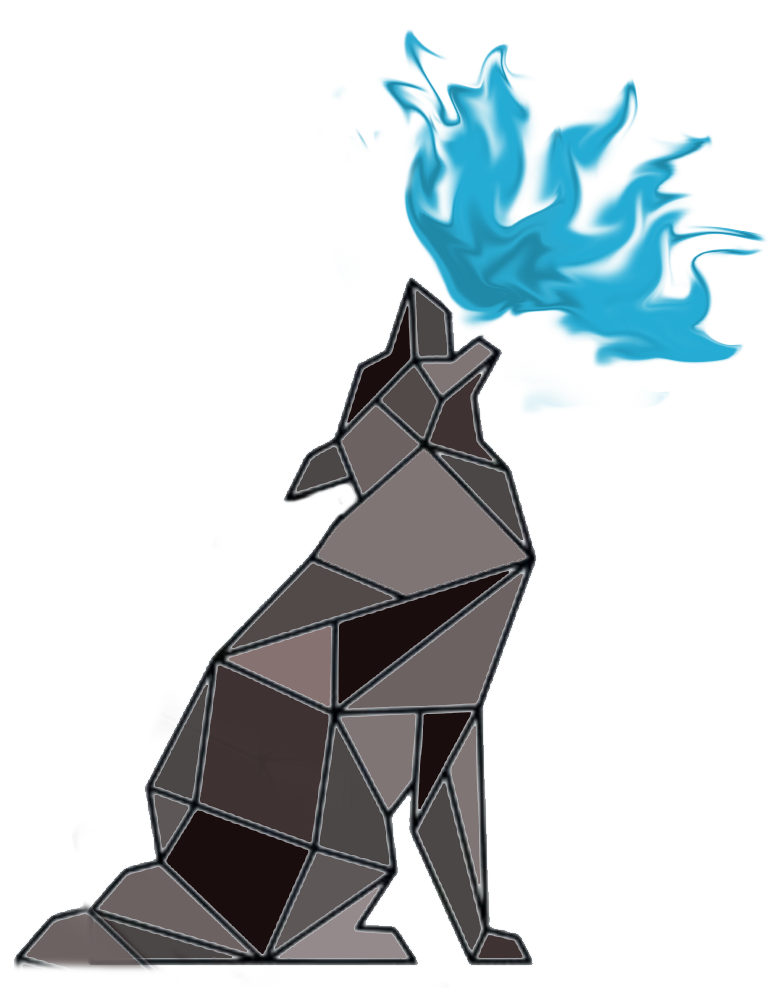
**Maintenance Manual**

**Breath of the Coyote**

****

**Spring 2020**

**Cameron Maclean – Project Manager**

**Stephen Anderson – Assistant Project Manager**

**Valentino Fernandez – Software Engineer**

**Jugen Fornoles – Software Engineer**

**Christian Morales – Software Engineer**

**Brandon Edmonds – Software Engineer**

**Section 1: File Structures**

Our file structure is well kept and managed using GitKraken as the main interface when working through git. All of our files are saved on a repository on GitHub in order to make it possible for all group members to freely access and branch off to make changes without jeopardizing the master files.

**Assets**

When downloading our project and opening it on Unity, the main folder worked on is the Assets folder. All of the other folders not in this folder should be ignored, as they are folders Unity uses, and not for editing.

**Animations**

This folder holds all of the animation controllers and calls. There are several animations for the player that have calls to our scripts, so if you are unsure when the player calls a certain program, you should look at the end of the animation Timelines.

**Asset Store**

This folder holds all the free online assets we used from the Unity Asset Store. Inside their own folders are the meshes we used during development of the second level.

**Audio**

The audio Folder holds all of our audio clips for the game. The folder is divided into SFX, Music, and the Mixers.

**Joystick Pack**

The joystick pack is a free pack directly from the Unity asset store. It has several pre made scripts that edit the existing Unity controls scripts and enables us to use buttons and joysticks properly.

