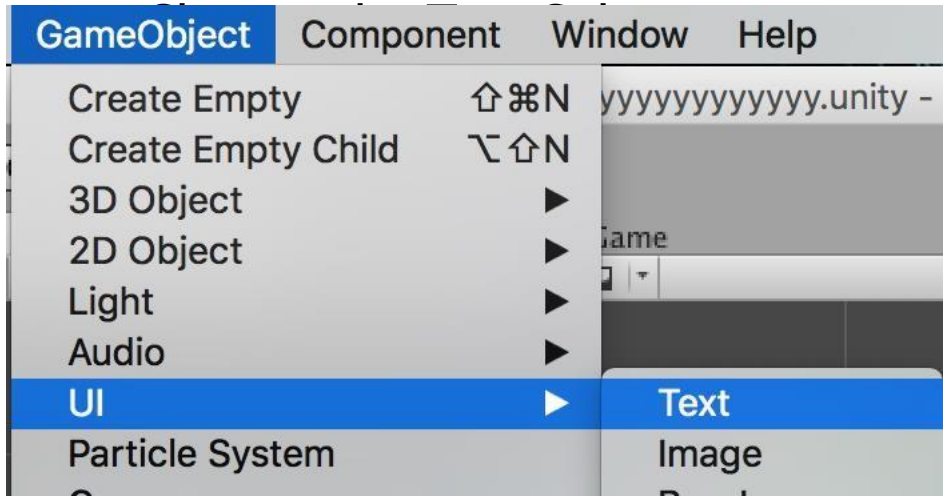
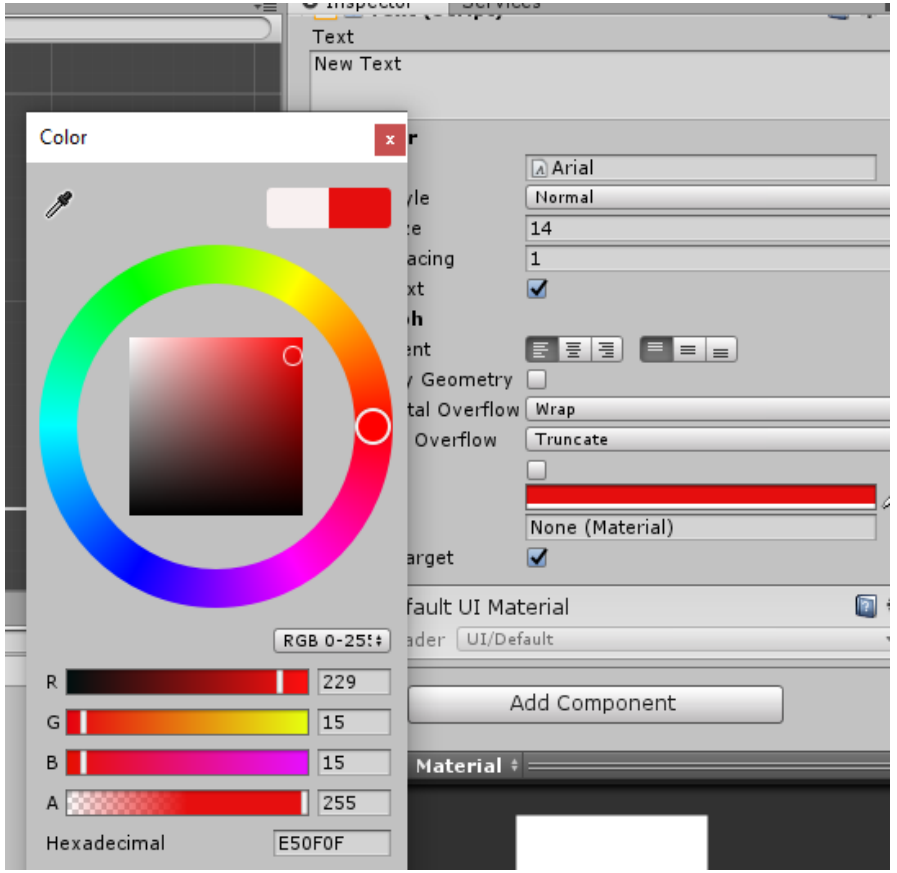
Score|Health|SceneTransitions|Pause & Resume

8

Exercise #34 - Score UI

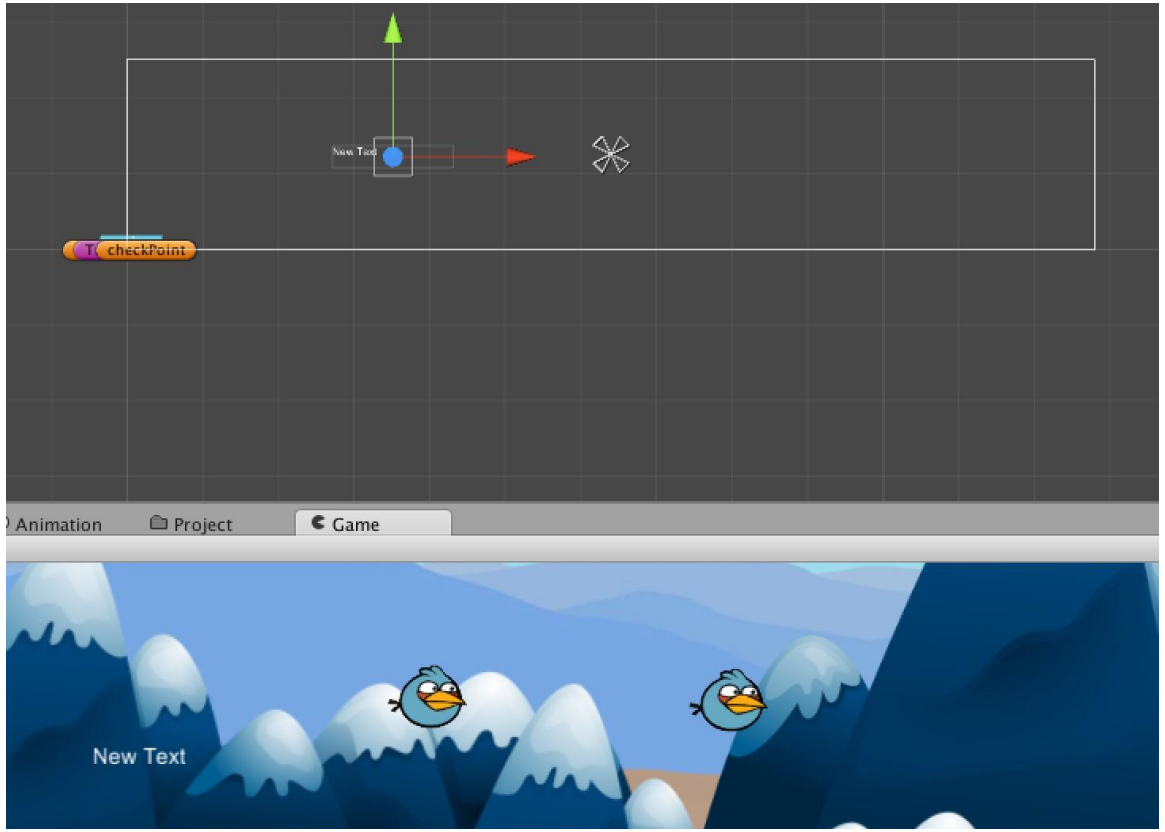
- You'll need to create text that appears on your game window at the top left that makes you see your score and remaining health!
- Go to GameObject > UI > and select **Canvas**
- Go to GameObject > UI > select **Text - TextMeshPro** for the Score label
- Write a suitable name for it: "Score"





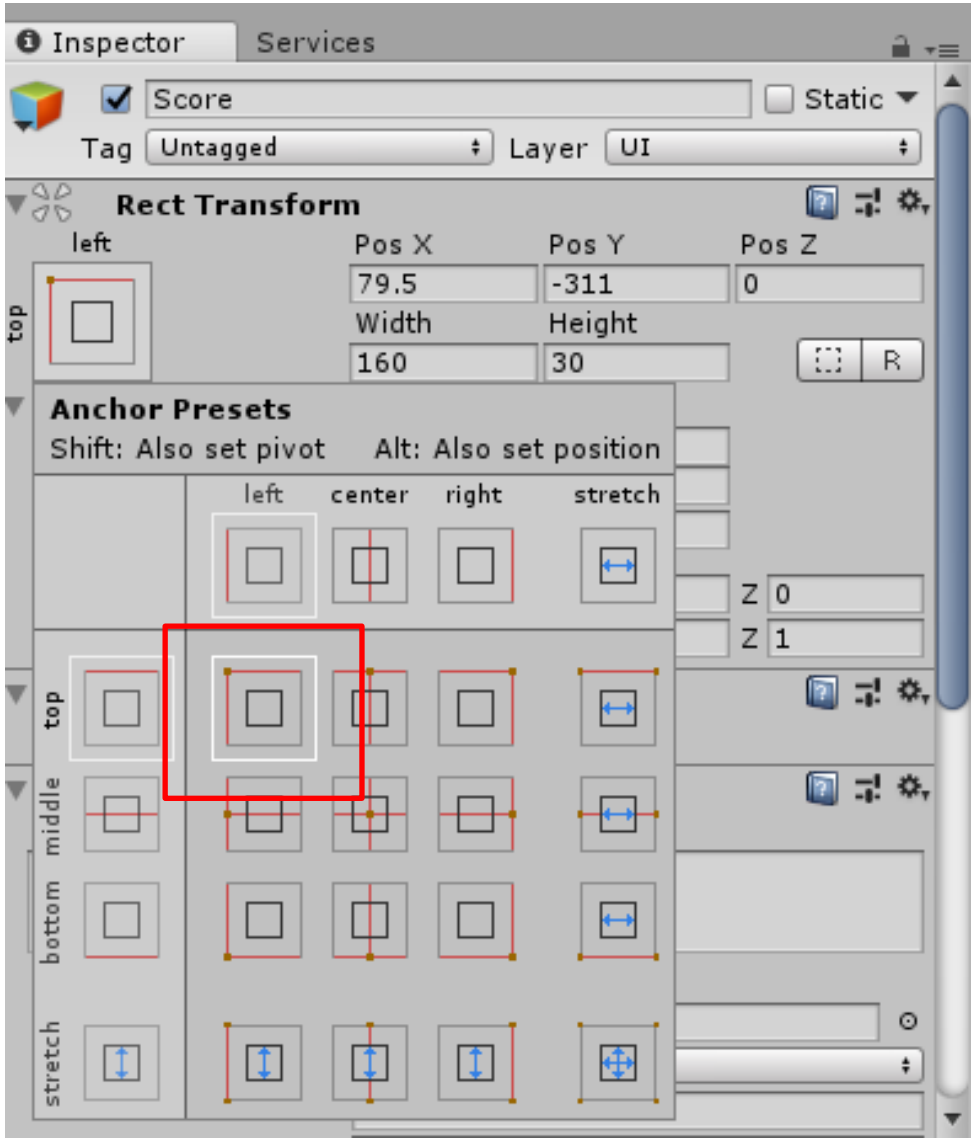
Exercise #34 - Score UI(Cont.)

- Zoom out to see your Score Text position
- Its position is relative to the anchor you see in your scene
- Experiment with the positioning till you get the look you want
- Normally, the score is placed at the top left of your game level



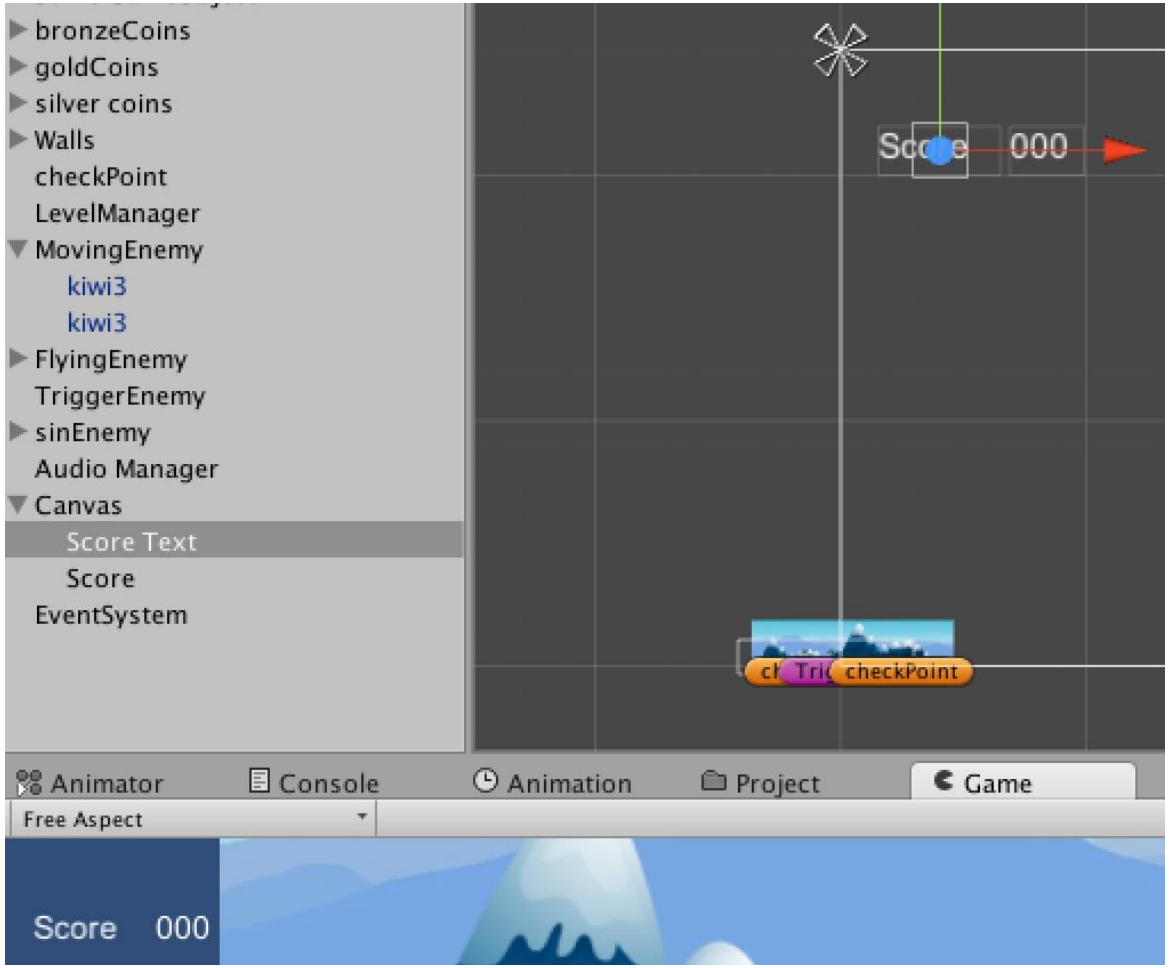
Exercise #34 - Score UI(Cont.)

