UI DRAWER

DOOZY UI
COMPLETE
UI MANAGEMENT
SYSTEM

Contents

Introduction

UIDrawer is a complex UI component for Unity. It is made up of native Unity components that have been set up in a controlled manner to behave like a Navigation Drawer that can be opened/closed from/to the Left/Right/Up/Down side of the screen.

The UIDrawer will work with any other asset that uses Unity's native UI framework. It also integrates very well with DoozyUI, a complete UI management system for Unity.

What is DoozyUI?

DoozyUI is a complete UI management system for Unity. It manipulates native Unity components and takes full advantage of their intended usage. This assures maximum compatibility with uGUI, best performance and makes the entire system have a predictable behavior. Also, by working only with native components, the system will be compatible with any other asset that uses uGUI correctly.

Easy to use and understand, given the user has some basic knowledge of how Unity's native UI solution (uGUI) works, DoozyUI has flexible components that can be configured in a lot of ways. Functionality and design go hand in hand in order to offer a pleasant user experience (UX) while using the system.

Artists and designers can realize their creative vision without coding by creating blazing fast user interfaces with any animations they can imagine.

Programmers can add a powerful UI management system to their toolbox that can be interfaced with scripts or used alongside Playmaker for a code-free UI solution.

It comes with an UI Animator System that uses math-based tweens creating reliable resolution independent animations and an Orientation Manager that allows anyone to create views for both portrait and landscape modes and not have to worry about which view should be active.

Eases the usage of Particle Systems within the UI layers and has a quick workflow for creating modal windows, named UI Notifications.

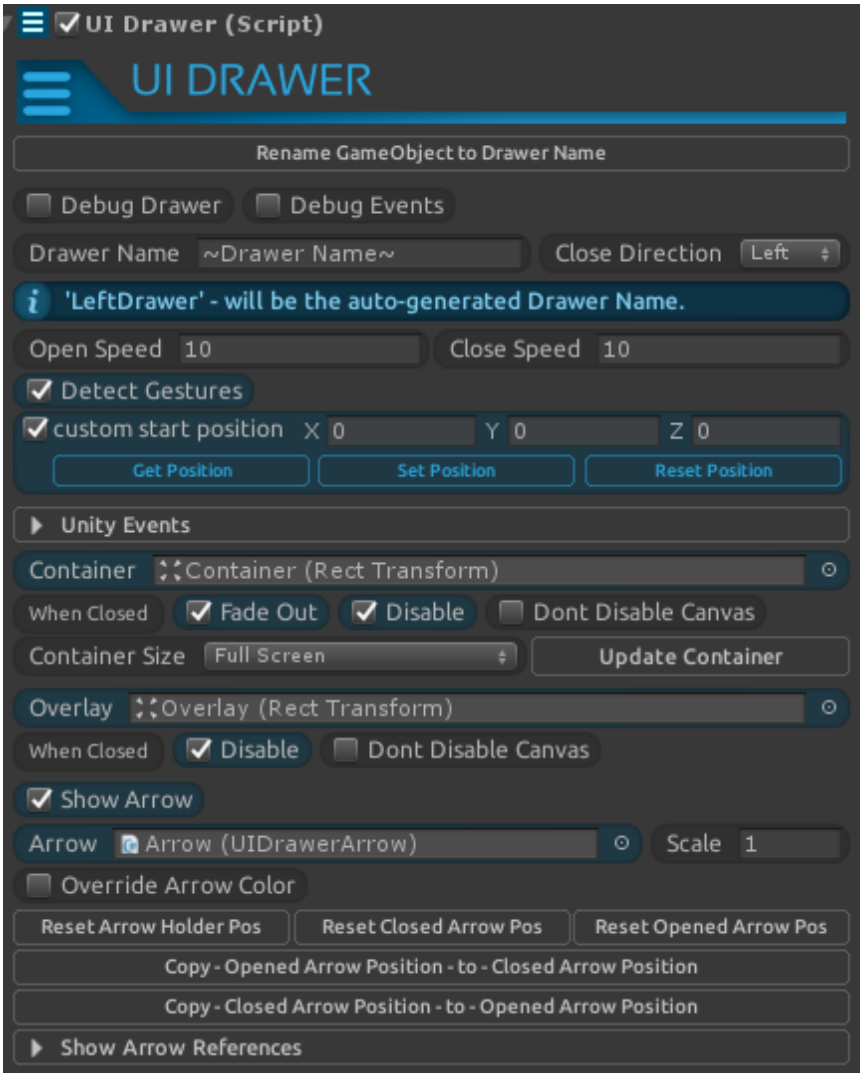
It supports all platforms and is extensively optimized to minimize its memory usage. Full C# source code, dedicated support and YouTube tutorials are available to help anyone understand and then master the system.

Quick Start

To start using the UIDrawer in your project you need to do the following:

1. Import the UIDrawer asset from the Unity Asset Store
2. Done! 😊
3. (optional) Import DoozyUI from the Unity Asset Store → to activate the DoozyUI integration

UIDrawer



The UIDrawer is a complex component that reacts to swipe gestures. It can be opened or closed via a drag performed either with a mouse or a finger, or by using buttons.

Rename GameObject to Drawer Name: renames the gameObject that this UIDrawer is attached to 'UID – drawer name'

Debug Drawer: prints debug messages related to opening and closing of this UIDrawer
Debug Events: prints debug messages related to all the UnityEvents this UIDrawer has.

