12.01.23 Naming Conventions

## PascalCase

First Letter of Each Word UpperCase

ie: RichardOfYorkGaveBulldogsIncredibleViolins

* Methods/Functions: PascalCase
* Scripts: PascalCase
* Property: PascalCase

## camelCase

first Word LowerCase, Other Words UpperCase

ie myVeryExcellentMotherJustServedUsNoodles

* Variables & Arguments: camelCase

8.28.23 Initial VR set up

* Follow PDFs to set up the VR game (found in folder with this doc)
  + Setting Up Unity for VR Development
  + Setting Up Unity for VR Development v3
  + VR Class Project- Part 1 (this includes setting up the player, teleportation area & XR Grab Interactable)
* Have Plenty of Time for updates
* *DO NOT INSTALL THE OCULUS PACKAGE FROM THE UNITY STORE.*
* Double check the rotation on the hands. (90 & -90 for left and -90 & -90 for right).

## InputData Script (because it is a pain to write)

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.XR;

public class InputData : MonoBehaviour

{

public InputDevice \_rightController;

public InputDevice \_leftController;

public InputDevice \_HMD;

// Update is called once per frame

void Update()

{

if (!\_rightController.isValid || !\_leftController.isValid || !\_HMD.isValid)

InitializeInputDevice();

}

private void InitializeInputDevice()

{

if (!\_rightController.isValid)

InitializeInputDevice(InputDeviceCharacteristics.Controller | InputDeviceCharacteristics.Right, ref \_rightController);

if (!\_leftController.isValid)

InitializeInputDevice(InputDeviceCharacteristics.Controller | InputDeviceCharacteristics.Left, ref \_leftController);

if (!\_HMD.isValid)

InitializeInputDevice(InputDeviceCharacteristics.HeadMounted, ref \_HMD);

}

private void InitializeInputDevice(InputDeviceCharacteristics inputCharacteristics, ref InputDevice inputDevice)

{

List<InputDevice> devices = new List<InputDevice>();

InputDevices.GetDevicesWithCharacteristics(inputCharacteristics, devices);

if (devices.Count > 0)

{

inputDevice = devices[0];

}

}

}

10.01.23 Unity Version Information

Make sure the unity version is 2022.3.5f1. This ensures that there are no VR property errors from the wrong version.

11.13.23 Headset Not connecting through Rift Link

It has nothing to do with the temperature of the PC as adding ice packs will not fix the issue (self -\_-). However, you can test the cable and make sure that enough data is being passed through it. If the cable is not the issue, perform a hardboot of both the headset and computer.

Headset:

1. Hold down the power button and volume down button for 30 seconds until you hear the boot-up sound
2. If the boot-up sound is not heard:
   1. Hold the power and volume (-) button down simultaneously until the boot screen loads on your headset.
   2. Use the volume buttons on your headset to highlight "Reboot Device" and press the power button.
   3. Use the volume buttons on your headset to highlight Yes and press the power button.

PC:

1. Hold down power button until computer turns off.
2. tap it again to turn computer back on
3. select continue on the BOIS screen
4. select restart on the WINDOWS DIDN’T LOAD CORRECTLY screen

If these don’t work, contact Meta support. <https://www.meta.com/help/support/?g1=5677250625703607>

11.13.23 Play Button is not Working with Oculus/Controllers do not appear

You can build the game to the oculus (.apk file) and still test your programming that way. However, if you do not install the Oculus Integration package you do not have issues with the play button. Try removing that package or building a new game without it.

10.30.23 Oculus Controller Resources

## Button names

Primary: left controller

Secondary: right controller

Index trigger: regular trigger

Hand Grip: Side trigger

## Unity References

<https://developer.oculus.com/documentation/unity/unity-ovrinput/>

this page tells you the different controller functions (built in)

near the bottom it talks about the controller set ups and the different button names

<https://developer.oculus.com/documentation/unity/unity-isdk-interaction-sdk-overview/>

this gives general info about the SDK interaction toolkit

<https://docs.unity3d.com/560/Documentation/Manual/OculusControllers.html>

gives “in depth” definitions of the controllers and buttons/button types

<https://docs.unity3d.com/Manual/VROverview.html>

less than helpful overview of unity VR functionality

<https://docs.unity3d.com/Packages/com.unity.inputsystem@1.5/manual/index.html>

small page talking about the new input system

this can be used in addition to or instead of the SDK vr interaction toolkit

11.6.23 Complete XR Origin Set Up Variant movement/position/transform info

THE Complete XR Origin Set Up Variant TRANSFORM/POSITION DOES NOT CHANGE. IF YOU NEED TO TRACK THE PLAYER POSITION/MOVEMENT YOU NEED TO TRACK THE XR Origin (XR Rig).

