Lesson 6

- Dont forget to download the project from GitHub if you happen to miss a lesson or need a reference!

A Quick Review of the Unity UI Hierarchy

- All Unity UI elements require a canvas first
  - The canvas is the largest UI element and is required as a parent to all other UI elements
- Canvases hold UI elements
  - Panels: Like a smaller canvas usually hold other elements that you want to be able to hide
  - Buttons: Special elements that run off the Event System, best for menus
  - Images, Text, etc… All the other basic UI elements

Upgrading the UI Controller Script for a Pause Menu

Scripting the Pause Menu (UIController)

- Create a reference for our panel as type GameObject.
- Create a variable to hold the pause state of the game.

```
public class UIController : MonoBehaviour
{
    public TextMeshProUGUI coinsText;

    // A variable to keep track if the game is paused
    bool isPaused = false;
    // A reference to the Panel that contains our pause UI
    public GameObject pauseUI;

    void Update()
    {
        coinsText.text = "Coins: " + GameController.GetCoins();
    }
```

Scripting the Pause Menu...

- Create a method to pause or unpause the game based on the variable we made. Make sure this method is public .

```
// Pause or Unpause the game
public void Pause()
{
    // If the game is currently paused then unpause it
    if (isPaused)
    {
        isPaused = false;

        Cursor.visible = false;
        Cursor.lockState = CursorLockMode.Locked;

        pauseUI.SetActive(false);
        Time.timeScale = 1f;
    }
    // If the game is currently unpaused then pause it
    else
    {
        isPaused = true;

        Cursor.visible = true;
        Cursor.lockState = CursorLockMode.None;

        pauseUI.SetActive(true);
        Time.timeScale = 0f;
    }
}
```

Scripting the Pause Menu...

- Time.timeScale changes the actual speed of the game, meaning anything that relies on time is changed.
  - Importantly, setting the time scale to 0 stops physics.

- This is why UI Buttons are useful because they are controlled by the Event System which is not affected by the speed of the game.

Pause Menu in the Scene

- Create a Panel under the Canvas.

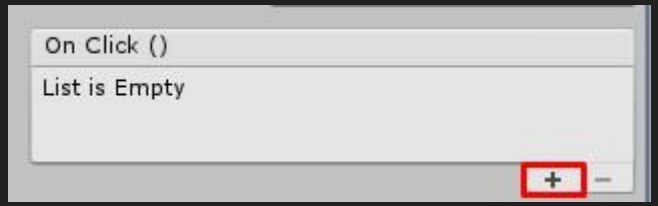
- Drag this Panel into the GameObject reference in the UIController script.

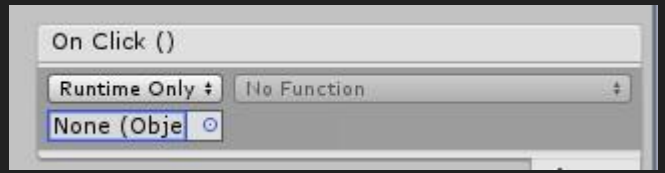
Pause Menu in the Scene: Buttons

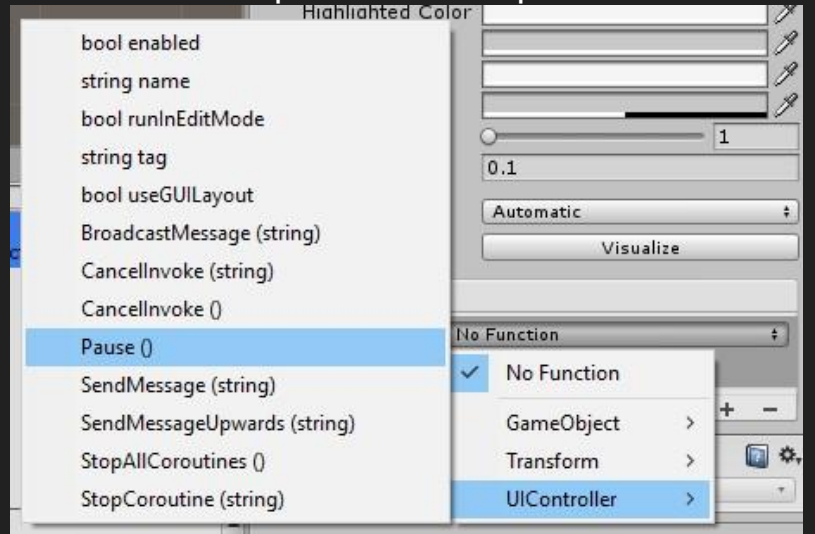
- Creating a Resume Game Button
  - Create a Button under the Panel
  - You can drag the button around on screen to a location that looks good
  - Don’t forget to change the Text element that is a child of the Button!

Pause Menu in the Scene: Buttons...

- Calling our Script with the Button
  - Click on the Button object.
  - Look for the Button (Script) Component in the inspector & find the area labeled OnClick().
  - Press the plus, then drag the GameObject that has the UIController script attached to it into the labeled area. In our case it is the UIMaster object.
  - Click on the button labeled “No Function” and find the UIController Script from the dropdown menu, then find and press the option labeled Pause().







Pause Menu in the Scene: Buttons...

- This Button will now call the Pause function whenever you click it!
- The problem now is that once it disappears we can’t get it to show back up.
  - We could fix this by creating another button that appears when the game is not paused and have it also call the Pause function.
- What we will do is call the function when the user presses the escape key in the Update function of UIController.

Pause Menu.

