

WIDGET MANAGER

Introduction

Thank you for downloading this package stay tuned this is just beginning or you may call it proof of concept more features are in queue.

After spending one year in Gaming industry I realize managing UI's in larger projects is headache, you have to keep following things in mind while developing some UI management system for your projects.

- Must have low memory footprint (have pool system)
- Easy to Manage & Plug n Play
- Easy way of communication with UI's
- UI's Persists between scenes, don't destroy on scene change
- When you create new UI, UI Manager class remains same means no code addition required, this is useful if multiple people are working on UI's no conflicts occurs in version control.
- Keep UI's separate from Games scenes it also prevents from version control conflicts.

Setup

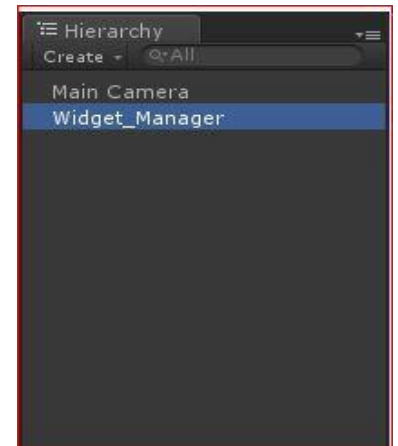
Setting Widget Manager is very simple just follow following 5 simple steps.

1. From menu Items Click Tools->Widget->Create Widget Manager.



It will create Widget_Manager game object in scene like this.

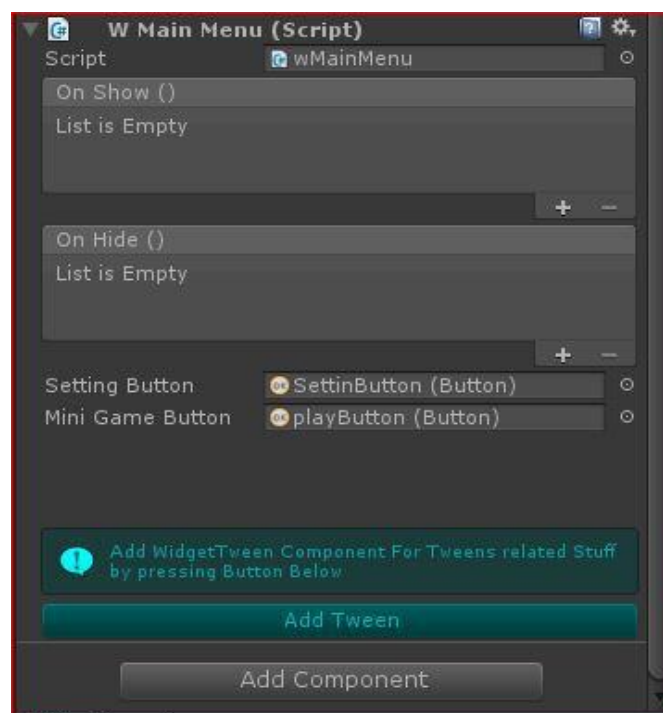
- 2: Make new Tag "Parent" and assign it to Canvas, under which you want to Instantiate you UI's.



