













UI Framework Documentation



Welcome, Game Devs! This UI Framework is designed to take the pain out of managing UI in your Unity projects. With a centralized UIManager, you can handle all your UI elements with ease. Panels, Buttons, Toggles, and more – let's dive in and make your UI shine! 🚀

Table of Contents

- 1.  Introduction
- 2.  Installation
- 3.  Getting Started
- 4.  Adding UI References
- 5.  Using the UIManager
 - - Accessing UI Components
 - - Setting Component Properties
 - - Adding Listeners to Components
- 6.  Panel Management
- 7.  UI Library
 - - Updating the UI Library
- 8.  Editor Integration
- 9.  Example Usage
- 10.  Best Practices
- 11.  Troubleshooting
- 12.  Support

Introduction

Hello fellow game devs! This framework is here to streamline UI management in your Unity projects.

It centralizes UI elements so you can easily manage Panels, Buttons, Toggles, and more. Now, less time on UI, more time on gameplay! 🎮

Installation

1. Import the Package: Import the UI Framework package into your Unity project by navigating to Assets > Import Package > Custom Package and selecting the downloaded package file.
2. Dependencies: Make sure TextMeshPro is installed in your project (it's essential for some components like TMP_Text).
 - TextMeshPro is available in the Unity Package Manager if you don't already have it!

Getting Started

1. Add the UIManager: Create an empty GameObject named "UIManager" and attach the UIManager script to it. Alternatively, let the framework handle the setup for you!
2. Set Up Your UI: Design your UI elements in the Unity Editor (Panels, Buttons, etc.) just like you normally do.

Adding UI References

The UIManager needs references to your UI elements to manage them.

1. Open the UIManager Inspector: Select the UIManager GameObject in the hierarchy to see it in the Inspector.
2. Click Refresh: Click "Refresh UI Hierarchy" to scan the scene for all UI elements.
3. Add Elements: Use the "Add" button next to elements you want to manage, like Buttons, Panels, etc.

Make sure ****Panels**** have `_Panel` as a suffix, or the UIManager might get confused. 🤔

Using the UIManager

The UIManager lets you access and manage your UI components without diving deep into the scene hierarchy. Here's how:

****Accessing UI Components****

```
Button myButton =  
UIManager.Instance.GetUIComponent<Button>(UI_Library.MyButton_Path);
```

****Setting Properties****

```
UIManager.Instance.SetUIComponentProperty<Text>(UI_Library.MyText_Path,  
text => {  
    text.text = "Hello, Player!";  
});
```

****Adding Event Listeners**:**

```
UIManager.Instance.SetUIComponentListener<Button>(UI_Library.MyButton_Path, OnButtonClick);
```

```
void OnButtonClick() {  
    Debug.Log("Button was clicked! 🌟");  
}
```

Panel Management

The framework makes it easy to toggle Panels on and off.

****Activate a Panel**:**

```
UIManager.Instance.SetPanelActive(UI_Library.Settings_Panel_Path,  
true);
```

****Deactivate Other Panels**:**

```
UIManager.Instance.SetPanelActive(UI_Library.MainMenu_Panel_Path, true,  
deactivateOthers: true);
```

UI Library

The UI_Library class holds references to all your UI elements.

When you add or modify elements, simply update the library by hitting "Update UI Library" in the Inspector.

After that, you can access elements with constants like `UI_Library.MyButton_Path` – no more hardcoding! 🎉

Example Usage

Check out this code for adding event listeners and initializing UI components:

```
private void Start() {  
  
    UIManager.Instance.SetUIComponentListener<Button>(UI_Library.PlayButton_Path, OnPlayButtonClick);  
  
    UIManager.Instance.SetUIComponentProperty<Text>(UI_Library.WelcomeText_Path, text => { text.text = "Welcome to the Game!"; });  
}
```

```
}
```

Best Practices

Here are a few tips for using the UIManager effectively:

- **Naming Conventions**: Be consistent! Panels should end with `_Panel`, buttons should be descriptive.
- **Singleton**: Only have one instance of UIManager in your scene (avoid duplicates).
- **Keep Hierarchy Clean**: Keep your UI hierarchy organized to avoid confusion.

Support

For support, reach out to us at [your-email@example.com]. We're here to help! 😊