# PRISON RUNNER DEVELOPMENT LOG
# BY CHARLIE COLL

# PRE-PRODUCTION

This process begun with the creation of this checklist, which I made to organize the development of the game into sections and the order that they will be implemented in

## A1 Free runner modifications checklist

Art – .Mood boards for new character design

.Design new character

.2-4 new obstacles mood board, then designs

3D - .New character asset

.New character animations (Static running to the right, jump, slide, death)

.2-4 new obstacle assets

.Change background image

Programming – .Modify player controller script to get an input for sliding, add an is-sliding variable that is true when you slide and set it back to false after a 2 second co routine, play the slide animation when its input key is pressed and the is-sliding variable is false

.Modify the spawn manager script to randomly select an obstacle from an array of obstacles and spawn one every 2 seconds (At their correct positions)

# ART – MOOD BOARD

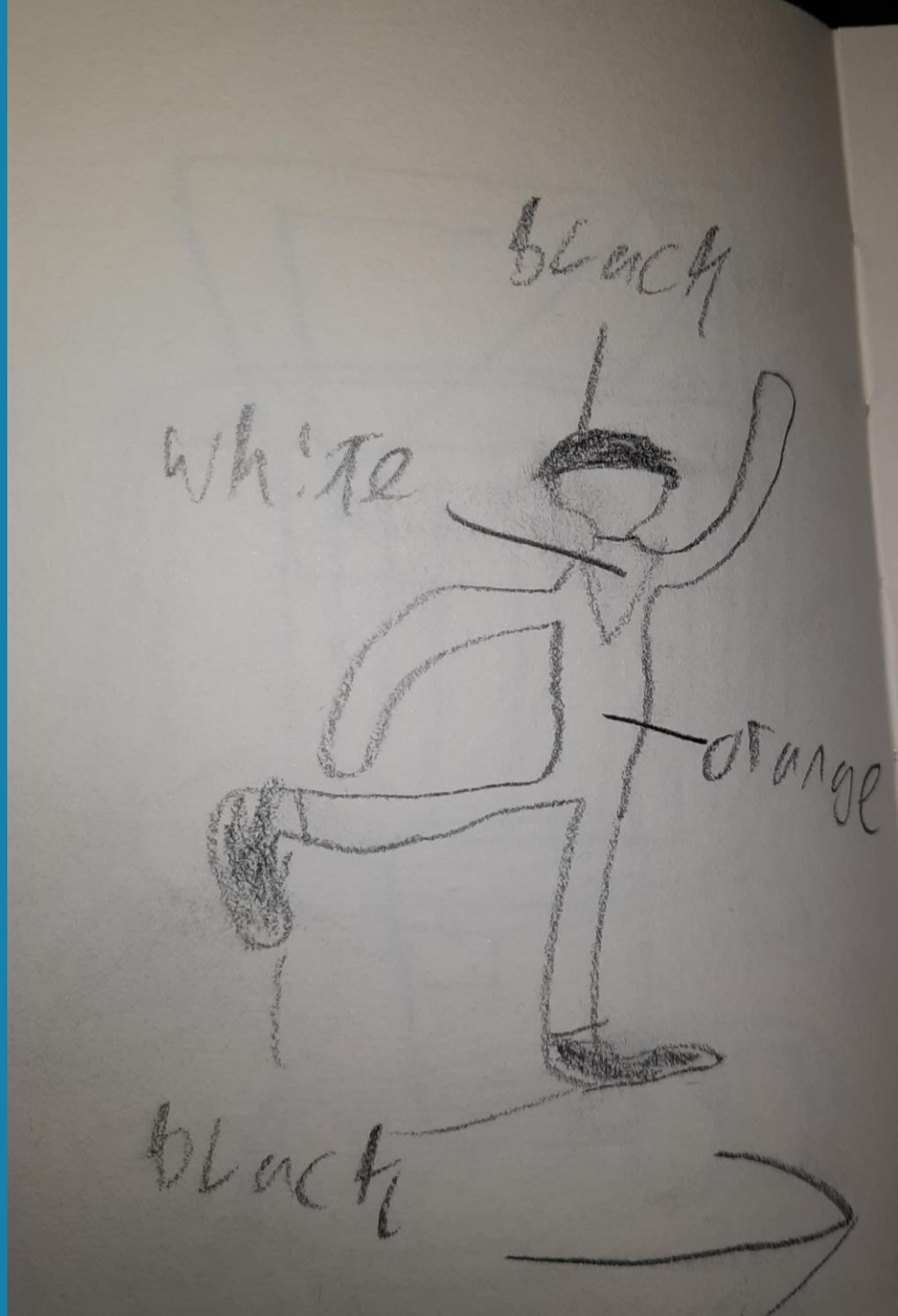
I started planning out the art and visual style/theme of the game by creating this mood board for the character and obstacles – this will be used for reference when creating my own assets during the next process. I chose the two images from a movie I've watched at the top of the mood board for my character and turret obstacle inspiration as this movie originally made me choose the theme of escaping prison. The theme of a prison made me think of the ps3 and psvita game "I must run", so I found images of the gameplay and its obstacles and put these into the middle of the mood board. The bottom of the mood board is full of images that inspire and reference obstacles for my game, they're all barriers and signs that would be enforced to stop someone instead of normal objects you'd find on the street as I wanted the game to feel as if the all obstacles that you're jumping over have been setup by the police in a desperate attempt to capture you.

