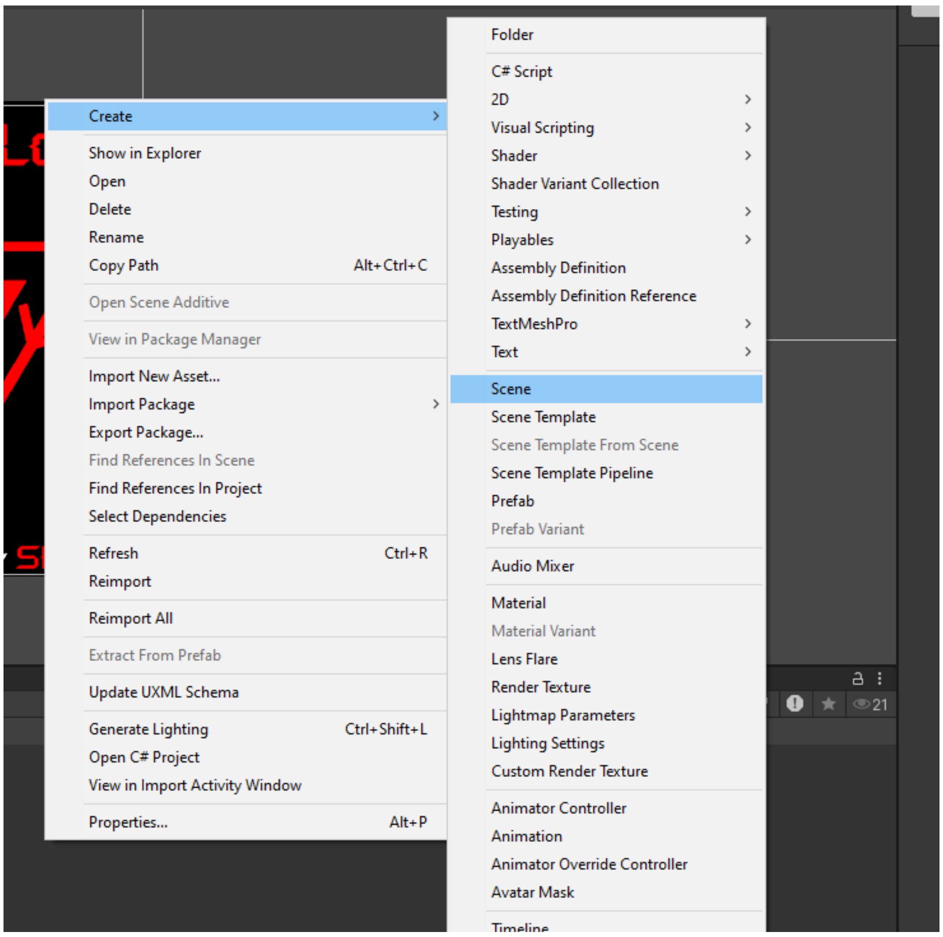
Unity Tutorial 4: Adding Win and Lose conditions to your game

Most video games have a win and lose condition. Well some games don't really have an ending, especially simulator games where you just simply play the game, earn in game currency to buy better items, and get rich. Win and Lose conditions are one of the most important parts of game design as it gives the player a reason to challenge themselves knowing they may lose or gain something by winning.

