

UI Framework

Copyright @2022 GamesTan

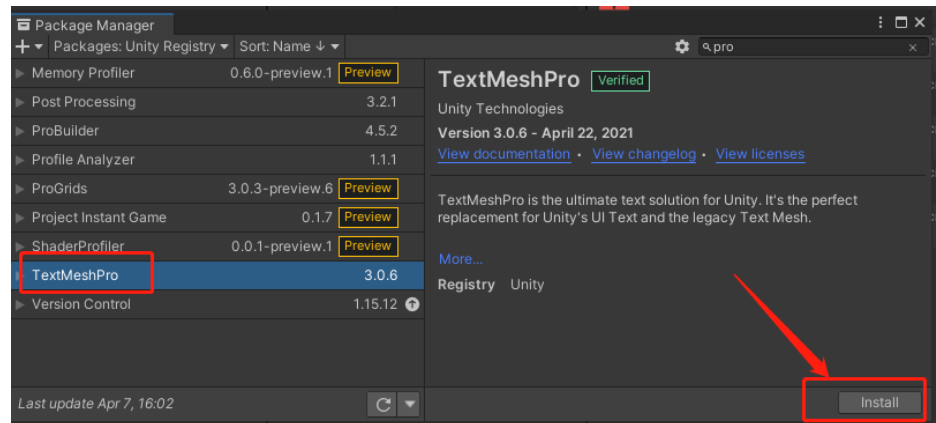
Catalogue

1. Quick Start.....	3
2. Start To Use	5
1. Config Project setting	5
2. How to add a new window.....	5
1. Create a Prefab.....	5
2. Attach Script.....	6
3. Create some UI elements	6
4. Define UI Instantiate info	7
5. Create a Script (script's name has to be the same as prefab's name).....	8
6. Auto Bind Reference.....	9
7. Open UI by script.....	9
3. Script API.....	11
1. Asset Management	11
1. Naming rule.....	11
2. LoadConfig.....	12
3. Load Prefab & Instance Prefab	12
2. API	13
1. ResourceManager API	13
2. UIBaseWindow API	14
3. Features.....	14
1. ScrollView.....	14
2. UIModel.....	15
3. Sub Panel.....	15