- Now that we can pause and unpause the game with Escape and unpause with the button we want the PauseUI to start off as being hidden.
- Add some functionality in Start() of the UIController script.

```
void Start()
{
    pauseUI.SetActive(false);
}
```

Some Problems...

- As of now pausing the game doesn’t stop the character from rotating or the platform from moving back and forth.
- This is because these objects use the Update function and do not take time into account when they move.
- Fix situations like this by using the Time.deltaTime component of Time.
- Taking the change in time into account when calculating movement will effectively stop time for that object.

Fixing the Problems...

- In PlayerController:

```
// Update is called once per frame
void Update()
{
    yaw += Input.GetAxis("Mouse X") * 150 * Time.deltaTime;
    transform.eulerAngles = new Vector3(0, yaw, 0);

    Vector3 velocity = new Vector3(0,0,0);
    velocity.x = Input.GetAxis("Horizontal") * speed * Time.deltaTime;
    velocity.z = Input.GetAxis("Vertical") * speed * Time.deltaTime;
```

- We actually used Time.deltaTime before, which is why the player can only turn and not move.

Fixing the Problems...

- In MovingPlatform:

```
// Update is called once per frame
void Update()
{
    if (Vector3.Distance(transform.position, waypoints[index].transform.position) < .1)
    {
        index = index + 1;
        if (index > waypoints.Length - 1)
        {
            System.Array.Reverse(waypoints);
            index = 1;
        }
    }
    else
        transform.position = Vector3.MoveTowards(transform.position, waypoints[index].transform.position, 2 * Time.deltaTime);
}
```

Fixed the Problems!

- Now our game should come to a full stop once we pause the game!

Scene Management

Review on Scene Management in Unity

- Each separate scene has a unique number that identifies that scene.
- We can use this number to load and reload scenes however we like.
  - You can also use the scene's name.
- You can check the scenes in the game by navigating to: File→ Build Settings
  - The scenes in build show the scenes which will be included in the final game, you can add scenes by pressing the Add Open Scenes button.
  - The number to the right of the scenes in the Scenes In Build list is the unique number that identifies each scene, called its index.

Restarting a Level

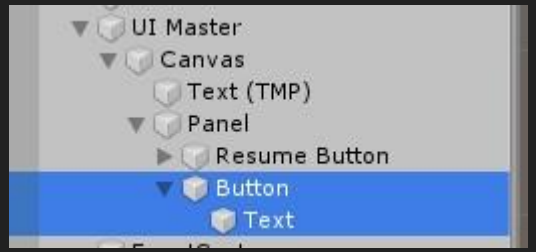
- Let’s create another button for the game scene, a Restart button using the Scene indexes.
- Start with the function. We’ll put this in GameManager since that seems a better place for this function than UIController.
  - We need to include the SceneManagement package in the GameManager.
  - Make sure it is not a static function.

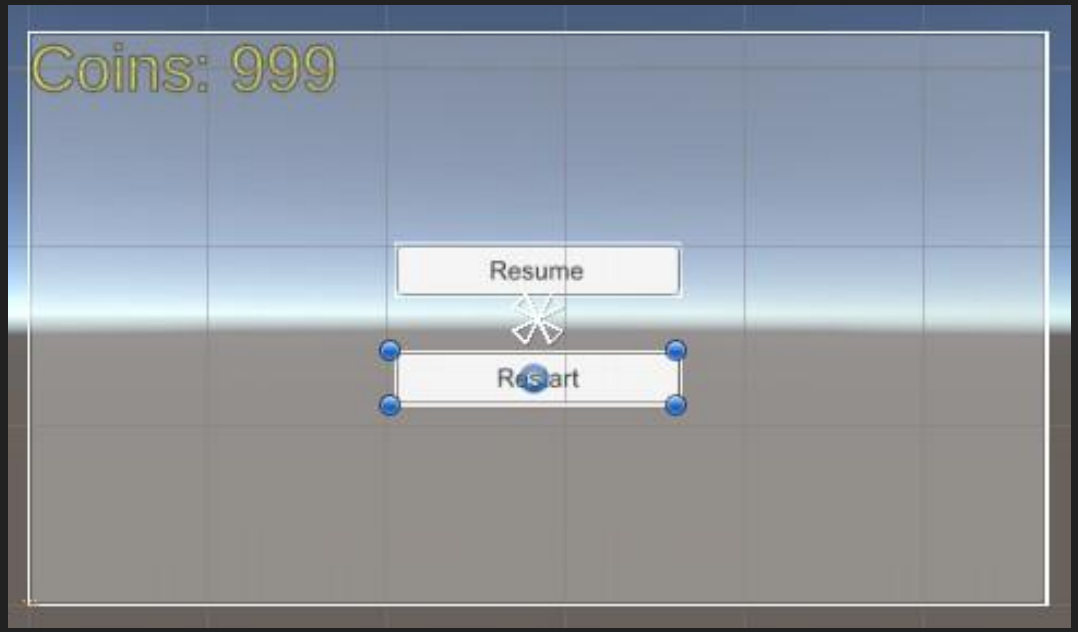
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class GameController : MonoBehaviour
{
```

```
    // Restart the level using the SceneManager
    public void RestartLevel()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
    }
}
```

Buttons in Scene

- Do what we did before and create another button under the panel and change the text component in its child to say “Restart”.





Buttons in Scene

- Go to the Button and find the Button (Script) component. Under the OnClick() hit the plus and drag the GameManager object into the box.
- Click on “No Function” and find the GameController Script from the dropdown menu, then find and press the RestartLevel() option.