F1

Software Engineering Document

Table of Contents

[Game Mechanics 1](#_Toc513682048)

[Game Scenes 1](#_Toc513682049)

[Game Flow 1](#_Toc513682050)

[Classes and Scripts 2](#_Toc513682051)

# Game Mechanics

## Game Scenes

The game will have four basic scenes:

* The “Menu” scene which server as the welcoming scene for the player, in here the player is welcomed with a simplistic splash screen with the game title and a button to start playing.
* A game scene “Level01” where the player will adapt to the role of a red cube, and begin moving forward constantly on an infinite track where gray rectangular prism will spawn throughout the track and act as obstacles that the player has to avoid, as well as avoiding falling to the sides of the track.
* A “Questions” scene, where the player will get to whenever they fall from the track or collide with one of the obstacles in the Level01 scene, the player will be asked a truth or false question that in turn decides if the player can continue playing or not.
* A “Credits” scene, where the player will get to if they get wrong an answer in the Questions scene. The leaderboards and final score of the player will be presented, as well as an input field and a button to submit the obtained score to the leaderboards. The leaderboard shows the first five players with the highest scores. This scene also presents the player two more buttons which in turn will let the player quit the game or to retry altogether from the beginning.

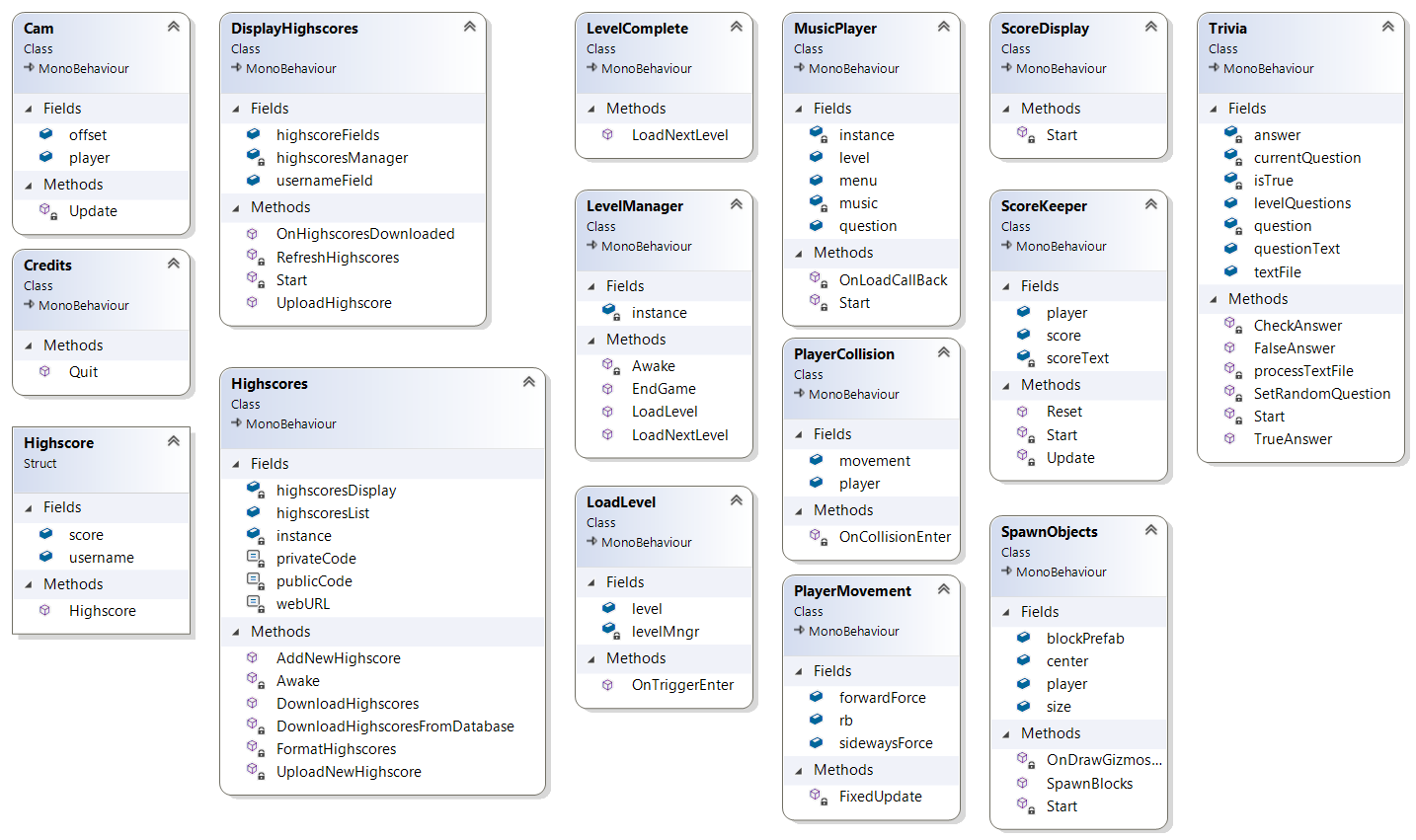
## Game Flow

* “Menu” scene will lead to the Level01 scene, the scene where the actual gameplay experience for the player occurs.
* The “Level01” scene will lead to the “Questions” scene when the player collides with an obstacle on the track, or when the player falls from the sides of the track.
* The “Questions” scenes will lead to:
  + The “Level01” scene when and if the player responds correctly to the trivia question shown in the present scene.
  + The “Credits” scene when and if the player responds incorrectly to the trivia question shown in the present scene.
* The “Credits” scene will allow the player to quit the game altogether or to retry which in turn will lead them back to the Menu scene.

# Classes and Scripts

We handle a set of scripts throughout the game:

* A Cam, that allows the camera to move alongside the player trough the scene.
* Credits, allows us to quit the application.
* Highscores, it contains methods to add, download and give format to the highscores.