1.Quick Start

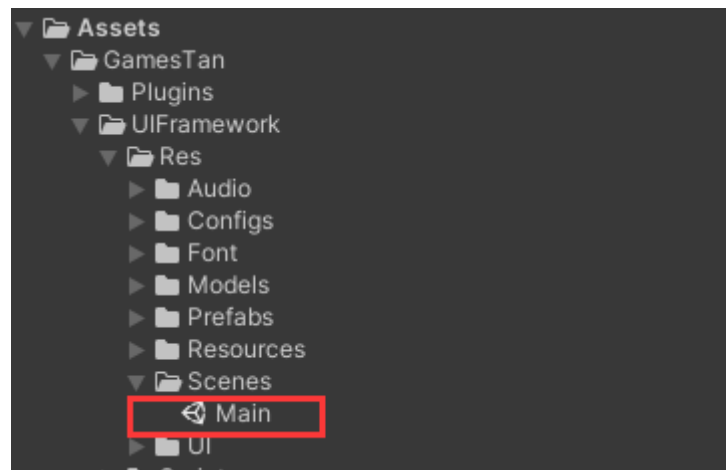
1. Install TextMeshPro

Open “Window/Package Manager” Install **TextMeshPro**

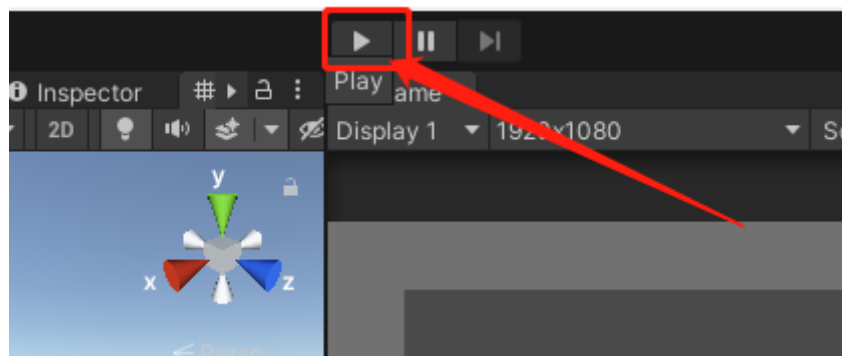


2. Open scene

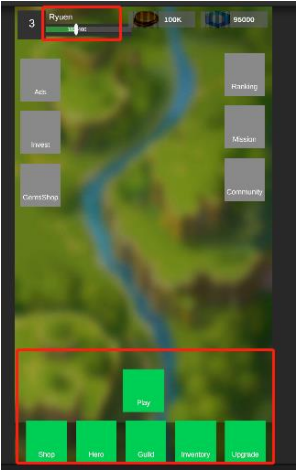
“Assets/GamesTan/SuperScrollRect/Examples/**Main.unity**”



3. Click “Play” button ,and you will see the demo

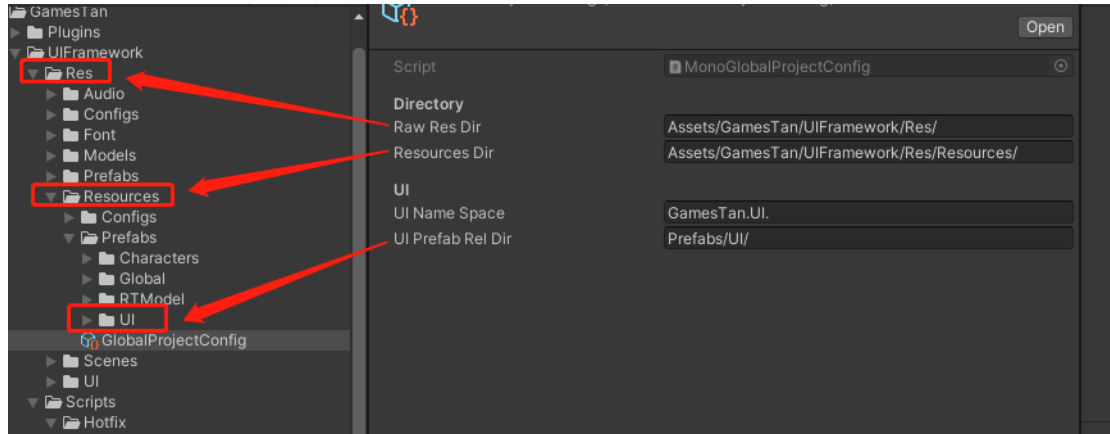


4. All Green Buttons can open some window, try to click them.



2. Start To Use

1. Config Project setting



Raw Res Dir	Project global resource directory (include all resource)
Resources Dir	Resources Directory (where code can direct load) Anything that can be load from ResourceManager should be in this directory or it's child directory
UI Prefab Rel Dir	The Directory which UI prefab putted into UI Full path = ResourcesDir + UIPrefabRelDir In this demo : "Assets/GamesTan/UIFramework/Res/Resources/Prefabs/UI/"
UI Namespace	UI script's namespace (used to bind with prefab by framework)

