**Trivia Assignment Documentation**

# Introduction

The assignment is a quiz game, based on Open Trivia DB. The application uses the OpenTriviaDB to collect questions.

It collects the questions and creates quiz rounds for the player(s). One round is finished when every player chose an answer for the actual question, or the time has expired for the players. Every player has a time limit to answer all of the questions.

In the end, the result is going to appear and show the reached scores of all of the players and the menu appears again.

The game was developed in Unity 2021.1.6f1 version.

# Project structure

## Folder structure

Text

Description automatically generated with medium confidence

* Prefabs folder contains the created UI prefabs
* The ProjectAssets folder contains the basic static game files, like fonts and sprites. It has two folders: Fonts (with the fonts) and the UI with the basic UI sprites like buttons.
* The Rendering folder contains the rendering pipeline assets.
* The Scenes folder contains the scenes, currently, only one scene exists.
* The Scripts folder contains the scripts used in the game.
* Finally, the TextMeshPro is a third-party library to handle text elements easily on the surface.

## Scene structure

Text

Description automatically generated

* The MainCamera is the camera of the scene.
* Canvas contains every element of the game. It has 4 panel children.
  + MenuPanel contains the first section of the game, the main menu. It has a header with a title and also a body with a form. The form contains the user inputs fields: the number of questions, category selection and the number of players.
  + LoadingPanel contains the loading circle and if the panel is active, then the loading circle animation is working.
  + QuizPanel contains the gameplay elements. It has a header which has the question, the question information, like actual question number and the actual player’s score.
  + PopUp panel is responsible for the PopUp window handling. The PopUp window is a prefab, and every instance of it will be added to the PopUp panel as a child. The UIPopUpHandler has been attached to the panel, that handles the prefab creation.
* Manager is a non-visible object. It has the GameController and the UIController instance.

## Prefabs

### AnswerButton

The AnswerButton prefab corresponds to one answer button. The buttons will be added to a grid. Depending on the type of the question (true/false or multiple choice), there are different number of possible answers. The AnswerButton instances will be created for every question and deleted at the end of the actual question.

### PopUpPanel

The UIController is responsible for calling the UIPopUpHandler to create PopUpPanel instances to the PopUp panel. Every PopUp window is a different instance created by the LIFO way. It means that the first created PopUp window will de front of the others, so it will be visible. If other PopUp windows will be created, then those will be visible, if the first closed, like a queue.

# Components

## Model

### Answer

The Answer class represents an answer. Every question has some incorrect answers and one correct answer. This is the smallest object of a question, it has two parameters, the option, which is the answer text, and a bool which checks whether the answer is correct or not.

### Category

The Category class is necessary for the JSON deserialization of the category requests. It represents the categories and their properties based on the OpenTrivia DB.

### Player

The Player class represents a player of the game. Every player has an id, name, remained time, and a bool which checks whether the player is still playing or already finished the game (remained time = 0).

### Question

This class helps to process the requests from the OpenTriviaDB. The response comes in JSON format, and it represents the question information.

### GameState

The GameState stores the actual state of the game. It is used by the GameController. When a game has started, a new GameState instance is going to be created, and it stores everything about the gameplay: players, questions, round information. It also has some additional methods.

## Backend

The backend is responsible for communicating with the Open Trivia DB. An ITriviaAPI interface has been defined, and OpenTriviaApi class implemented the interface. This solution helps to change the class to communicate with another API easily.

The OpenTriviaAPI stores the base URL, and it can handle the request’s parameters, handles the JSON deserialization, and processes the results. There are two main methods, GetQuestions() and GetCategories().

The GetQuestions() returns with the questions based on the requests, and the GetCategories() collects the optional categories from OpenTrivia.

## Controller

### GameController

The GameController controls the gameplay and the backend as well. It communicates with the backend (OpenTriviaAPI) and with the UI too. It controls the logic of the gameplay.

It has an OpenTriviaApi, GameState, and a UIController instance as well. This is the central object. When the game is started, it creates a new GameState and controls the gameplay using the GameState information and methods. It defines which stage is coming after which stage.

It also sends the necessary information to the UI, and processes the user inputs.

### UIController

The UIController is responsible for the UI. It checks the form validation, shows the panels, and PopUp menus, and it handles the user interactions.

It controls the UI inputs, the panel creation, and everything which is UI related.

The application has 4 major panels. The first is the MenuPanel, this is the first, where the user can add the number of questions, chose a category, and add a player number. After there is a loading panel, which is visible until the GameController collecting the questions using the backend.

After that, there is the QuizPanel, which is the gameplay panel, it shows the questions and the answers as well, and also the player and global information, like player score, and actual question number.

# Overall

The game is playable in a single-player way (number of players equals 1) or a maximum of 8 local players. The user can add a number of questions between 1 and 50 (OpenTrivia gives a maximum of 50 questions from one response). Finally, a category can be selected.

When the user sent the inputs, a short validation happening by the UIController.

When the form has been successfully filled, the backend starts to download the questions async way and in the meantime the loading screen is visible.

After the questions are collected, the backend sends them to the GameController and it initializes the GameState and sends the information to the UIController. The UIController shows the QuizPanel with the data.

The process of the quiz: every player has a timer separated from each other. The timer applies to the entire quiz, not just one question (players get 10s for every question). The actual player can select an answer, when the answer is selected, the next player gets the turn with the same question. If a player has run out of time, then that player score will be saved and in the other rounds that player cannot choose answers.

The game has been finished if there are no other questions or all of the players have run out of time. At the end of the game, the result will be visible with the scores of the players.