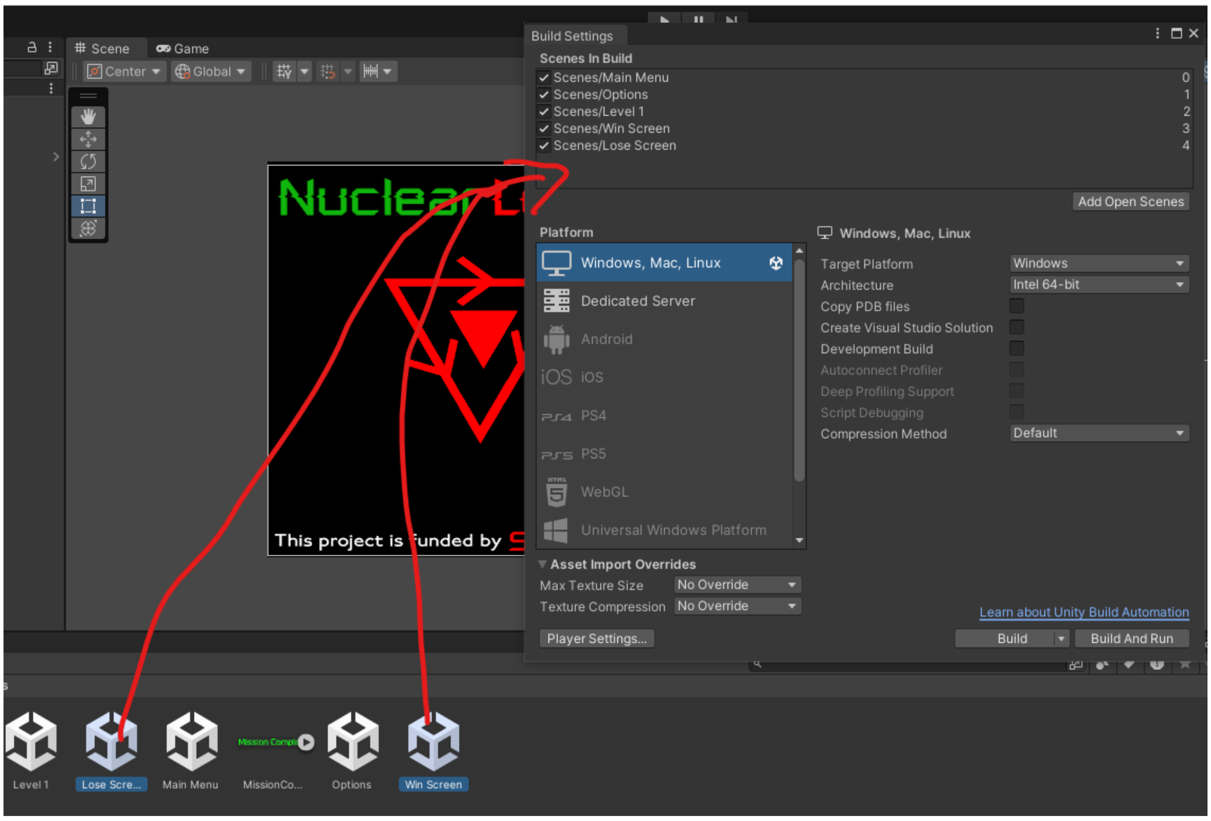
Let's get onto the tutorial shall we?

1. Creating the Win and Lose screens when the player wins/loses in the game

The first step is to add 2 scenes and name it "Win Screen" and "Lose Screen." These will be the screens you will direct the players to when they win or lose the game.



Now that you have created your two scenes remember to add it to your build settings!



Build Settings

Scenes In Build

✓ Scenes/Main Menu 0
✓ Scenes/Options 1
✓ Scenes/Level 1 2
✓ Scenes/Win Screen 3
✓ Scenes/Lose Screen 4

Add Open Scenes

Platform: Windows, Mac, Linux

Target Platform	Windows
Architecture	Intel 64-bit
Copy PDB files	
Create Visual Studio Solution	
Development Build	
Autoconnect Profiler	
Deep Profiling Support	
Script Debugging	
Compression Method	Default

