Geo Run

CPSC 254-01

California State University, Fullerton

Group GPEJ
Eric Chu, Kiyuen88ec@csu.fullerton.edu
Jake Wong, jakedwong@csu.fullerton.edu

Table of Contents

Group Information

Our group name is GPEJ. The members are Eric Chu (kiyuen88ec@csu.fullerotn.edu) and Jake Wong (jakedwong@csu.fullerton.edu).

Link to repository here: https://github.com/ericchu1329/Geo-Run
Play the game here: https://ericchu1329.github.io/Geo-Run-Host/

Initial Project Description

The game is an endless runner. The gameplay will be similar to Subway Surfer and Temple Run. The general look of the game will be similar to Geometry Dash. The game will be created in Unity 2020.3.27f1 and will be played via web browser on GitHub (or via desktop executable if GitHub does not work).

Tentative Plan:

Week	Goal
1-2	3d Modeling and asset Creation
3-6	Coding the functional requirements
7-8	Bug testing
9-end	Finalizing

Initial Functional Requirements:

ID	Requirement
REQ-01	The system shall allow a user to be a player.
REQ-02	The system shall provide an endless layout for a player to run in.
REQ-03	The system shall have collectible items within the layout to raise the player’s score
REQ-04	The system shall give the player three lives.
REQ-05	The system shall take away one life from the player if they run into an obstacle.
REQ-06	The system shall make the player invincible for three seconds after they have lost a life

REQ-07	The system shall end the game if the player runs out of lives.

Description of Submitted Project

Geo Run is an endless runner type of game. The gameplay is very similar to Subway Surfer. The general aesthetic of this game is a mixture of Geometry Dash and Subway Surfer. The goal of the game is to survive as long as possible while dodging various obstacles on the runway. Collecting crystals on the way will boost your score.

Progress Log

Week	Goal
1-2	White boxing end layout, initial character design
3-5	Coding the functional requirements
6-7	Finalizing 3D models and character design
8-9	Code refactoring
10-end	Testing and Finishing touches

Our progress on this project for the most part followed the plan from the initial project pitch. We created our initial models for the character and platforms and just played with those to see how they felt. Characters were good, but the platforms needed a redesign. That's why we ended up finalizing our 3D models near week 8-10. We also did some code refactoring since we wanted to try a different implementation. Other than things we did listed above, we mostly followed the initial plan.

These are the following functional requirements for Geo Run:

Game Basics

ID	Requirement
REQ-01	The system shall allow a user to be a player.
REQ-02	The system shall allow a user to pause the game.
REQ-03	The system shall allow a user to change characters.
REQ-04	The system shall allow a user to change the volume of the game.

REQ-05	The system shall provide an endless layout for a player to move in.

Endless Layout

REQ-06	The system shall have a collectible item within the layout that raises the player's score.
REQ-07	The system shall have obstacles within the layout.

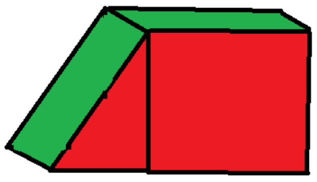
Player

REQ-08	The system shall give the player one life.
REQ-09	The system shall allow a player to move horizontally.
REQ-10	The system shall allow a player to jump vertically.
REQ-11	The system shall allow a player to fall quicker.

Collision Detection

REQ-12	The system shall detect collisions between a player and **obstacle objects.**
REQ-13	The system shall detect collisions between a player and **platform objects.**
REQ-14	The system shall end the game if a player collides with **death areas.**

Obstacle Objects: Obstacles are game objects that can be traversed on only if a player traverses it the intended way. Examples of obstacles are ramps or tall boxes.
Platform Objects: Platform objects are game objects that a player can move on. This is essentially the default ground.
Death areas: Death areas are areas on an obstacle that if a player collides with, they will instantly die. This is to simulate a vehicle crash. For example, consider this ramp object. The area in green means that a player can safely collide with and move on. The area in red means that the player will instantly die if they collide with it.