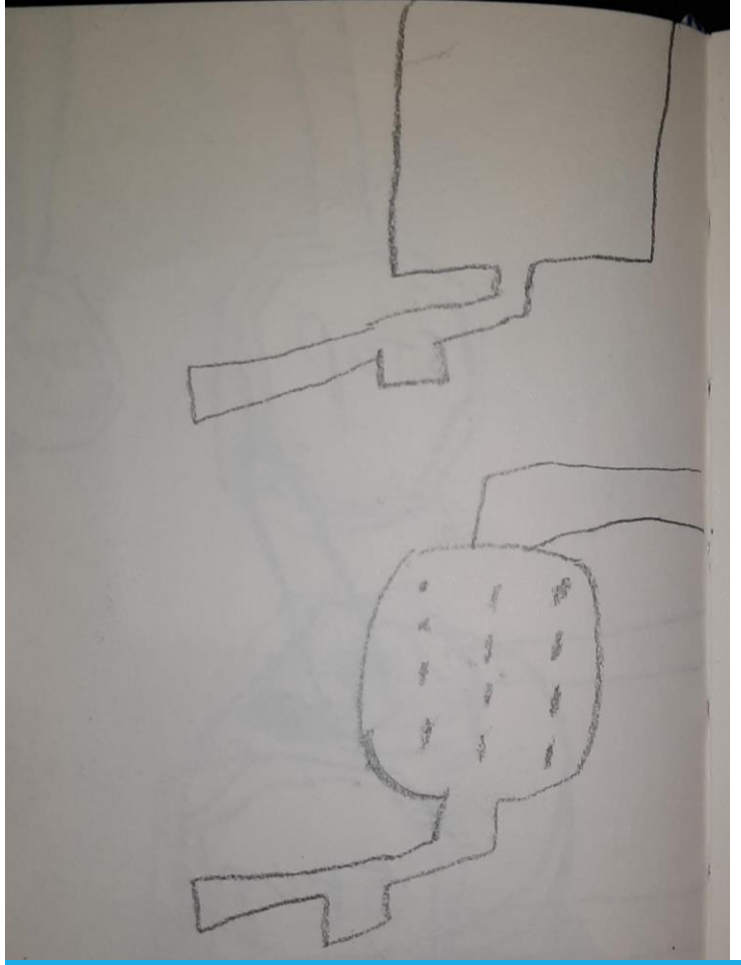


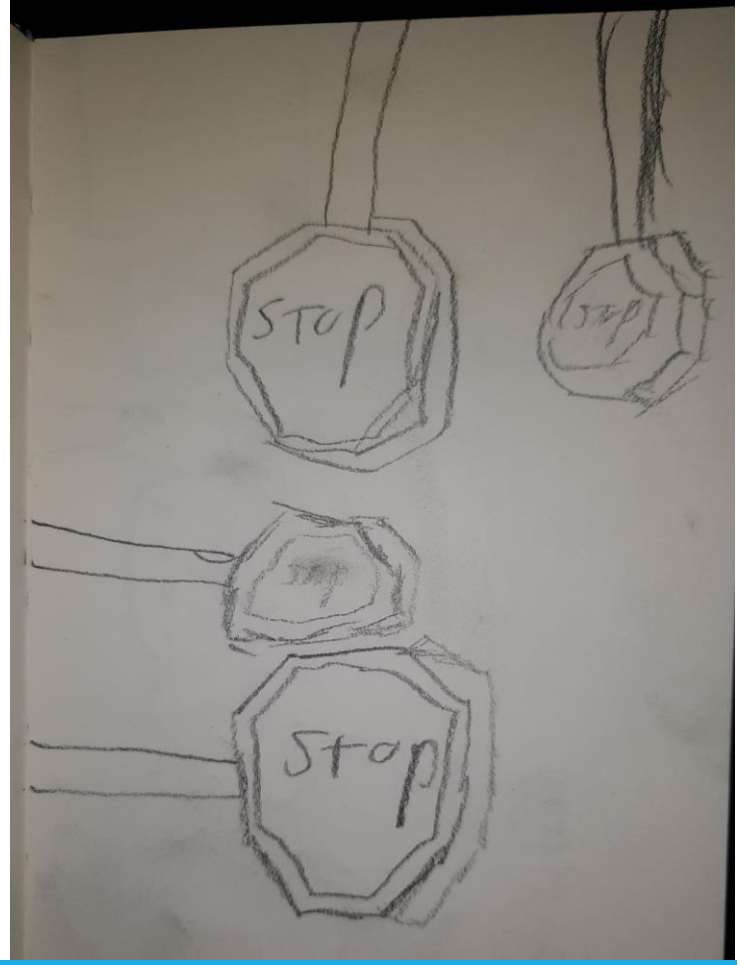


# ART – CHARACTER CONCEPT DESIGN

Following the guidelines that I had setup with the mood board, I sketched out this concept design for the prisoner character of the game – I knew I wanted the character to be wearing an orange jumpsuit and to be quite basic/anonymous, so I only dedicated a short amount of time to this concept art as I had the vision for the character (developing the character fully in 3D later)

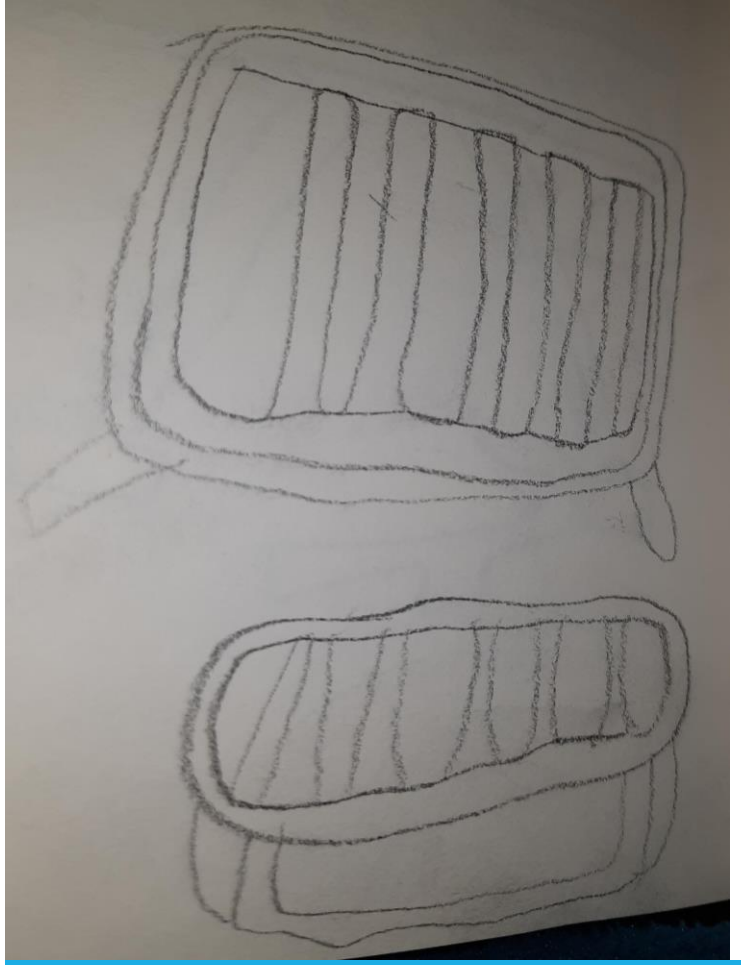







# ART – OBSTACLE CONCEPT DESIGNS

This slide shows the concepts for the hanging obstacles that I planned to add to the game, sketching out different variations of the obstacle that I can choose from when making the final 3D model later

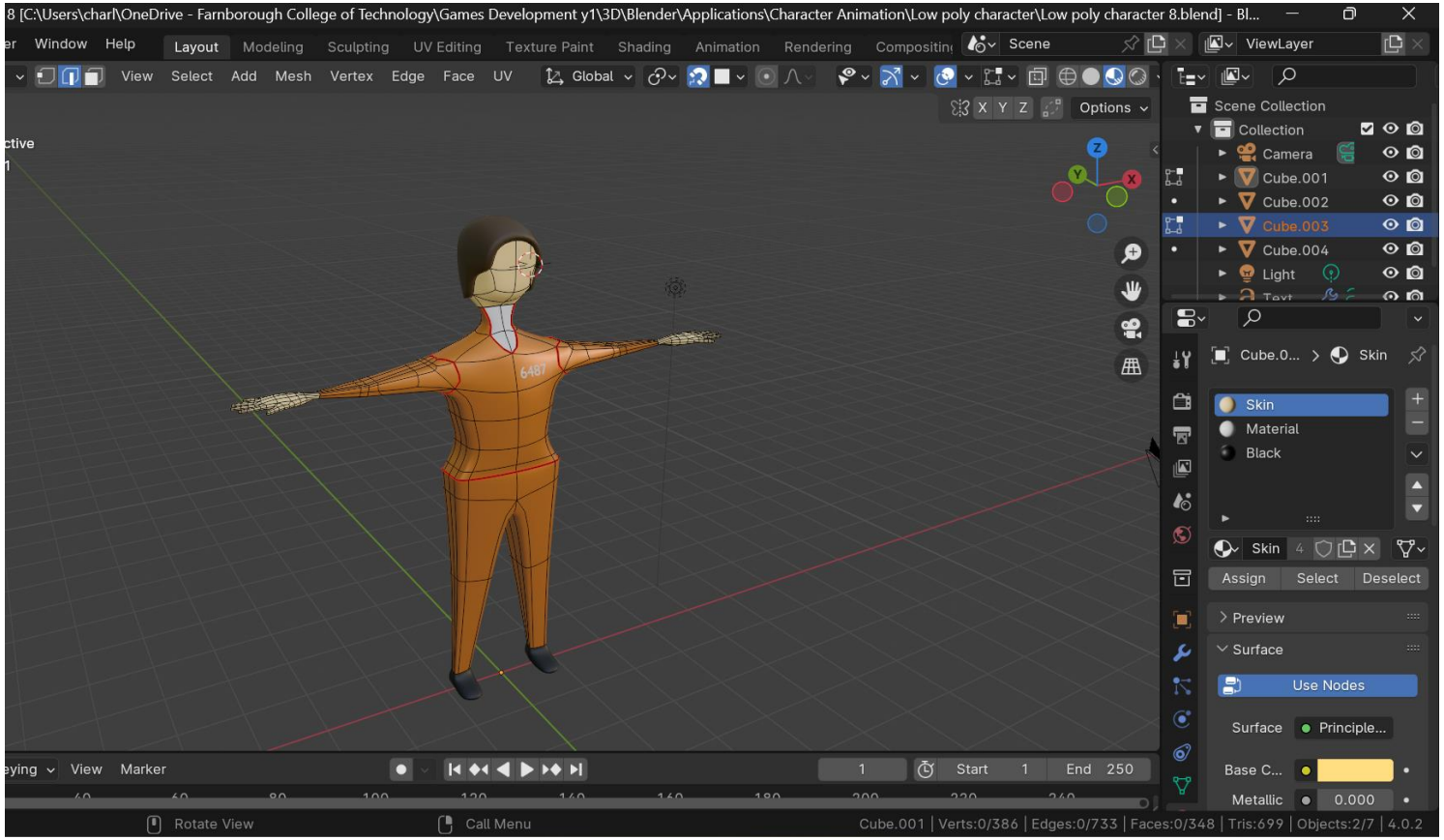



# ART – OBSTACLE CONCEPT DESIGNS

These are the ground obstacle concept designs, which again are original designs that are inspired by the mood board that I made prior to this

# 3D – CHARACTER ASSET

Now I had the mood board and concept design, I made this low poly character in the top right via box modelling alongside the scale and rotate tools, and the mirror modifier. Once the basic shape was complete, I applied the mirror modifier and a subdivision modifier to smooth out the rough edges and angles. The bottom right shows a new blender file I made after receiving feedback on the hands of my character and recommendations on how to improve it, where I created a hand using the reference image seen behind the model in the viewport and the same methods as the character model itself, that I will import into the character's file and add to the model.



