Spring
2020

Implementing artificial intelligence mechanisms in mobile video gaming.

ITC 4918 – SOFTWARE DEVELOPMENT CAPSTONE PROJECT

PREPARED BY: DIMITRIOS MAVROFRYDIS

SUPERVISED BY: PROF. E. VAGIANOU

Table of Contents

1. Introduction

1.1. Aims and objectives

During the past few years there has been an unbelievable demand in the video game industry, especially when dealing with mobile video games and applications. With the recent release of the two major game subscription services offered by Apple and Google, the demand for new and innovative mobile video games with stunning graphics and interesting scenarios is higher than ever. However, the current mobile video game industry has not yet managed to successfully implement the notion of a mobile gaming software which adapts to the user's behavior. Therefore, the final product of this study provides a 3D infinite runner mobile video game experience, which implements an artificial intelligence (machine learning algorithm), capable of altering the difficulty of the level according to the gameplay style and the performance of the player. Additionally, the product is expected to make a great impact on the market due to the implemented smooth gameplay mechanics, eye-catching graphics, as well as the dynamic difficulty adjustment agent. Therefore, the combination of all of the above offer a unique mobile gaming experience different from all the other competitor games on the market. Furthermore, implementing a machine learning / neural network type of algorithm, while following all the major artificial intelligence principles, allows for an improved user and gameplay experience. Such a feature allows the dynamically generated level, the obstacles, the environment, the background elements as well as the enemies that appear during the gameplay sessions, to adapt to the user's needs, gameplay habits and behavior. The program's main functionality aims to fulfil the user's need for entertainment, anywhere and anytime by simply downloading the app on their smartphone for free. In order to provide a unique

experience that will be enjoyable to every type of user, an error free and flawless gameplay environment had to be carefully developed. In order to achieve that, the most efficient and optimized algorithms have been used, ranging from applying several optimization techniques to the level generation process, the character movement, and even to the User Interface elements and animations. By effectively optimizing processes such as: the automatic generation of the content, the recycling of the game objects, as well as the sound effects, music and the animations that are included, the game is able to consistently and flawlessly run at 60 frames per second on the majority of the smart devices on the market. Implementing an artificial intelligence agent which handles a large amount of input variables every second, requires many different resources to correctly operate. Nevertheless, the final product provides a smooth gameplay experience without heating up the device, neither excessively draining the battery of the mobile device. Finally, two different online gaming services are integrated, which promote the competitive character of the game by allowing access to the online score leaderboards and achievement/challenges that are available for the game. The main goal of this software solution is the successful integration of an A.I agent, capable of identifying and studying the user's behavior in a specified timeframe, in order to generate the appropriate level content based on the player's performance.

1.2. Background research

In order to develop a technologically advanced, flawless and addicting mobile gaming experience which offers unlimited dynamically generated levels based on the user's behavior, the appropriate research had to be conducted. Therefore, extensive research was performed on many different technological fields, with the utmost important topic being

the implementation of Artificial Intelligence and machine learning algorithms into Unity's engine. Having acquired such important knowledge, has allowed for the conception and development of a PCG DLD mechanism which uses a neural network algorithm, acting as the main brain and functionality behind the artificial intelligence agent. The term PCG stands for procedural content generation, while DLD stands for dynamic level design. This means that the artificial intelligence agent, gathers information from many different inputs, and stores them into many different networks and data structures. The collected information is carefully analyzed, in order to develop a personalized difficulty profile for the user which is then applied during the design and the development of the level generation process. Additionally, such implementation allows for better CPU performances especially on mobile devices, due the use of same materials, shaders and graphics in order to develop a wide variety of unique and eye-catching levels. These mechanisms bring life and knowledge to the machine learning algorithms, in order to generate fun, interesting, stable and symmetric levels, that will trick the user into thinking that the level was actually designed by hand and not by a computer. Moreover, after further research and studying, it was proved that such mechanisms and techniques, can have an important connection with altering the difficulty of the level. In order to achieve that, a DDA (dynamic difficulty adjustment) algorithm is implemented, as to allow the content generator to develop worlds with infinite possibilities and designs, based on the user's behavior and past experience. Implementing such a sophisticated and advanced mechanism, requires a variety of resources to correctly operate, especially on mobile devices. Thus, after researching many articles on optimizations techniques, the basic and most important steps into optimizing the game, were already implemented since the first day of

development. This has allowed the algorithms to be developed while considering the optimization of the software as the number one priority. Developing a desktop video game allows the programmer to exploit a lot of CPU and GPU power, in terms of on-screen draw calls. However, designing and developing a stable mobile video game, requires no more than 100-200 draw calls per second. Such a task can seem impossible for some. Nevertheless, using visual design techniques and optimizations techniques that are discussed in many of the research papers on video game optimization out there, the game has achieved to reach a max of 30-160 draw calls per second. This allows even the lowest-end smartphones to run the game at 60fps without any visible stutter and jitter. Finally, the research process was also extended on the physics system and whether the built in or custom physics would be used for the specific type of game. After researching the respective literature, as well as trusted sources on the internet, a custom-made physics engine was built from scratch into the game, which has already proven to be more energy and CPU efficient than Unity's default physics system. Using custom-made physics, has allowed for a more stable and accurate touch control of the main character (more realistic and appropriate movements), as well as reduced number of unwanted built in methods and calls during the gameplay. To conclude, researching on the most important and crucial topics for designing and developing the mobile video game, has already proven to be useful by simplifying the planning, as well as the development process.

1.3. Development Resources (tools used and reasoning)

During the development of this project, many different programs where used, each one of which served for a different functionality towards the completion of the videogame. First and foremost, the most important software package that was used, is the Unity video game

engine 3D (2019 version), which allowed for the creation of the 3D environment, while also connecting the written code and functions with the generated content. Additionally, Microsoft's Visual Studio was used as the main coding editor, which is completely compatible with Unity 3D. Moreover, in order to create most of the 3D graphics, as well as polish some of the already existing visual assets that were imported from previous projects and work, the Blender 3D and Adobe's Illustrator / Photoshop, were primarily used on a daily basis. Moving on, all of the code that is related with the development of the video game inside of the Visual-Studio editor was written in C# language, which is also compatible with Unity's engine. However, in order to correctly compile and install the videogame in the IOS and Android platforms, the necessary code was written in Swift and Java respectively. In detail, Apple's X-Code program and Google's Android Studio editors, were used in order to provide the executables for each one of the mobile platforms. Finally, some of the functionality inside the videogame was elevated by some third-party apps, which include: The High Mountains Vibration API system, and Gley's Apple and Google SDK plugin. The former allowed for the quick and easy integration of many different vibration feedback mechanisms inside the product, which allowed for more time to work on more crucial features, due to the strict time constraints. The latter was used for the quick implementation of Apple's and Google's online service SDK into the final product.

2. Systems Analysis

2.1. Product Perspective

This is the finalized version of a commercial single-user standalone video game, which is programmed to launch on both major mobile platforms (IOS and ANDROID) in the near future. This software solution covers the user's need for entertainment purposes. In order to create a unique and exciting experience for the user, a procedural content generation manager was implemented, which creates an infinite number of completely different non-ending levels for the user to enjoy. However, this videogame completely differs from the regular infinite-runner type of video games, by integrating a unique artificial intelligence agent who is capable of identifying, storing and finally studying the user's behavior. This way, the artificial intelligence agent, is able to alter the difficulty of the generated content, in order to make the level easier or harder according to the user's needs. As a result, the longevity as well as the replay ability of the game are enhanced, by providing a different experience every time the user plays the game.

Direct Market: The final product's main market is dedicated to users from all around the world who have access to a smart device, that is compatible with the software requirements. Additionally, most users are expected to have some kind of experience with similar types of video games. However, any user will be able to instantly adapt to the gameplay style and learn how to play the game, because of all the indicators and instructions that appear on the screen during the first gameplay sessions. In detail, the main target group of this video game, includes users who engage in downloading similar endless runner video games from the Google play or the App store.

Indirect Market: Third party corporations and gaming companies may be interested in the functionality of the adapting algorithm, which develops and constructs the gameplay environment based on the user's previous actions and recognized patterns. Moreover, another opportunity may arise in the case where the game is qualified to be submitted into Apple's Arcade or Google Play's gaming service.

2.2. Product Features (a high-level summary of the functions/processes)

- Online Game Services (Leaderboards, Achievements, Login, Registration)
- PCG: Procedural Content Generator (Dynamic Level Generation)
- Game Manager
- Player Manager
- Artificial intelligence Agent / Dynamic Difficulty Adjustment (DDA)

2.3. User Classes and Characteristics

Registered users

A. Needs: Login to score leaderboard database, submit highest score, view and submit achievement progress, view database statistics and information on the available scores. The most important trait for the registered user, is to be able to play the game, in which the artificial intelligence agent alters the difficulty accordingly.

B. Social characteristics:

   ✓ Age: 4+

   ✓ Gender: Female, Male, Other

- ✓ Knowledge background: Basic knowledge on how to download applications and games from the app store. Knowledge on using the Google Play / Game center online services. Additional but optional knowledge includes previous experience on playing infinite runner games.

Unregistered users:

A. Needs: Register to online services in order to access the score leaderboards and the achievements. However, the most important trait for the Unregistered user, is to be able to play the game, in which the artificial intelligence agent alters the difficulty accordingly.

B. Social characteristics:

- ✓ Age: 4+
- ✓ Gender: Female, Male, Other
- ✓ Knowledge background: Basic knowledge on how to download applications and games from the app store. Moreover, some basic but optional knowledge includes registering and creating a profile on the google play or apple's game center services. Additional but also optional knowledge includes previous experience on playing infinite runner games.

Administrator

A. Needs: Login to administrative services, database content management (Edit leaderboards, ban players, delete hackers etc.), submit updates and bug fixes during

the post-go-live period. Have access to the respective google developer and apple developer account pages on the internet in order to fulfill all of the above.

B. Social characteristics:

- ✓ Age: 18 +
- ✓ Gender: Female, Male, Other
- ✓ Knowledge: Coding skills, Internet Programming skills, general knowledge on database data manipulation, an apple developer account and a google developer account, in order to be able to publish the game, including future updates. Finally, other skills and general knowledge are required for using Apple's and Google's developer pages, where he/she can alter settings and preferences regarding the state of the videogame on the store.

2.4. Operating Environment

The operating environments that the finalized product supports, include Apple's IOS and Google's Android mobile platforms. Additionally, the final product supports a wide variety of smartphone and tablet devices. In detail, any IOS device running a version of 10.0 or higher, and any android device running android KitKat 4.4 or higher, is compatible with the software in discussion.

2.5. Design and Implementation Constraints

When designing a mobile video game, a lot of constraints may slow down or even terminate the development process. Developing and optimizing such a powerful 3D mobile video game for a wide variety of screen sizes and different processors, turned out to be very

tricky and hard to handle. However, this problem became even more apparent while testing the game on devices with a small screen size. The main issue was that, screen sizes with a size of 4 inches or smaller, would not allow for all the different GUI icons and menus to physically appear on the screen. Not only that, but the user would not be able to comfortably control the character that would appear on the screen. However, after many trial and error attempts to design the “optimal” user interface layout, a simple yet practical layout was settled, which involved the integration of simple to understand icons with as little text as possible. This way, by using indicators for each one of the functionalities, the user is able to easily understand the function of each one of the available UI buttons without any confusion. Another design and implementation constraint that surfaced during the development period, was related to delivering the finalized video game on lower end devices. Having to use the same resources with much less horsepower available, would not allow these devices to perform and deliver the same quality of gameplay that a newer high-end device would. Therefore, many different optimization techniques and coding schemes were used, which make the game playable on almost every available device out there. All this, while achieving a consistent 60 frames per second mark rate, which allows for smooth and lag-free movements of the player and the elements on the map. Moving on, as mentioned earlier in this study, Google’s and Apple’s online services were implemented in the finalized product, as part of the core game services feature. This ensures complete and error free compatibility with Apple’s and Google’s operating systems. However, this can become a great issue, in the case where the services are not accessible due to an internal error in any of the two companies’ servers. As a result, the users will not be able to access neither the leaderboards nor the achievements. Nevertheless, the

game was designed in such a way as to ensure that in such a case, the players statistics, achievements, challenges progression, as well as their best score, will be submitted once a valid connection is established.

2.6. Assumptions and Dependencies

This study does not have any known dependencies or security requirements.

2.7. Interface Requirements (To be sectioned if necessary. Categories: User, Hardware, Software, Communication)

The final product required many different graphic assets and elements, as well as software interfaces. In detail the required interface requirements include the 3D assets and the 2D graphics that were used for the development of the User Interface. Developing an immersive gaming experience, requires appropriate and impressive interface design elements. In order to provide such an experience to every user, a variety of 3D graphics and UI assets were implemented into the final product, some of which were downloaded from the Unity Asset Store, or even imported from other video game projects that I have previously developed. The required graphic elements used in the product, are categorized in the following list:

- **User input**: buttons, checkboxes (custom-made or 3rd party assets).
- **Gameplay Assets**: background themes and destinations, 3D character models and accessories (custom-made or 3rd party assets).
- **UI Navigation Elements:** sliders, 2D custom made graphics, icons, round buttons, back buttons (Both custom-made, but also purchased from the Asset Store).

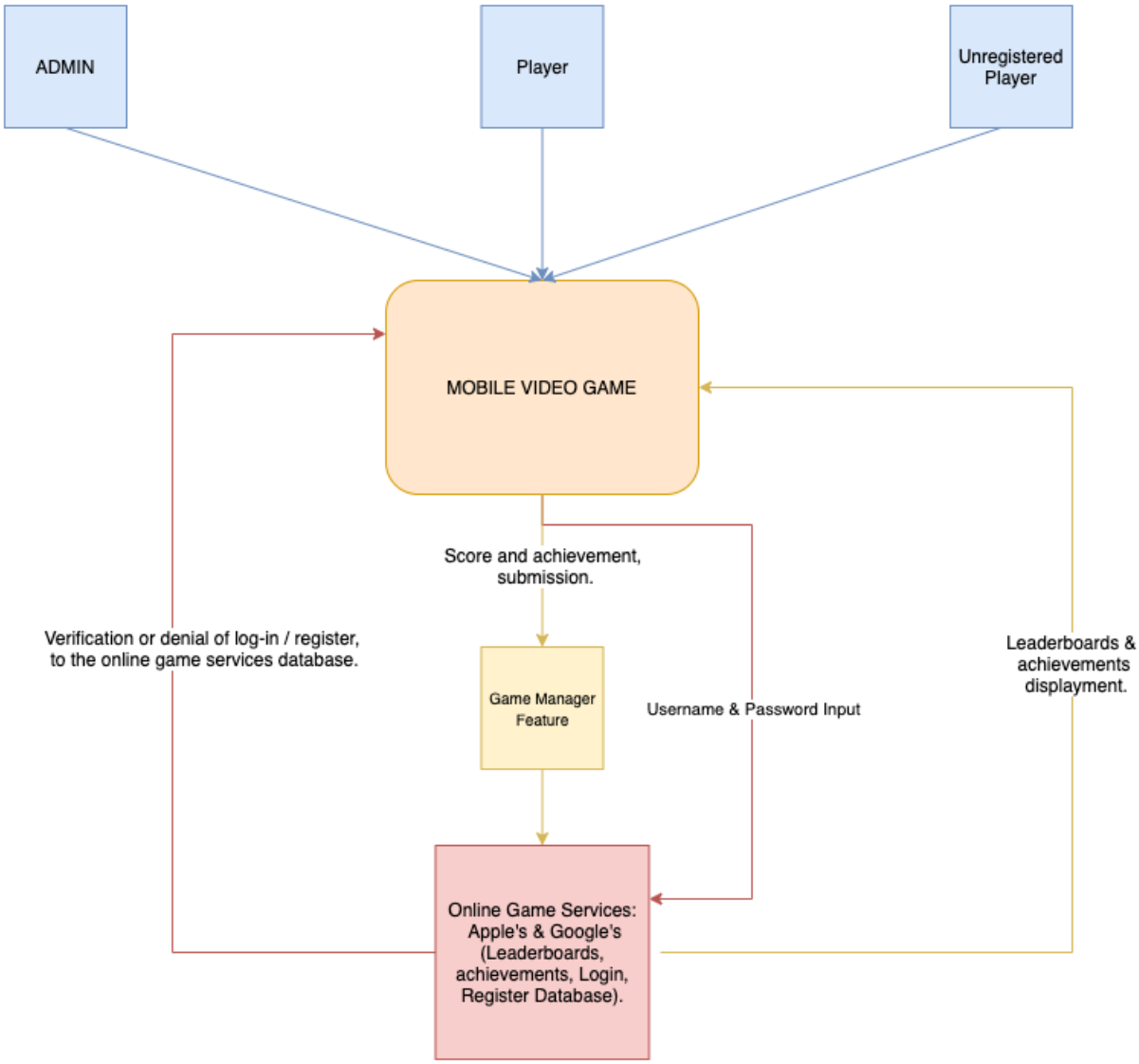
- **Containers and UI Boxes**: Rounded box Assets, GUI Container Elements and Icons, Pop Up Boxes, Hint boxes (Both custom-made and unity's defaults).
- **General UI elements**: Company Icon, achievements-leaderboards and challenges custom made icons representing each of the three different requirements, login screen logo, splash screen company logo (custom-made).
- **3D assets and graphics**: These elements include: all the shaders, textures and meshes that were used for the character, the obstacles, the decorations, the background elements as well as the sea surrounding the island.

Despite the final product requiring a wide variety of 3D graphics and assets, no additional requirements were crucial regarding the Hardware and Communication interfaces.

3. Systems Design

For the development of this product, many different components and subsystems were required. In detail, a lot of the subsystems exchange crucial information and data, in order to correctly operate and provide a flawless gaming experience to the user. Therefore, the final product of this study is composed of five major features, which allow it to correctly operate and provide the user with a unique and entertaining 3D gaming experience.

Architectural context diagram (ACD):



3.1. System Features

3.1.1. Online Game Services (Leaderboards, Achievements, Sign-in, Register):

The online game services feature allows the player to easily connect with the online gaming community, by tracking their performance and unlocking specific rewards. Unregistered players will be automatically prompted to create a free account at the start of the game.

Additionally, the player can access the registration and login service though the dedicated buttons, which are available on the main menu. Users who have signed in or registered their account with the integrated google play or game center service, will be able to easily access the worldwide score leaderboards, as well as the achievement/challenges list. Moreover, the signed in users will be able to track their score performance, as well as their current achievement progress. During the gameplay sessions, certain milestones are set, in order for the player's process to be tracked and immediately submitted to the respective online services. The player's highest score represents the highest distance traveled in a single run. However, in order to comply with the artificial intelligence agent which alters the difficulty and the speed of the player, the highest score is submitted according to the run time of the player's current level. Additionally, the different achievements are submitted as Boolean values (completed or not) and can range from completing different challenges such as: collecting 1000 coins, to getting a score higher than 10.000 and many more.

Input:

The input for most of the sub-features of the online game services are integer values, which represent the progress of the player (best score). T available achievements and challenges can be submitted either as integer or Boolean values, in order to indicate whether the player has completed them. All of the values are linked to the user's unique ID and name (PK), using Apple's and Google's databases.

Output:

All the submitted values, are then displayed on two different pop-up screens provided by Apple and Google. These custom User Interfaces are provided by the Apple and Google SDK.

These pop-up windows can be easily accessed by the player with a single press on the dedicated buttons, which can be found on the main menu of the game.

3.1.2. PCG: Procedural Content Generator (Dynamic Level Generation):

In order to provide an immersive and replay-able gaming experience to the users, a fully functional, error free gameplay feature had to be implemented. In order to achieve that, the video game inherits some of the most important and efficient methods for dynamically generating the required content during the gameplay sessions. This means that most of the features that are integrated into the final product, are complementary to this feature. Therefore, a lot of the preplanning and early development stages, were focused on designing and implementing an efficient, but most importantly stable procedural content manager generator. By implementing a PCG, the video game would be able to generate an infinite number of possible obstacle and background element combinations, while carefully applying the appropriate rules in order to avoid potential bugs and errors. Moreover, this is the best possible option for a content generation manager, as the content is generated as the player is progressing through the level. Setting a specific distance from the player at which the content should be generated, enabled for the easy integration of the artificial intelligence agent during the development proccess. Another positive outcome of implementing the procedural content generator, is the use of Object Pooling. This functionality allows the procedural content generator to activate and deactivate elements by recycling them when they are not needed. Consequently, there is no need to instantiate and destroy objects during the content generation, as it is one of the most expensive processes in video game development.

Input:

The input of the procedural content generator comes from two main sources. The first input and the most crucial one comes from the Artificial Intelligence Agent, who is responsible for providing the PCG with the most important information regarding the state of the game. The input can vary between integer values regarding the number of elements to spawn at a specific point, to the frequency of spawning such elements, but even to float values indicating the size and the exact position of an object. All of the input data that the PCG receives from the A.I agent, can be found in the ERD diagram. Some other inputs that the PCG receives from the DDA agent include: the frequency for spawning pick-up items, power-ups, decorations, background elements and finally the frequency for spawning one, two or even three different obstacles. The second input comes from the wide variety of available object pooling lists, which contain the generated objects to be used and recycled by the manager. This feature accepts an object (obstacle object, background object, decoration object), based on the input of the DDA agent.

Output:

The output is primarily based on the DDA agent's input. The final result of the PCG is the complete generation and activation of a valid level which spawns on the screen and is consisted of many different obstacles, background elements, decorations, as well as pick-up items.

3.1.3. Game Manager:

This is another feature of great importance which controls most of the user interface, sounds and on-screen elements. This is the feature that connects all of the entities together by connecting all the UI components on the Canvas, every button and most of the animations in the game. Additionally, this feature includes the "GameManagerScript" which is a singleton

Object in control of the game's state. Every UI button's action is controlled by this feature, which includes providing the appropriate feedback (sound & vibrations) to the player when it is appropriate. Moreover, this feature contains the management of the background music and the sound effects that are included in this product. Most importantly, the Game Manager is responsible for tracking some of the user's progress through the level, by receiving input from many different Game Objects, including the distance that the character has traveled. Likewise, the Game Manager is mostly responsible for changing the game's state each time an input is received, which includes the press of a specific button. Lastly, the Game Manager is also closely related with the DDA agent, due to having the responsibility to track the currently set difficulty, in order to alter the state of specific UI elements, while also changing specific variables for the player character.

Input:

This feature accepts a wide variety of inputs, most of which are integer values and Boolean values. This way, when an input is inbound, the respective script can alter the game conditions accordingly. Other input elements include UI button actions, which can contain float values (for playing a sound effect at a specific volume), integer values, Boolean values (for activating and deactivating UI elements) and many more. Finally, this feature accepts input from all the different pick-up items, power-up items, as well as the player script, which can then use in order to increment/decrement specific values such as: the number of pick-ups collected, the user's high score and many more.

Output:

The output of this feature might as well vary because of the wide variety of inputs. In the case where a UI button action with a Boolean input is received, the output can range from activating an animation, to starting an enumerator coroutine, or even playing/stopping specific sounds effects. Another event that can occur due to receiving an input, is the constant update of specific UI elements including colors, text and images, according to the nature of the input. For example, the Game Manager can receive the distance that the player has traveled as a float value, convert it into an integer value, and finally update the text element which indicates the player's current score on the screen.

3.1.4. Player Manager:

The Player Manager feature is mostly related with controlling the main character. However, this feature exchanges input and output with almost every other game feature, due to the fact that this is the entity that the user interacts with the most. First and foremost, the Player Manager allows the user to manipulate the position of the character according to the swipe gestures that are recognized on the screen. Additionally, this feature sends inputs to the DDA agent, the Game Manager as well as the PCG manager. By doing that, a stable and consistent connection between all of the available components is achieved.

Input:

This is another feature which accepts one but very important input. This input is received by the smart device's screen, which is then recognized and handled by the "Input-Script", which is a sub-feature of the Player Manager. This way, every swipe gesture and touch gesture is firstly recognized, and then translated into Boolean values. These Boolean values indicate the direction of the swipe. For example: the "Swipe-Up" Boolean value will be true if

the final position of the finger on the screen, is above the center of the screen inside a specified offset radius. As a result, the Player-Manager receives the Boolean inputs that are being sent every few milliseconds, which are later translated into actual movements.

Output:

According to the direction of the input that was received in the few previous milliseconds, an appropriate physics force is generated, which is applied to the custom-made character controller. This results in the character smoothly teleporting into the direction that was specified by the user's touch input (swipe gesture).

3.1.5. Artificial intelligence Agent / Dynamic Difficulty Adjustment (DDA):

Finally, the most crucial and important feature is the Artificial Intelligence agent, whose main purpose is to dynamically alter the difficulty of the level, according to the player's actions and recent behavior. This is a very complex and significant neural network, which is attached to a singleton game object and is instantiated once the game starts for the very first time. In detail, this agent begins by reading the basic difficulty preset settings from a Json file, which then saves into the appropriate class, which can be accessible by all the other scripts in the scene. In addition, this initial setting is saved as the default difficulty preset, which is provided to the PCG manager every time the game restarts. This allows the game to reset the difficulty and allow the user to have a fresh start. Additionally, the agent also reads all the available difficulty presets from the Json file, which then saves into a list of classes, where each class holds the attributes of each one of the nine available difficulty presets. The format of these difficulty classes can be found in the ERD diagram. In detail, this algorithm receives a large amount of different inputs, which are then analyzed in order to provide an estimate for the

player's performance during a specified given time framework (in this case the setting for the time framework is set to ten seconds). This means that the DDA agent will have a ten second period in order to analyze the player's performance and generate the new and updated data regarding his/her performance. Therefore, after the cycle is complete, the player's estimated performance is generated, and the new data is updated throughout every feature. Furthermore, the appropriate calculations are performed in order to represent the player's performance as a percentage. In other words, this percentage represents the user's performance on a scale from 0 to 100, during the previous ten seconds. As a result, the agent can understand how much the player has scored on a scale from zero to a hundred. With zero being a really bad performance and a hundred being an amazing performance. This calculation is generated based on a wide variety of factors which include the following:

- Number of coins and power-ups that are presented
- Number of coins and power-ups that were collected
- Number of coins and power-ups that were missed
- Number of successful dodges (obstacles & enemies)
- Number of near hitting the obstacles
- Distance traveled and time alive

Finally, each one of these different factors are given a weight, which allows the agent to understand the importance of each one of these rules, while generating the percentage for the player's performance. In addition, this behavior is then compared with previously saved difficulty states and performances, in order to determine whether the agent should tweak the difficulty preset higher, lower or allow it to stay the same. The next step is for the agent to

update the information for each one of its available methods in order to allow for the PCG manager to retrieve the new difficulty preset for the next ten seconds of generating new content. However, in order to make the algorithm even more personalized to the player's needs, the necessary calculations are made in the case where the player manages to overpass the available difficulty presets. In this case, the agent will increment or decrement the values of the previously used difficulty preset in order to create a personalized difficulty profile according to the performance of the player. This new and personalized difficulty profile has different values than all the nine difficulty presets that already exist in the Json file.

Input:

The input of this agent includes integer values from a wide variety of Game Objects which include: the pick-ups (coins), power-ups, collectables, (Trigger values from the Obstacles), as well as input (objects/classes) from the Json file, in order to form the necessary difficulty presets and save them into the appropriate data types. Additionally, the DDA agent accepts similar values and statistics from the Game Manager feature, which is also responsible for keeping track of the player's behavior.

Output:

The result of receiving all the required input values, is the generation of the player's performance in a scale from 0 to 100, which indicates whether the user performed better or worse than in the previous iteration. This result is then used as an input for the PCG manager, in order to generate the appropriate content.

3.2. Use Cases

This is a behavioral graphical representation of the interactions between the available users (admin, player, unregistered player) and the functionalities, subsystems of the mobile video game.

3.2.1. Use Case 1: Database feature (Online Services: Leaderboards, Achievements, Challenges)

1) Primary Actor: Player (Signed-in User).
2) Goal in context: Initiate the online services and check his/her current position on the worldwide score leaderboard.

3) Preconditions: Being signed-in on Apple's game center or Google's play service. Have an active internet connection.
4) Trigger: The dedicated leaderboards button is pressed.
5) Scenario:

    • The player clicks on the dedicated leaderboard button.

    • Apple's or Google's pop-up window is revealed, showing all the available leaderboards for the current game.

    • The player chooses the only available leaderboards option (Score Leaderboard).

    • The player's best personal score is displayed on top of the leaderboard as well as his/her position (e.g. 25th).

    • The player scrolls through all the different players and checks the best score in the world.

    • He/she clicks the close button and continues playing the game.

Exceptions:

1 The player does not yet have any score available to be displayed.

• Priority: Essential for a mobile infinite runner video game.

• When available: on release date.

• Frequency of use: Multiple times a day, every day.

• Channel to actor: via Apple's or Google's integrated online services system.

• Secondary actors: Apple and Google database/servers.

• Channel to secondary actors: none.

• Open issues: No open issues.

3.2.2. Use Case 2: Artificial intelligence Agent / Dynamic Difficulty Adjustment (DDA)

1) Primary Actor: Any of the three users (Registered, not registered, administrator).
2) Goal in context: Progress through the level, according to the difficulty set by the DDA agent.
3) Preconditions: No necessary preconditions need to be met.
4) Trigger: Press the play button in order to initialize the start of the game.
5) Scenario:

    • The player clicks on the “tap to start” button.

    • The character starts moving.

    • The DDA agent is activated in order to allow the PCG manager to generate the level.

    • The player fails to keep up with the game.

    • The DDA agent is once again activated, and changes the difficulty, making it easier for the player to continue playing the game.

    • He/she effectively adapt to the new level conditions.

Exceptions: None

• Priority: Essential, main functionality behind generating the level.

• When available: on release date.

• Frequency of use: During every single run.

• Channel to actor: via the Player Manager, DDA agent and the PCG manager.