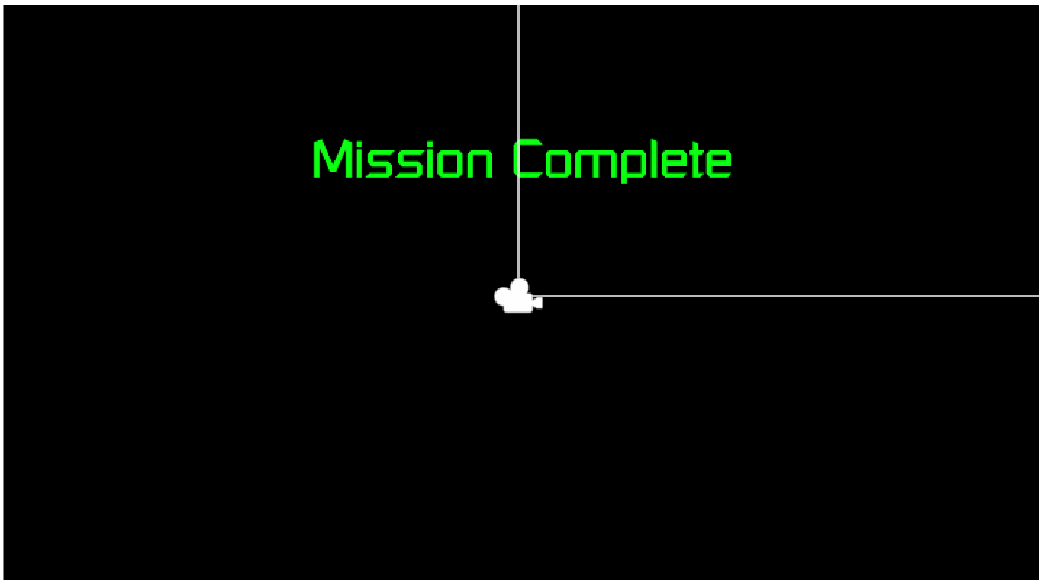
Asset Import Overrides

Max Texture Size	No Override
Texture Compression	No Override

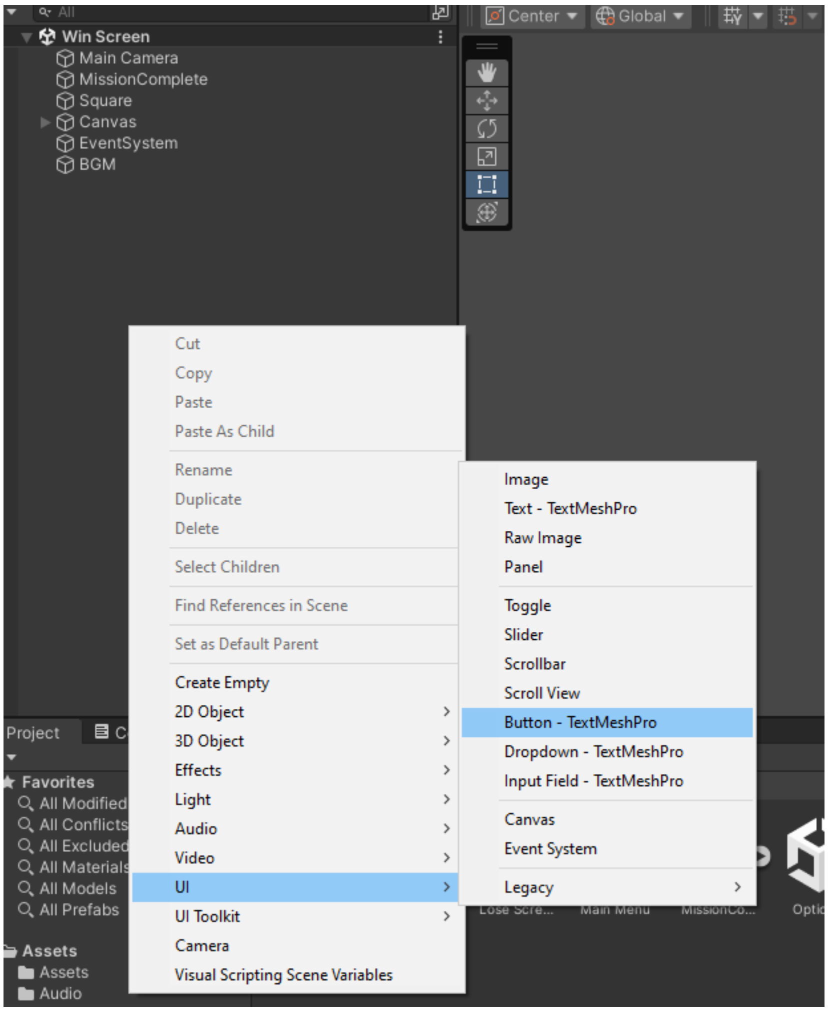
Now that you have added your scenes to your build settings you can now redirect your players there!

2. Adding Button functionality!

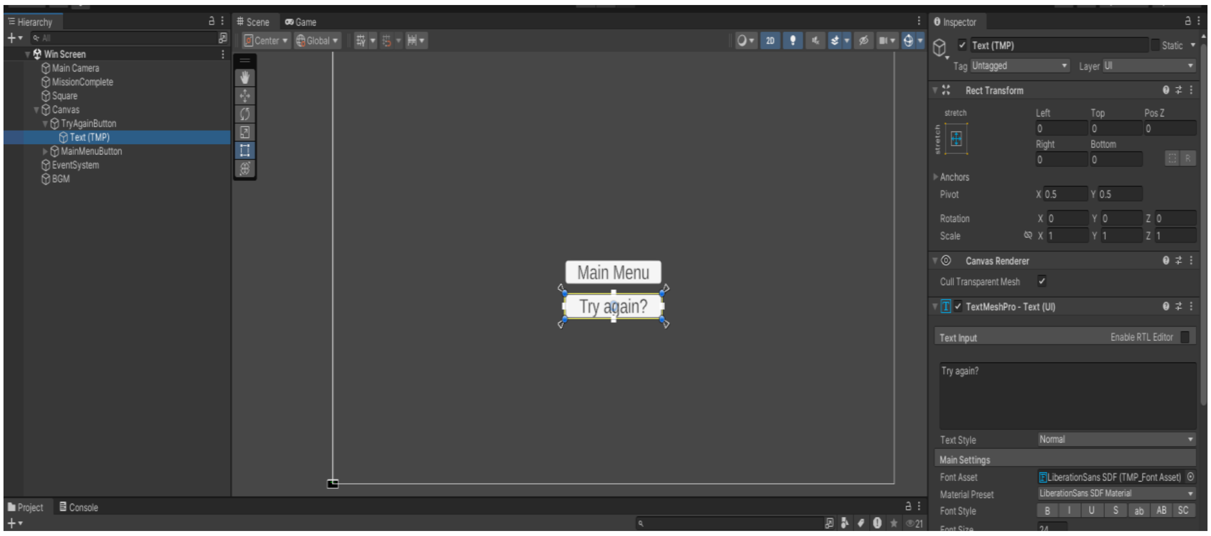
Now open one of your scenes and remember to add simple text! Just so the player knows whether they have won or died in the game.



