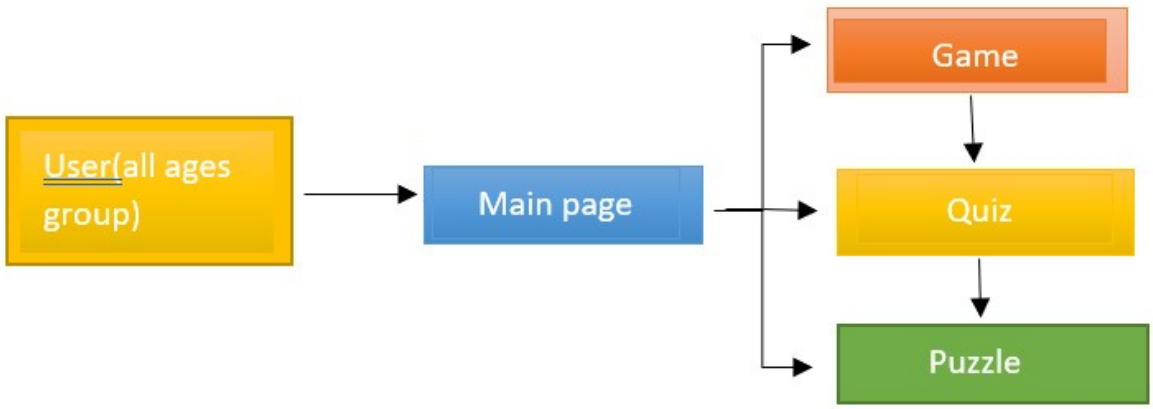
1.2 Purpose and Scope of This Specification

We researched and gathered information relevant to our project, we interviewed people as well to have accurate information. The main idea of the game was to be accessible to any age. The game aims to entertain every tourist and make it easier to get to know the cities along with their beautiful traditions. We decided to create this turtle game, as an understandable, simple, educational and above all fun game. This game consists of three levels. At first the tourist will choose the desired destination. Information about the traditions and customs of the selected destination, interesting and useful information for tourists will then be displayed. After the game starts, when the blue turtle (the player) catches the black turtle within a time limit, the player gains points and eventually passes to the next level. If you don't catch the other turtle in time, you lose. But even though you lose, the tourist will have the opportunity to pass to the next level with the help of a quiz about the selected destination. Each level will have a question in case of loss. If the tourist passes all the levels, he will solve a puzzle and have the opportunity to see the wonderful destination he has chosen. We as a team aimed to create an effective design, with bold colors, in combination with the predominant colors of each destination.

2 Product/Service Description

2.1 Product Context

To create this project we have taken inspiration from the systems that are in museums, in which it makes it possible for you to choose which of the data you want to read. It is a dependent project, but the difference between that system and our application is because we are doing it in game form. This application depends on other connected systems, as we are building this through the Python language and we need some Python framework to execute the program. The context of this game is informative, to introduce foreign tourists to new places. The following diagram shows the components, interconnections and external interfaces:



To use the program, after having installed Python, you simply launch it. Immediately after, the program directs you to the main page. In the main page, you find games, quiz and puzzle. Quizzes are directly related to the information about the place you chose.

2.2 User Characteristics

This game can be played by all age groups, all foreign tourists from children to the elderly.

Name	Stela	Simon
Surname	Simpson	Dhimitri
Age	12	25
Reason of choosing our program	I want to learn some interesting things about Presevo since I have never visited.	I want to learn about Medvedja because I know very little about that place and I want to know more.

2.3 Negotiation

Negotiation is an essential aspect of any software project since every requirement, task, and other project-related decision is determined through negotiation between coworkers and stakeholders. It signifies that both sides' requests have been met. Our team has always tried to negotiate with the stakeholder in requirements elicitation. Although we had a lot of inconsistencies of ideas, and concerns with each other we realized that through these clashes we reached a good conclusion of our project.

2.4 Assumptions

- The user is part of a general audience, tourists, ranging from small children to the elderly, they are unfamiliar with the history or traditions of the locations we are trying to teach them about.
- The user is assumed to have little to no experience with electronic devices.
- The program is compatible with all operating systems (Windows, Mac-OS, Linux etc.) which have Python available to them.

2.5 Constraints

- Code is written in Python programming language.
- Only works with devices which have a Python Framework (e.g. PyCharm).

2.6 Dependencies

- The Turtle module must be part of the packages that comes with Python.
- There is no internet connection required to run the software.

3 Requirements

- User will be presented with a start screen where they can select which of the three cities to start with. (R1)
- User will use the mouse left-click to select the city.

- User will use the arrow keys to control the character.(R2)
- User is on a timer, if the timer runs out before a set amount of points is reached, the level fails. (R3)
- Should the user run out of time, they will be presented with a quiz as a second chance. If they select the correct answer, they may continue to the next level, else they'll have to repeat the current level. (R4)
- Once the user clears all levels, they will be presented with a puzzle. The puzzle will be an image of the chosen destination. (R5)
- Information/imagery must be informative and educational. (R6)
- Finished product must be easy to use/navigate. (R7)
- Information must be easy to understand, not too long. (R8)
- Finished product must be available in English. (R9)

3.1 Functional Requirements

Req. ID	Requirements	Comments	Priority	Date	Review
R1	Start Screen and navigation via mouse. Three sections for each province.	Sections separated in three columns, each with a picture of the respective province.	1	30/05/2022	Nevina Dervishi
R2	Controls with arrow keys	Generic controls	1	24/04/2022	Marko Maksuti
R3	Timer, Point and Info. mechanic	Each point earned adds to timer, X amount of points(display string)	1	01/06/2022	Marko Maksuti
R4	Quiz	User runs out of time, solve quiz to continue, else game over	1	10/06/2022	Elisa Gjuraj
R5	Puzzle	Left-Click, bonus feature	2	05/06/2022	Glen Teneqexhiu, Marko Maksuti

3.2 Non-Functional Requirements

UI Requirements

- Start Screen; three columns each displaying images of their respective provinces. Top on screen user is given prompt "Select a destination". Each column has the name of the province at the top.
- User controls the character using arrow keys, timer and points are displayed on the screen in real time, amount of points needed to reach the next level is also displayed.
- When the timer runs out, the user will be asked if they want to repeat the level or solve a quiz. If they choose the quiz, the user will use the mouse to click on an answer.
- Each city has its own set of images to be displayed during the puzzle.
- Information will be displayed in the center of the screen, the game will be paused during this time. The information will be automatically dismissed after a few seconds.