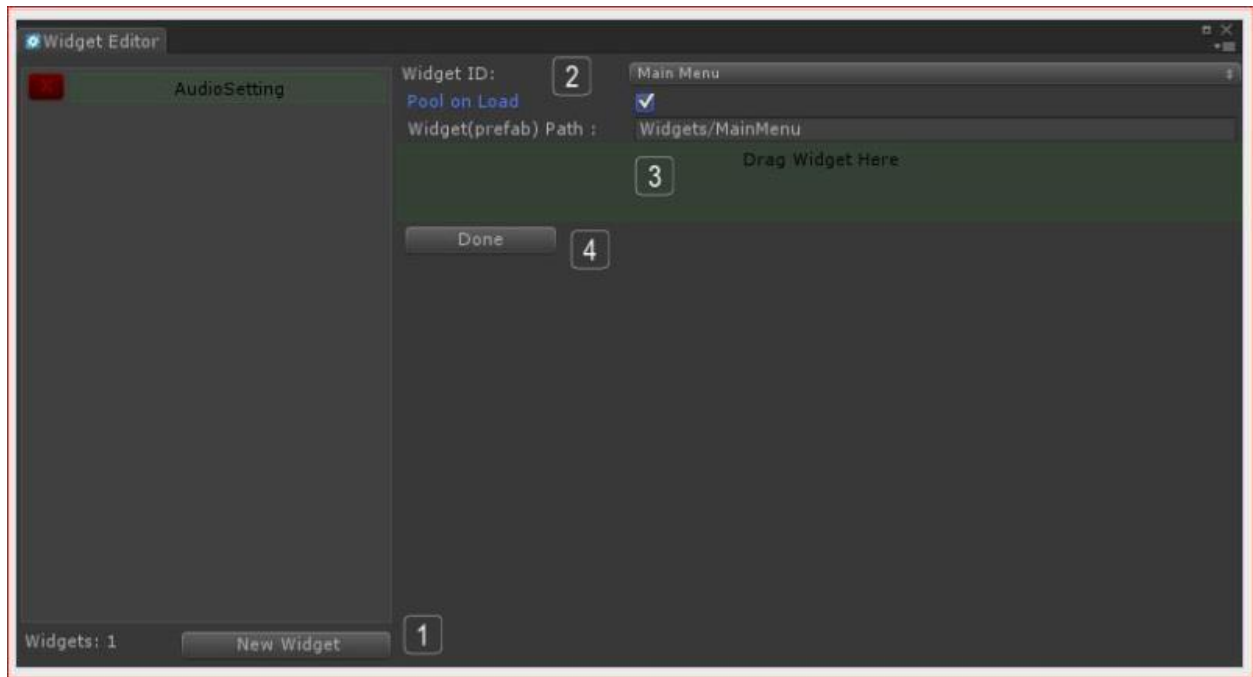
3: Open WidgetName Script (Widgets->Source Code->Common->WidgetName) and add your Widget name in it. NOTE: always add new enums at end.

```
namespace eeGames.Widget
{
    /// <summary>
    /// Here Add your Widget ID's
    /// </summary>
    22 references
    public enum WidgetName
    {
        MainMenu,
        Setting,
        AudioSetting,
        GameSetting,
        SaveSetting,
        MiniGame
    }
}
```

4: Create your UI in my case I name it MainMenu and attach a Script wMainMenu with it. All you have to do is inherit this class with Widget class. Now you inspector look like this, save prefab of this UI in Resources Folder.



5: Now you are one step away to see this Widget in action now open widget editor window (**Ctrl + Shift + w**)



