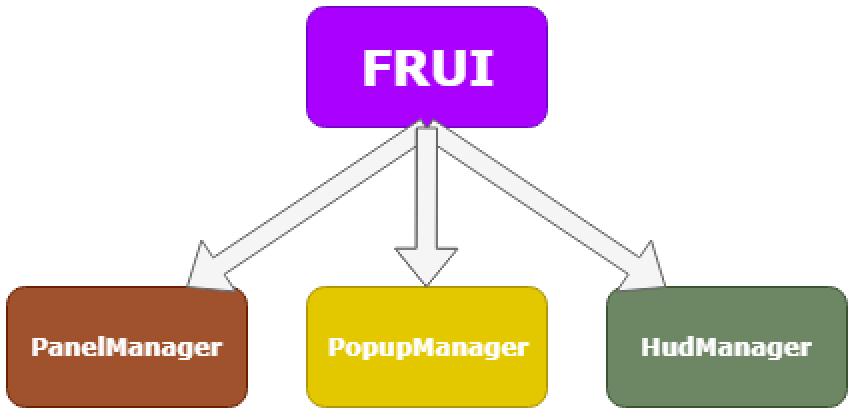
~FieryRanker UI System~

“FieryRanker UI System” is a UI System to utilize the GUI navigation for developers. This system handles the backbone of the Panel-Popup-Hud navigation, provides useful events and neither programming nor design is restricted to expand.

Let’s start with an overview:

How does the whole system work?

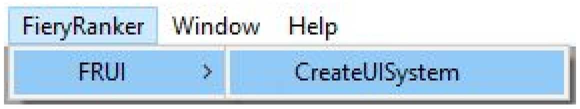


-Image 1-

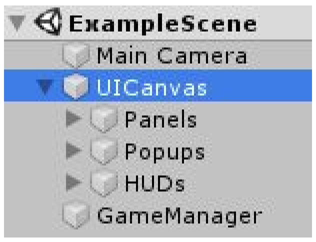
FRUI : Main Module.
PanelManager : Controls panel navigation.
PopupManager : Controls pop-up naviagation.
HUDManager : Controls HUD navigation.

How to use?

To start using FRUI, after importing the package; let the editor compile scripts first, then "FieryRanker" will appear in your top menu. Just click to the FieryRanker->FRUI->CreateUISystem (Image 2), after this point you will see there will be a "UICanvas" in the hierarchy (Image 3).