- Change the anchor position of “Score” to be on top left of the screen



Exercise #34 - Score UI(Cont.)

- Add another TextMeshPro Control for the actual Score value
- Write a suitable name for it: “Score Value”
- Change its Text Color



Exercise #34 - Score UI(Cont.)

- In the “PlayerStats” class:
  - Add the following variable

```
public TextMeshProUGUI ScoreUI;
1 reference
```

  - UI tools and TMP classes are inside these namespaces so you need to add:

```
using UnityEngine.UI;
using TMPro;
```

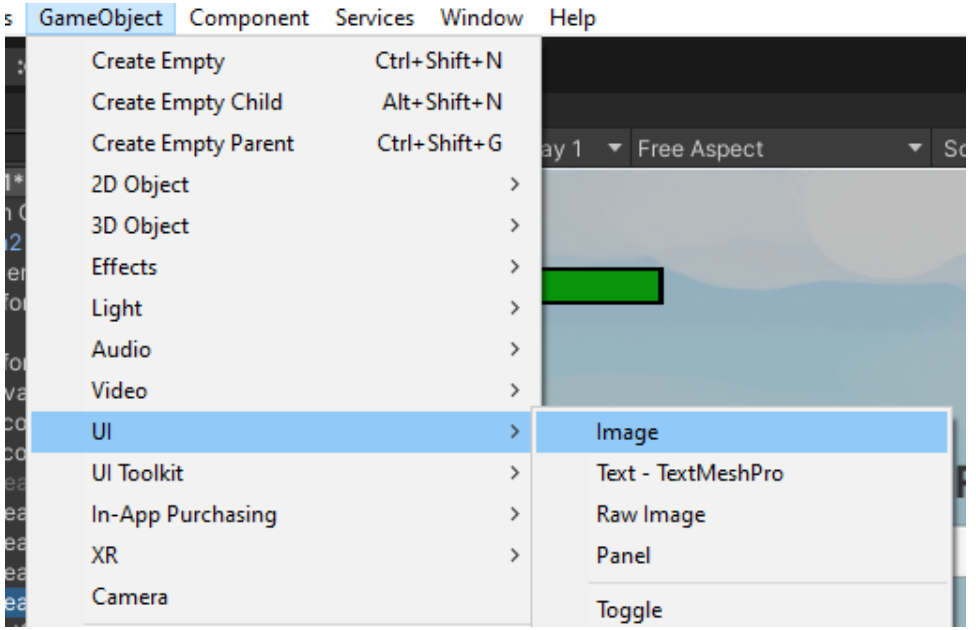
  - Don’t forget to update scoreUI’s value in Update() function:

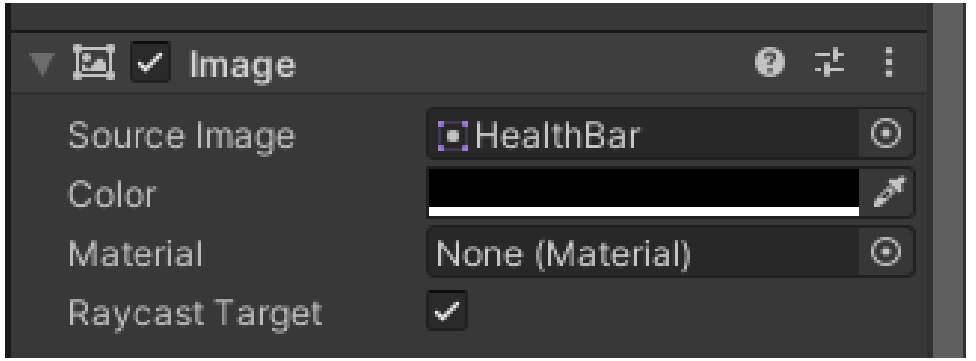
```
ScoreUI.text = "" + coinsCollected;
```

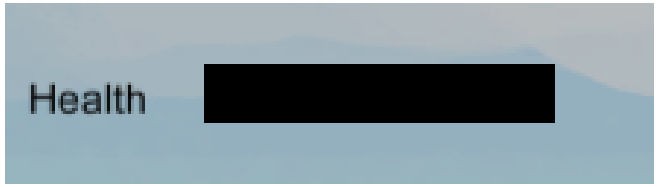
  - Go back to Unity, Drag and drop the Score control into scoreUI variable of “PlayerStats” class

Exercise #35 - Health Bar UI

- To create a Health Bar:
- Create a Text - TextMeshPro UI again and change it to the word ‘Health’
- Download the 3 health bars from e-learning into your Unity project
- Right click on Canvas > Choose UI > Image
- Drag the BLACK health bar into the Image’s Source Image
- If the bar looks skewed, click the **Set to Native Size** button, then change the dimensions as you wish
- Edit its position according to its top left anchor





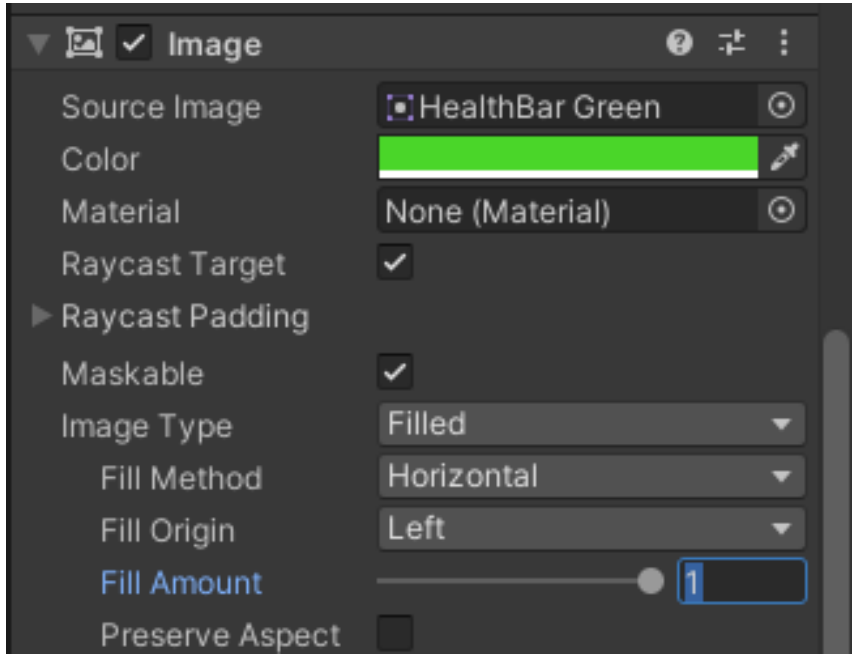


Exercise #35 - Health Bar UI

- Now create a 2nd Image, and this one will contain the RED health bar
- Resize it so it's slightly smaller than the black one, and place it on top of the black one

- Now create a 3rd Image, and this one will contain the GREEN bar and it is the SAME size as the red bar and placed on top of it

- Make sure that 'Maskable' is checked in
- The green bar, and set the properties
- below it as shown in the screenshot on the right.



Exercise #35 - Health Bar UI

Go to PlayerStats script and add the following library:

```
using UnityEngine.UI;
```

Add an Image variable and call it HealthBar for ex.:

```
public Image healthBar;
```

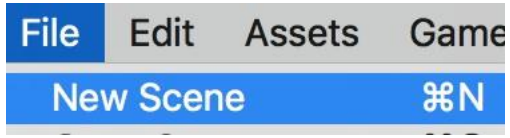
In TakeDamage() function, amend to add the line below.

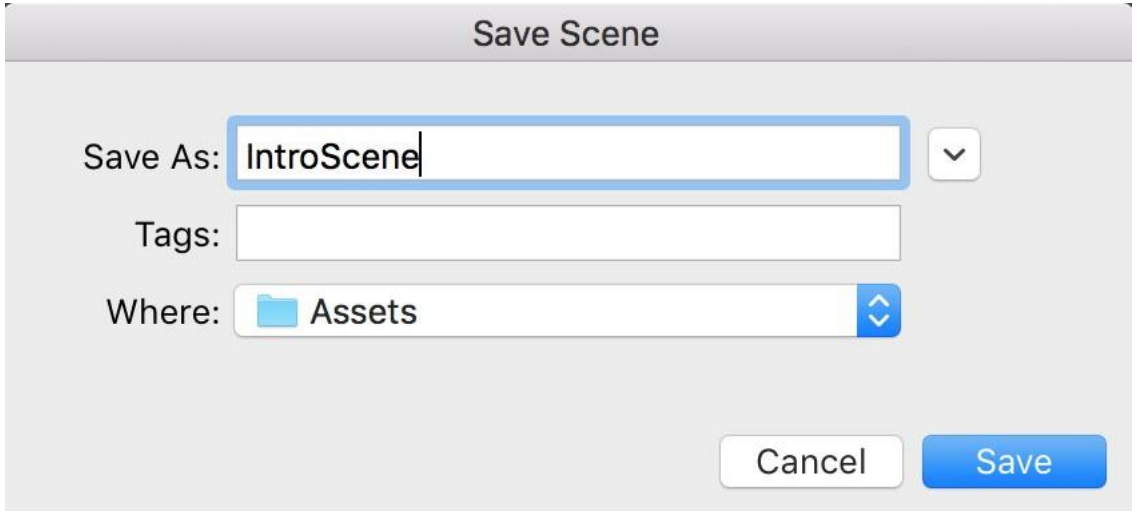
```
this.health = health - damage;
healthBar.fillAmount = this.health/3f;
```

NOTE: Change 100f to whatever health total YOUR player has!

Exercise #36 - Adding Scenes

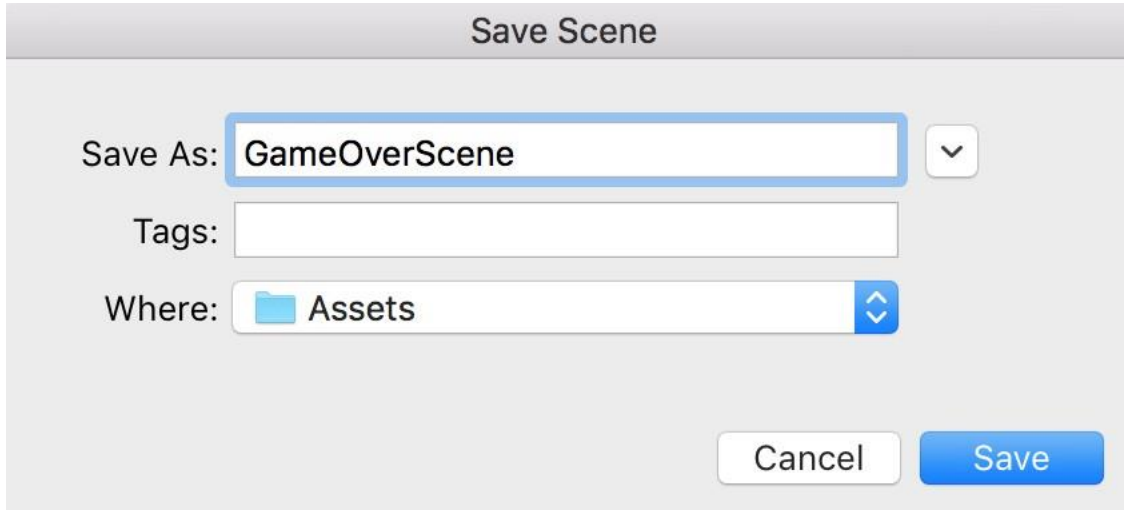
- Click on File then New Scene
- Save Scene “IntroScene”

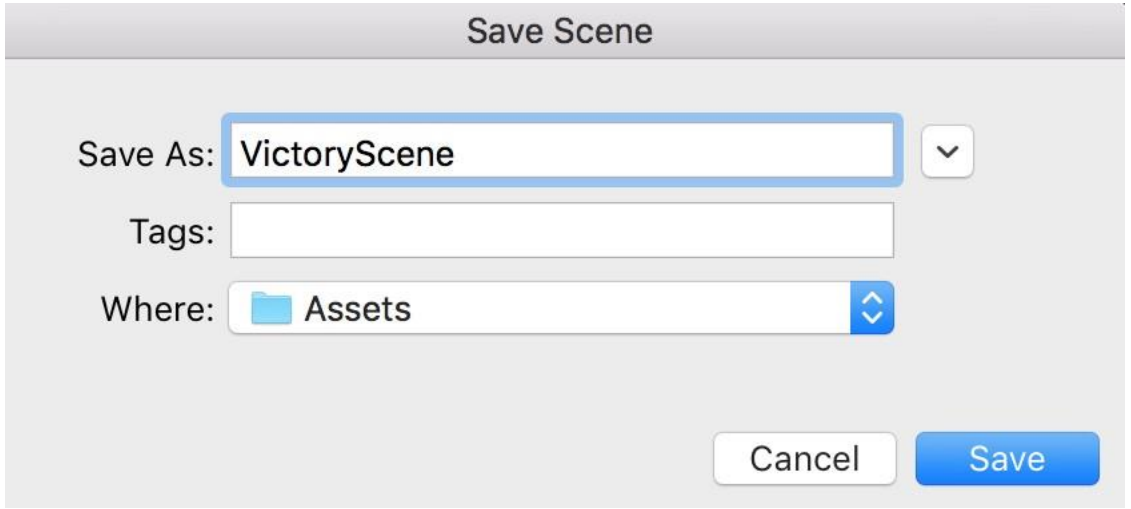




Exercise #36 - Adding Scenes (Cont.)

