**Learning Journal**

16/02/2021

Started development on an optimal player controller that used Physics to be interactable with other objects and be usable for other games. I realised that a Rigidbody was necessary for what I wanted and attached that to my player. Encountered a strange bug that made the controls flip if the player's Y axis was changed in any way, was told to use Velocity with Vector3 rather than my original method to avoid this issue. I think I had AddForce originally which caused the issues.

Then I went to work on making a script for the camera to follow the player, I researched that having the camera being a child of the player would bring so problems so instead I just set the offset as the camera’s transform position - the players transform position, then made sure to keep the cameras transform position = to the players + that offset.

Since I wanted to make an almost all-purpose Player Controller I decided to add a jump mechanic to it, I was able to get it done by referencing the RigidBody and then adding force to the up axis, I used a float called jump height to designate how high I wanted to jump to be which worked out. This led to my next issue of allowing the player to jump only once and not able to jump again until on the ground again. This was more simple than I expected by just naming all my platforms “Ground” and having a public bool called “isGrounded”, this was set to false and was only true when the player’s RigidBody touched a game object named “Ground”.

23/02/2021

Deciding on making a falling platform package for my third package, I underestimated how much was needed to make a close to all purpose script for a falling platform and ran over my baseline estimate by more than two hours.

The first problem came when I figured it be a good idea to have the platform come back once it fell, this brought a problem though as that meant I could not just add negative force to the Y axis of the platform because it would continue adding that force and might cause issues, that and it will not come back up.

My tutor showed me that I could attach a RigidBody to the platform and periodically turn on Is Kinematic and Use Gravity when the need occurred to rise and lower the platform.

I had set up two floats called downSpeed which dictates how fast the platform would drop and resetTime which would bring the platform back up after a certain amount of time.

I was then informed that it would be best to return the platform to its original position rather than add force to its Y axis as it would set itself into position.

I was able to develop a reset position script by referencing the RigidBody and turning on and off its Is Kinematic and Use Gravity as well as setting a bool called isRising to true, isRising means if the platform position is lower than its original position, then it will drop lower, I then set it to return to its original position once a few seconds had passed.

01/03/2021

I was developing a sound manager for one of my briefs, I researched what would be the best way to develop this and found a method that allows me to incorporate as many sound effects as I need.

I set up a sound array and wrote foreach sound effect in the array, it would add a component, that being an audio source, I also made two settings which allowed me to change the volume and pitch in another script called “sounds”

[Range(0f, 1f)]

public float volume;

[Range(.1f, 3f)]

public float pitch;

I made it a public class by erasing the mono behaviour and then making it viewable through the inspector by writing [System.Serializable]

Then to allow me to choose a particular sound effect I wrote:

public void Play (string name)

{

Sound s = Array.Find(sounds, sound => sound.name == name);

s.source.Play();

}

02/03/2021

To make for my last Unity Package, I wanted to make a good reset of the game script that would reliably reload the active scene if the player went through a certain collider.

To do this, I set up two different scripts as I was going to use Unity’s events system to accomplish this, I made am “Events system” script with a string called “triggerTag” which sets up the code to only trigger when a specific tag meets the codes conditions and a public unity event called “triggerAction”. I could now attach this to any game object and it would allow me to call different functions as if I were using a button on a canvas.

I then set up the method that I wanted to happen to call functions, which was an OnTriggerEnter and then made it so the triggerTag would invoke the triggerAction. This means that the trigger tag could be changed in the inspector to meet different needs and call the corresponding action.

Then I just made a reload the scene script by writing

public void ReloadScene()

{

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);

}

09/03/2021

Made a countdown system to go along with my RCR project, was able to develop a countdown function by using a canvas text and making two variables called currentTime and startingTime.

startingTime would be set to ten for the text and currentTime would be set to 0. I also made a [SerializeField] Text countdownText; which would be my countdown text.

I then made currentTime = startingTime in the start method and made currentTime-= 1 \* Time.deltaTime; which meant that the currentTime will be 10 and will countdown one per second. Finally, I made countdownText.text = currentTime So the canvas text would display the current time counting down.

I ran into a small issue when Unity told me I couldn't "convert to string" but I was able to rectify that by adding .ToString("0");. Also adding the zero meant that the game would countdown using whole numbers rather than decimals.

20/04/2021

Began working on my AGP project wanted to get a good handle on my game idea, initially went for a vaulting mechanic that would happen on a lowered part of the wall, I worked with one tutors using a collider at a certain point, when an input is pressed the player would jump over a ledge using add force. When tested it seemed too uncontrolled, adding force during the X and Y axis seemed to push my character up but not over the wall. I put the mechanic on the back burner until I could figure it out or find a better mechanic.

After testing the vault mechanic, I made the choice to forgo it, instead, a hide mechanic which allows the player to hide within a cube and hide from a patrolling enemy, my tutor suggested the following.

We made a cube that was a little bigger than a player and with a trigger box collider. Now that the player can enter the box, I need to make it invisible to the enemies’ Raycast vision script. I had asked if it was possible to change the tag of my player game object to be invisible to the enemies’ Raycast, He told me that there is an easier way to get that effect simply by creating an entirely new layer called Player, he informed me Unity has a layer called “Ignore Raycast” which does as its named. We decided that once the player enters the box, they would switch to the “Ignore Raycast” layer and then switch back to the “Player layer” once they leave the box.