-Image 2-

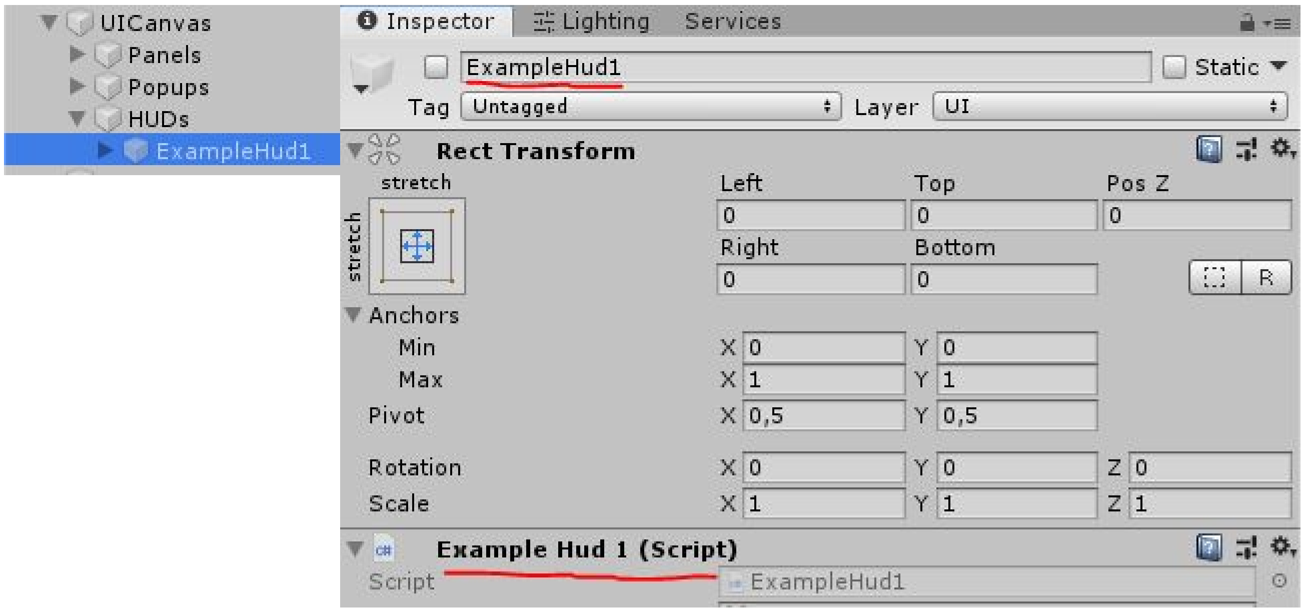


-Image 3-

- Every Panel created must be under "Panels" and must have a script derived from "PanelView".

- Every Pop-up created must be under "Popups" and must have a script derived from "PopupView".

- Every HUD created must be under "HUDs" and must be inherited from "HUDView".



UICanvas
Panels
Popups
HUDs
ExampleHud1

Inspector | Lighting | Services

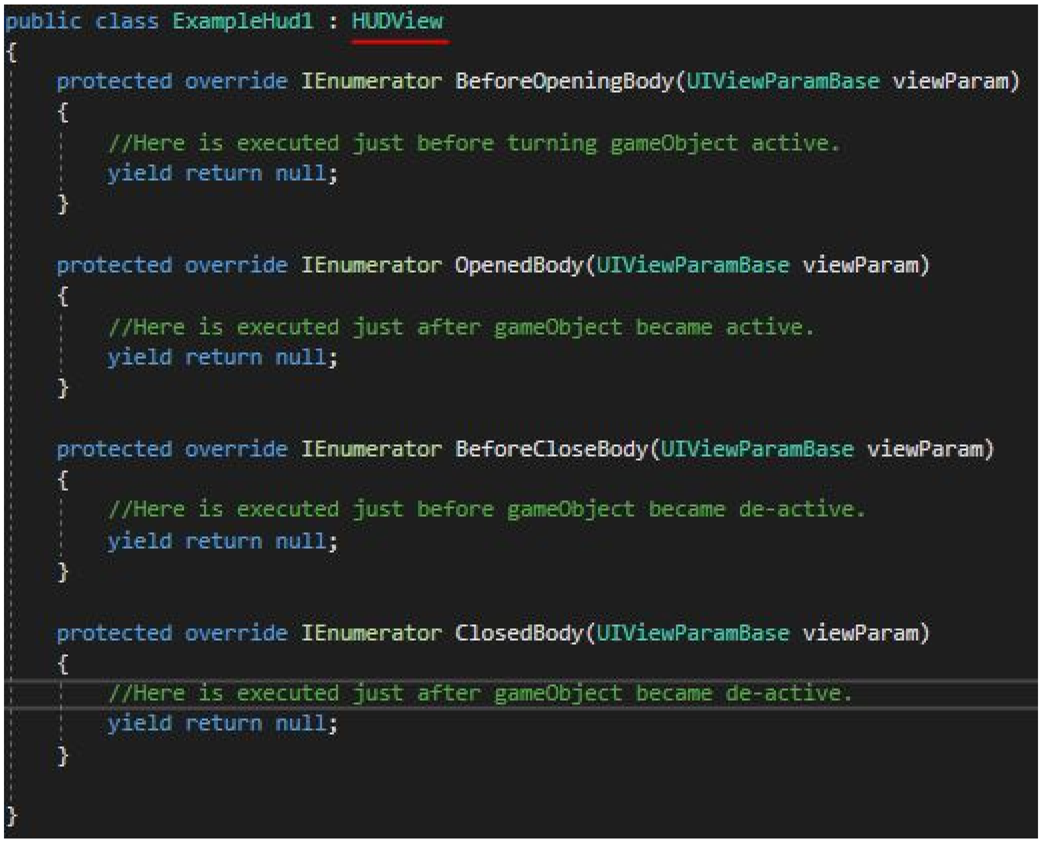
ExampleHud1 | Static
Tag: Untagged | Layer: UI