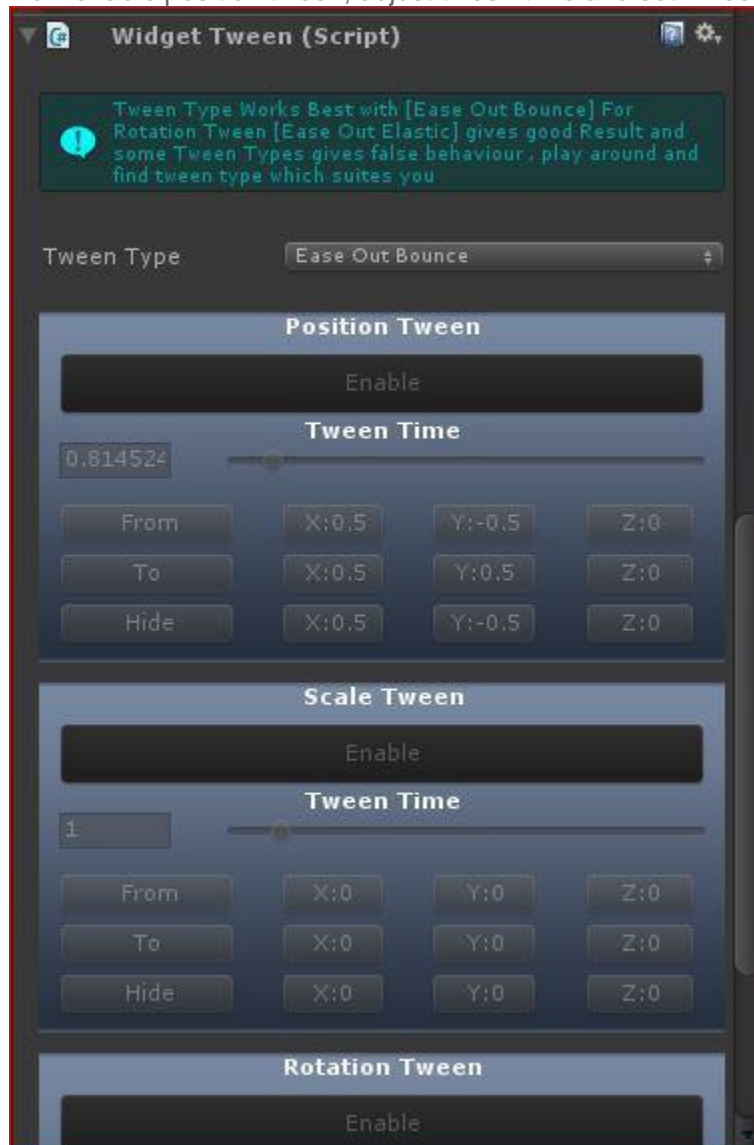
- 1: click New widget button
 - 2: select widget id from drop down and check Pool on Load if you want to
 - 3: drag you Widget prefab to green area.
 - 4: click "Done" button
- that's all you are ready now load you widget by calling

```
WidgetManager.Instance.Push(WidgetName.MainMenu)
```