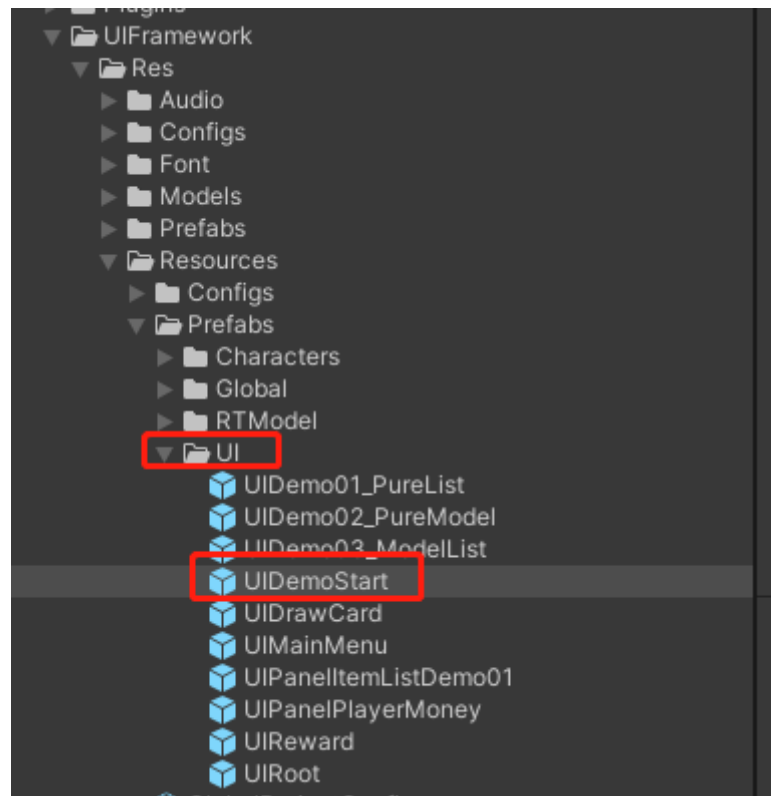
2. How to add a new window

1. Create a Prefab

Create a prefab (eg: "UIDemoStart") in directory

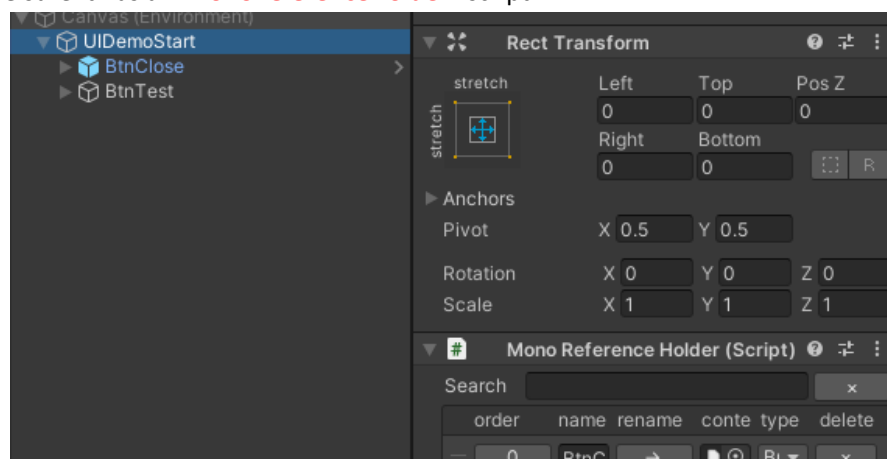
(GlobalProjectConfig. ResourcesDir + GlobalProjectConfig.UIPrefabRelDir)

"Assets/GamesTan/UIFramework/Res/Resources/Prefabs/UI/"