10.30.23 How to debug when building to headset: Console inside game

1. add a new canvas (WHILE THIS CAN WORK WITH A STAND ALONE CANVAS IN HIERARCHY, IF IT GOING TO BE FOR THE PLAYER IT IS ALWAYS BEST TO NEST IT IN THE PLAYER OBJECT)
2. add a Legacy text object (not textmeshpro) to canvas
3. set canvas to camera world space and adjust the view distance (I am currently running at .5 to 1)
4. write the script that adds text to the object

Using.UnityEngine.UI;

Public Text nameofVariable;

nameofVariable.text = whatYouWantToPrint;

Where: whatYouWanToPrint is --

1. [a function that results in an int or float].ToString()
2. [a function that results in a string]
3. “[what you phrase you want to print]”

Use + to add any of a-c together

EXAMPLE: “c” + [a].ToString() + b;

10.31.23 Controller Input to Bool

If you want to use a button press to set a bool as True or False, use the below function:

if (\_inputData.\_rightController.TryGetFeatureValue(CommonUsages.gripButton, out bool gripPressed))

{if (gripPressed){//do something here}

}

12.01.23 Tracking Controller-In-Space Data (X,Y,Z coordinates)

Add the following function to the script you want to track the controller. Make sure to call the function somewhere in the script. Location in this script acts like the HUD object (see 10.30.23 How to Debug when building to headset).

void recordControllerMovement()

{

\_inputData.\_rightController.TryGetFeatureValue(CommonUsages.devicePosition, out controllerPosition);

location.text = controllerPosition.ToString();

//Values will print in (X, Y, Z) format

}

You will also need to add:

using UnityEngine.UI;

public Text location;

private InputData \_inputData;

11.6.23 VR Player Gravity

You can add a rigid body to the player if you REALLY need to. However, you would need to add it to the parent object and it may cause in game issues. The XR player does not automatically come with gravity and you will need to program some if you don’t want the player to float/fall indefinitely.

This script resets the player position to an emptyobject called “resetBase” that you set in the inspector. It also prints the player location to your headsupdisplay. This does not cause the player to fall when not in contact with another object, it simply allows the user to allow the in game player to fall (using the down control on joystick) and if they “fall” below a the “ground” then move the player.

## Teleport Reset Player Script

using UnityEngine;

using UnityEngine.UI;

public class TeleportResetPlayer : MonoBehaviour

{

public GameObject player;

public float ground = 22f;

public Transform resetBase;

public Text HUD; //headsUpDisplay abbreviation which is the name of the text object

void Update()

{

HUD.text = player.transform.position.ToString();

if (player.transform.position.y <= ground)

{

HUD.text = player.transform.position.ToString();

player.transform.position = resetBase.position;

}

}

}

10.26.23 State Machine ACT() function

ACT(){} in a statemachine functions like update and runs continuously.

10.27.23 Calling A Method From Another Script (Referencing “External” Scripts)

myObject.GetComponent<MyScript>().MyFunction();

ZombieScript1.GetComponent<ZombieScript1>().SetExperience();

gameObject.player.GetComponent<PlayerHealth>().TakeDamage(1);

* my object can be the name of the script or the object the script is on
* Examples: ScriptName, ObjectName, other.gameobject.GetComponent, etc
* GetComponent<>(). Is a function and as such you do not change the name of
* GetComponent and CANNOT mess with any of the formatting
* <this is where the script name goes>
* <>().This is where the function name goes();
* functionName(this is where anything you need to pass into the function goes. –ie damage amount--);

11.30.23 Passing Bools between scripts

* You can check if a bool is true just by calling the name of the bool

if(boolianName){} = if(boolianName == true){};