The primary difference between the initial plan and our final product is that we did not include the multiple lives and invincibility mechanic. We felt as if that style of gameplay did not suit an endless runner game that we are making. Endless Runner games are typically movement-based games. The three lives and invincibility mechanic that we initially proposed is more suitable for fighting-based games. That is why we scrapped that idea and added requirements that were more meaningful to our game.

Coding Style

The coding style for this project follows the Google C++ style guide. The reason we chose this is because we are most familiar with C++ style.

Variables: all lowercase with underscore for spaces
Constants: same as variables but prefixed with “k”
Functions: PascalCase
Classes: PascalCase
Enums: PascalCase

In hindsight, it would have been better to stick with a C# style guide because Unity’s primary scripting language is C#. However, most of the code was already written in C++ style, so it would have been impractical to switch styles.

Workload Distribution

Eric Chu

- 3D Modeling and texturing
- Scripting
- Sound
- UI Logic

Jake Wong

- Testing
- Level Layout
- UI Descriptions

Eric Chu was assigned more tasks as he had the most experience with Unity and 3D Modeling. Jake Wong was assigned tasks that involved level design, editing the UI, as well as testing. The goal of distributing the workload this way was to finish the project as fast and efficiently as possible.

Version History

There are three versions of the game. The main version is a WebGL version which is meant to be played on a web browser. The other two versions are desktop versions: Windows and Linux.

Testing

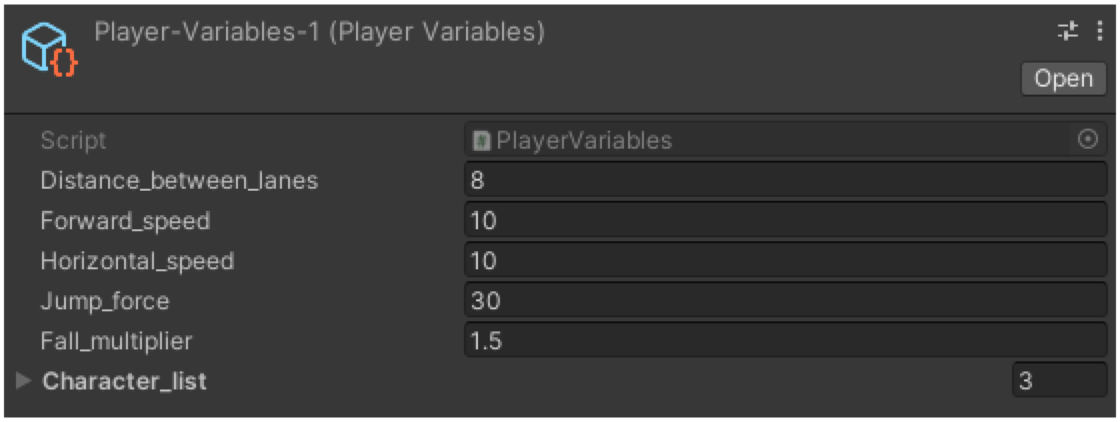
The main testing technique we used was quality assurance (QA testing) and unit testing. QA testing allowed us to refine the multiple values of the player movement variables as well as figure out the design of the UI. These included horizontal speed, initial forward speed, growth of forward speed over time, jump force, etc. Unit testing was another testing technique that we used.

We chose QA testing because it helped us converge toward the final look and design of our game.

We chose unit testing as a supplementary testing technique. However, this was particularly difficult to test, as unit testing in Unity requires a certain coding pattern. To explain, tests in Unity are split into Edit Mode tests and Play Mode tests. In those test suites, you cannot call a private function of an object to test it. Since most of the functions of each class were private, it was very difficult testing them. A majority of the unit tests were testing public fields and functions.

