Design Prototyping Fundamentals

Exercise – Basic script usage in Unity

All but the smallest of projects tend to have lots of scripts that perform different actions to drive gameplay and supporting systems. This exercise hands you four basic scripts in a project **BasicScriptApplication**, and lets you apply them to a pre-created scene, so you can generate some very simple interactivity and gameplay.

A project with the four scripts has been set up for you, they reside in the Scripts folder. There are also a series of coloured materials you can drag and apply to objects, and they reside in the Materials folder.

Note that we are going to make an extremely basic third-person control system in this game. It is basic to the point you will unlikely want to use it in other projects - We will create a much more robust FPS character controller for that later.

Step 1 – Understand your level

The scene included is very simple, but it is good to understand what has been provided for you to work with, as some of it is set up to be especially to work with the scripts.

- **PlatformX** – Platforms are the basis of your gameplay area. The player can walk on these.
- **PadX** – Just a small 'plate' in the middle of the ground of each platform. Each one is set up as a trigger, meaning we can detect when another object collides with them - These would make a great teleporter entrance…
- **ColourPoleX** – Is a solid pole, coloured white by default. You could add a colour to it from the Materials provided…
- **ZoneX** – These are transparent trigger cubes around each of the ColourPole objects.
- **BottomOfWorld** – This is just a sort of 'floor' to the level. It is just a solid platform, but you could adjust it to be more useful in your game…
- **ExitX** – Empty game objects, they only have a location, making them mostly useful as a destination for say, teleporting…
- **PlayerCube** – A cube, coloured black, meant as a placeholder player character. It has a Rigidbody component attached to it, so it won't move through solid objects and will fall from gravity, etc.
- **Main Camera** – In this project, the main camera has been made a child of PlayerCube. This means it is attached to it, and will move with it.

Note – All 'X' numbers are the same for each platform / area. Pad2 is on Platform2, as is Zone2, ColourPole2 and Exit 2 as well, etc.

Next, on to the scripts…

Step 2 – Check your scripts

The project has a Scripts folder under Assets, and this contains four scripts:

- **PlayerController.cs**
- **ChangeMaterialColour.cs**
- **MouseAirCamera.cs**
- **Teleporters.cs**

Here is how they work, and how you should apply them to your scene. It is highly advised you read this section without doing anything in Unity first.

PlayerController.cs
Add this script as a component to a GameObject and that will make the object controllable.

- An object called 'PlayerCube' was created for this purpose. It has already been tagged 'Player' in the inspector too.

ChangeMaterialColour.cs
Add this script to an object that has its collider set as a trigger (make sure Is Trigger is checked)

- Any object that enters that trigger will have its material colour changed to the RBG variables defined by the public variables, Red, Blue, Green.
- The variables can be changed in the Inspector in Unity.
- The object that will change colour must have a material on it other than a default material.

MouseAirCamera.cs
Apply this script to your main camera, which needs to stay as a child to **PlayerCube**.
Tips:

- In the Inspector window you can set the camera transforms X, Y ,Z to 0, 0, 0 to center the camera to the GameObject. Then adjust its position as desired.
- You need to set the public variable "Target" in this script as your player object, your player will rotate according to your horizontal mouse movement.

Teleporter.cs
When this script is applied to a trigger object the player can move into, it will then teleport the player to a location of another object in the scene. script that will move a player who moves into a trigger with If you, your player will teleport to that location after entering a trigger with the teleporter.cs script on it.
Tip:

- The only objects in the scene really suited to apply this to are the **PadX** ones.
- Note, this only works on GameObjects tagged as "Player", like the **PlayerCube** is.
- You will need to set a target destination for the teleport by dragging an object into the 'Teleporter Out Location' field in the Inspector, or it won't take you anywhere.

Step 3 – Basic functionality testing

Very simply, having looked at the scripts, actually apply them to the game objects, hook things up, and make sure things work as expected.

We want to ensure things work before we bother trying to do special things with them. So ensure you can do the following before you do anything else:

- The player cube can move, and you have horizontal mouse look working.
- There is at least one functioning teleporter
- There is at least one functioning colour changing trigger

Step 4 – Make something more interesting – A puzzle challenge

It should be fairly clear what mechanics you have to play with to make a game out of, but let's clarify to be sure:

- 2-axis movement
- The ability to rotate the camera around the player
- The ability to teleport from a trigger object to a target object
- The ability to make the player change colour if they enter a trigger object

So we can move and look about, enter teleporters, and have our player's colour changed.

Tips on making things interesting:

- The most obvious sort of game we can make here is a puzzle, where the player needs to reach a target destination, and they have to do things with teleporters and colour states for that to be possible.
- Crawl before you walk, walk before you run – Get a basic puzzle working first, then make that more complex.
- You are absolutely allowed to make more objects in your scene! Reasons you might want to do that:
    - Maybe you want multiple teleporter entrances on a single platform, or
    - Maybe you need to make objects to help distinguish the different platforms, or
    - Maybe you just need more things on a platform for a puzzle concept of yours to work, or
    - Maybe you need to make some sort of 'final destination' platform.
    - Maybe you have some entirely different way to make this interesting – Make platforms, walls, whatever. Just keep it manageable in scope.
- You are allowed to make more scripts! Or, you can edit the existing scripts. Ultimately, your scope to change the game without doing one of these two things is quite limited.
- The BottomOfWorld platform would really make an ideal 'reset' trigger.
    - Note – Teleporting to the start alone will NOT reset other things in your level that may have changed from the original state!
- It was intentional that all 4 platforms were identical in physical form.
    - If you want to make direction even more unclear, try adjusting the rotation of the Directional Light in this scene.
    - If you want to make direction easier, just make more visual changes and landmarks.
- Using Booleans to set and check what state things are in could allow you to do many things with teleporter requirements.