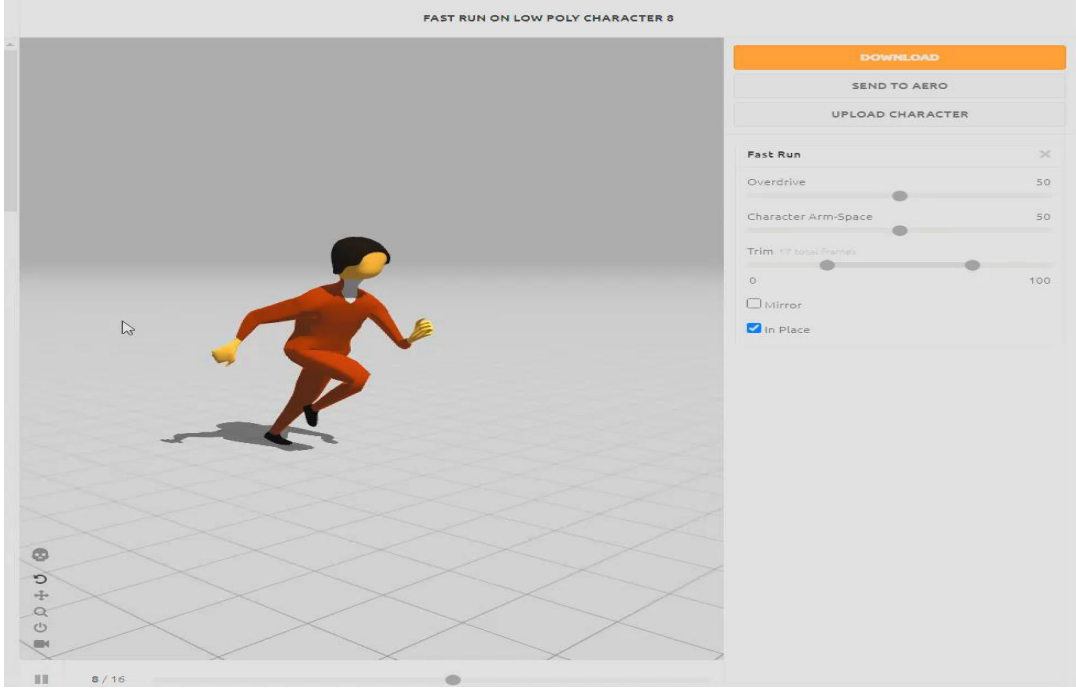

# 3D – CHARACTER ASSET

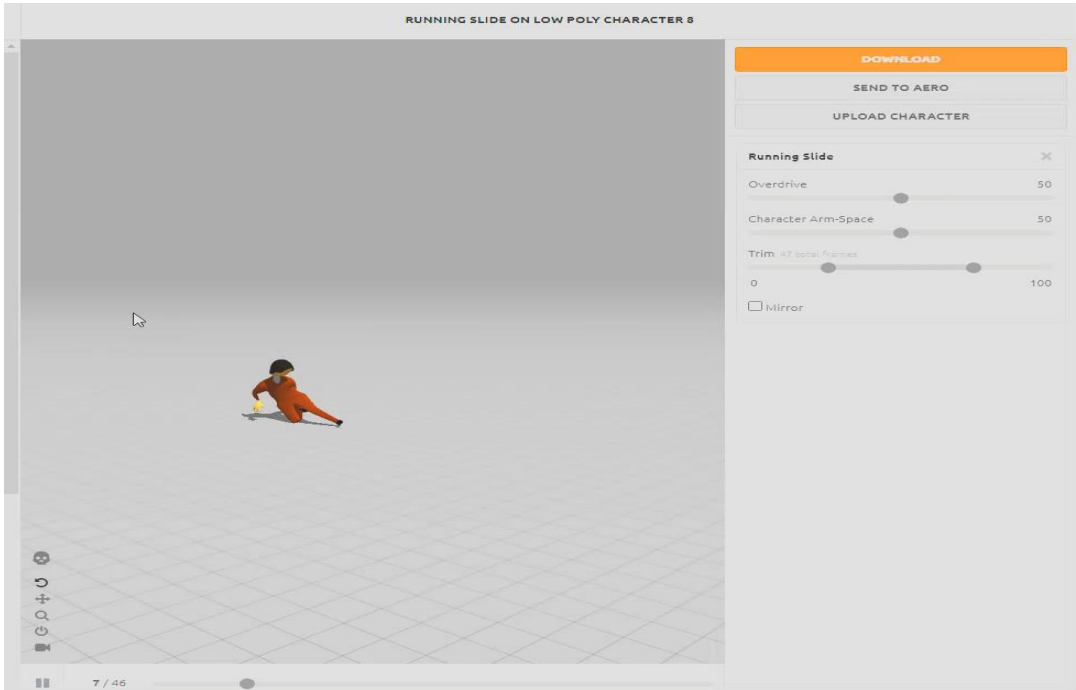
Here is the character model at its final stage: with the new hands imported in and attached via the bridge tool, hair via separating faces by selection off the head and applying a solidify modifier to them, materials applied to the model in sections defined by the marked seams, and a number on the outfit of the prisoner via adding text and attaching it to the character with the shrink-wrap modifier

# 3D - CHARACTER ANIMATIONS

After the character was uploaded into mixamo as an FBX, I was able to auto-rig the character and export animations of the character that I had selected without skin for application in unity

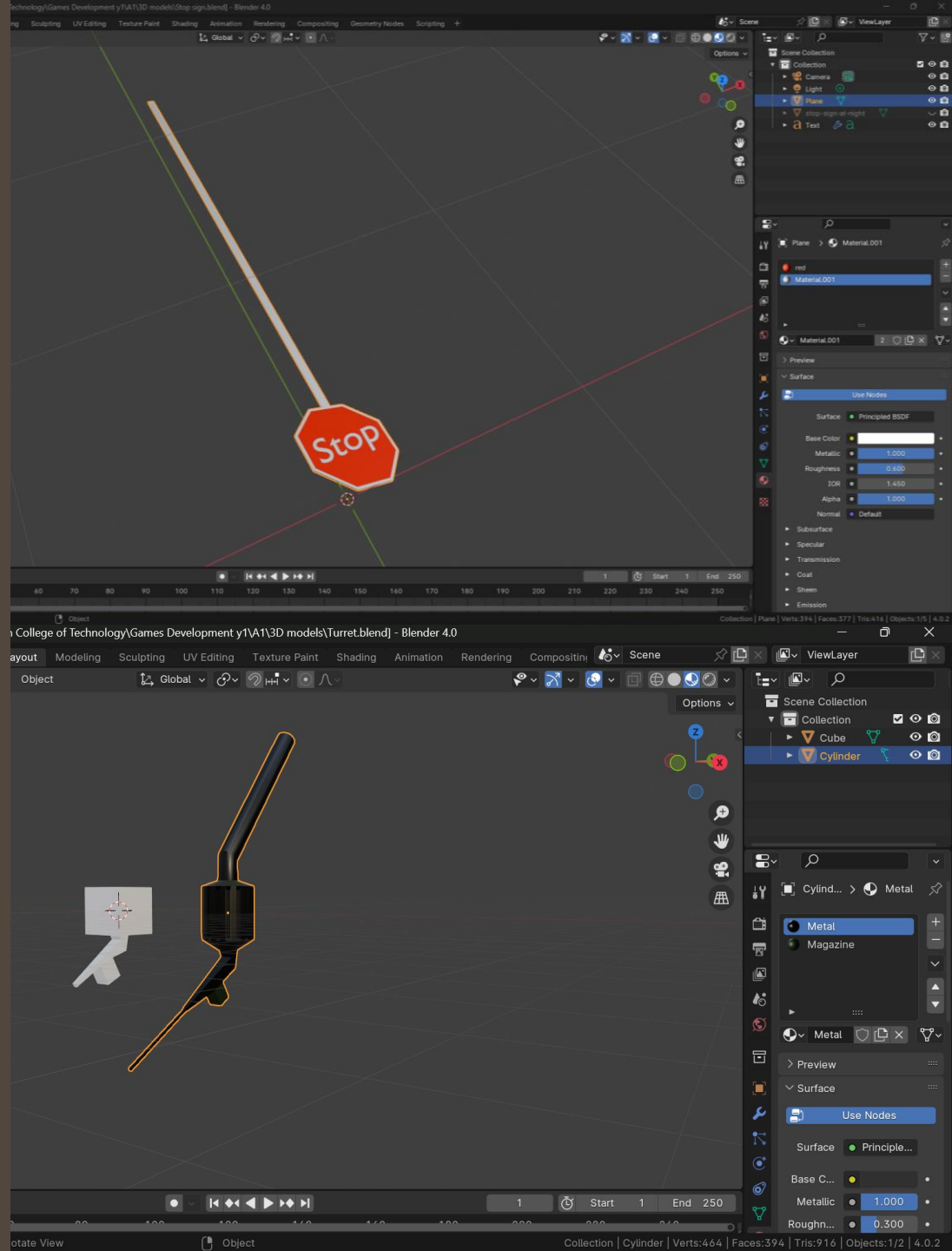
On the right are two playable videos of the running and sliding animations selected in mixamo.



