Γραφικά Υπολογιστών

Ιόνιο Πανεπιστήμιο
Τμήμα Πληροφορικής

Στέργιος Παλαμάς, Επίκουρος Καθηγητής

Μάθημα 7:

User Interfaces στο Unity

Get started with user interfaces

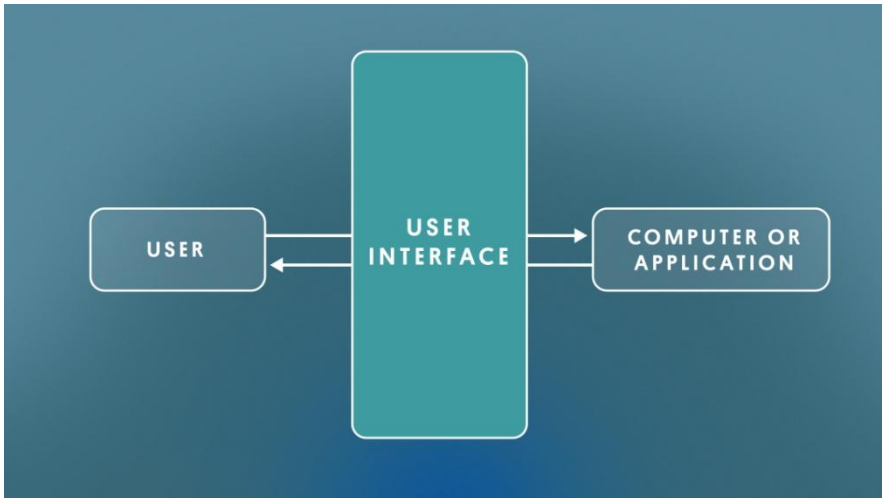
A user interface (UI) is what allows a user to interact with a program and a UI designer is responsible for making those interactions as clear and enjoyable as possible. In this tutorial, you'll learn a bit more about what UI design is and the tools available in Unity to help you create UIs. Then, you'll open your project and begin customizing your scene.

Το Tutorial στο Unity-Learn

Υλικό που θα χρειαστεί στο Project
CC_UI.zip

Το κατεβάζετε και το αποσυμπιέζετε σε έναν φάκελο του υπολογιστή σας

A **User interface** (**UI**) is any system that allows a user to interact (interface) with a computer or computer application.



In the context of real-time 3D projects, a UI is usually a combination of text, buttons, checkboxes, sliders, and toggles.

These elements can communicate with the user using rules or notifications. UI can also help the user carry out tasks, such as saving progress or adjusting settings.

Every published application you've ever used has likely included some kind of UI.

In this tutorial and the ones that follow, you will learn how to use Unity's UI systems to create an interactive settings screen.

3.What does a UI designer do?

Responsibilities

UI designers mostly concern themselves with what the UI elements look like and where they're located on screen. More specifically, a UI designer will:

- Design menus, buttons, HUDs, pop-ups, etc. according to the art direction.
- Make sure all relevant information and functionality is present through the various UIs.
- Design with technical constraints and optimization guidelines in mind.

If the UI designer is comfortable with basic programming, they may also be asked to implement some simple UI functionality. More complex UI functionality, such as a character customization screen, would fall to a programmer.

Background

Most UI artists have the following foundational skills and experience:

- Traditional artistic skills, including knowledge of proportion, composition, color, etc.
- Good understanding of usability, clarity, and accessibility of interfaces.
- Up-to-date with the current design and typographical trends.
- Basic programming is a plus, since UI is usually tied to simple functionality.

Tools

In addition to Unity, UI designers might use or be familiar with other software programs, including:

- 2D image editing software, such as Adobe Photoshop, to create textures and lay out interfaces.
- 2D vector illustration tools, such as Adobe Illustrator, to draw images, buttons, and icons.

What is UI design?

UI design is the process of laying out **graphical UI** (**GUI**) elements in a compelling, yet intuitive way. UI should be clear, consistent, accessible, and appealing. **When you're using a well-designed UI, the experience is seamless and enjoyable.**

If you're not thinking about the UI design very much as you're using it, that's usually a good sign.

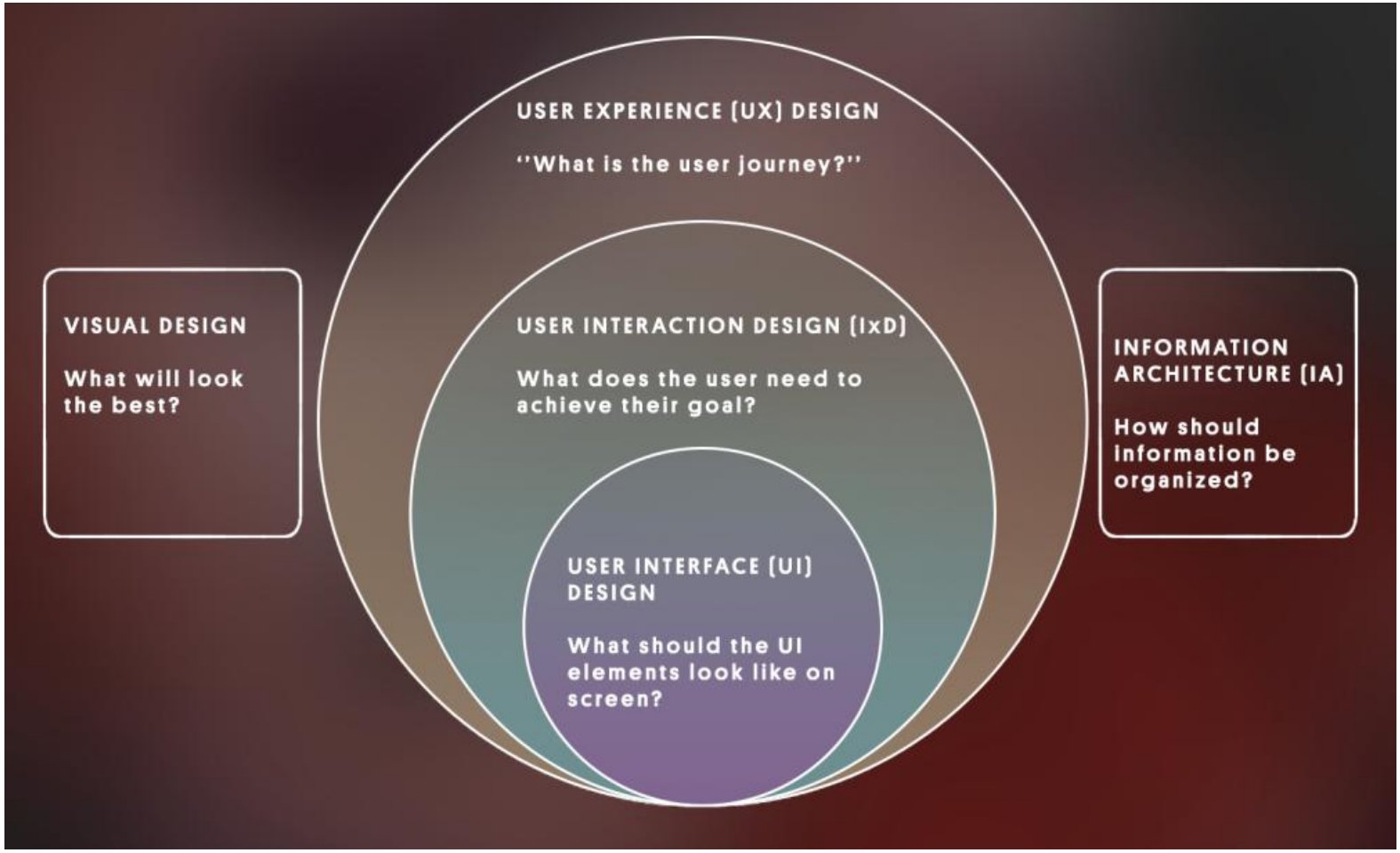
In fact, people say that a **good UI should be invisible and get out of the user's way.**

Check out the example below from Last Day on Earth, which was made with Unity. Notice how the UI does not take attention away from the gameplay, which should be the focus.

What UI design is not

However, **defining UI design can get confusing when you consider it alongside similar-sounding areas like user experience and user interaction design.**

Below are some of those disciplines and how they relate to each other:



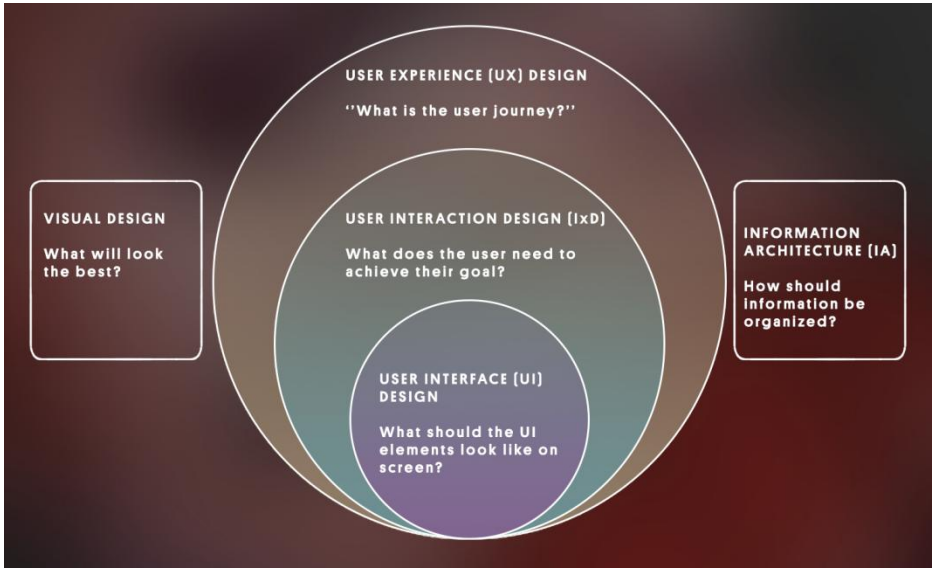
User experience (UX) design
Answers the question: What's the user journey?
Goal: Ensure a positive overall experience with a product from start to finish.

