Project Runner
Thomas Ankerholz, Sarah Wadley,
Laura Sarmiento

App Description

Jump Runner is a simple infinite game to pass time with. Help Boxy keep running and jumping between each platform to get coins and get the best high score.
Every time you get a new high score, you can register your name or nickname and check your progress in the High Score menu
Our goal is to make a game that is easy to play and get addicted to. Our target audience are people of all ages that can enjoy a fun and simple game like Flappy Bird or Dinosaur Run
The project is hosted at the following GitHub repository:
https://github.com/lifeunsubscribe/Project-Jump

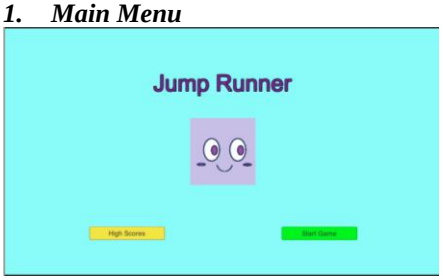
App Features

1. Gameplay: Play Jump Runner touching the screen to jump
2. High Score Database: Shows the first 10 High scores in the game
3. New High Score: When a new high score is reach, you can submit to the High Score Database your username and new high score.
4. Main menu: Provides the options to enter the High score list and start the game.

UI Design

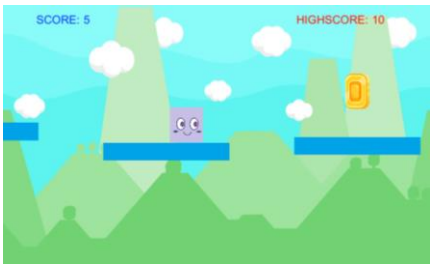
Jump Runner contains 2 main activity UIs/Scenes.

1. Main Menu

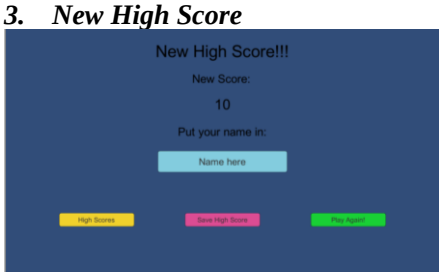


We use two buttons, one to start the game and to see the high scores

2. Game



3. New High Score



It shows the new high score and the option to send your name and send it to the high score board. We use four Text Views, one EditText, and three buttons, to play again, save the high score in the database, and to see the high scores.

Code Design

The game contains the following C# classes/files:

a) player.cs: Controls player commands and movements.
b) PlayerScore.cs: Controls the high scores, saving and retrieving of data for Firebase.
c) Background.cs: Brief description of the class.
d) bgScroll.cs: Controls the background scroll movement.
e) UserInfo.cs: Has the User object details for the database.
f) scoreButton.cs, startButton.cs, and titleButton.cs: Each controls a specific button to go to another scene.
g) gameController.cs: Score counter.

APIs

In this project we used two APIs:

- **Firebase real-time database:** This API is a cloud-hosted database by Google. The high score data of the game is saved in real-time and stored as JSON. The link to the Firebase Database is here:

https://jumprunnercop4656.firebaseio.com/

- **RestClient**: The REST API is used to make the implementation of the Firebase SDK feature easily. It was installed from the Unity Asset Store.

Challenges

We have many challenges along the way. We could not use many tools that we learned in class to help is along the way because we were working on Unity instead of Android Studio. We had to learn how to use Unity and C# along the way. Also, the implementation of the Firebase Database was also different of what we saw in Android Studio.
Outside of the code, we had experience before working in group project but much difficult to work in group projects in a remote/online class because of different schedules and availability.

Deviation from proposal/Incomplete Features

We deviated from the proposal for the Profile Login, Profile Create, Switch Profile, and Logout mention in the proposal and instead, we just created a database where you can submit your name with your high score just like old games that you can find in an arcade or Touhou. We did this change because it was better feature for this kind of game, and it was easier to implement. We would use Firebase Auth to implement another way to access an account or profile, but we didn't have the time or knowledge to do so.

The incomplete feature that we had was the High Score Database. The data gets into the Firebase Database but to retrieve it was basically impossible with the amount of time that we had left. We tried using REST and the Firebase SDK but the Unity crash or didn't respond at all. Also, because of this issue, the New High Score does not work correctly and just appears after getting an score equal or greater than 10.

Lessons Learned

In this project we learn how to work and create a game with Unity and coding with C#. We also experimented how to work with Firebase Database in another platform outside of Android Studio and install and use the free tools that the Unity Asset Store can bring.

We also learn how to create a Git repository, how to push into it and work inside the repository to see the changes that we did as a group.

How About This

For this project, we think the class can we more in depth about how Firebase Database worked. Unity and Android Studio are different but to retrieve information from Firebase Database was kind of rough. A video tutorial, just like the rest of the topics in the class, it would have been very useful. Also, we talked about Unity was the last topic of the class, although we thought that we wouldn't talk about it at all in class, it was kind of useful at last minute for minor problems in our project. If we had a class about Unity earlier in the semester it would have also been a really good tool. Just to finalize, the video tutorials for Unity can be more in depth if someone wants to see them to learn from themselves.

Team Contribution

- Sarah Wadley:
  - player.cs
  - Background.cs
  - bgScroll.cs
  - MainScene
- Thomas Ankerholz:
  - Proposal
  - startButton.cs
  - Music implementation
  - TitleScene
  - Debugging and code testing for Background.cs and player.cs
- Laura Sarmiento:
  - Implementation of Firebase and REST API
  - PlayerScore.cs
  - gameController.cs
  - UserInfo.cs
  - scoreButton.cs and titleButton.cs
  - HighScoreScene and NewScoreScene
  - Debugging and code testing for all C# classes.
  - Project Report