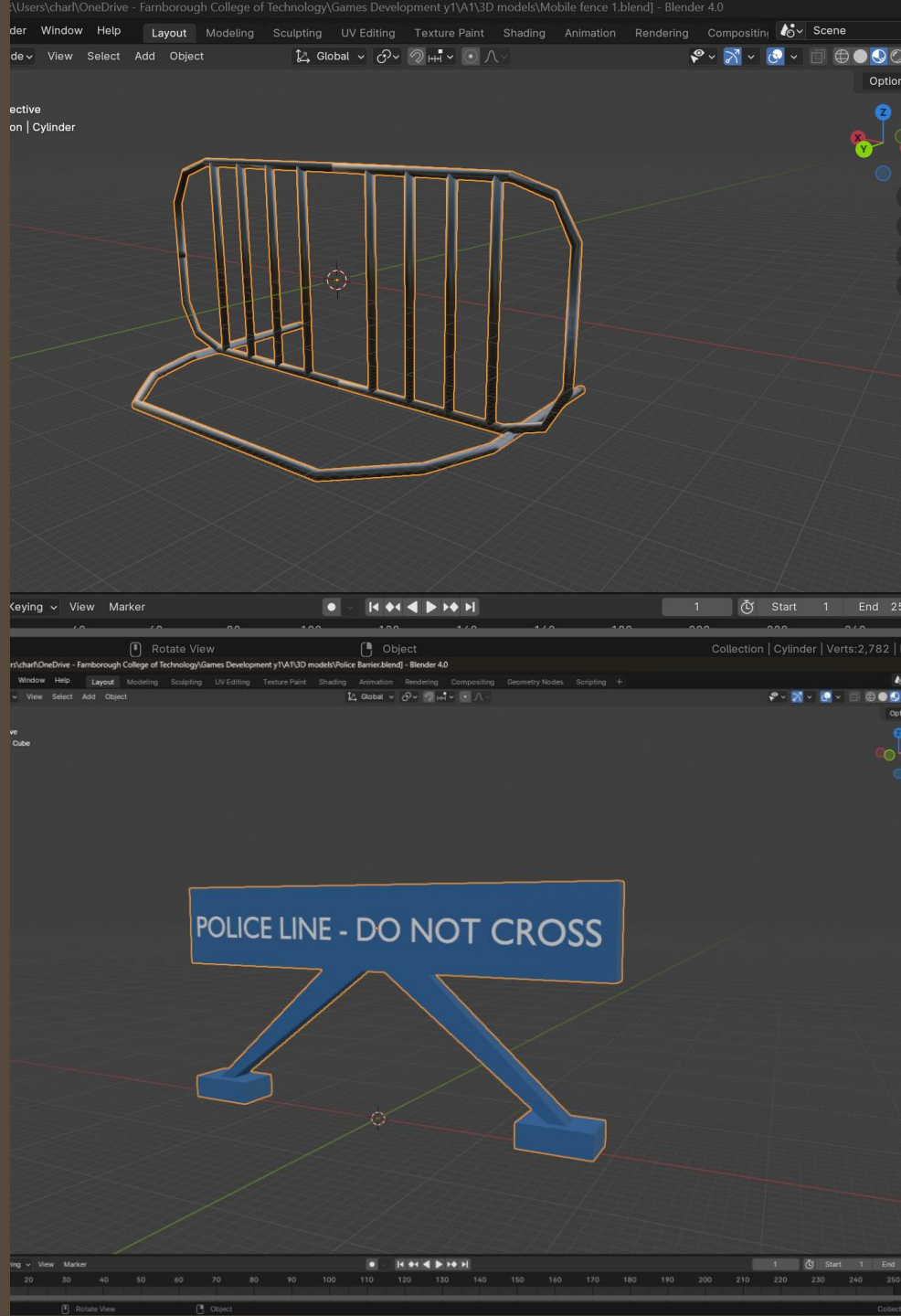
# 3D – OBSTACLE ASSETS

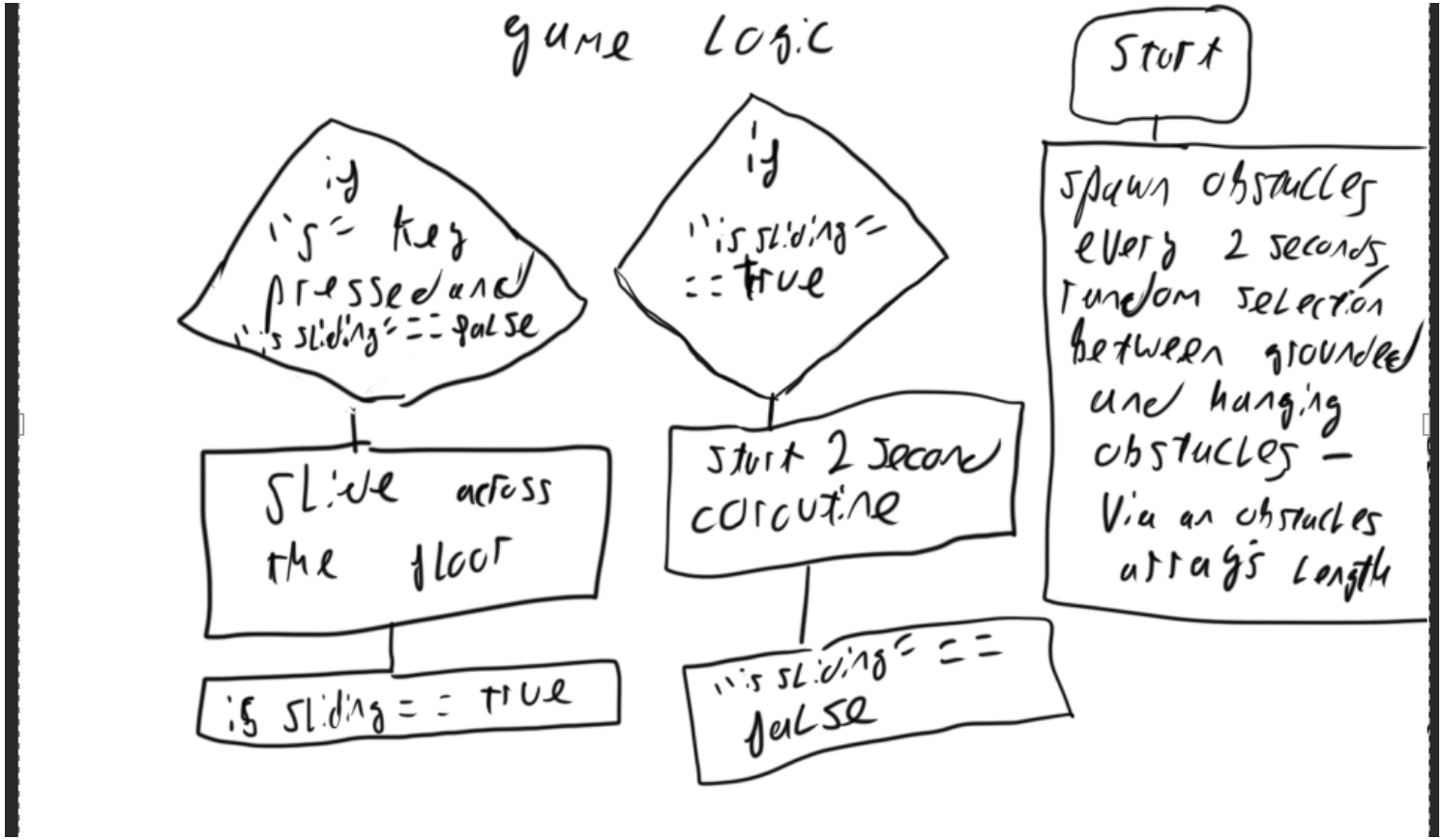
The hanging obstacles are presented on this slide, which I made from the concept designs I sketched previously. The stop sign was created by tracing vertices (from a plane object merged at the center of the reference image to create a vertex) around the original mood board image, then extruding the outline down to add depth to the sign, and then extruding out the pole, adding materials, and adding text to the sign just as I added text to the character. The turret was created from a cylinder that I extruded and scaled the top and bottom of, scaled in faces in intervals around the cylinder, and creating the gun barrel and mount at the top via extrusion.



# 3D – OBSTACLE ASSETS

The grounded obstacles are shown on this slide – made from the same extrusion techniques as the hanging obstacles and the use of the mirror modifier. The police barrier used the bridge tool to join the sign to the supports on the ground, via faces that were created using the inset and move tools that were mirrored. The mobile fence consists of mirrored cylinders that have a metallic material applied to them.



