### 

Lucine.UISystem

**UI manager**



Lucine.UISystem

## Introduction

This documents explains how the Lucine.UISystem works.

## Some definition

### Screens

A screen is the base class for all UISystem. A screen is something that need to be displayed. Ideally it is a GameObject with all the needed children to display something that may be interactable or not. This root game objects has a script on it that inherit from UIScreenController<>. It is responsible to handle interactions on these elements. A Screen has parameters that are inherited from UIScreenParameters and contains in and out transition that are inherited from UITransition to handle the way they appears and disappear when asked by the user. A Screen is supported on a layer.

Screens have an unique id (the name of the GameObject supporting it). This id will be used to indicate to which screen orders are given. Screens keep trace of their visibility. A screen implements also 4 callback on which other can register. OnInTransitionFinished, OnOutTransitionFinished, OnCloseRequest, OnScreenDestroyed. All these information are implemented on screen using the IUIScreenController Interface.

### Panels

Panels are the simple type of screen, it is aimed to support information that will be displayed with no interaction, such as life counter, what you have in hand and so on. It is inherited from UIScreenController with UIPanelParameters. Panels can be showed or hidden. Panels are to be stored in UIPanelLayer hierarchy. There are not supposed to be interactive but nothing impeach you to make it so. They are always on the position of the hierarchy you defined. If you want to change them in the hierarchy of the panel layer, you have to do it yourself.

### Windows

Window are inherited from UIScreenControllers with UIWindowParameters. Windows can be Opened and Closed. When opened a windows always go to the top of other displayed windows. Windows have basic parameters that tells if it should be closed when loosing foreground and if they are popup. If they are popup they will set to front when opened and a BlackBackground GameObject (prefab furnished in sample) will be placed just below it in order to make everything below grayed.. Window are all stored in UIWindowLayer Hierarchy. Currently there’s no history management, but UIWindowLayer may be used to implement that kind of features if needed.

UIWindowLayer if front of UIPanelLayer by specification. UIWindowLayer and UIPanelLayer are specialization of UILayerController.

We never speak to screens directly (even if we could). To Open/Close windows and Show/Hide panels we use the UIController. UIController is the manager of UISystem. It requires a Canvas and a graphics raycaster. UIController owns two layers reference, the UIPanelsLayer and the UIWindowsLayer. It finds them automatically in its children on initialization. UIController is also responsible to disable interaction with all Unity UI elements, so it need to register to some events to be notified when interaction need to be disabled or enabled. UIController script need to be called before everything on startup to initialize everything (UIPanelsLayer & UIWindowsLayer).

### Parameters

UIScreen are templated with parameters, UIWindow are UIScreen with UIWindowParameters, UIPanel are UIScreen with UIPanelParameters. Right now only UIWindow defines parameters specific to windows. You will have to define specific UIScreen types with their appropriate UIParameters inherited from UIScreen, UIWindow or UIPanels while developing the controller for each screen of your interface.

### Events

UIController needs to know when it needs to disable or enable Unity UI events. In UISystem windows disable interaction during opening and closing transitions. The window layer is responsible to inform the listeners of these events that a windows is closing or opening. That’s why UIWindowsLayer expose two Action : DisableInteractionRequest and EnableInteractionRequest that UIController will listen to do the job.

The UIWindowsLayer itself need to know when screen start to appear and disappear to handle this case. That’s why it register to screen Actions : OnInAnimationFinished and OnOutAnimationFinished to do the job. It also registers to screen OnCloseRequest action to handle the case where a windows close is asked by the window itself (press a button for instance, public UnityClose function of UIWindow that send the event could be linked in button event trigger)

UIController

We’re going through him to do things

OpenWindow/CloseWindow => this will be asked to be done by the UIWindowLayer

ShowPanel/HidePanel => this will be done by the UIPanelLayer

Can disable/enable user interaction (raycaster of the main canvas). This behaviour should be asked by UIWindowLayer when a window is blocking window (such as a popup). It could be done by registering to an event, OnBlockScreenEnable, OnBlockScreenDisable

Can give the Main Canvas and the render camera. This camera should only display UI layer

This object root should be conserved when loading scene that’s why it is a Singleton

Make it a prefab

On initialization the UIController finds in his children the UIPanelLayerController and the UIWindowLayerController and initialize them

The UIController can close the current window asking to the UIWindowLayer to do it. So we can ask to close the current window.

We can ask if a panel or window is open

We can maybe have a HideAll function to hide all panels and windows

HideAllPanels only

CloseAllWindows only

UILAYER<TScreenController>

Registered screens

Show/Hide screen with or without UIParameters

Initialize created the registered screens

Register/unregister screen

IsScreenRegistred

Subscribe to Screen callback OnScreenDestroyed to remove register screens

UIWindowLayer = UILayer<UIWindowController>

Current window is stored here

List of active transition

Function to know if a transition is running

ShowScreen with or without UIparameters

HideScreen with or without UIparameters

HideAll close all windows registered to this layer

CloseCurrentWindow close the current window

Should register/unregister to UIScreen OnInOutTransitionFinished to store/remove currenttransition (transition are started by UIScreen with showscreen/hidescreen methods)

Should register/unregister to UIScreen OnCloseRequest to be able to handle to window close request sent by the UIScreen

UIPanelLayer = UILayer<UIPanelController>

ShowScreen with or without UIparameters

HideScreen with or without UIparameters

IsPanelVisible

UIScreen<TParameters>

Show

Play UITransition when showing

Call intransitiondone when finish in transition

Hide

Play UITransition when closing

Call outtransitiondone when finish out transition

Callback for Transition In done, callback for transition out done, callback for close request, callback for screen destroy

UIWindow = UIScreen<UIWindowParameters>

Hide window when foreground is lost ?

Is Popup indicates that should be drawn before all with dark screen

Callback closerequest when Close function called by Unity events is called (UnityClose)

UIPanel = UIScreen<UIPanelParameters>

Just like a UIScreen right now may think about some specialization later

UITransition

Base class for UITransition

These are monobehaviour with only one function to do wanted animation.

Parameter of this function is the transform to move, and a callback to call when finished

The Sample

The sample shows a main window with an input field and 3 buttons.

One button to show a first panel, another button to show a second panel, then a quit button that when clicked open a popup to confirm that you want to exit or cancel.

When a panel is open the corresponding button in the main window should be disabled to avoid to click one more time.

When a panel is closed, the corresponding button in the main window should be enabled to permit to open the panel once again.

When clicking on the quit button of the main window, the starter GameObject should kill the application.

When click on cancel the dialog box should close and enable background windows to become active once again.







