Coding Journal

Coding

* ***Countdown Timer Tutorial (15th October - 22nd October):***
* This was the first tutorial I did, and at first I struggled with getting used to 2D, as I hadn’t done much coding before in the last semester. However, I got used to the Unity controls fairly quickly, and I was able to move around in Unity thanks to some helpful YouTube tutorials, as well as some help from my friends.
* Finding a useful tutorial was the first hard bit, as there are are multiple ways to make a timer that counts down. However, I found a YouTube tutorial from a channel called Speed Tutor, who had done a tutorial that looked easy to understand and simple to follow. I understood his instructions about where to place game objects in Unity, as well as how to change them to match the tutorial. However, getting started with coding was a challenge at first.
* When I got started, I had no idea what any of his code meant. I didn’t know what affected his timer and what affected his text. However, as I continued to follow his tutorial, parts of the tutorial became easier to understand. I was able to learn what some of the values meant in his code, for example the code “uiText.text = timer.ToString(“F”);” means that the code identifies the text as a string, which in C# terms, is a word.
* Some parts of the code were still tricky to figure out, so I spent a lot of time trying to work out what everything meant. But I managed to work everything out and keep going with the tutorial, eventually finishing it with a working timer, and a button that reset the timer back to the number it was originally set to once pressed.
* Creating the button was slightly trickier to figure out at first, but once the main parts of the code were down, it became easier to understand. The code stated that under a new function called “ResetBTN,” canCount = true and doOnce = false. This code means that when the button is pressed, it will be able to count down from the start again.
* ***2D Top Down Movement Tutorial (9th November - 15th November):***
* This was the second tutorial that I made, and this one I found a lot easier than the countdown timer. The thing I struggled with at first was finding a good tutorial, like the countdown timer before, as some games had different ways of implementing top down movement in a game. However, I found a video from a YouTube channel called BlackThornProd that was easy to understand, so I used his tutorial as inspiration.
* Starting the code was easy as the first parts simply declared that the variable rb was equal to the RigidBody2D component that would be added to the player (which is important because the Rigid Body would make the player move through scripting). However, changing the player’s body type confused me at first because of the difference between Dynamic and Kinematic (the type needed to be switched to Kinematic so the player wasn’t affected by gravity).
* One other part I struggled to understand was how to move the player. The code that was given was Vector2 moveInput = new Vector2(Input.GetAxis(“Horizontal”), Input.GetAxis(“Vertical”)); At first I didn’t understand what it meant, but after some research on Google, I discovered that it only means that it was calling the horizontal and vertical axis’ depending on which button the player presses. For example, if the left arrow key is pressed, it gives a value of -1, whereas if you press the right key, it gives a value of 1.
* Once the rest of the code was done, for example adding a variable for speed and changing the movement to be more smooth by changing “GetAxis” to “GetAxisRaw,” the tutorial was finished. Overall I found it easier than the countdown tutorial, but it was still a challenge.
* ***Coin Collectable Tutorial (14th November - 23rd November):***
* I continued this tutorial from the 2D Top Down Movement tutorial previously done, and this tutorial seemed easy at first. However, like the first two, the tutorials were hard to find at first, as they were either too hard to follow or the person giving the tutorial was difficult to understand. However, I did find a tutorial on YouTube by a channel called Xlaugts which was easy to understand. However, a change I made from the tutorial in the video is that I didn’t include floating coins and I didn’t make the coins bounce in Unity. Instead, I wanted to make a coin that sat still that the player could still collect and add to a score count.
* This code took me the longest to finish as there were 3 different scripts that I had to create; updating the player controller from before, a new score manager script and a coin script. The update to the player controller was the easiest part, as the only addition was to state that when the player collided with a game object with the tag “coin,” it destroyed the object. The only thing that was slightly challenging was figuring out why it didn’t work in Unity at first, although this was easily fixed by adding a Circle Collider 2D to the player as well as the coin.
* Creating the score manager seemed easy at first, as the only things needed to accomplish were to create the text in Unity and write a script that stated whenever a game object with the “coin” tag attached to it got destroyed, the score counter increased by 1. However, it wasn’t as simple as I thought, as I used TextMeshPro instead of the default text. Therefore, since I changed the text, I had to change the code to state that I was using TextMeshPro. Thanks to a quick Google search though, I found that I needed to use different text functions to tell the code I was using TextMeshPro. For example, instead of stating “using UnityEngine.UI,” I had to say “using TMPro” to call the TextMeshPro function, and I had to replace “text” with “TextMeshProUGUI.” Other than that however, the code was easy, and I had barely any problems with the coin script at the end.
* ***Random Spawn Point System Tutorial (1st December - 8th December):***
* For this tutorial, I continued from the Coin Collectable tutorial, and at first I thought this would be an easier tutorial than the others as I had done a respawn system before for my 1st Semester Mobile Demo Build module. However, it wasn’t as easy as I thought, and I had to find a new tutorial to help. Luckily, I found a tutorial painlessly this time, as a YouTube channel called PekkeDev had a very useful tutorial. However, there was a main change that I did and it was that in his tutorial, he spawned 3 different kinds of enemies that fell. Instead, I made the coin a prefab and I wanted to spawn it from multiple different spawn points randomly.
* For the code, once I watched the tutorial a couple of times, I understood the code and got to work implementing it, for example understanding that timeBtwSpawns = startTimeBtwSpawns meant that this made a variable in Unity that allowed a time delay between when the coin respawned, so I could make a coin spawn in a random location every 1 second for example. However, the one line of code that I had to edit extensively was Instantiate(coins[0], spawnPoint.transform.position, Quaternion.identity); which was the line that spawned the coins. However, I had to replace [0] with [randPosition] which meant that the coins would spawn on the spawn points labelled in Unity. Other than this line, there weren’t many issues with the code at all.
* ***Final Component (9th November - 9th December):***
* The main change with the final component is that I changed my countdown timer slightly by not including the button in the game. This is because I didn’t know how I could implement it into my game. I have noted this down though, and over the Christmas break, I may continue to work on this component and work out how I can implement the button as, for example a button that when pressed, restarts the whole game, including the score.
* Implementing all of the tutorials into the final component wasn’t very hard, as I combined the 2D Movement, Coin Collectable and Random Spawn Point tutorials into 1 game project. The only tutorial that I had to implement in the code was the Countdown Timer tutorial, which was painless apart from changing the text to TextMeshPro and stating that in the code.