We did this by going into the players movement script and setting up a private Boolean called “insideLocker” and setting it to false, we then created an OnTriggerEnter function where once the player enters the box, insideLocker will equal true. Once insideLocker = true the player will change to layer 2 which is the “Ignore Raycast” layer. We finished the code by setting up an OnTriggerExit setting the “players” layer back to the created Player layer. The code looks like this.

Void OnTriggerEnter(collider other)

{

If (insideLocker = true)

{

gameObject.layer = 2

}

}

Void OnTriggerExit(collider other)

{

If (insideLocker = false)

{

gameObject.layer = 8

}

}

Lastly, I wanted to make sure the hiding object would disappear after a few seconds so the player wouldn’t hide there for the entirety of the game, so we did this by creating a public float called “destroyTime” which would allow us to change the time it takes to destroy itself. We then set up an IEnumerator function called “Die” which would destroy itself after a few seconds set by me and reset the layer of the player as once the hiding game object is destroyed, the player should be visible. The code looks like this:

IEnumerator Die()

{

yield return new WaitForSeconds(destroyTime);

player.ResetLayer();

Destroy(gameObject);

}

23/04/21

Today for one of my specialism briefs, I was tasked with making a rhythm game, timing the inputs of my keyboard to the beat of music to avoid obstacles and get pickups. I wanted to make a continuous moving script that would allow the player to move along a long platform. To do this, I needed to reference the players Rigidbody as a public variable and set a public float called speed to 5.

(I set the platform to be 15 units wide along side the X Axis, from -5 to 5, this is important to note for something I do in my code)

I set up a private void FixedUpdate() and made a Vector3 forwardMove script where the players forward position would go in that direction multiplied by the speed float I set at the top of the code, multiplied by Time.fixedDeltaTime to make sure it happens at a consistent time link to my PC’s performance. Underneath that, I referenced the players RigidBody to move its position by its current position + the value of forwardMove that I distinguished above.

After that, in the start function, I used GetComponent to reference the RigidBody at the start of the game, then in the Update function I used transform translate to set the players position based on what Input.GetKeyDown(KeyCode.)) I used, I made the players position, -5x when A was pressed, 0x when S was pressed and 5x when D was pressed.

27/04/2021

I went back to my RCR project and wanted to further develop the hiding mechanic that was worked on, I wanted the player to interact with the game as much as possible, so I decided to make the hiding mechanic on a button pressed as compared to just walking into the locker (Cube stretched into a standing rectangle) and becomes invisible to enemies.

To get the desired effect, I had to take into account where the player could enter the locker, the front position of the locker, if the player were touching the entrance to the locker as well as if the player were in or out of the locker. I set this up bym making lockerPosition and frontOfLocker private Vector3’s and making touchingLocker and insideLocker private Booleans set to false.

To make the locker interactable, I made the locker have two box colliders, one normal and another a trigger extended out in front of the cube, this will be used to determine the front of the locker.

I also had to change the code I originally had since I couldn’t use my OnTriggerEnter functions, I replaced though in the Update functions and tied the original changing layer code to an if (Input.GetKeyDown(KeyCode.Space) as well as checking if the player was touching the lockers collider.

I made a second if statement checking if the player also was not inside the locker, then, the following would happen:

• insideLocker = true;

• gameObject.layer = 2;

• locker.SelfDestruct(this);

• frontOfLocker = transform.position;

• transform.position = lockerPosition;

The former two parts of code would set insideLocker to true and sets the layer of the player to “Ignore Raycast” as mentioned in my 20/04/2021 entry. The latter two parts of the code references the front of the locker to recognise its transform.position as the player and then making the players transform position the same as the lockers.

The locker.SelfDestruct(this); is a reference to a function in my HidingObject script which I developed later down the line to fix an issue I encounter that I reference later in which the lockers collider stops the player from entering the locker because of its solid collider. I also had an else function that did the following:

• transform.position = frontOfLocker;

• gameObject.layer = 8;

• insideLocker = false;

• locker.RestoreCollider();

The first bit of code would put the player outside of the locker where the front of the lockers transform was and set the players layer to the “Player layer”. To fix the issue of my player not entering the locker, I went to the HidingObject script and referenced the players movement script and the collider of the locker and made a function called SelfDestruct(Dungeon\_Player\_Movement p). In the function, I had set the lockers collider to false to deactivate the solid collider and made a StartCoroutine(“Die”) which references the IEnumerator Die()to destroy the game object once the a few seconds pass. I then made a second function RestoreCollider() which would reactivate the solid collider of the locker once called. I called that function in my players movement script once the player leaves the box.

With that finished I went forward to finalise my rhythm game, I realise that I needed the player to travel to each lane horizontally rather than go to them via coordinates. I changed my previous script to once a key is hit, they would move a set amount instead of going to that coordinates, I also set a limit on how far they can go in that direction by writing.

if (Input.GetKeyDown (KeyCode.A))

{

x -= 5f;

if (x < -5f)

{

x = -5f;

}

}

This would be repeated for the D key except the X would go to 5x rather than -5x.