Gladiator Production Log

To start I needed to get the first-person movement. I created a plane and a capsule by clicking the plus in the Hierarchy and under 3d objects. I positioned the plane to 0, 0, 0 and the capsule to 0, 1, 0 to position it in the middle and on the ground. I attached the camera to the capsule by dragging it onto it and set the position to 0, 1, 0.

A screenshot of a video game

Description automatically generated

I now wanted to get the movement made. I created a new folder in assets called “Scripts” and created a new C# script in there called “Movement”. I found a helpful website to help me with my FPS code (<https://sharpcoderblog.com/blog/unity-3d-fps-controller>). To stop me from stealing the code directly, I only took what I needed from the code. For example, I took the simple movement and crouching myself since it isn’t included. To add the script to the capsule, I dragged the script into the component section, and I edited the speed there to get the right settings and then imported them into the script. Some may be the same as the website, but this is because of how perfect it is. Also, I added a “Character Controller” to the capsule. I tested out the code and then explained what each line did using comments. You can see this state in “Update 1.00” along with this production log and the start of the GDD and a readme file explaining the GitHub

Next, I added sprinting and jumping. After trying around, I couldn’t make my own original code, so I turned back to the website for support. I learned how to make a floor without the use of Collisions and Tags, which was a lot easier, and the use of ? in Boolean statements. This update is called “Update 1.10” in the GitHub with the code being explained in the comments.

I made a quick mood board and then headed to 3DS max to create my arena. I wanted to make metal gates where the enemies would spawn, lifting and allowing them to walk out and try to get the player. I started off small and created the arena area before I make the seating area. Since this is based around programming, I will not add textures or make it look as pretty as I would if I had more time, or if it was more relevant.

The model took around 5 minutes, and this was the result. I made it basic as the programming is more important than how the level looks, but I still really like how the arena looks. With this I decided to start making my character model. This will be a low poly character like someone out of Super-HotA picture containing shape

Description automatically generated

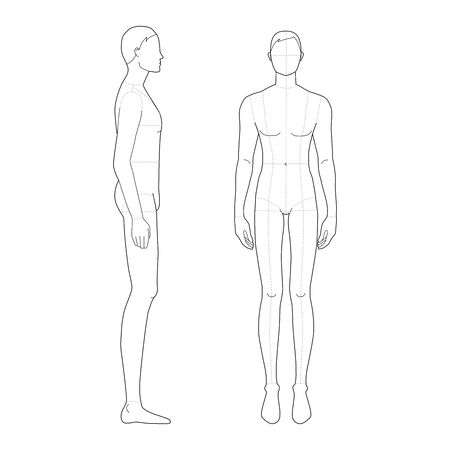
To put this model into Unity, I exported it as a FBX and then dragged it into the assets folder in a new folder called “models”

A screenshot of a video game

Description automatically generated

This can be found in Update 2.00. This means it is the end for movement as crouching isn’t a necessity, however, if I have time I will come back to it.

Using this reference image (from <https://www.stocklib.com/media-141450106/fashion-template-of-standing-men-9-head-size-for-technical-drawing-with-main-lines-gentlemen-figure-front-and-side-view-vector-outline-boy-for-fashion-sketching-and-illustration.html?keyword=standing%20mannequin>) I will make my main character. The picture is a standard person that could easily be transformed into a female to add enemy variety using a 50 / 50 chance. Once the model was made and I had added bones to it (using this video: <https://www.youtube.com/watch?v=Im8saU3k52E>) I saved my work in github under “Update 2.10”



Diagram

Description automatically generated

Next, I skinned the object. This was easily done. In the modifier list, I choose “skin” and added all the bones in the body. Now that they connected, I created a walking animation and exported it as a .FBX file and placed it into unity assets. Simple update. This is saved under “Update 2.15”

Next, I created a punching animation in 3DS max and exported it. I created a new script called “Animations” which will be attached to the player. The code is explained in the script but I will go through it quickly here.

Text

Description automatically generated

I assigned the variable “animator” to the Animator component. In start we defined “animator” with the Animator attached to the player with GetComponent.