User interaction design (IxD)
Answers the question: What does the user need to achieve their goal?
Goal: Allow users to accomplish their goals in the most efficient way.
Is part of UX design.

User interface (UI) design
Answers the question: What should the UI elements look like on screen?
Goal: Make the interface easy to understand and easy to use without getting in the way.
Is part of IxD, which is part of UX.

Information architecture
Answers the question: How should information be organized and structured (not necessarily for interactions)?
Goal: Structure content so users know where to go to find the information they need.
Informs UI design, IxD and UX design.

Visual design
Answers the question: What will look the best (not necessarily for interactions)?
Goal: Make something visually appealing.
Informs UI design, IxD and UX design.



Who does what?

The screenshot below shows the inventory menu from an action RPG, which allows the player to equip items, use items from the inventory, and view current player stats.

User experience: Ensure the experience of equipping items is fun and satisfying for the user.

Interaction design: Ensure the user can equip or use items in two clicks or fewer.

User interface: Design the icons and interface layout so that it is very clear what everything is.

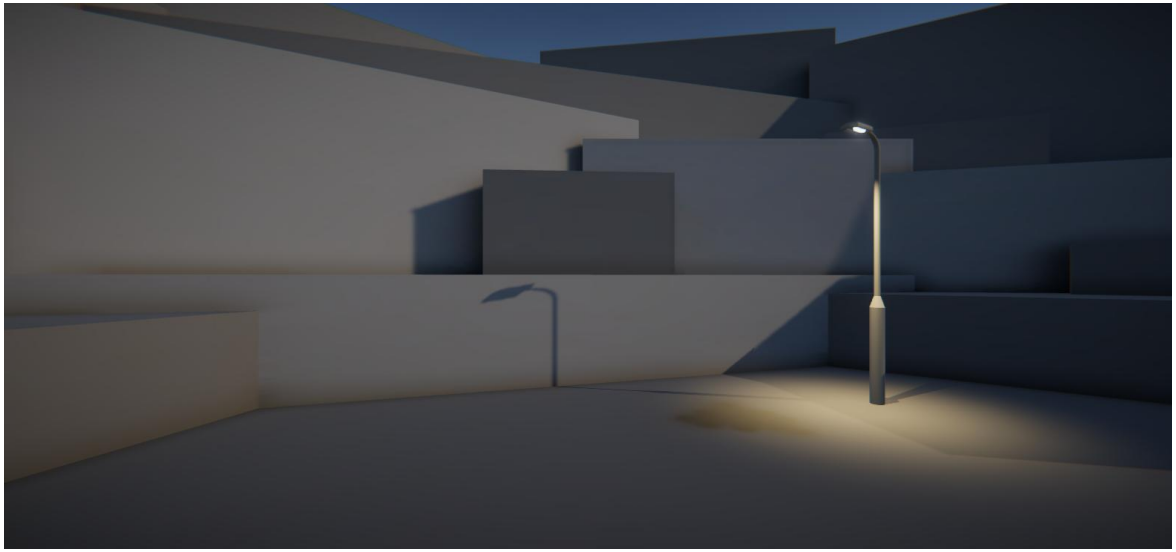
Information architecture: Determine if it makes sense to split up items that can be equipped vs items that cannot.

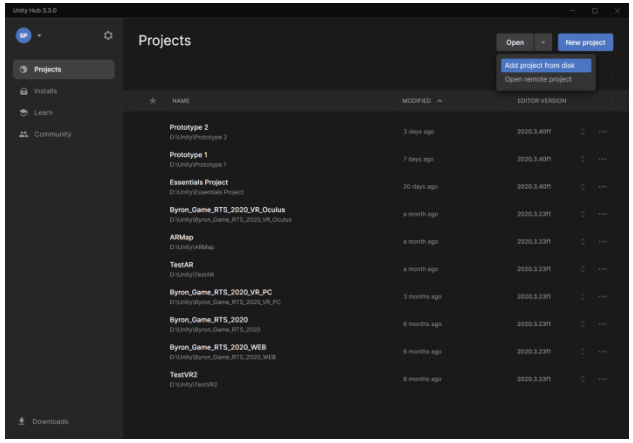
Visual design: Select the best possible colors for the background and text.

6.Open the UI project and scene

To set up your Unity project:

1. Install **Unity 2020.3 LTS**, if you haven't already done so.
2. Download the project for this learning experience.
3. Identify a suitable location on your computer and unzip the project folder there. Remove the Unity project folder from its empty parent folder.
4. Add the Unity project to the Unity Hub.
5. Open the project from the Unity Hub.
6. In the Unity Editor's Project window, open **Assets** > **CreativeCore_UI** > **Scenes** and then open the scene **TutorialScene_UI_Outdoor**.
In the **TutorialScene_UI_Outdoor** scene, you will find an outdoor scene with a single street lamp in it.



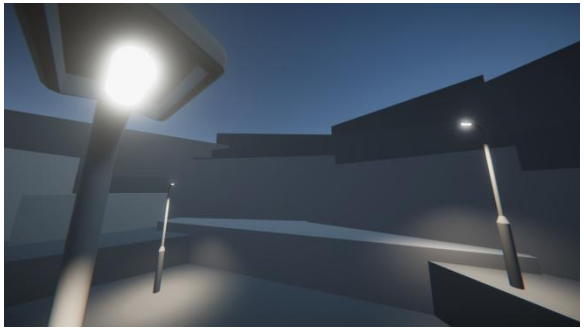


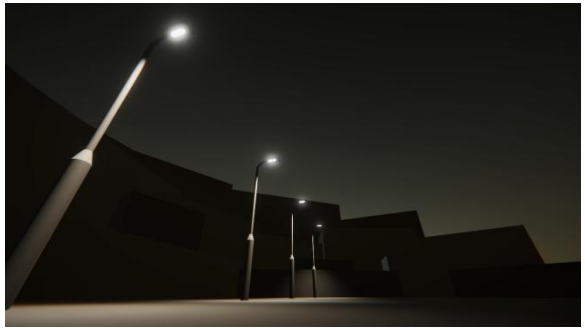
7.Personalize the backdrop

Press **Play**. What you see in Game view will be the backdrop of your UI. The title and menus will sit on top of what you see there.

If you don't really care what your final project looks like, you can skip this step and just use what is currently framed in the Game view.

However, if you want to personalize your UI scene, take some time now to make it look how you want. With just a few quick changes to the scene and your camera position, it could look very different.





1. From the Project window, in **Assets** > **CreativeCore_UI** > **Prefabss**, add additional props to your scene that you want to include in your UI backdrop. You can also use primitive shapes like cubes and spheres to build your own custom objects. If you're familiar with Probuilder, you can use that too!

2. Adjust the **Rotation**, **Color**, and **Intensity** of the **Directional Light** to change the time of day, as you learned in the **Lighting** tutorials.

3. Reposition the **Main Camera** to frame the objects nicely in your scene. Use the **Field of View** property to expand or restrict what's in frame, as you learned in the **Camera** tutorials.

Once you have your scene framed just right, you're ready to move on!

Add a title to your scene

Text is arguably the most critical element of any UI. So, when you add text elements, you should also make sure it's easy for everyone to read. In this tutorial, you'll add title text to your project, then make sure it meets basic accessibility requirements.

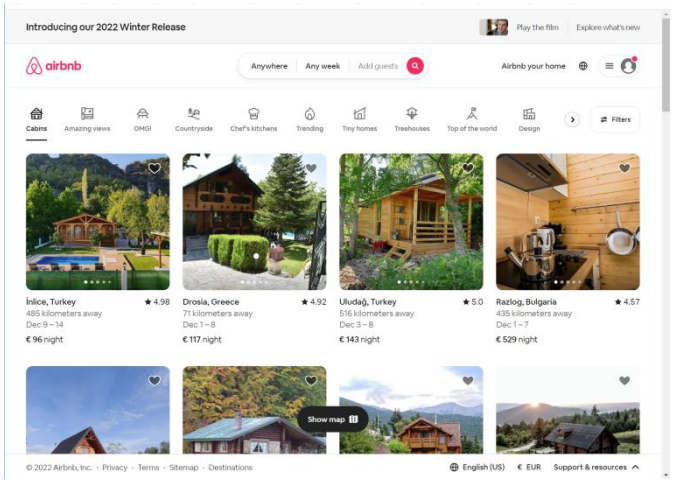
1.Overview

Text is at the core of UI design. It is a primary way to communicate information to users. If designed well, text is incredibly helpful and clear.

For an example of clean, consistent UI, check out Airbnb.com.

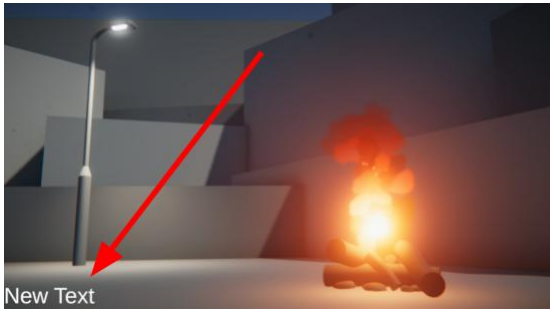
If designed badly, text can be ugly or overwhelming.

For an example of cluttered, confusing UI, check out Craigslist.com.
In this tutorial, you will add a title to your scene and make sure that it looks great.

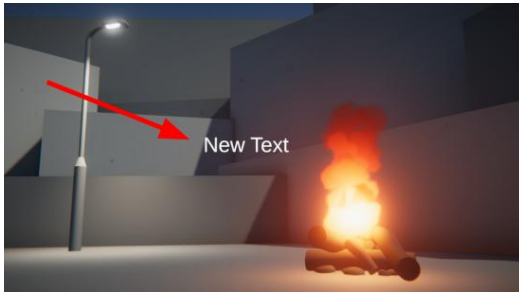


2.Add and center your title text

Let's add your project's first UI element: the title.



