Exercise 3 - Shooty Gun 3D:

Welcome to Exercise 3! Here You'll be taking the skills You learned in the previous exercise and translating them to 3D. You'll be learning about creating a 3D Player Character, generating Random Objects, Input.GetAxis, the Cursor class, and will be experimenting with Game Design for the first time to create Your first 3D Game!

Chapter 1 - Shooty Gun... But in 3D!:

a. *Import Models* - Instead of Sprites, import 3D Models from Sketchfab for Your Objects, as demonstrated in Class. Import 3D Models for a Player Character, a Bullet, and a Monster. In addition, add a Plane Object as the floor.

b. *Set Scene* - Set the Scene with the Player and the Bullet on the Plane.

c. *Attach Camera* - Attach the Camera to the Player's Head.

d. *Implement Camera Rotation according to Mouse* - The AimScript should now be two separate scripts, AimXScript and AimYScript. The AimXScript should be attached to the Player, while the AimYScript should be attached to the Camera. You don't need to implement a separate Gun Object. The Scripts each individually calculate two values, x and y, using Input.GetAxis("Mouse X") * sensitivity * Time.deltaTime and Input.GetAxis("Mouse Y") * sensitivity * Time.deltaTime respectively, and store them as floats. Make sure to use Math.Clamp to keep y between -90 and 90! Use transform.rotation = Quaternion.Euler(0, x, 0) and transform.rotation = Quaternion.Euler(-y, 0, 0) respectively in the two scripts.

e. *Implement 3D Movement* - Furthermore, adjust Your implementation of the MovementScript and PlayerControlScript to three Dimensions. Instead of the old left and right movement, You want to make sure that forward motion is done with the W Key and uses Quaternion.Euler(0, x, 0) * Vector3.forward * speed to get the current direction the Player is facing. Additionally, jumping should be done with KeyCode.Space now.

g. *Implement Shooting* - Attach the ShootyGunScript to the Camera this time, and make the Bullets shoot out of the Camera, for accurate shooting. You shouldn't have a Gun in this Scene.

h. *Finalize and Debug* - Make sure All the other necessary steps, such as assigning Rigidbodies and setting Active statuses, have been carried out.

Press Play. Do You have a 3D ShootyGun Game? Good! We're ready for the next step.

Chapter 2 - Spooky Scary Skeletons:

a. *Import Monster* - Import the Monster into the Scene. Make it a perfect copy of the Player except:

1) No Camera or ShootyGunScript

2) No PlayerControlScript

b. *Implement Monster's Basic Movement* - Instead, add a MonsterControlScript, which invokes the MovementScript much in the way PlayerControlScript does, but instead of receiving Player Input, the MonsterControlScript will implement motion in the following way: The Monster should always be moving forward at a constant speed.

c. *Randomize Monster Strategy* - A float, monsterTimer, will increase Every frame according to Time.deltaTime, until it reaches the size of a predetermined Serialized Parameter monsterTime, at which point it will reset to zero. Any time this happens, use Random.Range(0, 1f) to randomize a float between zero and one, and if it is smaller than a predetermined float Serialized Parameter chasePlayerChance, set a bool chasePlayer to true, otherwise set it to false.

d. *Implement Both Strategies of Monster* - When chasePlayer is set to true, rotate the Monster around the y axis at a constant rate to face the Player, and when chasePlayer is false, rotate the Monster around the y axis randomly using Random.Range (do not set the rotation, use Transform.Rotate). Use Serialized Parameters in All relevant places.

Press Play. Is it hesitantly chasing You? Good!