- During the puzzle section, the user will be presented with a scrambled image related to the province separated in a grid. The user will click on the tiles to move them.

Usability

- Simple controls with mouse left-click and arrow keys, suitable for individuals of all ages and backgrounds.
- It takes the user 30 seconds to complete one level.
- information displayed is concise and digestible.
- Users have had positive reactions with the controls, claimed they found them easy to get used to.
- Users reported the information was interesting and brief.
- Users reported they were satisfied with the length of each level.

Performance

- Product does not require advanced hardware or software to function, can run on any average computer or laptop.
- Product does not require an internet connection to function, making it independent.
- Information to be computed by product is simply mouse or key inputs from user (limited to left-click and arrow keys).

Manageability/Maintainability

- Product has a simplistic structure, leaving little room for errors.
- There are no special requirements for maintaining product health since its fairly simplistic in its structure.
- There are no special operations/actions required by the user.

Security

- Product is designed to be available to the public, as such the interactions users have with it are limited to mouse left-click and arrow keys.
- Product will be downloaded onto the desired device to be ran offline.
- Internet connection is not required to use the product, meaning internet security measures do not apply.

Standards Compliance

The activities included under the category of software engineering/development, as well as the procedures to be followed by taxpayers exercising such activities, which are subject to 5 percent corporate income tax (CIT), are now provided for in Decision No. 870 approved by the Albanian Council of Ministers on 12 December 2018.

Other Non-Functional Requirements

- Information/imagery must be informative and educational. (R6)
- Finished product must be easy to use/navigate. (R7)
- Information must be easy to understand, not too long. (R8)
- Finished product must be available in English. (R9)

3.3 Domain Requirements

- The only individuals allowed to edit the content the product provides are developers of said product.
- In order to edit content provided by the product, a request must be made by the stakeholder/s.

4 Design Thinking Methodologies

4.1 Negotiation

Negotiation is an essential aspect of any software project since every requirement, task, and other project-related decision is determined through negotiation between coworkers and stakeholders. It signifies that both sides' requests have been met.Our team has always tried to negotiate with the stake- holder in requirements elicitation. Although we had a lot of inconsistencies of ideas, and concerns with each other we realized that through these clashes we reached a good conclusion of our project.

4.1.1 Negotiation with the stakeholder

We had a negotiation between friends after we decided to implement the best requirements. We then asked our project stakeholders what our idea looked like, we had moments when we changed some small requirements but they did not pose a big risk to our program, we met very often and discussed about these requirements to do with good of this information platform of these countries.

4.1.2 Negotiation within the team

We as a group have had meetings once a week. We have looked at the work of each member of the group to what extent it has reached and we have discussed new ideas members of the group wanted to add to the program. We had many disagreements with each other but we tried to select the best ideas. For example, a member of the group wanted to use audio in the program but we deemed this feature unnecessary and we did not add it to the final version.

4.2 Empathy

To create a good program, it is very necessary to see this program through the eyes of the user and to understand how they feel or think, and what allows us to do this is Empathy.

Empathy is something very important which is closely related to team members as the more understandable we are with each other the simpler we have to create a good program. For example each member has explained his task in detail they even made some explanations of the code in order to make it easier to build the documentation.

We have even used many pages about user sensitivity and understanding to create the best, most understandable and compelling program. We have sometimes even neglected our ideas to do what a user wants.

4.3 Noticing

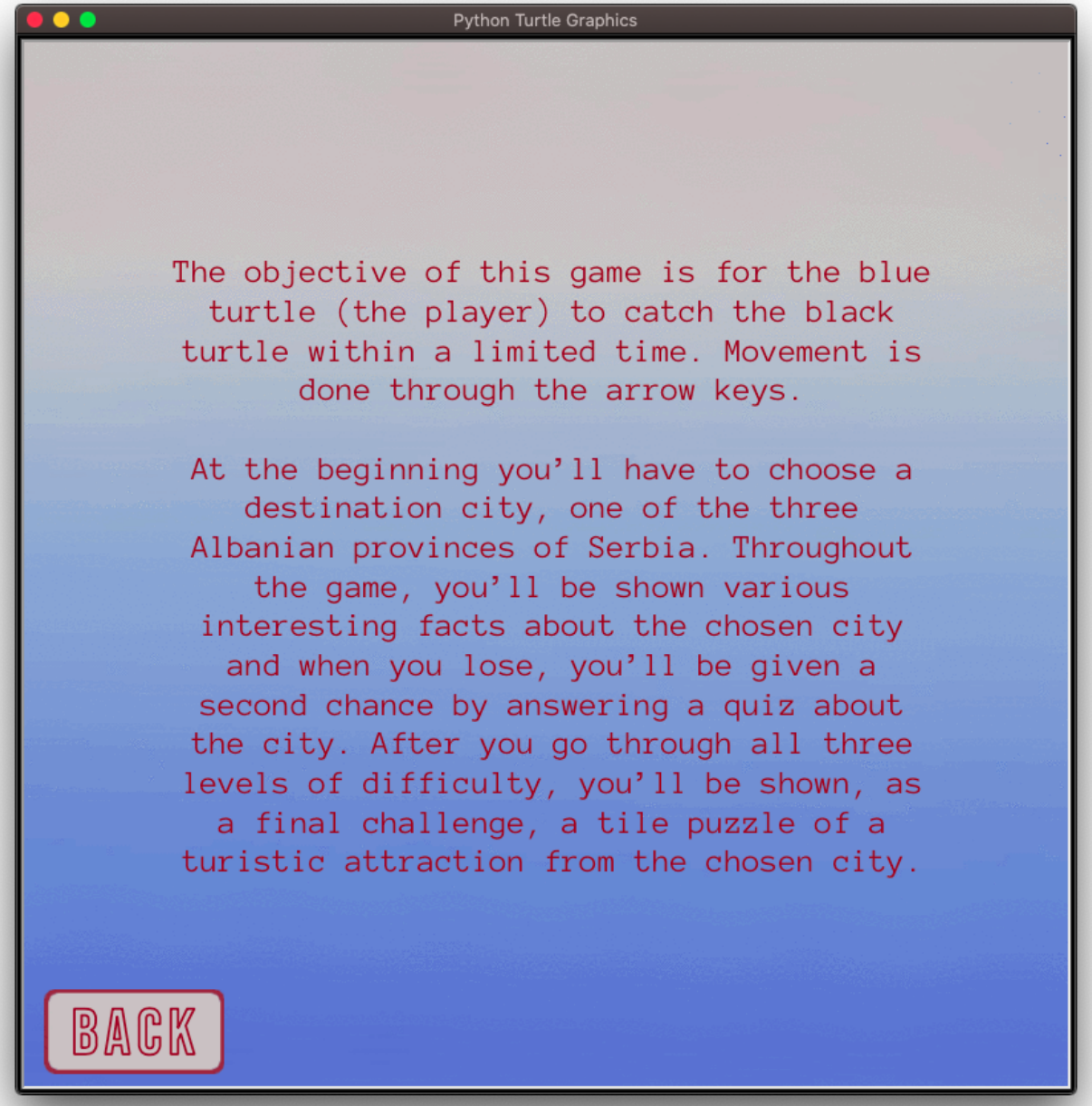
This application is very simple to use, everything that is given in it is direct, the information we have entered in it is concise and understandable to anyone. It is an attractive application for those who play it because every turtle you catch earns you one point. So, when a user plays it, he gets many points in the game as well as information about the place he has selected and this makes the user more keen on playing the game as much as possible without losing, to collect as many points and get as much information as possible.