- Add another two Scenes
  - GameOverScene
  - VictoryScene





Exercise #36 - Adding Scenes (Cont.)

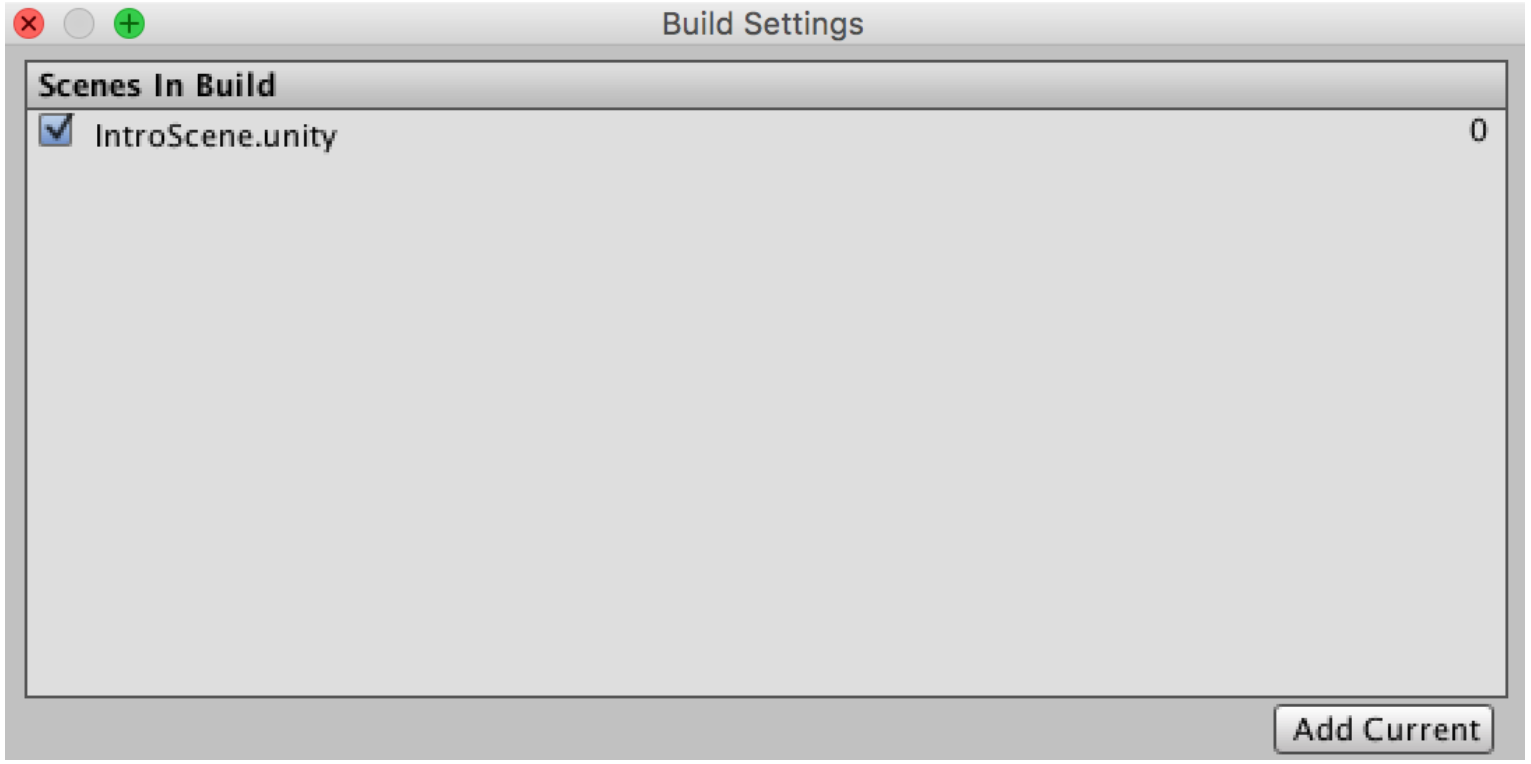
- Also don’t forge to edit your actual game level’s Scene name

Exercise #36 - Adding Scenes (Cont.)

- Add all Scenes to Build Settings

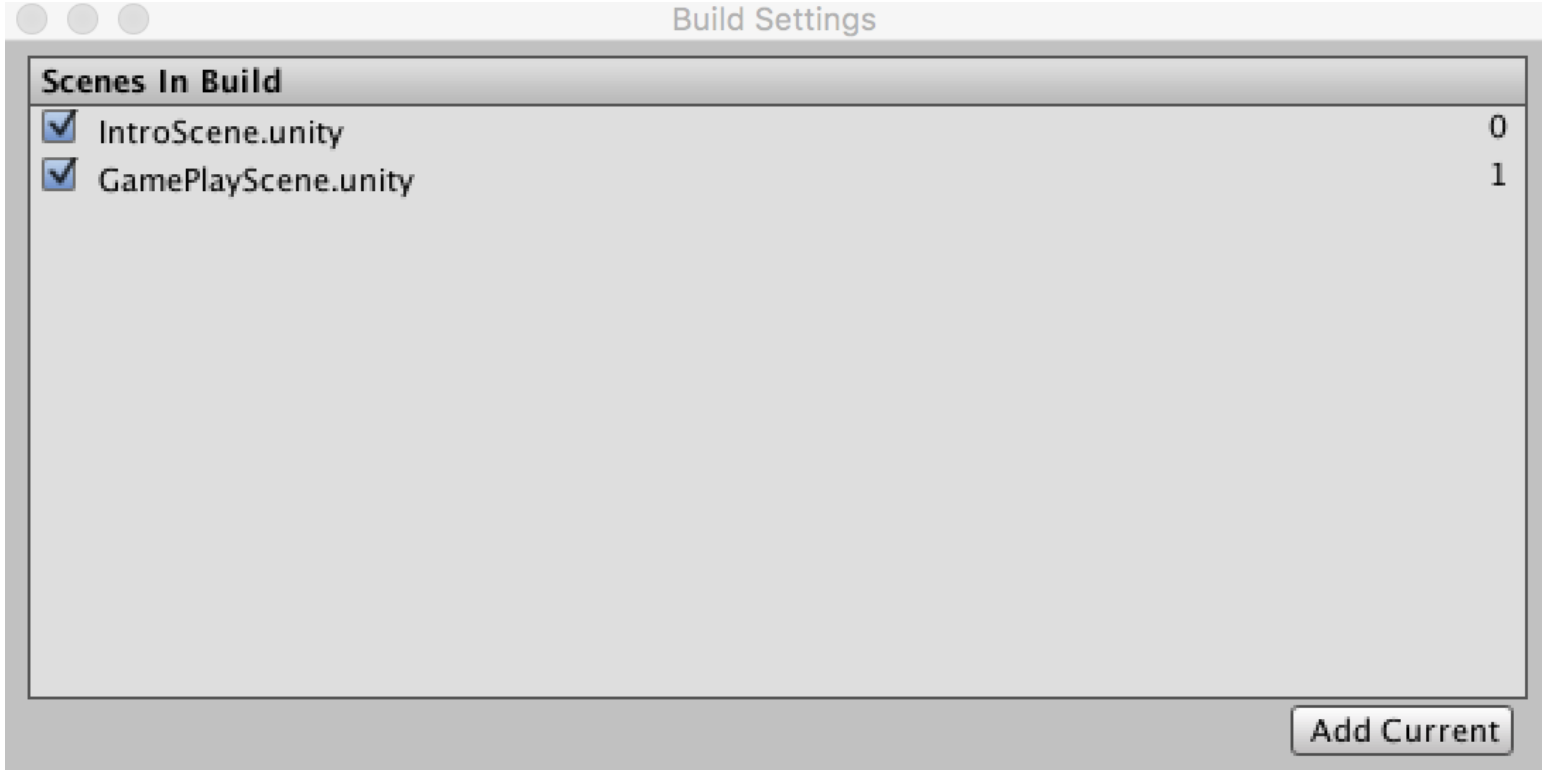
Exercise #36 - Adding Scenes (Cont.)

- Open IntroScene then click on “Add Current” button



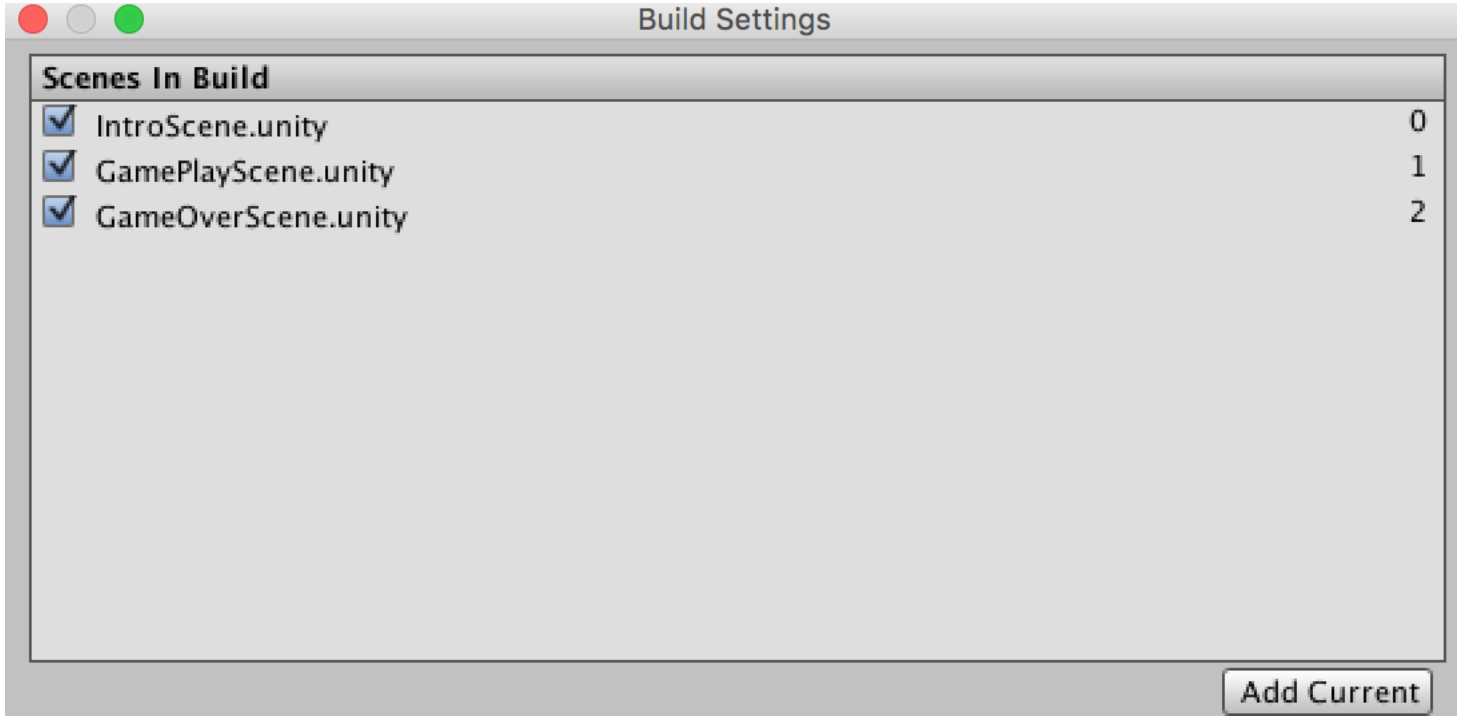
Exercise #36 - Adding Scenes (Cont.)

- Open GamePlayScene then click on “Add Current” button



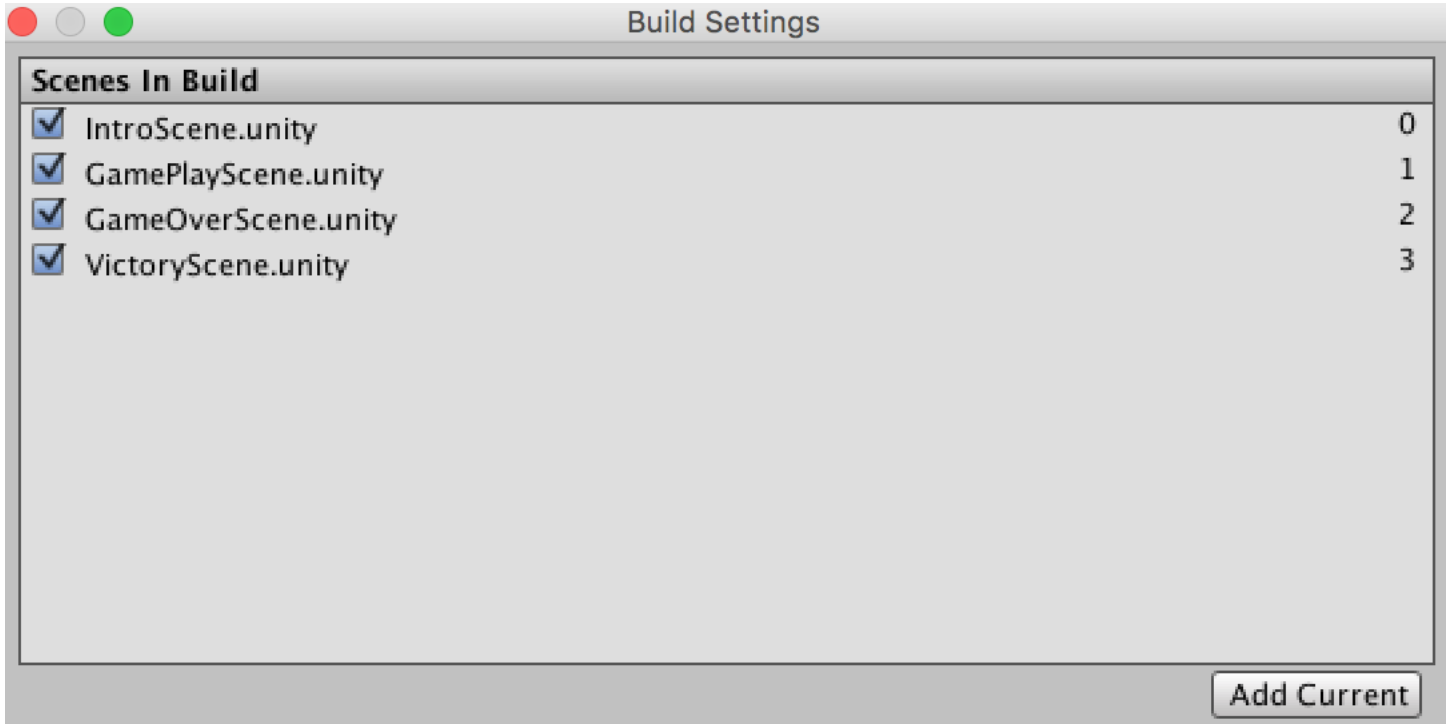
Exercise #36 - Adding Scenes (Cont.)