Rect Transform
stretch
Left: 0 | Top: 0 | Pos Z: 0
Right: 0 | Bottom: 0
Anchors
Min: X 0 | Y 0
Max: X 1 | Y 1
Pivot: X 0,5 | Y 0,5
Rotation: X 0 | Y 0 | Z 0
Scale: X 1 | Y 1 | Z 1

Example Hud 1 (Script)
Script: ExampleHud1

-Image 4-

As you can see in the “image 4” a created HUD is in the “HUDs” and has a script named “ExampleHud1” (naming is not mandatory it could be anything, yes even “cat.cs” too…) look like this (Image 5) :



```
public class ExampleHud1 : HUDView
{
    protected override IEnumerator BeforeOpeningBody(UIViewParamBase viewParam)
    {
        //Here is executed just before turning gameObject active.
        yield return null;
    }

    protected override IEnumerator OpenedBody(UIViewParamBase viewParam)
    {
        //Here is executed just after gameObject became active.
        yield return null;
    }

    protected override IEnumerator BeforeCloseBody(UIViewParamBase viewParam)
    {
        //Here is executed just before gameObject became de-active.
        yield return null;
    }

    protected override IEnumerator ClosedBody(UIViewParamBase viewParam)
    {
        //Here is executed just after gameObject became de-active.
        yield return null;
    }

}
```

-Image 5-

<u>The most important</u> part here is that; our HUDView(ExampleHud1) **<u>is derived from</u>** "**<u>HUDView</u>**". The convention is the same for panels and pop-ups too. If this was ExamplePopup1 for example, it would be derived from **<u>PopupView</u>**; or for ExamplePanel1, from **<u>PanelView</u>**.

After deriving your script, the base classes will force you to implement "BeforeOpeningBody", "OpenedBody", "BeforeCloseBody" and "ClosedBody". You can use any advantage of Coroutines in these scoops, you can wait something to happen or a while for example, you can move UI object or change them in a time directly or with your favorite tween… it just depends on how you want to do, what you want to do.

1. BeforeOpenedBody
   This scoop is executed before opening the FRUI element(Panel, Popup or HUD). Best place for resetting items in the element. For example, you have a Text component with a value but you want to show it as blank, you can set it here because no one will see hence the gameObject is not activated yet. You can reset your animations here too.

2. OpenedBody
   This scoop is executed after opening the FRUI element, that means the gameObject is activated now. You can start your animations here. For example, if you want your pop-up to really pop-up(I mean scaling up in time) you can do it here.

3. BeforeClosedBody
   This scoop is executed before closing the FRUI element, that means gameObject is still active. You can use here for disappearing animations for example.

4. ClosedBody
   This scoop is executed after closing the FRUI element that means the gameObject is deactivated now. You can use here if you have a specific action bounded to this element's appearance like resetting items' positions in it.

How to control navigation?

To control element navigation, you should reach related managers. For example, if you want to change a panel you can do something like this :

```
FRUI.PanelManager.ChangePanel(
        new PanelViewParamBase(typeof(MyPanelView))
        );
```

This will close the current panel and open the “MyPanelView” panel. Other methods are described below:

PanelManager Has :

- void ChangePanel(PanelParams)
  Example usage:

  ```
  FRUI.PanelManager.ChangePanel(
  new PanelViewParamBase(typeof(MyPanelView))
  );
  ```

  Changes the current panel to given panelview(in this case to “MyPanelView”).

- PanelView GetPanelView<T>()
  Example usage:

  ```
  FRUI.PanelManager.GetPanelView<MyPanelView>();
  ```

  Returns the instance object of given PanelViewType (in this case returns an instance of the “MyPanelView”).