4.4 GUI

Below are attached screenshots for every page of the program.

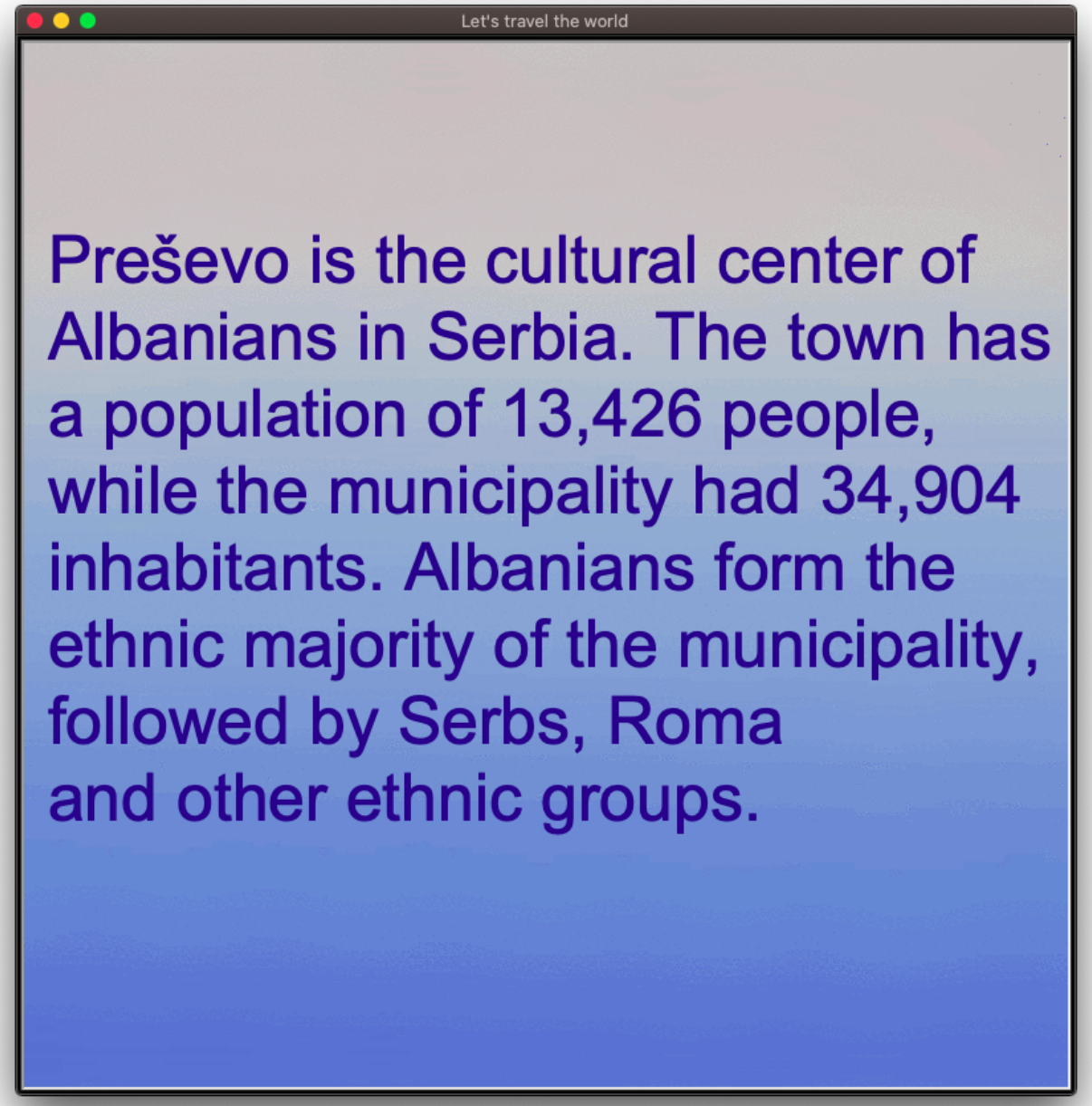
Main menu:

Help screen:

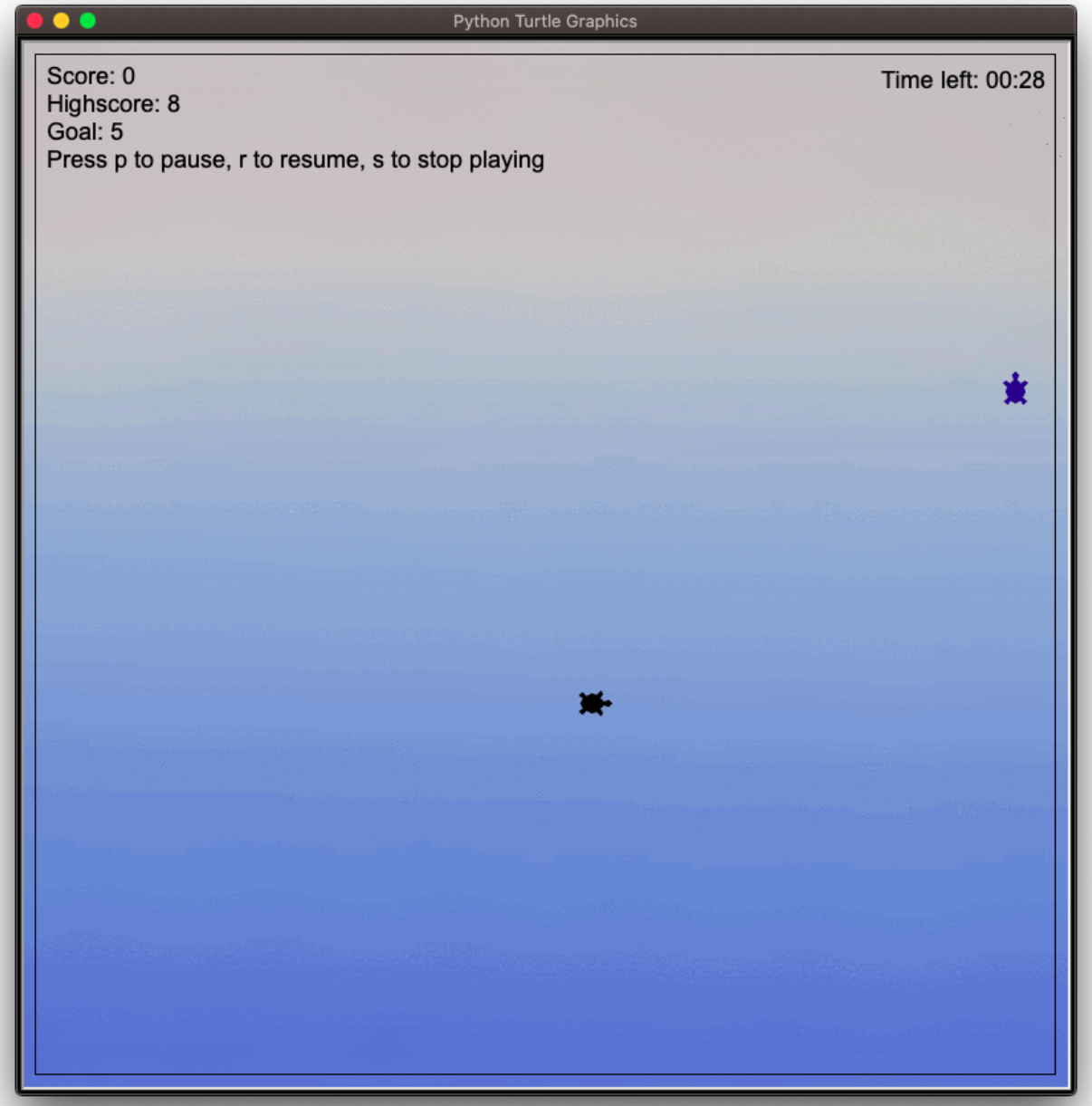

Python Turtle Graphics
The objective of this game is for the blue turtle (the player) to catch the black turtle within a limited time. Movement is done through the arrow keys.
At the beginning you'll have to choose a destination city, one of the three Albanian provinces of Serbia. Throughout the game, you'll be shown various interesting facts about the chosen city and when you lose, you'll be given a second chance by answering a quiz about the city. After you go through all three levels of difficulty, you'll be shown, as a final challenge, a tile puzzle of a turistic attraction from the chosen city.
BACK

"Choose destination" screen:

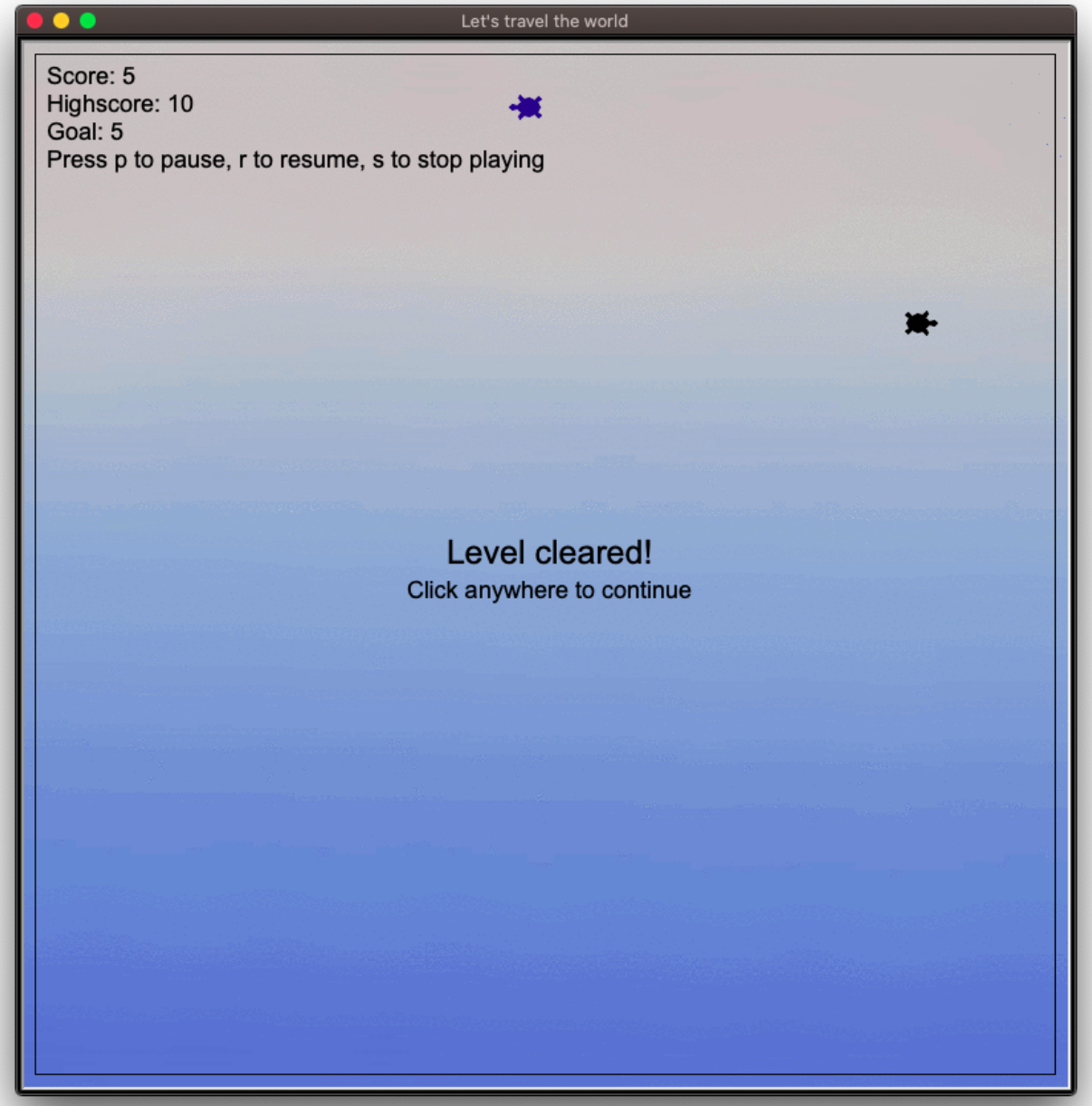
Information:



Game screen:

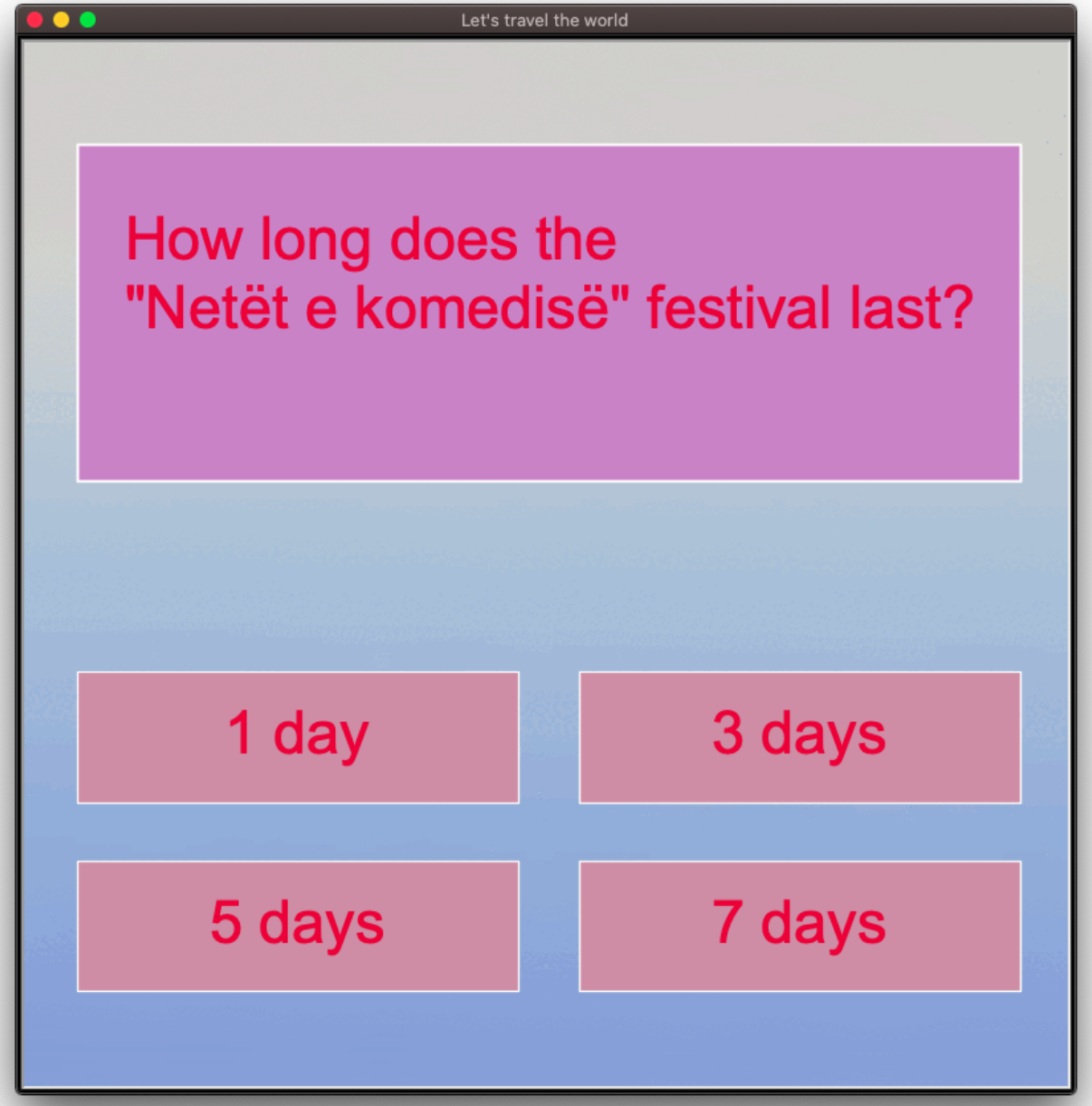

Python Turtle Graphics
Score: 0
Highscore: 8
Goal: 5
Press p to pause, r to resume, s to stop playing
Time left: 00:28

”Level cleared” screen:

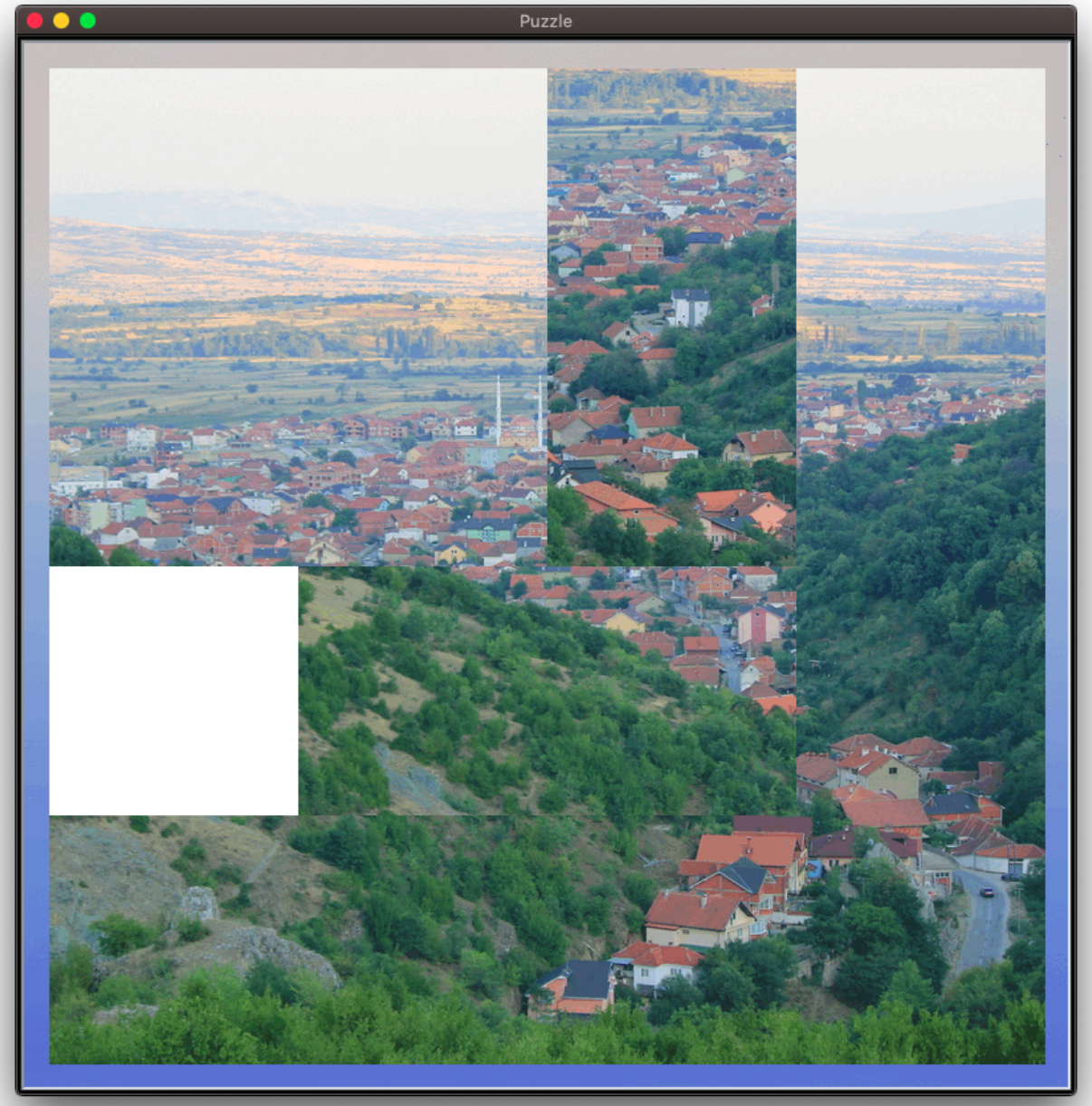


”Level failed” screen:

Quiz:


Let's travel the world
How long does the
"Netët e komedisë" festival last?
1 day
3 days
5 days
7 days

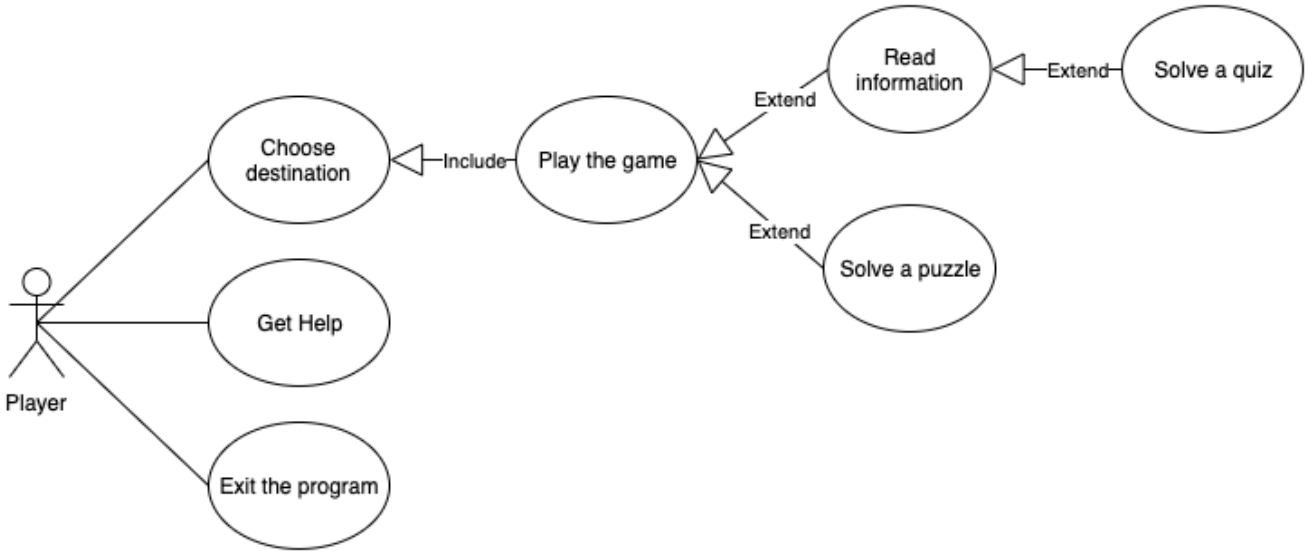
Puzzle:



5 Software Design

5.1 Use Case

This program is simple and it only has one type of user (actor), the player. On the diagram below you can see the functions that the player will be able to perform while using this program and how those functions are related to each other:



The first use cases are "Choose destination", "Get Help" and "Exit the program"; those are connected directly to the user because they do not depend on other uses cases in any way. Those are actions that the user will be able to perform right from the launch of the program.