- Open GameOverScene then click on “Add Current” button



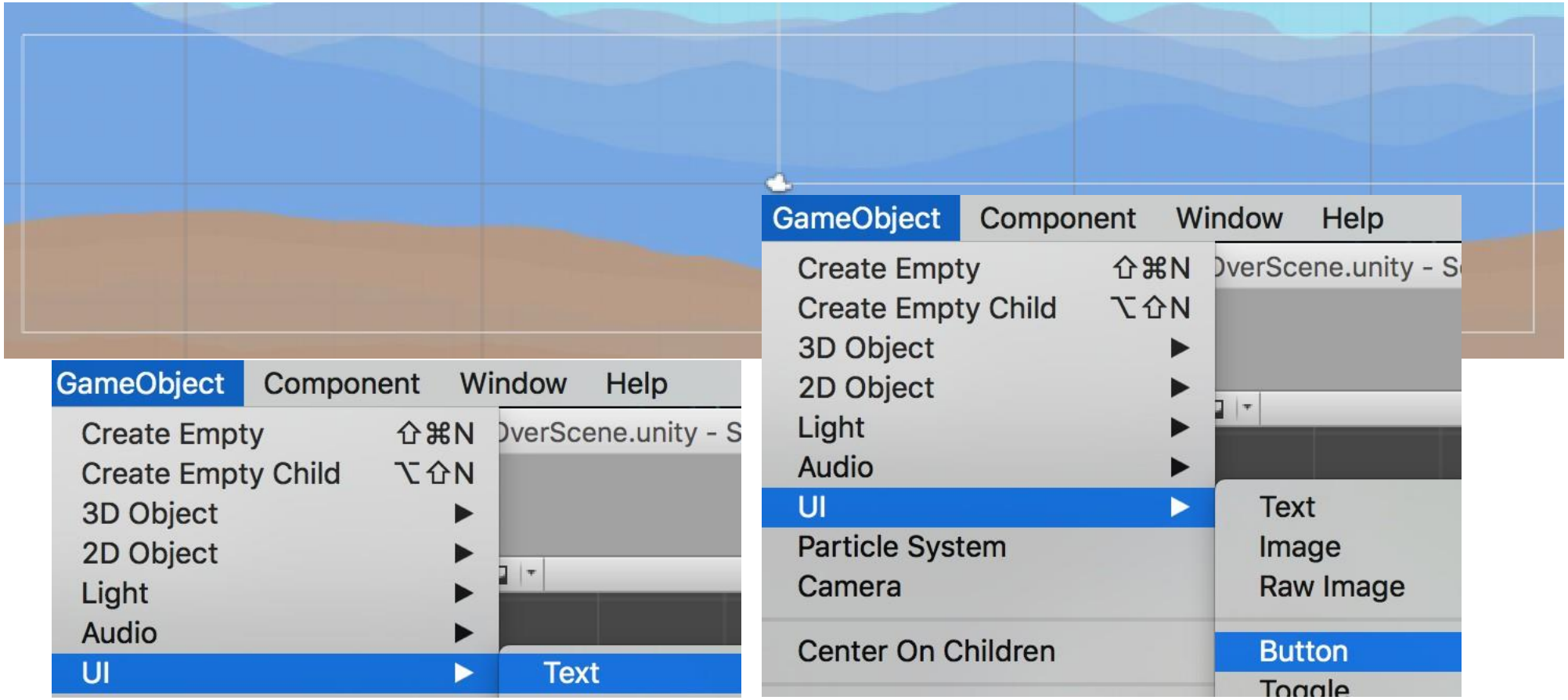
Exercise #36 - Adding Scenes (Cont.)

- Open VictoryScene then click on “Add Current” button
- Index beside each scene will be used to navigate between them



Exercise #37 - Intro Scene

- Add a background, a Text UI control and a Button UI control



Exercise #37 - Intro Scene (Cont.)

- Align them in the center using the centered anchor

Exercise #37 - Intro Scene (Cont.)

- Add a new Script to the canvas“ called NavigationController”

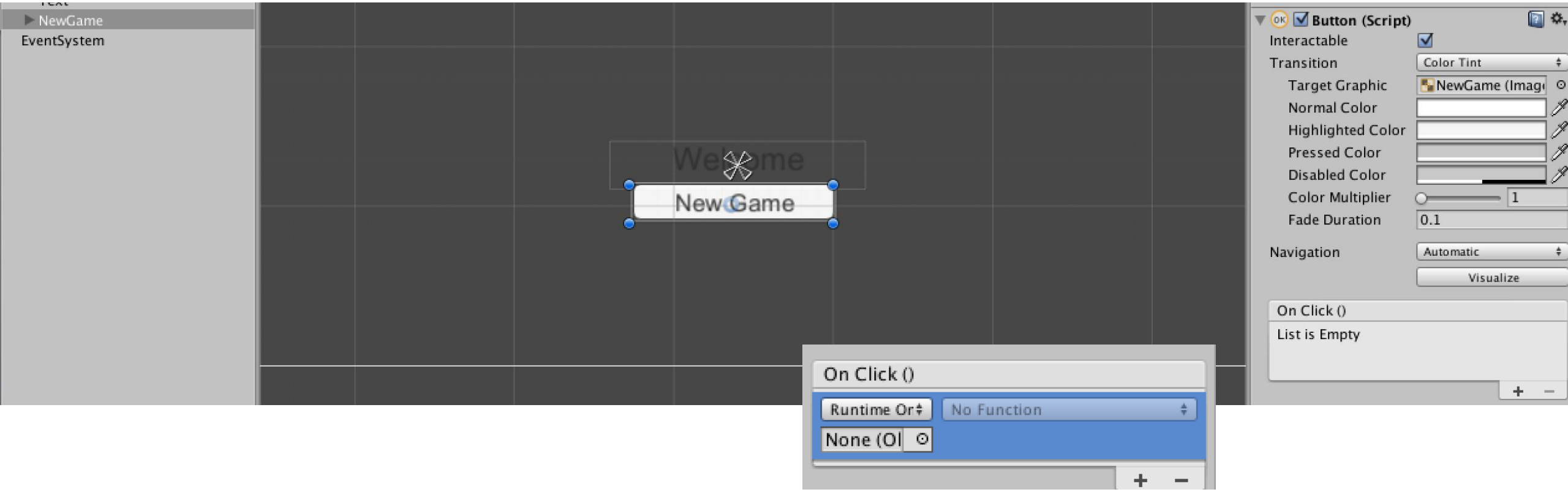
Exercise #37 - Intro Scene (Cont.)

- “Navigation Controller ”script is responsible for navigating between all scenes
- LoadLevel is a static function inside Application class which loads a scene identified by its index in build settings

```
public class NavigationController : MonoBehaviour {

    public void GoToIntroScene()
    {
        Application.LoadLevel(0);
    }

    public void GoToGameScene()
    {
        Application.LoadLevel(1);
    }

    public void GoToGameOverScene()
    {
        Application.LoadLevel(2);
    }

    public void GoToVictoryScene()
    {
        Application.LoadLevel(3);
    }

    public void Quit()
    {
        Application.Quit ();
    }
}
```

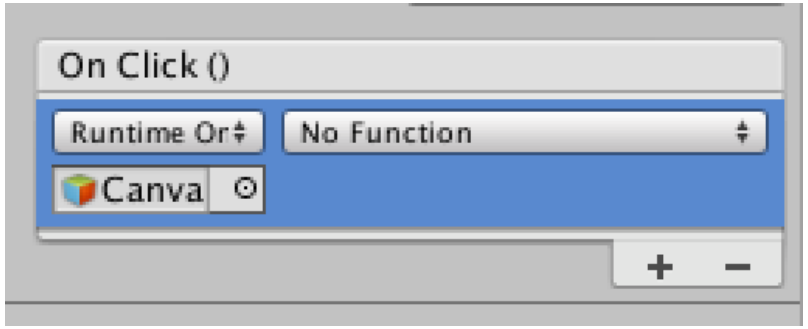
Exercise #37 - Intro Scene (Cont.)

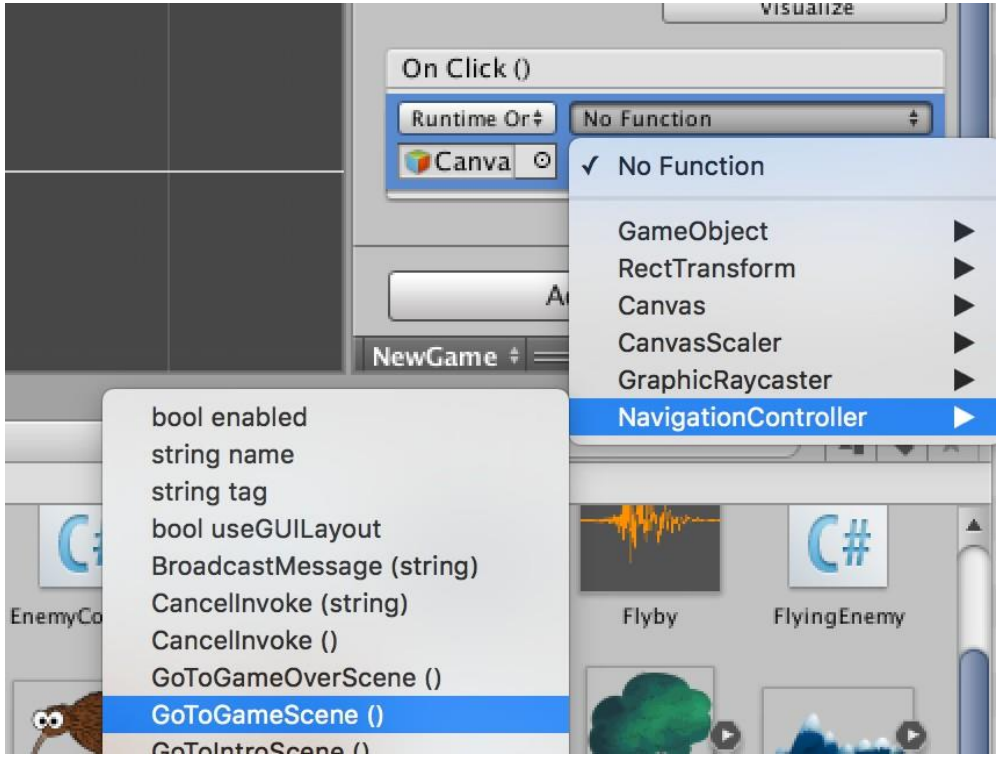
- Select NewGame button
- Add a new On Click handler



Exercise #37 - Intro Scene (Cont.)

- Drag and drop the canvas
- Then choose the GoToGameScene function of NavigationController class

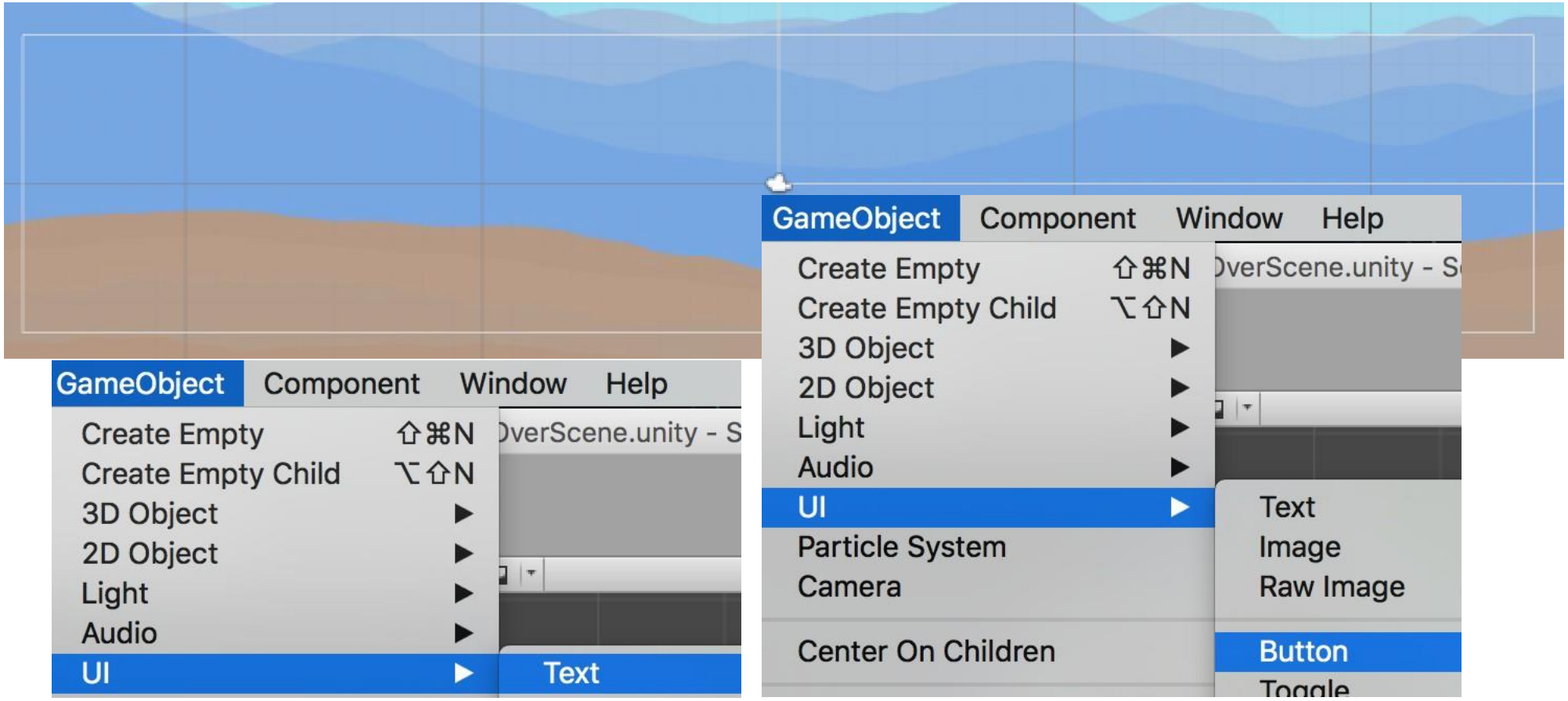




Exercise #37 - Intro Scene (Cont.)

