Endless Runner Log

- Created a new 2D project and made a script PlayerController that currently lets the player walk side to side so I can test the controls out easier. Also created a script for the camera so that it will follow the players horizontal movement only.

- Added jumping to the PlayerController script, which is located inside a coroutine and set up so that the character's jump height changes depending on how long the jump is held. Added a grounded check so the player can't jump forever, uses a box cast that tells the PlayerController it's grounded if it collides with anything on the Platform layer.

- Redid the jump to make it a bit snappier, it's no longer in a coroutine it's just in Update for now. If the player can jump and hits space it changes the rigidbody velocity to be directed up by a given jump distance. Then for the variable jump time, the speed at which the character falls is changed depending on whether the jump button has been held or not. I also added in some animations for idle and running, and will add in the other animations later.

- Created an ObjectPool script using a singleton pattern, it takes in a prefab and an integer for how many objects you want to instantiate when the game starts. The GetPooledObject() function returns the first inactive object in its list that it finds, but if all the objects are currently in use it will create a new one and return this instead.

- Created a PlatformManager script to generate the platforms as the game runs, and it takes in the minimum and maximum distance you want between the platforms horizontally, as well as a transform called GenerationPoint. This transform is an empty object in the scene that is placed some distance ahead of the player and will spawn platforms if the current position of the PlatformManager is behind it. Currently all platforms spawn at the same height but with a gap between them randomly set between the min and max horizontal distance given in the inspector. It also takes in an object pool for the platforms and if a platform needs to be activated it calls the GetPooledObject function and places it at the generated position.

- Created a new script to go on each platform that deactivates them when they fall behind the player too far, called PlatformReset. It finds a game object in the scene and when the object this script is on moves past it, it is turned off and returned to the object pool.

- Set the platforms to activate at different heights, clamped between a minimum and maximum height range.

- Changed how the object pool works, no longer using a singleton pattern so that the scene can have multiple different pools active at once. This is to allow different platform types to be easily shuffled into the selection of platforms, currently have 3 different types that the game picks from. Edited the PlatformManager to take in a list of object pools, and now the platform that is selected is randomly chosen from one of the available pools.

- Edited the line of code that sets the distance between platforms so that now they will no longer overlap, as the x-coordinate is now set to start from the ends of the platform instead of the middle. Also recentred the children of the platform prefabs as that was causing issues with positioning.

- Created collectable prefab with an animator that causes it to play an idle animation until it is collected. Also created a Collectable script to go onto this object that causes the object to play the collected animation when it is triggered by the player, which then turns the object off and resets the animation to idle when over as it's hooked up to an animation event. This script also holds how much a collectable is worth.

- Created a CollectableManager script to handle activating and positioning the collectables. Has a reference to an object pool of collectables as well as the buffer between the collectables, and has a function PlaceCollectable that takes in a starting position, the number of collectables to activate and the length of the platform to activate over. In order to activate them evenly over the platform firstly it makes a new integer called spacing and makes it equal to the distance between collectables. Then when positioning the activated collectable, it sets the position to be at the passed in starting position plus the spacing. It does this in a loop so it can activate the given number, and then before the end of the loop, it adds the platform length divided by the number of collectables to the spacing so the next object is placed further along, and is evenly spaced and centred on the platform.

- Added in some randomness so now the PlatformManager will only spawn the collectables if a random number between 1 and 100 is lower than the percent chance stored in the CollectableManager. This adds some variety, and looks nicer than making the CollectableManager handle the collectable activation for each one in a set (this made the collectables look uneven when spawned in, and could lead to the level looking really empty)

- Created a ScoreManager script so the player can get points for how far they run plus bonus points for picking up the collectables. It works by taking in a text UI element and setting it to the current score in the update function, which is increased by the points per second each frame and then rounded so you don't need to see all the floating points. Collectables use the AddPoints function when they're picked up to add to the score. This script uses a singleton pattern so the current points can be seen from any script, as the game will only ever need one ScoreManager active.

- Made a main menu with buttons that will either start the game or quit, plus a currently non-functioning highscore button for possible use later. Also created a death and pause menu for the game.

- Created a GameManager script for the game, which currently handles loading the game, main menu, restarting or handling the player pausing or dying. The death screen currently pops up when the player falls out of bounds and dies, and tells the player their

final score as well show buttons that let the player either restart or quit to the menu. The pause menu is brought up by a toggle in the scene and allows the player to either restart or go to the menu as well, to unpause you just click the pause toggle again. It currently doesn't pause the game while up.

- Set it so the character auto-runs now instead of taking user input as the jump doesn't need much testing anymore.

- Pause menu now stops gameplay by turning the Physics2D autosimulation off and telling the PlayerController script that the game is paused. In the PlayerController, if the game is paused it won't let the player jump and it prevents the character from auto running. It also stops it's animation. When the game is unpaused it turns the Physics2D back on and the player can move again. Added new function called turnPhysicsOn() to the game manager so that when the player loads the main menu or restarts the game, it checks to see if the Physics2D is turned on and if not sets it to true.