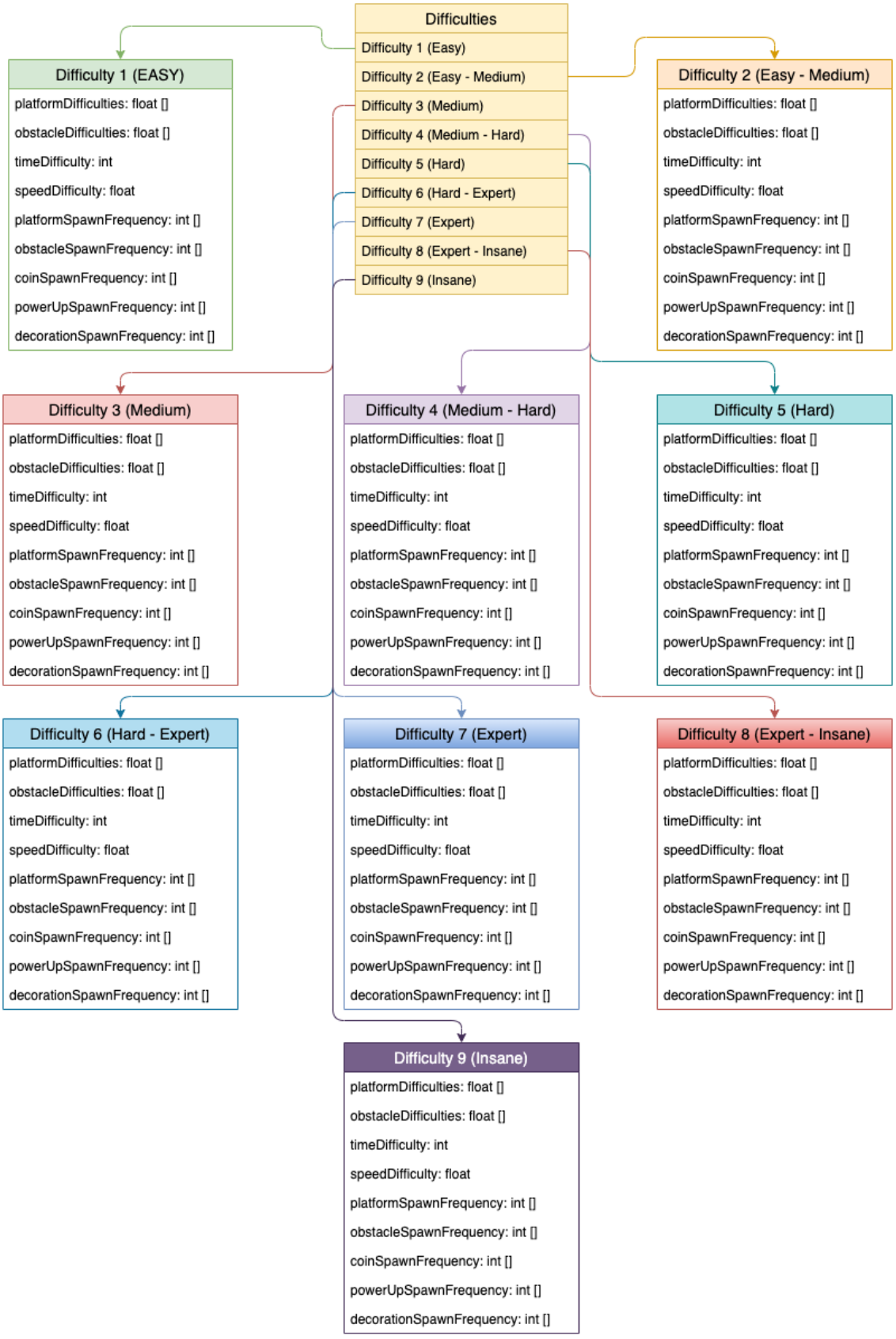
• Secondary actors: Game Manager.

• Channel to secondary actors: none.

• Open issues: Make the DDA agent more efficient by predicting future behaviors. Should there be an option to deactivate the DDA agent and provide basic difficulty settings?

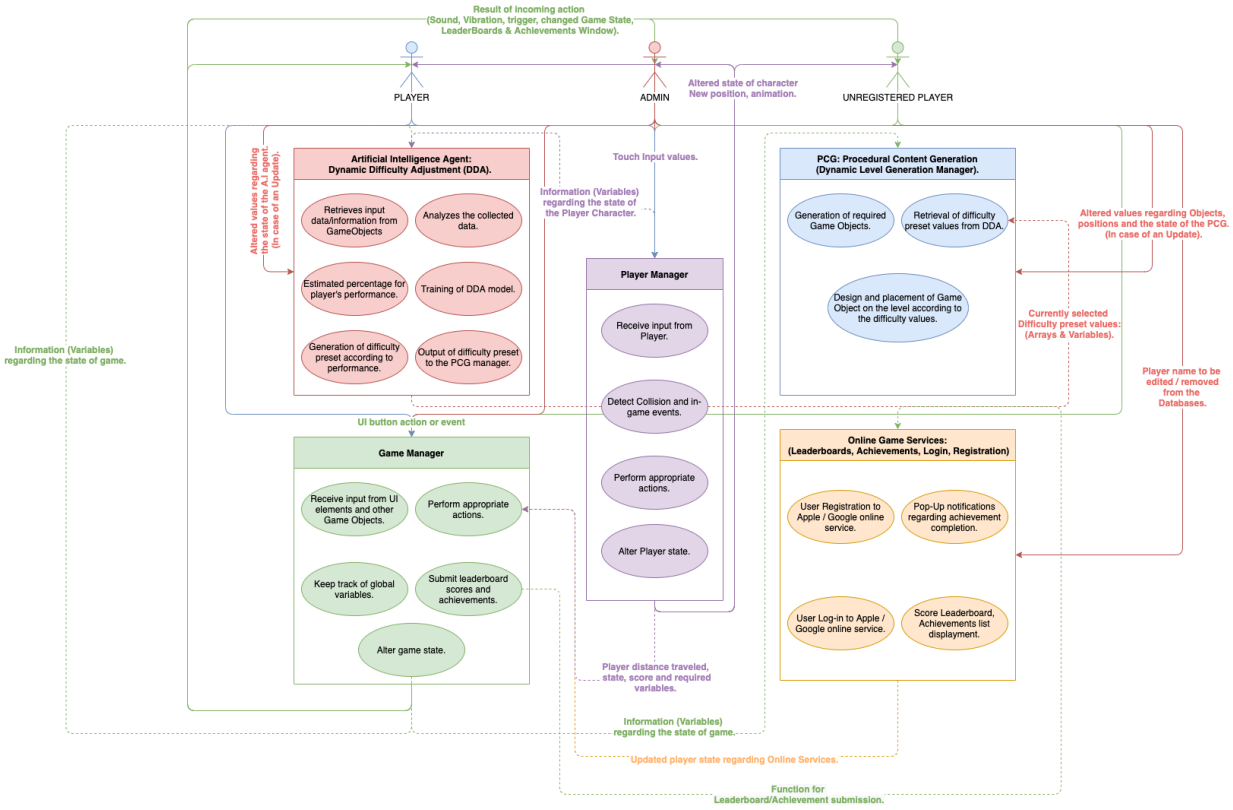
3.3. Interactivity and Associations

3.3.1. Dataset


Difficulties
Difficulty 1 (Easy)
Difficulty 2 (Easy - Medium)
Difficulty 3 (Medium)
Difficulty 4 (Medium - Hard)
Difficulty 5 (Hard)
Difficulty 6 (Hard - Expert)
Difficulty 7 (Expert)
Difficulty 8 (Expert - Insane)
Difficulty 9 (Insane)
Difficulty 1 (EASY)
platformDifficulties: float []
obstacleDifficulties: float []
timeDifficulty: int
speedDifficulty: float
platformSpawnFrequency: int []
obstacleSpawnFrequency: int []
coinSpawnFrequency: int []
powerUpSpawnFrequency: int []
decorationSpawnFrequency: int []
Difficulty 2 (Easy - Medium)
platformDifficulties: float []
obstacleDifficulties: float []
timeDifficulty: int
speedDifficulty: float
platformSpawnFrequency: int []
obstacleSpawnFrequency: int []
coinSpawnFrequency: int []
powerUpSpawnFrequency: int []
decorationSpawnFrequency: int []
Difficulty 3 (Medium)
platformDifficulties: float []
obstacleDifficulties: float []
timeDifficulty: int
speedDifficulty: float
platformSpawnFrequency: int []
obstacleSpawnFrequency: int []
coinSpawnFrequency: int []
powerUpSpawnFrequency: int []
decorationSpawnFrequency: int []
Difficulty 4 (Medium - Hard)
platformDifficulties: float []
obstacleDifficulties: float []
timeDifficulty: int
speedDifficulty: float
platformSpawnFrequency: int []
obstacleSpawnFrequency: int []
coinSpawnFrequency: int []
powerUpSpawnFrequency: int []
decorationSpawnFrequency: int []
Difficulty 5 (Hard)
platformDifficulties: float []
obstacleDifficulties: float []
timeDifficulty: int
speedDifficulty: float
platformSpawnFrequency: int []
obstacleSpawnFrequency: int []
coinSpawnFrequency: int []
powerUpSpawnFrequency: int []
decorationSpawnFrequency: int []
Difficulty 6 (Hard - Expert)
platformDifficulties: float []
obstacleDifficulties: float []
timeDifficulty: int
speedDifficulty: float
platformSpawnFrequency: int []
obstacleSpawnFrequency: int []
coinSpawnFrequency: int []
powerUpSpawnFrequency: int []
decorationSpawnFrequency: int []
Difficulty 7 (Expert)
platformDifficulties: float []
obstacleDifficulties: float []
timeDifficulty: int
speedDifficulty: float
platformSpawnFrequency: int []
obstacleSpawnFrequency: int []
coinSpawnFrequency: int []
powerUpSpawnFrequency: int []
decorationSpawnFrequency: int []
Difficulty 8 (Expert - Insane)
platformDifficulties: float []
obstacleDifficulties: float []
timeDifficulty: int
speedDifficulty: float
platformSpawnFrequency: int []
obstacleSpawnFrequency: int []
coinSpawnFrequency: int []
powerUpSpawnFrequency: int []
decorationSpawnFrequency: int []
Difficulty 9 (Insane)
platformDifficulties: float []
obstacleDifficulties: float []
timeDifficulty: int
speedDifficulty: float
platformSpawnFrequency: int []
obstacleSpawnFrequency: int []
coinSpawnFrequency: int []
powerUpSpawnFrequency: int []
decorationSpawnFrequency: int []

This is a representation of the values and the data structure of the included Json file, which provides the Artificial Intelligence agent with the basic difficulty settings. Additionally, this Json file includes a list of nine different difficulty presets, which the A.I agent can use in order to provide the PCG manager with the necessary data for the position, the size as well as the number of objects to be spawned on the level. Moreover, a new personalized difficulty preset may be created by the Artificial Intelligence agent by using one of these nine presets. For example, if the player outperforms the ninth and most difficult preset, a new personalized preset will be generated based on the final preset. Accordingly, if the user fails to keep up with the first and easiest difficulty preset, a new personalized and easier difficulty preset will be generated based on the values of the first preset. As a result, this Json file provides the most basic and significant data to the game, which allows the PCG as well as the DDA agent to perform as expected.

3.3.2. Dataflow



This is a visual representation model of the relationships and the flow of the data/information between the users of the video game and each one of the different components and functionalities. This model allows for a better understanding of how the data is actually exchanged between the various features and the users while they are playing the video game.

3.3.3. Class Model – Code Model:

If the link below is not working, try locating the folder named: CapstoneDoxygen and then clicking on the folder html. Finally locate the index.html file and open it with your preferred browser. In order for the link to work, the CapstoneDoxygen folder has to be in the same directory as the report.

Press here to see the code and Class Model webpage for the whole project.

4. System Documentation

When it comes to the user documentation and guidelines, the mobile video game follows the newest design gaming principles. In other words, this notion promotes the integration of the user guide and documentation inside the application, without the use of the old fashioned and boring user-guides in the form of text instructions. Instead, the PCG manager alters the level accordingly, in order to create an introductory/tutorial type of level followed by on screen instructions and animations, as to help the player learn the most basic movements and rules for playing the game. This tutorial/user documentation stage appears during the very first gameplay but can also be triggered by pressing on the dedicated "?" button that appears on the main screen of the game. This enables the user to quickly and easily understand the rules of how to actually play the game. In detail, this allows for a more pleasurable and enjoyable time of learning how to operate the specific software, without having to read any manuals or even having to research any instructions online. In addition, once the application is made available on the app store, a detailed guide on how to play the game will also be available in the description of the game's online listing. As a result, activating and playing the game is one of the easiest and well-designed functionalities of this software solution.

5. Conclusions

Overall, developing a standalone mobile video game in such a sort amount of time as a single man job, can seem impossible to some. However, the final product was successfully developed with no known game breaking issues and bugs. Additionally, most of the main functionalities and features that were proposed in the first version of the application, were

successfully implemented and even improved than the proposed version. It is important to acknowledge that the final product runs smoothly at 60 frames per second on more than ninety percent of the available devices on the market. Additionally, the final product was developed using the most significant coding techniques for making an efficient mobile video game, capable of running smoothly without consuming too many resources. Methods and procedures such as: using only a single scene without any reloads, implementing object pooling for the generation of the Game Objects, while using a procedural content generator in order to allow for an infinite number of generated levels and the recycling of the graphical assets, makes this video game really interesting and entertaining for the player. In order to top things off, the implemented artificial intelligence agent, acts as a dynamic difficulty adjustment mechanism, which in combination with the dynamic generation of the levels, allow for an unforgettable and personalized gaming experience. Generally, the final product exceeded the expectation of the original concept, by integrating beautiful 3D environments and graphics, amusing sound effects and background music, and eventually a refreshing endless runner gaming experience for the user. When it comes to the testing process, the most basic trial and error methodology was followed, which allowed each and every feature and functionality of the game, to be tested during its implementation. Some of the future plans and improvements for this version of the game include: the incorporation of an in-app purchasing system, an advertising system, and finally the publication of the videogame on the Android Google Play Store and the IOS App Store.

REFERENCES

The references that are cited below, include sources and articles/books which cover the following topics:

- Implementing Artificial Intelligence mechanisms in mobile video games.
- Mobile video game Optimization Techniques.
- Technologies Used.
- Physics engine and gameplay traits.

Aiolli, Fabio, and Claudio E. Palazzi. “Enhancing Artificial Intelligence on a Real Mobile Game.” International Journal of Computer Games Technology, Hindawi, 4 Nov. 2008, www.hindawi.com/journals/ijcgt/2009/456169/.

Chaudhari, Kunal. “AI for Unity Games: Emulate Real-World Senses in NPC Agent Behavior.” *Packt Hub*, 20 Sept. 2018, hub.packtpub.com/ai-unity-game-developers-emulate-real-world-senses/.

Equal, Iron. “Unity: Android Optimization Guide.” Medium, IronEqual, 24 Aug. 2017, medium.com/ironequal/android-optimization-with-unity-3504b34f00b0.

Equal, Iron. “Unity: CHARACTER CONTROLLER vs RIGIDBODY.” Medium, IronEqual, 4 Aug. 2017, medium.com/ironequal/unity-character-controller-vs-rigidbody-a1e243591483.

Gesota, Rudra. “How to Ace the #FINITE State Machine Model in #Unity AI Implementation.” "*TheAppGuruz*", 18 June 2016, www.theappguruz.com/blog/ai-implementation-using-finite-state-machine-model.

Giannakas, Filippos, et al. “A Critical Review of 13 Years of Mobile Game-Based Learning.” *Educational Technology Research and Development: A Bi-Monthly Publication of the Association for Educational Communications & Technology*, vol. 66, no. 2, 2018, pp. 341–384., doi:10.1007/s11423-017-9552-z.

Hodam Kim, et al. “Machine-Learning-Based Detection of Craving for Gaming Using Multimodal Physiological Signals: Validation of Test-Retest Reliability for Practical Use.” *Sensors*, vol. 19, no. 16, 2019. doi:10.3390/s19163475.

Interfaces, Holographic. “Learn How to Implement NEAT AI in Unity.” Medium, Medium, 29 Sept. 2017, medium.com/@HolographicInterfaces/learn-how-to-implement-neat-ai-in-unity-157168eeae7e.

Ke, Fengfeng. “Designing and Integrating Purposeful Learning in Game Play: A Systematic Review.” *Educational Technology Research and Development*, vol. 64, no. 2, 2016, pp. 219–244.

Loïc, Allart. “(Tutorial) How to Create an AI in Unity.” *Synnaxium* Studio, 19 Feb. 2019, www.synnaxium.com/en/2019/02/creating-an-a-i-with-unity/.

Nwankwo, Gilbert, et al. “Procedural Content Generation for Dynamic Level Design and Difficulty in a 2D Game Using UNITY.” *International Journal of Multimedia and Ubiquitous Engineering*, vol. 12, no. 9, 2017, pp. 41–52., doi:10.14257/ijmue.2017.12.9.04.

Nwankwo, Gilbert, et al. “Procedural Content Generation for Dynamic Level Design and Difficulty in a 2D Game Using UNITY.” *International Journal of Multimedia and Ubiquitous Engineering*, vol. 12, no. 9, 2017, pp. 41–52., doi:10.14257/ijmue.2017.12.9.04.

Risi, Sebastian, and Julian Togelius. “Procedural Content Generation: From Automatically Generating Game Levels to Increasing Generality in Machine Learning.” *GroundAI*, GroundAI, 29 Nov. 2019, www.groundai.com/project/procedural-content-generation-from-automatically-generating-game-levels-to-increasing-generality-in-machine-learning/1.

Seungback, Shin. “Game Level Generation Using Neural Networks.” *Gamasutra*, 27 Feb. 2018, www.gamasutra.com/blogs/SeungbackShin/20180227/315017/Game_Level_Generation_Using_Neural_Networks.php.

Shekar, Siddharth, and Wajahat Karim. Mastering Android Game Development with Unity. Packt Publishing, 2017. EBSCOhost, search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=1526149&scope=site.

Shekar, Siddharth, and Wajahat Karim. Mastering Android Game Development with Unity. Packt Publishing, 2017. EBSCOhost, search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=1526149&scope=site.

Stephenson, John. “6 Ways Machine Learning Will Be Used in Game Development.” *Logikk*, 29 Nov. 2018, www.logikk.com/articles/machine-learning-in-game-development/.

APPENDICES

A) Report on the effort reflecting your experience addressing the challenges you faced, deviations from the original proposal and reasoning, deviations from your suggested time schedule and reasoning.

This has been one of the most interesting and challenging projects since the beginning of my academic and work career. Having to develop a 3D mobile video game from scratch in such a sort amount of time seemed impossible at first. In order to make things more difficult, the Covid-19 virus and the household restrictions, really slowed down the development process. Despite having to go through this very difficult situation, the final product was successfully finished at the given timeframe. Nevertheless, some features had to be altered or even completely removed in order to allow room and time for the development of the core mechanism of the game. This mechanism is the Artificial Intelligence agent, which was also successfully implemented in the final version. The most challenging but also creative aspect of this study was the implementation of such a mechanism. The process of effectively developing and integrating the agent into the main core of the video game, required a lot of research, but also many trial and error attempts. New and quality knowledge was acquired during the background research period, which contributed to the algorithm being stable, but also efficient enough in order to be used by almost every smart device on the market. Moreover, many small things have changed since the very first proposal of this study. Some of the features that were overlooked or completely removed include the in-app purchasing system and the advertisement service. Both of these services require the game to be published on the online App Stores, which is not yet completed due to all the time constraints and the unfortunate

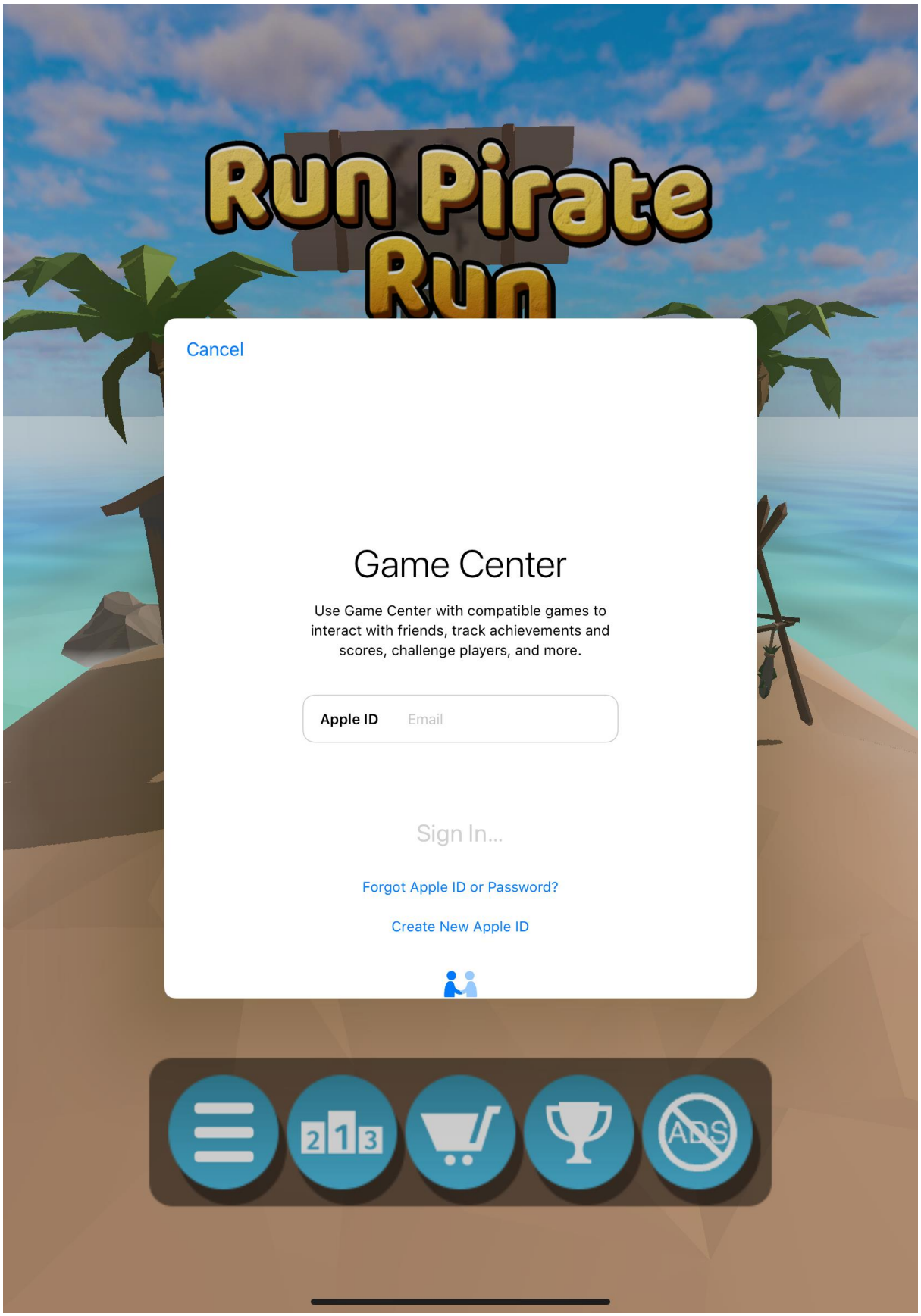
events. Despite having some small delays due to the covid-19 virus, the game is expected to be uploaded on the mobile App Stores in the upcoming months.

B) Screen shots, which demonstrate the working features of your application &

C) Testing information: data, proof & Other supporting content:

All the test and proof of the working functionalities can be identified while playing the video game. However, the following screenshots that were taken through Unity's editor, allow for a better understanding of how every feature is implemented, as well as how the Artificial Intelligence agent alters the difficulty of the game during the gameplay.

Online Game Services Screenshots and images:


Run Pirate Run
Cancel
Game Center
Use Game Center with compatible games to interact with friends, track achievements and scores, challenge players, and more.
Apple ID
Email
Sign In...
Forgot Apple ID or Password?
Create New Apple ID
ADS

Cancel

Game Center

Use Game Center with compatible games to interact with friends, track achievements and scores, challenge players, and more.

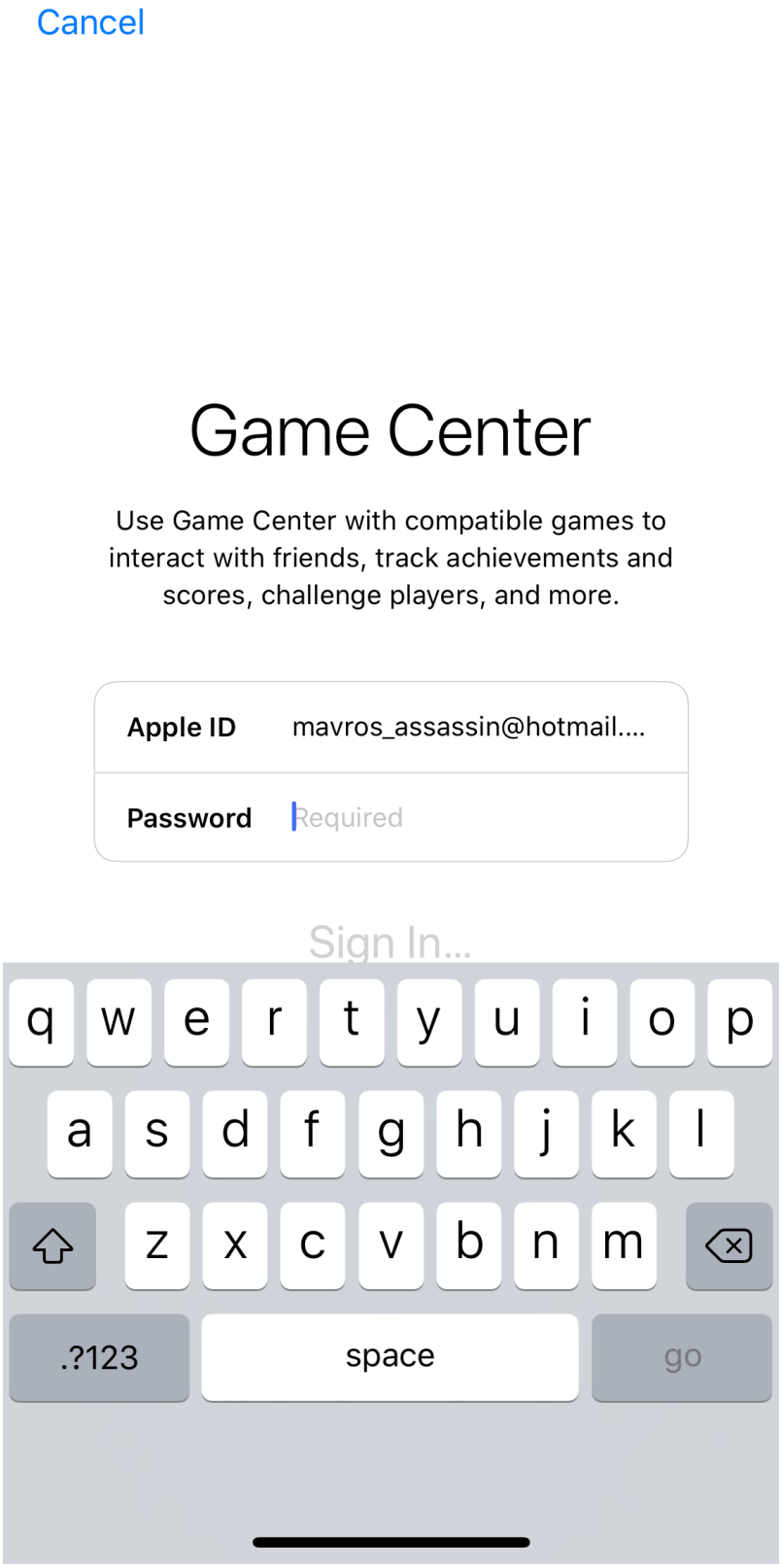
Apple ID Email

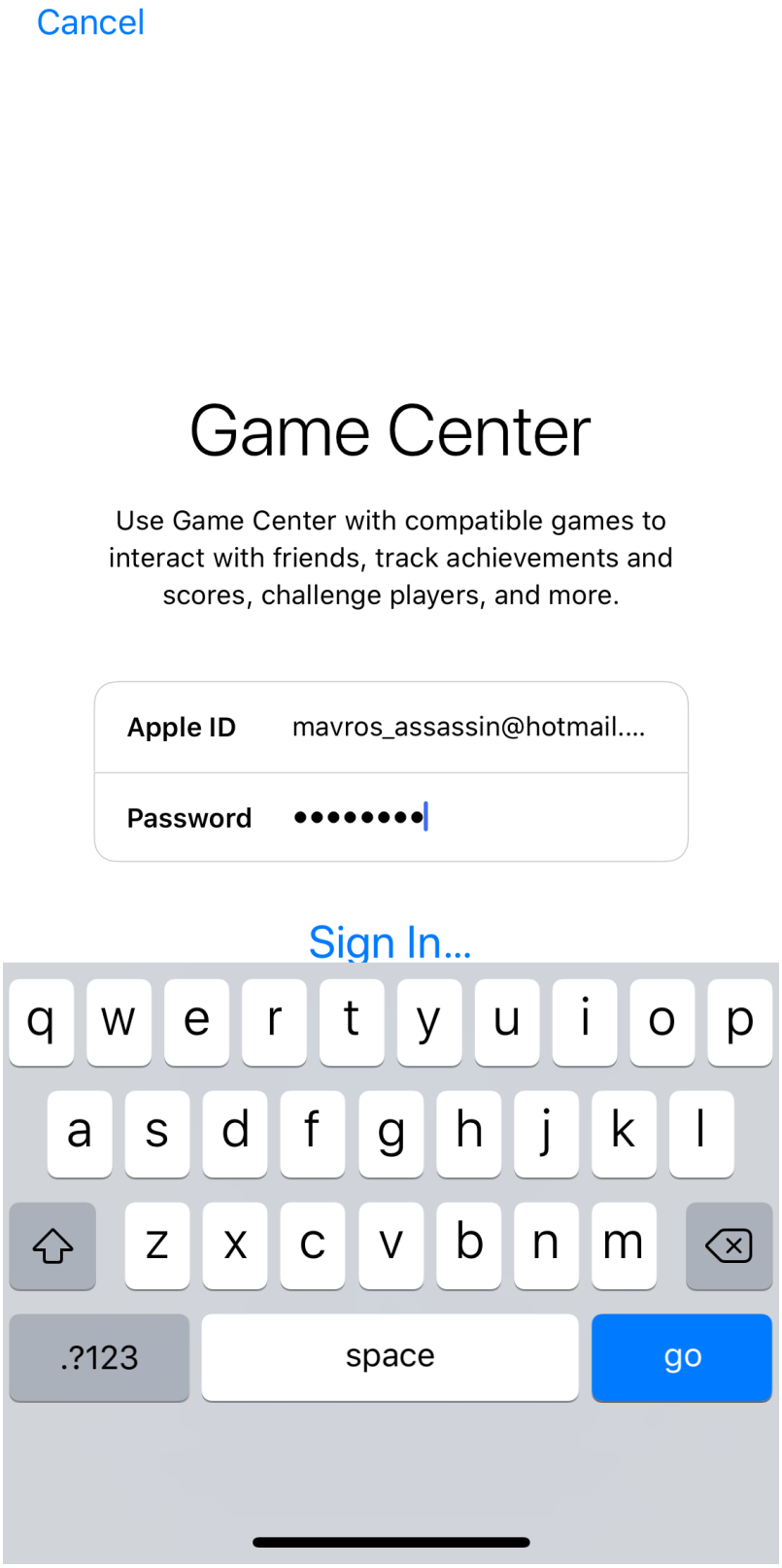
Sign In...