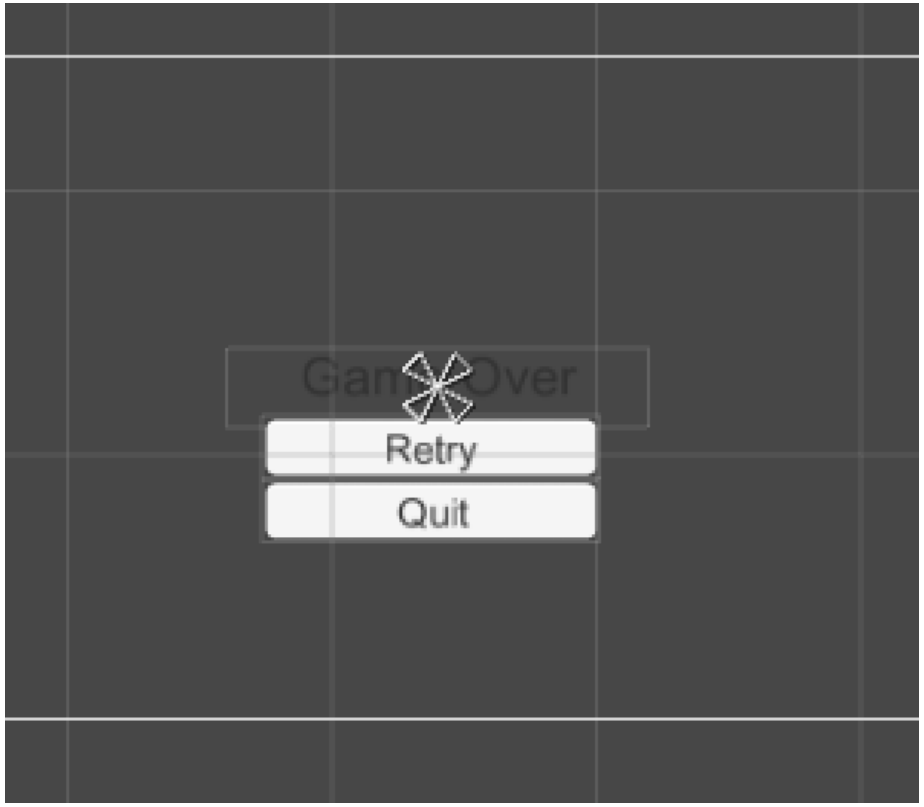
Exercise #37 – Game Over Scene

- Add a background, a Text UI control and 2 Buttons UI control



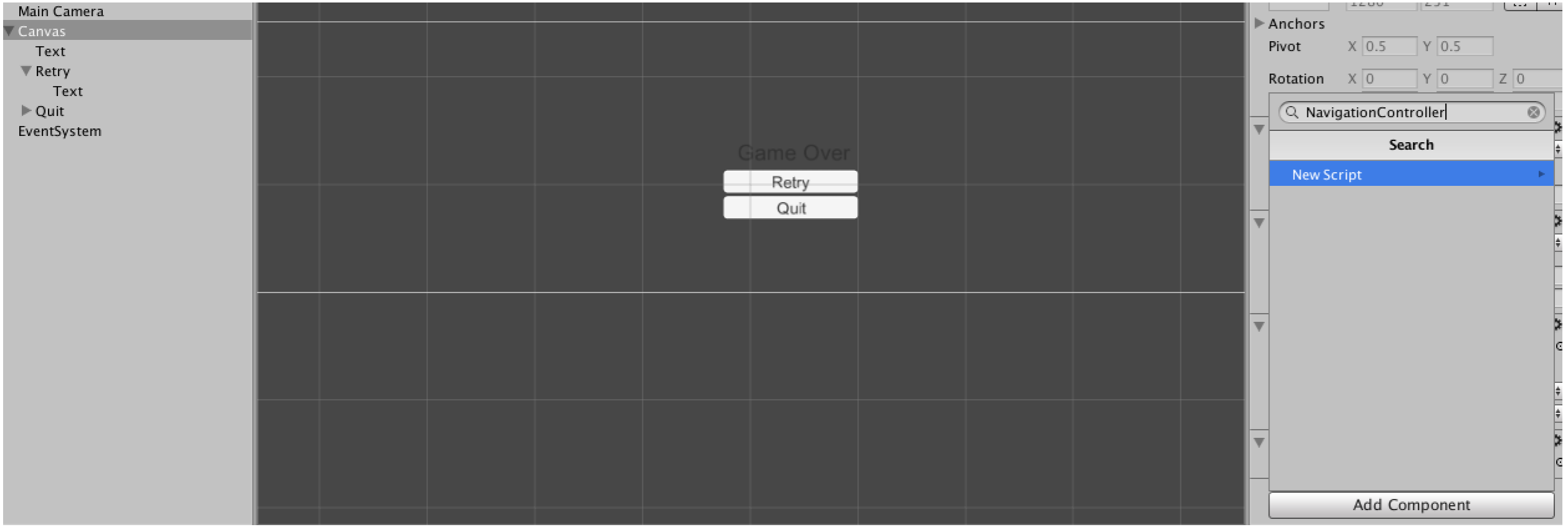
Exercise #37 – Game Over Scene (Cont.)

- Align them with their centered anchor



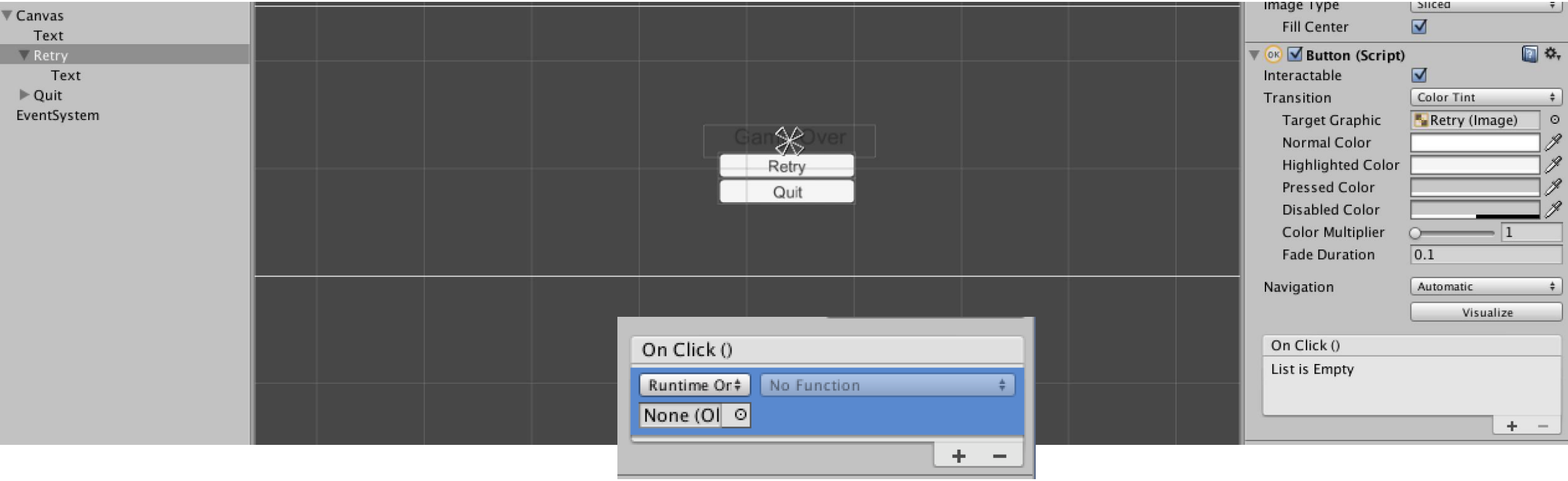
Exercise #37 – Game Over Scene (Cont.)

- Add “NavigationController ”script to the canvas



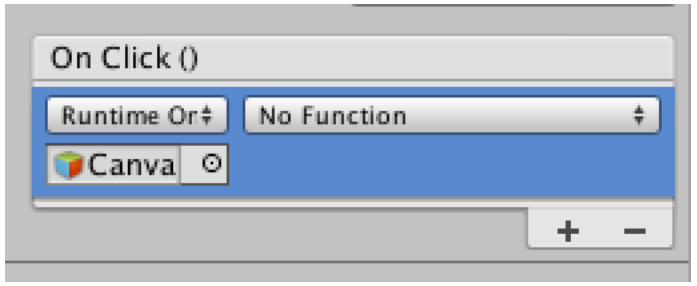
Exercise #37 – Game Over Scene (Cont.)

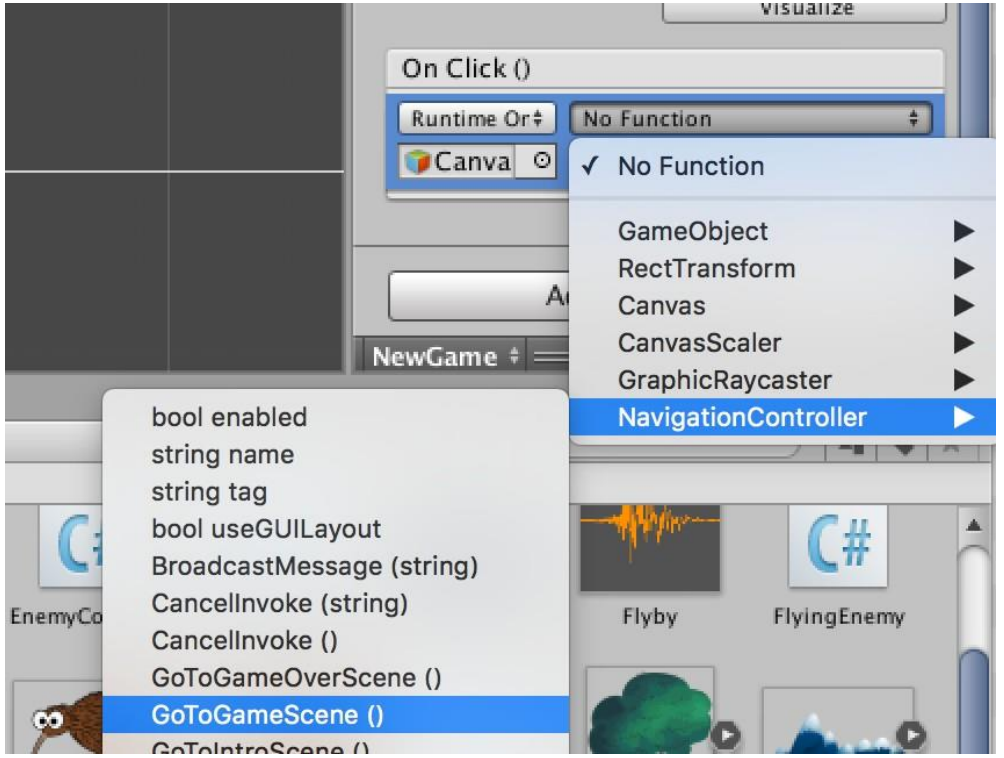
- Select Retry button
- Add a new On Click handler



Exercise #37 – Game Over Scene (Cont.)

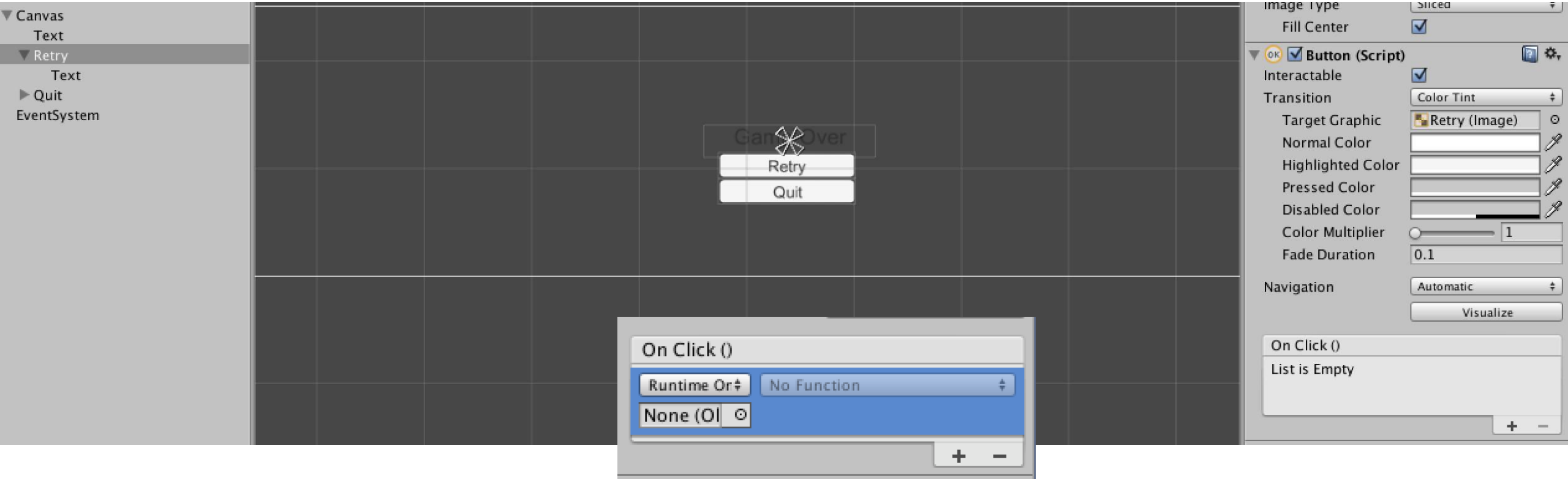
- Drag and drop the canvas
- Then choose the GoToGameScene function of NavigationController class





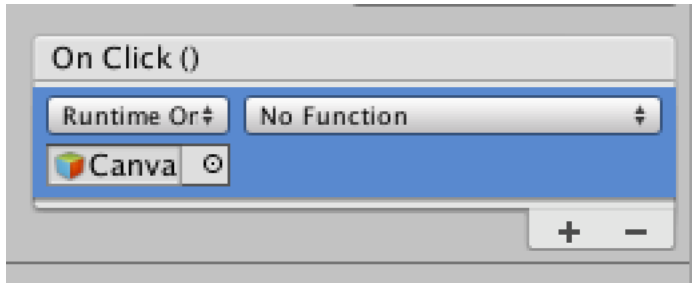
Exercise #37 – Game Over Scene (Cont.)