- Moved some code around in the PlayerController script so that all movement code is done in FixedUpdate while all input checks are done in Update.

- Added in the mobile controls so that the player taps anywhere on the screen to cause the character to jump, changed the PC controls over to using buttons instead of keys for input.

- Created an obstacle for the player to run into to die, gave it an animator so that it looks deadlier.

- In the PlayerController I added the OnCollisionEnter2D function so that the player can detect when they've collided with an obstacle, tagged as "Obstacle". When they hit one it sets the hitObstacle bool to true and then plays the death animation. In the GameManager I then edited the death check in the Update function so it also checks if the player has hit an obstacle and if so it says the player is dead. I also edited the waiting time inside the DeathState() coroutine so now it either waits a given time or, if the player died to an obstacle, it waits until the animation has completed.

- Fixed an issue where platforms were spawning offscreen by moving the minimum and maximum height objects onto the camera from the platform manager object, and also added a buffer to each one based on the height of the platform.

- Made an ObstacleManager script that holds a reference to an object pool plus a minimum length a platform needs to be in order to have an obstacle spawn on it. It also takes in a buffer so the obstacle doesn't spawn on the very edges of a platform. It currently only spawns one obstacle per platform due to me not wanting the game to be too hard. The current chances of spawning an obstacle are 30%, and if it fails to spawn one then it will try to spawn the collectables at an 80% chance of happening.

- Changed where the check for if a platform is long enough for an obstacle to spawn occurs, it now happens in the PlatformManager so that small platforms still have a chance of getting collectables.

- Tweaked the speed of the player and the max distance the platforms can spawn away from each other to account for zones the player can't jump from.

- Added background music to camera.

- In order for the music to not restart each time the game is reset, I am redoing the Restart function in the GameManager so that it doesn't just reload the scene. It now tells the player that they're not dead, paused or hitting an obstacle, resets the player position in the world and calls a new function from the ObjectPool, ResetAll. This function deactivates all objects in a pool and resets their position. The GameManager now also takes in a game object for the first section of the track, so that when the player is reset they have something to land on again.

- Fixed a bug with the pause toggle not being reset properly when the player resets by passing it into the GameManager and, when the game is reset, manually invoking its events and then setting it to be off. Without this the pause toggle needed to be clicked twice after a reset in order to bring the menu up.

- In the PlayerController in the android controls the player can only jump if they aren't hitting a UI element when clicking. This prevents the character jumping when clicking the pause toggle.

- Added sounds for jumping, dying and collecting items, as well as the jumping, falling and dying animations.

- Fixed bug where the score wouldn't reset after restarting the game, or pause during the pause screen.

- Fixed issue where the death animations length wasn't being used to calculate the wait time for death coroutine.

- Fixed a small issue where the PlayerController would play the jump sound on restarting the game if you tried to jump while dying by adding an isDead check as well as an isGrounded check before checking for any input.

- Added in a scrolling background. Was initially going to use an object pool to make them and then a manager to control the movement but ended up just placing the needed 3 into the scene and then having them control their own movement with the BackgroundScroll script. GameManager calls a reset function on the BackgroundScroller when the game is reset so that the background is put back to its initial position.

- Added an instruction page to the main menu, changed the UI text colours to be more readable. Removed the high score button also.

- Changed how android builds detect if you're tapping over a ui element as it wasn't working on mobile. It now loops over the current touches and if the pointer isn't over UI it sets jump to true.

- Changed how the obstacles are placed when activated. Now the ObstacleManager takes in the player's collider width and finds a position from the left edge of a platform plus the player's width to the right edge. This makes sure that the obstacles leave enough room for the player to land on the leading edge of the platform.

- Edited how positions for the activation, deactivation, minimum and maximum heights are calculated by using the camera width and height instead of using transforms in the scene. For the generation, the position is calculated by adding half the width and a buffer to the cameras current position and if the managers horizontal position is less than this number it will generate another platform. For the deactivation it's very similar, but instead the buffer and half width is subtracted. The max height is the height of the camera minus the players and platform height (so you can still see them when they're running along them), and the minimum point is the bottom of the camera but with the platform height and a buffer added so the full platform is onscreen.

- Instructions now change depending on what build it's set to, so android builds say to tap to jump while others will say hit space.

- Cleaned up code and removed scripts that are not being used.

- Removed reference to the background scrollers from the GameManager, instead the position of the backgrounds is reset when the camera moves too far to the left of the effect.

- Cleaned up code a bit more, made the main menu music loop, and added an icon.

- Changed the PC controls to be a mouse click as it feels better than hitting space, also removed the ability to use a mouse to jump from the android controls as having them in there was making the player jump when hitting pause due to how it checked for if the pointer was over a game object twice.

- Fixed small issue where the pause menu wouldn't turn off properly when you pause before dying.

- Built the game and edited the gitignore file so it could be added to the repo.