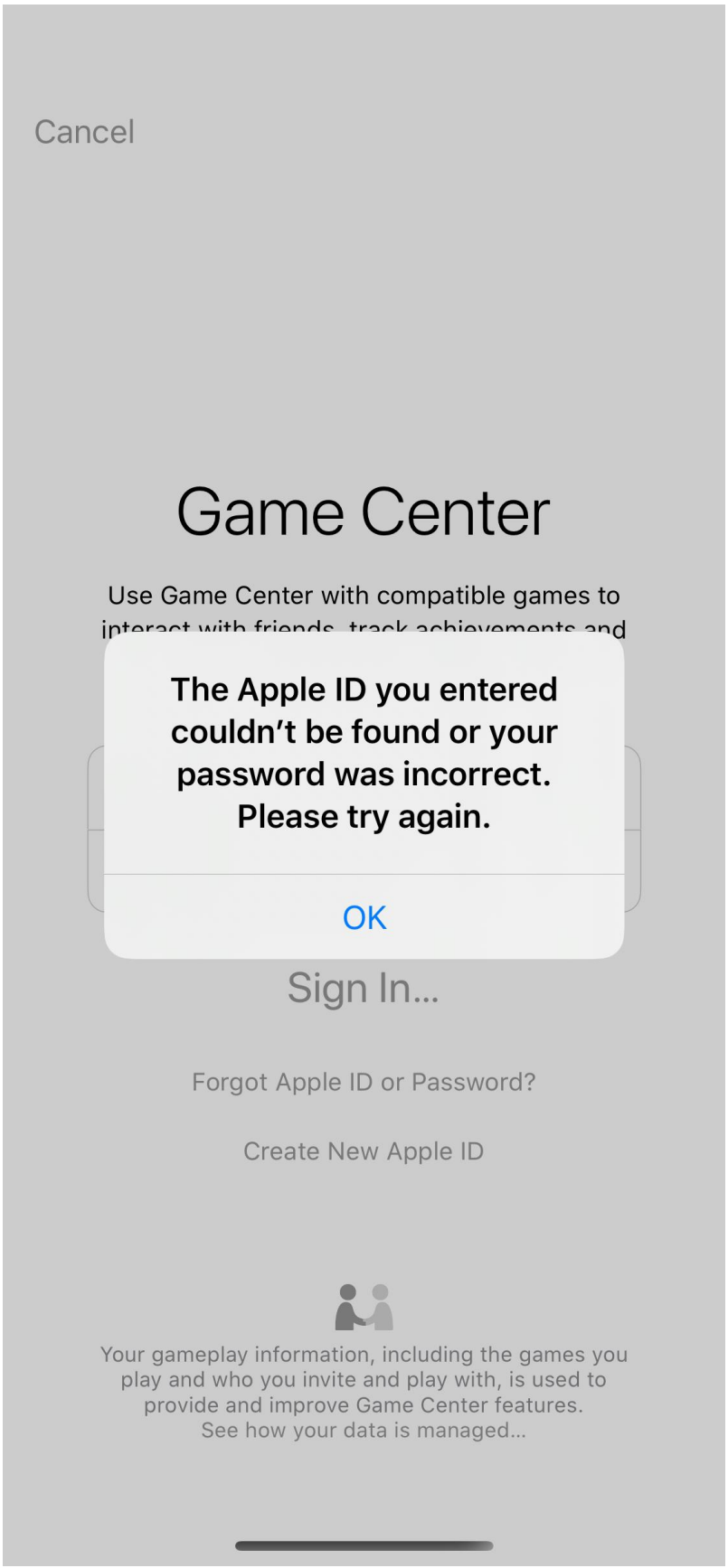
Forgot Apple ID or Password?

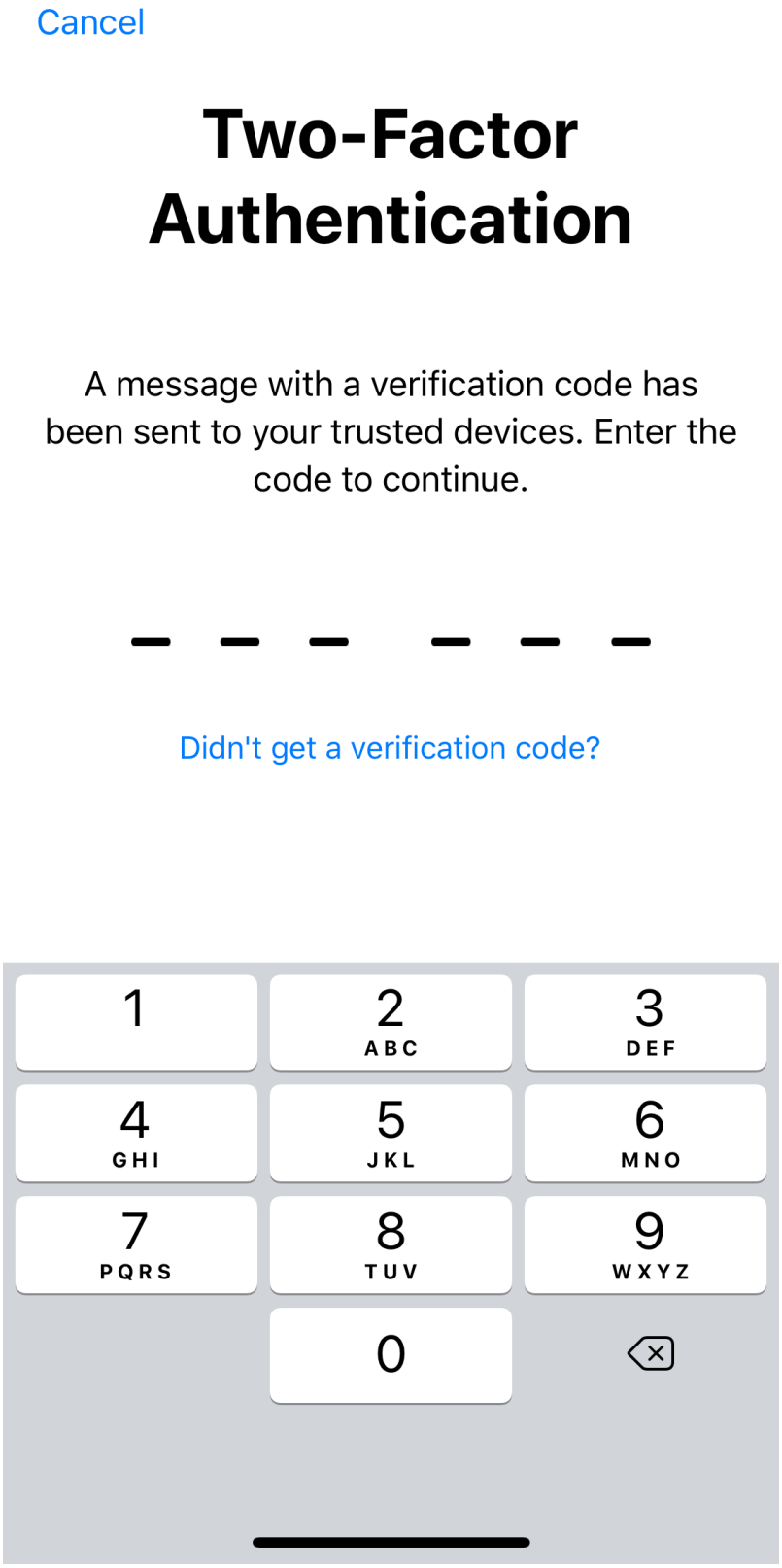
Create New Apple ID

Your gameplay information, including the games you play and who you invite and play with, is used to provide and improve Game Center features.
See how your data is managed...


Cancel
Game Center
Use Game Center with compatible games to interact with friends, track achievements and scores, challenge players, and more.
Apple ID
mavros_assassin@hotmail....
Password
Required
Sign In...
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
.?123
space
go


Cancel
Game Center
Use Game Center with compatible games to interact with friends, track achievements and scores, challenge players, and more.
Apple ID
mavros_assassin@hotmail....
Password
Sign In...
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
.?123
space
go


Cancel
Game Center
Use Game Center with compatible games to
The Apple ID you entered couldn't be found or your password was incorrect. Please try again.
OK
Sign In...
Forgot Apple ID or Password?
Create New Apple ID
Your gameplay information, including the games you play and who you invite and play with, is used to provide and improve Game Center features.
See how your data is managed...

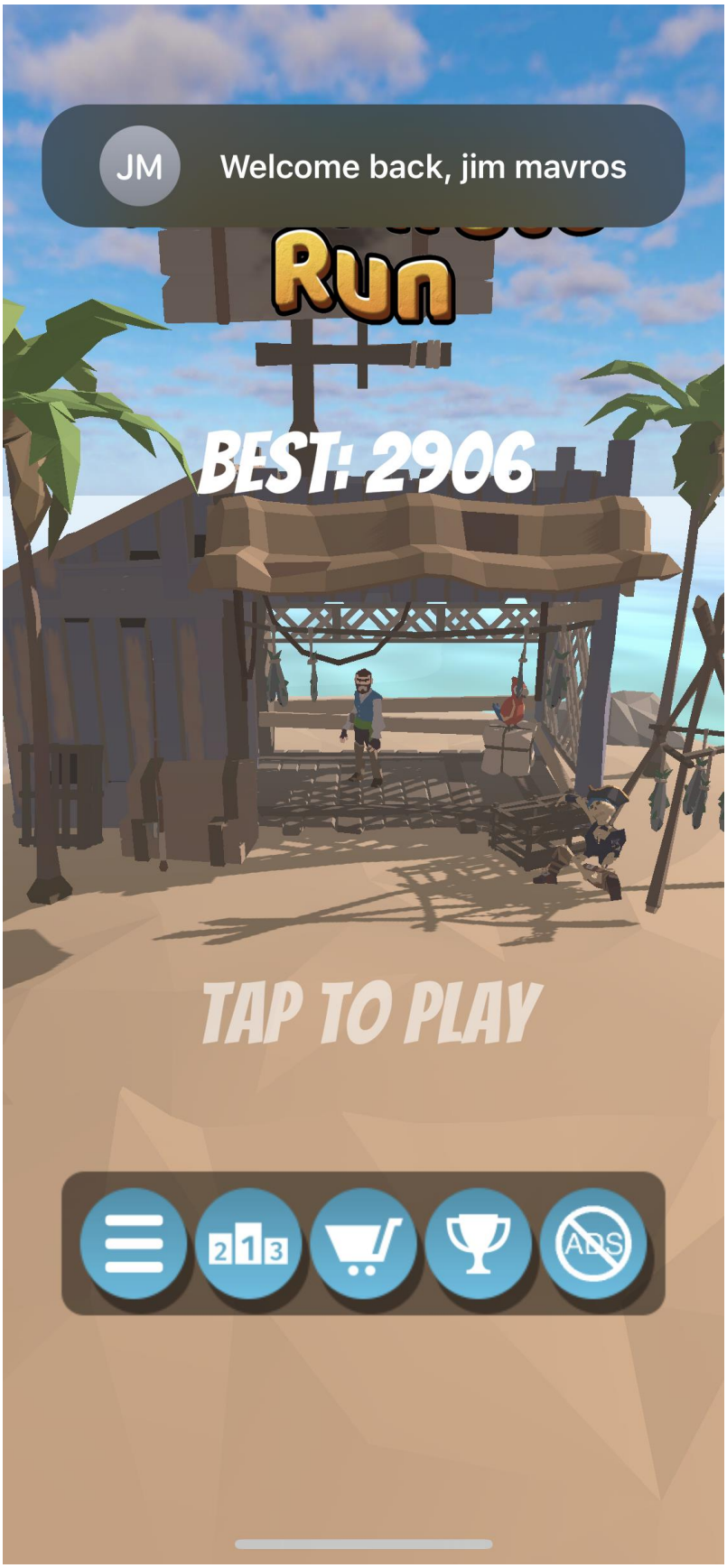

Cancel
Two-Factor Authentication
A message with a verification code has been sent to your trusted devices. Enter the code to continue.
Didn't get a verification code?
1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
0

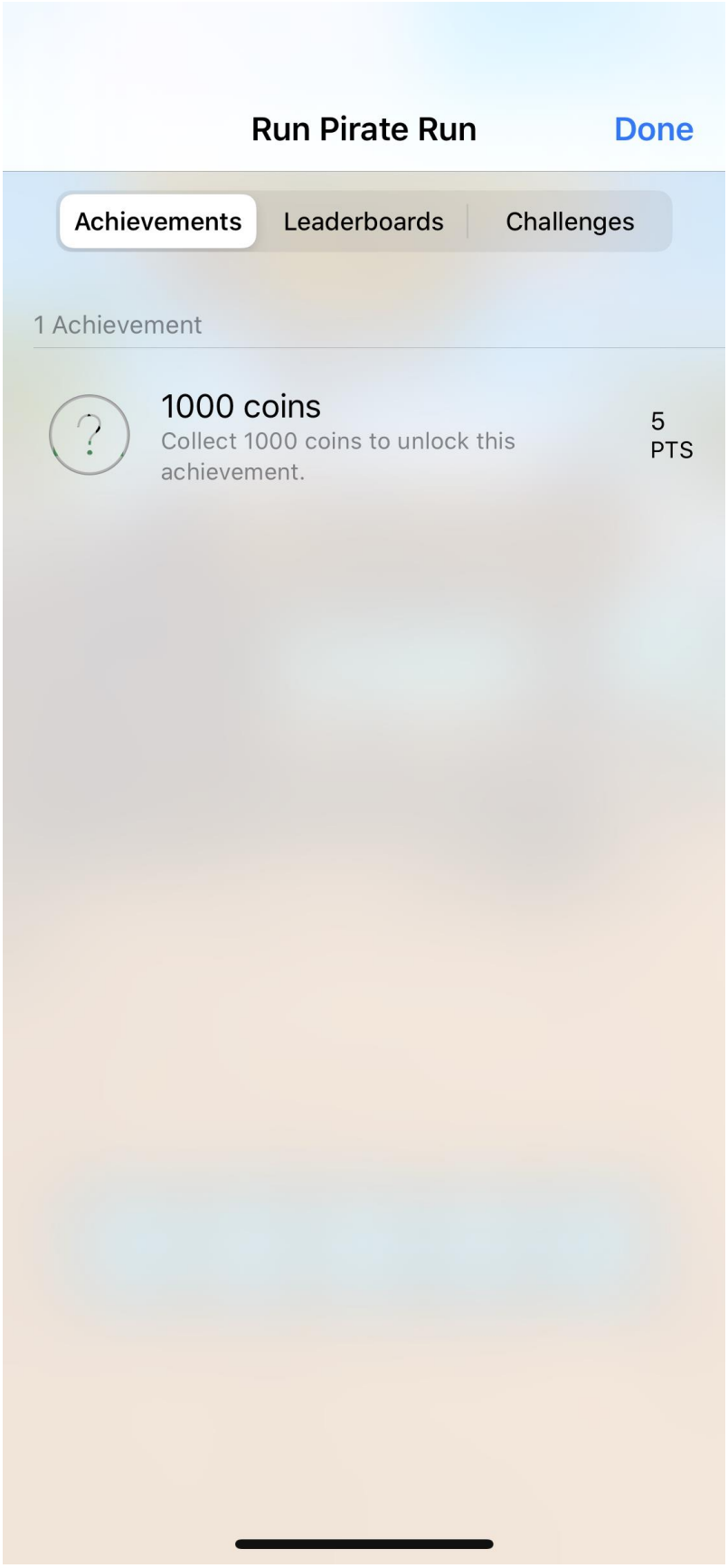
Cancel

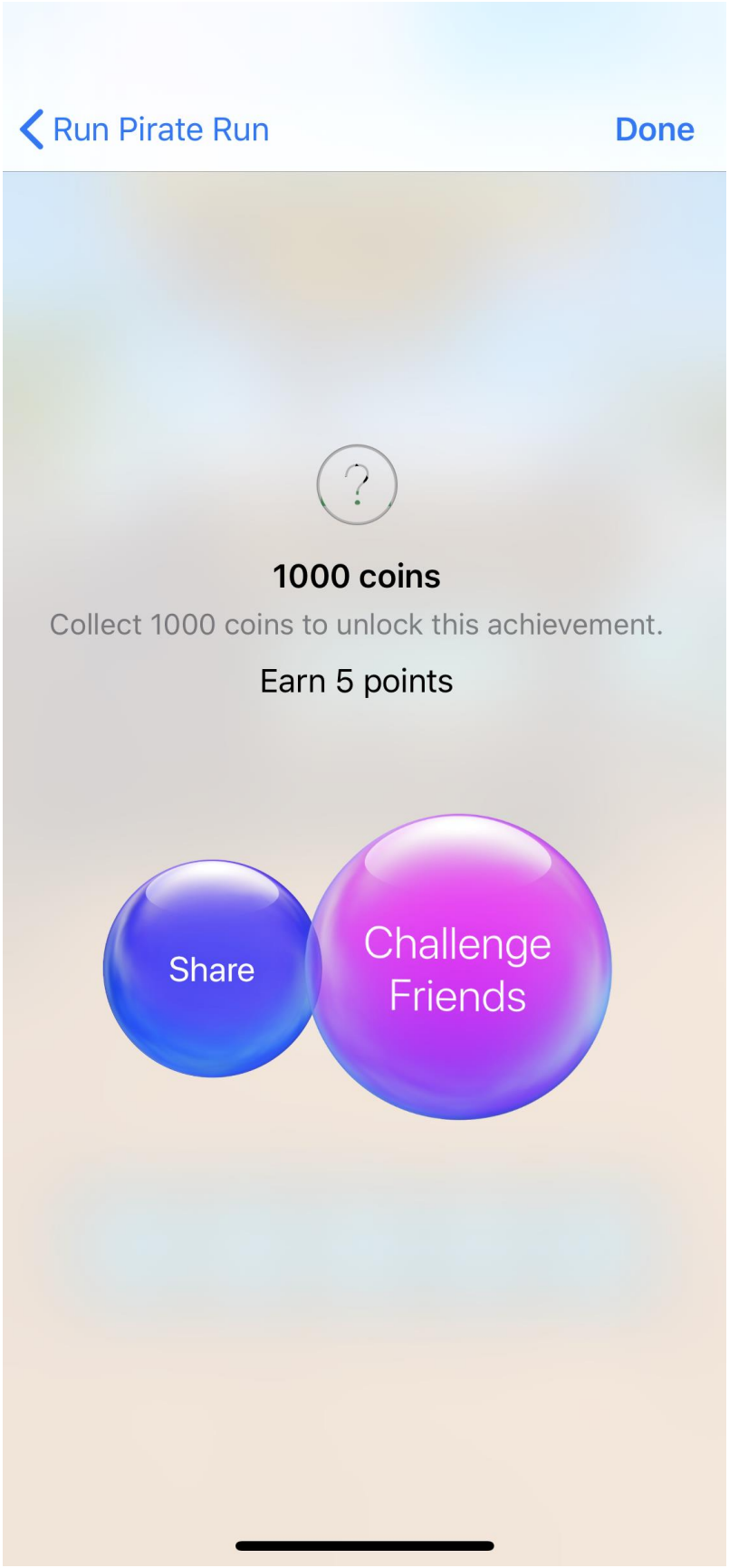
Game Center

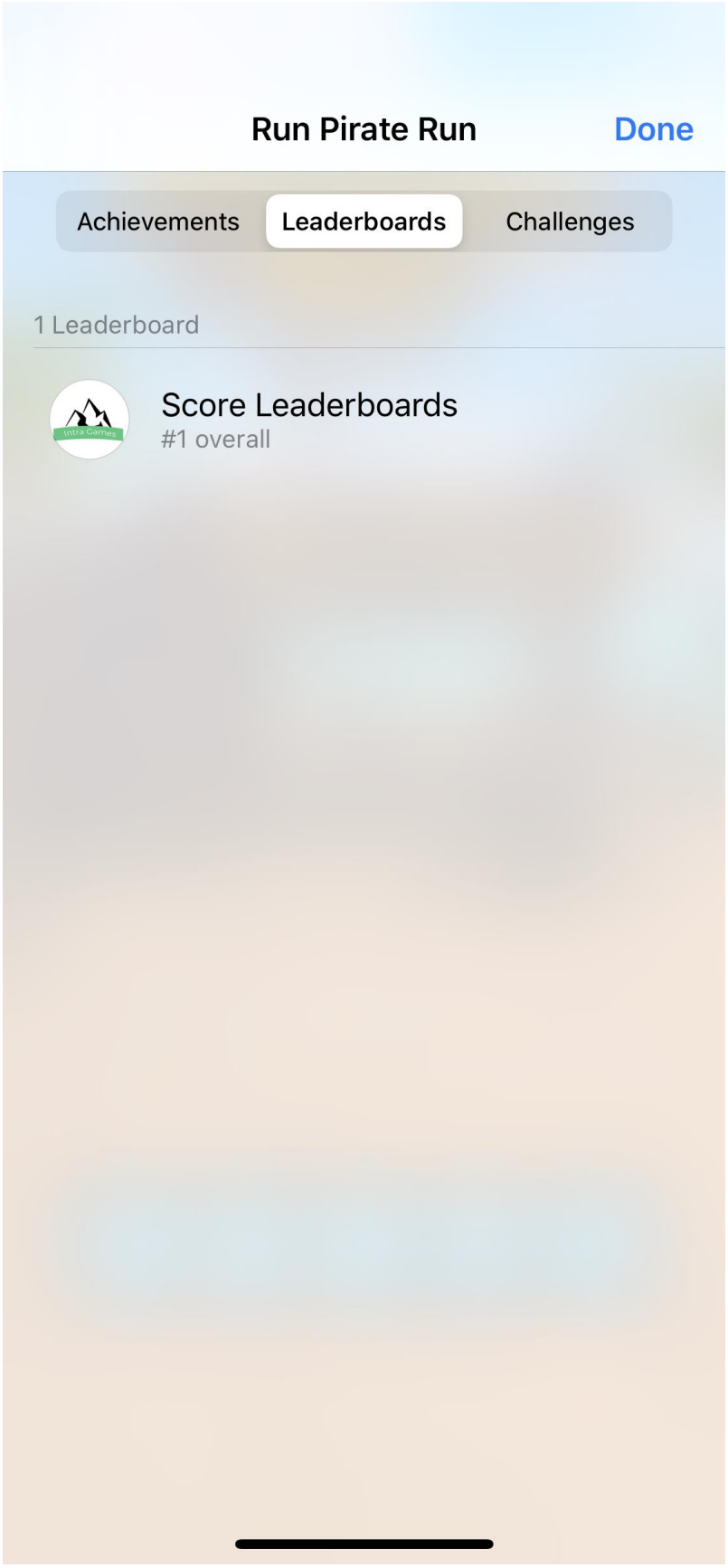
Use Game Center with compatible games to interact with friends, track achievements and scores, challenge players, and more.

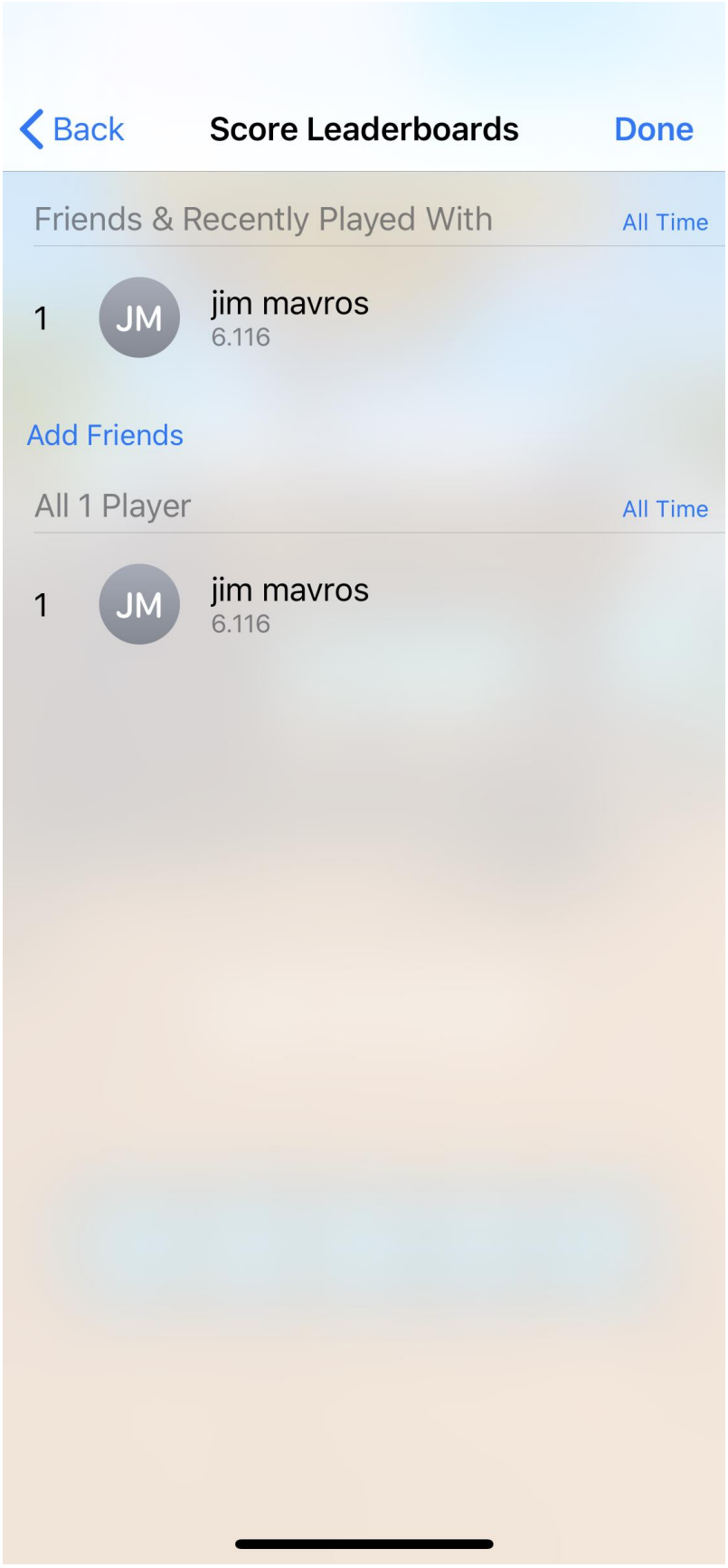
Your gameplay information, including the games you play and who you invite and play with, is used to provide and improve Game Center features.
See how your data is managed...


JM
Welcome back, jim mavros
Run
BEST: 2906
TAP TO PLAY
ADS


Run Pirate Run
Done
Achievements
Leaderboards
Challenges
1 Achievement
1000 coins
Collect 1000 coins to unlock this achievement.
5
PTS

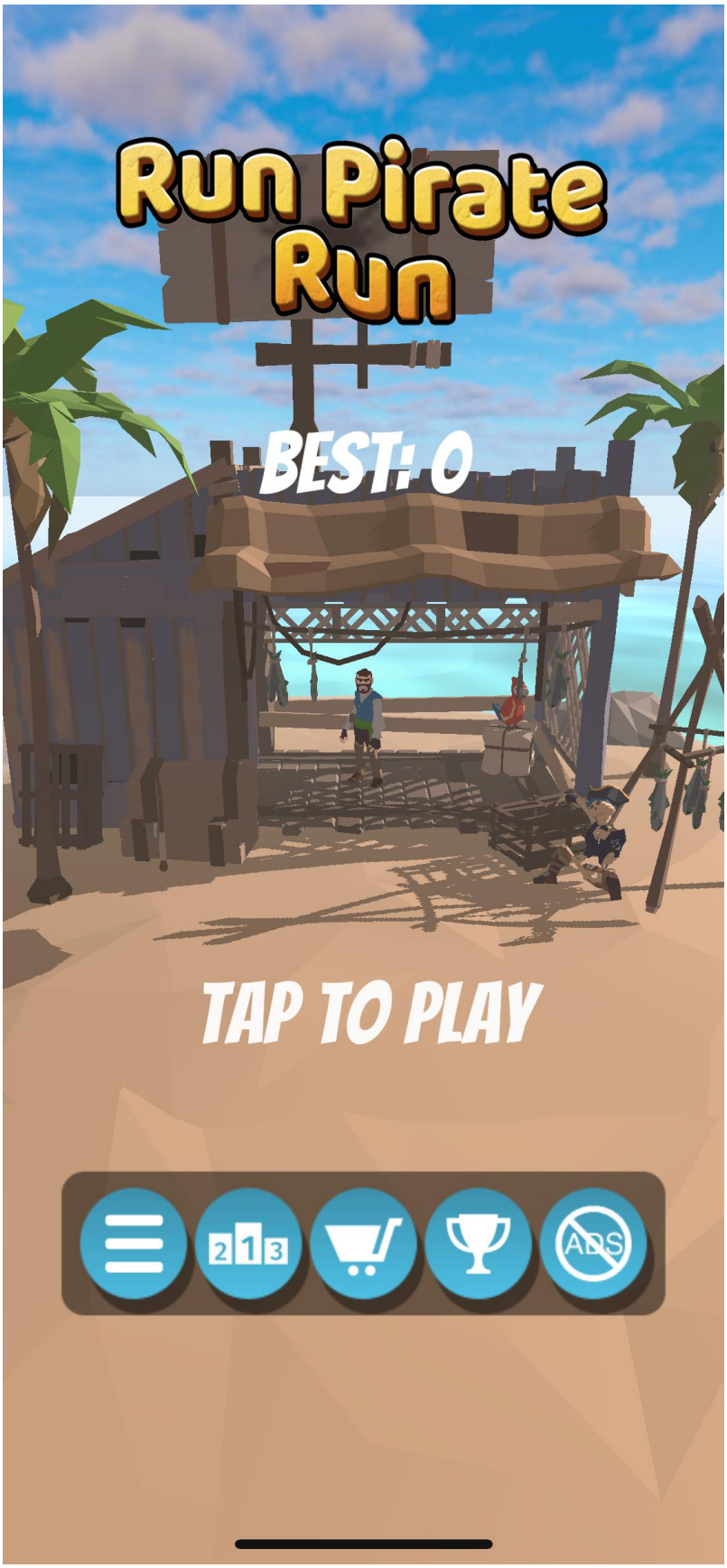

Run Pirate Run
Done
1000 coins
Collect 1000 coins to unlock this achievement.
Earn 5 points
Share
Challenge Friends

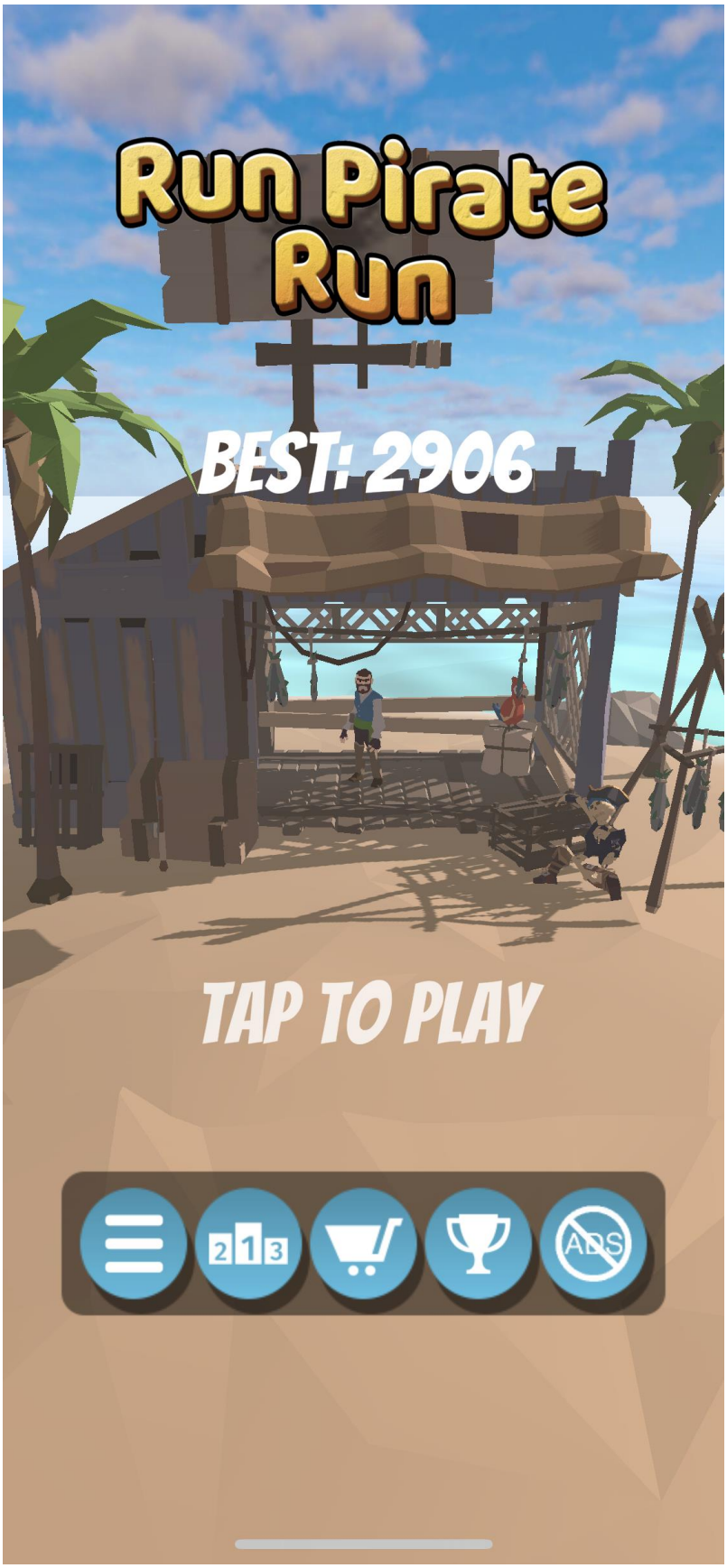

Run Pirate Run
Done
Achievements
Leaderboards
Challenges
1 Leaderboard
Score Leaderboards
#1 overall