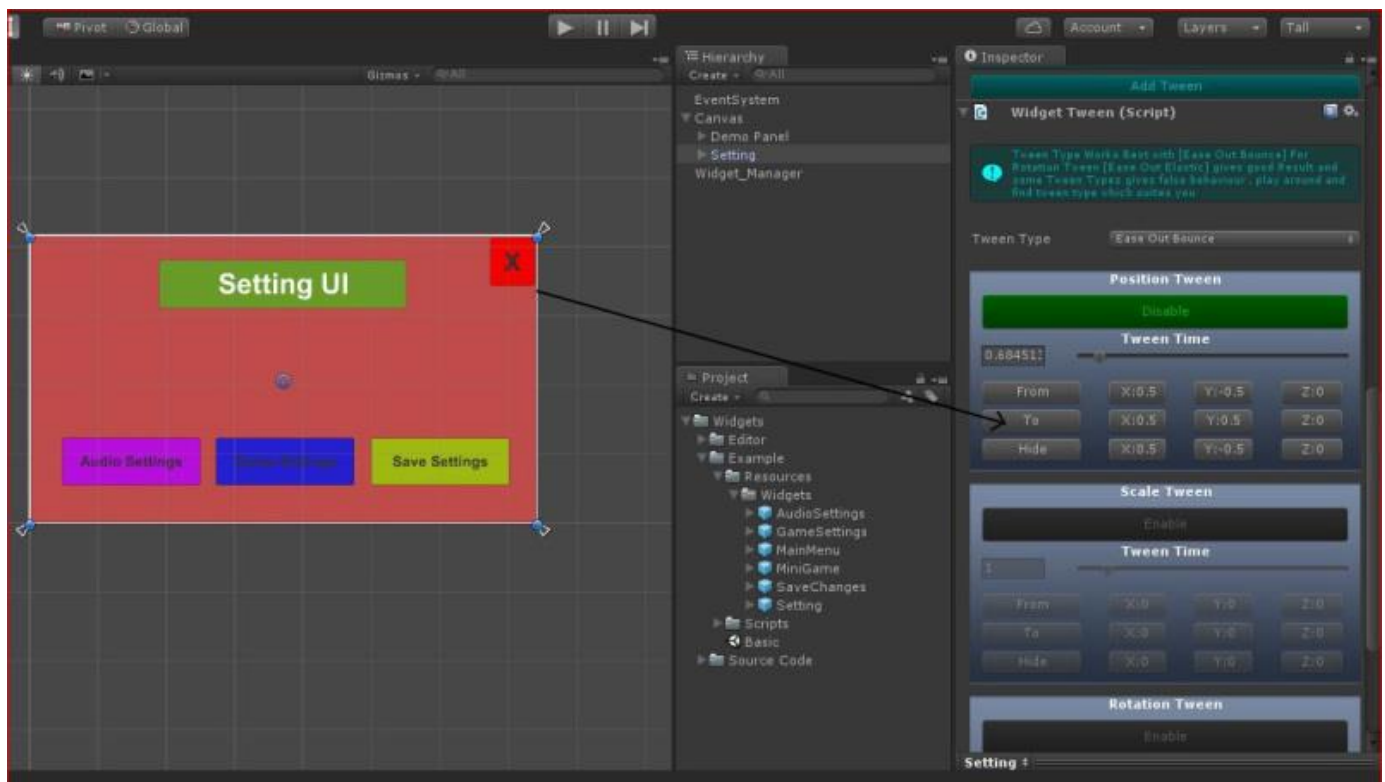
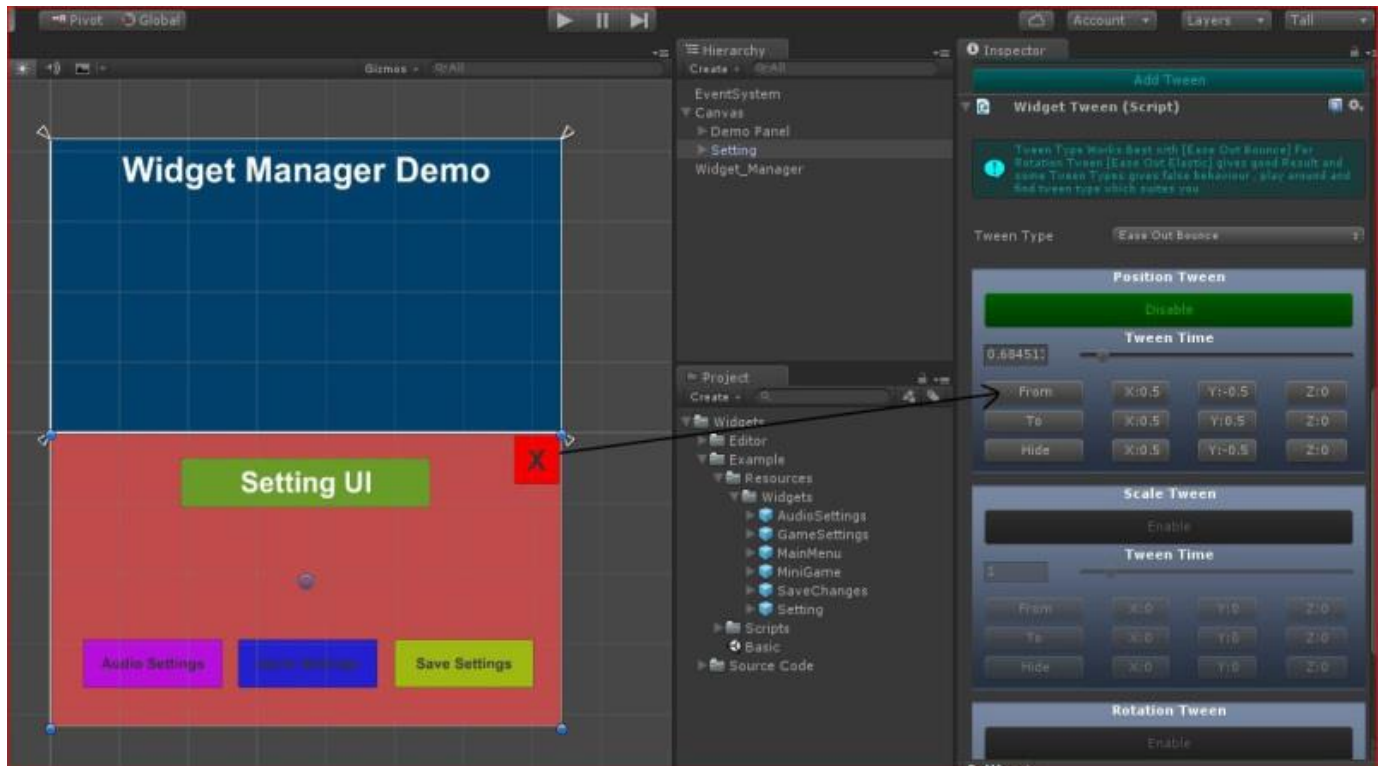
Adding Tween