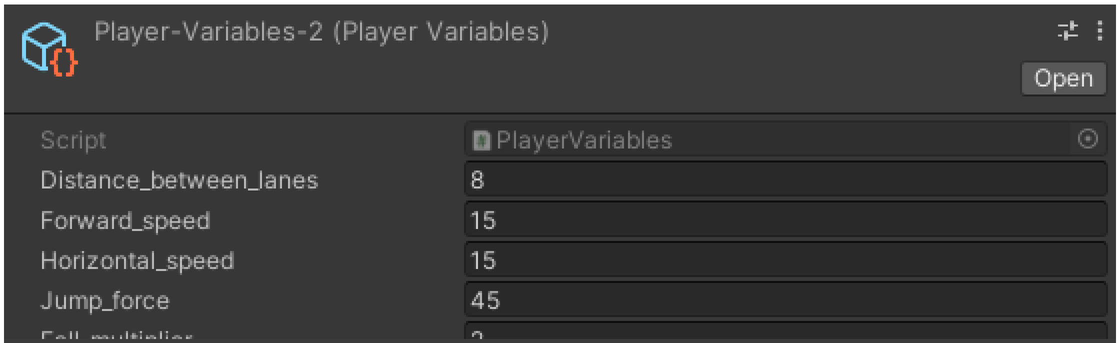
Results:

To figure out what values we liked for player move speed, jump power, and fall multiplier (gravity), we used the Scriptable Object pattern. This allows us to create a bunch of different player variable objects that have different values for player movement. We had five different player scriptable objects.

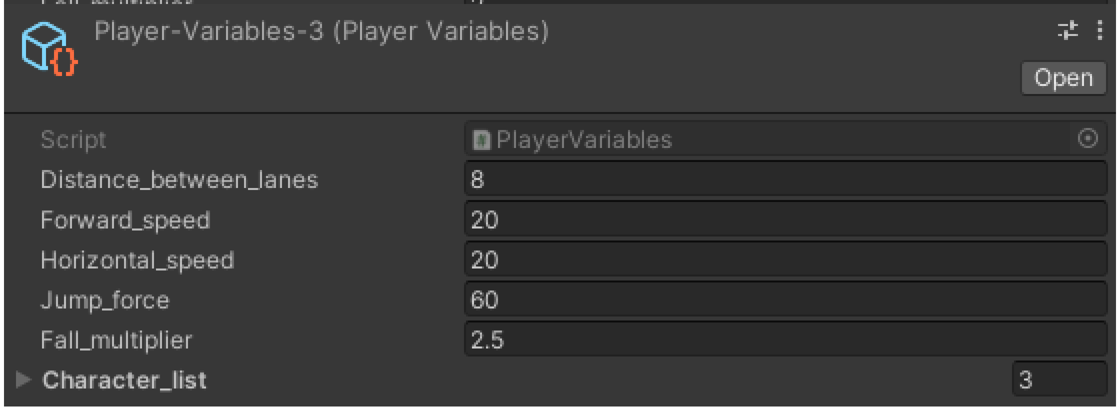


Player-Variables-1 (Player Variables)

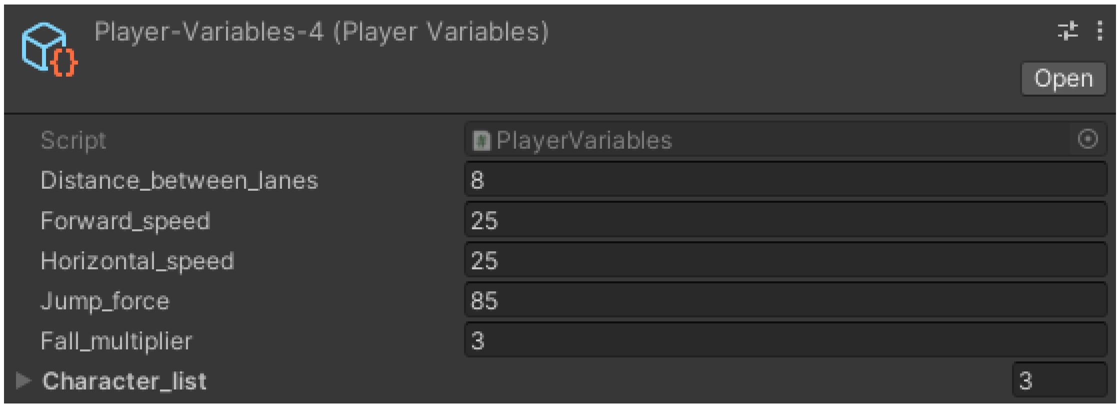
Script	PlayerVariables
Distance_between_lanes	8
Forward_speed	10
Horizontal_speed	10
Jump_force	30
Fall_multiplier	1.5
Character_list	3