Now that you finished adding text to notify the players what happened, you can now work on the buttons. Simply right click on the hierarchy and select UI - > Button- TextMeshPro and it will automatically create the canvas along with the Button as part of the UI.

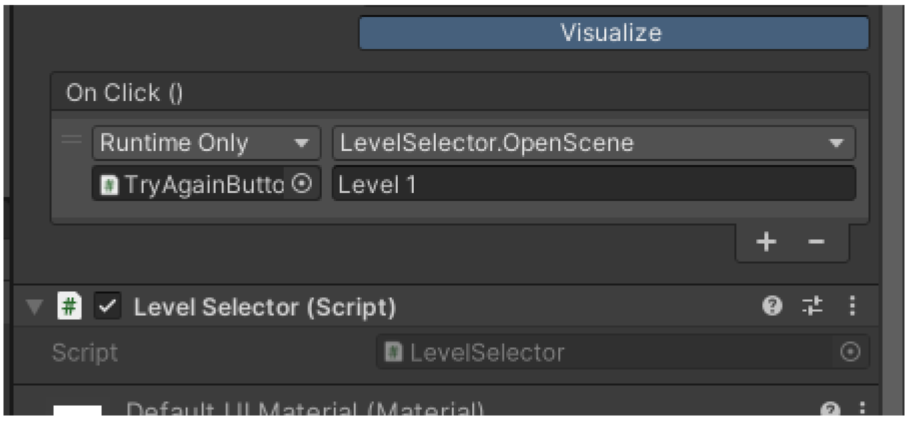


Similar to the main menu tutorial you label these buttons.

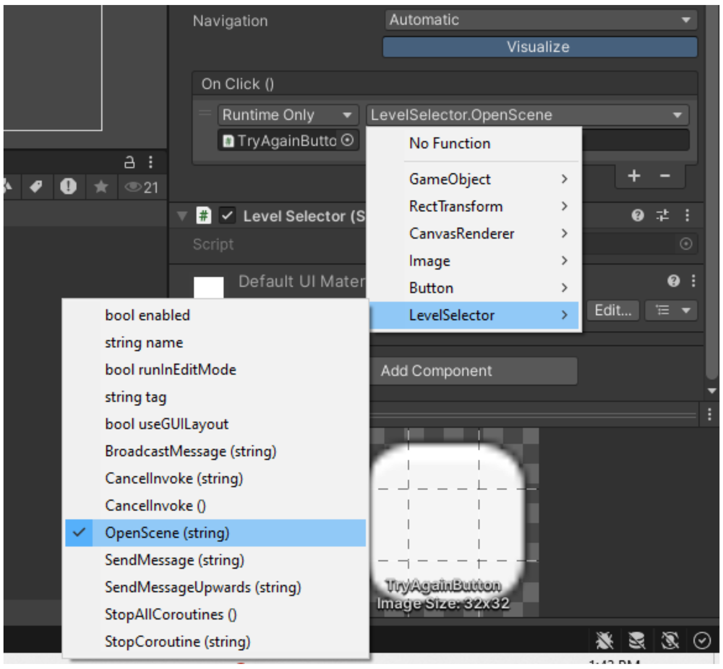


Text (TMP)
Tag Untagged Layer UI
Rect Transform
Left 0 Top 0 Pos Z 0
Right 0 Bottom 0
Pivot X 0.5 Y 0.5
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Canvas Renderer
Cull Transparent Mesh ✓
TextMeshPro - Text (UI)
Text Input
Try again?
Text Style Normal
Main Settings
Font Asset LiberationSans SDF (TMP_Font Asset)
Material Preset LiberationSans SDF Material

Now that you have added the buttons let's add the button functionality.