  + Also included is a Highscore struct which lets us give an object structure to the username and the score.
* DisplayHighscores, allows the presentation of the highscores in the UI of the game.
* LevelComplete, let us change the level by the index of the scenes.
* LevelManager, lets us move from one scene to another.
* LoadLevel, allows us to change to alevel when a collision is detected.
* MusicPlayer, allows us play different music throughout the scenes.
* PlayerCollision, allows us to detect when the player collisions with obstacles.
* PlayerMovement, allows us to define the behaviours of the player by its inputs or conditions.
* ScoreDisplay, allows us to represent in the UI the total final score the player achieved.
* ScoreKeeper, keeps track of the overall score the player obtains.
* SpawnObjects, obstacles will appear throughout the track with the help of this script.
* Trivia, allows us to add the trivia like behavior to our game, reads the text file where the questions are stored, sets the questions to be presented, and checks if the user answered correctly.



Cam

* Attributes:
  + Player, the transform position of the player.
  + Offset, the position of the camera.
* Methods:
  + Update(), transforms the position of the camera so that it follows the player position in the x-axis.

Credits

* Attributes:
* Methods:
  + Quit(), quits the game.

Highscores

* Attributes:
  + PrivateCode, the string of the private code for the dreamlo plugin.
  + PublicCode, the string of the public code for the dreamlo plugin.
  + webURL, the url of the private database for the dreamlo plugin.
  + highscoresList[], list of the highscores uploaded to the database.
  + Instance, instance of the Highscores class.
  + highscoresDisplay, instace of the DisplayHighscores class to print into the UI.
* Methods:
  + Awake(), assigns the instance to the object and initializes the highscoresDisplay variable.
  + AddNewHighscore(string username, int score), adds a new highscore with a couroutine.
  + UploadNewHighscore(string username, int score), uploads a new highscore to the database.
  + DownloadHighscores(), downloads the highscores from the database.
  + DownloadHighscoresFromDatabase(), couroutine to download the highscores from database.
  + FormatHighscores(string textStream), gives format to the string downloaded in order to separate it into the highscoresList[]. Makes use of a public struc Highscore;
    - Makes an object with the string and score.