Player-Variables-2 (Player Variables)

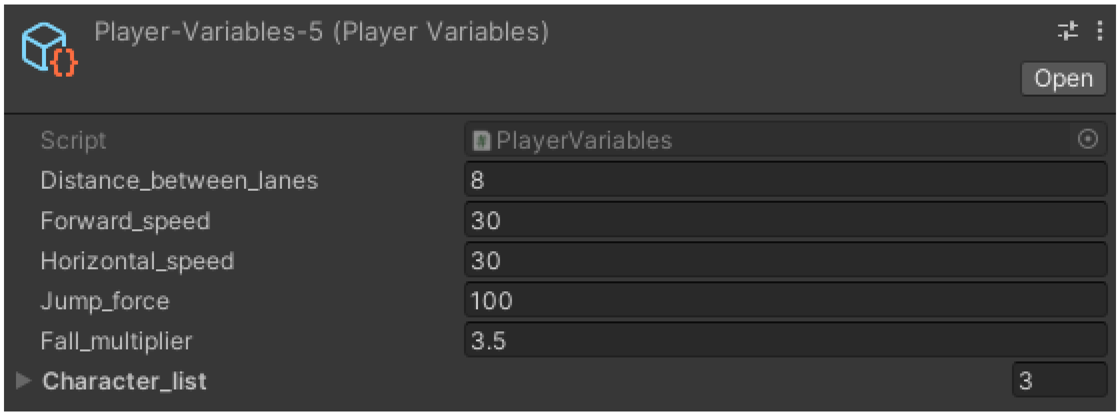
Script	PlayerVariables
Distance_between_lanes	8
Forward_speed	15
Horizontal_speed	15
Jump_force	45
Fall_multiplier	2


Player-Variables-3 (Player Variables)

Script	PlayerVariables
Distance_between_lanes	8
Forward_speed	20
Horizontal_speed	20
Jump_force	60
Fall_multiplier	2.5
Character_list	3

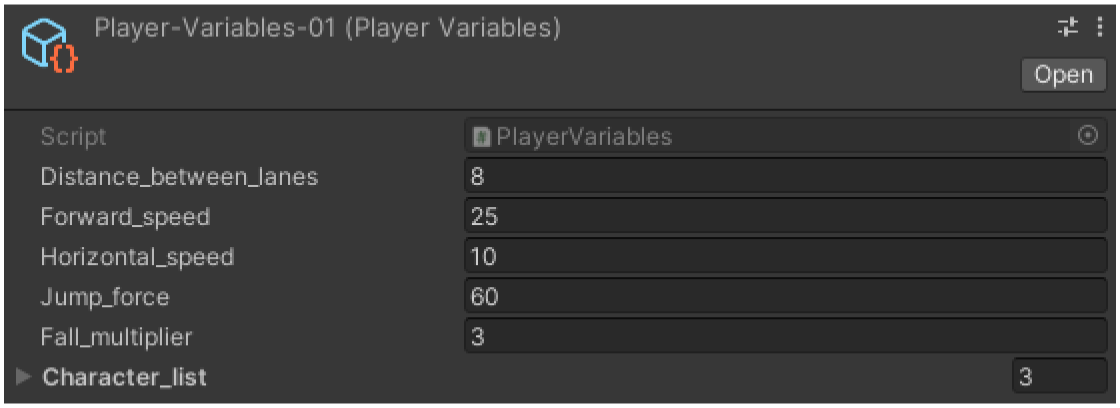

Player-Variables-4 (Player Variables)

Script	PlayerVariables
Distance_between_lanes	8
Forward_speed	25
Horizontal_speed	25
Jump_force	85
Fall_multiplier	3
Character_list	3