1. In the Hierarchy, **right-click** > **UI** > **Text - TextMeshPro**. If a TextMeshPro Importer window pops up, select **Import TMP Essentials**. TextMeshPro (TMP) is a robust, easy-to-use text editing tool that is also optimized to display clearly on different screens. The words New Text should now appear in the Game View screen in a pretty small font. If your text does not appear in exactly the same location or is a different size, that's OK.



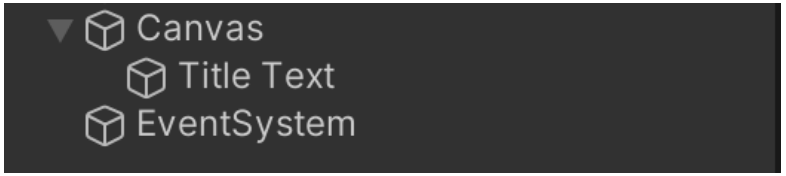
2. With the **Text** object selected in the Hierarchy, Locate the **Pos X** and **Pos Y** properties in the **Rect Transform** component. The Rect Transform component controls the position and scale of 2D rectangles.

3. Set **Pos X** to 0 and **Pos Y** to 0 to center the text on the screen. If your text is right on top of another object, making it difficult to see, adjust the **Pos X** and **Pos Y** to fix that.

4. In the Hierarchy, rename the **Text** (**TMP**) GameObject to "Title text".

You will also notice that a couple of new GameObjects were also automatically added to the Hierarchy the moment you created that text object.

There's a **Canvas** object, which is a required parent object for any UI elements, and an **EventSystem** object, which manages UI interactions like button or toggle selections.



3.Customize your title text

1. With the Text object still selected, locate the **TextMeshPro - Text (UI)** component. You'll use this component to customize the appearance of your text.

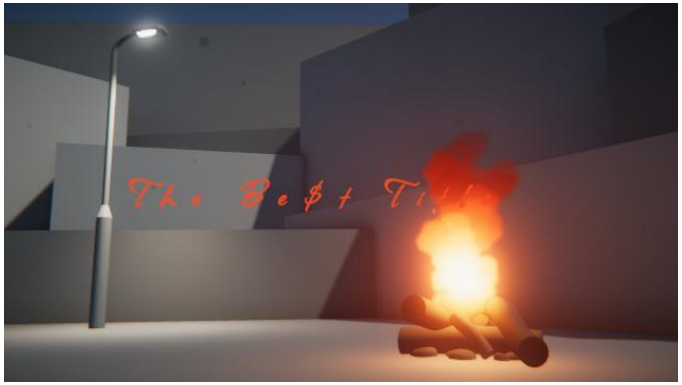
2. In the **Text Input** field, enter a new title for your project. It can be whatever you want. If it's much longer than New Text, it will probably wrap onto multiple lines – we'll fix that next.

3. Change the **Wrapping** property to **Disabled** to allow your title to stay on one line.

4.Make sure your text is accessible

What is text accessibility?

Before you become too attached to your title, let's make sure it meets some basic **accessibility** requirements.



Accessibility refers to how well people with different physical abilities can successfully interact with your application. In the case of text, that mostly relates to visual accessibility.

Here are are a few specific things to consider:

- **Text size:** text should be large enough to read comfortably.
- **Text contrast:** text should stand out sufficiently from the background.
- **Text font / typeface:** text font should be easily legible.
- **Text content:** text should use familiar words written in a way that can be read by a screen reader.

Manage screen size and anchors

Summary

You can spend a ton of time making your UI look perfect on your screen, but what happens if someone opens your application on a screen with a different size or a different shape? In this tutorial, you'll learn how to consider the screen's aspect ratio and use Canvas Anchors to make sure your UI elements stay where you want them to.

1.Overview

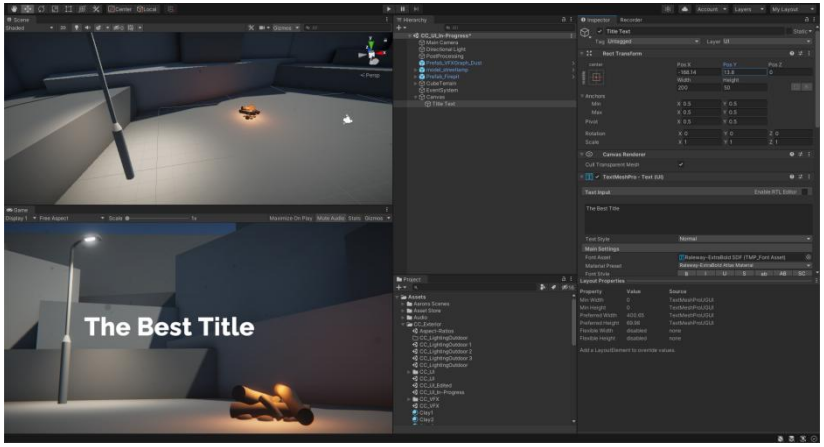
With your title, you explored some basic text formatting. But how will that text respond to changes in screen size or shape?

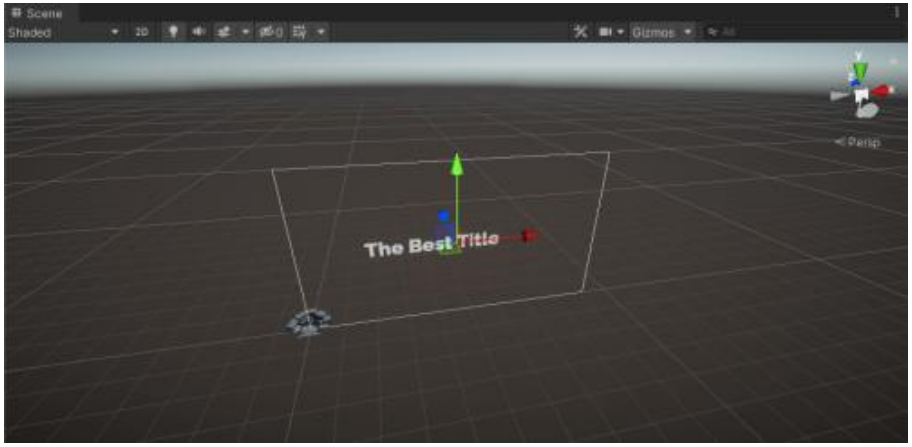
Luckily, there are lots of tools available in Unity to help you with this.
In this tutorial, you will use Unity's **Canvas** and **Anchor Point** system to make sure your UI elements always appear in the right place, regardless of the screen they're on.

2.Arrange the Editor for working with UI

When making edits to your UI, it's helpful to see both the Scene view and Game view at the same time.

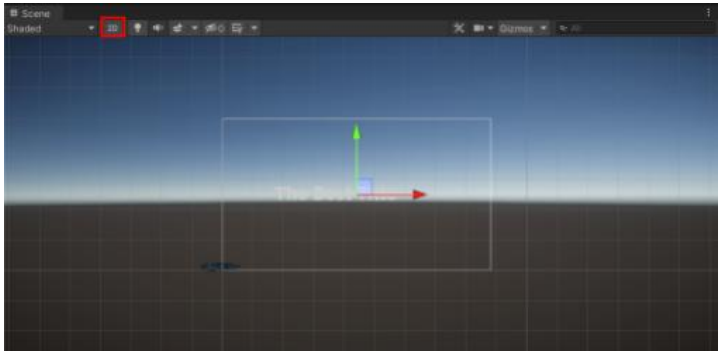
1. Rearrange the Unity Editor layout so that both the Scene view and Game view are on top of each other. From the **Layout** selector, the 2 by 3 option will work – or you can customize your own layout like the one shown below.





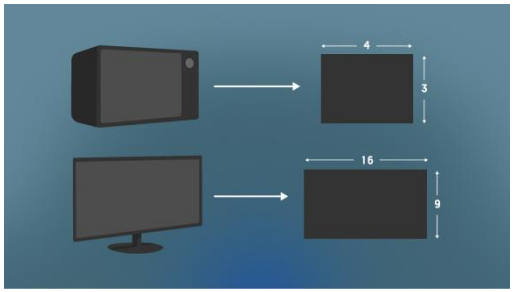
2. In the Hierarchy, double-click the **Canvas** GameObject to frame it in Scene view. Your perspective will zoom way out to reveal a gigantic 2D rectangle, your 3D scene now appearing tiny next to the bottom-left corner of the Canvas. If you don't see the rectangle's outline, you may have to enable **Gizmos** in the Scene view. This rectangular **Canvas** represents the screen that your final project will be viewed on – all UI elements must fall within the rectangle to be viewable on screen.

3. In the Scene view, enable **2D** mode. Since your Canvas is only two dimensional, 2D mode allows for the simplest editing, restricting movement to only the X and Y dimensions. To exit 2D mode, just select the **2D** button again.



3.Select an aspect ratio

What is an aspect ratio?

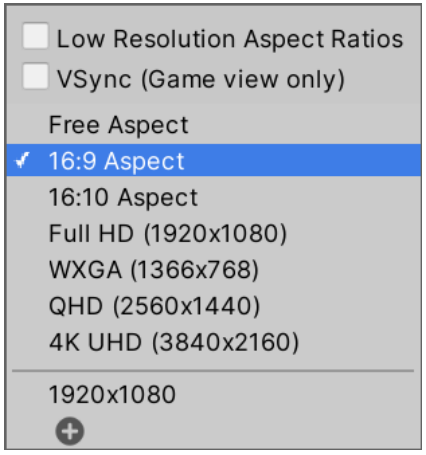


Now that you can actually see the outline of the screen (the Canvas), let's set the shape – or **aspect ratio** - of that screen.

The aspect ratio describes the relationship between the width and the height of a screen.