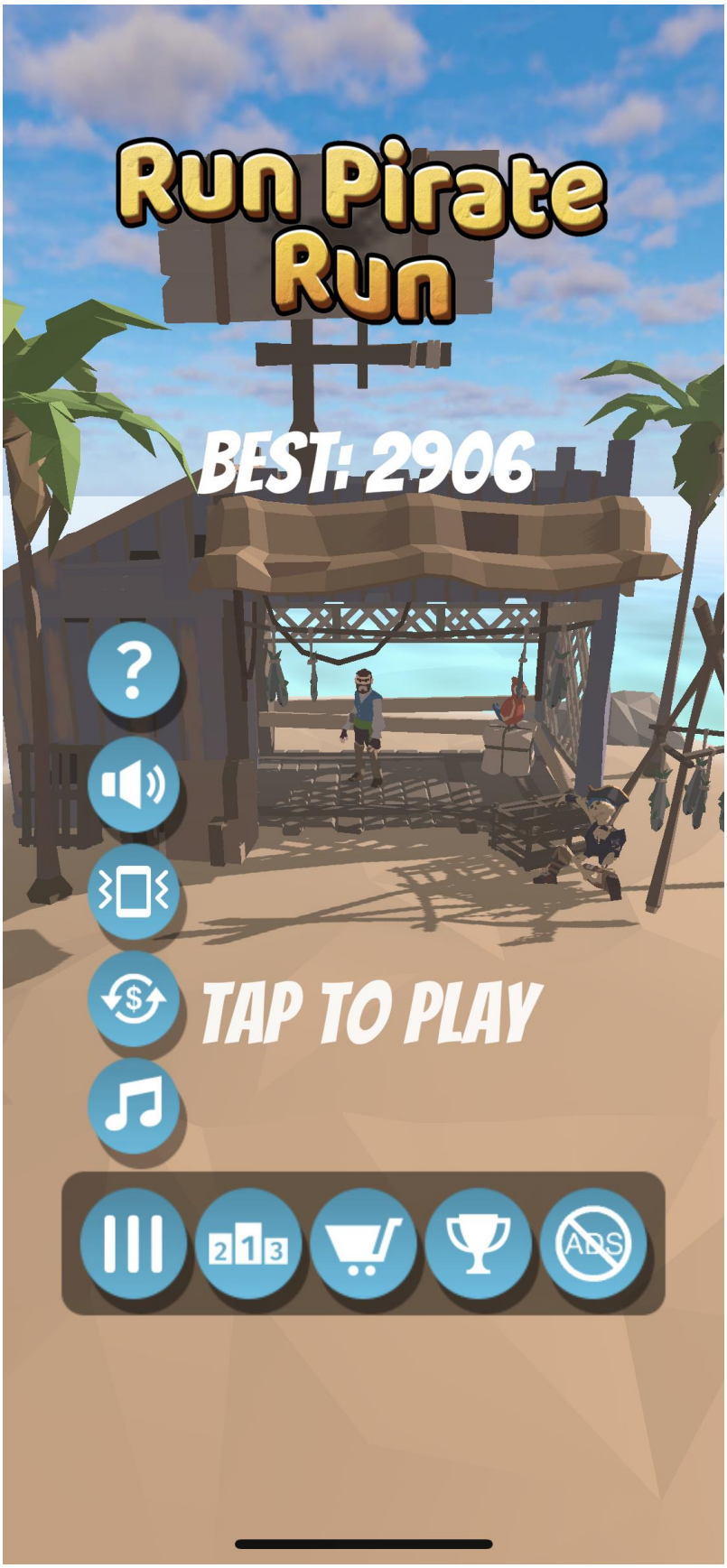

Back
Score Leaderboards
Done
Friends & Recently Played With
All Time
1
JM
jim mavros
6.116
Add Friends
All 1 Player
All Time
1
JM
jim mavros
6.116

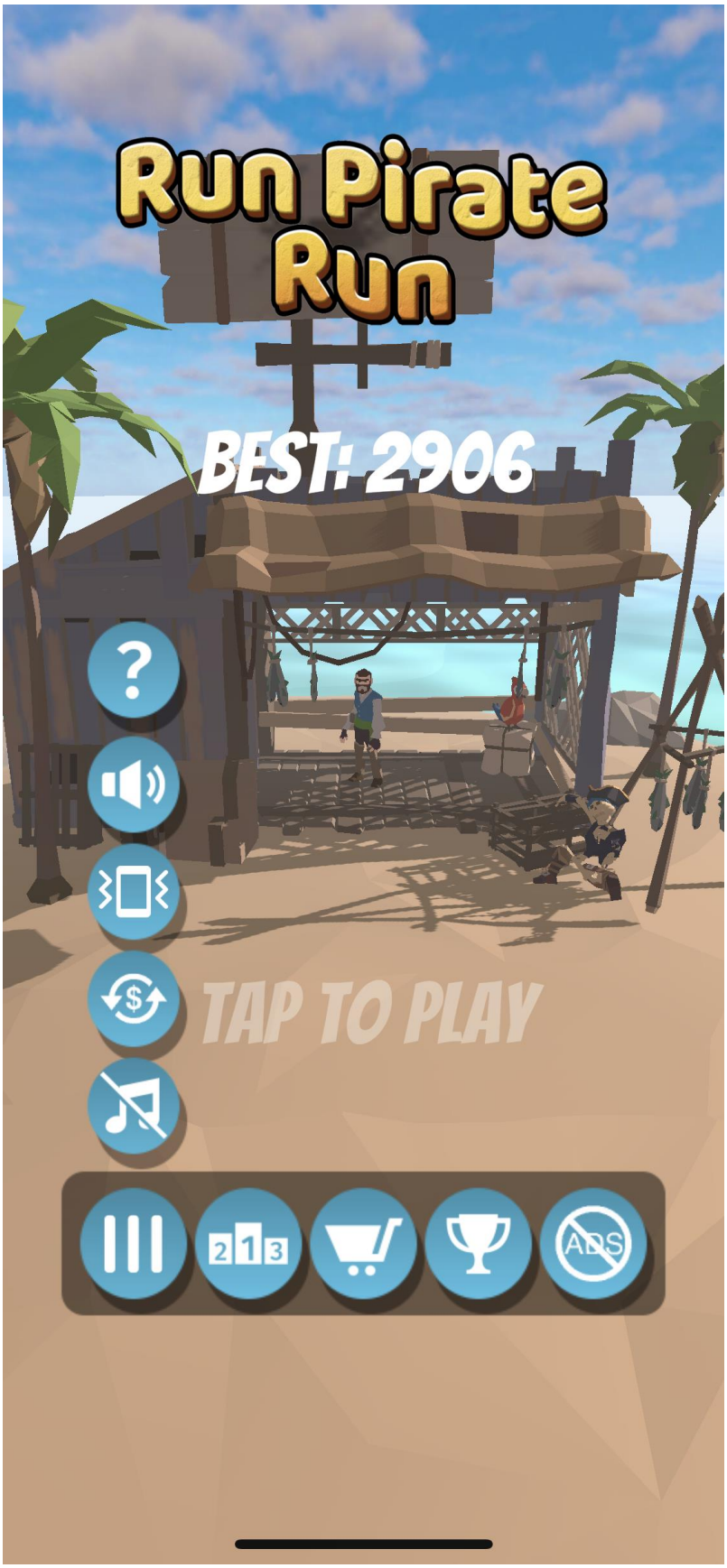

Score Leaderboards
Report
JM
jim mavros
6.116
#1 earned Saturday 1:06 AM
Share
Challenge Friends

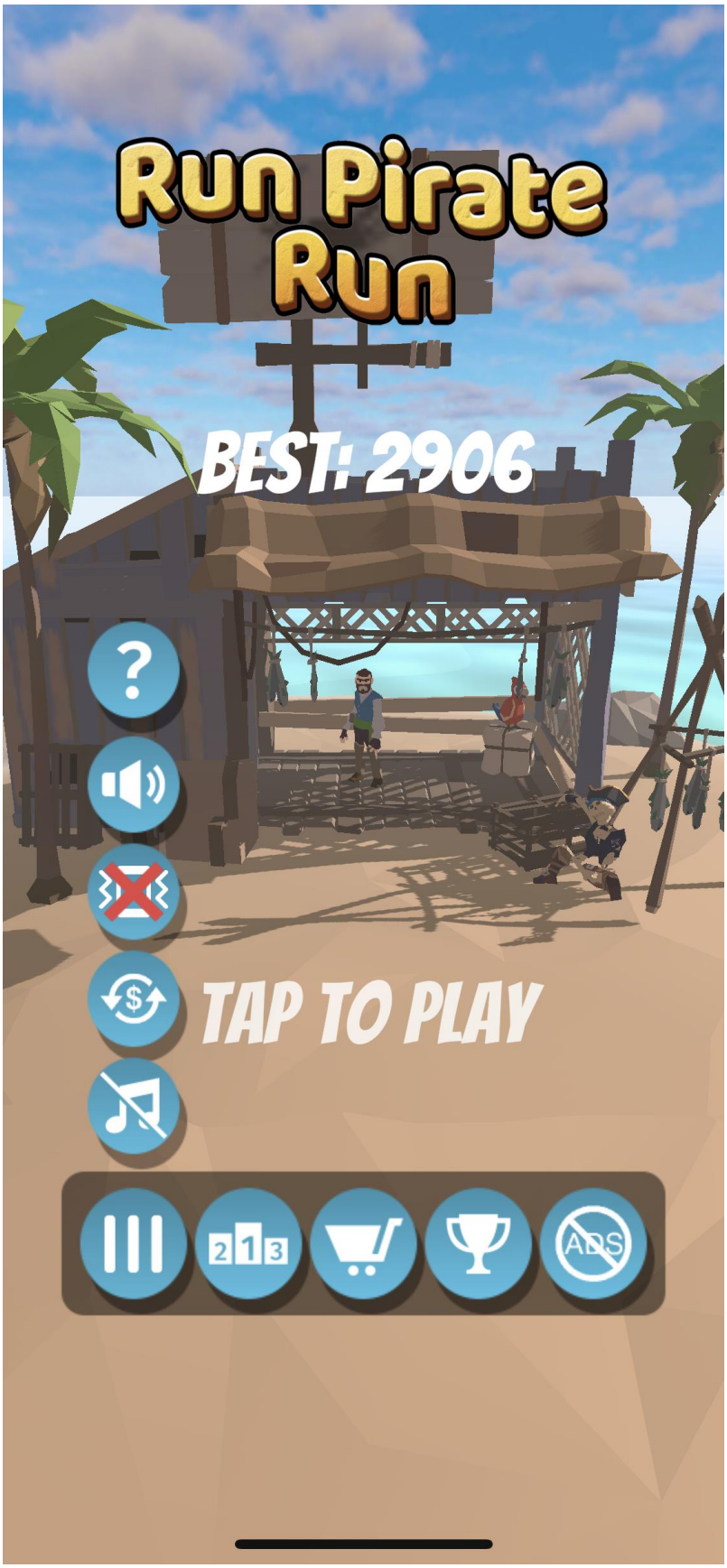
Main Menu, User Interface & UI Button Events:

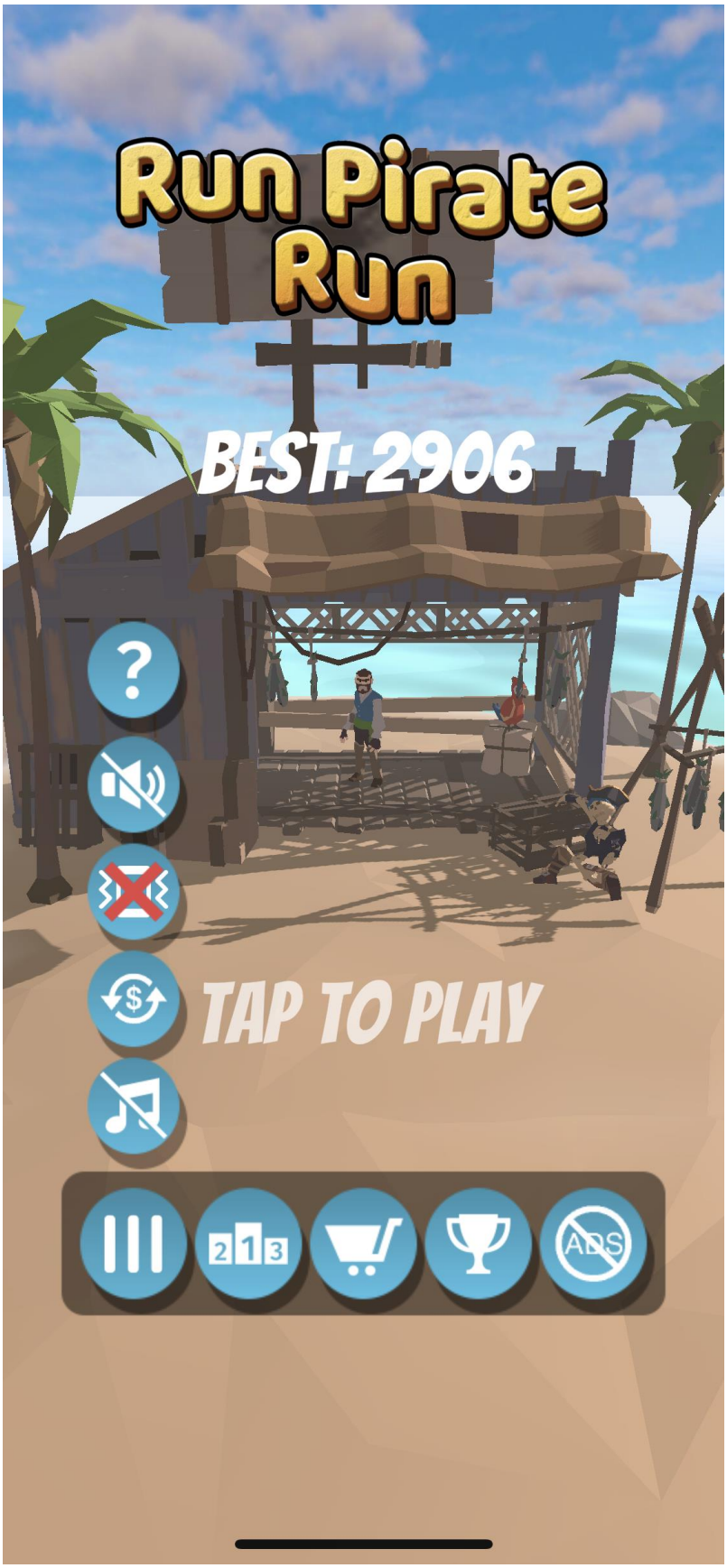

Run Pirate Run
BEST: 0
TAP TO PLAY
ADS


Run Pirate Run
BEST: 2906
TAP TO PLAY
ADS

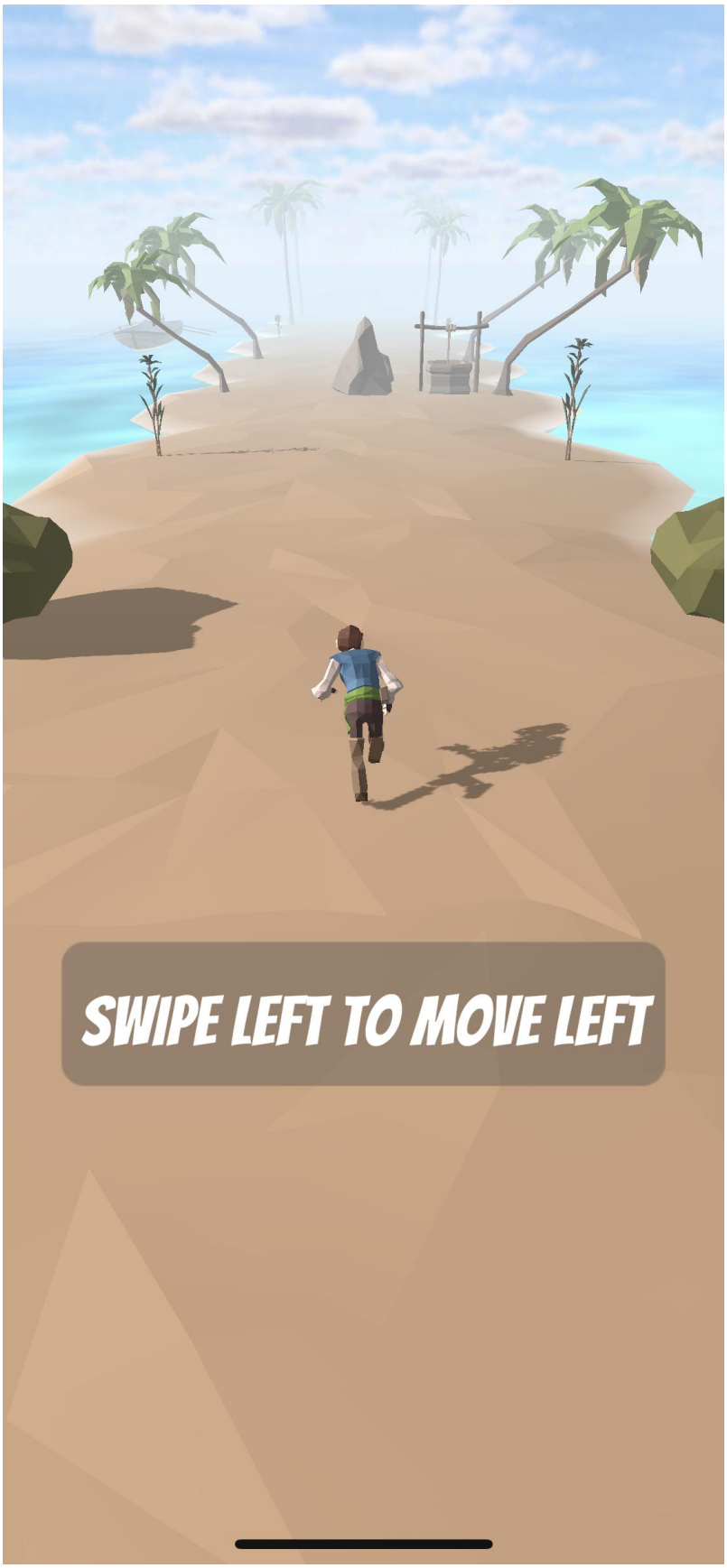

Run Pirate Run
BEST: 2906
TAP TO PLAY
ADS


Run Pirate Run
BEST: 2906
TAP TO PLAY
ADS


Run Pirate Run
BEST: 2906
TAP TO PLAY
ADS

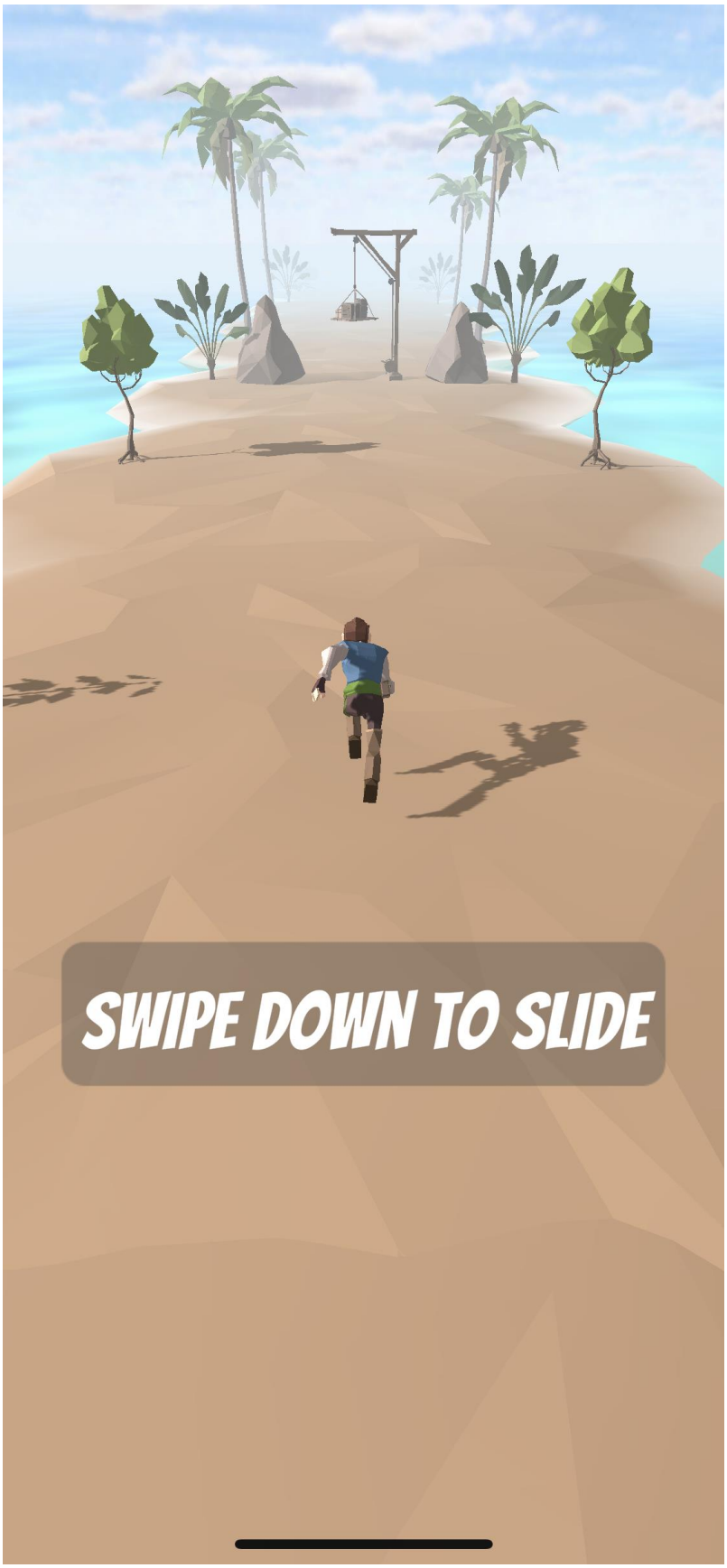

Run Pirate Run
BEST: 2906
TAP TO PLAY
ADS

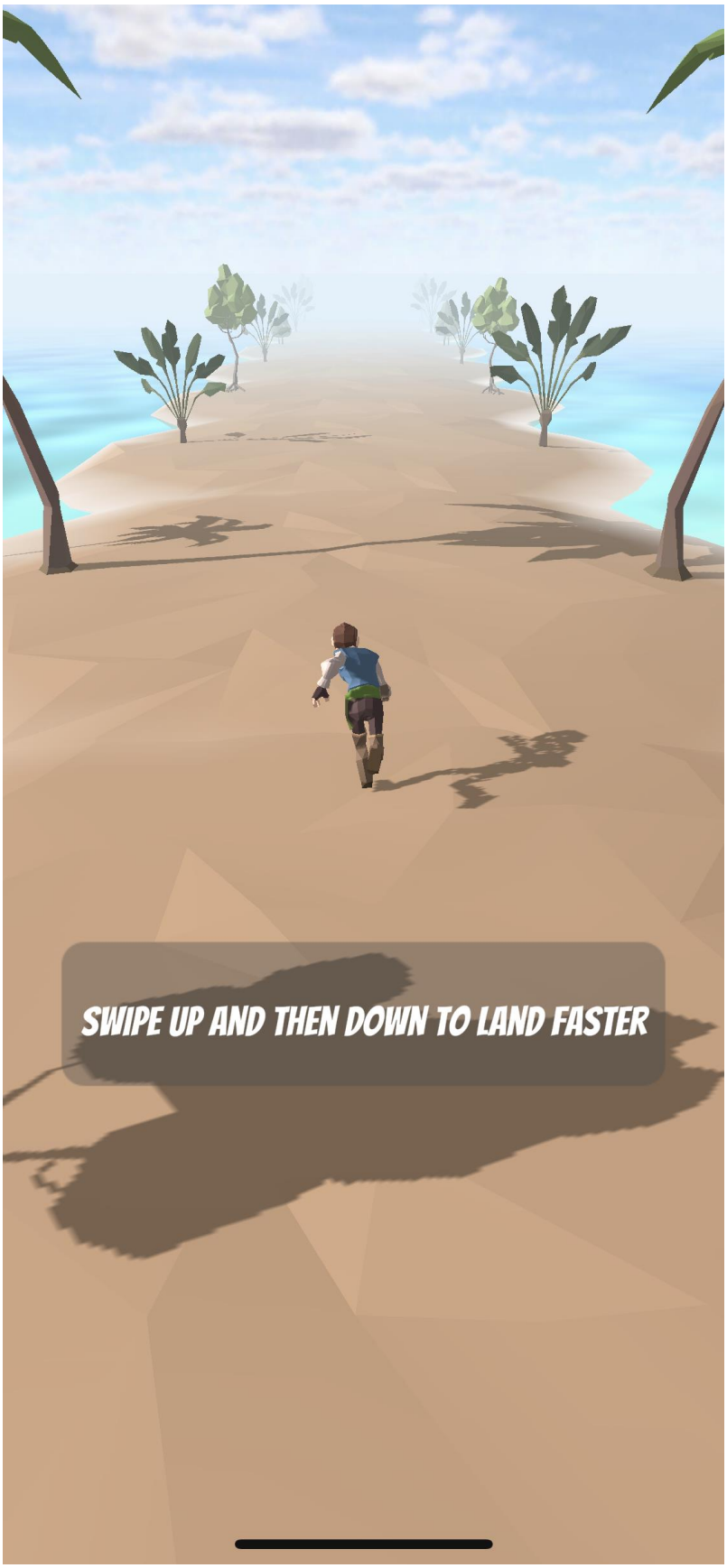
Tutorial Level:

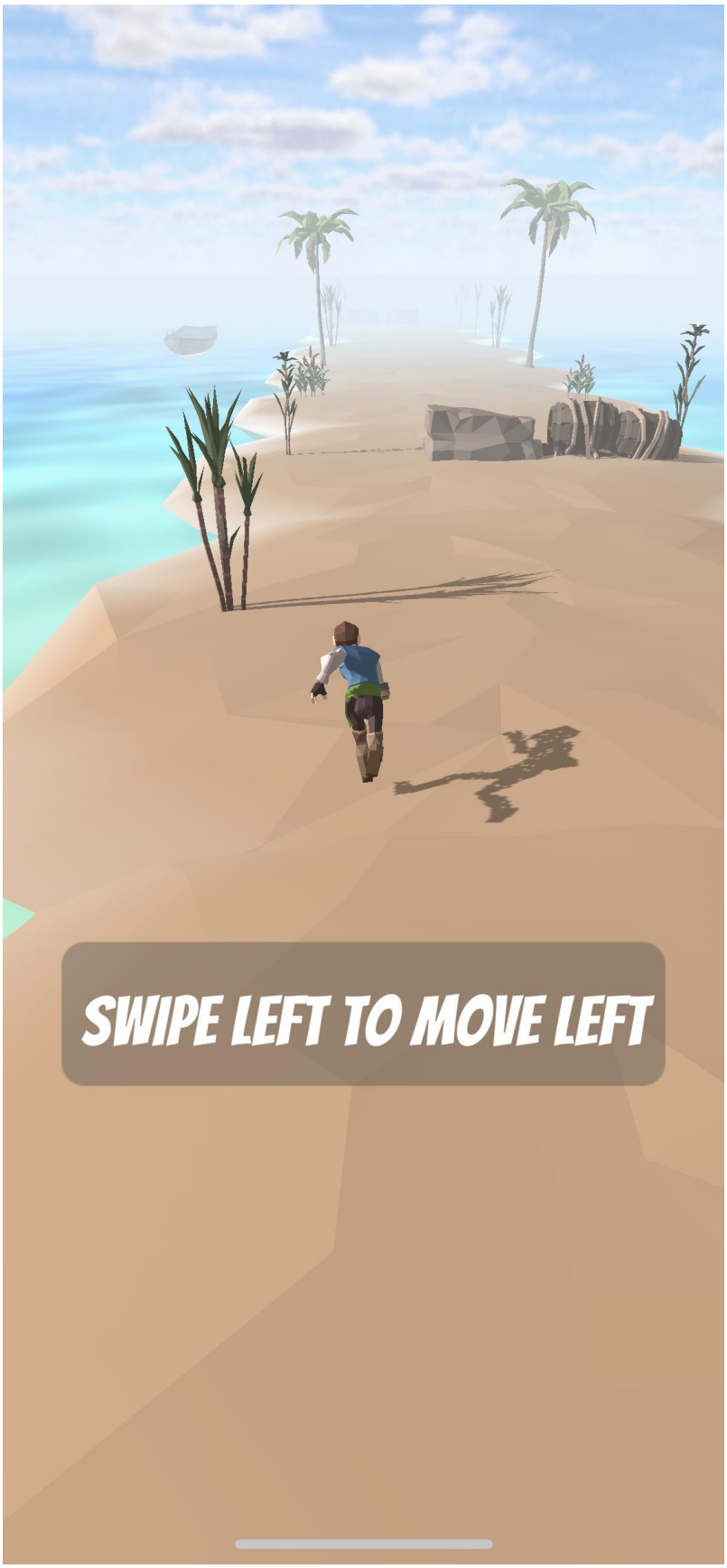

SWIPE LEFT TO MOVE LEFT


SWIPE RIGHT TO MOVE RIGHT

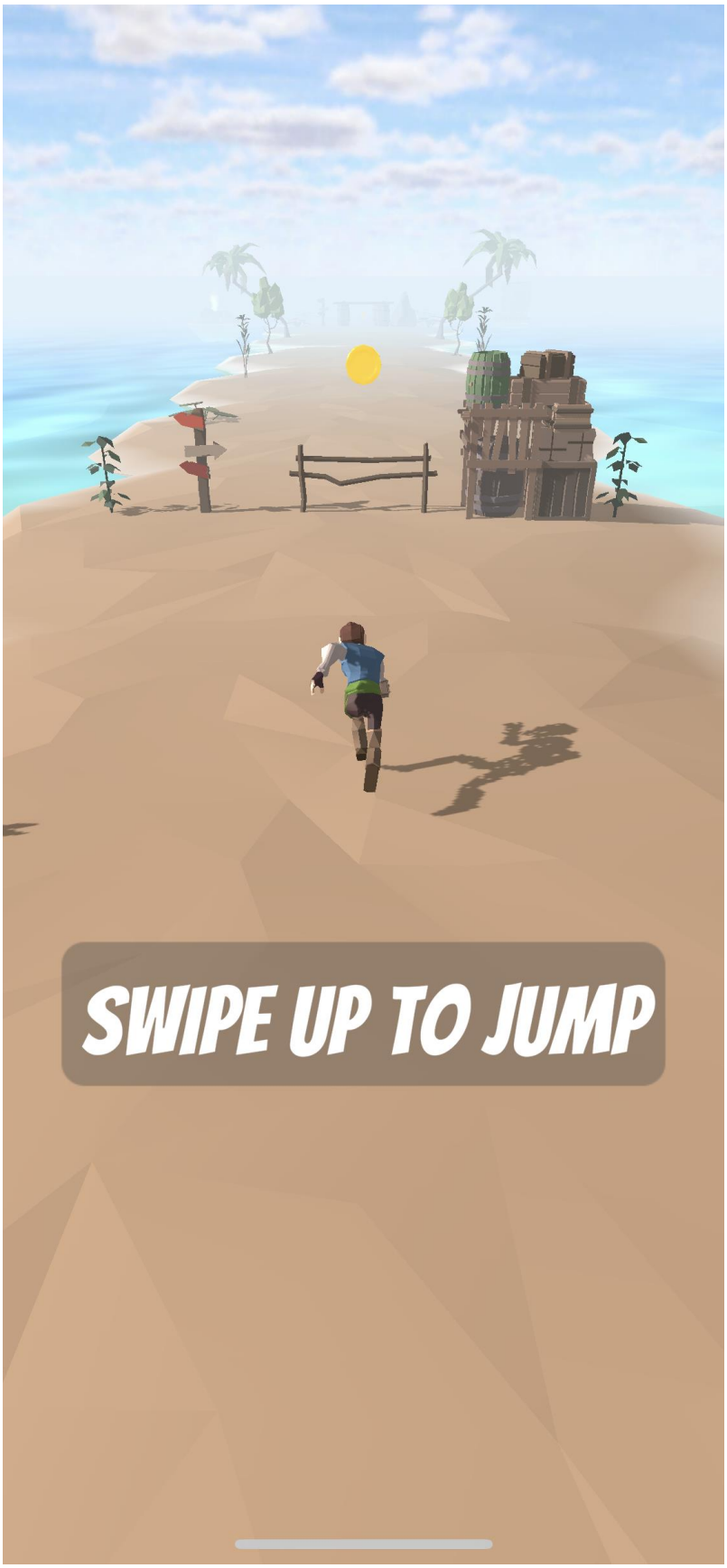

SWIPE UP TO JUMP


SWIPE DOWN TO SLIDE

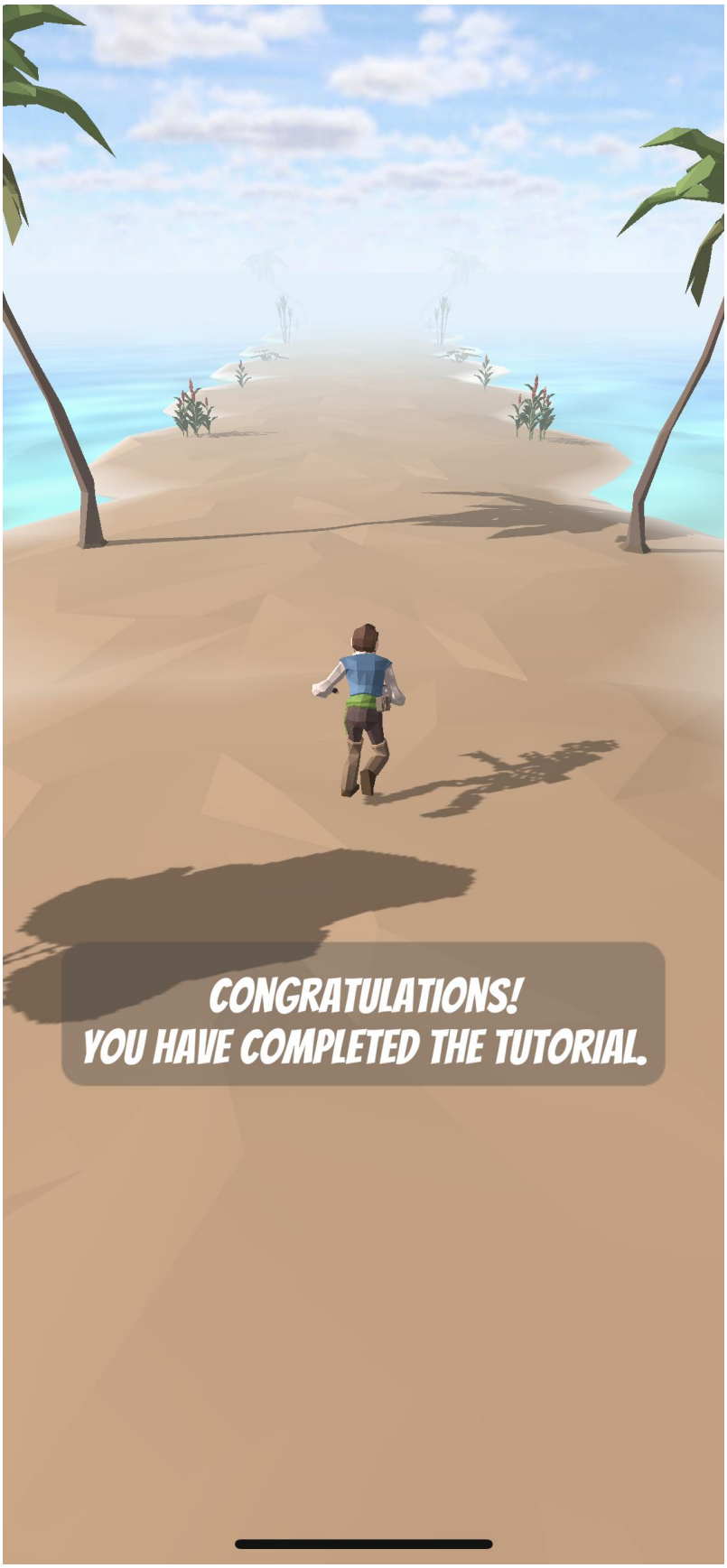

SWIPE UP AND THEN DOWN TO LAND FASTER


SWIPE LEFT TO MOVE LEFT


SWIPE RIGHT TO MOVE RIGHT

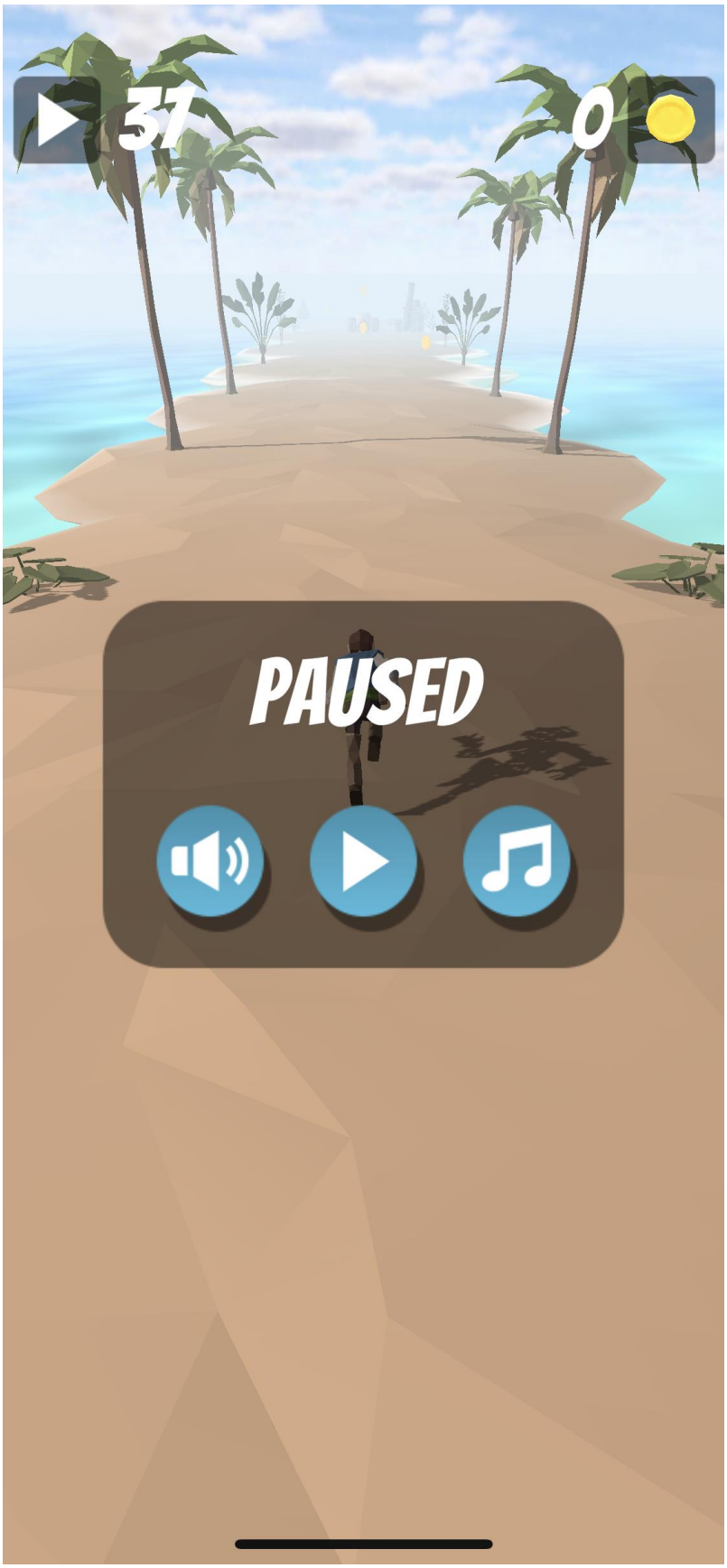

SWIPE UP TO JUMP


NOT SO WELL, RETURNING TO BASE..

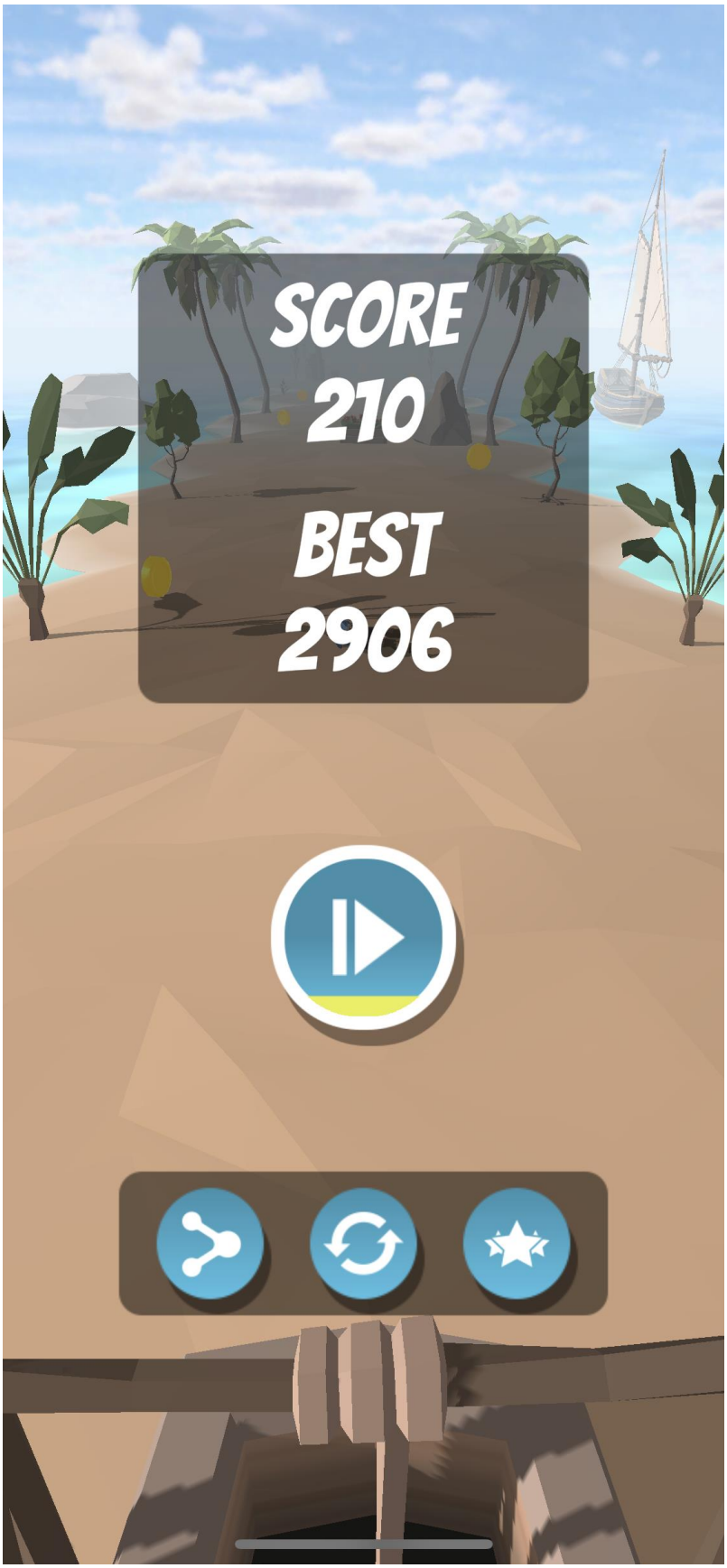

CONGRATULATIONS!
YOU HAVE COMPLETED THE TUTORIAL.


RETURNING TO BASE!

Game Manager, Player Manager & Gameplay:


37
0
PAUSED


129
3


SCORE
210
BEST
2906


689
23


876
30


949
34


SCORE
972
BEST
2906

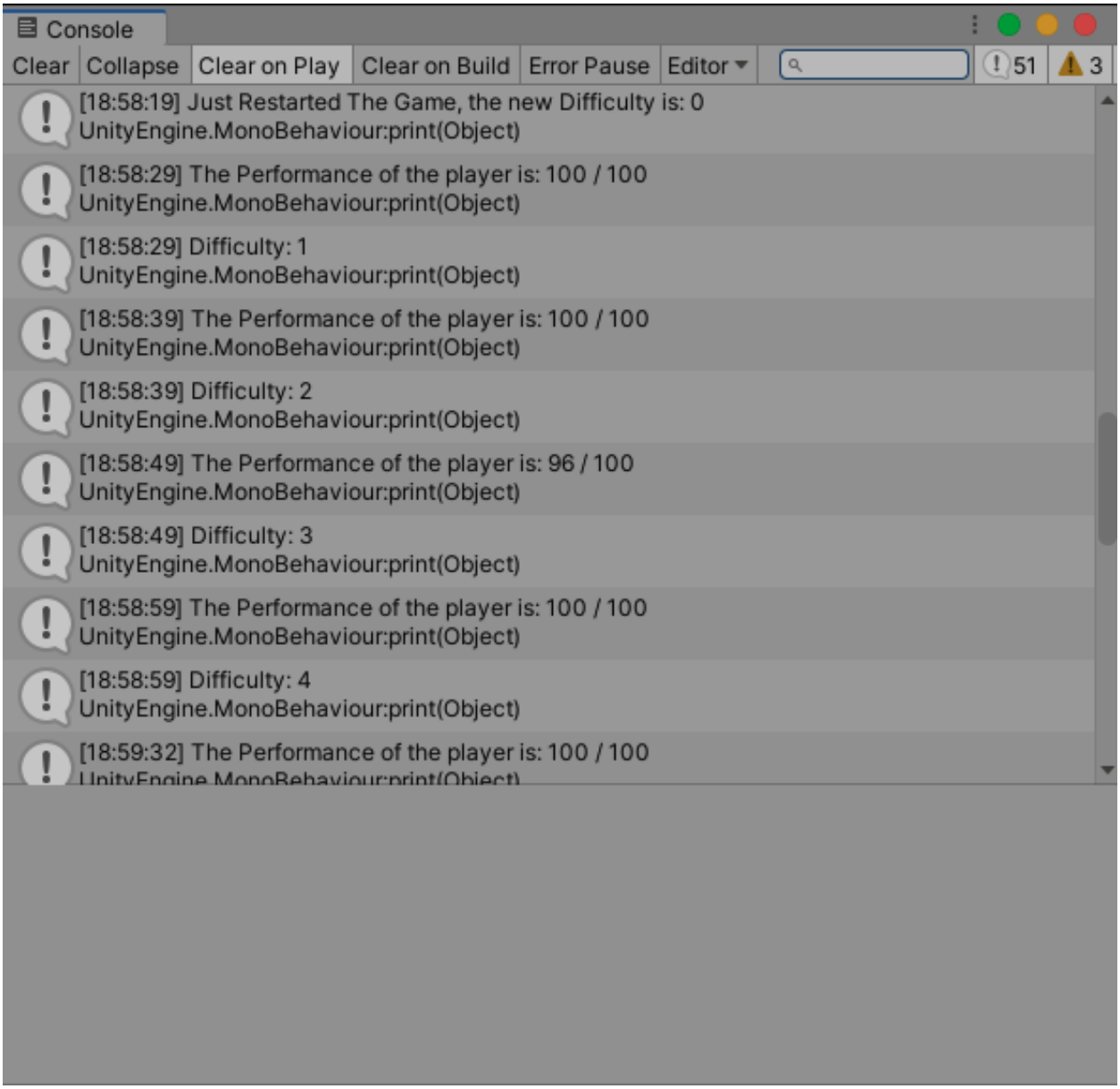

581
9

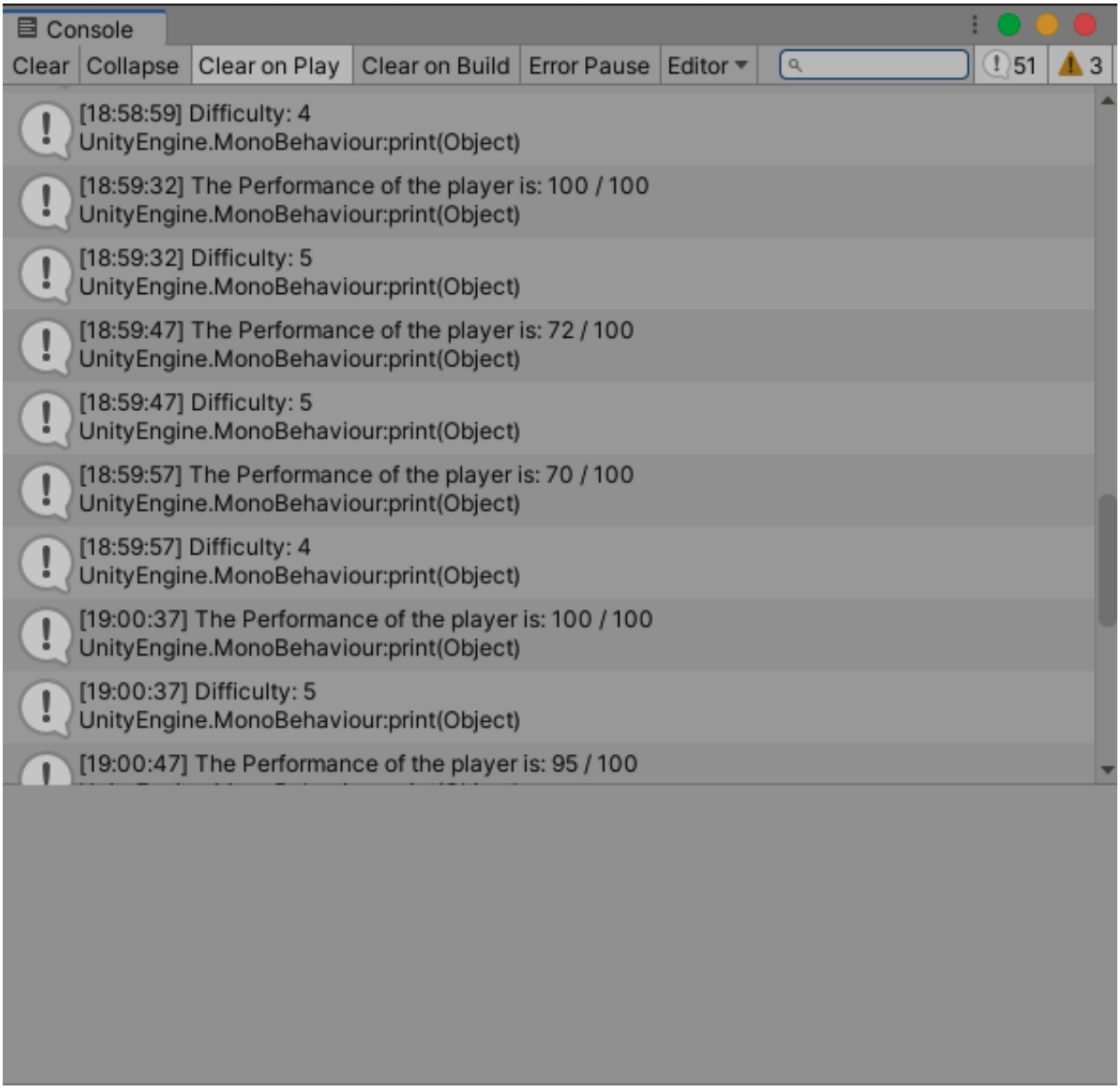

719
11

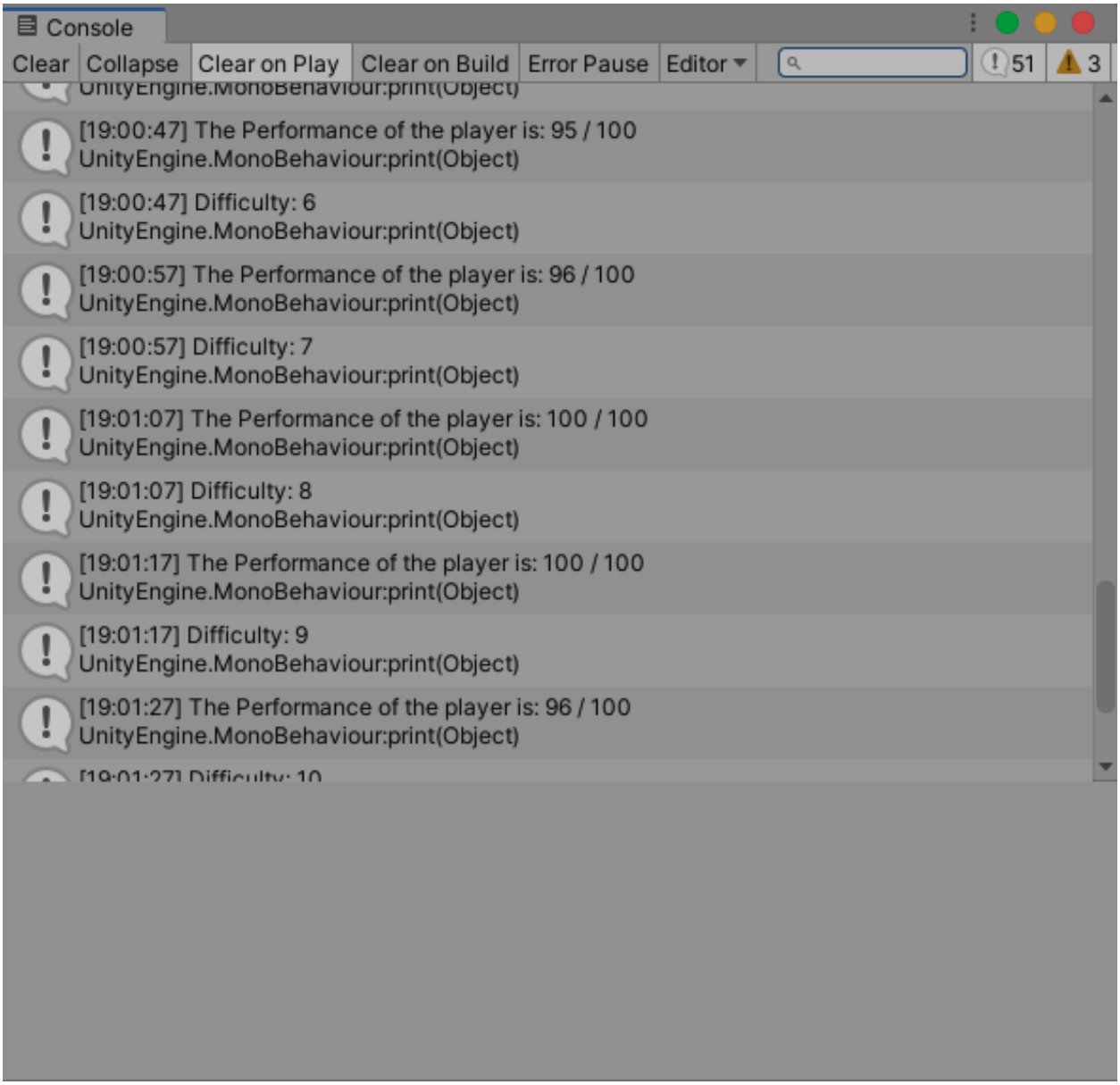

1868
55

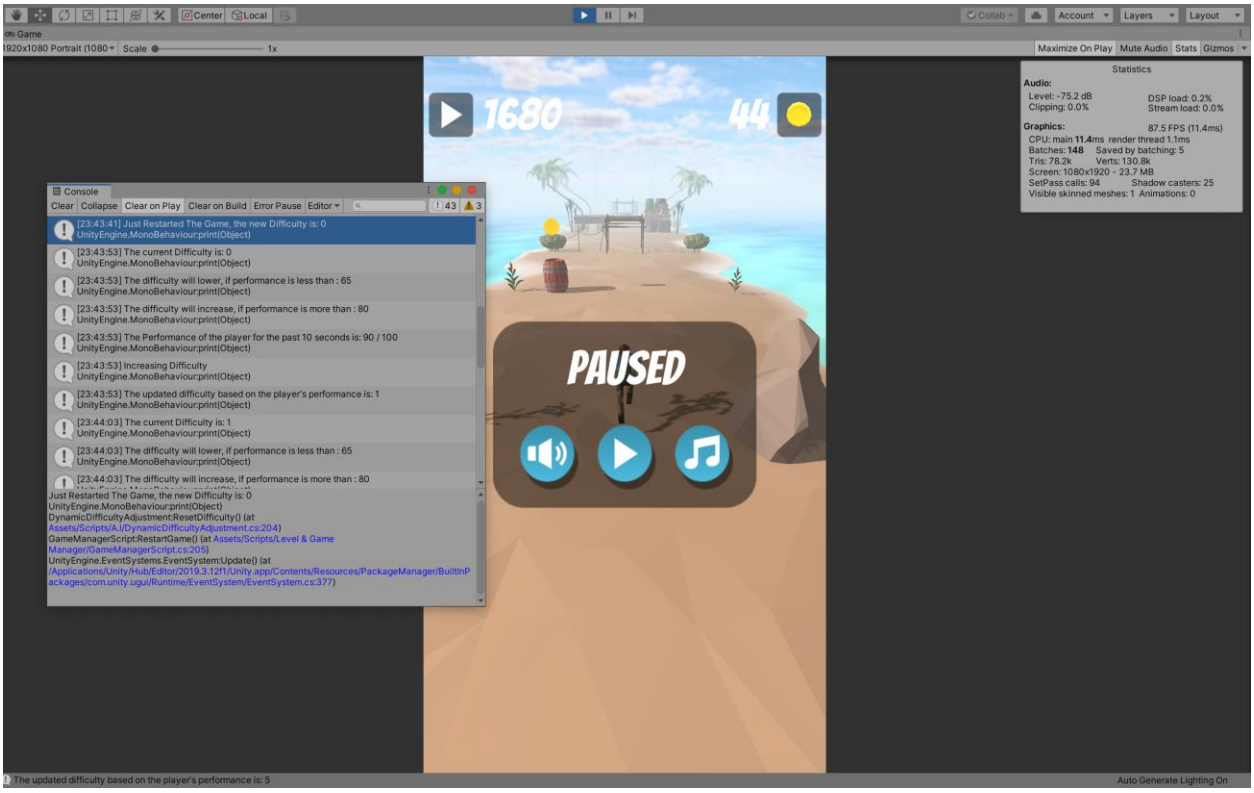

2344
65

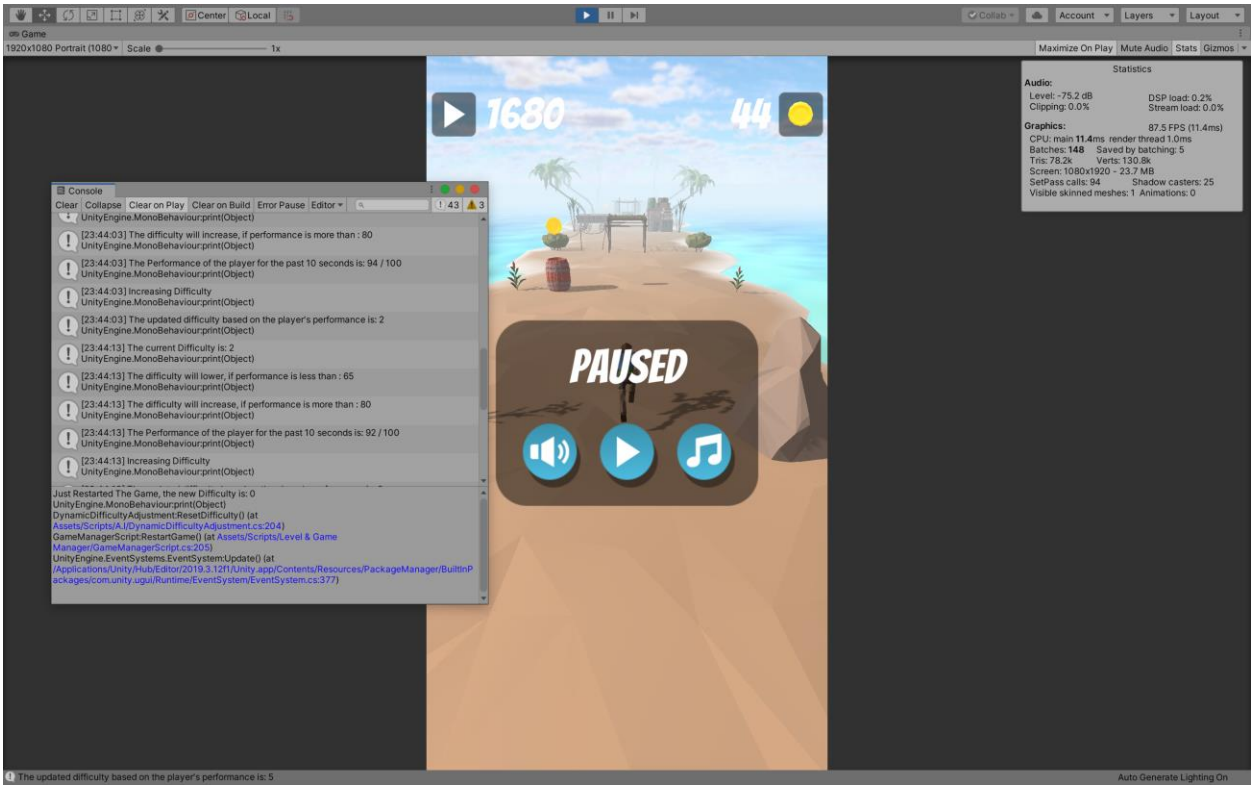
PCG manager, Artificial Intelligence DDA agent & other components/features:

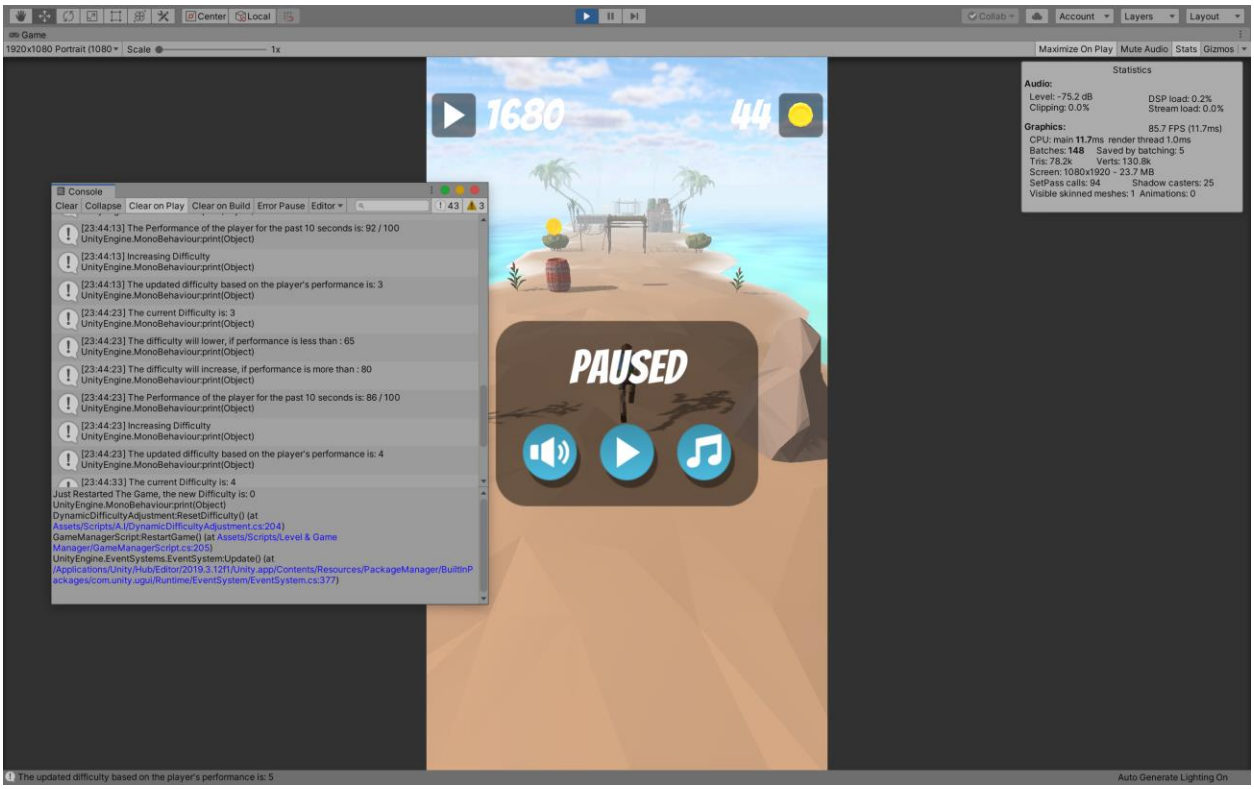

Console
Clear
Collapse
Clear on Play
Clear on Build
Error Pause
Editor
51
3
[18:58:19] Just Restarted The Game, the new Difficulty is: 0
UnityEngine.MonoBehaviour:print(Object)
[18:58:29] The Performance of the player is: 100 / 100
UnityEngine.MonoBehaviour:print(Object)
[18:58:29] Difficulty: 1
UnityEngine.MonoBehaviour:print(Object)
[18:58:39] The Performance of the player is: 100 / 100
UnityEngine.MonoBehaviour:print(Object)
[18:58:39] Difficulty: 2
UnityEngine.MonoBehaviour:print(Object)
[18:58:49] The Performance of the player is: 96 / 100
UnityEngine.MonoBehaviour:print(Object)
[18:58:49] Difficulty: 3
UnityEngine.MonoBehaviour:print(Object)
[18:58:59] The Performance of the player is: 100 / 100
UnityEngine.MonoBehaviour:print(Object)
[18:58:59] Difficulty: 4
UnityEngine.MonoBehaviour:print(Object)
[18:59:32] The Performance of the player is: 100 / 100

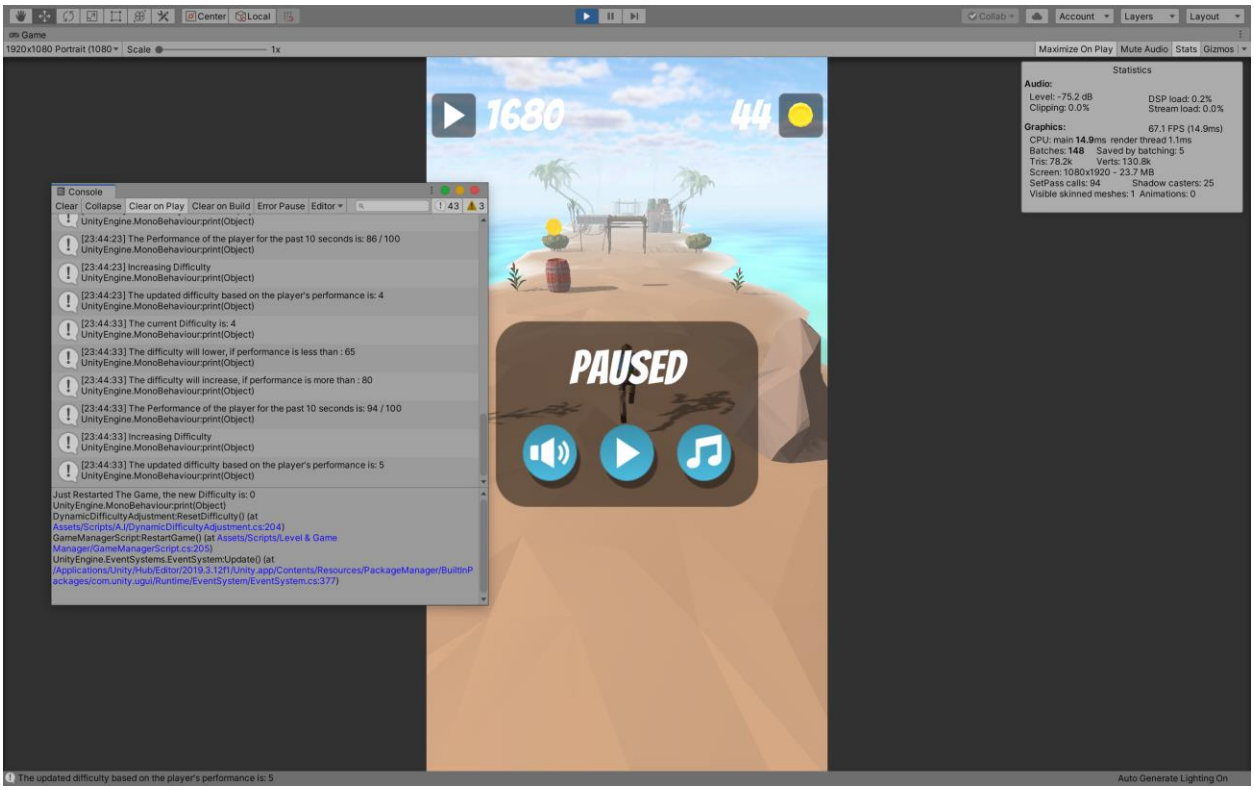

Console
Clear
Collapse
Clear on Play
Clear on Build
Error Pause
Editor
51
3
[18:58:59] Difficulty: 4
UnityEngine.MonoBehaviour:print(Object)
[18:59:32] The Performance of the player is: 100 / 100
UnityEngine.MonoBehaviour:print(Object)
[18:59:32] Difficulty: 5
UnityEngine.MonoBehaviour:print(Object)
[18:59:47] The Performance of the player is: 72 / 100
UnityEngine.MonoBehaviour:print(Object)
[18:59:47] Difficulty: 5
UnityEngine.MonoBehaviour:print(Object)
[18:59:57] The Performance of the player is: 70 / 100
UnityEngine.MonoBehaviour:print(Object)
[18:59:57] Difficulty: 4
UnityEngine.MonoBehaviour:print(Object)
[19:00:37] The Performance of the player is: 100 / 100
UnityEngine.MonoBehaviour:print(Object)
[19:00:37] Difficulty: 5
UnityEngine.MonoBehaviour:print(Object)
[19:00:47] The Performance of the player is: 95 / 100

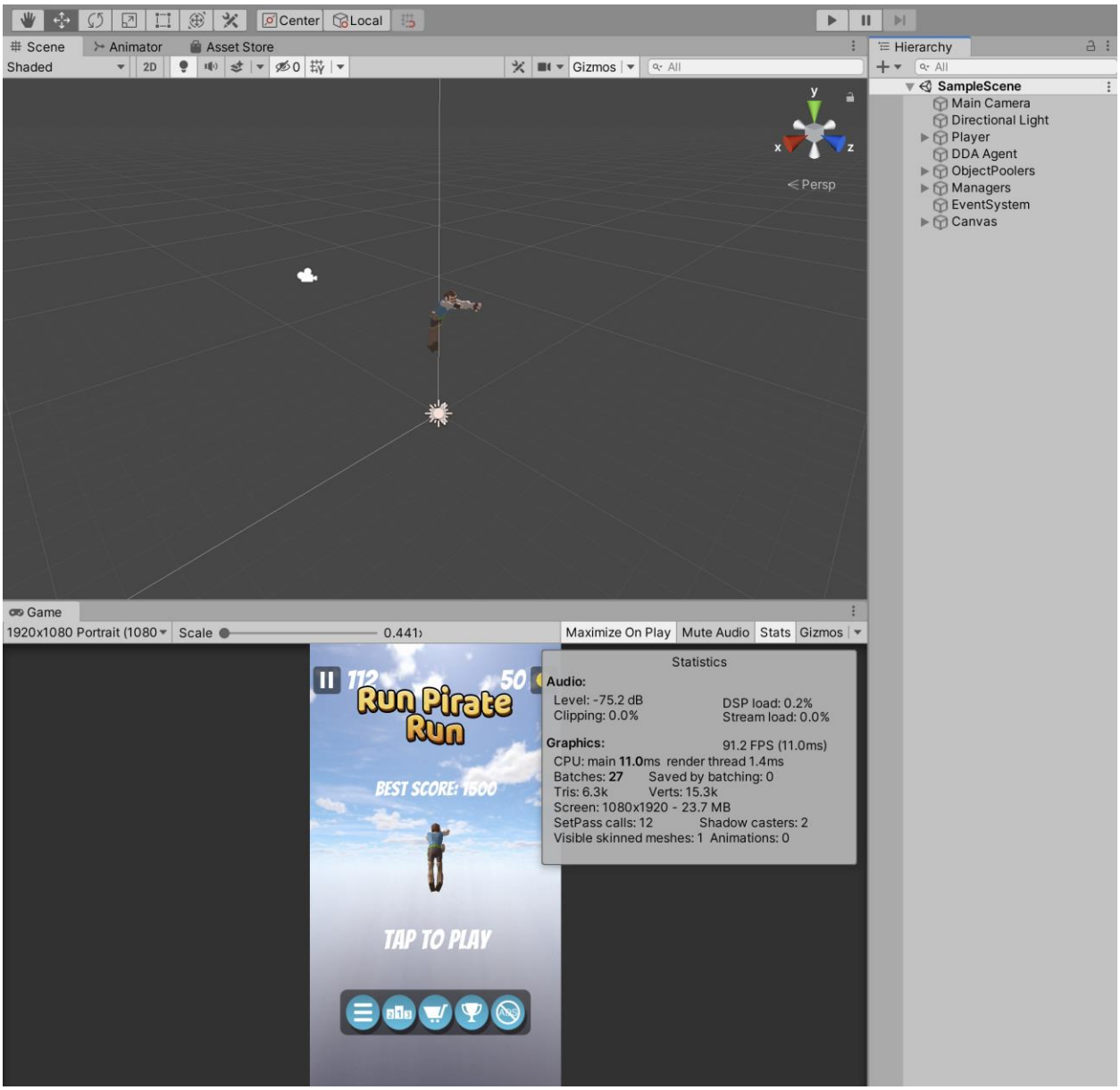

Console
Clear
Collapse
Clear on Play
Clear on Build
Error Pause
Editor
51
3
[19:00:47] The Performance of the player is: 95 / 100
UnityEngine.MonoBehaviour:print(Object)
[19:00:47] Difficulty: 6
UnityEngine.MonoBehaviour:print(Object)
[19:00:57] The Performance of the player is: 96 / 100
UnityEngine.MonoBehaviour:print(Object)
[19:00:57] Difficulty: 7
UnityEngine.MonoBehaviour:print(Object)
[19:01:07] The Performance of the player is: 100 / 100
UnityEngine.MonoBehaviour:print(Object)
[19:01:07] Difficulty: 8
UnityEngine.MonoBehaviour:print(Object)
[19:01:17] The Performance of the player is: 100 / 100
UnityEngine.MonoBehaviour:print(Object)
[19:01:17] Difficulty: 9
UnityEngine.MonoBehaviour:print(Object)
[19:01:27] The Performance of the player is: 96 / 100
UnityEngine.MonoBehaviour:print(Object)

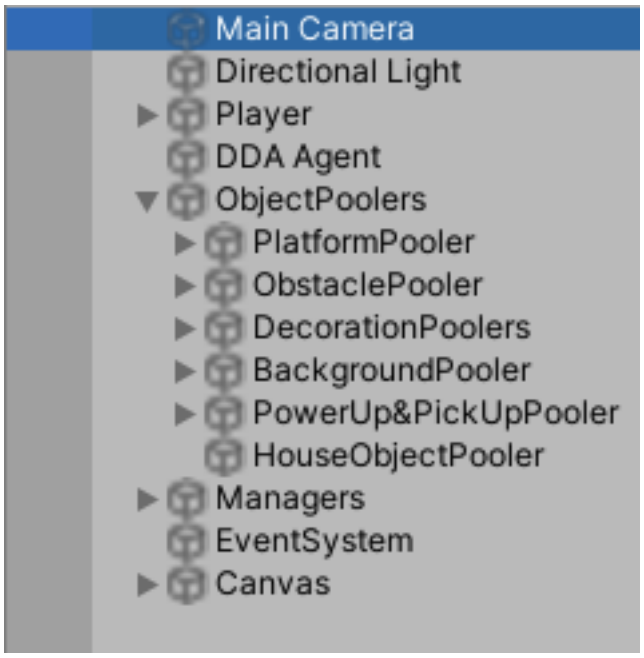

1680
44
PAUSED
Console
Clear
Collapse
Clear on Play
Clear on Build
Error Pause
Editor
[23:43:41] Just Restarted The Game, the new Difficulty is: 0
[23:43:53] The current Difficulty is: 0
[23:43:53] The difficulty will lower, if performance is less than : 65
[23:43:53] The difficulty will increase, if performance is more than : 80
[23:43:53] The Performance of the player for the past 10 seconds is: 90 / 100
[23:43:53] Increasing Difficulty
[23:43:53] The updated difficulty based on the player's performance is: 1
[23:44:03] The current Difficulty is: 1
[23:44:03] The difficulty will lower, if performance is less than : 65
[23:44:03] The difficulty will increase, if performance is more than : 80
Statistics
Audio:
Level: -75.2 dB
Clipping: 0.0%
DSP load: 0.2%
Stream load: 0.0%
Graphics:
87.5 FPS (11.4ms)
CPU: main 11.4ms render thread 1.1ms
Batches: 148 Saved by batching: 5
Tris: 78.2k Verts: 130.8k
Screen: 1080x1920 - 23.7 MB
SetPass calls: 94 Shadow casters: 25
Visible skinned meshes: 1 Animations: 0
The updated difficulty based on the player's performance is: 5
Auto Generate Lighting On

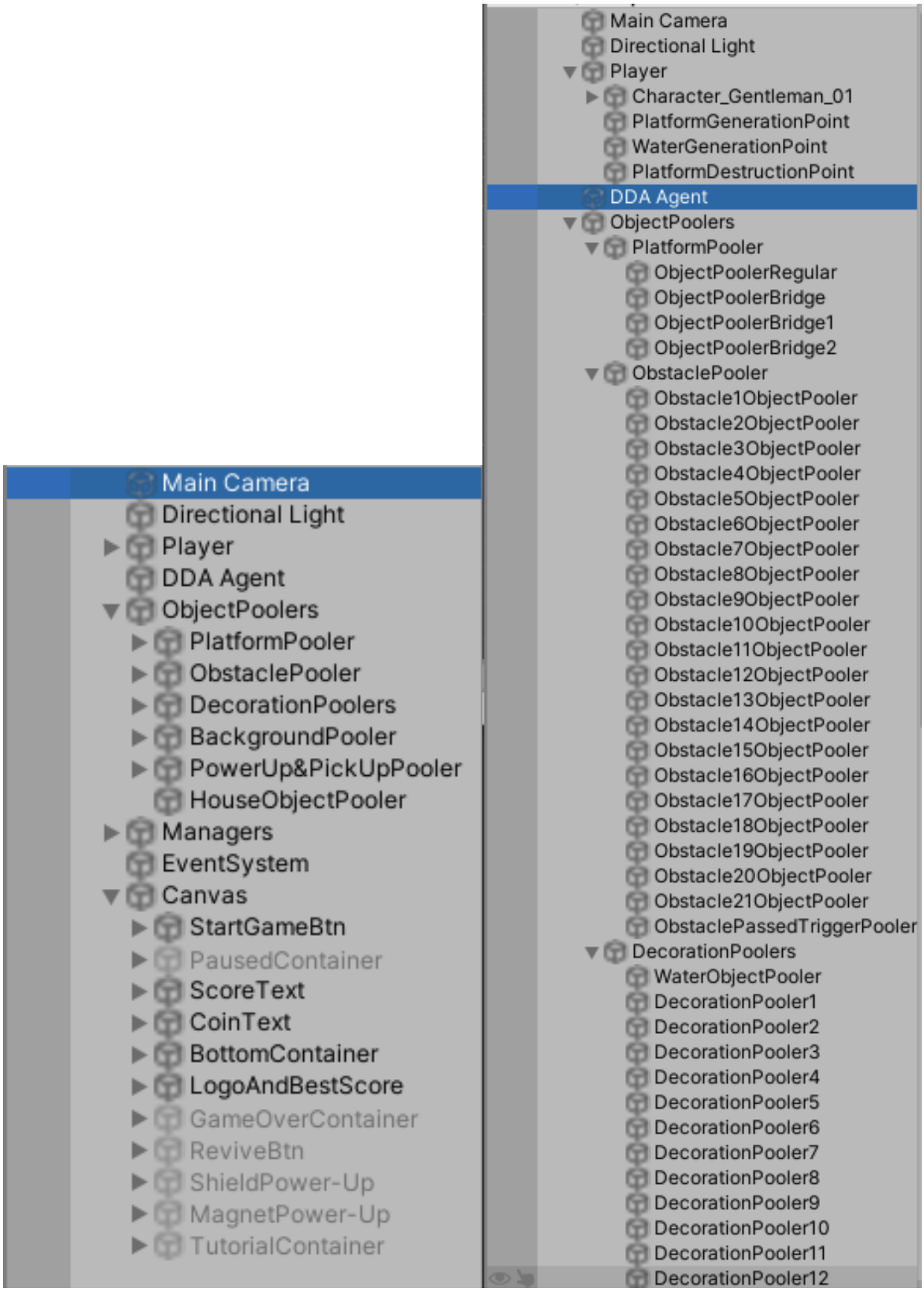

1680
44
PAUSED
Console
Clear
Collapse
Clear on Play
Clear on Build
Error Pause
Editor
[23:44:03] The difficulty will increase, if performance is more than : 80
[23:44:03] The Performance of the player for the past 10 seconds is: 94 / 100
[23:44:03] Increasing Difficulty
[23:44:03] The updated difficulty based on the player's performance is: 2
[23:44:13] The current Difficulty is: 2
[23:44:13] The difficulty will lower, if performance is less than : 65
[23:44:13] The difficulty will increase, if performance is more than : 80
[23:44:13] The Performance of the player for the past 10 seconds is: 92 / 100
[23:44:13] Increasing Difficulty
Statistics
Audio:
Level: -75.2 dB
Clipping: 0.0%
DSP load: 0.2%
Stream load: 0.0%
Graphics:
87.5 FPS (11.4ms)
CPU: main 11.4ms render thread 1.0ms
Batches: 148 Saved by batching: 5
Tris: 78.2k Verts: 130.8k
Screen: 1080x1920 - 23.7 MB
SetPass calls: 94 Shadow casters: 25
Visible skinned meshes: 1 Animations: 0
The updated difficulty based on the player's performance is: 5
Auto Generate Lighting On


1680
44
PAUSED
Console
[23:44:13] The Performance of the player for the past 10 seconds is: 92 / 100
[23:44:13] Increasing Difficulty
[23:44:13] The updated difficulty based on the player's performance is: 3
[23:44:23] The current Difficulty is: 3
[23:44:23] The difficulty will lower, if performance is less than : 65
[23:44:23] The difficulty will increase, if performance is more than : 80
[23:44:23] The Performance of the player for the past 10 seconds is: 86 / 100
[23:44:23] Increasing Difficulty
[23:44:23] The updated difficulty based on the player's performance is: 4
[23:44:33] The current Difficulty is: 4
Just Restarted The Game, the new Difficulty is: 0
Statistics
Audio:
Graphics:
The updated difficulty based on the player's performance is: 5

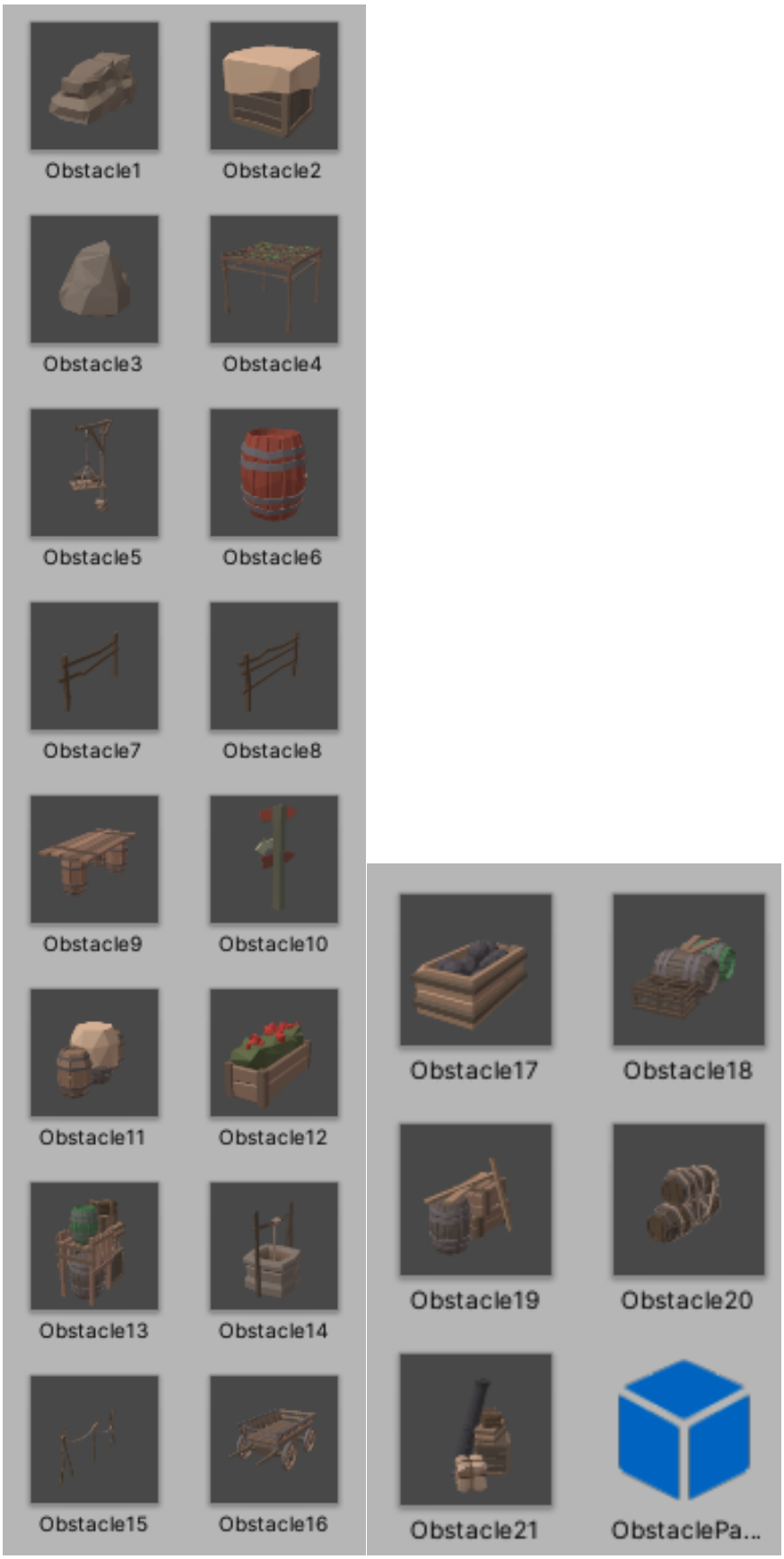

1680
44
PAUSED
Console
[23:44:23] The Performance of the player for the past 10 seconds is: 86 / 100
[23:44:23] Increasing Difficulty
[23:44:23] The updated difficulty based on the player's performance is: 4
[23:44:33] The current Difficulty is: 4
[23:44:33] The difficulty will lower, if performance is less than : 65
[23:44:33] The difficulty will increase, if performance is more than : 80
[23:44:33] The Performance of the player for the past 10 seconds is: 94 / 100
[23:44:33] Increasing Difficulty
[23:44:33] The updated difficulty based on the player's performance is: 5
Just Restarted The Game, the new Difficulty is: 0
Statistics
Audio:
Graphics:
The updated difficulty based on the player's performance is: 5

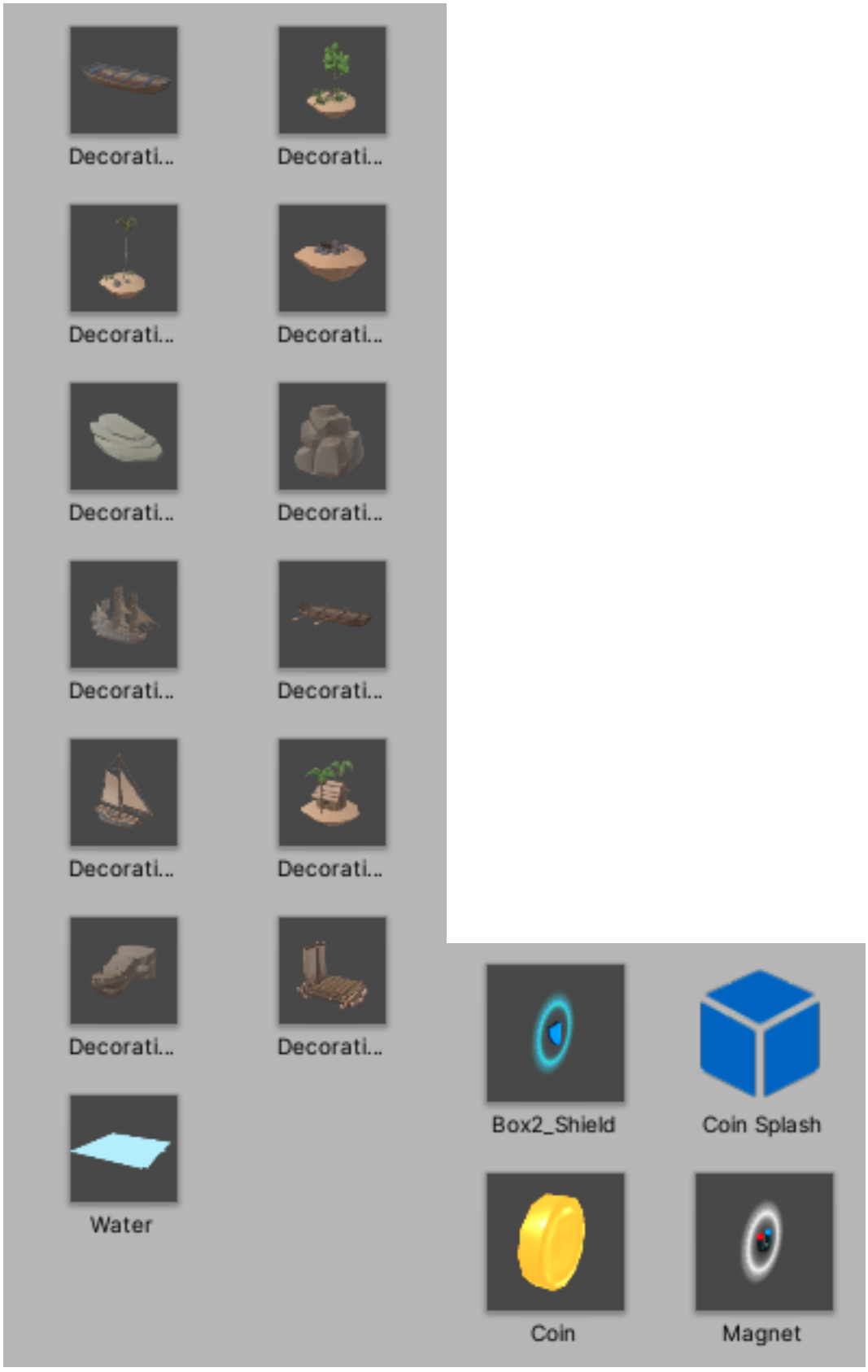

Center
Local
Scene
Animator
Asset Store
Shaded
2D
Gizmos
All
Persp
Hierarchy
SampleScene
Main Camera
Directional Light
Player
DDA Agent
ObjectPoolers
Managers
EventSystem
Canvas
Game
1920x1080 Portrait (1080
Scale
0.441
Maximize On Play
Mute Audio
Stats
Gizmos
Statistics
Audio:
Level: -75.2 dB
Clipping: 0.0%
DSP load: 0.2%
Stream load: 0.0%
Graphics:
91.2 FPS (11.0ms)
CPU: main 11.0ms render thread 1.4ms
Batches: 27
Saved by batching: 0
Tris: 6.3k
Verts: 15.3k
Screen: 1080x1920 - 23.7 MB
SetPass calls: 12
Shadow casters: 2
Visible skinned meshes: 1
Animations: 0
112
50
Run Pirate Run
BEST SCORE: 1800
TAP TO PLAY

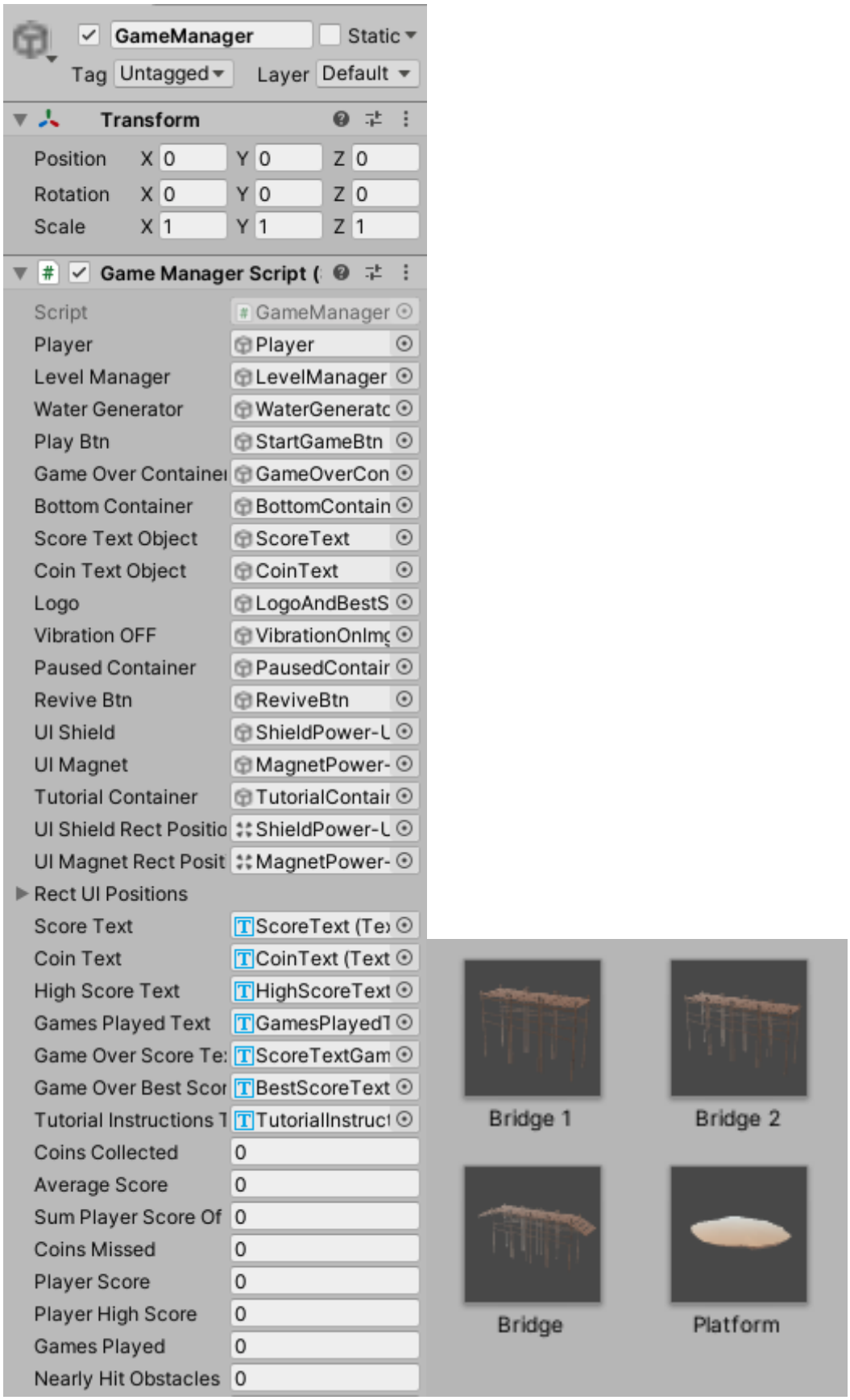

Main Camera
Directional Light
Player
DDA Agent
ObjectPoolers
PlatformPooler
ObstaclePooler
DecorationPoolers
BackgroundPooler
PowerUp&PickUpPooler
HouseObjectPooler
Managers
EventSystem
Canvas