Click Add Tween button Widget Tween Component Gets added to your UI.

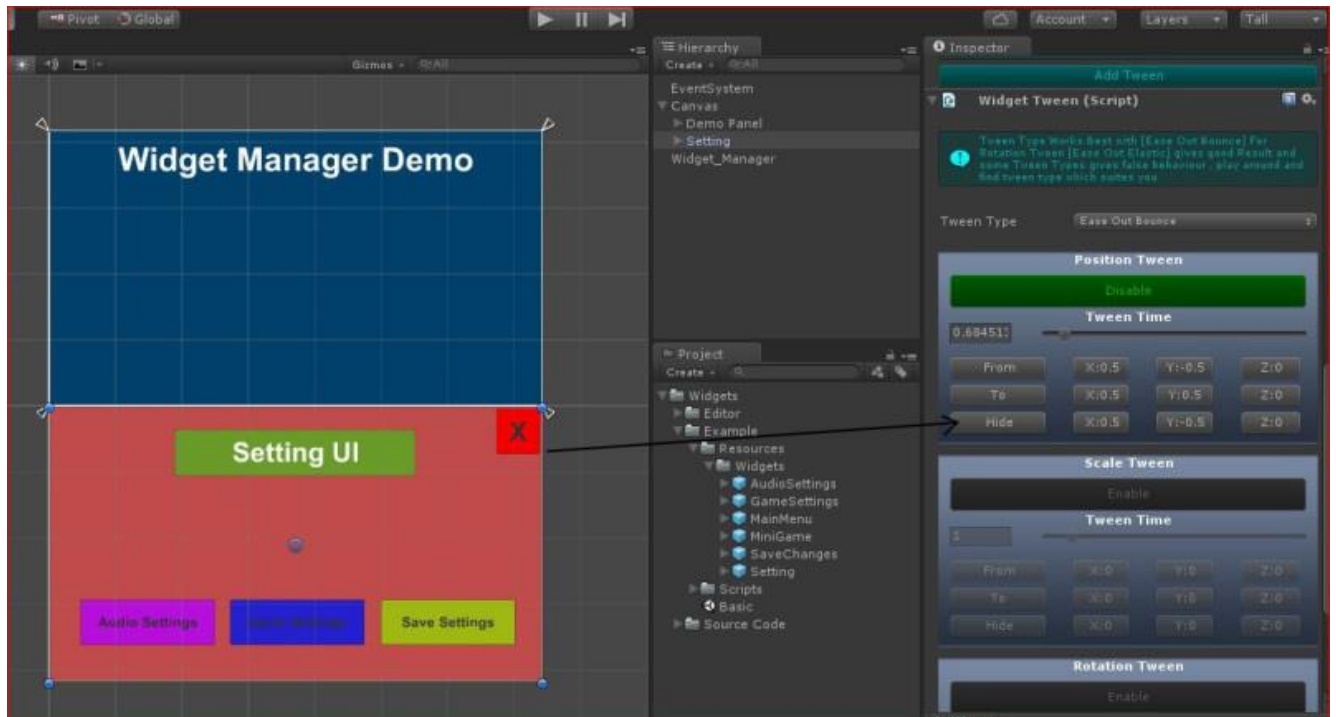
Now enable position tween, adjust tween time and set Tween type.