Old-fashioned TVs and the very first films, for example, used a 4:3 aspect ratio. That means that the screen was 4 units wide and 3 units high. However, modern widescreen TVs and most new films use a wider 16:9 aspect ratio.

Lock your project's aspect ratio
Unity allows you to choose which aspect ratio you want to use for your Canvas.
1. In the Game view, use the screen resolution dropdown to select **Free Aspect**. This means that the screen is free to change its aspect ratio**.**



With **Free Aspect** enabled, the Canvas will change in shape and size as you resize the Game View window. This can be helpful for testing if you're trying to make sure your scene looks good on screens of different shapes.

2. Move the edge of your Game view window to make it narrower and wider. Notice that the Canvas in the Scene view changes freely to reflect the size and shape of the Game View window.

You will also notice that your text likely moves off-screen as you do this – that is a good demonstration of how difficult it is to design experiences that are responsive to different screens. To keep things simple, we are going to stick with a single shape (or aspect ratio).

3. In the Game view, use the screen resolution dropdown to select either the **16:9** or **16:10** aspect ratio. 16:9 is the standard widescreen ratio, so it's the safest option. Designing for a single predetermined aspect ratio will make designing your UI a lot easier.
With a fixed aspect ratio, you'll now notice that changing the width or height of your Game view window will not affect the aspect ratio of the Canvas in the Scene view.

4.Add a settings button in the corner

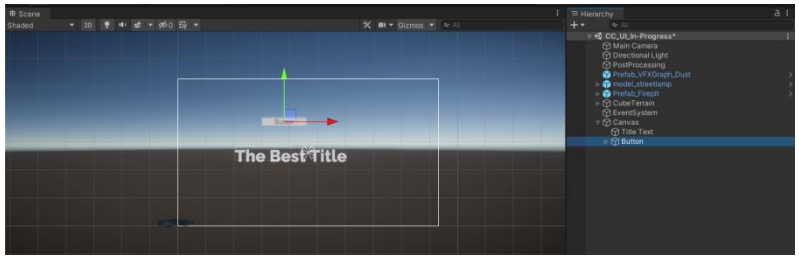
To be able to access the settings screen, you'll need a settings button.

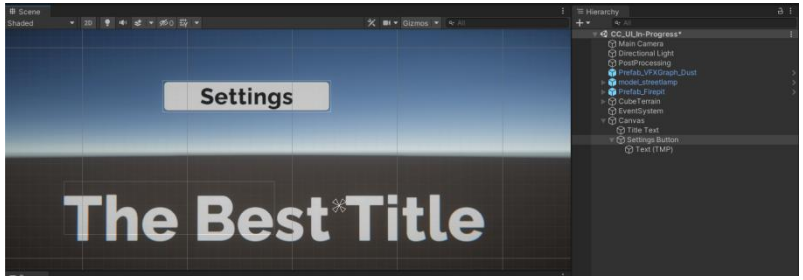
1. In the Hierarchy right-click > **UI** > **Button - TextMeshPro**. This will add a button somewhere on your UI as a child of the **Canvas** GameObject.

2. In the Hierarchy, rename the "Button" as "Settings Button".

3. Expand the **Settings Button** GameObject and select its **Text (TMP)** child GameObject. This is a Text Mesh Pro element just like your title. Change the text to say "Settings" and edit the style to match your title.

4. Use the **Move** tool to position the button in one of the corners of your Canvas, still leaving space for margins between the button and the edge of your screen. Choose whichever corner looks best in your scene.





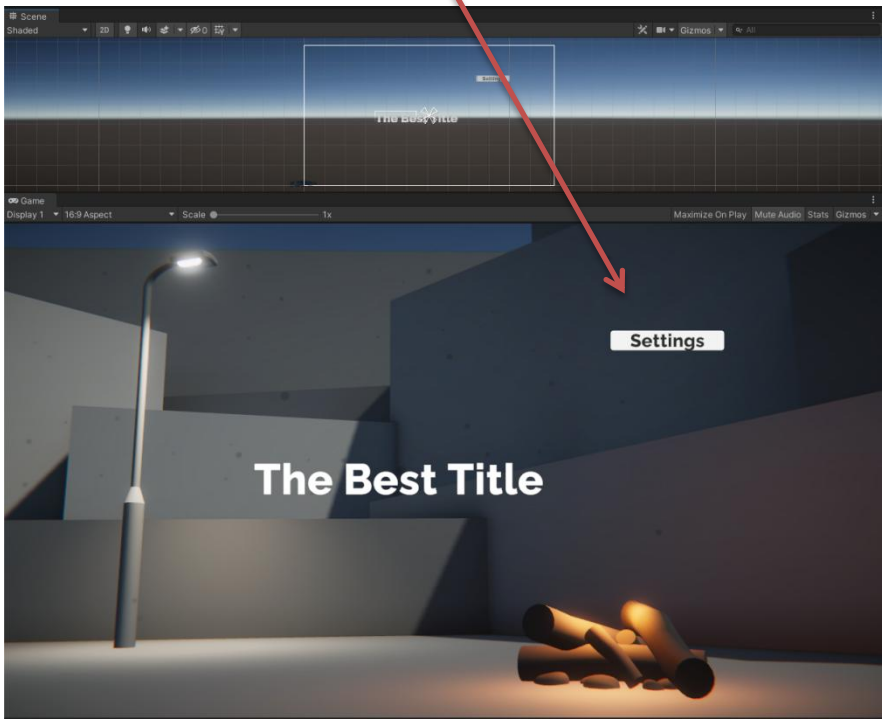
5.What are anchors?

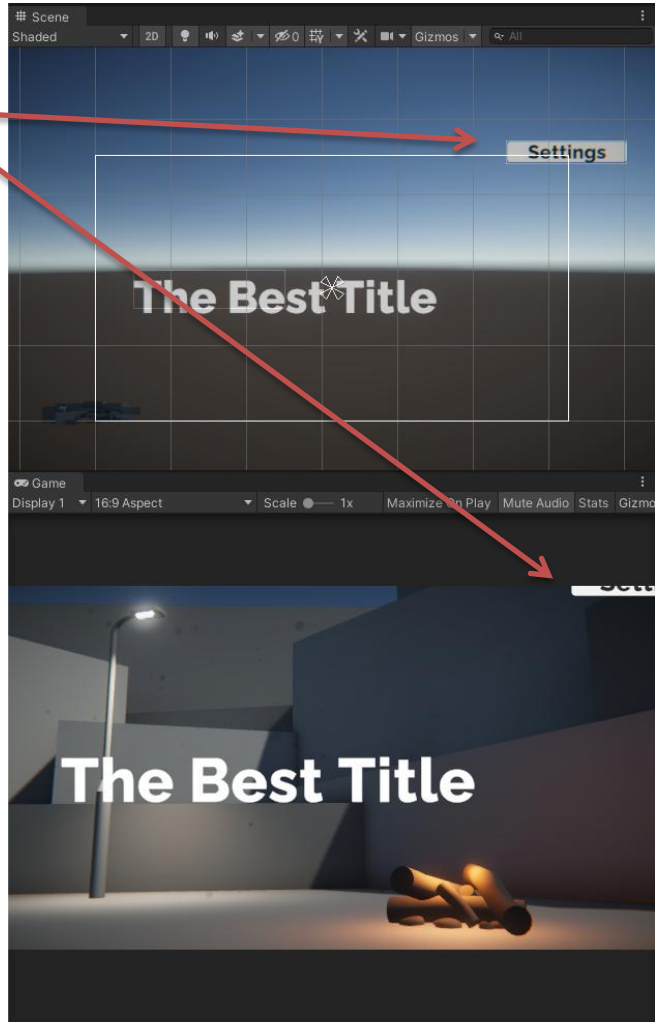
Why do you need anchors?

You carefully positioned your settings button in a desirable corner location. However, if you move the edge of the Game view window, increasing or decreasing the size of the screen, the button will drift towards undesirable locations.

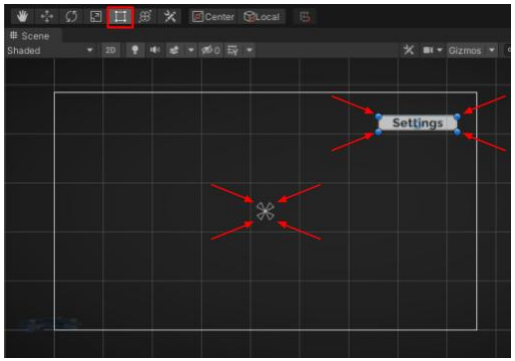
If the screen is too small, the settings button will drift off the screen.

If the screen is too large, the settings button will appear too close to the center.





This movement is controlled by the Canvas' anchor system. **Every UI element on the Canvas is anchored to a particular position on its parent object – and it will move relative to that position when the screen size changes.**



Visualize the anchors

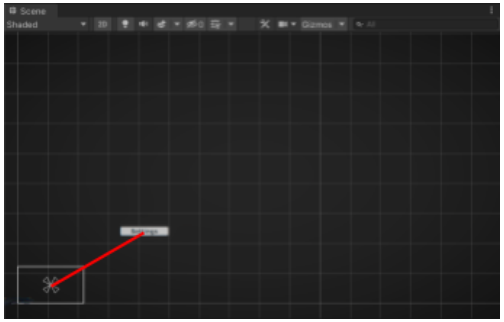
From the Scene view toolbar, select the **Rect** tool (or press **T on your** keyboard when in the Scene view) and then select your **Settings Button** object.

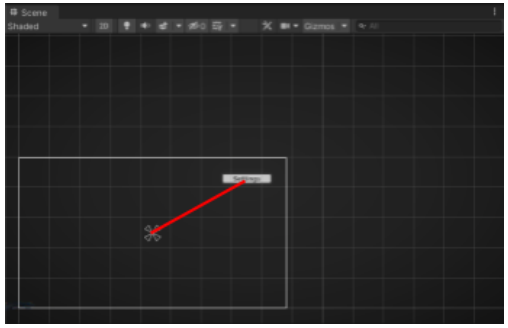
With the Rect tool selected, you will be able to see the four corners of your button's Rect Transform, represented by blue circles.