**Materials**

This folder holds all the materials for our meshes. Materials are simply described as items you use to paint meshes, otherwise everything would be bland and unpainted. If any colors come up as pink, you must go to the top left corner on the browser and select edit -> Render Pipeline ->Lightweight Render Pipeline ->Upgrade Project Materials to LightweightRP materials and allow it to fix them all. Afterwards, that should be fixed.

**Models**

This folder is where we store all of our 3D assets. Almost every 3D asset we have in this project is in this.

**Placeholder Assets**

This is the original source of our enemies in the game. This is the file we originally received online, and since then we have worked on it ourselves and moved the needed things to their respective folders.

**Prefabs**

This is where all of our project prefabs are. For Enemy prefabs, you will be going to the first Enemy folder and then going to the enemies folder inside of that one. That is where the enemy p[refabs are. For prefabs that we used to decorate the map, you will go to the Environment folder. For the prefabs for the level 2 challenge zones, you will go to the Level2zones folder. For the elevator prefab found on level 1, you will be going to the Platform folder. For everything player related, you will be going to the Player folder. For the challenge zones on level 1, you will go to the Puzzles folder. For all of the in game UI, you will be going to the UI folder. For the VFX used in the game, you will go to the VisualFX folder.

**Scenes**

This folder has all the test scenes and game scenes we used. If continuing on this project, I strongly recommend starting out with making your own scene for you to play in and test in. The Scenes used in the game are the Main Menu scene, Load Game scene, Game Scene, Level2 Scene, and the Credits scene.

**Scripts**

This is the most important folder in the entire project. We divide this folder into 5 main parts.

**Enemy Folder**

The first part, the enemy folder, is where the state system for the enemy is held. Those states are rather new and very important to the new AI system. Included in this folder is also the player combo system, KeyCombo.cs, that allows the player to have his 3 combo attack, and all the die states for the enemies.

**Environment Folder**

This folder has all the checks for a completed section , and it also has all the platform movement scripts for the moving platforms in the game. For level 2, everything in victoryscriptslvl2 are the scripts that decide it’s victory. FallTP.cs and BoundTP.cs teleport the player to a starting position if they go into an area they aren’t supposed to go to, or if they go somewhere they shouldn’t be able to. HealPlayer.cs heals the player if they collect certain things on the map.

**Player Folder**

This folder holds all the player controls and data that the player can have. In this, PlayerCharacterController.cs does a lot of the heavy lifting, as it is responsible for almost all of the player movement and the calls for the player win state so it can move to the next scene. Most of the win state calls are at the bottom of the script. PlayerData and Player are mainly used in the save system. They mostly store the player information needed for loading up the game or saving it to the save. BattleMusicPlayer, plays all the music when in combat. CMFreelookControlOverride enable the joystick to be used for controls.

**UI Folder**

The UI folder holds a lot of the save system information, as that is mainly handled by the UI in the game. PauseHandler.cs pauses the game while playing. The MenuManager.cs controls all the menus and the movement between them. The savesystem saves and loads all the info from the different saves in the game.

**Shaders**

These are all the assets created by the Unity Shader Graph. Using the Shader Graph, we were able to create the special shader used in the game, build custom lighting, and create the water you see in the game as well.

**Terrain**

This folder holds all of the painting tools we used for the project. All of these were used on the in game terrains to make them what they are now.

**Textures**

This Folder holds valuable items used in the Shader Graph and the VFX Graph. These assets are free assets provide by Unity.