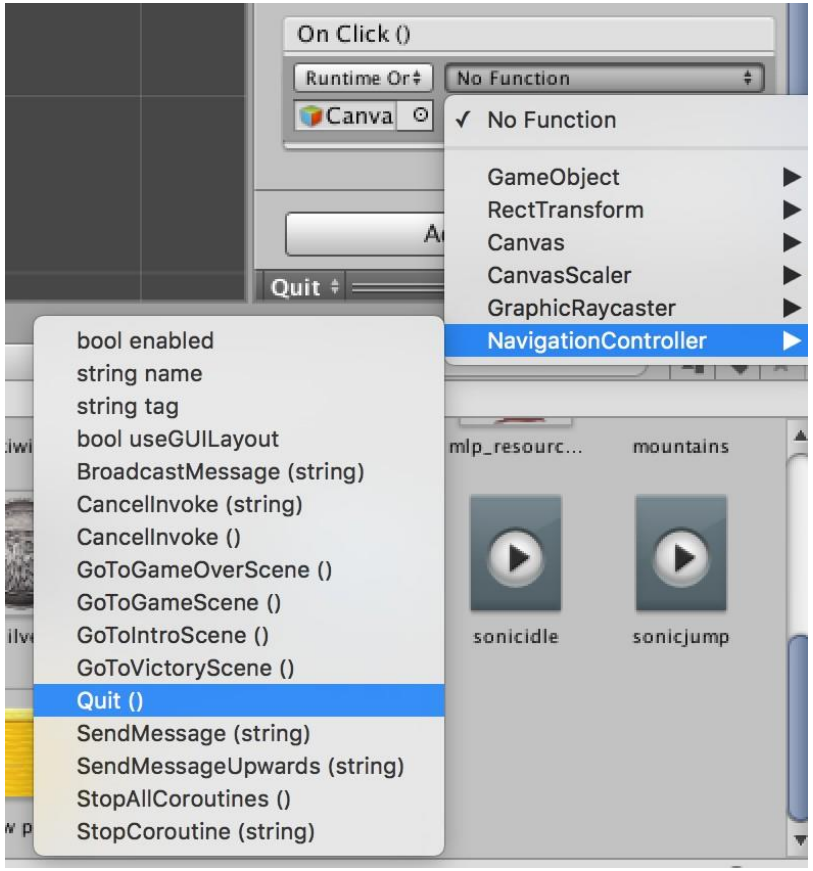
- Select Quit button
- Add a new OnClick handler



Exercise #37 – Game Over Scene (Cont.)

- Drag and drop the canvas
- Then choose the Quit function of NavigationController class





Exercise #37 – Game Over Scene (Cont.)

- Add the following line inside the TakeDamage function of “PlayerStats ”class

```
else if (this.lives == 0 && this.health == 0)
{
    (new NavigationController()).GoToGameOverScene();
    Debug.Log("Gameover");
    Destroy(this.gameObject);
}
```

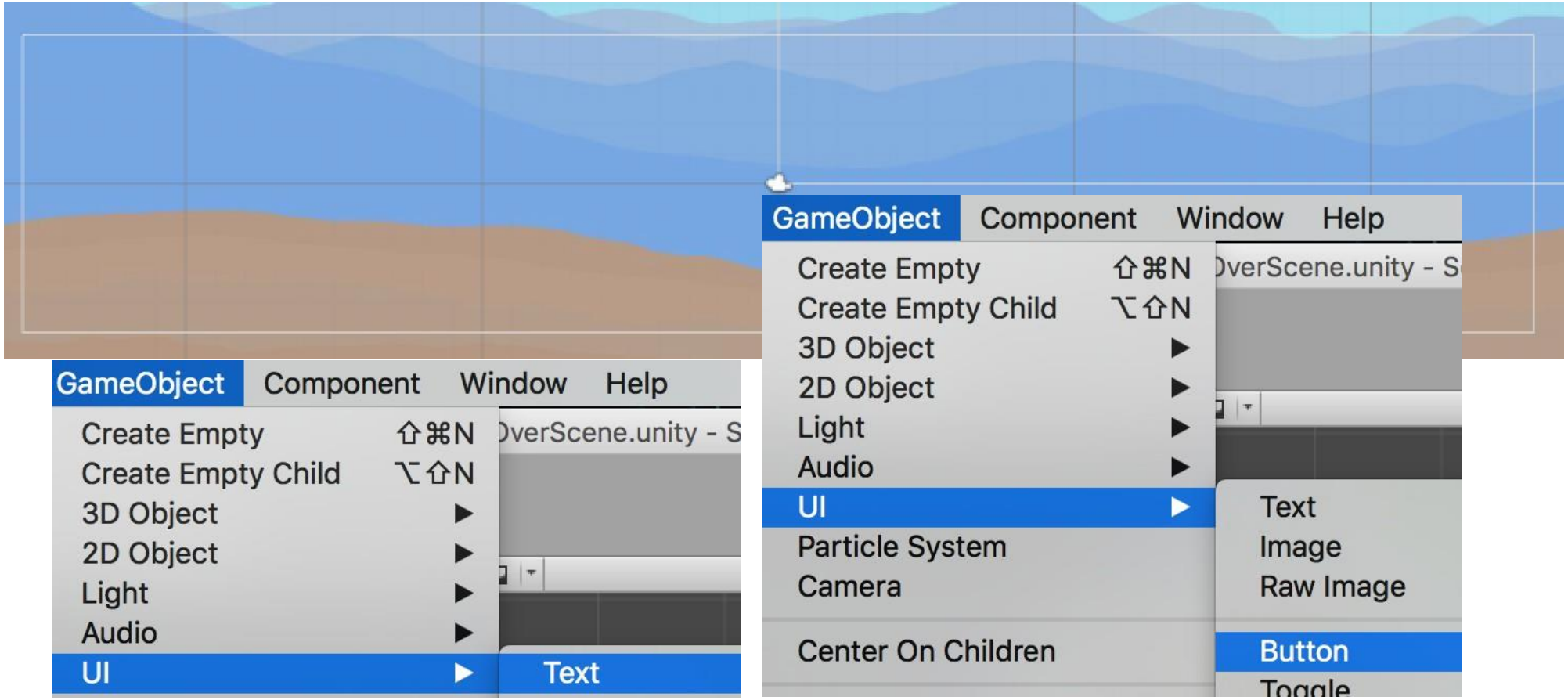
USEFUL TIP:

- You also have the option of creating a static instance of NavigationController beforehand, as we did with AudioManager in the previous tutorial. Then you won’t need to use “new NavigationController() above. You’ll just use the static instance and call the function when you need it
  (e.g. variable in NC class: public static NavigationController *navinstance*
  Function call: NavigationController.navinstance.GoToGameOverScene())

Exercise #37 – Game Over Scene (Cont.)

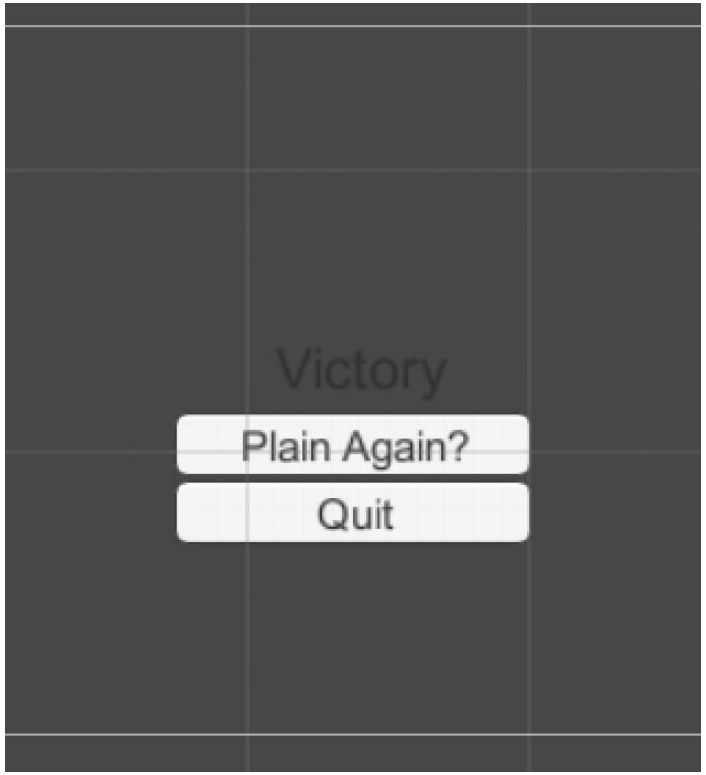
Exercise #37 – Victory Scene

- Add a background, a Text UI control and 2 Buttons UI control



Exercise #37 – Victory Scene (Cont.)

- Align them with their centered anchor
- Add “NavigationController ”script to the canvas
- Do the same steps of GameOver scene steps except that the retry button’s text will be “Play Again”?



Exercise #37 – Victory Scene (Cont.)

- Add the following inside the CollectCoin function of “PlayerStats ”class

```
if(this.coinsCollected >= 30)
{
    (new NavigationController()).GoToVictoryScene();
}
```

Exercise #37 – Victory Scene (Cont.)

Exercise #38 - Pausing and Resuming

First, we will need to create a canvas

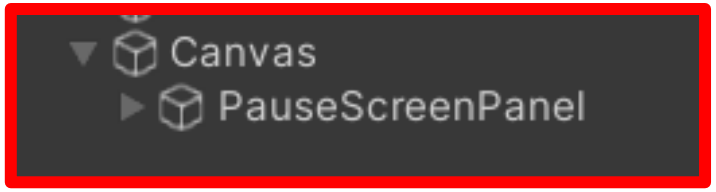
- Right click in the hierarchy panel

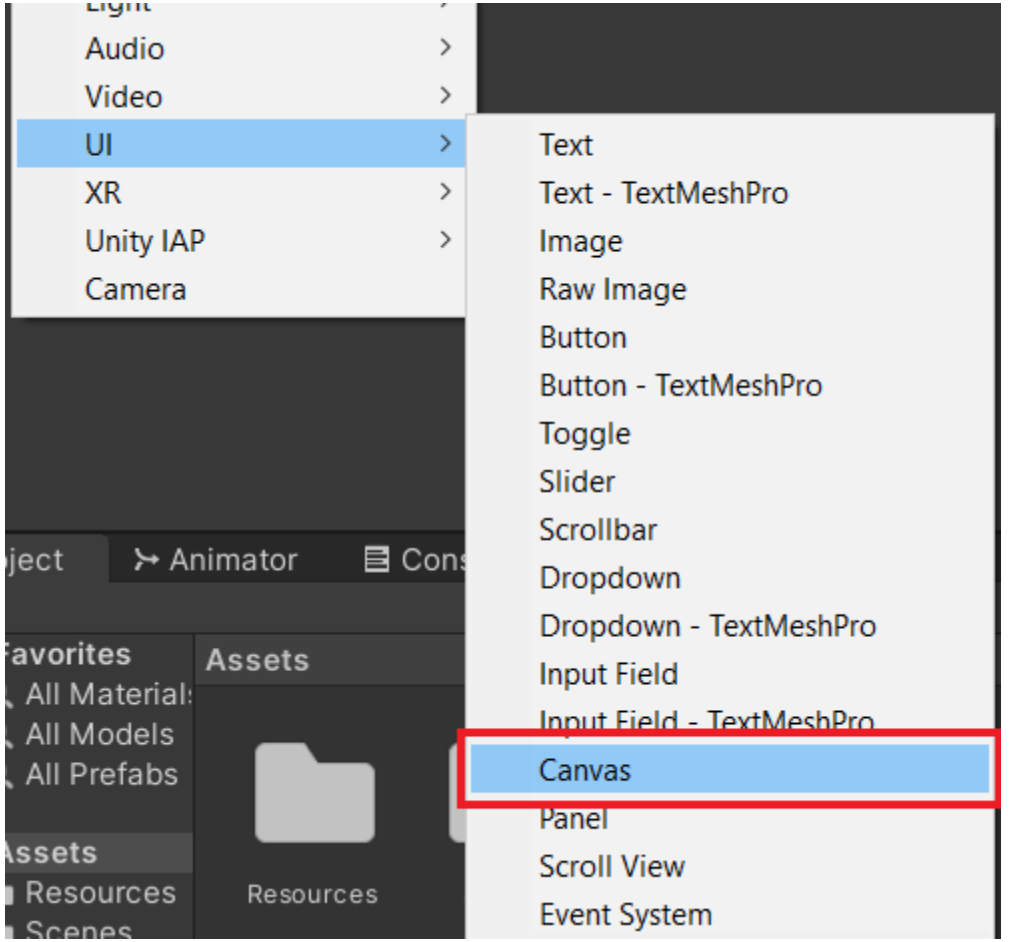
  Go to UI > and select Canvas

You can double click on the Canvas object in the hierarchy to see the full canvas.