Now Capture OnShow Tween positions by clicking From & To button



Here capture OnHide Tween Position by clicking hide button.



Passing Data to Widgets

just make the function public in widget class you want to call for example you have Populate method in Inventory class which takes List of UserItem as argument you can call it from anywhere simple by couple of line like

```
var widget = WidgetManager.Instance.GetWidget(WidgetName.Inventory) as wInventory;  
if(widget != null) widget.Populate(new List<UserItem>());
```

Update Widget

override UpdateWidget method in your class and write you update logic in it, you can call this method from anywhere simply by calling

```
WidgetManager.Instance.UpdateWidget(WidgetName.MainMenu);
```

Update # 2

Introducing Actors

Actor Component allows you to add animations to your UI element through inspector without writing any code.

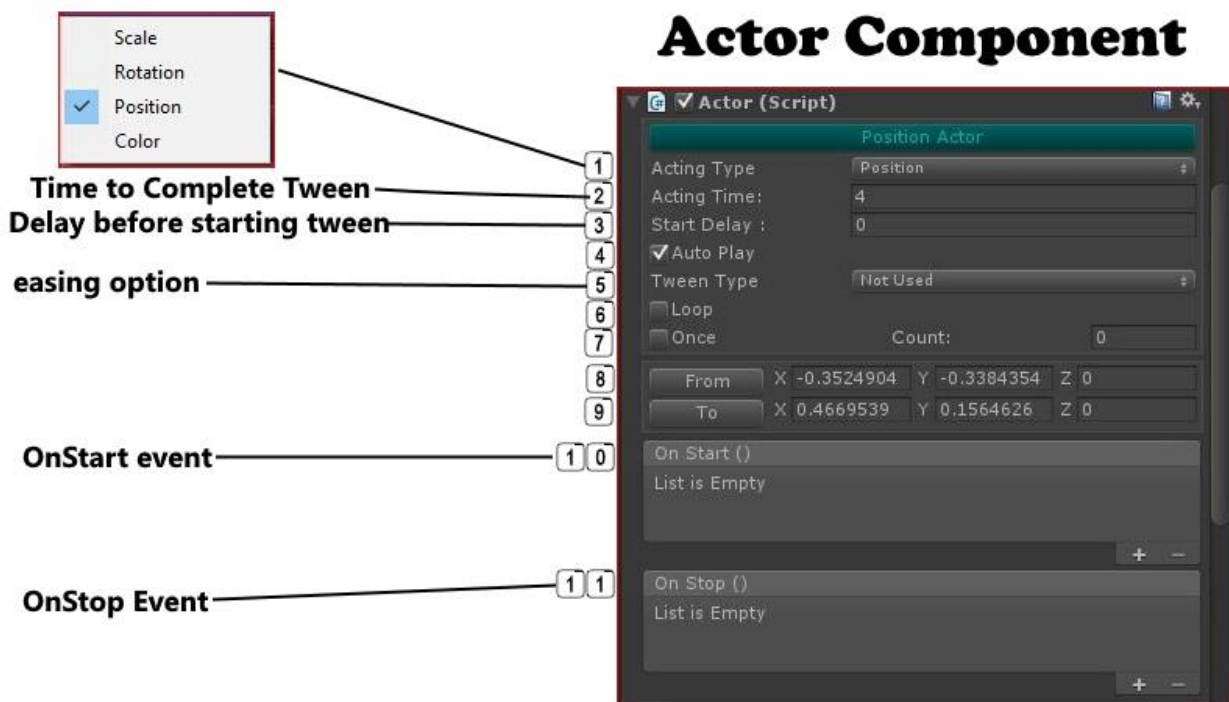
Introduction:

If you have many animating UI elements in your game, you have to create unity native animation for each element which becomes so verbose. Actor allows you to ditch unity's native animation and you can add tweens to UI elements by couple of clicks.