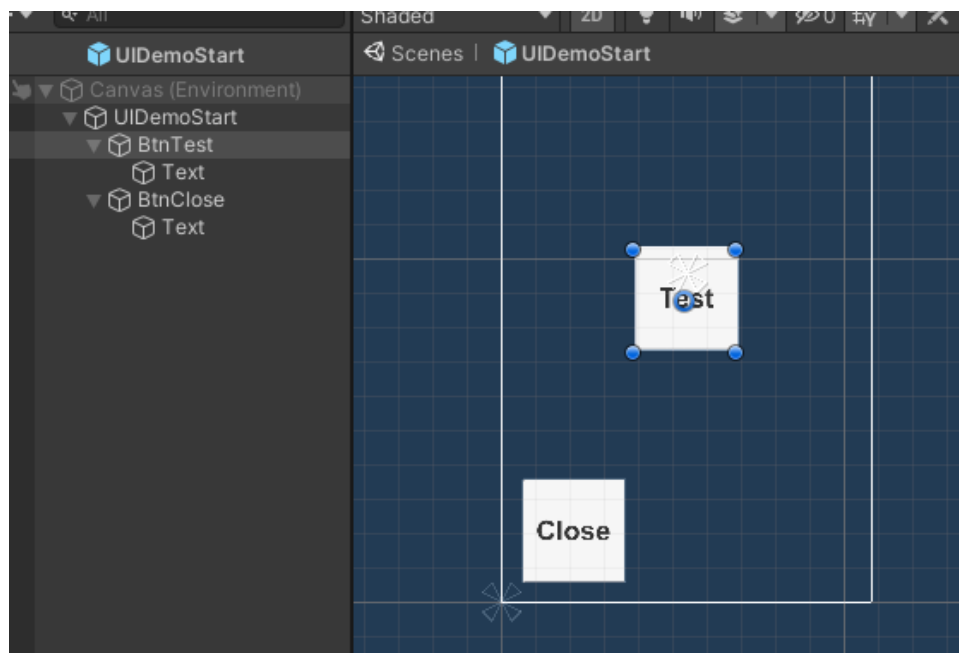
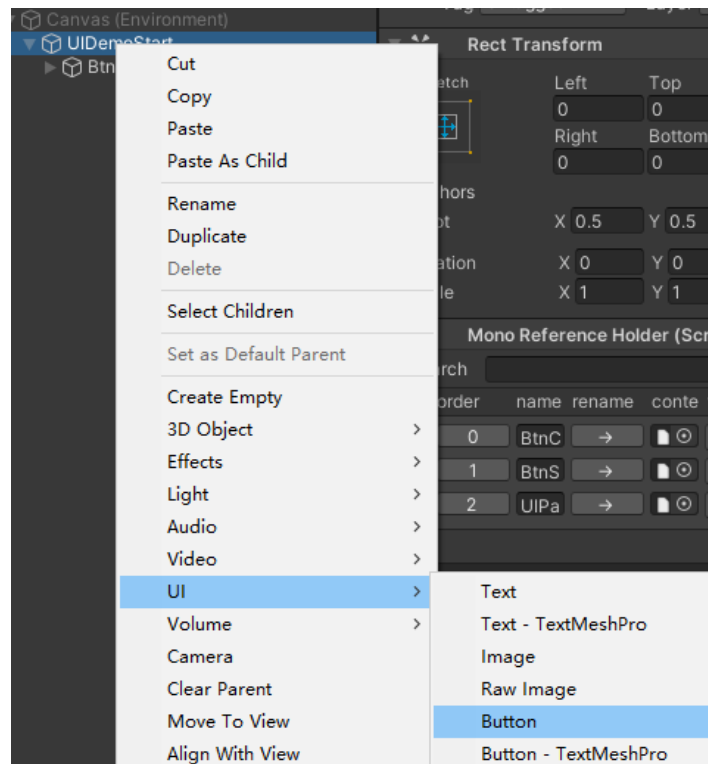
2. Attach Script

Make sure it has a " **MonoReferenceHolder**" script



3. Create some UI elements

In demo : we create 2 button , "BtnClose" and "BtnTest"



4. Define UI Instantiate info

Define a **WindowCreateInfo** object in file **UIDefine.cs**

The screenshot shows a C# script named `UIDefine.cs` in an IDE. The script is part of the `GamesTan.UI` namespace. It contains several static `WindowCreateInfo` objects for different UI elements. The `UIDemoStart` object is highlighted with a red box, and its `dir` property is set to `"UIDemoStart"`, which is also highlighted with a red box.

```
namespace GamesTan.UI {  
    public partial class UIDefine {  
        // Common  
        public static WindowCreateInfo UIChat = new WindowCreateInfo( dir: "UIChat", dep: EWindowDepth.Normal);  
        public static WindowCreateInfo UIMainMenu = new WindowCreateInfo( dir: "UIMainMenu", dep: EWindowDepth.Normal);  
        public static WindowCreateInfo UIRoot = new WindowCreateInfo( dir: "UIRoot", dep: EWindowDepth.Normal);  
        public static WindowCreateInfo UIDrawCard = new WindowCreateInfo( dir: "UIDrawCard", dep: EWindowDepth.Normal, isEffectMode: true);  
        public static WindowCreateInfo UIDemoStart = new WindowCreateInfo( dir: "UIDemoStart", dep: EWindowDepth.Normal);  
    }  
}
```