* You can check if a bool is false by !

if(!boolianName){} = if(boolianName == false){};

* If you need to call a bool from another script:

if(scriptName.boolianName){} or if(scriptName.!boolianName){};

11.30.23 Passing Variables Between Scripts

If you need just to pull the variable from one script into another:

ScriptTwo.CheckVariable(passingVariable);

If you need to call the variable and set it to an amount, you need to build your second script like this:

public int variable = 100;

private ScriptOne scriptOne;

if (condition) //this is just something to call the function

{

scriptOne.scriptOneFuctionName(variable);

}

11.30.23 Invoke Repeating Function

Invoke Repeating is a built in function which calls another function to repeat:

InvokeRepeating();

You have three parameters to give it (time is always in seconds)

InvokeRepeating("functionName", waitTime, repeatTime);

waitTime 🡪 how long to wait before beginning repeat

repeatTime 🡪 how long before occurrences of repeating (ie how long between dealing damage, shooting bullets, changing color, etc)

10.12.23 Collision VS Trigger

## OnCollisionEnter

*Use this function when you want to detect a physical collision between two objects. If you have a situation where two objects need to physically interact, such as a ball hitting a wall, and you want to play a particle effect when they collide, then* ***OnCollisionEnter*** *is a suitable choice.*

## OnTriggerEnter

*Use this function when you want to detect when one collider enters the trigger zone of another without necessarily causing a physical collision. If you have a scenario where, for example, a player walks into a specific area and you want to trigger a particle effect without a physical collision occurring, OnTriggerEnter is more appropriate.*

12.04.23 Destroying a Game Object

## Syntax

Destroy(GameObjectName);

or

Destroy(GameObjectName, delayInSeconds);

## Scenario 1: destroying the gameObject the script is on

Destroy(this.gameObject);

## Scenario 2: destroying an object upon contact

### Trigger

private void OnTriggerEnter(Collider other)

{

if (other.gameObject == collidingGameObject)

{

Destroy(other.gameObject);

}

}

### Collision

private void OnCollisionEnter(Collision collision)

{

if (collision.gameObject == collidingGameObject)

{

Destroy(collision.gameObject);

}

}

## Scenario 3: destroying another object in the game

void DestroyObjectFromDistance(GameObject objectToDestroy)

{

Destroy(objectToDestroy);

}

// Somewhere in your code, when needed:

DestroyObjectFromDistance(otherGameObjectName);

12.04.23 Instantiating a Game Object or Prefab

To create an object in the game, use the Instantiate function

Instantiate(nameOfPrefabToSpawn, position, rotation);

## Example Code

public class SpawnManager : MonoBehaviour

{

public GameObject myPrefab; // Reference to your prefab assigned in the Inspector

private Vector3 spawnPosition = new Vector3(0, 0, 0); // Declaration of spawn position

private Quaternion spawnRotation = Quaternion.identity; // Declaration of spawn rotation

void SpawnPrefab()

{

// Instantiate the prefab at the specified position and rotation

GameObject spawnedObject = Instantiate(myPrefab, spawnPosition, spawnRotation);

}

// you must call SpawnPrefab somewhere else in the script for it to run

}

12.04.23 Setting the position of instantiated object

You can set the position on an instantiated object upon spawn in two ways:

## Spawn Position Object

Here you have an empty object with the position and rotation you want your prefab to have

### Example code

public GameObject myPrefab; // Reference to your prefab assigned in the Inspector

public Transform spawnPoint; // Reference to the empty GameObject in the scene

void SpawnPrefabAtEmptyObject()

{

Instantiate(myPrefab, spawnPoint.position, spawnPoint.rotation);

}

// you will need to call SpawnPrefabAtEmptyObject somewhere else for it to run

## Vector3 Hard Code

Here you set the position and rotation of the prefab by hand

public class SpawnManager : MonoBehaviour

{

public GameObject myPrefab; // Reference to your prefab assigned in the Inspector

private Vector3 spawnPosition = new Vector3(0, 0, 0); // object position

private Quaternion spawnRotation = Quaternion.Euler(xRotation, yRotation, zRotation); // object rotation

void SpawnPrefab()