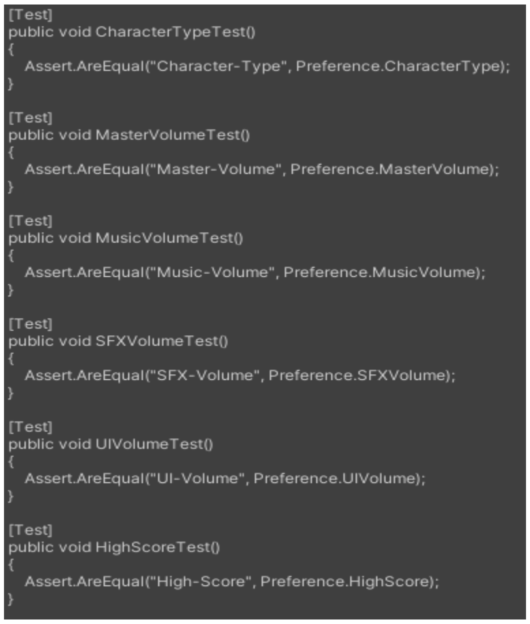

Player-Variables-5 (Player Variables)

Script	PlayerVariables
Distance_between_lanes	8
Forward_speed	30
Horizontal_speed	30
Jump_force	100
Fall_multiplier	3.5
Character_list	3

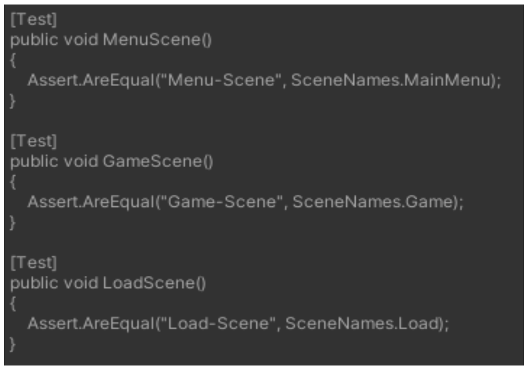
After playing the game with each of object, we chose these values for player movement



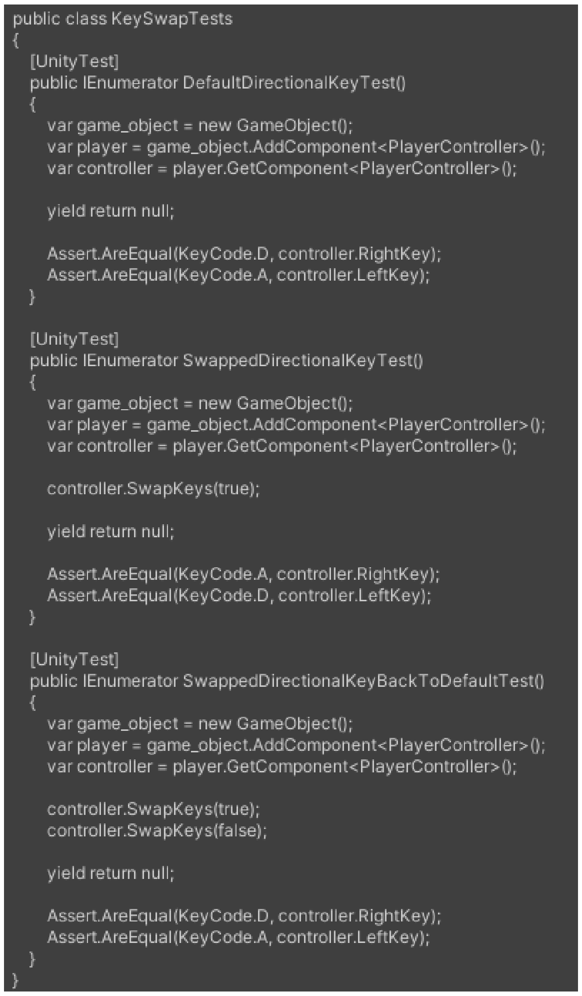
Script	PlayerVariables
Distance_between_lanes	8
Forward_speed	25
Horizontal_speed	10
Jump_force	60
Fall_multiplier	3
Character_list	3