5. Create a Script (script's name has to be the same as prefab's name)

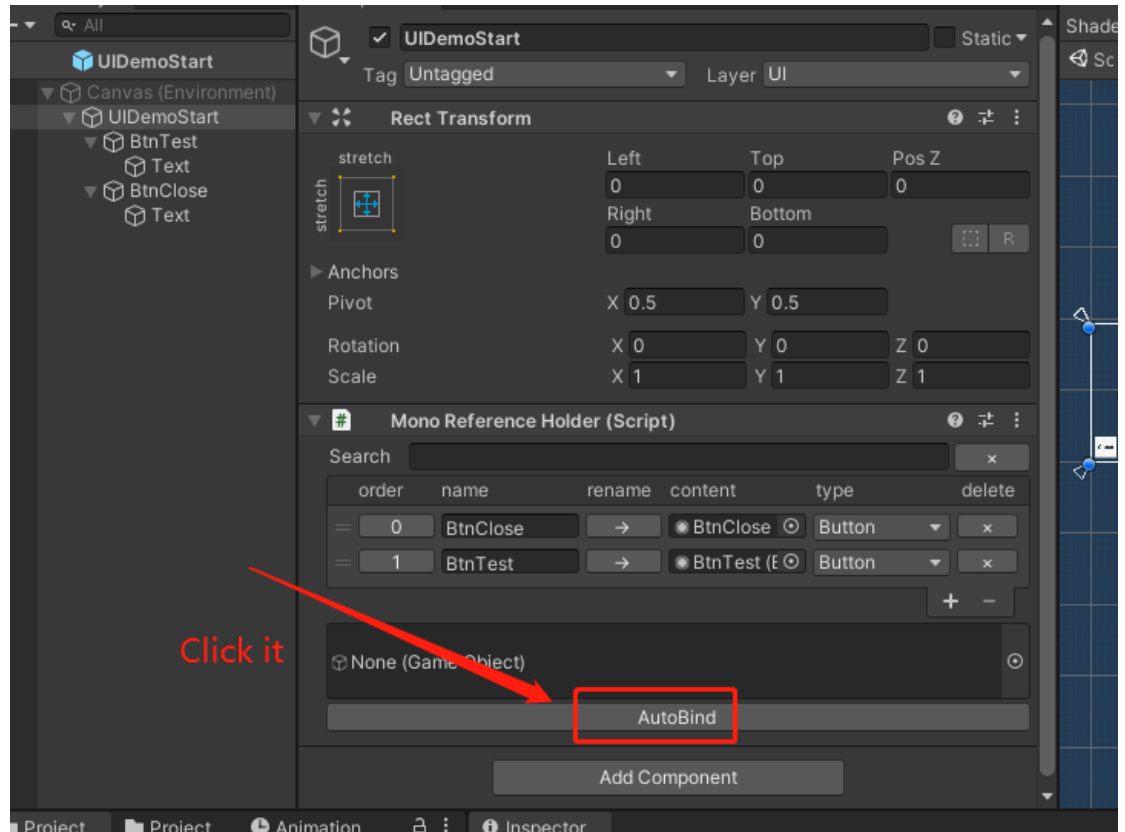
Create a script **"UIDemoStart.cs"**

The screenshot shows a C# script named `UIDemoStart.cs` in an IDE. The script is part of the `GamesTan.UI` namespace. It inherits from `UIBaseWindow`. The `UIDemoStart` class is highlighted with a red box, and its `namespace` is also highlighted with a red box. A red arrow points from the `UIDemoStart` class to the `GamesTan.UI` namespace, with a red text annotation: "script's namespace should be same as GlobalProjectConfig.UINamespace".

```
namespace GamesTan.UI {  
    public class UIDemoStart : UIBaseWindow {  
        private Button BtnClose => GetRef<Button>( name: "BtnClose");  
        private Button BtnTest => GetRef<Button>( name: "BtnTest");  
  
        public override void OnClick_BtnClose() {  
            base.OnClick_BtnClose();  
            Debug.Log( message: "UIDemoStart:BtnClose");  
        }  
  
        public void OnClick_BtnTest() {  
            Debug.Log( message: "UIDemoStart:OnClick_BtnTest");  
        }  
    }  
}
```