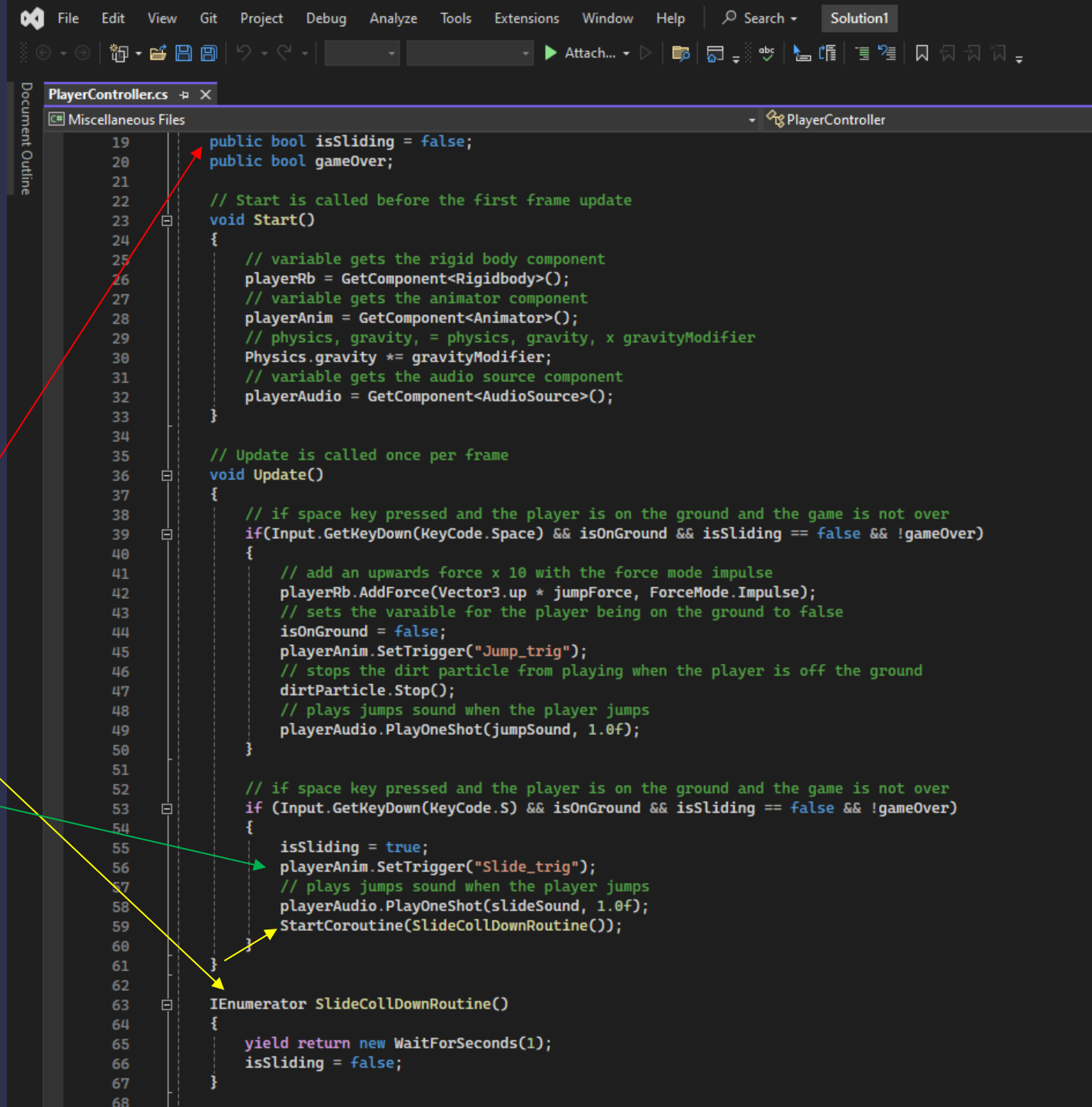



# PROGRAMMING — LOGIC FLOWCHART

I began the creation of my new game mechanic with this flowchart that I made in photoshop, planning out what decisions and actions must take place in the new code that I will create to implement the previously made 3D assets, animations, and new game mechanic.

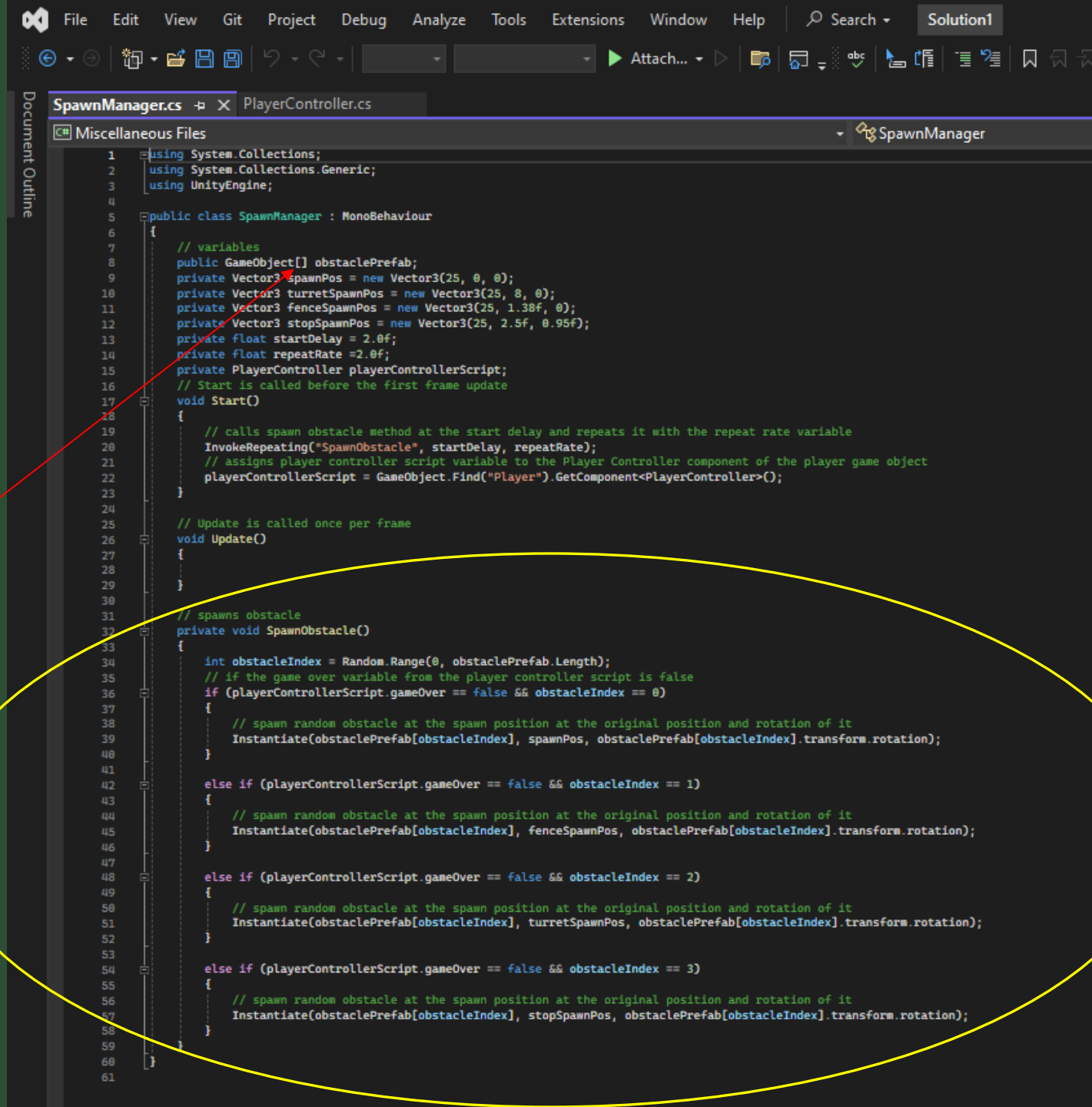
# PROGRAMMING – MODIFIED PLAYER CONTROLLER

Following the flow chart and pre-production checklist, I coded the sliding mechanic into the game with an is-sliding variable that is true when the S key is pressed and is false after a second long coroutine, and by getting the input from the player to play the sliding animation and sound effect upon the S key being pressed.