PopupManager Has :

- bool IsInQueue<T>()
  Example usage:

  ```
  FRUI.PopupManager.IsInQueue<MyPopupView>();
  ```

  Returns true if given popup type is in the queue of will-shows(in this case returns true if MyPopupView is queued to show).

- void RequestPopup(PopupViewParamBase)
  Example usage:

  Non-custom parametered:

  ```
  FRUI.PopupManager.RequestPopup(new
  PopupViewParamBase(typeof(MyPopupView)));
  ```

  This usage just requests a popup of given type without any parameter passed; instead of using this, when a popup will be requested without any parameter using RequestPopup<MyPopupView>() will be more correct.

  Custom parametered:

  ```
  FRUI.PopupManager.RequestPopup(new
  MyPopupParameters(fancySpriteToShow));
  ```

  This usage requests a popup with custom parameters passed; custom parameters will be explained in the following sections.

- void RequestPopup<T>()
  Example usage:

  ```
  FRUI.PopupManager.RequestPopup<MyPopupView>()
  ```

  This usage just requests a popup of given type without any parameter passed(in this case MyPopupView is requested).

- void CloseActivePopup()
  Example usage:

  ```
  FRUI.PopupManager.CloseActivePopup()
  ```

  This usage closes the active popup.

- void CloseActivePopupAndDo(Action)
  Example usage:

  ```
  FRUI.PopupManager.CloseActivePopupAndDo(SomeFunction)
  ```

  This usage closes the active popup and executes given function.

- PopupView GetPopupView<T>()
  Example usage:

  ```
  FRUI.PopupManager.GetPopupView<MyPopupView>()
  ```

  Returns the instance of given PopupView type(in this case the instance of MyPopupView is returned).

HUDManager Has :

- HUDView GetHudView<T>()
  Example usage:

  ```
  FRUI.HUDManager.GetHudView<MyHUDView>()
  ```

  Returns the instance of given HUDView type(in this case the instance of MyHUDView is returned).

- void OpenHud<T>()
  Example usage:

  ```
  FRUI.HUDManager.OpenHud<MyHUDView>()
  ```

  Opens the given HUDView in type(in this case the MyHUDView is opened).

- void OpenHud(HUDViewParamBase)
  Example usage:

  Non-custom parameters:

  ```
  FRUI.HUDManager.OpenHud(new
  HUDViewParamBase(typeof(MyHUDView)))
  ```

  Opens the given HUDView without any custom parameters (in this case the MyHUDView is opened). Using OpenHUD<T>() will be more correct.

Custom parameters:

```
FRUI.HUDManager.OpenHud(new
MyHUDViewParams(somethingCool))
```

Opens the given HUDView without any custom parameters (in this case the MyHUDView is opened). Using OpenHud<T>() will be more correct.

- void CloseAllOpenHuds()
  Example usage:

  ```
  FRUI.HUDManager.CloseAllOpenHuds()
  ```

  Closes the all open HUDs.

- void CloseHud<T>()
  Example usage:

  ```
  FRUI.HUDManager.CloseAllOpenHuds()
  ```

  Closes the HUDView of the given type.

- void CloseAllOpenHudsExcept<T,T1,T2..>()
  Example usage:

  ```
  FRUI.HUDManager.CloseAllOpenHudsExcept<MyHUDView>()
  ```

  Closes the all HUDViews except the given types.

Useful properties and events

PanelManager Has :

- static event Action<Type> PanelChanged

  Fired when panel changing completed with a Type parameter which represents the opened PanelView type.

- static PanelView ActivePanelView

  Returns the currently active PanelView instance.

PopupManager Has :

- static event Action<PopupView> PopupOpening

  Fired when a PopupView is started opening with the parameter which references to the opening PanelView instance.

- static event Action<PopupView> PopupOpened

  Fired when a PopupView is completed opening with the parameter which references to the opened PanelView instance.

- static event Action<PopupView> PopupClosing

  Fired when a PopupView is started closing with the parameter which references to the closing PanelView instance.

- static event Action<PopupView> PopupClosed

  Fired when a PopupView is completed closing with the parameter which references to the closed PanelView instance.