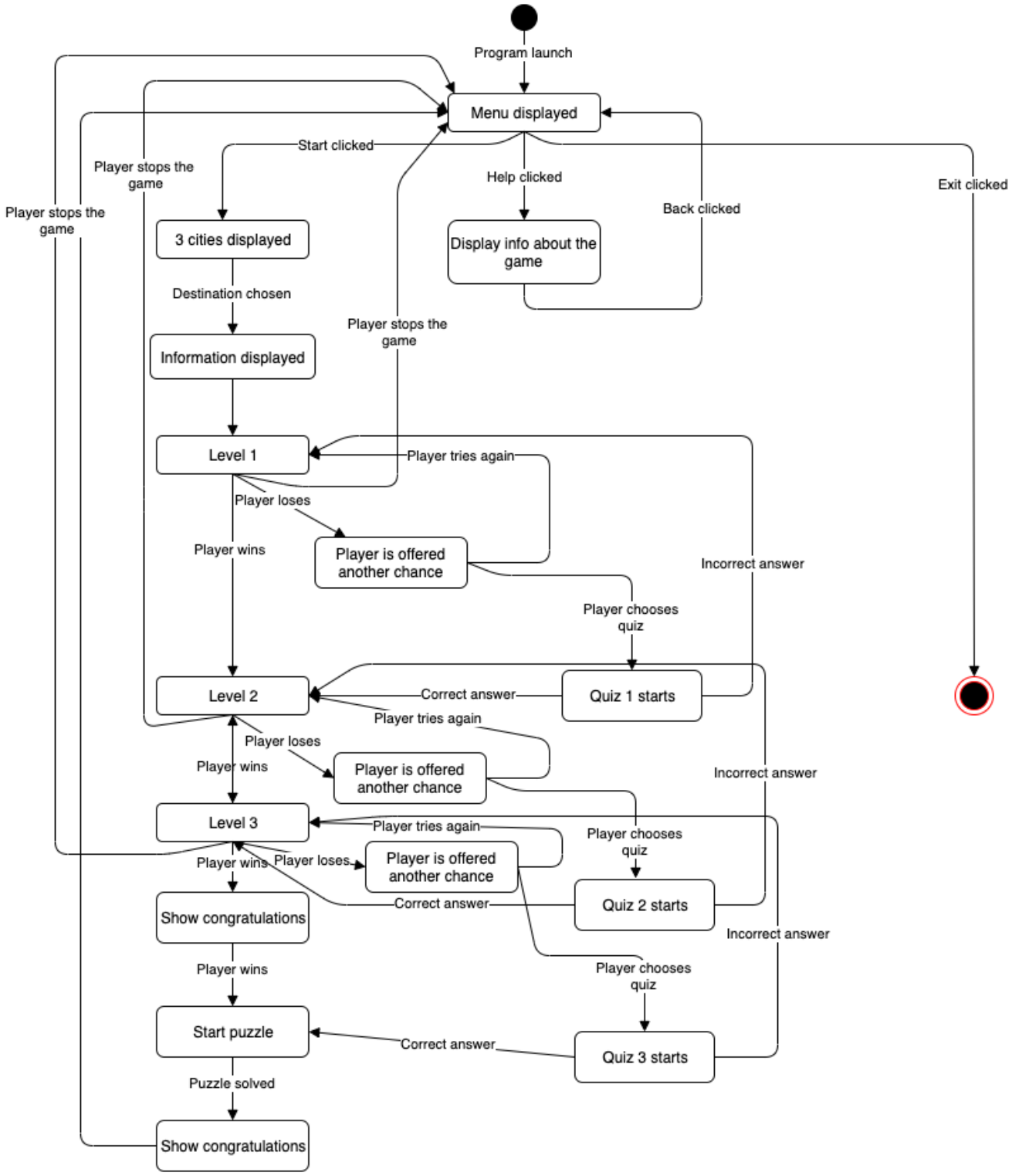
The next use case is playing the game. In order to play the user has to choose a destination first, hence why "Play the game" and "Choose Destination" are connected via an "Include" relationship, in this case the "Play the game" use case includes the "Choose Destination" one.

Then during the game the user can read information about the destination they chose. While this part requires the game to have started, it is not crucial for the game itself, but an extension of it, hence why it's connected via the "Extend" relationship. Similarly, solving a puzzle is an extension to the game.

The last use case is "Solve a quiz", this an extension to the "Read information" use case. Solving a quiz is not required after reading the information, but an extension that comes after the user reads the information.

5.2 State Diagram

Below you can see the state diagram. This diagram shows how the program changes states depending on user input.



Right after program launch, the program displays a menu. From that menu the user could click "Start", "Help", or "Exit".

"Exit" will just terminate the program, as shown by the black-filled circle inside the red one.

"Help" will display a help screen with information about the game. From the help screen the user can click "Back", which will redirect the user back to the main menu.

"Start" will make the program display an image of three cities. From there the user can choose one of them

as their destination.

Once a destination is chosen, relevant information about the city will be displayed and shortly after level 1 of the game will start. The user will then either lose or win the level depending on the points they get.

If they win the level, they'll automatically pass to the next one and another piece of information is shown. If they lose they'll be offered another chance if they answer a quiz correctly. If they choose to try again instead, level 1 will start again.