Next, I check if the Animator is there with the if statement “If(animator != null). Null means the variable isn’t assigned and ! means not. That code reads “If animator is not equal to null”

I created a bool called “moving” and assigned it to true is W, A, S or D is being pressed. The bool “Moving” in the animator is equal to the bool we created in the script “moving”

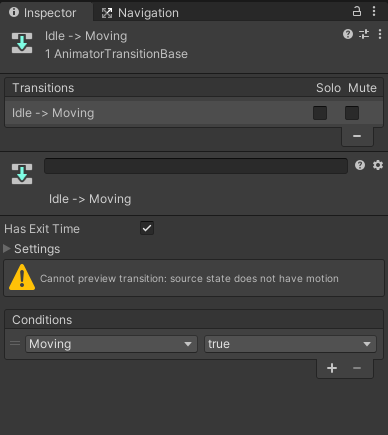
Finally, if the left mouse button is clicked “if(Input.GetMouseButtonDown(0))” the animator plays the animator “Attack”

In the animator, from the entry phase I have an empty state called “Idle”. If I have a idle animation it would play that animation. From “Idle” I made a connection to “Walking” and from walking back.

A screenshot of a computer

Description automatically generated with medium confidence

The animator will move from idle to moving if the bool “Moving” is set to true from the animator. I created the bool in the animator by using the parameters menu where you can also add floats, ints, and triggers. By clicking on the transition from “Idle” to “Moving” I can set it to only transition if the bool in the Animator “Moving” is true, using conditions.

Graphical user interface, application

Description automatically generated

In a new layer called “Attacks” I simply have the state Attack going into idle since we play it when we are in the script.

A picture containing graphical user interface

Description automatically generated

Next, I created two scripts called Human and Player, then placed them in a folder called “Entities”. These are the controller for each individual entity type. I also made two GameObjects called “EntityManager” and “RoundStats” so that I can access their members in different scripts. Inside RoundStats I have one script called “RoundManager” which handles how many enemies there are meant to be, the round you are on and where the enemies spawn. EntityManager holds both Player and Human script.

In the human script, I set the entities health, speed and strength. To access the round stats, I get a GameObject (GameObject.find()) and then get the RoundManager script from that. (roundmanager.GetComponent<>). Finally in this script I destroyed the Enemies if their health is less than or equal to 0 and decreased the EnemyCount (For UI and Spawning) by 1.

In the Human script I did more. This is because it is the player and was more complex. Firstly I assigned numbers to health, strength and stamina. I also created a variable for the script Human called hu for later. I have a bool that is called “isAttacking” set to true when you click left mouse button, however, that was soon scrapped. The most important part was the collisions. Since I am using box colliders, I used OnTriggerEnter instead of OnCollisonEnter. It checks if the tag is Enemy and if it is, it will decrease its helf by our strentgth (hu.health -= strength). We assigned hu by using hu = other.GetComponent<Human>(); other is assigned to whose ever box collider we hit.

Another important script is the Ent\_Behaviour. This script will determine how the human will move, attack and their health. For now, all they do is move the AI. First of all, I need to start of with making variables for everything I need. I have one for a GameObject called Player, NavMeshAgent called Enemy, a GameObject called EntityManager and finally the script Human called hu. I assigned Enemy to the GameObject we are using as an enemy, and player to the person you control using Player = GameObject.FindGameObjectWithTag("Player"); In update, I used Enemy.SetDestination to make the enemy go towards the player.

I imported another model, the same as the player, and saved it as a prefab called “Exported Enemy”. At first their model was the wrong way round so they were walking backwards, but it is now fixed. The player has the components: Rigidbody, Box Collider, Animator, the script called “Ent\_Behaviour” and a NavMeshAgent.

In a folder called Resources, I have a prefab call “Text Round” (Renamed to Text shortly after). I needed to make some ui. In a script called “scr\_GUI”. In this script I positioned it and changed its text. The code is very simple and explained in the script. In the script I use instantiate, I get components, change the text, reposition them and also change their sizes.