- static PopupView ActivePopupView

  Returns the currently active PopupView instance if there is otherwise null.

HUDManager Has :

- static event Action<HUDView> HudOpening

  Similar to the PopupManager’s

- static event Action<HUDView> HudOpened

  Similar to the PopupManager’s

- static event Action<HUDView> HudClosing

  Similar to the PopupManager’s

- static event Action<HUDView> HudClosed

  Similar to the PopupManager’s

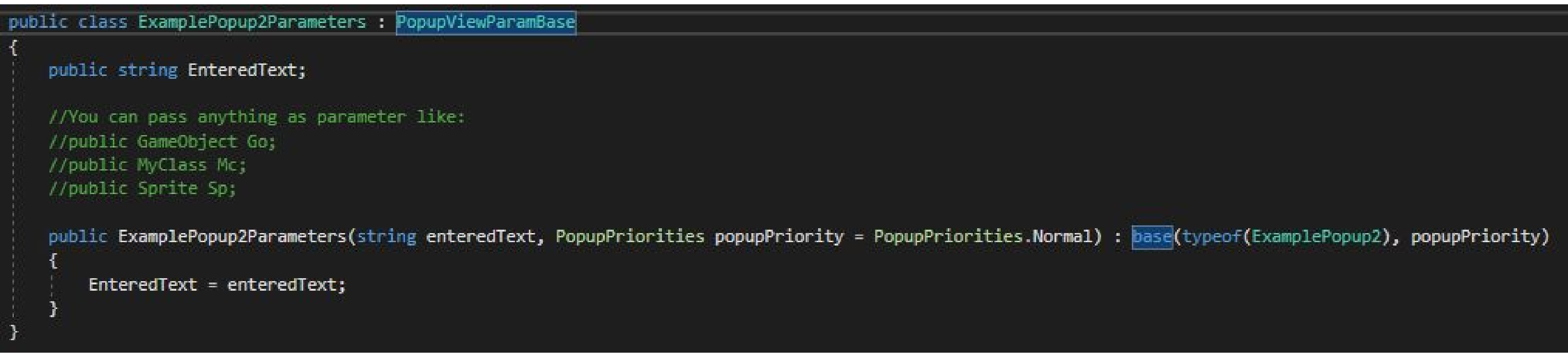
- readonly List<HUDView> ActiveHudViews = new List<HUDView>();

  Returns a read-only list of active HudViews.

How to pass custom parameters?

To pass your parameters to Panels, Popups and HUDs the same technique is used - “Extending the ParameterBase”.

This means you need to extend PanelViewParamsBase to pass custom parameters to a Panel, PopupViewParamsBase for Popups, HUDViewParamsBase for HUDs



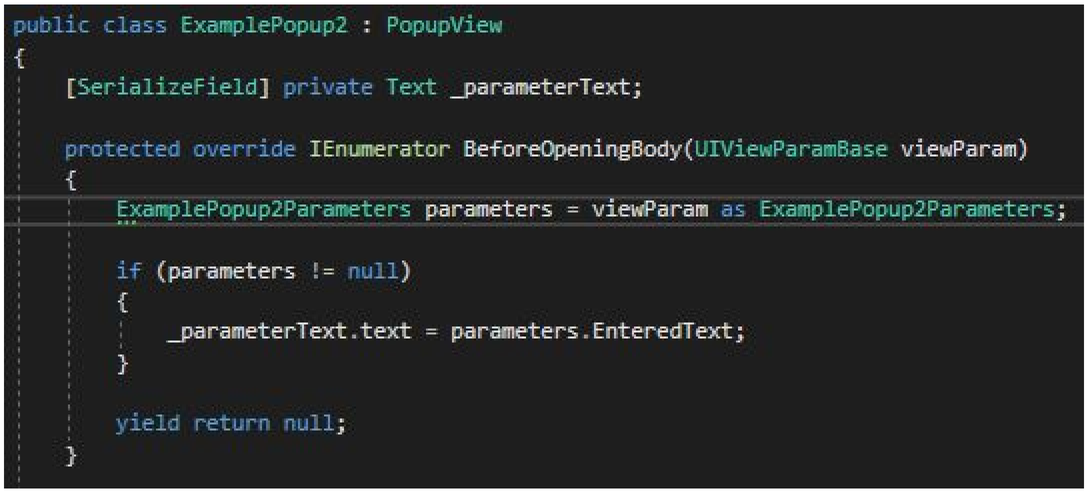
```
public class ExamplePopup2Parameters : PopupViewParamBase
{
    public string EnteredText;

    //You can pass anything as parameter like:
    //public GameObject Go;
    //public MyClass Mc;
    //public Sprite Sp;

    public ExamplePopup2Parameters(string enteredText, PopupPriorities popupPriority = PopupPriorities.Normal) : base(typeof(ExamplePopup2), popupPriority)
    {
        EnteredText = enteredText;
    }
}
```