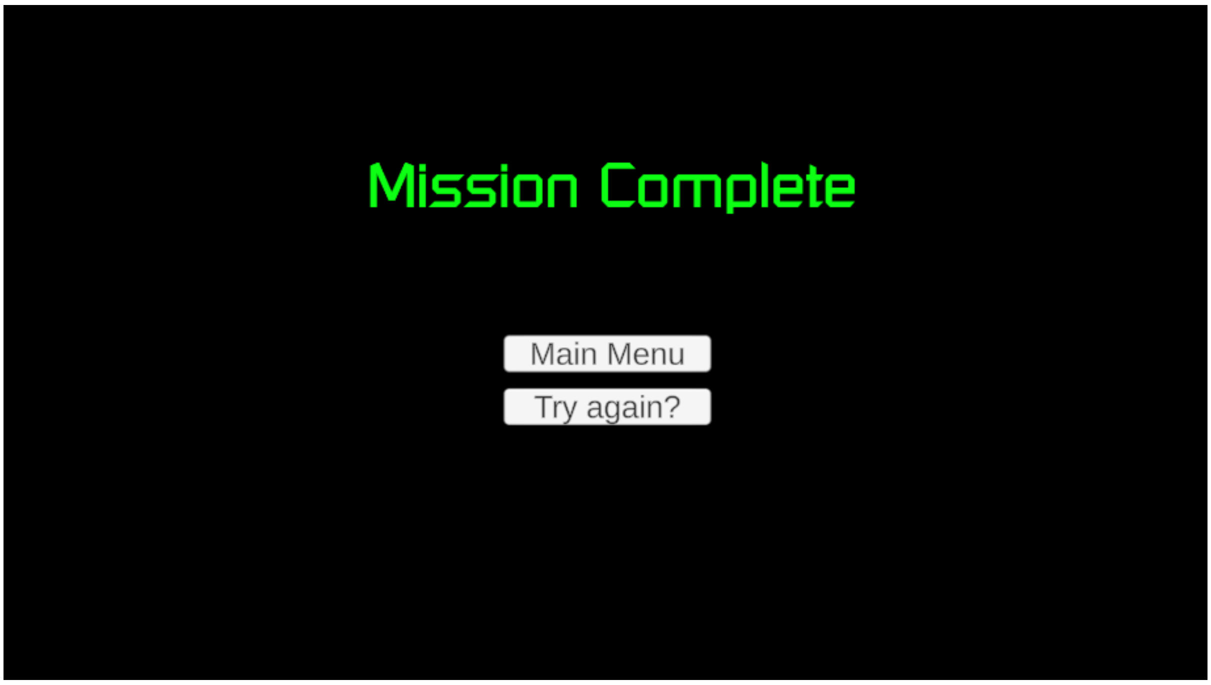
Drag your Level selector script which you made from the main menu tutorial into the inspector for the button (**NOT the text)**.



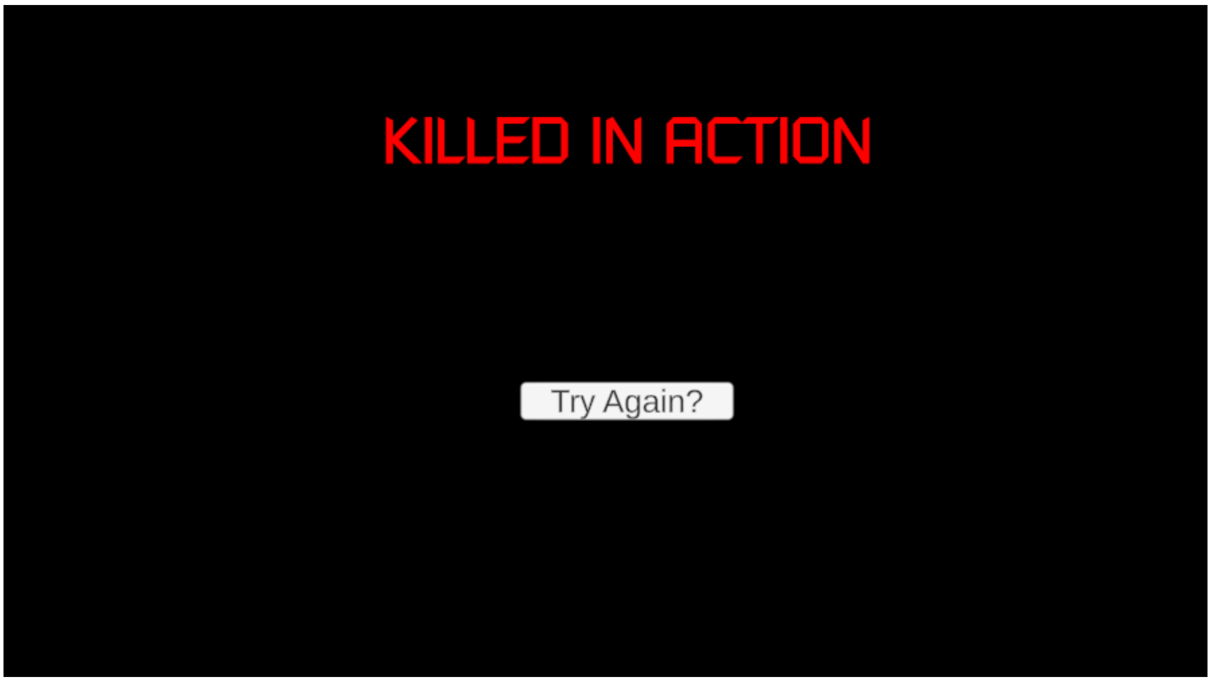
Click on the drop down menu next to the Runtime Only dropdown and select Level selector -> OpenScene (string) and type in the name of the scene you want to redirect your button

Now that you have added functionality to your button repeat the same thing for both the win and lose screens.