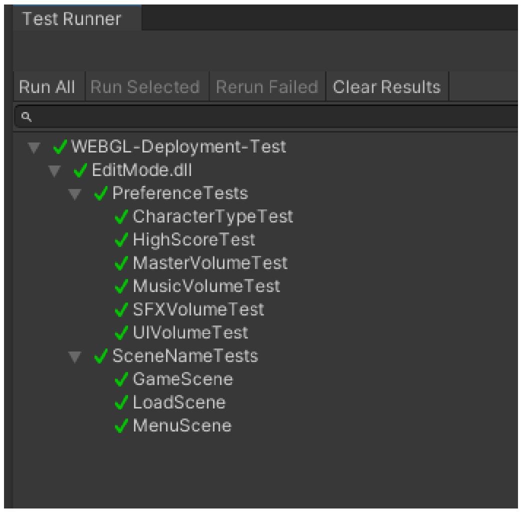
```
[Test]
public void CharacterTypeTest()
{
    Assert.AreEqual("Character-Type", Preference.CharacterType);
}

[Test]
public void MasterVolumeTest()
{
    Assert.AreEqual("Master-Volume", Preference.MasterVolume);
}

[Test]
public void MusicVolumeTest()
{
    Assert.AreEqual("Music-Volume", Preference.MusicVolume);
}

[Test]
public void SFXVolumeTest()
{
    Assert.AreEqual("SFX-Volume", Preference.SFXVolume);
}

[Test]
public void UIVolumeTest()
{
    Assert.AreEqual("UI-Volume", Preference.UIVolume);
}

[Test]
public void HighScoreTest()
{
    Assert.AreEqual("High-Score", Preference.HighScore);
}
```



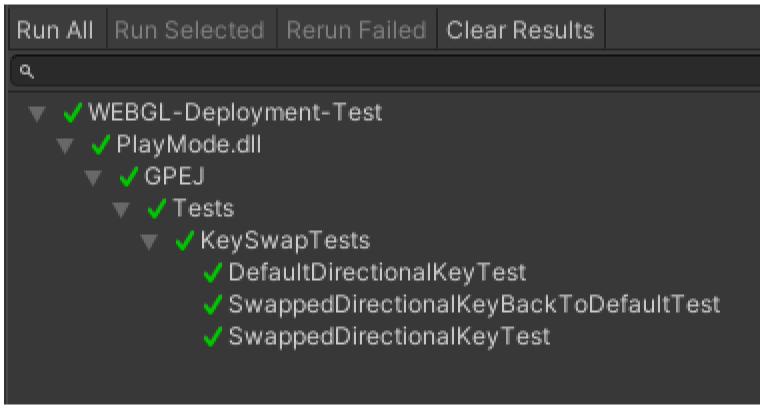
```
[Test]
public void MenuScene()
{
    Assert.AreEqual("Menu-Scene", SceneNames.MainMenu);
}

[Test]
public void GameScene()
{
    Assert.AreEqual("Game-Scene", SceneNames.Game);
}

[Test]
public void LoadScene()
{
    Assert.AreEqual("Load-Scene", SceneNames.Load);
}
```



```
public class KeySwapTests
{
    [UnityTest]
    public IEnumerator DefaultDirectionalKeyTest()
    {
        var game_object = new GameObject();
        var player = game_object.AddComponent<PlayerController>();
        var controller = player.GetComponent<PlayerController>();

        yield return null;

        Assert.AreEqual(KeyCode.D, controller.RightKey);
        Assert.AreEqual(KeyCode.A, controller.LeftKey);
    }

    [UnityTest]
    public IEnumerator SwappedDirectionalKeyTest()
    {
        var game_object = new GameObject();
        var player = game_object.AddComponent<PlayerController>();
        var controller = player.GetComponent<PlayerController>();

        controller.SwapKeys(true);

        yield return null;

        Assert.AreEqual(KeyCode.A, controller.RightKey);
        Assert.AreEqual(KeyCode.D, controller.LeftKey);
    }

    [UnityTest]
    public IEnumerator SwappedDirectionalKeyBackToDefaultTest()
    {
        var game_object = new GameObject();
        var player = game_object.AddComponent<PlayerController>();
        var controller = player.GetComponent<PlayerController>();

        controller.SwapKeys(true);
        controller.SwapKeys(false);

        yield return null;

        Assert.AreEqual(KeyCode.D, controller.RightKey);
        Assert.AreEqual(KeyCode.A, controller.LeftKey);
    }
}
```



```
▼ ✓ WEBGL-Deployment-Test
  ▼ ✓ EditMode.dll
    ▼ ✓ PreferenceTests
        ✓ CharacterTypeTest
        ✓ HighScoreTest
        ✓ MasterVolumeTest
        ✓ MusicVolumeTest
        ✓ SFXVolumeTest
        ✓ UIVolumeTest
    ▼ ✓ SceneNameTests
        ✓ GameScene
        ✓ LoadScene
        ✓ MenuScene
```