6. Auto Bind Reference

Click "**MonoReferenceHolder**" component's "**AutoBind**" button



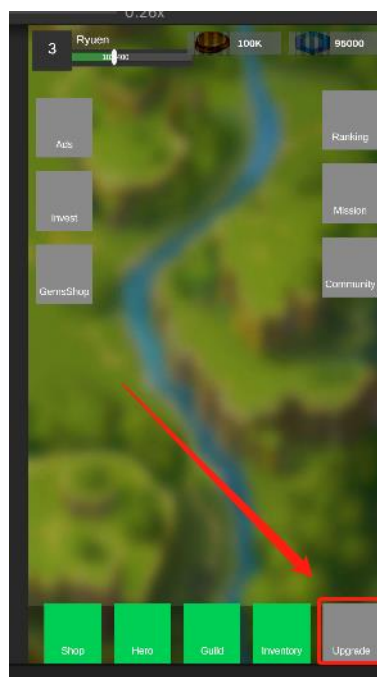
7. Open UI by script

Use script to open the window wherever you want

```
otfix > Game.View > UI > Systems > C# UIMainMenu.cs
Define.cs x C# EGameEvent.cs x C# UIMainMenu.cs x C# UIDemoStart.cs x C# UI
e modifying a script while Unity Editor is being in Play Mode. This can lead to a loss of the s

Debug.Log( message: "OnClick_BtnInventory");
RewardDatas.SetItems( coin: 10, exp: 20, dirty: 30, diamo
OpenWindow(UIDefine.UIReward);
}
jiepengtan *
public void OnClick_BtnUpgrade() {
    Debug.Log( message: "OnClick_BtnUpgrade");
    OpenAndClose(UIDefine.UIDemoStart);
}
jiepengtan
protected void OnSlider_SliderExp(float value) {
```

A. Run and Click the button



3. Script API

1. Asset Management

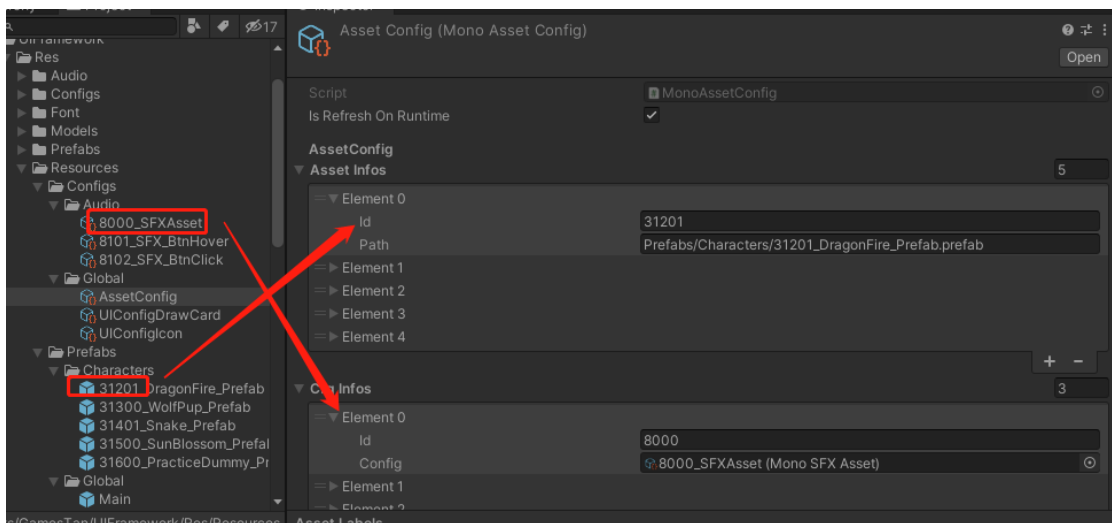
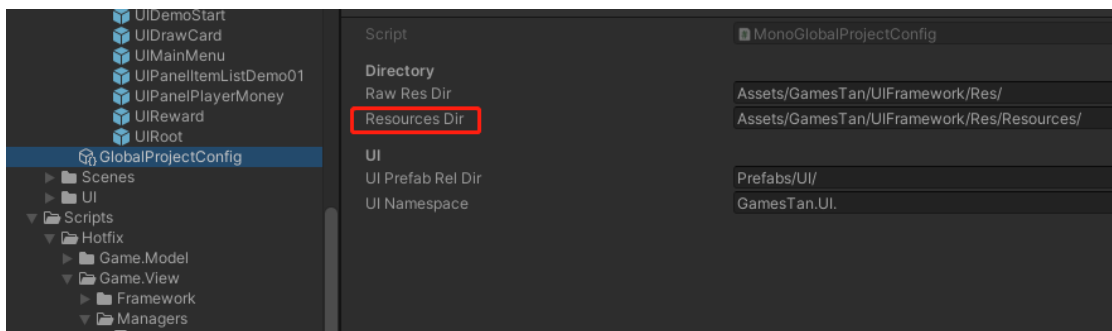
1. Naming rule