In the end you should end up with at least these screens



Example of a win screen

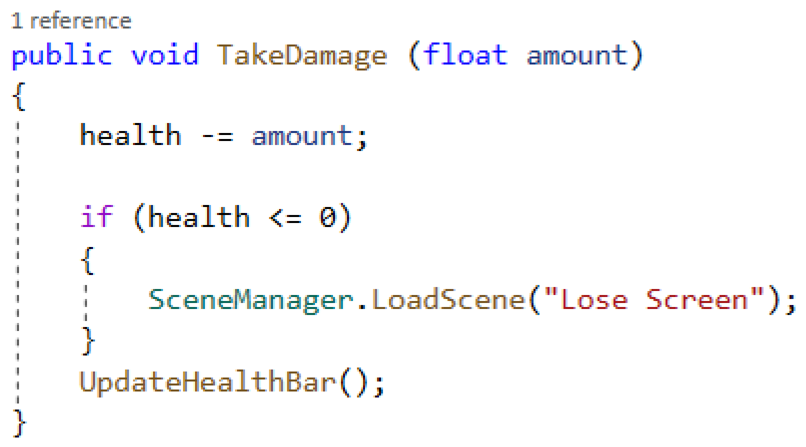


Example of a lose screen from my prototype

Now that you have the win/lose screens let's set up our win/lose condition!

3. Establishing a win/lose condition

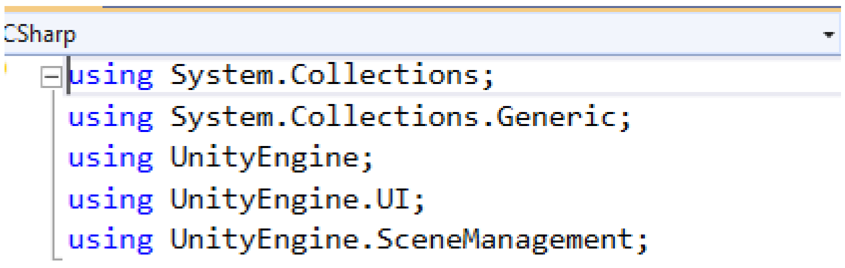
From the previous tutorial we recently implemented a player health system. Now let's go and change the conditions for the health so that the player gets redirected to the lose screen the moment their health hits 0 or below!



```
1 reference
public void TakeDamage (float amount)
{
    health -= amount;

    if (health <= 0)
    {
        SceneManager.LoadScene("Lose Screen");
    }
    UpdateHealthBar();
}
```

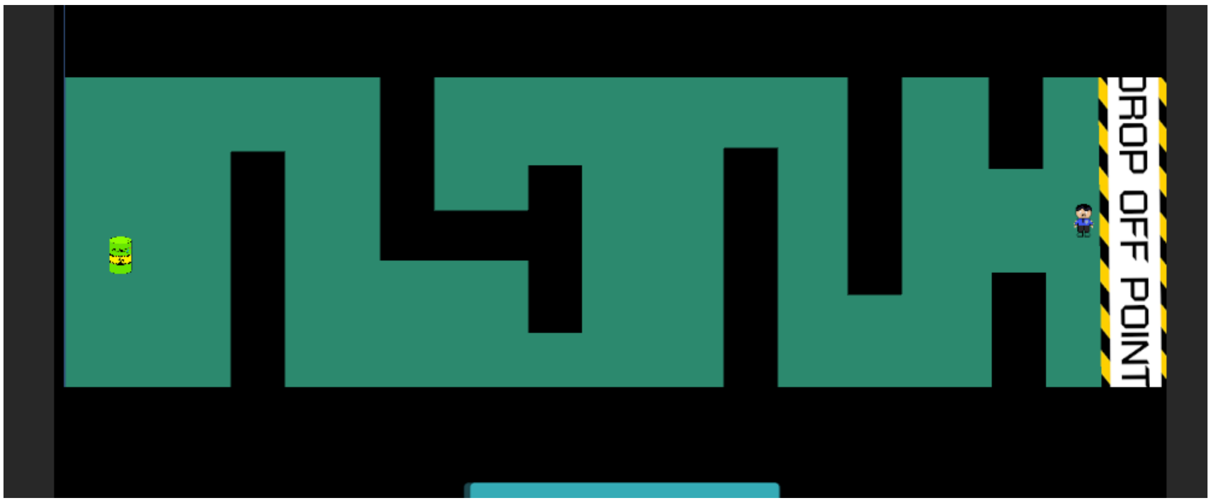
Add a code that redirects the player to the lose screen the moment they lose all their health. Don't forget to add "SceneManagement" to the namespace otherwise your code will NOT work!!!