Chapter 3 - War... War Never Changes:

a. *Add LifeTotalScripts* - Now, add a LifeTotalScript to the Player and Monster, which holds a float lifeTotal and initialLifeTotal (which start out the same, call lifeTotal = initialLifeTotal in Start), and calls Destroy(gameObject) if the lifeTotal is smaller than or equal to zero.

b. *Allow Monsters to Damage the Player* - Add a DamageScript to the Monster, which will use OnCollisionEnter to determine, via an appropriate Parametrized tag, whether the Monster is touching the Player, and in any frame it is, it will call GetComponent<LifeTotalScript>() from collision.gameObject to call a function from LifeTotalScript's API to damage the Player (meaning, reduce lifeTotal) by an amount determined by the Parametrized Random.Range(minDamage, maxDamage). After performing damage, the DamageScript should call Destroy(gameObject) to eliminate the Monster.

c. *Allow Bullets to Damage the Monsters* - In addition, add a DamageScript to the Bullet, making sure that the Parametrized tag causes it to damage the Monster on collision instead of the Player!

d. *Make Monsters Get Red when Hurt* - Add a new TintRedScript to the Monster, which calls in Start GetComponent<Renderer>().material = Instantiate(GetComponent<Renderer>().material) to ensure We can tweak the Color of the Monster's Material without affecting the Material of others, and which, in Update, gets lifeTotalRatio = lifeTotal / initialLifeTotal from GetComponent<LifeTotalScript>() and uses GetComponent<Renderer>().material.color = lifeTotalRatio * Color.White + (1f - lifeTotalRatio) * Color.Red to tint the Color of the Monster redder and redder according to the amount of lifeTotal it has left.

e. *Proliferate the Monster* - Copy the Monster a whole bunch of times (say, 100 times?) and spread them around the Scene. (Bonus: Instead of using Copy+Paste in the Editor, create a MonsterSpawner which uses a MonsterSpawnerScript to Instantiate a predetermined amount of Monsters at random locations throughout the Scene! (Above ground!))

Press Play... Can You beat it? Is it too easy? Play around with the Serialized

Parameters until it's fun to play!

***(Bonus - Will be part of next Week's Assignment)* Chapter 4 - High Score!:**

a. *Add Your First Text* - Add a TMP Text Object, with the appropriate Canvas. Play around with the size and margins until the Text appears at the top-left corner of the Screen.

b. *Implement the Life Bar* - Add a LifeBarScript to the Text Object, which contains the Player's LifeTotalScript as a Serialized Parameter, and updates the text to reflect how much lifeTotal the Player has left. (Hint: use GetComponent<TMPro.TextMeshProUGUI>().text = "HP: " + lifeTotal.)

c. *Implement the Score Bar* - In addition, add another Text in the top-right corner of the Screen which keeps track of the score, with a relevant ScoreScript, which stores as an int the Player's current score, displaying it much in the way We used the other text to display the lifeTotal. Add comboTimer and comboBonus Serialized Parameters in the ScoreScript, adding Time.deltaTime to comboTimer Every Frame. Add an addScore function to ScoreScript. Whenever addScore is called, it increases the score by 1f + comboBonus / comboTimer (We want to give more points for each Monster the faster the Player kills it) and resets comboTimer to zero.

d. *Make Killing Monsters Add Score* - Make sure each LifeTotalScript checks, before calling Destroy(gameObject), whether or not it has the Monster tag. If it does, it will call the addScore function in the ScoreScript (it keeps the ScoreScript Object as a Serialized Parameter).

e. *Implement Highscore* - In addition, add a third Text just under the score Text, with HighScoreScript attached. This Script should use, in Start, PlayerPrefs.HasKey("HighScore") to check whether a high score exists. If it does, it should retrieve it using PlayerPrefs.GetInt("HighScore") and display it in the Text. Any time the Player dies or kills the last Monster, this Script should save the high score using PlayerPrefs.SetInt("HighScore", score). (Hint: HighScoreScript should get score from the ScoreScript. How should these two Objects communicate?)

f. *Implement Victory/Failure Text* - Finally, add a large fourth Text in the center of the screen and set its Active status to false. Make it so that when the game ends, if it ends because the Player dies, the HighScoreScript sets the fourth Text to Active and changes the text to "You lose!" or some variation thereof, and if it ends because All the Monsters are dead (You need to figure out how to check that efficiently), the HighScoreScript sets the fourth Text to Active and changes the text to "You win!" or some variation thereof.

Play the Game a couple of times! Can You beat it? Is it fun? What sort of high score did You get? I hope You enjoyed this Assignment!

Good Luck!!!