Assignment 2

Database

Features Implemented

- The player's score should be kept between levels using the Player Preferences, along with the current level.
- The game should include a
  - splash screen
  - end-of-game screen
  - login and sign-up screen.
  - high score sceen
  - create account
- When logging-in, the user should enter his/her nickname and password;
  - if the user is not already registered, s/he should be offered to sign-up.
- At the end of each game the player's score should be saved on a database and s/he should have access to a high-score screen.
- The database should consist of a MYSQL database hosted online.
- The score should be updated and displayed at all times during gameplay.
- The player should be able to quit the game at all times; in this case, its current score should be saved, along with the current level.
- If the player decides to re-play the game, s/he should be offered to resume the game from the level where they exited the last time (i.e., restart from the beginning of the level and load the scene accordingly).

Level 1

Features Implemented

- The player needs to navigate the level and collect four items located in random locations.
- A 3D maze that is generated randomly using the binary tree algorithm.
- All walls should be textured.
- A First- or Third-person Controller for the player.
- 4 NPCs who will be chasing the player using Navmesh navigation.
- Once the player has collected all the relevant objects, it should be move to the second level.
- The score is based on the number of items collected.
- The level is restarted is the player is caught by the NPCs.
- The player should be able to eliminate them using a basic gun simulation (e.g., Raycasting).

Level 2

Features Implemented

- The player needs to navigate the level and collect four items located at random locations.
- The environment can be generated using either 3D meshes or 3D blocks.

- A First- or Third-person Controller for the player.
- Once the player has collected all the relevant objects, it should be moved to the third level.
- The level was created procedurally using meshes.

Level 3

Features Implemented

- A 3D solar system generated using an XML file.
- The player needs to navigate the level and collect four items located at random locations in space.
- Each of these items should be a satellite of one of the planets; in other words, it will need to rotate around one of the planets at a speed of your choice.
- The XML file should include the name of the planets, along with their size, color, distance to the sun, and rotational velocity, all relative to the sun.
- Each planet should rotate according to the data included in the XML file and include a label with their name.
- A trail should be generated for each planet.
- A spaceship based on Unity's Vehicles prefabs (e.g., Aicraft-Jet).
- Once the player has collected all the relevant objects, it should be move to the fourth level.

Level 4

Features Implemented

- The level consists of a multi-player tank simulation.
- The level includes a 3D environment with walls in places.
- Each player is represented by a tank.
- Collisions should be detected between the tank and the walls.
- Each tank can be made of basic shapes (e.g., cylinders, and cubes)
- Each tank can rotate and move forward and back.
- Each tank can fire missiles.
- Each player will be instantiated using a Round-Robin method.
- Each tank is destroyed after being hit 10 times.
- Once a tank has been destroyed it will be re-instantiated at a location determined by a Round-Robin method.
- Each player should include a camera that follows its movement (i.e., top-view).
- Each player should avail of a mini-map.
- Once the player has collected all the relevant objects, the game ends and the player has won.