DisplayHighscores

* Attributes:
  + highscoreFields[], the labels for the UI to display the highscores.
  + highscoresManager, instance of Highscores.
  + usernameField, the Input Field for the user to write his username.
* Methods:
  + Start(), fetches the highscoreFields[] labels from the scene, assigns the highscoresManager to the Highscores component.
  + OnHighScoresDownloaded(Highscore[] highscoreList), assigns the highscores downloaded form the database to the highscore labels.
  + RefreshHighscores(), updates the highscores every second.
  + UploadHighscores(), add a new highscore to the database.

LevelComplete

* Attributes:
* Methods:
  + LoadNextLevel(), loads the next level based on the scene build order.

LevelManager

* Attributes:
* Methods:
  + LoadLevel(string name), calls for a scene with the SceneManager by its name.
  + EndGame(), will terminate the game.
  + LoadNextLevel(), calls the next scene in the build settings order.
  + Awake(), singleton algorithm in order so that there’s only one gameObject from this type.

LoadLevel

* Attributes:
  + Level, the name for the level to be transitioned into.
  + levelMngr, the instance of the levelManager.
* Methods:
  + OnTriggerEnter(), loads the next level based on a collision from the player.

MusicPlayer

* Attributes:
  + Instance, the creation of the same music player.
  + Menu, audioclip for the menu.
  + Level[], audioclips for the level.
  + Question, audioclip for the questions scene.
* Methods:
  + Start(), uses the singleton algorithm, lets us initialize the MusicPlayer.
  + OnLoadCallBack(Scene scene, LoadSceneMode sceneMode), assigs the music to each of the different scenes.

PlayerCollision

* Attributes:
  + Player, the transform of the player.
  + Movement, the playerMovement of the user.
* Methods:
  + OnCollisionEnter(Collision collision), if the player collisions with an object it updates the score and loads the questions scene.

PlayerMovement

* Attributes:
  + rb, rigidBody of the player.
  + forwardForce, the force utilized by the player to move forward.
  + sidewaysForce, the force utilized
* Methods:
  + FixedUpdate(), adds behavior to the player based on the keyboard input sideways force with “a” and “d” keys, and changes to the questions scene when the player falls.

ScoreDisplay

* Attributes:
* Methods:
  + Start(), assigns the game component for the score label on the UI and writes the attained score to it.

ScoreKeeper

* Attributes:
  + player, the Transform of the player.
  + Score, the int that keeps track of the users’ score.
  + scoreText, the label to show the users’ score.
* Methods:
  + Starts(), assigns the scoreText to the label component in the UI.
  + Update(), updates the scoreText to show the position in the z-axis plus the previous score.
  + Reset(), sets the score to 0.

SpawnObjects

* Attributes:
  + blockPrefab, the prefab game object of the obstacle.
  + Player, the Transform of the player.
  + Center,
* Methods:
  + Start(), .
  + SpawnBlock(), .
  + OnDrawGizmosSelected(), .

Trivia

* Attributes:
  + textFile, the text file where the questions are stored.
  + questionText, this text is the question to be displayed
  + levelQuestions, an array filled with the questions stored on the textFile
  + currentQuestion, a question is randomly chosen and set as the currentQuestion
  + isTrue, the value of this boolean depends on the question displayed
  + answer, the value of this boolean depends on the players answer
* Methods:
  + Start(), Used for initialization
  + ProcessTextFile(), Splits the text file into strings and fills the array
  + TrueAnswer(), If the players answer is true this method will be called to set the answer boolean as true and then the checkAnswer method is called
  + FalseAnswer(), If the players answer is false this method will be called to set the answer boolean as false and then the checkAnswer method is called
  + CheckAnswer(), This method makes a comparison between the answer boolean value and the isTrue boolean value
  + SetRandomQuestion(), Selects a random text from the array, displays it and assigns the isTrue boolean value based on the text position in the array, odd numbers are questions which answer is false