To get the enemies to spawn, I made an array of vector3. I next simply instantiated the enemies at a random point by using

ran = Random.Range(0, Spawn\_Locations.Length);

Vector3 zomSpawn = Spawn\_Locations[ran];

This code simply picks a random value from 0 (the first value in the array) to the end (.length is the end of the array) and then set the spawn location as a vector 3.

Also, in “RoundManager” I have a function called “newRoundHealth”. If the round is less than 10, it simply adds 100 health to their total health. If the round is greater than 10, it will add 100 health and also multiply it by 1.1, which eventually will add up to mega health. I used the formula from the game “Call of Duty” to spawn the enemies. Before round 12, the formula doesn’t work meaning I had to code them in myself. I set them in an array called “PreRound12”. If the round is 12+, it uses the formula: 0.000058 \* R^3 + 0.074032 \* R^2 + 0.718119 \*R + 14.738699 rounded to nearest int (R = round). In code I used MathF and the int variable “Round” to declare the EnemyCount.

EnemyCount = Mathf.RoundToInt(0.0842f \* (Mathf.Pow(Round, 2)) + 0.1954f \* Round + 22.05f);

This is the end of “Update 2.20”

In “Update 2.30”, I created the health bar. I couldn’t figure out how to do this no matter how hard I tried. I used a well know Game Developer “Brackeys” tutorial on health bar. In addition, I took the sprite of the health bar from him aswell. In his description of his video, he has “► All content by Brackeys is 100% free. We believe that education should be available for everyone.” Meaning I am allowed to use his sprite without any copyright consequences. (<https://www.youtube.com/watch?v=BLfNP4Sc_iA>)

In the script “HealthBar”, I have accessed different components and simply used a slider to be equal to the value of the health. The code is explained in the script.

Back in “scr\_GUI” I have created a new text called “Points” which will display the points that you have displayed to spend on weapons and potential powerups.

In the Human script I created so that when they are killed, they award the player with 300 points.

After playtesting, I realised that the hitboxes were slightly off. I created a box collider around the enemy model to try and fix this. I simply dragged out each end slightly to just cover the body

A picture containing text, light

Description automatically generated

Finally, I made it so if the player reaches 0 or less health, the player is killed. Currently it just destroys the gameObject.

That is the end of “Update 2.30”

In “Update 3.00” I added health auto regeneration. I did this by using IEnumerators. I did this because you can add a useful method called “WaitForSeconds(int)” which simply does at it says.

Firstly, I checked to see if the health was less that the max health. If it was I would then check to see if they was already healing by using the bool “Healing”. If it was false it would start the healing process by using “StartCoroutine(heal())”. A coroutine is like a function and can be paused and resumed using WaitForSeconds and other methods. A good example can bee seen on the unity website (<https://docs.unity3d.com/ScriptReference/MonoBehaviour.StartCoroutine.html>)

The Coroutine will set healing to true to avoid any duplicate healing (which was a original problem) and then it waits for two seconds. After the two seconds it will heal you by 10 and then start another coroutine called healcool. This simply waits 5 seconds then sets healing back to false to allow you to heal if needed.

I created a new scene called Dead. The ui I will explain later but when you die it will load you to that scene. In addition, it will keep two gameobjects with certain values from the other scene. To do this, I used DontDestroyOnLoad(GameObject).

|  |
| --- |
| if(health <= 0) |
|  | { |
|  | Destroy(gameObject); |
|  | SceneManager.LoadScene("Dead"); |
|  | DontDestroyOnLoad(em); |
|  | DontDestroyOnLoad(rm); |
|  | } |

I also added what each line of code does to the animator script since I wont be using it anymore.

In the Human script, I created it so the player has a chance to dodge the punch. For this is once again used IEnumerator. I used one called attk, which will start if you enter their collider area. Once you are you have 2 secconds to escape it. If you are still inside their collider, it will then check if it is on a cooldown the same way I did for health. If it isn’t it will attack and deal damage, if it is they wont do any damage till the cooldown of 3 seconds is over. Finally, when they are killed it adds 1 to a new int called kill used for stats and the dead menu.

While testing I noticed again the collider wasn’t just right. Like with the enemy, I slightly adjusted the players hitbox and it immediately felt smoother, but still not perfect. I wanted to get other stuff done before I decided to change it anymore.