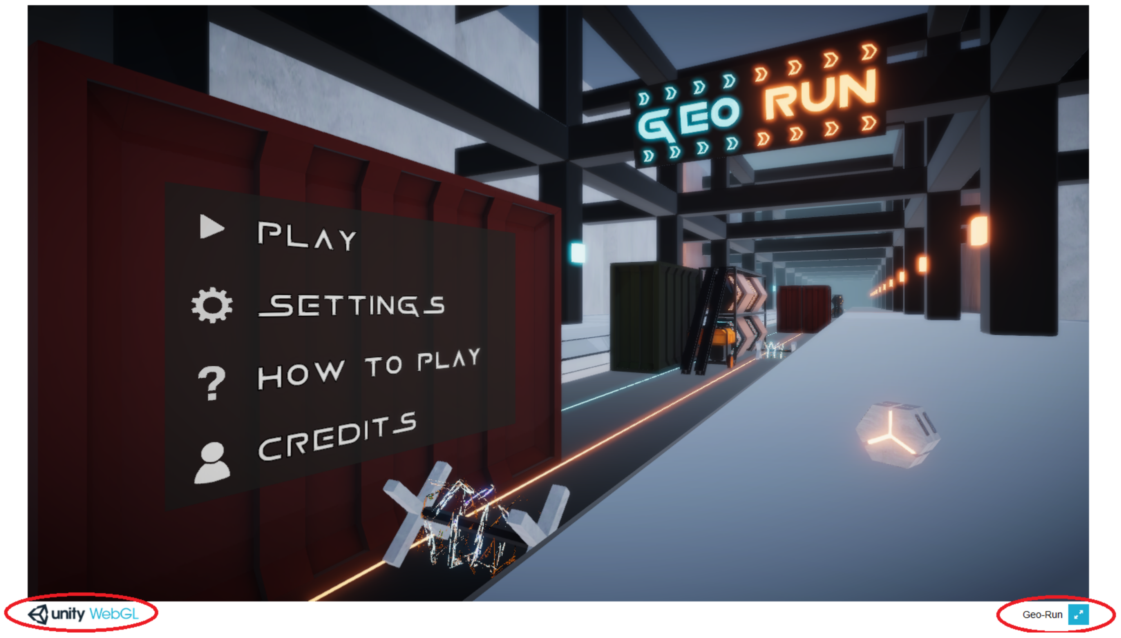
```
▼ ✓ WEBGL-Deployment-Test
  ▼ ✓ PlayMode.dll
    ▼ ✓ GPEJ
      ▼ ✓ Tests
        ▼ ✓ KeySwapTests
            ✓ DefaultDirectionalKeyTest
            ✓ SwappedDirectionalKeyBackToDefaultTest
            ✓ SwappedDirectionalKeyTest
```

Running the Project

Since this game is web-based, you only need to click on the following link to play the game. The game is hosted on GitHub pages. The link to the game is also located on the project repository.

Link to Game: https://ericchu1329.github.io/Geo-Run-Host/

If you do not see the unity.WebGL logo on the bottom left and the full screen button on the bottom right, please do the following.



1. Zoom out of the web page until you see those two items described above on the bottom. This can be done by pressing the keys “ctrl” and “-” together.
2. Then hit the full screen button.

Troubleshoot:

If the game does not run on your browser, there are two possible errors:

1. Your browser does not support WebGL 2.0.
2. Your browser does not have hardware acceleration enabled.

To check if your browser is compatible with WebGL 2.0, visit this site.

To enable hardware acceleration for your browser, please follow this guide.

If the game still cannot be run on your browser, please visit https://github.com/ericchu1329/Geo-Run/releases for the windows or linux desktop executable and download the .exe version.