{

// Instantiate the prefab at the specified position and rotation

GameObject spawnedObject = Instantiate(myPrefab, spawnPosition, spawnRotation);

}

}  
Quaternion Info

spawnRotation can be called by either:

private Quaternion spawnRotation = Quaternion.identity; //for no rotation

or

private Quaternion spawnRotation = Quaternion.Euler(xRotation, yRotation, zRotation); //for specific rotation

10.12.23 Particle system playing on Collision.

Particle System call script:

public class ppe: MonoBehaviour

{

public GameObject GoalParticles;

void OnCollisionEnter(Collision coll)

{

if (coll.collider.CompareTag("Ball"))

{

Play();

}

}

void Play()

{

GameObject dots = Instantiate(GoalParticles, transform.position, Quaternion.identity);

dots.GetComponent<ParticleSystem>().Play();

}

}

Where:

ppe 🡪 name of the script

GoalParticles 🡪 particle system

Ball 🡪 tag on the “ball”

Dots 🡪 script only object

ppe is attached to the GOAL object (object that the ball comes into contact with). The particle system is a child of the GOAL. Ball is tagged Ball. Don’t forget to drag the particle system into the GoalParticles slot in the script in the inspector.

11.29.23 Particle System Playing when called from other script

Other Script must call play();

Ie:

if (playParticles == true)

{

ppe.GetComponent<ppe>().Play();

}

The below script where the particle system is called.

ppe 🡪 name of the script

ParticleSystemName 🡪 particle system

Dots 🡪 script only object

public class ppe: MonoBehaviour

{

public GameObject GoalParticles;

void Play()

{

GameObject dots = Instantiate(GoalParticles, transform.position, Quaternion.identity);

dots.GetComponent<ParticleSystem>().Play();

}

}

12.04.23 Setting Particle System Positon

If you want to play a particle system in a specific location:

## Example 1 (best for enter/exit situations)

[SerializeField] private GameObject particles; //assign this in inspector

Private void start()

{

Particles.SetActive(false);

}

private void Update

//or replace with the fuction that calls the particle system

{

if(logic here)

{

particleSystemCall();

}

else

{

Particles.SetActive(false);  
}

}

Private void particleSystemCall()

{

Particles.SetActive(true);

Particles.transform.position = new Vector3 (x,y,0f);

}

## Example 2 (best for prefabs and set times)

public class ParticleController : MonoBehaviour

{

public GameObject particlePrefab;

public float playDuration = 2.0f;

private GameObject currentParticles;

// Method to start the particle system at a given position

public void StartParticleSystem(Vector3 position)

{

// Instantiate the prefab and set its position

currentParticles = Instantiate(particlePrefab, position, Quaternion.identity);

// Start the particle system

currentParticles.GetComponent<ParticleSystem>().Play();

// Start a coroutine to stop and destroy the particles after playDuration seconds

StartCoroutine(StopAndDestroyParticles());

}

// Coroutine to stop particle system after playDuration and destroy it

private IEnumerator StopAndDestroyParticles()

{

yield return new WaitForSeconds(playDuration);

if (currentParticles != null)

{

// Stop the particle system

currentParticles.GetComponent<ParticleSystem>().Stop();

// Destroy the instantiated particles after stopping

Destroy(currentParticles, 1.0f); // Optional delay before destroying

}

}

}

12.04.23 Coroutines: The basics

Coroutines in Unity are functions that can suspend their execution temporarily and then continue from where they left off. They're used to create more complex and responsive behaviors without blocking the main thread.

## How Coroutines Work:

* Suspendable Execution: Coroutines use the yield statement to pause execution at a specific point and then resume later.
* Non-Blocking: They allow you to perform time-based operations without freezing the entire game or script.
* Frame Independence: Coroutines can run across multiple frames, enabling time-delayed actions or continuous updates.