- Then Go to UI > and select Panel

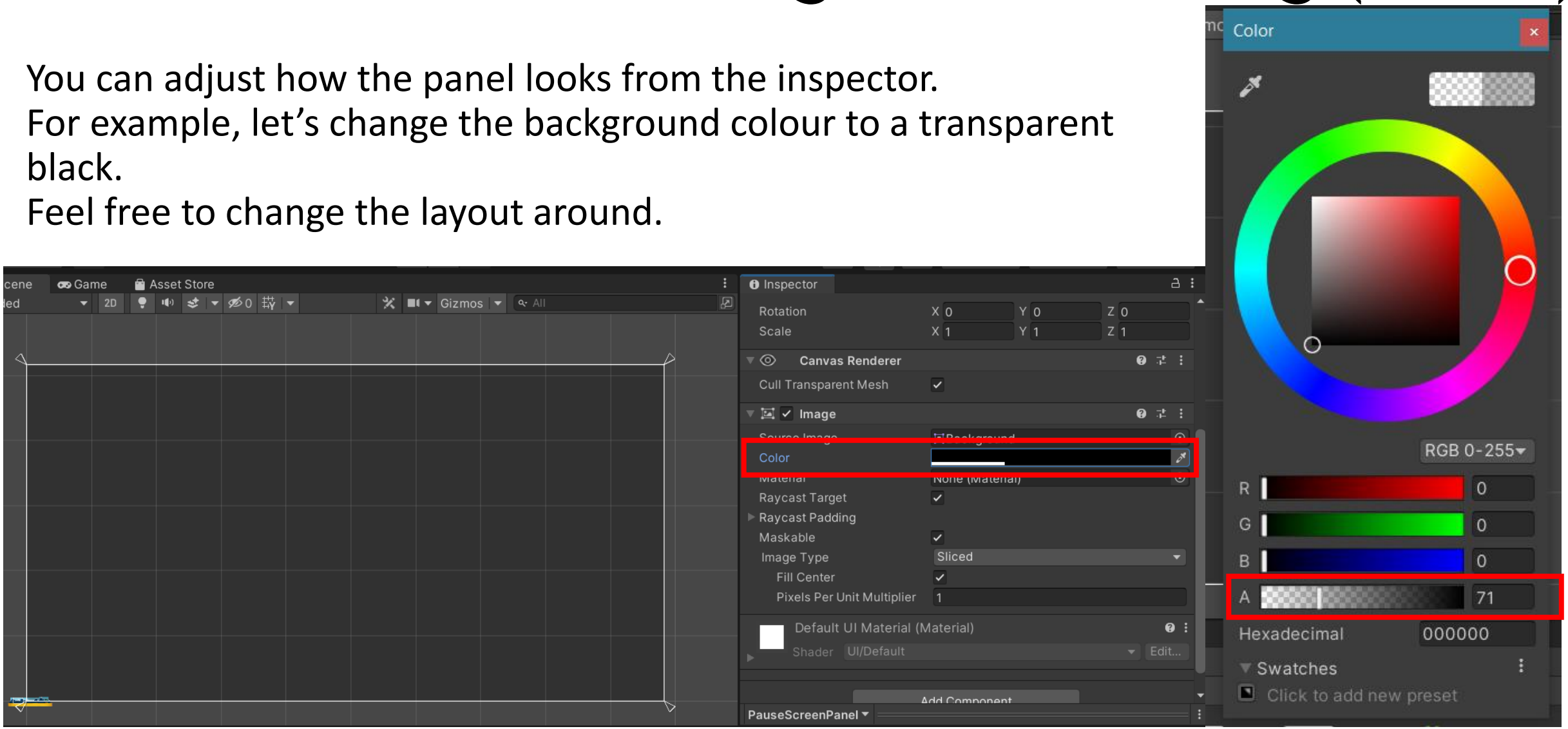
  And make it a child of the Canvas object we just created.





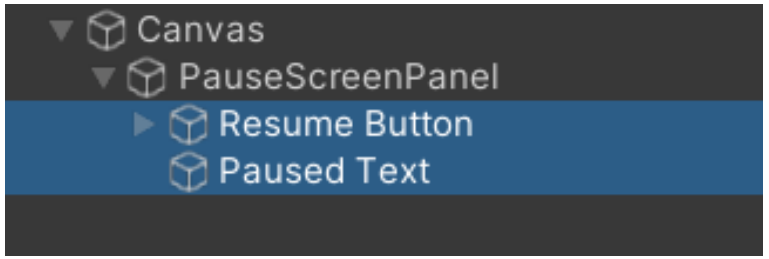
Exercise #38 - Pausing and Resuming (Cont.)

You can adjust how the panel looks from the inspector.
For example, let's change the background colour to a transparent black.
Feel free to change the layout around.



Exercise #38 - Pausing and Resuming (Cont.)

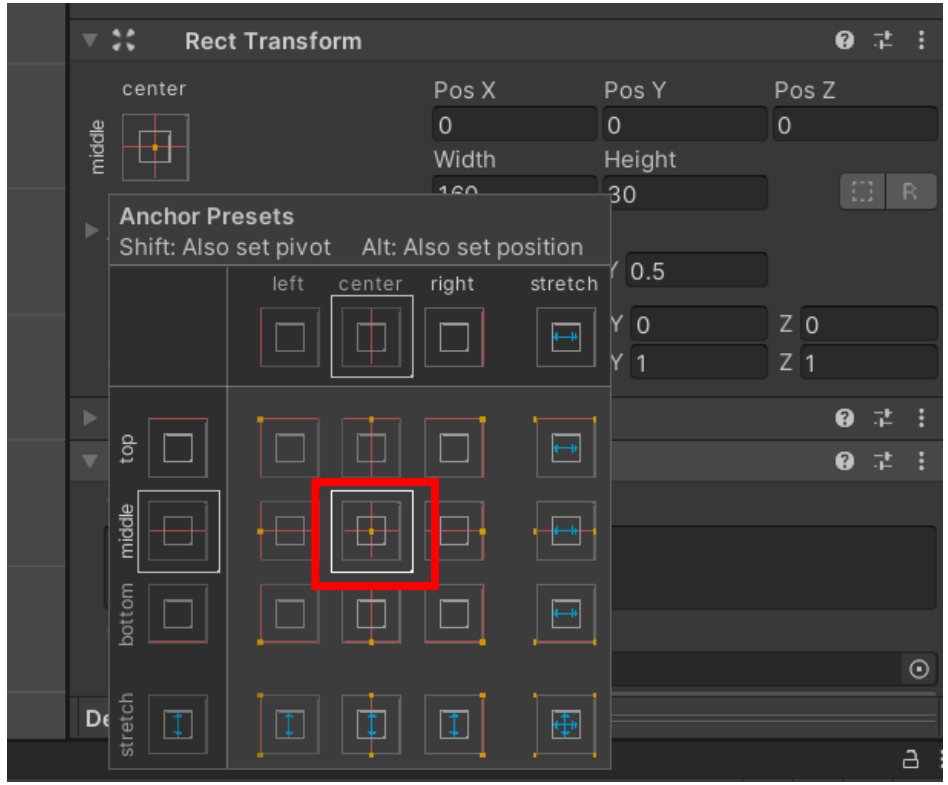
Then create a text UI object and a button as children to the Panel



Go to UI > Text

Go to UI > Button

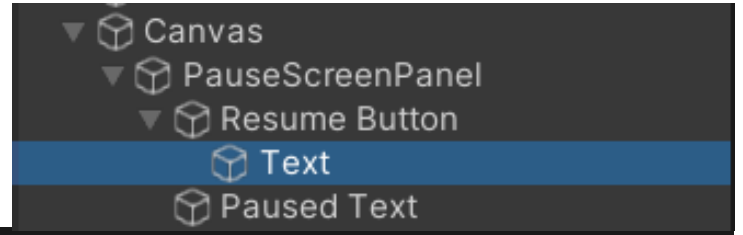
Make sure that their transform is set to middle-center

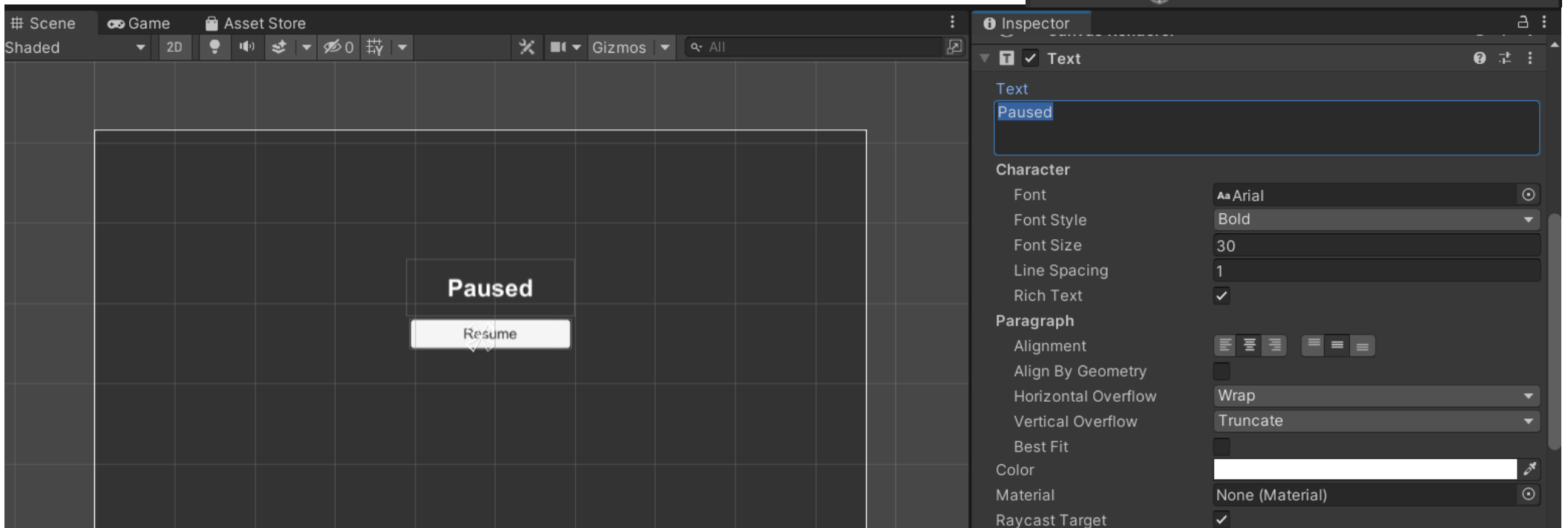


Exercise #38 - Pausing and Resuming (Cont.)

Adjust the text of the Text object and the button from the text component in the inspector.

To edit the button text, you will find a text object that is created automatically as a child to the button. You can edit the text from that object.





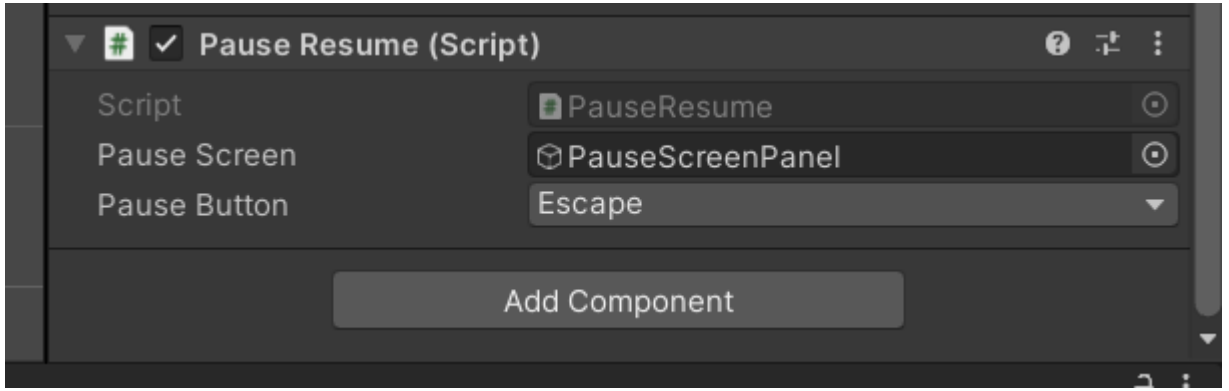
Exercise #38 - Pausing and Resuming (Cont.)

Next, we add a script to the Canvas object. Call it PauseResume and add the following variables.

- The paused Boolean variable is set to static in order to access it across all scripts (more on this later)

```
public class PauseResume : MonoBehaviour
{
    public GameObject PauseScreen;        //This will include the pause screen panel
    public static bool paused;            // this is to check if the game is already paused or not to freeze the scene
    public KeyCode PauseButton;           // the button we will use to pause the game
```

- Set the values of the variables in the inspector.
- Drag and drop the panel game object to the pause screen variable



Exercise #38 - Pausing and Resuming (Cont.) - Using Key Code

- Add the following initialization statements to the start function:

```
void Start()
{
    paused = false;
    PauseScreen.SetActive(false);          // This disables the screen so it doesn't show up when we run the game
}
```

- Check for user input in the update function.
- If the user presses on the pause button (escape button) and the game is not already paused, then pause.
- If the game is already paused and the user presses the pause button, then resume the game.

```
void Update()
{
    if (Input.GetKeyDown(PauseButton) && !paused)
    {
        Pause();
    }
    else if (Input.GetKeyDown(PauseButton) && paused)
    {
        Resume();
    }
}
```

Exercise #38 - Pausing and Resuming (Cont.) - Using Key Code

- Create the pause and resume functions

```
void Pause()
{
    PauseScreen.SetActive(true);    //enables the panel
    paused = true;
    Time.timeScale = 0;         //pauses the game
}
```

```
public void Resume()
{
    PauseScreen.SetActive(false);   //disables the panel
    paused = false;
    Time.timeScale = 1;             // resumes the game
}
```

- TimeScale is a function that allows us to play with how fast or slow time passes. A value of 1 means time passing in the game is the same as real time. Smaller values can be useful to create slow motion effects!