-Image 6-

For the sake of consistency and reliability, I always use one-to-one parameterizing like this example.

As you can see in the “image 6” ExamplePopup2Parameters class is inherited from PopupViewParamBase for using in popups. It could be used in any popup if popup type is given as a parameter instead of directly passing a specific PopupView type to the base constructor like in “image 6”. Any object can be passed as parameter.

How to pass use custom parameters?



```
public class ExamplePopup2 : PopupView
{
    [SerializeField] private Text _parameterText;

    protected override IEnumerator BeforeOpeningBody(UIViewParamBase viewParam)
    {
        ExamplePopup2Parameters parameters = viewParam as ExamplePopup2Parameters;

        if (parameters != null)
        {
            _parameterText.text = parameters.EnteredText;
        }

        yield return null;
    }
```

-Image 6-

As you can see in the “image 6”, passed parameters are handled by casting to the expected type. That is why one-to-one parameterizing is more useful. Usage is similar for PanelsViews and HUDViews too.

Extras

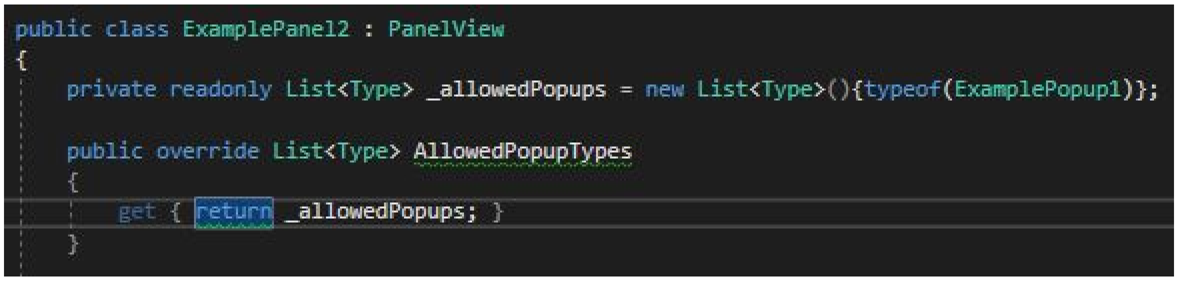
- **You can assign priorities to popups.**

  Via passing priority with PopupViewParams you can assign a priority to a popup. There are 5 priorities (VeryHigh, High, Normal, Low, VeryLow) you can assign. The popups will open according to their priorities

- **You can restrict popups to open panel-wise**

  Via providing a list to your PanelView, you can allow specific popup types and disallow others easily.

  Example :



```
public class ExamplePanel2 : PanelView
{
    private readonly List<Type> _allowedPopups = new List<Type>(){typeof(ExamplePopup1)};

    public override List<Type> AllowedPopupTypes
    {
        get { return _allowedPopups; }
    }
```

-Image 7-

As you can see in “image 7“; if you provide a list of popups in ExamplePanel2 for example, only ExamplePopup1 will be opened in ExamplePanel2 and if requested, disallowed popups will appear in the first panel they can.

Final words

If you have any trouble (bugs or bits of advice) feel free to mail to admin@btcoskuner.com and please rate the product if you liked it!