## Delayed Execution Example

IEnumerator DelayedAction(float delayTime)

{

yield return new WaitForSeconds(delayTime);

// Code executed after the delay

}

// Example usage

StartCoroutine(DelayedAction(2.0f)); // Starts a delayed action after 2 seconds

## Continuous Update Example

IEnumerator ContinuousUpdate()

{

while (true)

{

// Code executed every frame

yield return null; // Wait for the next frame

}

}

// Example usage

StartCoroutine(ContinuousUpdate()); // Starts continuous updating

## Animating Progress Example

IEnumerator AnimateProgress(float duration)

{

float elapsedTime = 0f;

while (elapsedTime < duration)

{

// Update progress over time

float progress = elapsedTime / duration;

// Apply progress to an object or variable

elapsedTime += Time.deltaTime;

yield return null; // Wait for the next frame

}

// Animation complete

}

// Example usage

StartCoroutine(AnimateProgress(3.0f)); // Animates progress over 3 seconds

## Multi-Step sequences Example

IEnumerator MultiStepSequence()

{

Debug.Log("Step 1");

yield return new WaitForSeconds(1.0f);

Debug.Log("Step 2");

yield return new WaitForSeconds(2.0f);

Debug.Log("Step 3");

// ...

}

// Example usage

StartCoroutine(MultiStepSequence()); // Executes a sequence of steps with delays

## Use Cases for Coroutines:

* Timed Operations: Delayed actions, timed effects, animations, etc.
* Continuous Updates: Smoothly updating values or behaviors over time.
* Multi-Step Processes: Sequences of actions with delays or waiting periods.
* Asynchronous Tasks: Loading assets, fetching data, or complex calculations without blocking the main thread.

## Syntax:

* IEnumerator: Indicates that the method is a coroutine and will yield control during execution.
* yield return: Pauses the coroutine and specifies what to yield control to. Examples include null (next frame), WaitForSeconds (time delay), or another coroutine.
* You start coroutines in Unity by using StartCoroutine(MyCoroutine()) or StartCoroutine("MyCoroutine") within a MonoBehaviour.

11.6.23 The Ground Is Lava: Environmental Damage

public float damageRate = 5f;

private void Start()

{

InvokeRepeating("OnTriggerEnter", 0f, damageRate);

//see InvokeRepeating info section

}

void OnTriggerEnter(Collider other) //see Trigger vs Collider section

{

if (other.gameObject.tag == "Player")

{

//this line says that if there a collision, get the component from the PlayerStats script called takedamage and send over the float 0.5f

other.gameObject.GetComponent<PlayerStats>().TakeDamage(0.5f);

}

}

11.30.23 health bar

## Method 1: Canvas Image

Unity Hierarchy Set-up:

1. Add Canvas Image object as follows:

* Player > Main Camera > Canvas > Image

1. Name the image object something like: PHealthBar (because you may have enemies that have health bar images as well and it makes it easy to tell them apart)
2. A screenshot of a computer

   Description automatically generatedCreate 2 Scripts: PlayerStats and PlayerHealthBar *(see scripts below)*
3. Add PlayerStats to the player object (ie. XR Origin (XR Rig))
4. Add PlayerHealthBar to the Canvas Image object (ie. PHealthBar)
5. Drag the player object into the player slot in the Image object’s inspector

### PlayerStats

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class PlayerStats : MonoBehaviour

{

public float PlayerHealth = 20f;

public Text HUD;

//HUD is your in game text canvas object to debug while in an APK see 10.31.23

private void Start()

{

HUD.text = PlayerHealth.ToString();

}

void Update()

{

if (PlayerHealth <= 0)

{

Died();

}

HUD.text = PlayerHealth.ToString();

}

public void TakeDamage(float damage)

{

PlayerHealth -= damage;

GameObject CanvasChild = transform.GetChild(0).gameObject;

GameObject HealthBarChild = CanvasChild.transform.GetChild(0).gameObject;=

HealthBarChild.GetComponent<PlayerHealthbar>().ChangeHealth(PlayerHealth);

}

private void Died()

{

//something goes here

HUD.text = "You Died. Try again";

}

}

### PlayerHealthbar

using System.Collections;

using System.Collections.Generic;

using UnityEngine.UI;

using UnityEngine;

public class PlayerHealthbar : MonoBehaviour

{

private float MaxHealthBarLength = 1.0f;

private float MaxPlayerHealth;

private float CurrentPlayerHealth;

private float HealthPrecentage;

private float CurrentHealthBarLength;

public GameObject Player;

private RectTransform rt;