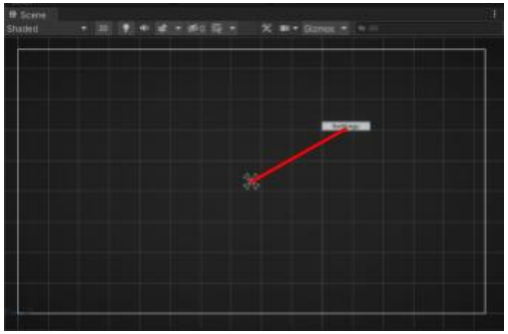
You will also see the anchor in the middle of the screen, represented by four small triangles pointing inward.

Note: in the image above, the title and skybox have been disabled to help you see the anchors more clearly.

The button will always stay the same distance from that anchor; it doesn't matter how big or small the screen is. Test this for yourself by resizing your Game view window.
Here's the position of the settings button relative to the center of the screen when the Game view is very small. The button would be off-screen:



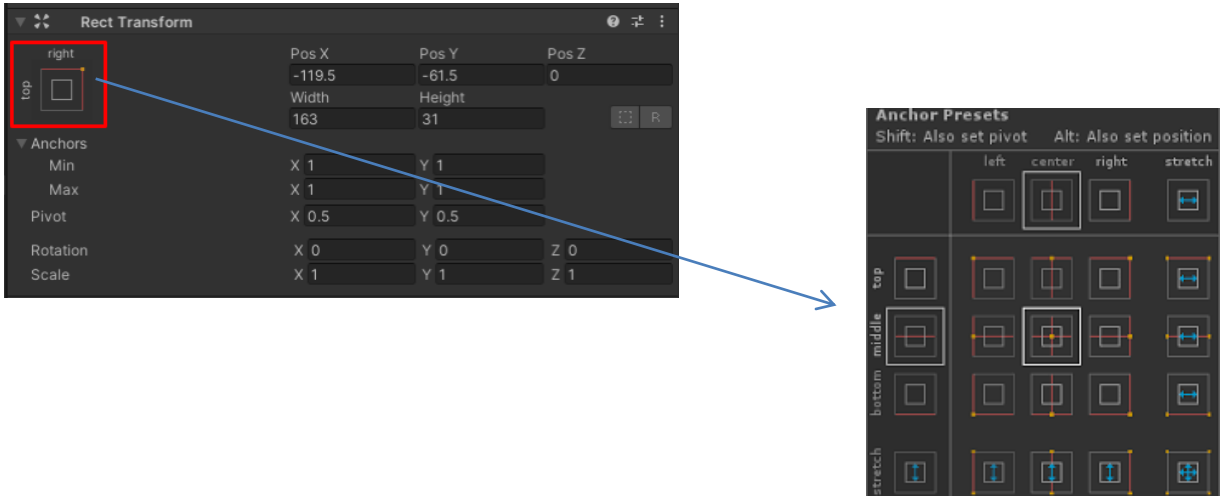




6.Set the anchor for the settings button

To save you the time and effort of moving anchors around and to help you be more precise, Unity has anchor presets that you can use.

3. With the **Settings Button** object still selected, in the Rect Transform component, select the **Anchor Presets** icon. This will bring up the **Anchor Presets**:



4. Select the various anchor presets and notice how the position of the anchor changes in the Scene view. Experiment by resizing the Game view window with different anchor presets, trying to find the one that works best for you.

8.Explore: Experiment with canvas scaling

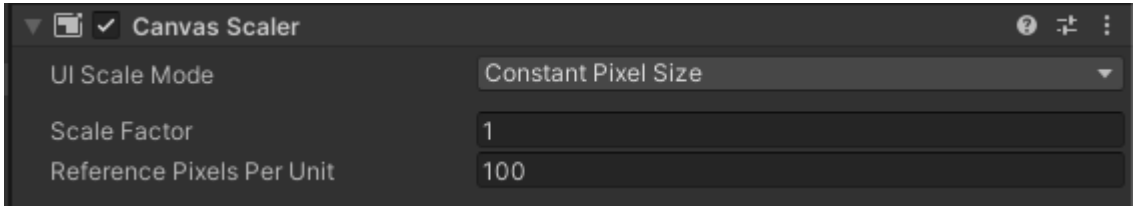
As you were designing your UI, you may have noticed that as you scaled the size of the Game view resolution up and down, the UI took up proportionally different amounts of the screen.

At higher resolutions, the UI looks proportionally smaller, but at lower resolutions, the UI looks proportionally bigger:

UI elements at a high resolution	UI elements at a low resolution
	

This has to do with the **Canvas Scaler**: a component that controls scale and resolution of your UI elements at different screen sizes.

1. Select the Canvas object and locate the Canvas Scaler component. Notice that the **UI Scale Mode** property is set to Constant Pixel Size by default. **With a constant pixel size, your UI elements will take up the same number of pixels, regardless of the overall size of the screen. This might be helpful if you expect someone to experience your scene on a very low resolution device – the text will likely still be readable**



2. Set the **UI Scale Mode** to **Scale with Screen Size**. This mode will scale UI elements proportionally to the screen resolution. The elements will take up the same proportion of the screen, regardless of resolution. By default, the Canvas Scaler will match the width of the screen, using 800x600 as the reference resolution

UI elements at a high resolution	UI elements at a low resolution
	

This may have increased or decreased the relative size of your UI in the Game view.

3. Increase or decrease the **X** (width) of the reference resolution to get your UI elements to a desirable relative size on your screen.

There are advantages to each type of UI scale mode:

- **Scale with Screen Size** will guarantee that all UI elements are positioned in the same relative position, regardless of screen size.
- **Constant Pixel Size** will make sure that UI elements are always the same number of pixels.
- **Constant Physical Size** will try to make UI elements take up the same physical size by also taking into account the device's dots per inch (DPI).

4. Select either Scale with Screen Size or Constant Pixel Size for your UI Scale Mode.

Create a menu background with images

Summary
Images are critical in the design of UIs, from simple backgrounds and icons to more complex heads up displays and dashboards. In this tutorial, you will add custom images for your settings menu background and button, making sure they still look good if they're stretched in different directions.

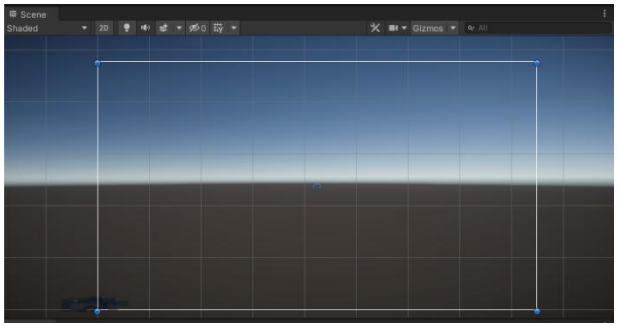
1.Overview

In the previous tutorials, you worked with text and button elements. Another major element in UI design is **Images**. Images can be used for backgrounds, icons, badges, and more.

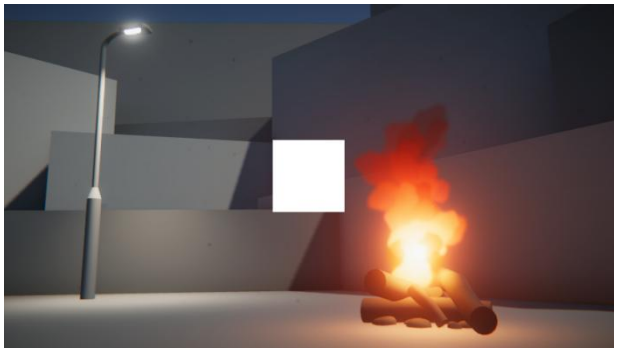
In this tutorial, you will add a new background image for your settings screen, replace your default button images with custom ones, and use a technique called 9-slicing to make sure the images scale appropriately.

2.Add a basic settings background

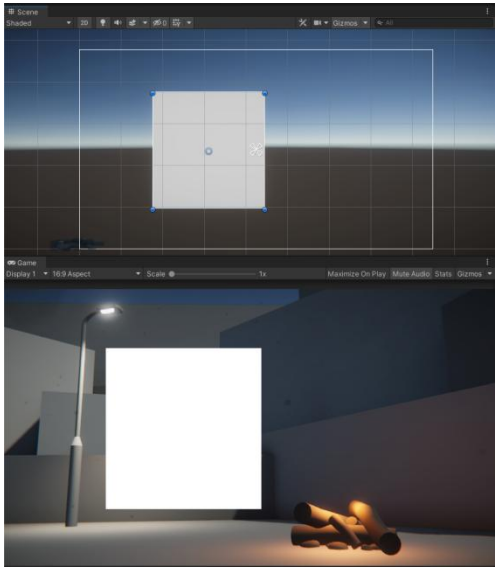
The settings menu elements will sit on top of a simple rectangle. To create that rectangle, you'll need to add an Image object.



1. First, clear your Canvas by deactivating the **Title Text** and **Settings Button** GameObjects. This will give you a blank slate to work from.



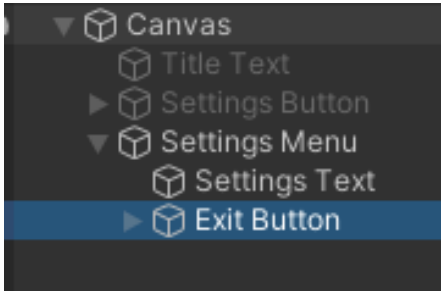
2. In the Hierarchy, **right-click** > **UI** > **Image**, then rename it "Settings Menu". An **Image** is a very simple UI element that can display any image you have imported into your project. By default, this will add a white square in the center of your scene.