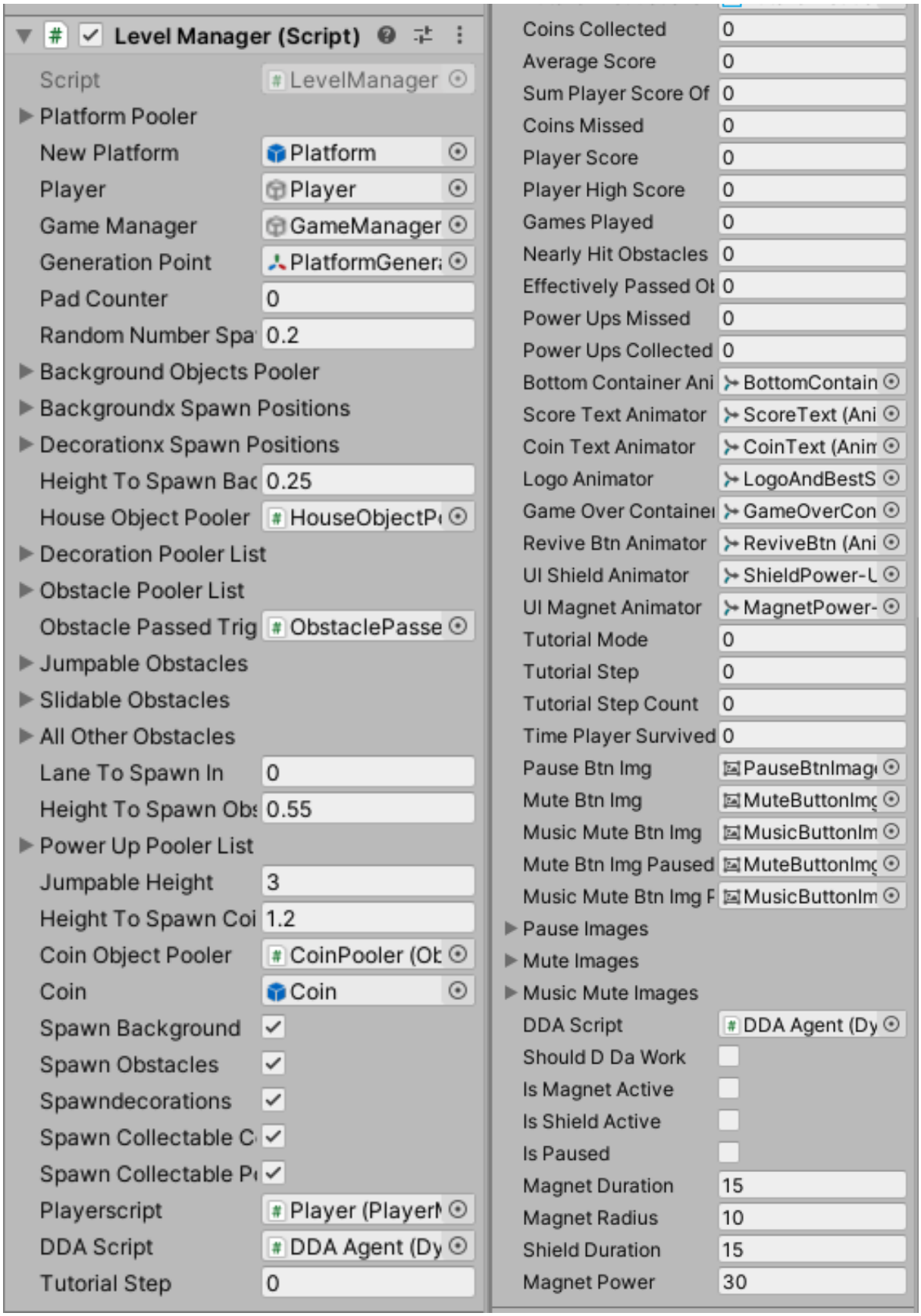

Main Camera
Directional Light
Player
DDA Agent
ObjectPoolers
PlatformPooler
ObstaclePooler
DecorationPoolers
BackgroundPooler
PowerUp&PickUpPooler
HouseObjectPooler
Managers
EventSystem
Canvas
StartGameBtn
PausedContainer
ScoreText
CoinText
BottomContainer
LogoAndBestScore
GameOverContainer
ReviveBtn
ShieldPower-Up
MagnetPower-Up
TutorialContainer
Main Camera
Directional Light
Player
Character_Gentleman_01
PlatformGenerationPoint
WaterGenerationPoint
PlatformDestructionPoint
DDA Agent
ObjectPoolers
PlatformPooler
ObjectPoolerRegular
ObjectPoolerBridge
ObjectPoolerBridge1
ObjectPoolerBridge2
ObstaclePooler
Obstacle1ObjectPooler
Obstacle2ObjectPooler
Obstacle3ObjectPooler
Obstacle4ObjectPooler
Obstacle5ObjectPooler
Obstacle6ObjectPooler
Obstacle7ObjectPooler
Obstacle8ObjectPooler
Obstacle9ObjectPooler
Obstacle10ObjectPooler
Obstacle11ObjectPooler
Obstacle12ObjectPooler
Obstacle13ObjectPooler
Obstacle14ObjectPooler
Obstacle15ObjectPooler
Obstacle16ObjectPooler
Obstacle17ObjectPooler
Obstacle18ObjectPooler
Obstacle19ObjectPooler
Obstacle20ObjectPooler
Obstacle21ObjectPooler
ObstaclePassedTriggerPooler
DecorationPoolers
WaterObjectPooler
DecorationPooler1
DecorationPooler2
DecorationPooler3
DecorationPooler4
DecorationPooler5
DecorationPooler6
DecorationPooler7
DecorationPooler8
DecorationPooler9
DecorationPooler10
DecorationPooler11
DecorationPooler12

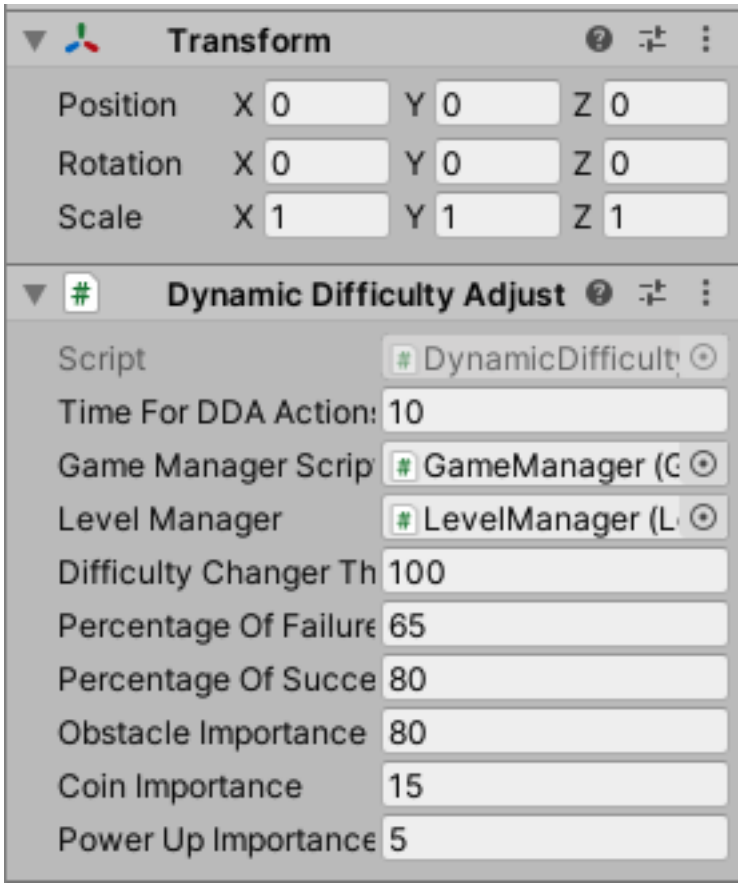

BackgroundPooler
BackgroundItemPooler1
BackgroundItemPooler2
BackgroundItemPooler3
BackgroundItemPooler4
BackgroundItemPooler5
BackgroundItemPooler6
BackgroundItemPooler7
BackgroundItemPooler8
BackgroundItemPooler9
BackgroundItemPooler10
BackgroundItemPooler11
BackgroundItemPooler12
BackgroundItemPooler13
BackgroundItemPooler14
BackgroundItemPooler15
BackgroundItemPooler16
PowerUp&PickUpPooler
CoinPooler
SplashPooler
ShieldPowerUpPooler
MagnetPowerUpPooler
HouseObjectPooler
Managers
LevelManager
GameManager
WaterGenerator
SoundManager
MusicManager
EventSystem
Canvas
StartGameBtn
PausedContainer
ScoreText
CoinText
BottomContainer
LogoAndBestScore
GameOverContainer
ReviveBtn
ShieldPower-Up
MagnetPower-Up
TutorialContainer
Backgro...
House

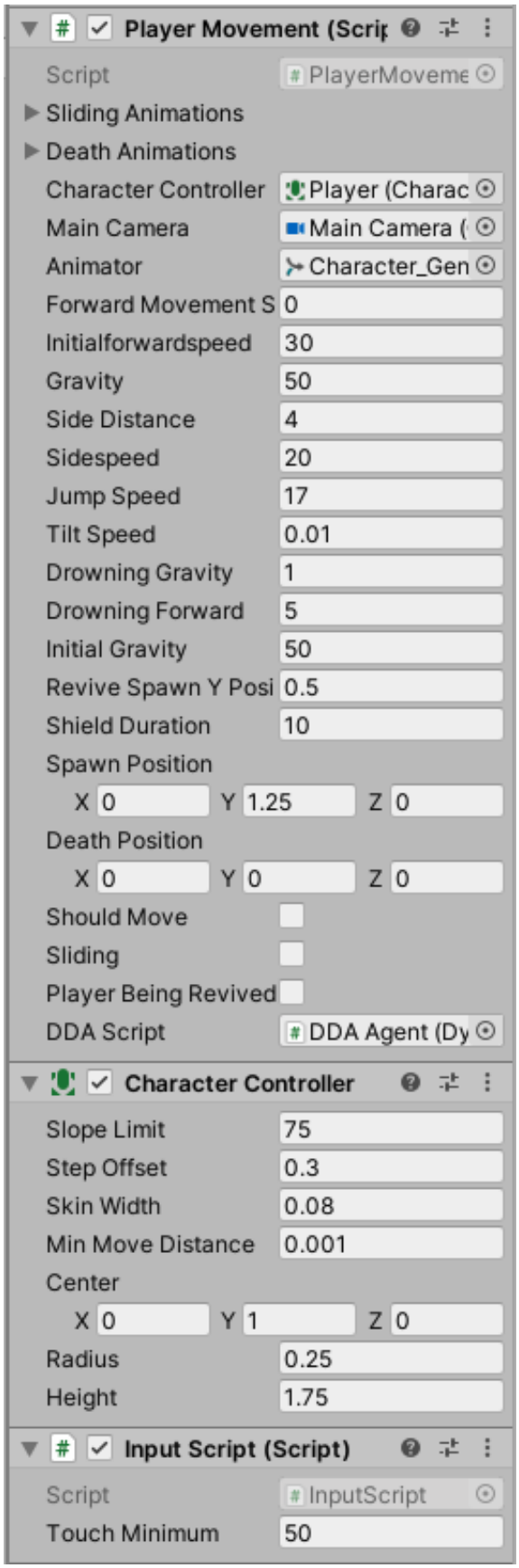

Obstacle1
Obstacle2
Obstacle3
Obstacle4
Obstacle5
Obstacle6
Obstacle7
Obstacle8
Obstacle9
Obstacle10
Obstacle11
Obstacle12
Obstacle13
Obstacle14
Obstacle15
Obstacle16
Obstacle17
Obstacle18
Obstacle19
Obstacle20
Obstacle21
ObstaclePa...

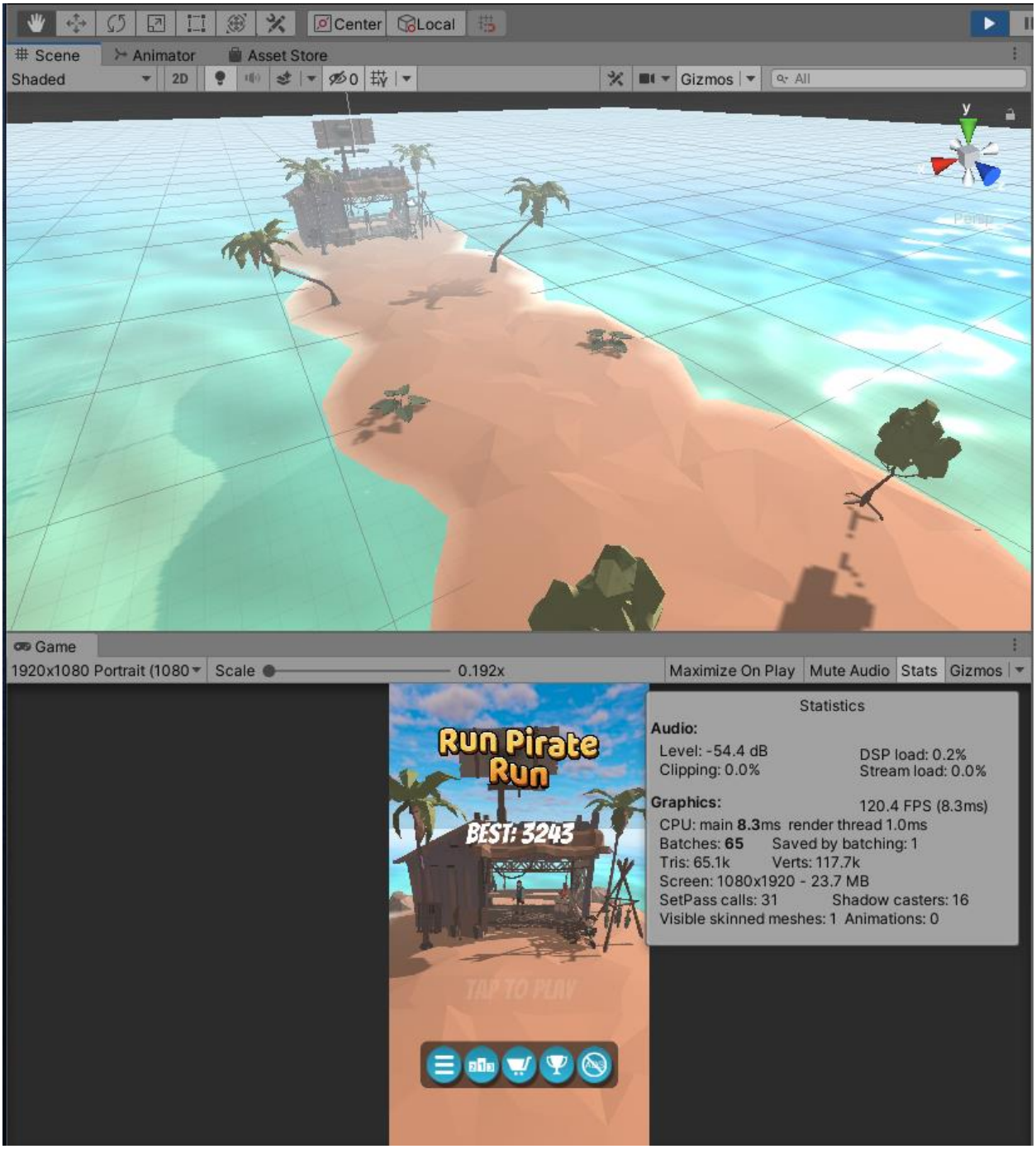

Decorati...
Decorati...
Decorati...
Decorati...
Decorati...
Decorati...
Decorati...
Decorati...
Decorati...
Decorati...
Decorati...
Decorati...
Water
Box2_Shield
Coin Splash
Coin
Magnet

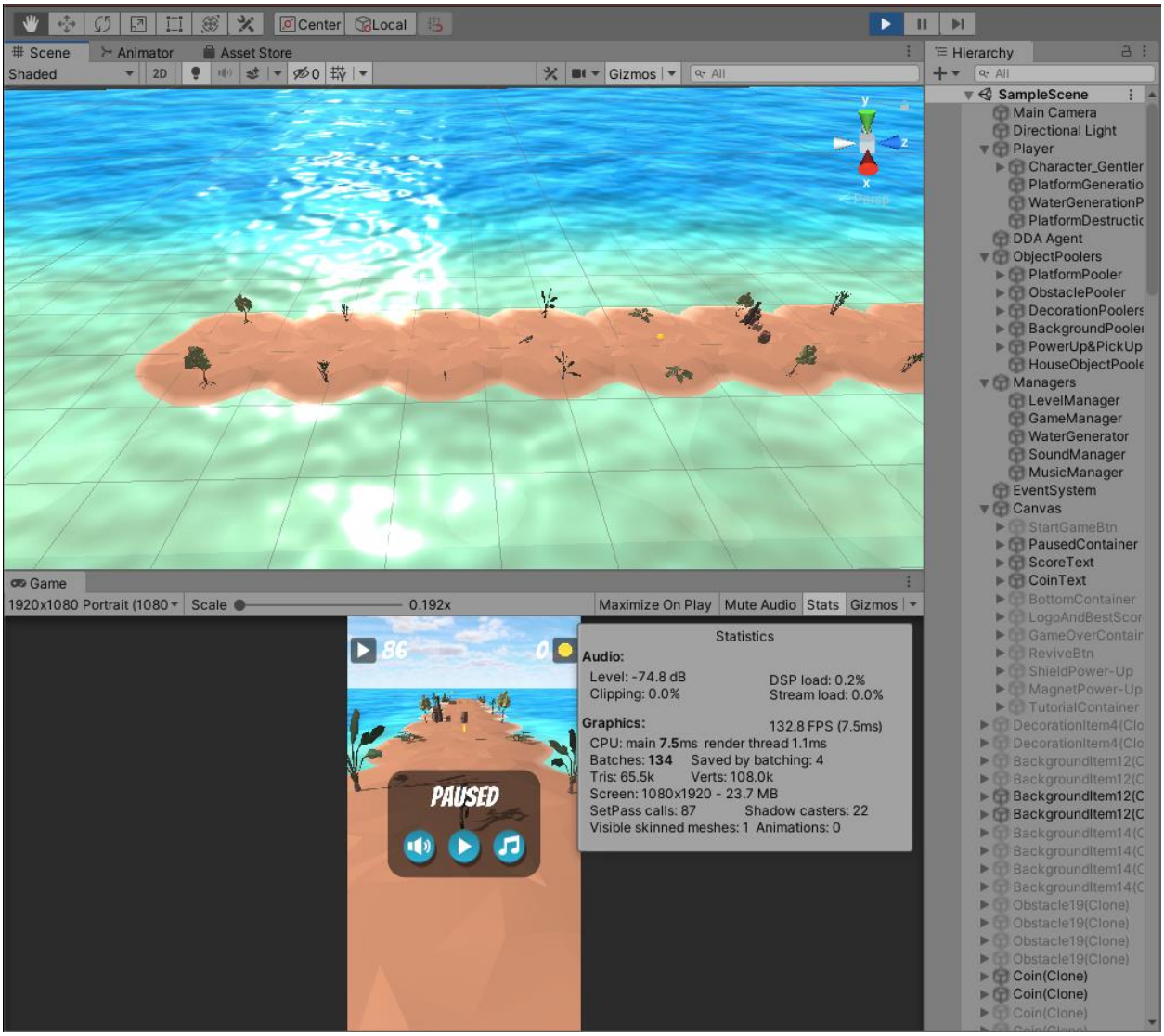

GameManager
Static
Tag Untagged
Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Game Manager Script (
Script GameManager
Player Player
Level Manager LevelManager
Water Generator WaterGeneratc
Play Btn StartGameBtn
Game Over Containei GameOverCon
Bottom Container BottomContain
Score Text Object ScoreText
Coin Text Object CoinText
Logo LogoAndBestS
Vibration OFF VibrationOnIm
Paused Container PausedContair
Revive Btn ReviveBtn
UI Shield ShieldPower-L
UI Magnet MagnetPower-
Tutorial Container TutorialContair
UI Shield Rect Positio ShieldPower-L
UI Magnet Rect Posit MagnetPower-
Rect UI Positions
Score Text ScoreText (Te
Coin Text CoinText (Text
High Score Text HighScoreText
Games Played Text GamesPlayedT
Game Over Score Te: ScoreTextGam
Game Over Best Scor BestScoreText
Tutorial Instructions T TutorialInstruct
Coins Collected 0
Average Score 0
Sum Player Score Of 0
Coins Missed 0
Player Score 0
Player High Score 0
Games Played 0
Nearly Hit Obstacles 0
Bridge 1
Bridge 2
Bridge
Platform

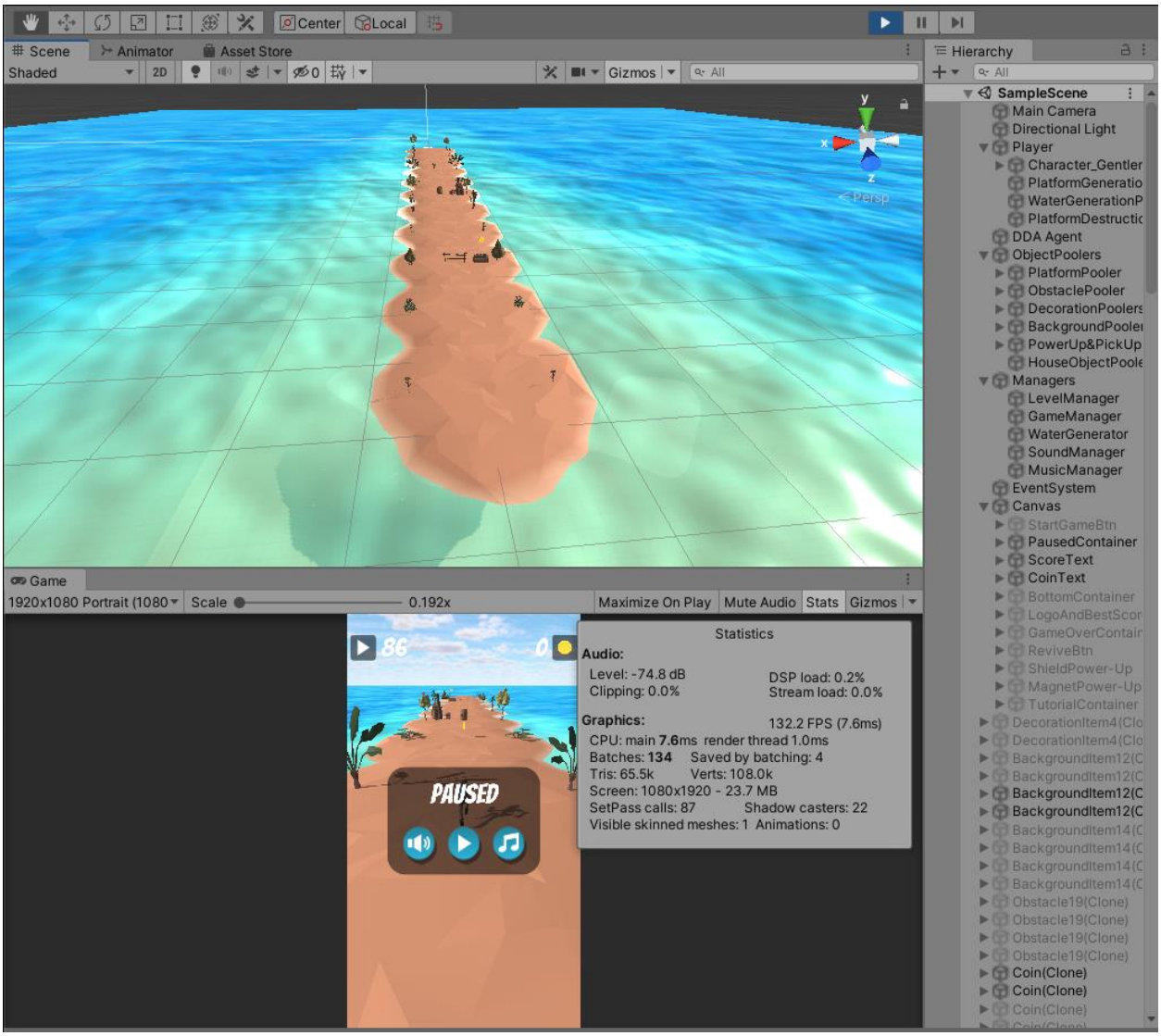

Level Manager (Script)
Script
LevelManager
Platform Pooler
New Platform
Platform
Player
Player
Game Manager
GameManager
Generation Point
PlatformGener
Pad Counter
0
Random Number Spa
0.2
Background Objects Pooler
Backgroundx Spawn Positions
Decorationx Spawn Positions
Height To Spawn Bac
0.25
House Object Pooler
HouseObjectP
Decoration Pooler List
Obstacle Pooler List
Obstacle Passed Trig
ObstaclePasse
Jumpable Obstacles
Slidable Obstacles
All Other Obstacles
Lane To Spawn In
0
Height To Spawn Obs
0.55
Power Up Pooler List
Jumpable Height
3
Height To Spawn Coi
1.2
Coin Object Pooler
CoinPooler (Ob
Coin
Coin
Spawn Background
Spawn Obstacles
Spawndecorations
Spawn Collectable C
Spawn Collectable P
Playerscript
Player (Player
DDA Script
DDA Agent (Dy
Tutorial Step
0
Coins Collected
0
Average Score
0
Sum Player Score Of
0
Coins Missed
0
Player Score
0
Player High Score
0
Games Played
0
Nearly Hit Obstacles
0
Effectively Passed Ob
0
Power Ups Missed
0
Power Ups Collected
0
Bottom Container Ani
BottomContain
Score Text Animator
ScoreText (Ani
Coin Text Animator
CoinText (Anim
Logo Animator
LogoAndBestS
Game Over Containe
GameOverCon
Revive Btn Animator
ReviveBtn (Ani
UI Shield Animator
ShieldPower-L
UI Magnet Animator
MagnetPower-
Tutorial Mode
0
Tutorial Step
0
Tutorial Step Count
0
Time Player Survived
0
Pause Btn Img
PauseBtnImag
Mute Btn Img
MuteButtonIm
Music Mute Btn Img
MusicButtonIm
Mute Btn Img Paused
MuteButtonIm
Music Mute Btn Img F
MusicButtonIm
Pause Images
Mute Images
Music Mute Images
DDA Script
DDA Agent (Dy
Should D Da Work
Is Magnet Active
Is Shield Active
Is Paused
Magnet Duration
15
Magnet Radius
10
Shield Duration
15
Magnet Power
30

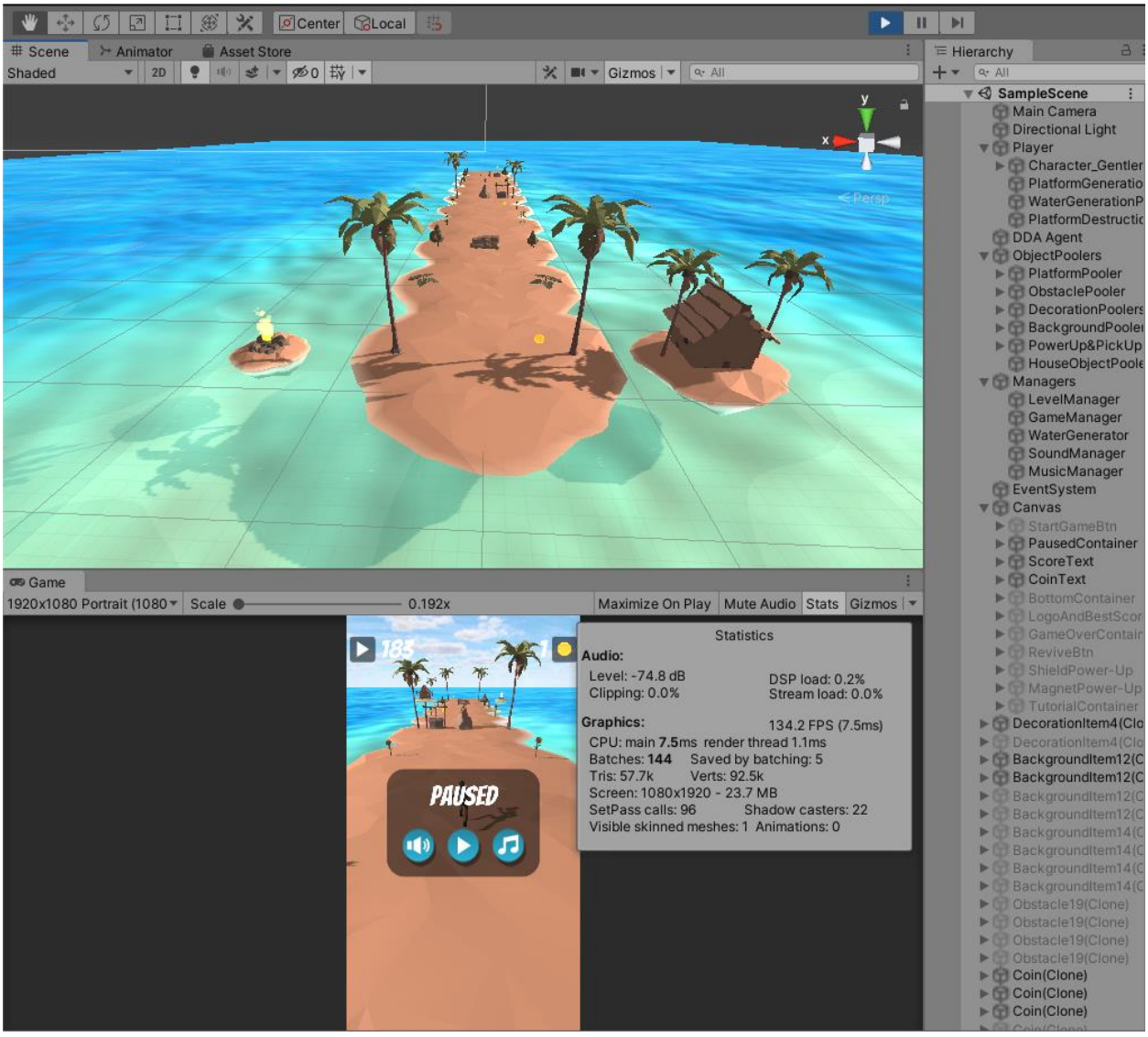

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Dynamic Difficulty Adjust
Script DynamicDifficult
Time For DDA Action: 10
Game Manager Scrip GameManager (G
Level Manager LevelManager (L
Difficulty Changer Th 100
Percentage Of Failure 65
Percentage Of Succe 80
Obstacle Importance 80
Coin Importance 15
Power Up Importance 5