// Start is called before the first frame update

void Start()

{

rt = GetComponent<RectTransform>();

CurrentPlayerHealth = Player.GetComponent<PlayerStats>().PlayerHealth;

GetComponent<Image>().color = Color.blue;

}

public void ChangeHealth(float PlayerHealth)

{

CurrentPlayerHealth = PlayerHealth;

HealthPrecentage = CurrentPlayerHealth / MaxPlayerHealth;

CurrentHealthBarLength = MaxHealthBarLength \* HealthPrecentage;

rt.sizeDelta = new Vector2(CurrentHealthBarLength, rt.sizeDelta.y);

ImageHealthColor();

}

private void ImageHealthColor()

{

if (CurrentPlayerHealth > 14)

{

GetComponent<Image>().color = Color.blue;

}

else if (CurrentPlayerHealth > 7)

{

GetComponent<Image>().color = Color.magenta;

}

else

{

GetComponent<Image>().color = Color.black;

}

}

}

## Method 2: Slider

1. Create Slider UI object in the Player Canvas (or enemy)
2. A screenshot of a computer

   Description automatically generatedName the slider HealthBar
3. Under the HealthBar parent object make sure the handle direction is left to right
4. Under Background, delete the sourceImage and change the color to red (or what ever you want the damage color to be)
5. Under Fill Area > Fill, again delete the source Image and change the color to green (or what ever you want the health color to be)
6. Under Handle Slide Area delete the Handle object
7. Create the PlayerHealth Script (see below)
8. Attach the script to the player

## PlayerHealth

using UnityEngine.UI;

public class PlayerHealth : MonoBehaviour

{

public float CurrentHealth;

public float maxHealth = 10;

[SerializeField] private Slider healthSlider;

public Text HUD;

// Start is called before the first frame update

void Start()

{

CurrentHealth = maxHealth;

healthSlider.minValue = 0;

healthSlider.maxValue = maxHealth;

healthSlider.value = CurrentHealth;

HUD.text = "Health set";

}

private void Update()

{

if(CurrentHealth < + 0)

{

Died();

}

HUD.text = CurrentHealth.ToString();

}

public void TakeDamage (float damage)

{

CurrentHealth -= damage;

healthSlider.value = CurrentHealth;

}

private void Died()

{

//something goes here

HUD.text = "You Died. Try again";

}

}

## Dealing Damage

Add the following to what ever scripts you are wanting to deal damage. Make sure you change the names here to what ever you used in your player scripts.  
 void Start()

{

InvokeRepeating("DealDamage", 1.0f, 1.0f);

//see 11.30.23: Invoke Repeating (only required if the DealDamage method is not called on an interaction: ontrigger, on collision, if else, switch state, etc)

}

public void DealDamage ()

{

gameObject.GetComponent<PlayerHealth>().TakeDamage(5f);

//see 11.29.23: Calling a method from another script

}

## Healing

Add this method to the playerHealth or playerHealthBar script. Make sure to call this method somewhere in the script.  
 public void Heal (float health)

{

CurrentHealth += health;

healthSlider.value = CurrentHealth;

}

12.03.23 Previous Object Interaction Tracking (not letting a trigger work twice in a row)

## Name of Example Script

public class TriggerBox : MonoBehaviour

{

public int count = 0;

public int totalCountNeeded = 6;

private bool lastTriggerWasA = false; // Flag to track the last trigger box

// This method is called when the object enters a trigger box collider

private void OnTriggerEnter(Collider other)

{

// Check if the object entered the trigger box

if (other.CompareTag("ObjectTag")) // Replace "ObjectTag" with the tag of your object

{

// Check if the object entered the opposite trigger box from the last one

if ((lastTriggerWasA && gameObject.CompareTag("TriggerBoxB")) ||

(!lastTriggerWasA && gameObject.CompareTag("TriggerBoxA")))

{

count++; // Increase the count if conditions are met

lastTriggerWasA = !lastTriggerWasA; // Update the flag for the last trigger box

}

// Check if the count reaches the desired number

if (count >= totalCountNeeded)

{

// Call your method to do something else

DoSomethingElse();