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

Now that you have a lose condition feel free to add any extra conditions if you feel like adding multiple ways to die in a game!

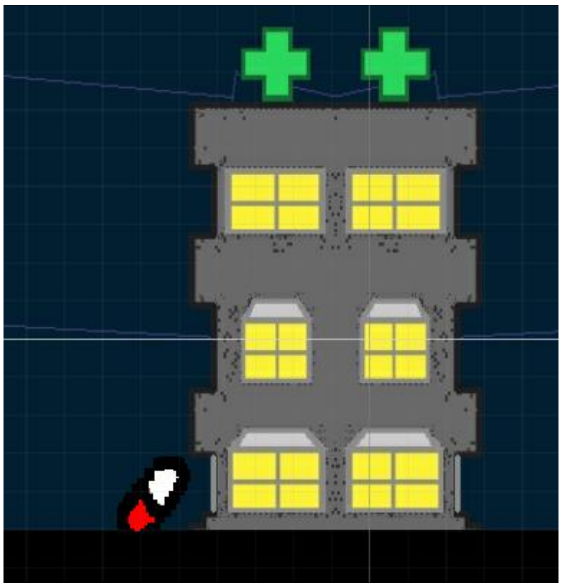
I personally added walls that kill off any players that go out of bounds.



Most games when there is a win condition they tend to make the players reach a finish line in a level or complete objectives while under attack by enemies.

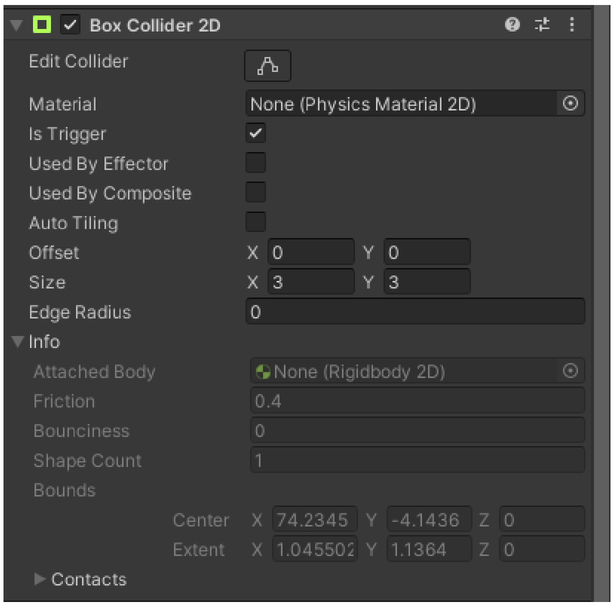
Feel free to add your own win condition but for this tutorial I will design a simple win condition where the player must reach a finish line.

First things first, you must have a finish line object where the players must go to in order to win.



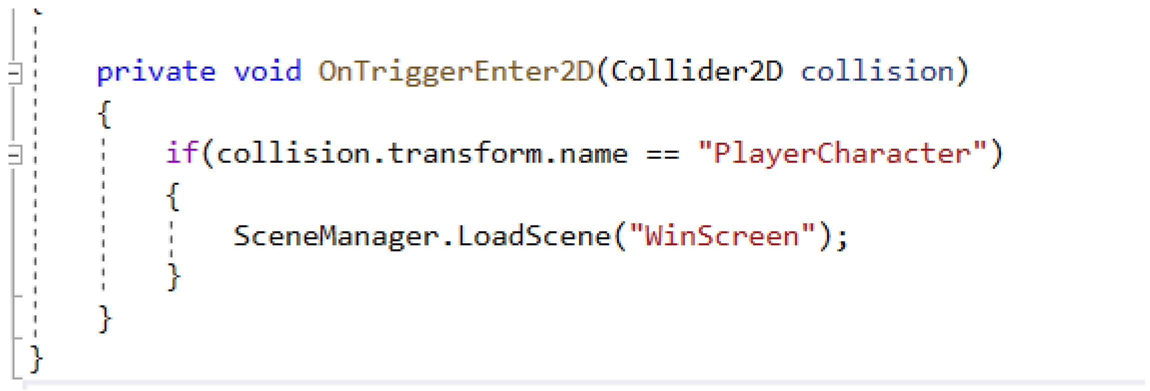
I will be using a screenshot from one of my old games which I developed. I used a pill as the "finish line."