# PROGRAMMING – MODIFIED SPAWN MANAGER

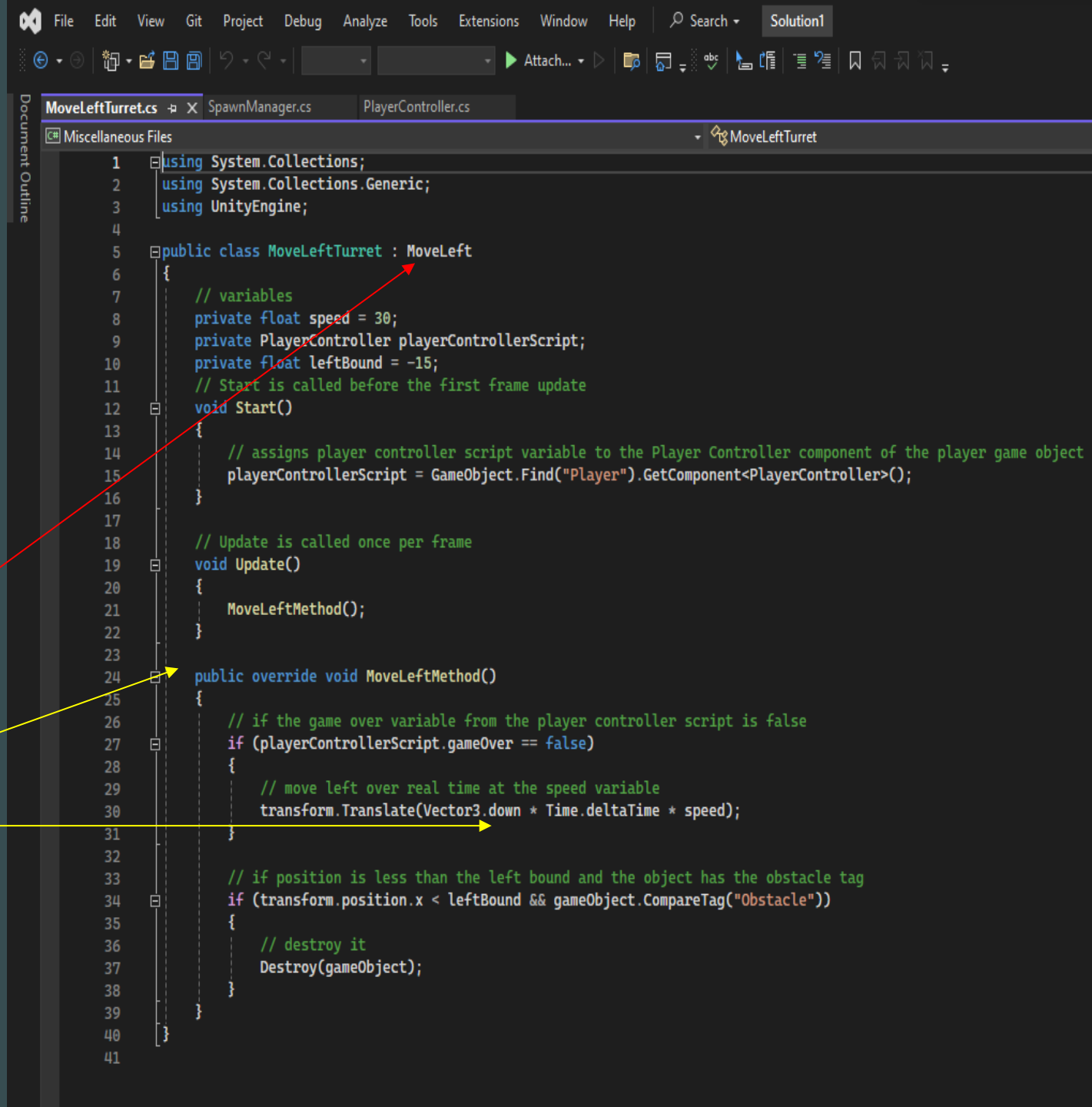
I changed the spawn manager script to allow all my obstacles to be spawned at random via an array, as the original game had a singular obstacle, and spawned them at their correct positions in the spawn obstacle method by identifying which obstacle is being spawned in and spawning it at the right spawn position.

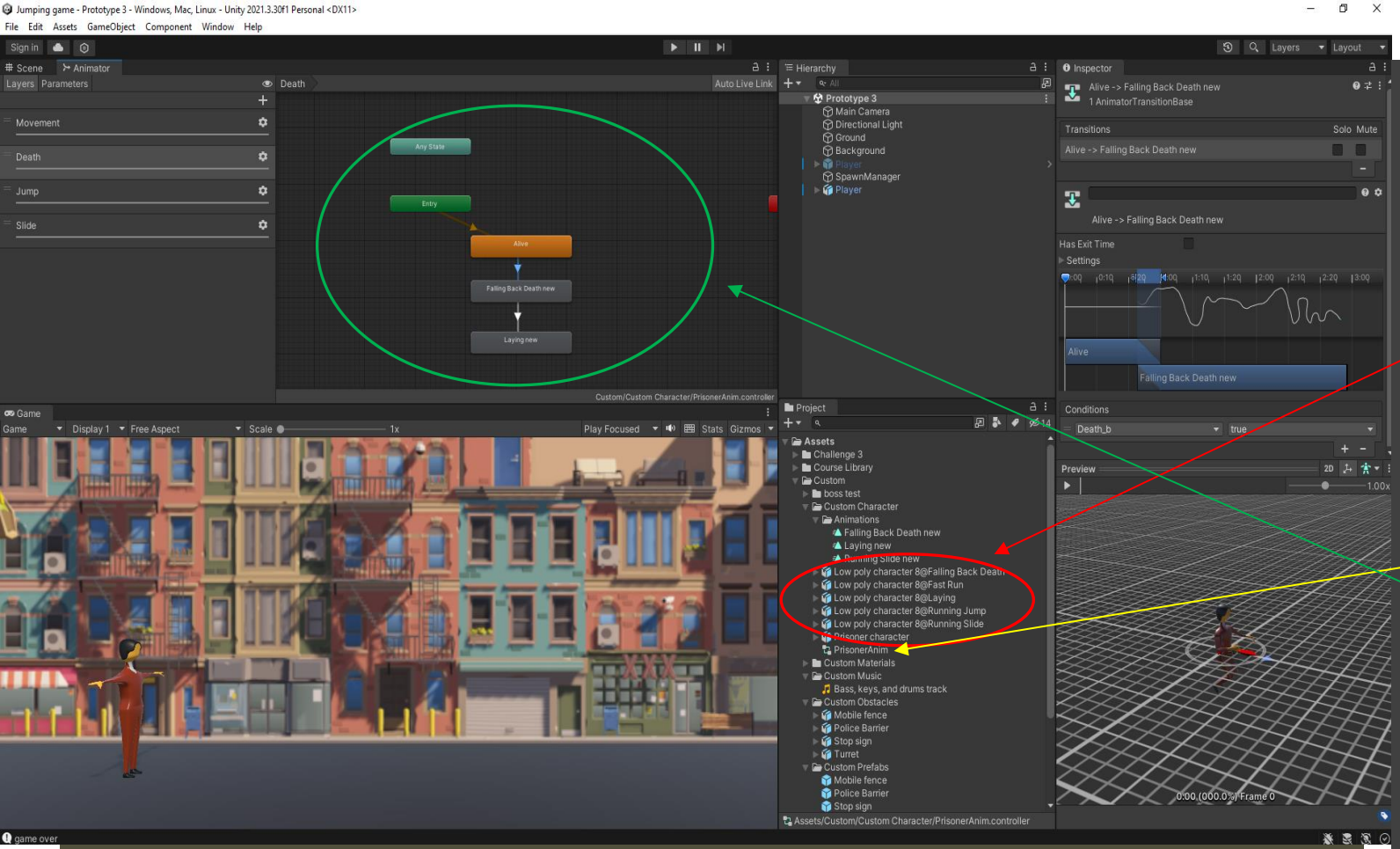



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class SpawnManager : MonoBehaviour
{
    // variables
    public GameObject[] obstaclePrefab;
    private Vector3 spawnPos = new Vector3(25, 0, 0);
    private Vector3 turretSpawnPos = new Vector3(25, 8, 0);
    private Vector3 fenceSpawnPos = new Vector3(25, 1.38f, 0);
    private Vector3 stopSpawnPos = new Vector3(25, 2.5f, 0.95f);
    private float startDelay = 2.0f;
    private float repeatRate =2.0f;
    private PlayerController playerControllerScript;
    // Start is called before the first frame update
    void Start()
    {
        // calls spawn obstacle method at the start delay and repeats it with the repeat rate variable
        InvokeRepeating("SpawnObstacle", startDelay, repeatRate);
        // assigns player controller script variable to the Player Controller component of the player game object
        playerControllerScript = GameObject.Find("Player").GetComponent<PlayerController>();
    }