**VFXGraph**

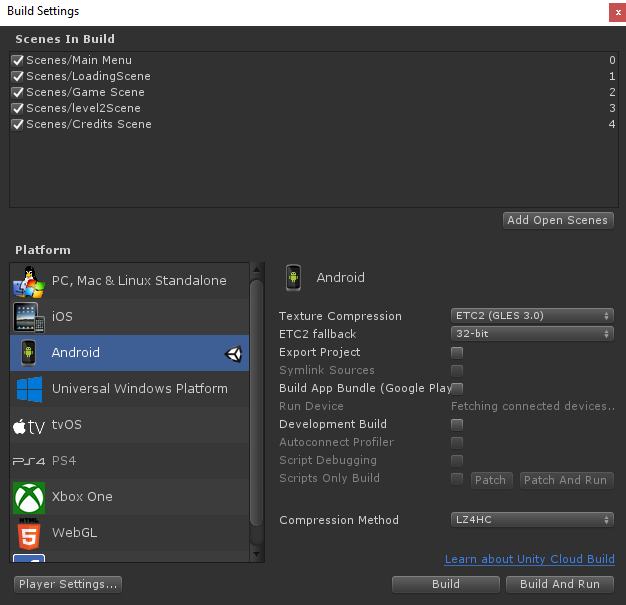
This folder is where we keep all the stuff created in the Unity VFX Graph. At the time of creating this VFX Graph was VERY experimental and the version of Unity we used was very barebones. Also, in this experimental version of VFX Graph, it was not well optimized for mobile use. In future version updates some of these optimization issues were fixed, but since we were on our version of Unity we could not get those new updates without risking the rest of the project.

**Section 2: Instructions**

The application can be run using one of the following options listed below:

1. Downloading Unity Engine and building it on there.
2. By downloading the APK and installing it onto an android device.

To build and run the app on Unity Game engine You must follow these steps

1. The first step is to download the project itself.
2. Once that is done, you must download Unity. You must make sure you have Unity Version 2019.2.3f1. Make sure when you download it to add the Android Build Support.
3. Once the project is open, go to the top left and select window, then select package manager. You need to make sure these packages are there, otherwise you must find them in the package manager and download them.
4. After that, as a safe precaution go to edit -> Render Pipeline ->Lightweight Render Pipeline ->Upgrade Project Materials to LightweightRP materials. This will make sure all materials are properly built to Lightweight Render Pipeline.
5. Now you should be ready to build the project. For this you can go to File -> Build Settings or press ctrl+shift+B. This will bring up the build interface where you can decide how to build it. These are the current build settings:
6. Select Build and select a place to save the APK. After that you just move the APK to your phone and now it is built

To Run the app on an Android Device, the following steps must be taken:

1. Take the existing APK and save it onto your android device
2. Navigate to your file manager. This can be done by wither going to all your apps and selecting my files or going to the settings menu where you can access the file manager there as well
3. Tap Downloads
4. Tap and run the APK file you just downloaded. It should download onto your device and now you have the game!

**Section 3: Good implementations**

Positives implementations of this app:

* This save system is well implemented and made to be universal for Android, PC, and iOS.
* Many of the systems have been made more universal so now the developers can add as many levels as they want with no issues on the scene changing systems and save system
* The controls are simplistic in nature and easy to understand
* Most of the tweaks for the controls are easy to change
* The source code is well documented and will provide detailed explanations for the developer to look at when or if it crashes
* Many of the old scripts that were designed for specific circumstances have now been edited to be more rugged and universal

**Section 4: Needs Improvement**

The following are weak points:

* The game is not well optimized and require a lot of power from the CPU. This is because of the dated versions of the Shader Graph, VFX Graph, and Lightweight Render Pipeline.
* There are some circumstances of the AI not properly traversing their state systems.
* This currently does not have an iOS counterpart.

**Section 5: Overall Recommendations**

All future development of this project will revert back to the client with these recommendations:

* Update the game to one of the newest Unity versions, and work on updating the game to the Universal Render Pipeline.
* Many of the issues with the game stem from the dated version. The single update will solve most of the problems with the game
* Lower the file size on the project. This can be done using the new compression models in the newer versions of Unity.
* The framework for making an iOS version is built and builds were actually experimented with. Simply research what it takes to create a full iOS version and how to upload it. That is all that is needed for iOS version