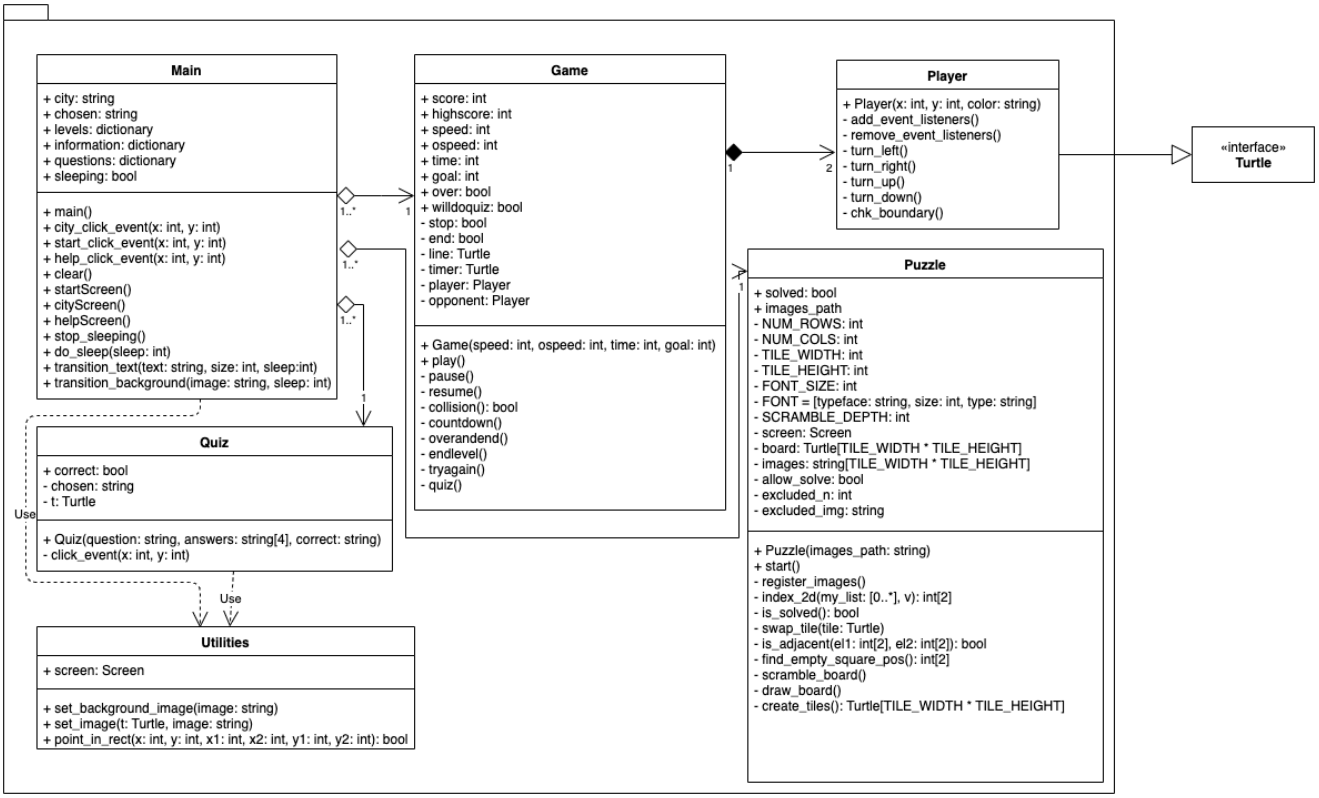
If they choose the quiz, they'll be shown the questions and the user will have to choose one of the four answers. If they answer correctly they go to level 2, otherwise they'll have to repeat level 1.

Similarly the same actions are repeated with level 2 and level 3.

If the user manages to win level 3, a congratulations message will be shown to them. Shortly after a bonus puzzle will start. If they solve that too they'll be shown another congratulations message before being brought back to the main menu with the option to play again.

5.3 Class Diagram

The following diagram shows the classes (modules) of the program and how they are connected.



The first class is Main. This is the entry point of the program. It has attributes and methods that control the main menu, the "Choose destination" screen, the "Help" screen and all the variables associated with them. It also connects all the classes and handles the main logic of the program. Main is connected to the Game class, the Quiz class, the Puzzle class and Utilities.

Utilities is a module that contains some general functions that will be used by Main and Quiz, it has no function of its own, and this is shown with the "Use" relationship connecting it to Main and Quiz. The dashed arrow connecting them means this is a dependency relationship.

The Game class handles all the code related to the game logic. It handles the turtles, keeping track of the score, highscore and timer. The Game class takes as input from the Main class the speed of the two turtles (which change depending on the level) the time set for a level and the amount of points the user has to reach to pass the level. The play method is called to start the game. The other attributes and methods which are shown with a '-' sign are private and are supposed to be used only internally by this class. The Main class will also be able to access attributes that describe the game status, such as score, highscore, willdoquiz, over, etc. The Game and Main classes have a one-to-many relationship, since there can only be one Game per Main,

but you can have multiple Main instances each connected to one Game instance. The arrow is diamond-shaped and filled white, which means this is an aggregation relationship. This means Game is part of Main but can exist independently. Game itself is connected to the Player class, which is just a subclass of the external Turtle module, with some extra attributes and methods that make its control easier (this inheritance relationship with the external module is shown through the white triangle arrow). The connection is two-to-one, as there can only be two Player instances for each instance of the Game class. This is a composition relationship shown by the black diamond. Player is part of Game and can't exist without Game.

Next, the Quiz class. This class handles the question/answer part of the program. It will take as input a question, an array of 4 answers and the correct choice. This class is also connected to Main. Main will create an instance of it whenever the user decides to answer a question. This is decided by the Game class and returned back to the Main class, using the attribute 'willdoquiz' in Game. The only public attribute is "correct" which tells the Main class if the user answered correctly. The Quiz also has a one-to-many relationship with Main, as in it can only be connected to one Main but Main can hold multiple Quiz instances if needed. The relationship is aggregation.

Lastly the Puzzle class handles the Puzzle code. The class takes as input from Main a string containing the path to the folder of images that will be used during the quiz. Main will then use the public 'solved' attribute to check if the Quiz was solved. The Puzzle class has the same type of relationship with Main as Quiz does, one-to-many, aggregation.