3. In Scene view, use the **Rect** Tool to reposition and resize the image so that it's appropriate for a settings menu.

4. In the Image component, select the use the color picker to change the color of the background. From the Color window, you can also use the Alpha property (**A**) to make the background semi-transparent.

4.Create a simple exit button

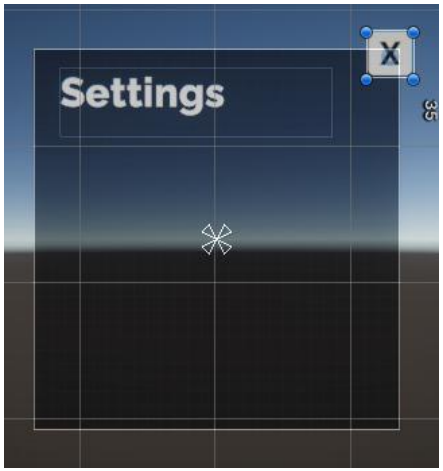


You have a settings button to take you to the settings screen. Now you need an exit button to get back to the title screen. As you create this button, we'll look more in depth at the ways that you can customize the design of UI elements using the Image component.

1. In the Hierarchy, right-click on the **Settings Menu** GameObject and select **UI** > **Button - TextMeshPro**. This will create a new Button child object, which you can rename "Exit Button".

By default, your button will appear with the word Button on it, using the default font.

2. Expand the Exit Button object and select the child Text object. Change the text to say "X" in your desired font. The "X" will serve as a symbol for close or exit.



Of course, the exit button is not in the ideal shape or location yet.

3. Use either the Rect tool in Scene view or the properties in the Rect Transform component to make the exit button a perfect square in the upper-right corner of the UI background.

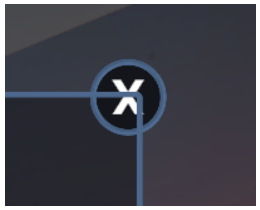
You are free to place this exit button anywhere you like, but keep in mind that users are accustomed to seeing this kind of button in the upper-right or upper-left corner – and one of the goals of UI design is to make it intuitive.

In the Image component, use the object picker button to browse through some other UI button images we've provided for you. Choose the one that you think looks best for your settings menu.

6.Explore: Continue customizing your UI

You know how to add and swap images in the UI. Apply these skills to make your UI elements look even better. You could have the settings button use a custom image instead of the default one, or you could add borders around the background to make it stand out more.

As you layer more details onto your UI elements, it's important to understand the **Draw Order** of the Canvas. Objects are drawn onto the screen in the order they are listed in the Hierarchy. This means that objects lower down in the Hierarchy will be drawn on top of objects that are higher up.
If objects are not ordered properly in the Hierarchy, you may end up with overlapping objects, like the one shown below.



To fix this, you just need to put that exit button below the background border in the Hierarchy.

Add basic button functionality

Summary

A button is the simplest and most common interactive UI element. Without buttons, you couldn't get very far. In this tutorial, you will make your buttons functional using Unity's Event System.

1.Overview

As you learned in a previous tutorial, a user interface is something that allows a user to interact (interface) with a computer or computer application. In many cases, those interactions involve clicking buttons, toggling checkboxes, or dragging sliders. You interact with these types of UIs all the time. The keyboard on your phone is made entirely of buttons. Search filters for flights or restaurants use checkboxes and toggles. Computer volume and screen brightness settings are usually sliders. In this tutorial, you will add the first bit of interactivity to your project, allowing the user to get to the settings menu and then back to the title screen.

2.Analyze some familiar UIs

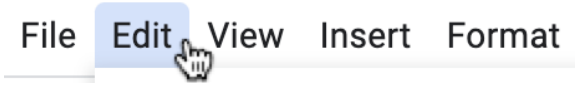
A well-designed UI clearly communicates with the user. The user should know what they can and cannot interact with on the screen. If they do try to interact with something, they should know whether or not they were successful. In other words, the UI should provide the user with lots of feedback.

Check this out for yourself in Unity or in any other application on your computer:

- Hover your mouse over a button. What happens?
- Select a button. What happens?
- When an object is selected, how do you know?

Check your work

As you hover over interactable objects, they tend to become highlighted in a different color – often a shade of gray. The text might also change color instead of the background. This effect is often pretty subtle. If the button is already a light color, it usually gets darker. When you actually click or select a button, it tends to change color again, usually in a more noticeable way. This often involves color rather than just shades of gray.



3.Edit your button transition colors

Unity allows you to control the color feedback provided by your buttons to better communicate with the user.

1. With the **Settings Button** GameObject selected, locate the **Transition** property in the Button component. This property should be set by default to **Color Tint**.

We will use **Color Tint**, which can change the color of the button when the user interacts with it. But, it is possible to swap out the image on interactions using the **Sprite Swap** setting or play custom button animations with the **Animation** setting.
Below are the default tint colors:

The **Normal Color** is pure white. This means the button won't have any color tint in its normal state.
The **Highlighted Color** is an extremely subtle shade of grey, just a bit darker than white. This appears when the user hovers over the button.
The **Pressed Color** is a more noticeable gray color.
The **Selected Color** is not relevant in this use case, since the entire button will disappear after being clicked.
The **Disabled Color** is not relevant either, since the button will always be enabled.
The **Color Multiplier** property will increase the effect that the color tint has on your button. If you have a darker button or a semi-transparent one, this may be helpful.

2. Run your application and see the effect of the tint colors on your button. During playmode, experiment with different colors to see what they look like.

3. Edit the Highlight Color, Pressed Color, and Color Multiplier properties for the settings and exit button in your application. The pressed tint should probably be more noticeable than the highlight tint.

4.Add an action to the On Click event

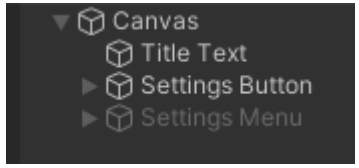
With the button now looking the way you want, let's make it functional. The users need a way to navigate between the main title screen and the settings menu.

When the user clicks the settings button, the settings menu should appear.

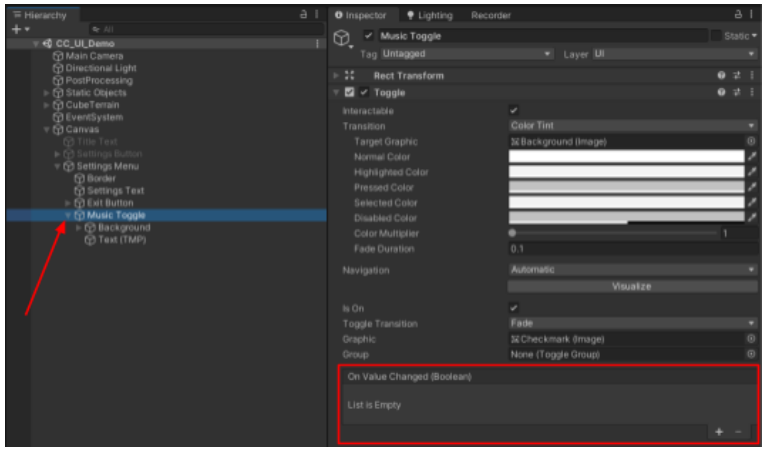
When the user clicks the exit button of the settings menu, they should be brought back to the title screen.

Let's start by making the settings button bring up the settings menu.

1. In the Hierarchy, activate the **Title Text** and **Settings Button** GameObjects, then deactivate the **Settings Menu** GameObject. You can do this using the checkbox next to the GameObject's name in the Inspector.



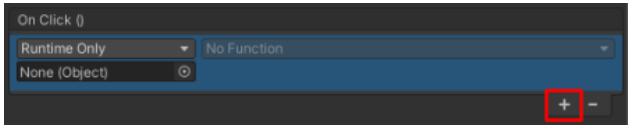
2. Select the **Settings Button** and locate the **On Click ()** section within the **Button** component.



Inside the **On Click** box, you will notice that it says List is Empty. This means that when the button is selected (on click), nothing will happen.

On Click is what's known as a **UnityEvent**. UnityEvents can trigger any number of actions if that specific event occurs. In this case, the "Event" is someone clicking on the button.

3. Select the **+** button to add a new action to the list within the **On Click** event.

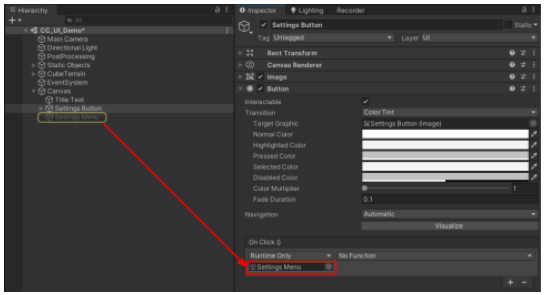


By default, the action you add is empty. It won't do anything yet. We'll make it do something next.

5.Make the settings menu appear on button click

Selecting a function for a UnityEvent is a two step process:

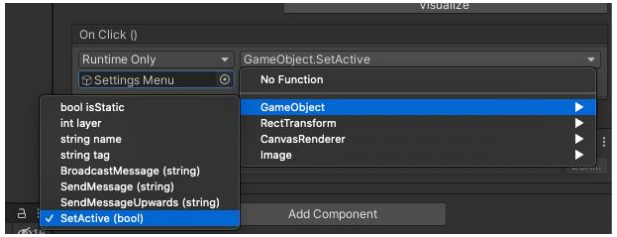
- •Select which object you want to run the function.
- •Select the function you want to run from that object's components.



By default, there is nothing assigned to the **Object** field (where it says None (Object)). Since you want the Settings Menu to appear, that's the object you want assigned to that field.

1. Click and drag the **Settings Menu** GameObject from the Hierarchy onto the empty Object field to assign it.

You'll notice that the **No Function** dropdown has now become available. No Function means that no function – or action – has been selected to run from this object yet. You now have to select which function you want the Settings Menu object to run when this event occurs.