Actor Component provides four types of tweens **Position, Scale, Rotation & Color + Alpha**.

How to Use Actors:

Just add Actor Component with UI game object & set fields that's all.



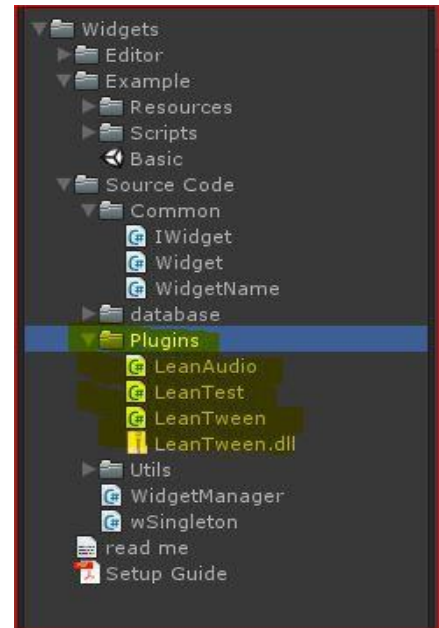
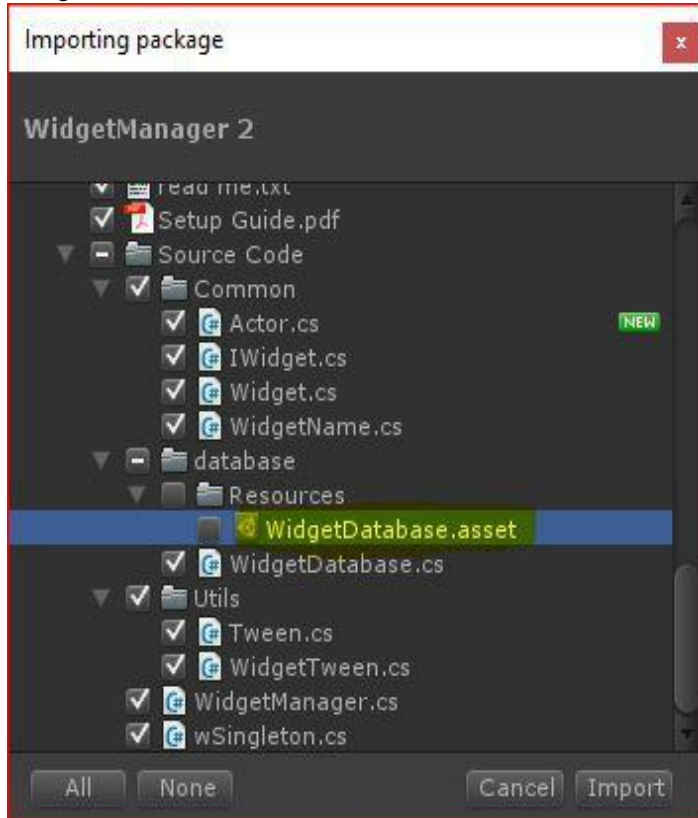
Actor Class Contains two public functions **public void PerformActing()** & **public void PerformReverseActing()** used them on events call backs.

There are 7 demo scenes explaining different Acting Types.

Upgrade Guide

If you are using old version of Widget Manager in your project & want to upgrade to latest version without breaking any things here is the guide just follow simple two steps.

1. Delete the Plugins folder which contains LeanTween stuff.
2. While importing latest package unchecked WidgetDatabase.asset



that's it, allow unity to recompile & you are good to go.