ConfigDir = GlobalProjectConfig.ResourcesDir + “/Configs”

PrefabDir = GlobalProjectConfig.ResourcesDir + “/Prefabs”

Any Prefab in directory **PrefabDir** or any ScriptableObject in directory **PrefabDir** whose name match the rule (AssetId + “_” + AssetName)

would be added to AssetConfig automatically, and can be loaded by **ResourceManager** with **AssetId**



2. LoadConfig

Eg: AssetId = 8000 and AssetName = TestAsset
8000_TestAsset.asset

You can load the config using the code below:

```
var config = ResourceManager.Instance.LoadConfig<ScriptableObject>( assetId: 8000);
```

3. Load Prefab & Instance Prefab

Eg: AssetId = 31201 and AssetName = DragonFire_Prefab
31201_DragonFire_Prefab.prefab

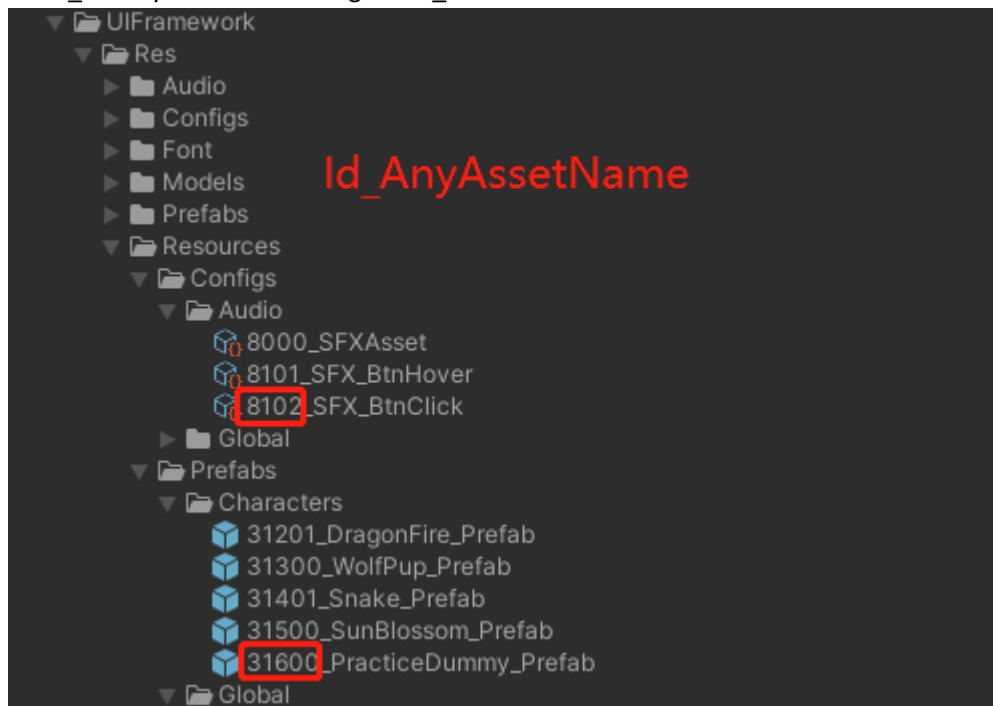
You can load or instantiate the prefab using the code below:

```
// Load prefab then instantiate it
var prefab :GameObject = ResourceManager.Instance.LoadPrefab( assetId: 31201);
var goFab :GameObject = GameObject.Instantiate(prefab);

// Instantiate Prefab directly
var go :GameObject = ResourceManager.Instance.Instantiate( assetId: 31201);
```