2. From the dropdown that is currently set to No Function, select **GameObject** > **SetActive (bool)**. When you navigate this menu, you are browsing through the scripts assigned to this object and then through the functions available in those scripts.

By selecting **GameObject** > **SetActive (bool)** - you are choosing to run the **SetActive function** from the **GameObject script**.

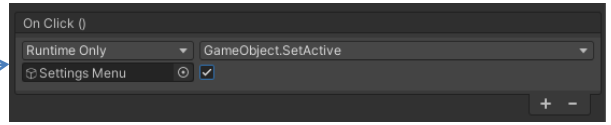
With the function now assigned, you will notice a subtle checkbox has appeared beneath the function dropdown.

The checkbox appears because the **SetActive (bool)** function requires a Boolean (or true/false) parameter.

You need to choose whether you want to call SetActive(true), which would activate the GameObject, or SetActive(false), which would deactivate it.

Since you want the Settings Menu to appear OnClick, you should use true.

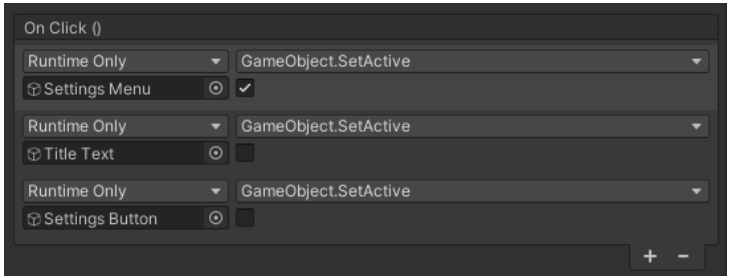
3. Use the checkbox to call SetActive(true) and activate the Settings Menu.



Try testing your app. If you select the settings button, the settings menu will appear on screen right on top of your title. Don't worry – we'll fix that next.

6.Make the title screen elements disappear

When you select the settings button, the settings menu should appear. But the title and settings button should also disappear.
You already know how to make an object appear. Making objects disappear is the same, except the SetActive Boolean should be set to false instead of true.



1. With the **Settings Button** GameObject still selected, use the **+** button to add two new actions in the On Click event.
2. Assign the Title Text object to one action and the **Settings Button** object to the other by dragging them from the Hierarchy.
3. Use the function dropdown to select **GameObject** > **SetActive**, then make sure the checkbox is disabled. This will call SetActive(false), making those objects disappear.

Now when you select the settings button, the menu should appear and the other elements should disappear.

7.Navigate back to the title screen

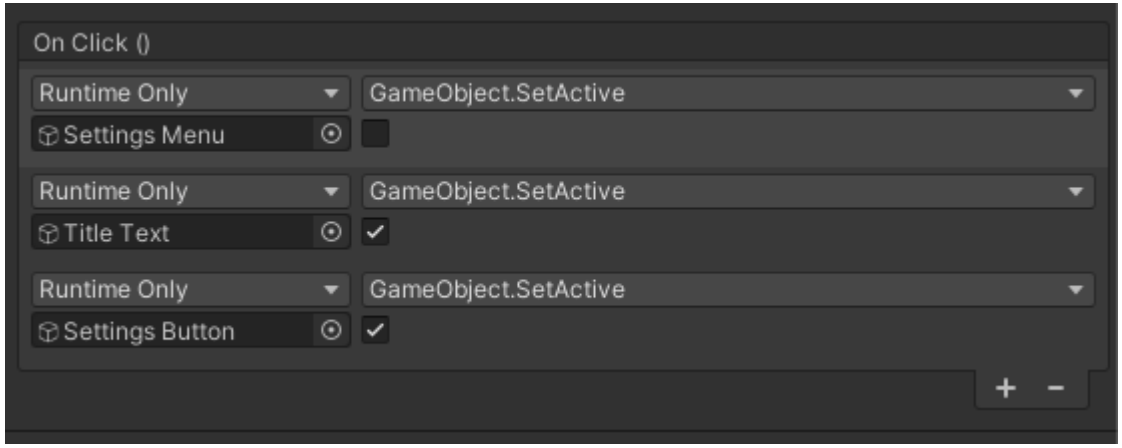
Navigation to the settings menu is complete. Next, the user needs to be able to navigate from the settings menu back to the title screen.

When the exit button on the settings menu is selected, the settings menu should disappear and the title screen elements should reappear:

1. Select the **Exit Button** GameObject and add three new empty actions to the On Click event.

2. Assign the **Settings Menu**, the **Title Text,** and the **Settings Button** GameObjects to the empty **Object** fields by dragging them from the Hierarchy.

3. Use the function dropdowns and the Boolean checkboxes to deactivate the **Settings Menu** GameObject, but activate the **Title Text** and **Settings Button** GameObjects.



Add toggles and sliders

Summary

As a UI becomes complex, you will inevitably need to implement toggles and sliders, which each allow the user a unique way to interact with an application. In this tutorial, you will add a toggle that allows the user to turn music on and off and a slider that allows them to control the volume.

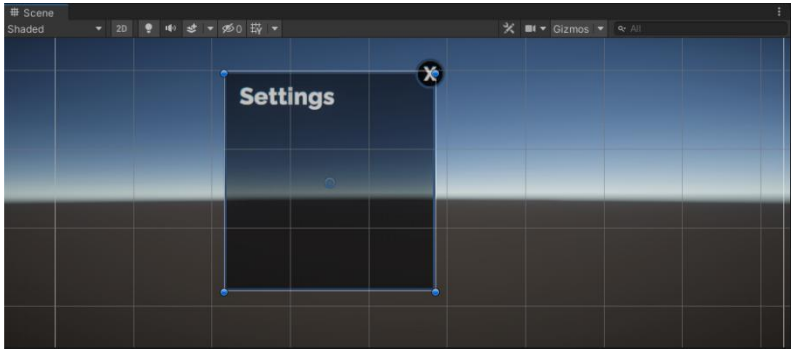
1.Overview

You currently have an empty settings menu. The obvious next step is to fill it with settings.
In this tutorial, you will add a toggle to turn music on and off and a slider to control volume. If you want, you can add additional settings too.

By the end of this tutorial, your project might look like this:

2.Add a music toggle setting

In the previous tutorial, you added functionality to buttons. Buttons are incredibly versatile UI elements.



Another very useful UI element is the toggle, which allows users to turn things on or off.

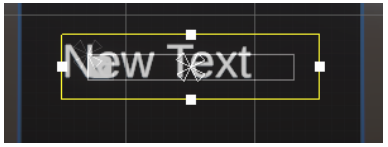
Let's add a toggle to allow users to turn music on or off in the scene.

First, let's bring up the settings menu and hide the title for easier editing.

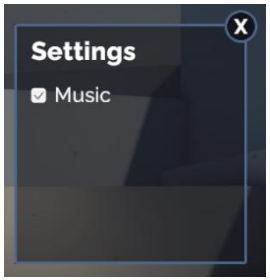
1. Activate the Settings Menu object and hide all other UI elements, then make sure you're in a good 2D perspective to edit the Canvas in the Scene view.

2. Right-click on the Settings Menu object and select **UI** > **Toggle**, then rename it "Music Toggle" in the Hierarchy. This will add a checkbox with the word Toggle next to it, anchored in the center of your menu.

By default, the Toggle label uses Unity's old text system rather than the more robust and optimized TextMeshPro. Let's replace that simple text with TextMeshPro.



3. Expand the **Music Toggle** GameObject and delete the **Label** object, then right-click on Music Toggle and select **UI** > **Text - TextMeshPro.** This will create a default New Text object, which is far too large and out of position.



4. Edit the text label to match the rest of your app and reposition it appropriately for the checkbox, then reposition the parent Music Toggle object where you'd like in the menu.

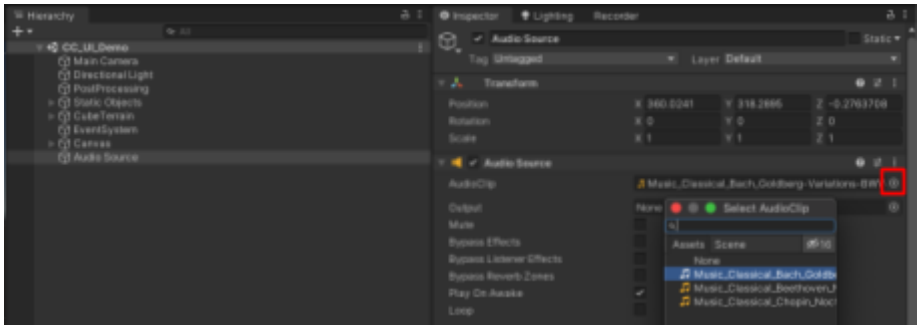
The toggle now looks great, but still doesn't do anything if you select it. Let's make it functional.

3.Add music to the scene

Before you can toggle the music on and off, you need to add some music into the scene!

1. In the Hierarchy, **right-click** > **Audio** > **Audio Source**. This is basically like adding a speaker into the scene.

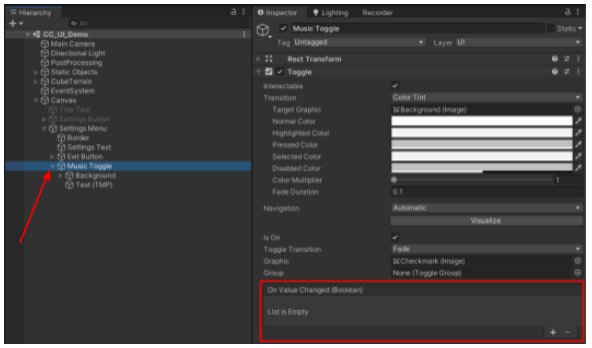
2. In the Audio Source component, for the **Audio Clip** property, use the object picker to select music for your project.



You can preview the music by playing your project. We have provided you with a few classical options, but you could also download and import other songs too!