}

}

}

// Method to perform something else when the count reaches the desired number

private void DoSomethingElse()

{

// Your logic here

}

}

12.10.23 Switch States

using UnityEngine;

using System.Collections;

public class ConversationScript : MonoBehaviour

{

public int intelligence = 5;

void Greet()

{

switch (intelligence)

{

case 5:

print ("Why hello there good sir! Let me teach you about Trigonometry!");

break;

case 4:

print ("Hello and good day!");

break;

case 3:

print ("Whadya want?");

break;

case 2:

print ("Grog SMASH!");

break;

case 1:

print ("Ulg, glib, Pblblblblb");

break;

default:

print ("Incorrect intelligence level.");

break;

}

}

}

Universal Render Pipeline

The easiest way to use the URP is to set up your project using the template. (this can be downloaded from unity hub).

However, if you did not set up your project with it, you can add it post.

1. Back-Up project. This makes irreversible changes and is nearly impossible to reverse. Back up here.
2. Go to -> Package manager
3. Packages: Unity registry
4. Search :Universal Render Pipeline
5. Download into your project
6. In the project window:
   1. Right click
   2. Create
   3. Rendering
   4. URP Asset (with universal renderer)
7. Save project (again)
8. Go to Window -> Rendering -> Render pipeline converter
9. Check all of the boxes (unless you know u don’t want one)
10. Initialize Converters
11. Convert Assets
12. Pray that your file doesn’t break

Using Starter Assets 3rd Person Controller

1. Delete Main Camera (if you do not, this will not work!)
2. Navigate: StarterAssets -> ThirdPersonController -> Prefabs
3. Drag & Drop into hierarchy “NestedParentArmature\_Unpack”
4. Right click on “NestedParentArmature\_Unpack” and select *Unpack* (not Unpack completely)
5. Shift Select all assets inside “NestedParentArmature\_Unpack” and drag into hierarchy to unnest
6. Delete joystick assets
7. TEST PROJECT TO MAKESURE THE CAMERA FOLLOWS!!!!

Mixamo Addins

## Character/Skin

1. Find skin on Mixamo.com
2. Change to unity FBX
3. Download without any animations
4. Drag into unity file
5. Select Asset -> Materials -> Extract Textures
6. Make sure to save in TEXTURES folder (keeps project clean)
7. Fix Now and Save anything it asks to
8. Drag and Drop characters into scene

## Animations/Movements

1. Find animation on Mixamo.com
2. Change to unity FBX
3. Change to download without skin
4. Download animation
5. Drag into unity file
6. Expand Animation but select the parent object (main square)

Parent object -> Animation

1. Check Loop Time & Loop Pose
2. Select apply
3. Drag Triangle Icon onto the character you want to animate
4. Add box collider to character

Hierarchy -> character asset -> animation controller

1. check apply root motion
2. select apply

Project -> character asset -> rig -> animation type

1. change from generic to humanoid & select apply

Project -> Parent Animation square -> rig -> animation type

1. change from generic to humanoid

Project -> Parent Animation square -> rig -> avatar destination

1. change to copy from other avatar
2. click the target and then the character model from the drop down
3. select apply

Hierarchy -> character asset -> animation controller -> avatar

1. click the target and then the character model from the drop down

Still to Add

* Open XR button functions 🡪 what they are, resources, how to get the button presses script example
* Notes on new meta VR package 🡪 set up steps and notes
* Text Mesh pro functionality
* App.quit method
* Movement script & interaction with WSAD w/ left mouse button 2d & 3d w/ optional jump function
* State machine
* OnTriggerStay
* onAwake?
* Accessing parent & child objects (including grand parent and grand children)
* OnTriggerExit
* OnCollisionDetection (referencing a different script)
* How to change object color/texture through just code
* Class set-up & inheritance syntax
* Setactive(bool)
* Setting up pause pop-up menu steps (mainly in unity inspector)
* Switch states & syntax
* Arrays & syntax
* Random functions (random.range, etc) w/ inclusion and exclusion principles
* audio functionalities (snap shots, zones, audio clips)
* scene manager/management
* checking tags & use cases
* checking names & use cases
* platform movement script & set up