Add a box collider 2D and check the "IsTrigger" to true so it can detect any collisions and trigger events



```
Material             None (Physics Material 2D)
Is Trigger           ✓
Used By Effector
Used By Composite
Auto Tiling
Offset               X 0        Y 0
Size                 X 3        Y 3
Edge Radius          0
Info
  Attached Body      None (Rigidbody 2D)
  Friction           0.4
  Bounciness         0
  Shape Count        1
  Bounds
           Center    X 74.2345   Y -4.1436  Z 0
           Extent    X 1.045502  Y 1.1364   Z 0
```

Add a script and name it "finish line" or anything you want.



```
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if(collision.transform.name == "PlayerCharacter")
        {
            SceneManager.LoadScene("WinScreen");
        }
    }
}
```

As usual use "OnTriggerEnter2D" so the object can detect the player colliding with it.

Use "Collision.transform.name == "[insert name of player]" as the condition when the finish line collides with the player and add a scene manager load scene request for the win screen.

Remember to add scene management to the namespace!!!

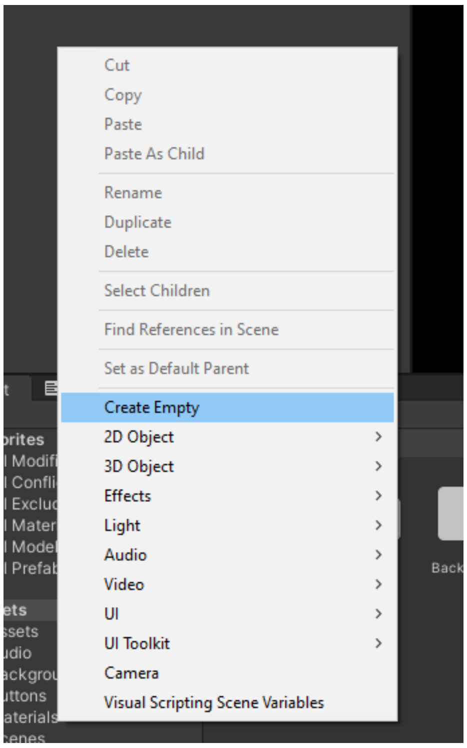
Remember to save your work and if everything goes well it should redirect you to the corresponding win screen!!!

Optional: Adding music to your game!

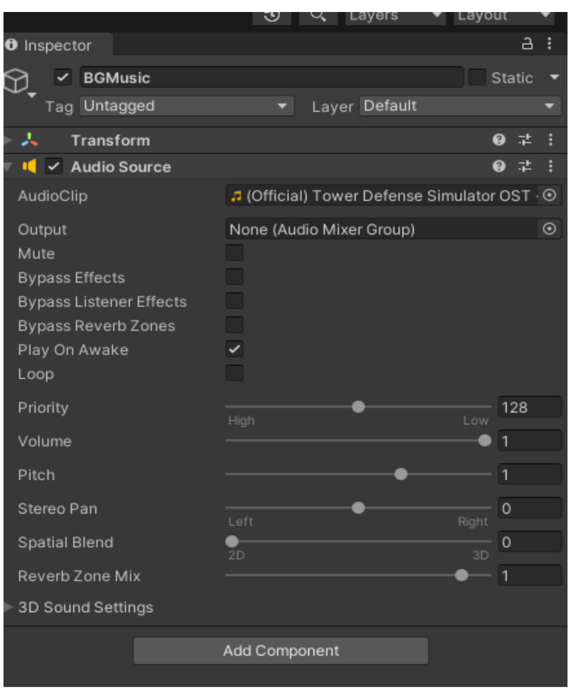
This section is optional. But now you have implemented your win/lose screens feel free to add funny music to it!!!

The steps are pretty simple!

Step 1: Create an empty game object



Simply right click the hierarchy and select create empty.



Name	BGMusic
Tag	Untagged
Layer	Default
AudioClip	(Official) Tower Defense Simulator OST
Output	None (Audio Mixer Group)
Mute	☐
Bypass Effects	☐
Bypass Listener Effects	☐
Bypass Reverb Zones	☐
Play On Awake	✓
Loop	☐
Priority	128
Volume	1
Pitch	1
Stereo Pan	0
Spatial Blend	0
Reverb Zone Mix	1

Name this game object BGM or background music or whatever name you want!

Now add an audio source component then import any music or audio file into your assets library.

After that you drag your audio to the audio clip section and you are done!

Your music will now play throughout the game!