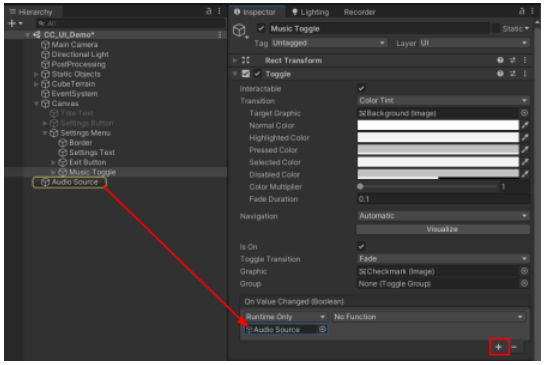
3. In the Audio Source component, set the **Volume** property to a value between 0.25-0.50. We don't want the music to be too loud or startling for people.

4.Make the music toggle functional



Now that you have music playing, let's add the ability to toggle it on and off. Just like with the Button component, there is a special Unity Event that is called whenever the toggle is turned on or off by the user.

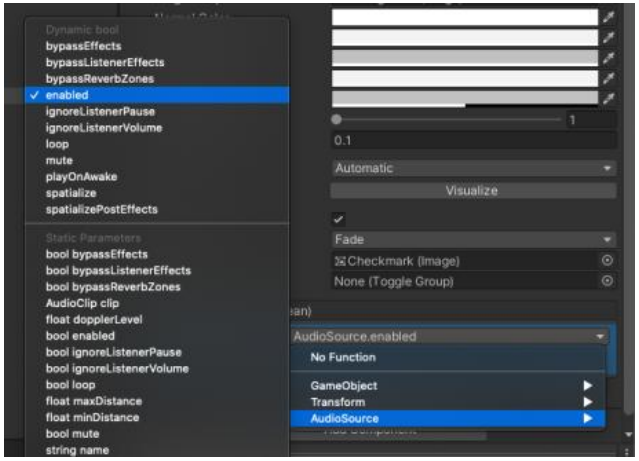
1. Select the Music Toggle object and locate the **On Value Changed (Boolean)** event at the bottom of the Toggle component.



In the **On Value Changed (Boolean)** Event, "Boolean" is the **parameter**. This Boolean parameter means that you can make different things happen if that Boolean is true or false. In this case, we want the music to play when it's true (toggle selected) and stop when it's false (toggle deselected):

Toggle on → OnValueChanged(true) → play music
Toggle off → OnValueChanged(false) → stop music

2. Select the **+** button within the On Value Changed Event to add a new action to the list, then assign the **Audio Source** GameObject. This will allow you to call a function from the **Audio Source** GameObject.



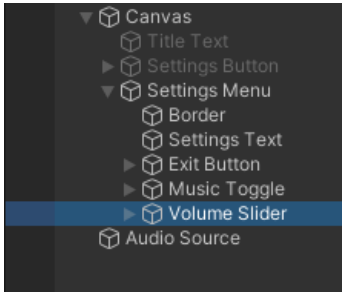
3. Using the action dropdown, which is currently set to **No Function**, select **AudioSource** > **enabled** from the upper section of the list.

As the toggle is set to true or false, so is the **enabled** property of the Audio Source. Now, when the user enables and disables the toggle, it will also enable and disable the Audio Source component, turning the music on and off.

Try testing your app. Using the toggle should now stop and play your music.

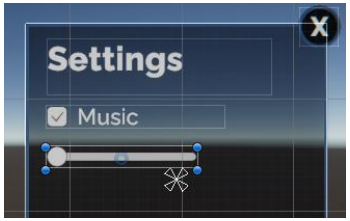
Some people might also want to adjust the volume of the music – let's add that functionality next.

5.Add a volume slider



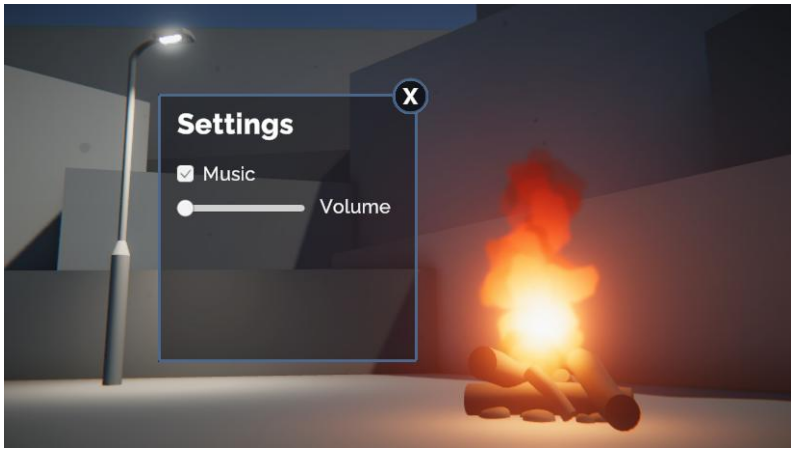
To control the volume, we'll use a new UI element: **the slider**. Sliders are very useful and allow more precise control over settings like volume, brightness, text size, price ranges, and more.

1. Right-click on the Settings Menu object and select **UI** > **Slider**, then rename it "Volume Slider" in the Hierarchy.



2. Use the Rect Transform component or **Rect** tool in the Scene view to resize and reposition the slider appropriately on the menu.

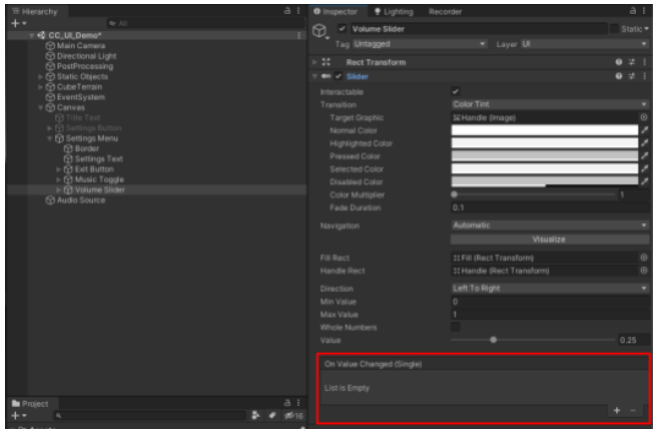
3. Right-click on the **Volume Slider** GameObject and create a child **Text - TextMeshPro** GameObject. Then edit the style and position of the text so that it says Volume next to the slider.



6.Make the volume slider functional

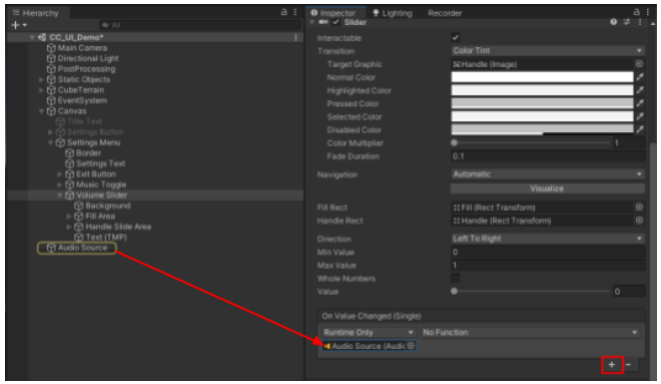
With the slider now looking good, we just need to make it functional. When the slider is adjusted by the user, the volume of the Audio Source should change dynamically along with it.

1. Select the Volume Slider object and locate the **On Value Changed (Single)** Event within the Slider component.



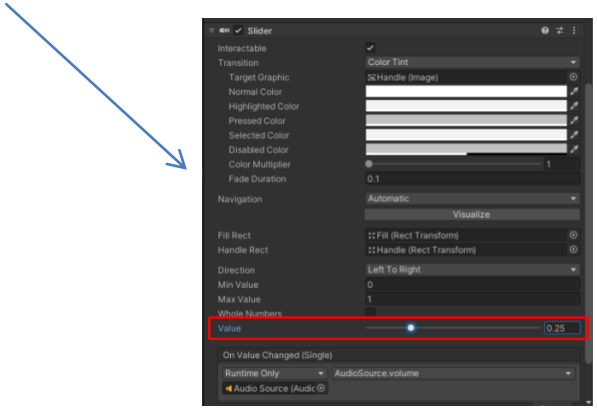
2. Add a new action to the list, then assign the **Audio Source** object to it.

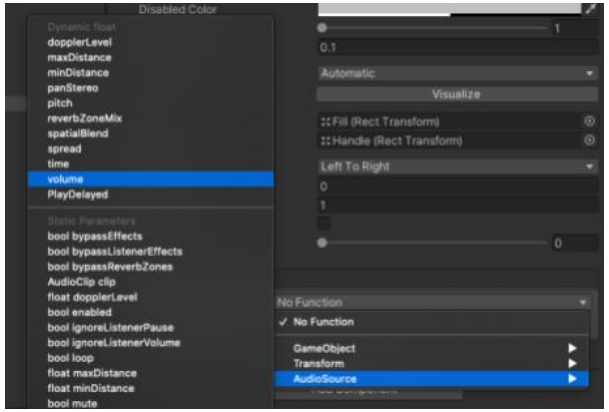
3. From the action dropdown, select **AudioSource** > **volume** from the upper section of the list.



In this case, the property is a **float** instead of the Boolean you used with the toggle. A float is a number with decimal values. The slider will set the volume to a decimal value between 0 and 1, depending on where the slider handle sits. If you test your app now, you'll notice something strange. The music will be playing at regular volume, but the slider will still be set all the way to the left, since that is its default position. **At the start of the scene, the slider position should match the current volume.**

4. In the Slider component, set the default **Value** property to match the starting volume of your Audio Source. This will move the slider handle in your scene as well.





7.Explore: Add new UI elements

You now understand how to create new UI elements and link them up with simple functionality.

Try adding some new settings to your menu to give the user more control over their experience!

You could add toggles to view or hide certain objects, sliders to control the brightness of lights, buttons to switch between night and day, or any other fun feature you can think of!