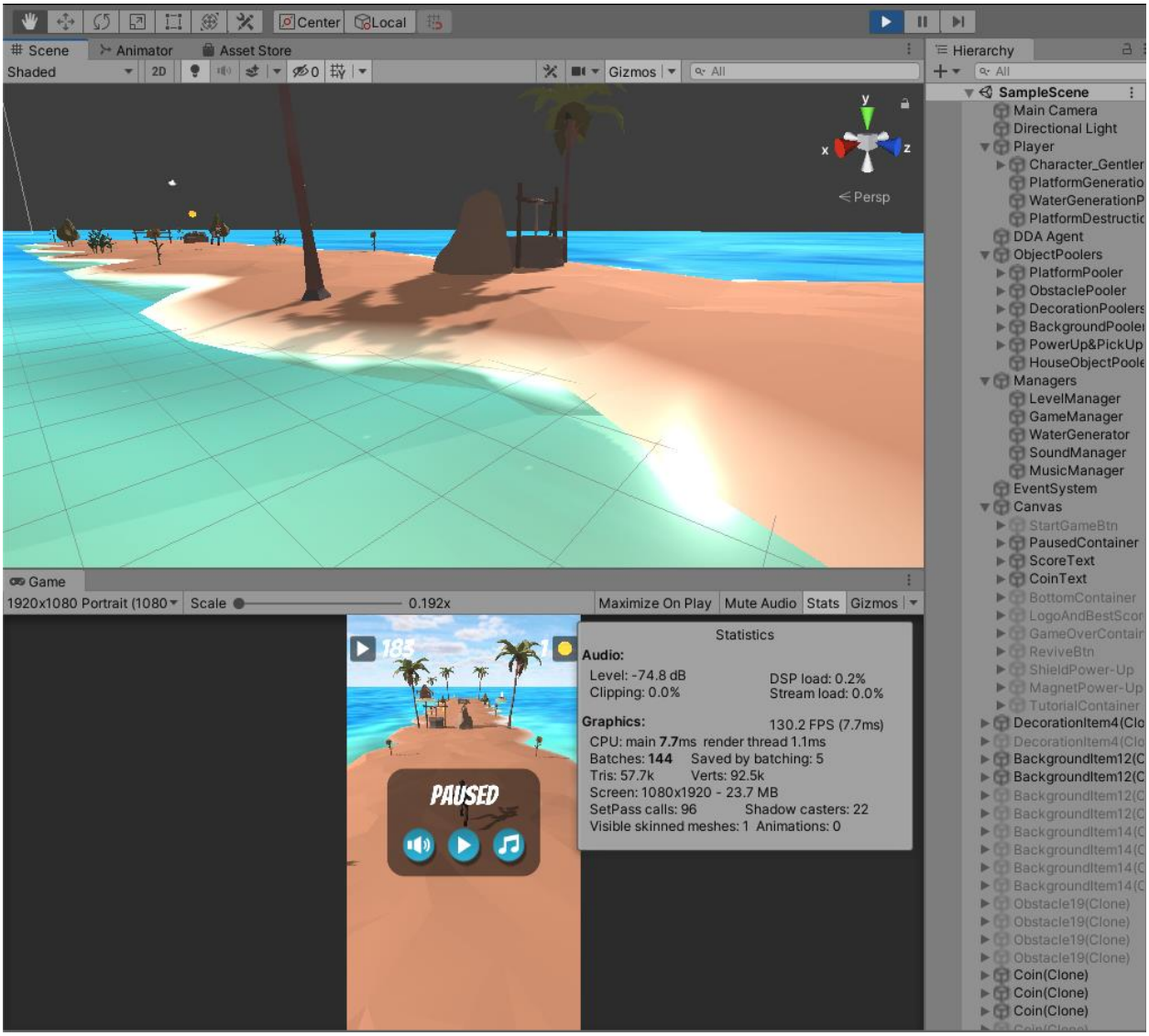

Player Movement (Scrip
Script
PlayerMoveme
Sliding Animations
Death Animations
Character Controller
Player (Charac
Main Camera
Main Camera (
Animator
Character_Gen
Forward Movement S
0
Initialforwardspeed
30
Gravity
50
Side Distance
4
Sidespeed
20
Jump Speed
17
Tilt Speed
0.01
Drowning Gravity
1
Drowning Forward
5
Initial Gravity
50
Revive Spawn Y Posi
0.5
Shield Duration
10
Spawn Position
X 0
Y 1.25
Z 0
Death Position
X 0
Y 0
Z 0
Should Move
Sliding
Player Being Revived
DDA Script
DDA Agent (Dy
Character Controller
Slope Limit
75
Step Offset
0.3
Skin Width
0.08
Min Move Distance
0.001
Center
X 0
Y 1
Z 0
Radius
0.25
Height
1.75
Input Script (Script)
Script
InputScript
Touch Minimum
50

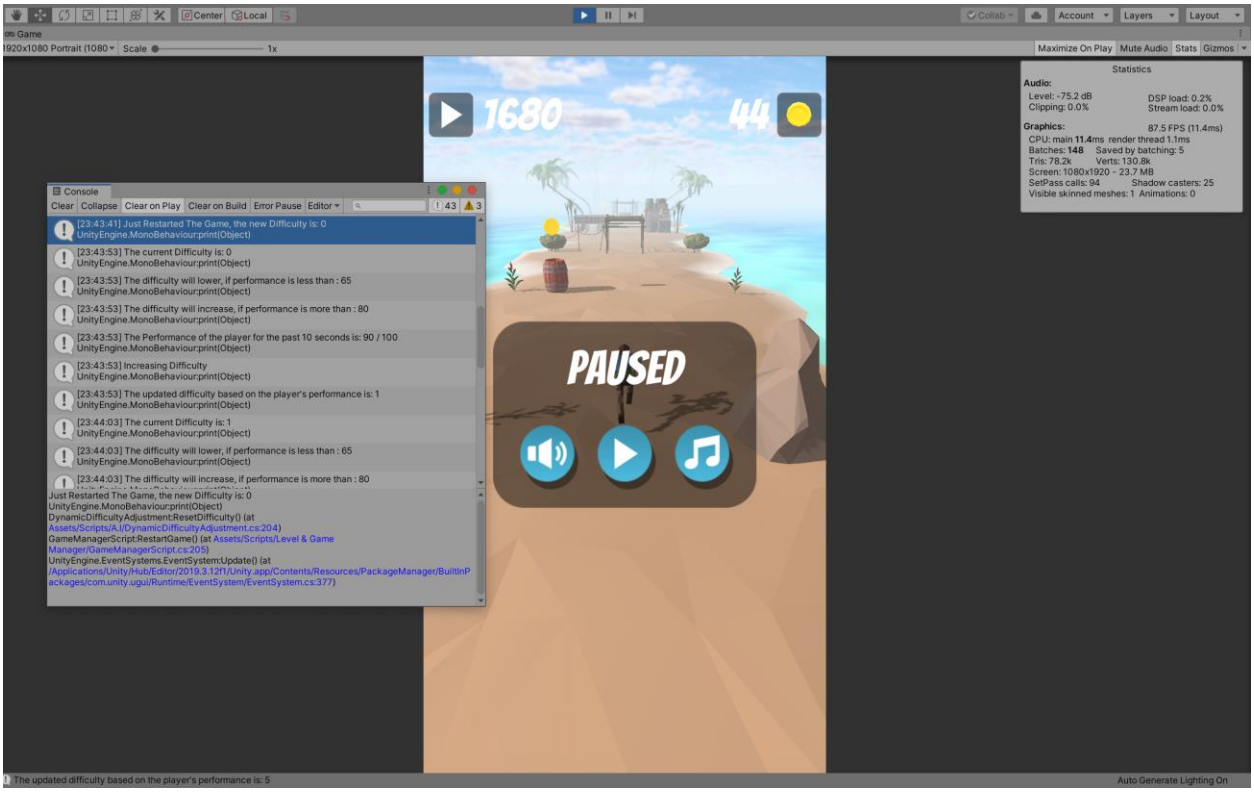

Center
Local
Scene
Animator
Asset Store
Shaded
2D
Gizmos
All
Persp
Game
1920x1080 Portrait (1080
Scale
0.192x
Maximize On Play
Mute Audio
Stats
Gizmos
Run Pirate Run
BEST: 3243
TAP TO PLAY
Statistics
Audio:
Level: -54.4 dB
Clipping: 0.0%
DSP load: 0.2%
Stream load: 0.0%
Graphics:
120.4 FPS (8.3ms)
CPU: main 8.3ms render thread 1.0ms
Batches: 65
Saved by batching: 1
Tris: 65.1k
Verts: 117.7k
Screen: 1080x1920 - 23.7 MB
SetPass calls: 31
Shadow casters: 16
Visible skinned meshes: 1
Animations: 0

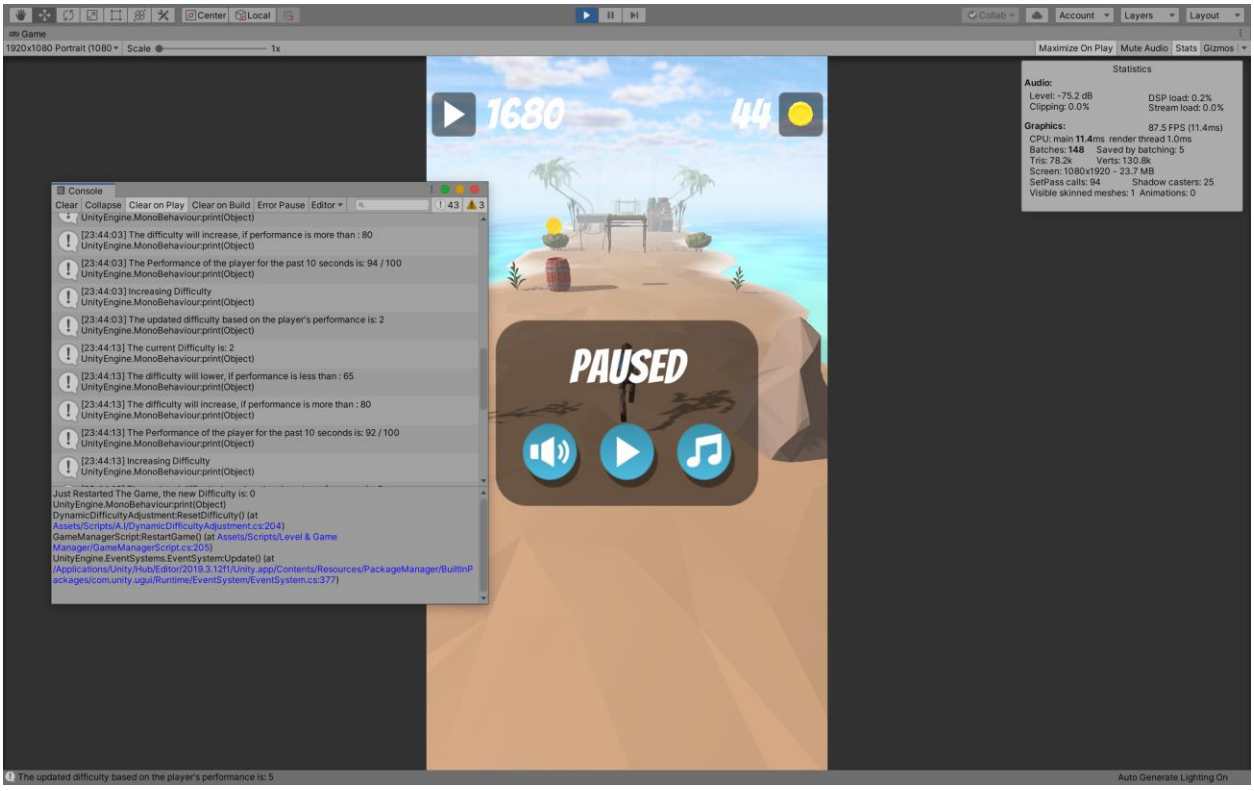

Center
Local
Scene
Animator
Asset Store
Shaded
2D
Gizmos
Hierarchy
SampleScene
Main Camera
Directional Light
Player
DDA Agent
ObjectPoolers
Managers
LevelManager
GameManager
WaterGenerator
SoundManager
MusicManager
EventSystem
Canvas
StartGameBtn
PausedContainer
ScoreText
CoinText
Game
1920x1080 Portrait (1080
Scale
0.192x
Maximize On Play
Mute Audio
Stats
Gizmos
Statistics
Audio:
Level: -74.8 dB
Clipping: 0.0%
DSP load: 0.2%
Stream load: 0.0%
Graphics:
132.8 FPS (7.5ms)
CPU: main 7.5ms render thread 1.1ms
Batches: 134 Saved by batching: 4
Tris: 65.5k Verts: 108.0k
Screen: 1080x1920 - 23.7 MB
SetPass calls: 87 Shadow casters: 22
Visible skinned meshes: 1 Animations: 0
PAUSED

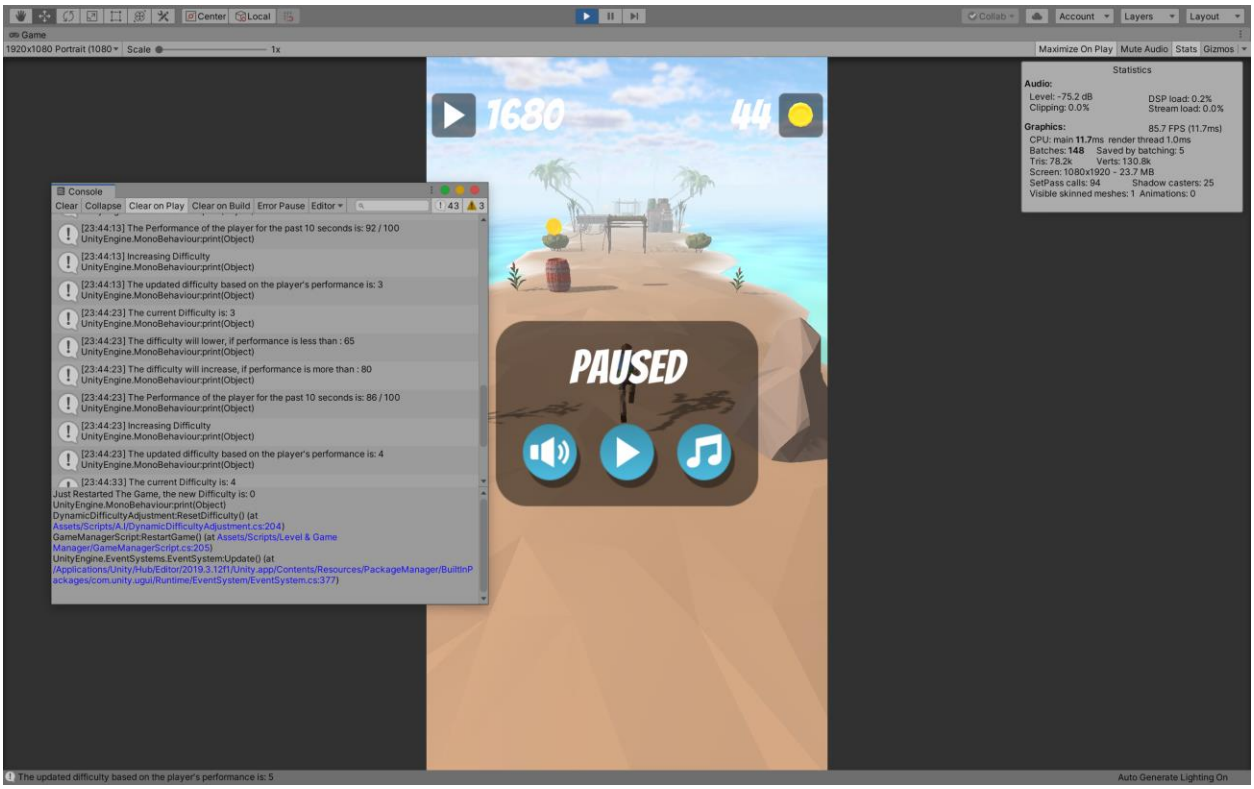

Center
Local
Scene
Animator
Asset Store
Shaded
2D
Gizmos
Persp
Hierarchy
SampleScene
Main Camera
Directional Light
Player
Character_Gentler
PlatformGeneratio
WaterGenerationP
PlatformDestructio
DDA Agent
ObjectPoolers
PlatformPooler
ObstaclePooler
DecorationPoolers
BackgroundPoole
PowerUp&PickUp
HouseObjectPoole
Managers
LevelManager
GameManager
WaterGenerator
SoundManager
MusicManager
EventSystem
Canvas
StartGameBtn
PausedContainer
ScoreText
CoinText
BottomContainer
LogoAndBestScor
GameOverContair
ReviveBtn
ShieldPower-Up
MagnetPower-Up
TutorialContainer
Game
1920x1080 Portrait (1080
Scale
0.192x
Maximize On Play
Mute Audio
Stats
Gizmos
PAUSED
Statistics
Audio:
Level: -74.8 dB
Clipping: 0.0%
DSP load: 0.2%
Stream load: 0.0%
Graphics:
132.2 FPS (7.6ms)
CPU: main 7.6ms render thread 1.0ms
Batches: 134 Saved by batching: 4
Tris: 65.5k Verts: 108.0k
Screen: 1080x1920 - 23.7 MB
SetPass calls: 87 Shadow casters: 22
Visible skinned meshes: 1 Animations: 0
Obstacle19(Clone)
Coin(Clone)

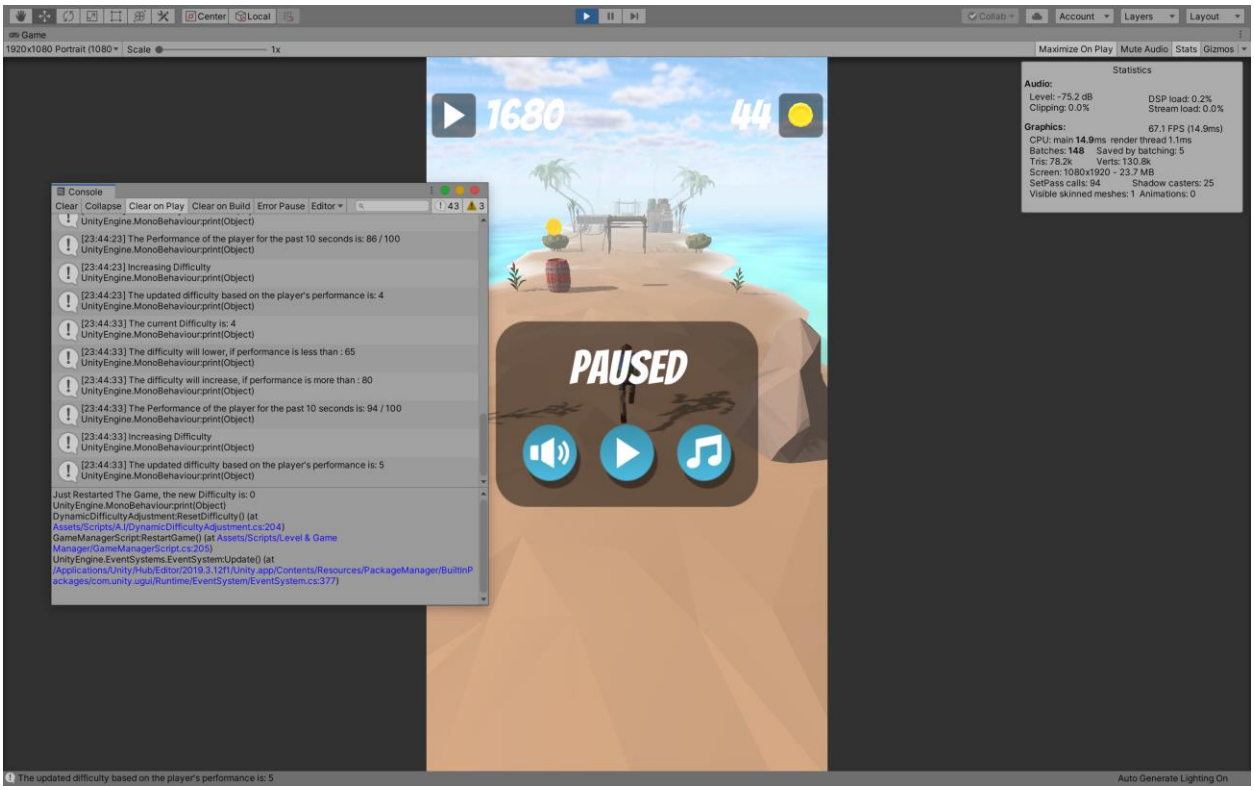

Center
Local
Scene
Animator
Asset Store
Shaded
2D
Gizmos
Persp
Hierarchy
SampleScene
Main Camera
Directional Light
Player
Character_Gentler
PlatformGeneratio
WaterGenerationP
PlatformDestructio
DDA Agent
ObjectPoolers
PlatformPooler
ObstaclePooler
DecorationPoolers
BackgroundPooler
PowerUp&PickUp
HouseObjectPoole
Managers
LevelManager
GameManager
WaterGenerator
SoundManager
MusicManager
EventSystem
Canvas
StartGameBtn
PausedContainer
ScoreText
CoinText
BottomContainer
LogoAndBestScor
GameOverContair
ReviveBtn
ShieldPower-Up
MagnetPower-Up
TutorialContainer
DecorationItem4(Clo
BackgroundItem12(C
BackgroundItem12(C
Obstacle19(Clone)
Obstacle19(Clone)
Obstacle19(Clone)
Obstacle19(Clone)
Coin(Clone)
Coin(Clone)
Coin(Clone)
Game
1920x1080 Portrait (1080
Scale
0.192x
Maximize On Play
Mute Audio
Stats
Gizmos
PAUSED
Statistics
Audio:
Level: -74.8 dB
Clipping: 0.0%
DSP load: 0.2%
Stream load: 0.0%
Graphics:
134.2 FPS (7.5ms)
CPU: main 7.5ms render thread 1.1ms
Batches: 144 Saved by batching: 5
Tris: 57.7k Verts: 92.5k
Screen: 1080x1920 - 23.7 MB
SetPass calls: 96 Shadow casters: 22
Visible skinned meshes: 1 Animations: 0


Center
Local
Scene
Animator
Asset Store
Shaded
2D
Gizmos
Persp
Hierarchy
SampleScene
Main Camera
Directional Light
Player
Character_Gentler
PlatformGeneratio
WaterGenerationP
PlatformDestructic
DDA Agent
ObjectPoolers
PlatformPooler
ObstaclePooler
DecorationPoolers
BackgroundPooler
PowerUp&PickUp
HouseObjectPoole
Managers
LevelManager
GameManager
WaterGenerator
SoundManager
MusicManager
EventSystem
Canvas
StartGameBtn
PausedContainer
ScoreText
CoinText
BottomContainer
LogoAndBestScor
GameOverContair
ReviveBtn
ShieldPower-Up
MagnetPower-Up
TutorialContainer
DecorationItem4(Clo
BackgroundItem12(C
BackgroundItem12(C
Obstacle19(Clone)
Obstacle19(Clone)
Obstacle19(Clone)
Obstacle19(Clone)
Coin(Clone)
Coin(Clone)
Coin(Clone)
Game
1920x1080 Portrait (1080
Scale
0.192x
Maximize On Play
Mute Audio
Stats
Gizmos
PAUSED
Statistics
Audio:
Level: -74.8 dB
Clipping: 0.0%
DSP load: 0.2%
Stream load: 0.0%
Graphics:
130.2 FPS (7.7ms)
CPU: main 7.7ms render thread 1.1ms
Batches: 144 Saved by batching: 5
Tris: 57.7k Verts: 92.5k
Screen: 1080x1920 - 23.7 MB
SetPass calls: 96 Shadow casters: 22
Visible skinned meshes: 1 Animations: 0


1680
44
PAUSED
Console
Clear
Collapse
Clear on Play
Clear on Build
Error Pause
Editor
[23:43:41] Just Restarted The Game, the new Difficulty is: 0
[23:43:53] The current Difficulty is: 0
[23:43:53] The difficulty will lower, if performance is less than : 65
[23:43:53] The difficulty will increase, if performance is more than : 80
[23:43:53] The Performance of the player for the past 10 seconds is: 90 / 100
[23:43:53] Increasing Difficulty
[23:43:53] The updated difficulty based on the player's performance is: 1
[23:44:03] The current Difficulty is: 1
[23:44:03] The difficulty will lower, if performance is less than : 65
[23:44:03] The difficulty will increase, if performance is more than : 80
Just Restarted The Game, the new Difficulty is: 0
UnityEngine.MonoBehaviour:print(Object)
DynamicDifficultyAdjustment:ResetDifficulty() (at Assets/Scripts/A.I/DynamicDifficultyAdjustment.cs:204)
GameManagerScript:RestartGame() (at Assets/Scripts/Level & Game Manager/GameManagerScript.cs:205)
UnityEngine.EventSystems.EventSystem:Update() (at /Applications/Unity/Hub/Editor/2019.3.12f1/Unity.app/Contents/Resources/PackageManager/BuiltInPackages/com.unity.ugui/Runtime/EventSystem/EventSystem.cs:377)
Statistics
Audio:
Level: -75.2 dB
Clipping: 0.0%
DSP load: 0.2%
Stream load: 0.0%
Graphics:
87.5 FPS (11.4ms)
CPU: main 11.4ms render thread 1.1ms
Batches: 148 Saved by batching: 5
Tris: 78.2k Verts: 130.8k
Screen: 1080x1920 - 23.7 MB
SetPass calls: 94 Shadow casters: 25
Visible skinned meshes: 1 Animations: 0
The updated difficulty based on the player's performance is: 5
Auto Generate Lighting On


1680
44
PAUSED
Console
Clear
Collapse
Clear on Play
Clear on Build
Error Pause
Editor
[23:44:03] The difficulty will increase, if performance is more than : 80
[23:44:03] The Performance of the player for the past 10 seconds is: 94 / 100
[23:44:03] Increasing Difficulty
[23:44:03] The updated difficulty based on the player's performance is: 2
[23:44:13] The current Difficulty is: 2
[23:44:13] The difficulty will lower, if performance is less than : 65
[23:44:13] The difficulty will increase, if performance is more than : 80
[23:44:13] The Performance of the player for the past 10 seconds is: 92 / 100
[23:44:13] Increasing Difficulty
Just Restarted The Game, the new Difficulty is: 0
UnityEngine.MonoBehaviour:print(Object)
DynamicDifficultyAdjustment:ResetDifficulty() (at Assets/Scripts/A.I/DynamicDifficultyAdjustment.cs:204)
GameManagerScript:RestartGame() (at Assets/Scripts/Level & Game Manager/GameManagerScript.cs:205)
UnityEngine.EventSystems.EventSystem:Update() (at /Applications/Unity/Hub/Editor/2019.3.12f1/Unity.app/Contents/Resources/PackageManager/BuiltInPackages/com.unity.ugui/Runtime/EventSystem/EventSystem.cs:377)
Statistics
Audio:
Level: -75.2 dB
Clipping: 0.0%
DSP load: 0.2%
Stream load: 0.0%
Graphics:
87.5 FPS (11.4ms)
CPU: main 11.4ms render thread 1.0ms
Batches: 148 Saved by batching: 5
Tris: 78.2k Verts: 130.8k
Screen: 1080x1920 - 23.7 MB
SetPass calls: 94 Shadow casters: 25
Visible skinned meshes: 1 Animations: 0
The updated difficulty based on the player's performance is: 5
Auto Generate Lighting On


1680
44
PAUSED
Console
Clear
Collapse
Clear on Play
Clear on Build
Error Pause
Editor
[23:44:13] The Performance of the player for the past 10 seconds is: 92 / 100
[23:44:13] Increasing Difficulty
[23:44:13] The updated difficulty based on the player's performance is: 3
[23:44:23] The current Difficulty is: 3
[23:44:23] The difficulty will lower, if performance is less than : 65
[23:44:23] The difficulty will increase, if performance is more than : 80
[23:44:23] The Performance of the player for the past 10 seconds is: 86 / 100
[23:44:23] Increasing Difficulty
[23:44:23] The updated difficulty based on the player's performance is: 4
[23:44:33] The current Difficulty is: 4
Just Restarted The Game, the new Difficulty is: 0
Statistics
Audio:
Graphics:
The updated difficulty based on the player's performance is: 5
Auto Generate Lighting On


1680
44
PAUSED
Console
Clear
Collapse
Clear on Play
Clear on Build
Error Pause
Editor
[23:44:23] The Performance of the player for the past 10 seconds is: 86 / 100
[23:44:23] Increasing Difficulty
[23:44:23] The updated difficulty based on the player's performance is: 4
[23:44:33] The current Difficulty is: 4
[23:44:33] The difficulty will lower, if performance is less than : 65
[23:44:33] The difficulty will increase, if performance is more than : 80
[23:44:33] The Performance of the player for the past 10 seconds is: 94 / 100
[23:44:33] Increasing Difficulty
[23:44:33] The updated difficulty based on the player's performance is: 5
Just Restarted The Game, the new Difficulty is: 0
Statistics
Audio:
Graphics:
The updated difficulty based on the player's performance is: 5
Auto Generate Lighting On

D) Deployment instructions

In order to download and play the game, a smart device running any of the versions mentioned in the section: 2.4. Operating Environment has to be available. In the case where the device runs on Google's Android software, the respective APK file to be installed on the device, can be found in the folder called: "Executables". The user has to transfer the file to their android device, allow apps from untrusted sources through the settings and finally click to install the video game (APK file). It is important to mention that due to the Covid-19 virus, the Google Play Store does not yet allow submissions for alpha and beta versions of online video games. Therefore, the game could not be published on the Play Store as an alpha version. Additionally, some of the functionalities such as: the leaderboards and the achievements may not work properly. When it comes to the IOS devices, any device running a valid IOS version that is mentioned in this section: 2.4. Operating Environment, can have the game installed on their device. However, the user will have to email the developer at: d.mavrofrydis@acg.edu in order to request a key download for the Alpha version of the game, which is already available on the App Store. After that, the user will just have to follow the instructions on the email in order to download the application on their IOS device. Finally, the Unity Project folder is included as a zip file, which can be opened in Unity by first unzipping the file, downloading the latest Unity 2019 version, and finally pressing open project and choosing the unzipped folder.

IOS Deployment Screenshot:

2:27

Apps

Run Pirate Run

Version 0.1.1 (1)
Expires in 90 days

INSTALL

What to Test

Software Development Capstone Project
Developer: Dimitrios Mavrofrydis
Supervisor: Prof. E. Vagianou more

Send Beta Feedback

Information

App Details

Notifications Push, Email

Previous Builds

Stop Testing

From the Developer

Run Pirate Run
Version 0.1.1 (1)
Expires in 89 days

What to Test

Software Development Capstone Project
Developer: Dimitrios Mavrofrydis
Supervisor: Prof. E. Vagianou
The American College of Greece – Deree
Spring Semester 2020

Next

E) Originality report