    // Update is called once per frame
    void Update()
    {

    }

    // spawns obstacle
    private void SpawnObstacle()
    {
        int obstacleIndex = Random.Range(0, obstaclePrefab.Length);
        // if the game over variable from the player controller script is false
        if (playerControllerScript.gameOver == false && obstacleIndex == 0)
        {
            // spawn random obstacle at the spawn position at the original position and rotation of it
            Instantiate(obstaclePrefab[obstacleIndex], spawnPos, obstaclePrefab[obstacleIndex].transform.rotation);
        }

        else if (playerControllerScript.gameOver == false && obstacleIndex == 1)
        {
            // spawn random obstacle at the spawn position at the original position and rotation of it
            Instantiate(obstaclePrefab[obstacleIndex], fenceSpawnPos, obstaclePrefab[obstacleIndex].transform.rotation);
        }

        else if (playerControllerScript.gameOver == false && obstacleIndex == 2)
        {
            // spawn random obstacle at the spawn position at the original position and rotation of it
            Instantiate(obstaclePrefab[obstacleIndex], turretSpawnPos, obstaclePrefab[obstacleIndex].transform.rotation);
        }

        else if (playerControllerScript.gameOver == false && obstacleIndex == 3)
        {
            // spawn random obstacle at the spawn position at the original position and rotation of it
            Instantiate(obstaclePrefab[obstacleIndex], stopSpawnPos, obstaclePrefab[obstacleIndex].transform.rotation);
        }
    }
}
```

# PROGRAMMING – TURRET SCRIPT

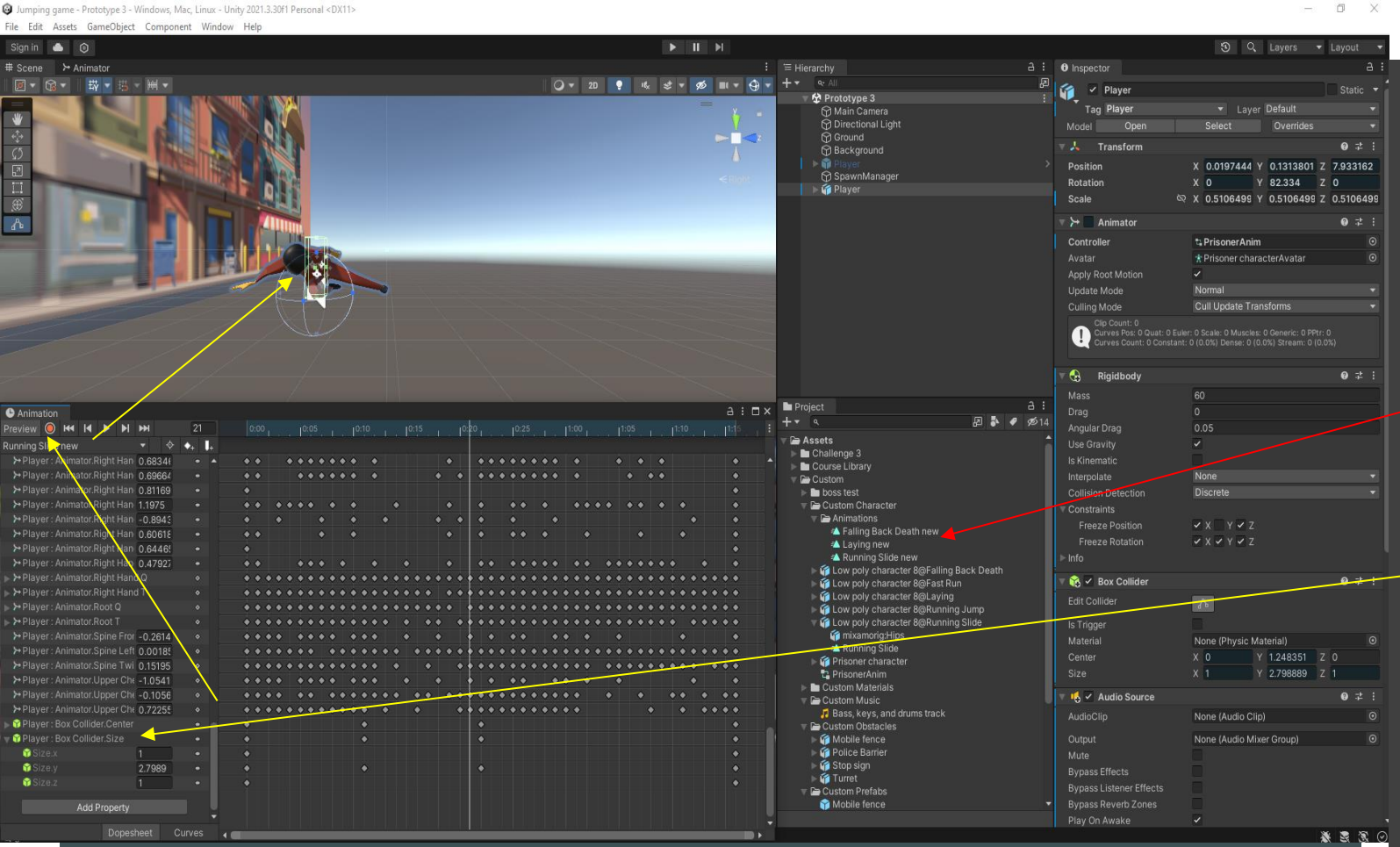
This is a new script that I created to get the turret obstacle moving in the right direction, as the axes are different between the unity and blender programs, via the use of inheritance from the original move left script – I called the overridden move left method that contained the changed axis in update.





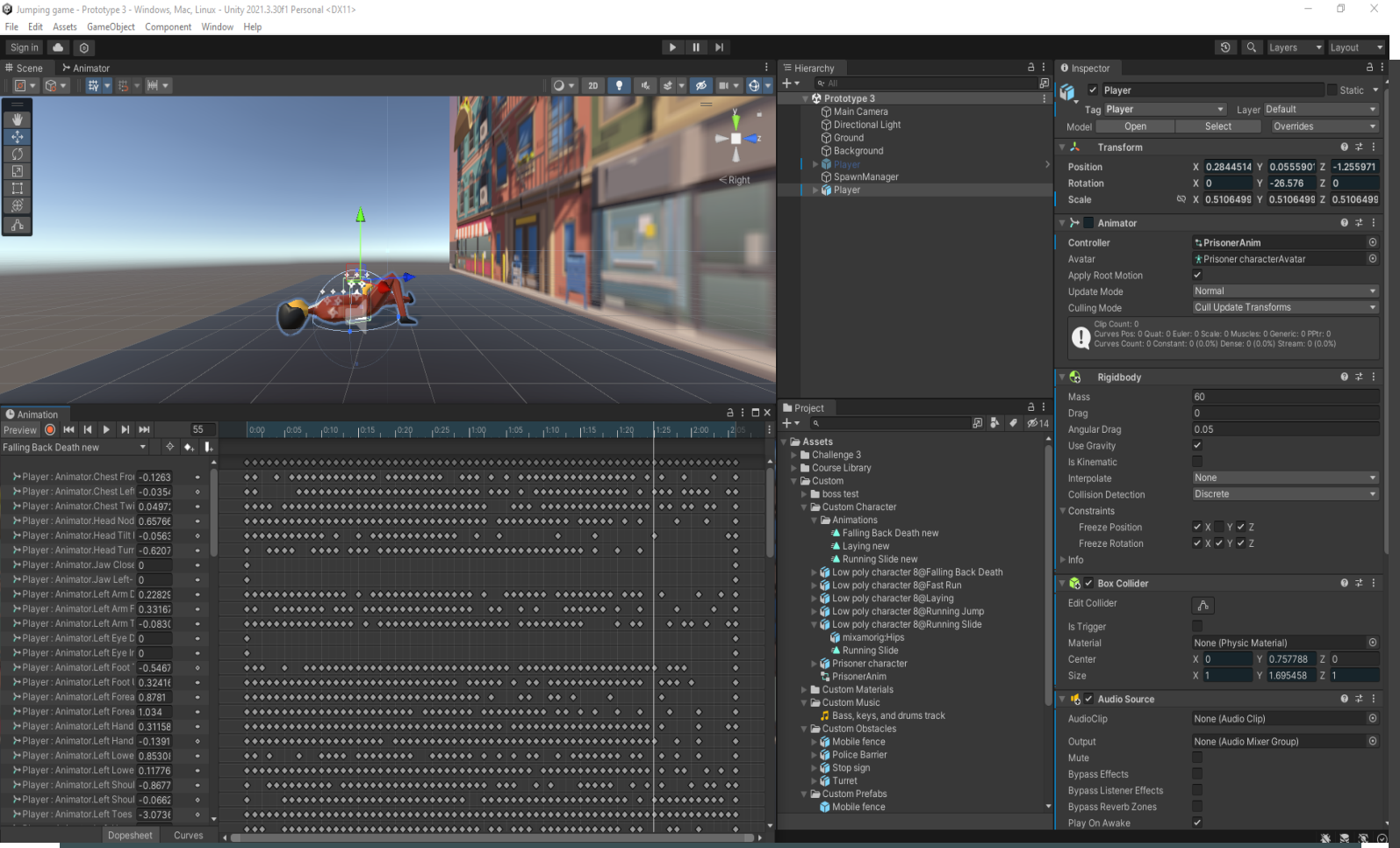

# PROGRAMMING – ANIMATING IN UNITY

To get the animations working with the player controller script, first I imported the FBX of the character (and the animations) and gave the new player all the necessary components and rotation/scale changes, next I created a humanoid avatar rig from the model and created the animations' rig from this, then I created an animator controller called "Prisoner Anim" and added the animations into the tree of animation states (as seen in the animator window in the image on this slide) with the same parameters as the script that became the conditions between changing states of animation (To go to the death animation, the death bool must be true) with some having "has exit time" unticked so the animation instantly plays. The fast run and laying idle animations also have loop time enabled, so they continue to loop instead of playing just once.