1. Naming Rule

- a) Id + "_" + AnyAssetName eg: 8001_SFXBtnHover



2. API

1. ResourceManager API

```
ResourceManager (in GamesTan)
  _gameConfig:AssetConfig
  _id2Prefab:Dictionary<int,Object>
  Instantiate<T>(int assetId, Transform parent):T
  Instantiate(int assetId):GameObject
  Instantiate(int assetId, Transform parent):GameObject
  Instantiate<T>(int assetId, Vector3 pos, Quaternion rot, Transform parent):T
  Instantiate(int assetId, Transform parent, Vector3 localPos, Vector3 localScale):GameObject
  Instantiate(int assetId, Vector3 pos, Quaternion rot, Transform parent, bool isDepartObj):GameObject
  LoadPrefab(int assetId):GameObject
  LoadConfig(int configId):ScriptableObject
  LoadAsset<T>(int assetId):T
```

Instantiate	Initialize a prefab by Id
LoadPrefab	Load Prefab by Id
LoadAsset	Load Any Type Asset by Id
LoadConfig	Load ScriptableObject by Id

2. UIBaseWindow API

Lifecycle:

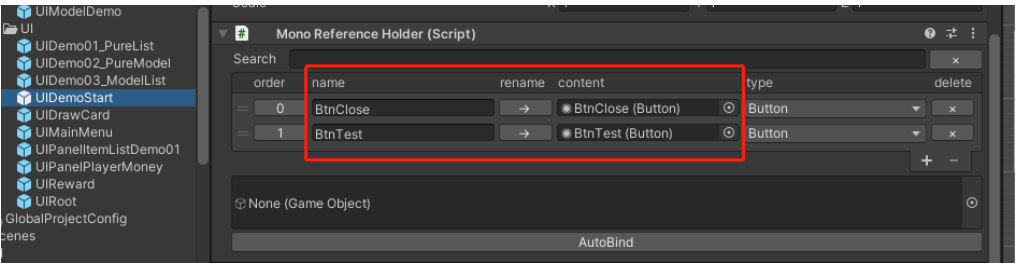
```
DoAwake():void
DoStart():void
OnClose():void
```

DoAwake	Called when window is opened
DoStart	Called after DoAwake when window is opened
OnClose	Called when the window was closed

API:

```
GetRef<T>(string name, bool isTry):T
CloseWindow(WindowCreateInfo windowInfo, bool isForce):void
OpenWindow(WindowCreateInfo windowInfo, bool isCloseOther):void
OpenAndClose(WindowCreateInfo windowName, bool isCloseOther):void
```

GetRef	Get object reference in component “MonoReferenceHold”
--------	---



CloseWindow	Close the window defined in UIDefine
OpenWindow	Open the window defined in UIDefine
OpenAndClose	Open the window defined in UIDefine and close self

3. Features

1. ScrollView

See demo script “UIDemo01_PureList.cs”

2. UIModel

See demo script "UIDemo02_PureModel.cs"

3. Sub Panel

See demo script "UIDemo04_MutilSubPanel.cs"