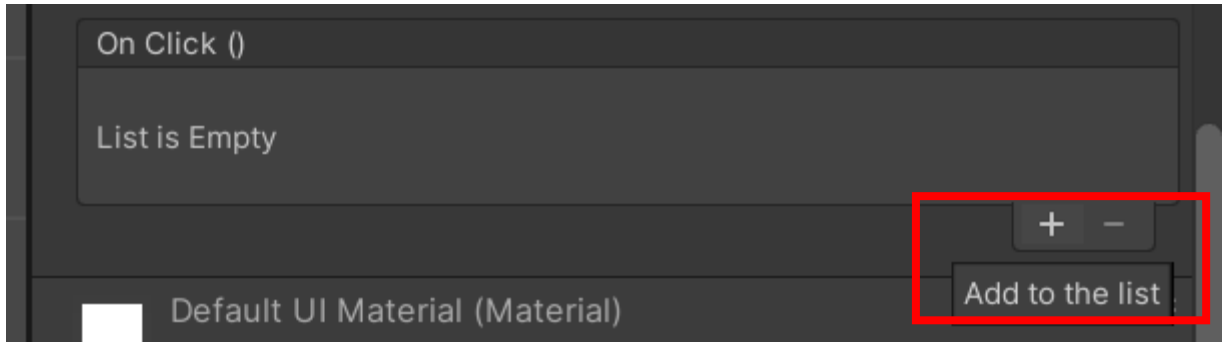
Exercise #38 - Pausing and Resuming (Cont.) - Using Key Code

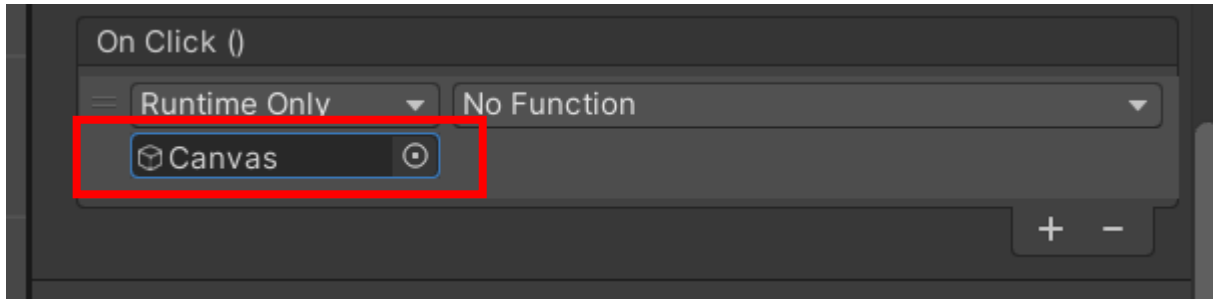
- To ensure that the game is frozen when it is paused and does not take any user input (i.e., key or mouse input) we will add a simple condition to the player controller script.
- This if-condition uses the static Boolean variable created in the PauseResume script to check if the game is paused and thus we won't allow user input.
- Note that this condition encompasses all movement of the player.

```
void Update () {

    if (!PauseResume.paused)
    {
        if (Input.GetKeyDown(Spacebar) && grounded) //When user presses the space button ONCE
        {
            Jump(); //see function definition below
        }
```

Exercise #38 - Pausing and Resuming (Cont.) – Resume Button

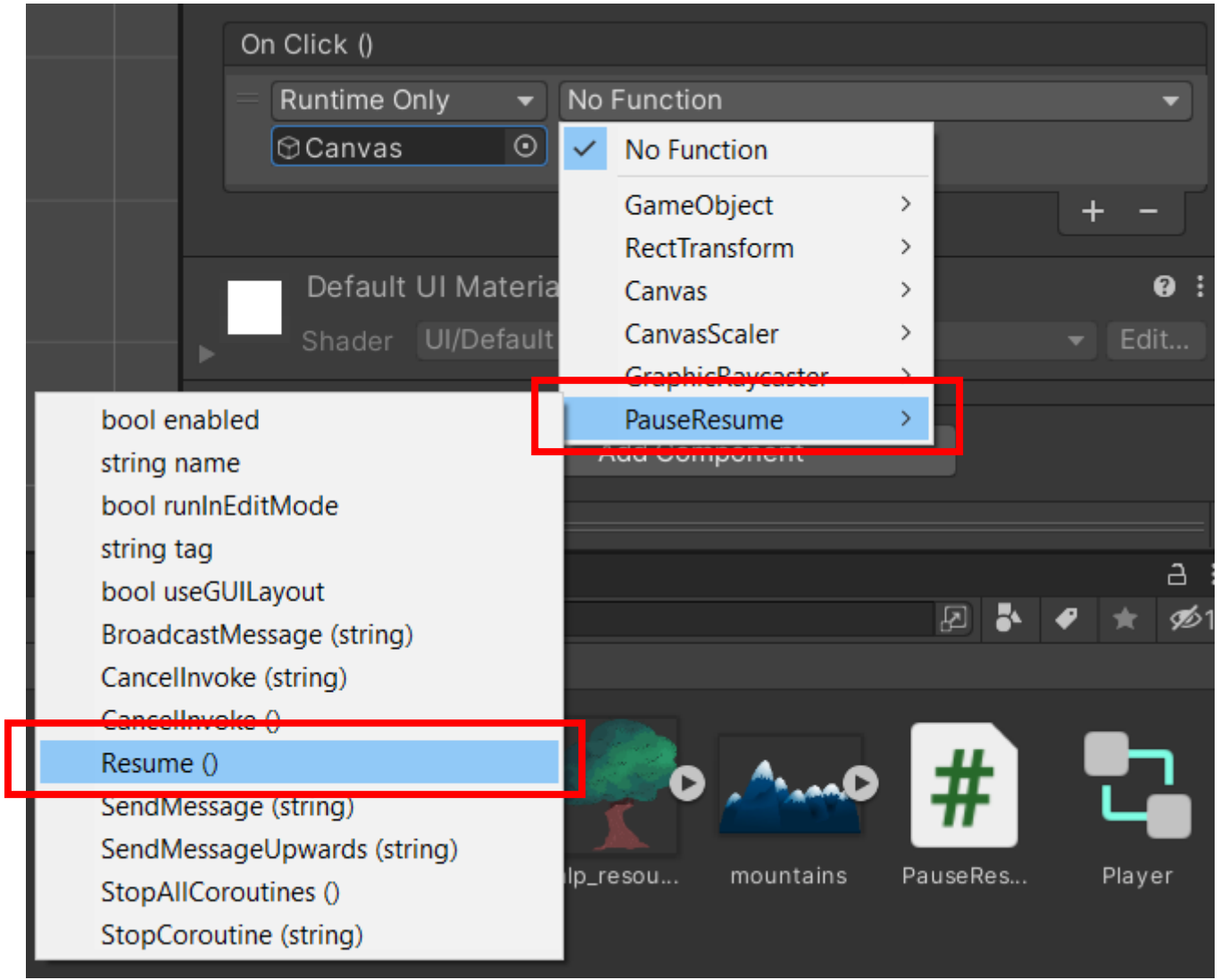
- Now we can pause the game by clicking on the pause button and resume once we press of the button once more
- The last step is to add functionality to the resume button we created in the panel so we can also resume the game by clicking on the button.
- Go to the resume button in the inspector and go to the button component
- Go to the On Click property and add to the list
- Then add the Canvas object





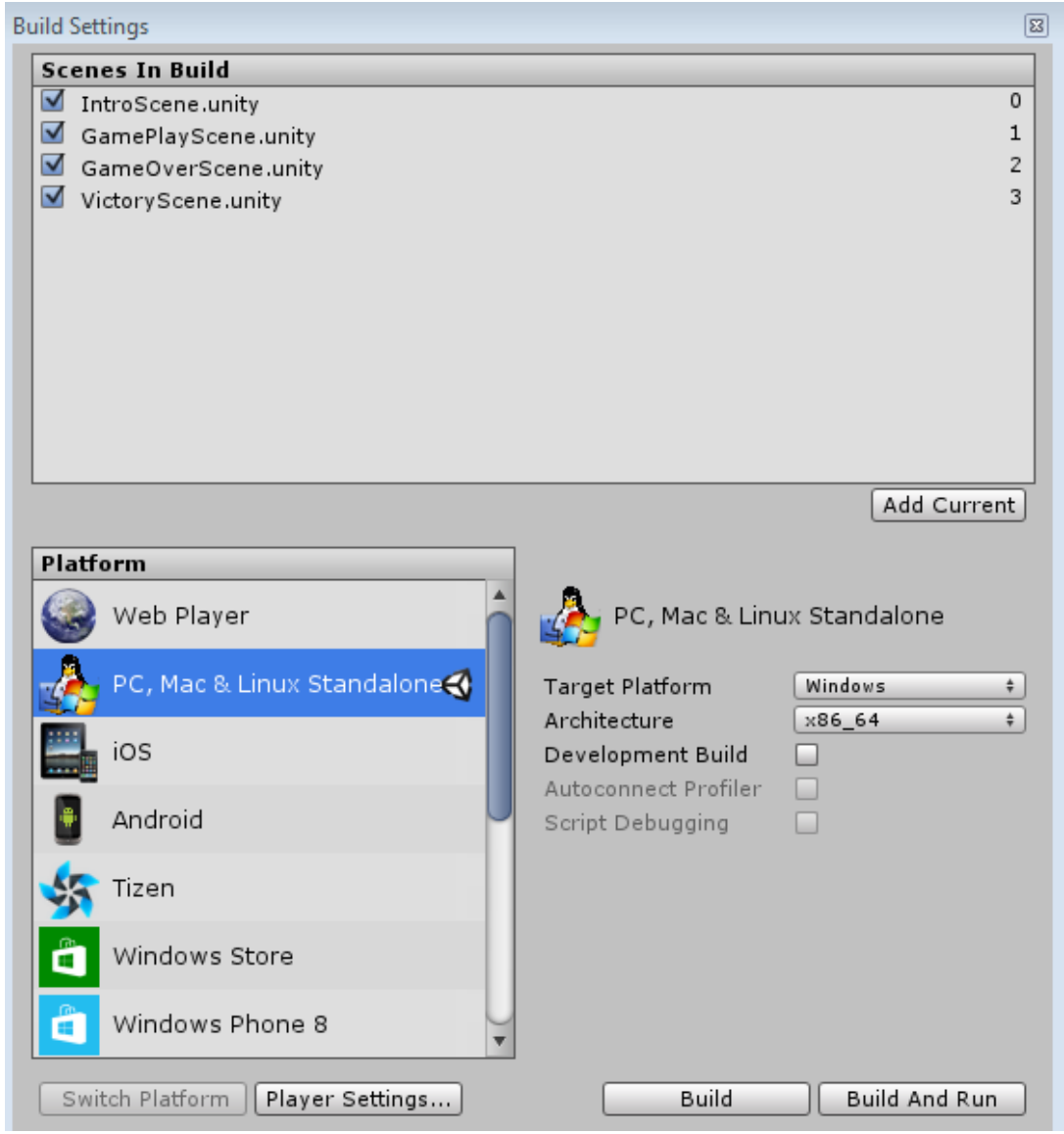
Exercise #38 - Pausing and Resuming (Cont.) – Resume Button

- Then from the drop-down menu select the PauseResume script then Resume function.
- So, when we click on the resume button it calls the resume function from the script.

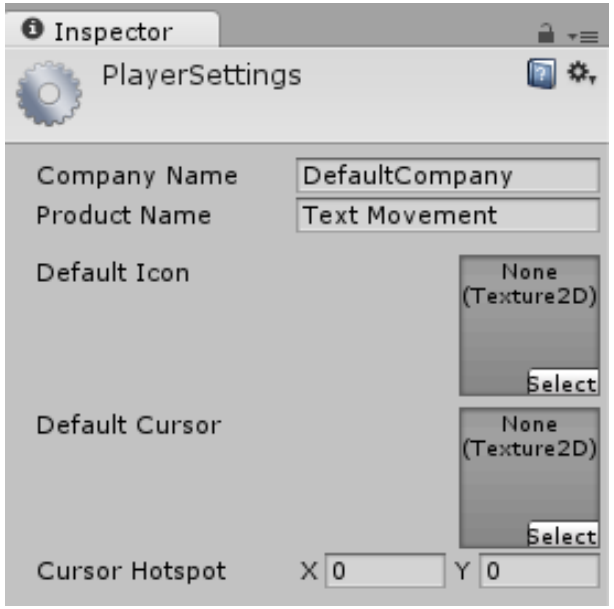


Export Your Game

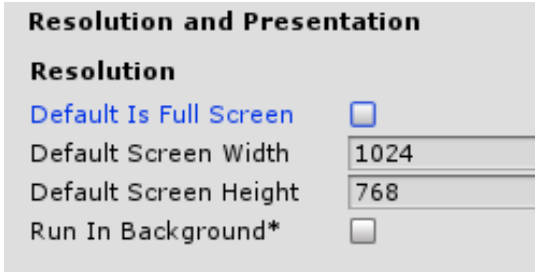
- In the previous exercises, you have already listed all your scenes in Build Settings
- Choose the *Target platform* to match your laptop
- Choose the correct *architecture* to match your laptop
- Then click on *Player Settings*



Export Your Game (Cont.)



- In the Inspector window, change the company name and product name if you wish. You can also add an icon so that your game is distinctly recognizable on your desktop
- In Resolution and Presentation, you do not need a full screen at the
- current time, so uncheck *Default is Full Screen*



- Now go back to the Build Settings window and click Build. Choose your desktop as the destination folder and name your export
- Double click the file on your desktop and choose the ‘Good’ quality from the drop-down list before playing

Useful References:

- **Exporting your 2D Game (Brackeys Tutorial). URL retrieved from:** https://www.youtube.com/watch?v=jxmlcL67ilg
- **Scenes. URL Retrieved from:** https://www.youtube.com/watch?v=sGIN3sWT9Pc
- **UI Buttons. URL Retrieved from:** https://www.youtube.com/watch?v=BRbgNtFXujg
- **Pause/Resume. URL retrieved From:** https://www.youtube.com/watch?v=3pK8nW4A_S4