# PROGRAMMING – ANIMATING IN UNITY

I also had to figure out how to change the size of the players box collider during the slide animation so this new game mechanic would work, this was done by duplicating the animation from the project window (as the original animation is read-only) and replacing the old animation state in the tree with the newly duplicated one – after that I was able to add the box collider property to this animation in the animation window, and record keyframes of the changing height of the box collider that I adjusted at different stages of the animation, I finished this process by changing some of the root transform positions and rotations of the animations to stop them from rotating unnecessarily and to make the character actually slide along the ground instead of in the air (as the root transform position on the y axis was not based upon the characters feet)

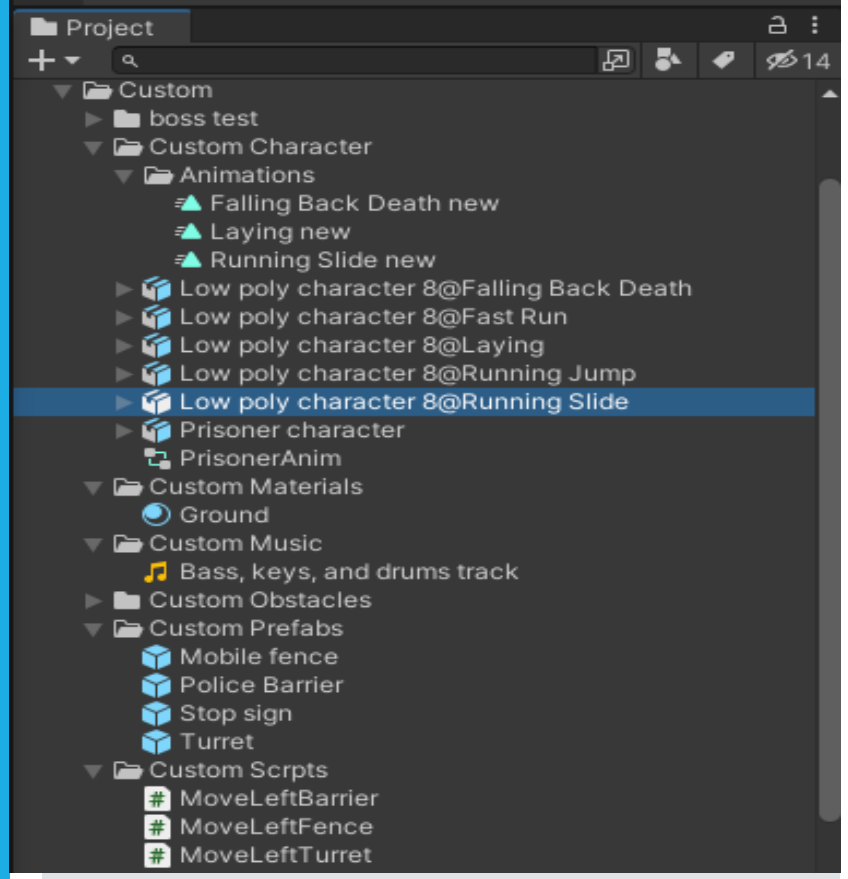


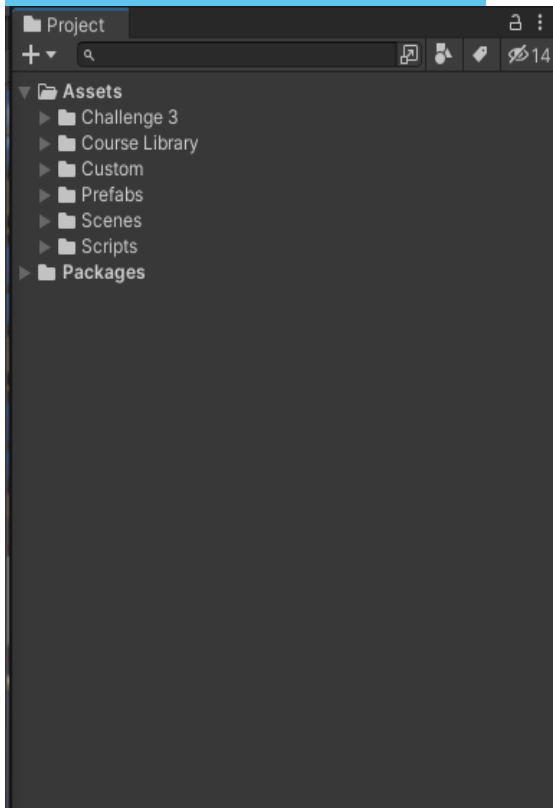
# PROGRAMMING – ANIMATING IN UNITY

I did the same to the death animation, as I had received feedback from a peer about this animation and how the collider repeatedly hits the hanging obstacle while falling to the ground – so I made sure to record new box collider keyframes in the animation window to get the collider down to the ground when you hit an obstacle, preventing the collider from colliding with the obstacle while the character is falling to the floor.

# PROGRAMMING – ORGANISATION AND NAMING IN UNITY

These screenshots show the organization of my project window in unity, presenting the “custom” folder I created within the assets folder that organizes my character animations, materials, music, obstacles, prefabs, and scripts into their own easily accessible sections.



# FINAL PRODUCT

As a result of all the previous productions, the game is now finished and contains my own custom music (that I made in logic pro with the built-in drums, and real keyboard and bass guitar instruments), animated character, obstacles, sliding mechanic, and random obstacle spawning.

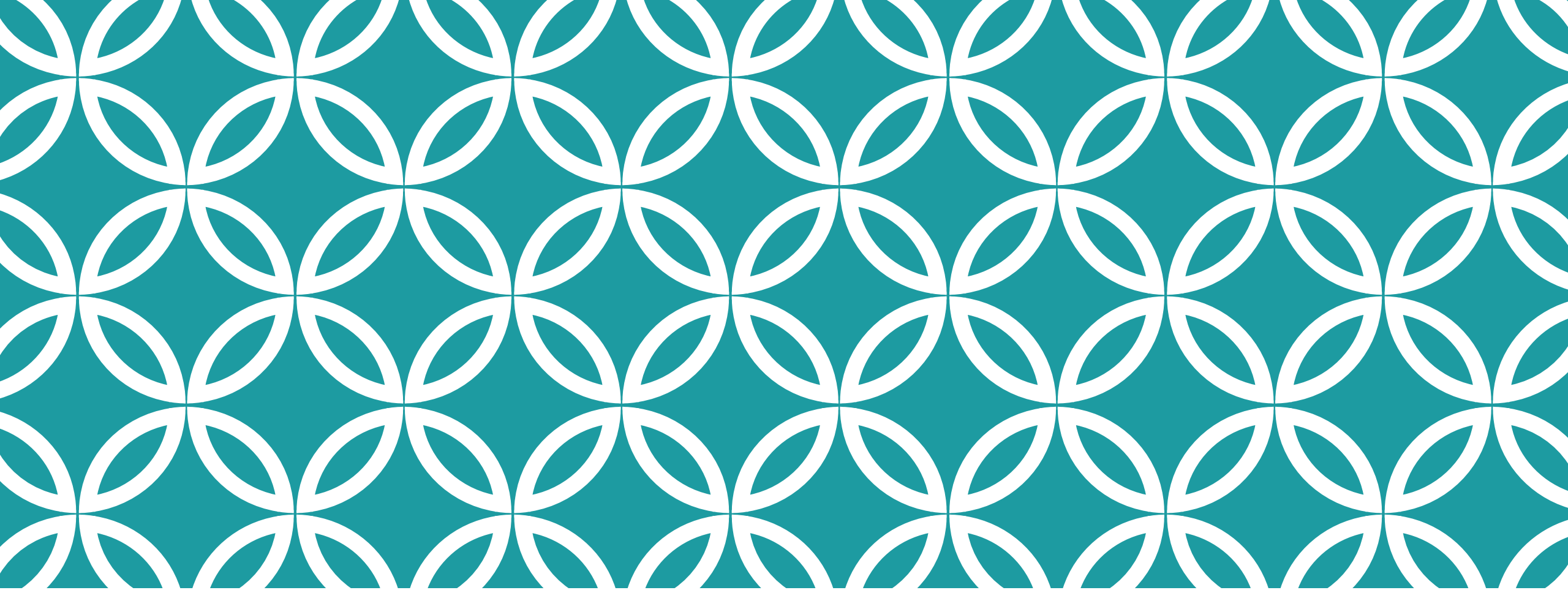
On the right is a playable video of the finished game (with audio).

# PEER FEEDBACK

One of my peers gave feedback of the game at its final stage:

Points for improvement – Could use some more obstacles, the character could have a face, and the jump feels unnatural (this is due to the jump force and gravity modifier variables, and I did adjust these a bit before the final recording that is featured on the previous slide)

Positive points – The custom music, sliding mechanic and its changing hitbox, and the existing obstacles



# THANKS FOR READING

By Charlie Coll