To be able to identify the UIDrawers, the system uses drawer names in order to trigger the Open or the Close animations.

Drawer Name: the drawer name
Close Direction: the drawer position when closed (this also affects what gesture opens/closes the drawer)
Open Speed: open drawer animation speed
Close Speed: close drawer animation speed

Detect Gestures: enables/disables the gesture detectors of this drawer. If disabled, this drawer will no longer react to gestures. Useful if you plan on opening/closing the drawer via a button or a script.

(use) custom start position: should this UIDrawer slide from or go to a set custom position. Default is set to true
custom start (anchored) position: anchored position (used by the UI) that this UIDrawer uses to calculate its Open and Close animations; if modified at runtime, it allows you to change the animations show and hide locations; this is also useful when designing your UI as you don't need to stack UIDrawers over another and it works as follows:

- create an UIDrawer in the position you want it to be visible at runtime
- enable the custom start position
- click the Get button to copy the current RectTransform anchored position values to the custom start position
- drag and drop the newly created UIDrawer anywhere in the scene
- after performing the steps, at runtime, the UIDrawer will 'snap' to the custom start position for Open or Close animations

Unity Events
OnDrawerOpened: UnityEvent invoked when the drawer opened
OnDrawerIsOpening: UnityEvent invoked when the drawer started the open animation. This event does not get triggered when the drawer is being dragged

OnDrawerClosed: UnityEvent invoked when the drawer closed
OnDrawerIsClosing: UnityEvent invoked when the drawer started the close animations. This event does not get triggered when the drawer is being dragged

OnDrawerBeginDrag: UnityEvent invoked when the drawer started being dragged
OnDrawerEndDrag: UnityEvent invoked when the drawer stopped being dragged

Container: reference to the drawer's container. This is the part that gets animated in (when opening) and out (when closing). This also contains all the drawer's contents (like texts, buttons, scrollers, layout groups...)

When Closed (container)

Fade Out: determines if the container should fade out when closed. This controls the fade in (when opening) and fade out (when closing) animations for the container (and its contents)
Disable: disables this UIDrawer's container when it is not visible (it is closed) by setting its active state to false. Use this only if you have scripts that you need to disable. Otherwise you don't need it as the system handles the drawcalls in an efficient manner.
Don't Disable Canvas: this will disable the optimization that disables the container's Canvas and GraphicRaycaster when the UIDrawer is closed. Do not enable this unless you know what you are doing as, when set to TRUE, this will increase your draw calls.

Container Size: the container's size: FullScreen / PercentageOfScreen / FixedSize
Percentage of Screen: the container's percentage of screen size if the containerSize is set to PercentageOfScreen
Minimum Size: the container's minimum size of screen size if the containerSize is set to PercentageOfScreen
Fixed Size: the container's fixed size if the containerSize is set to FixedSize
Update Container: updates the container to the current settings (close direction, container size, and so on…)

Overlay: Reference to the gameObject that is used as an overlay. It should have the following components attached: RectTransform, Canvas, GraphicRaycaster, CanvasGroup and Image.

When Closed (overlay)

Disable: disables this UIDrawer's overlay when it is not visible (it is closed) by setting its active state to false. Use this only if you have scripts that you need to disable. Otherwise you don't need it as the system handles the drawcalls in an efficient manner.
Don't Disable Canvas: this will disable the optimization that disables the overlay's Canvas and GraphicRaycaster when the UIDrawer is closed. Do not enable this unless you know what you are doing as, when set to TRUE, this will increase your draw calls.

Show Arrow: variable used to determine if the arrow should be shown or not
Arrow: reference to the arrow that gets animated when the drawer is in transition or dragged
(arrow) Scale: scale variable that overrides the scale of the arrow at runtime. This will set the localScale values of the arrow. (it will only override localScale.x and localScale.y, as localScale.z is useless for an UI)
Override Arrow Color: if set to true, the arrow color will get interpolated between arrowColorWhenClosed and arrowColorWhenOpened colors
(arrow color when) Closed: color variable that overrides the color of the arrow at runtime. This will set the Image color of the arrow to the set values when the drawer is closed
(arrow color when) Opened: color variable that overrides the color of the arrow at runtime. This will set the Image color of the arrow to the set values when the drawer is opened

Reset Arrow Holder Pos: resets the arrow holder position to the default value for the set close direction
Reset Closed Arrow Pos: resets the closed arrow position to the default value for the set close direction
Reset Opened Arrow Pos: reset the opened arrow position to the default value for the set close direction
Copy – Opened Arrow Position – to – Closed Arrow Position: moves the Closed Arrow Position to the Opened Arrow Position
Copy – Closed Arrow Position – to – Opened Arrow Position: moves the Opened Arrow Position to the Closed Arrow Position

Show Arrow References: references to all the positions used by the arrow for each close direction and state (opened/closed)

Fields

Name	*Description*
arrow	Reference to the arrow that gets animated when the drawer is in transition or dragged.
arrowColorWhenClosed	Color variable that overrides the color of the arrow at runtime. This will set the Image color of the arrow to the set values when the drawer is closed.
arrowColorWhenOpened	Color variable that overrides the color of the arrow at runtime. This will set the Image color of the arrow to the set values when the drawer is opened.
arrowContainer	Internal reference to the arrowContainer. This is used to sync the arrow position with the container position. Makes for a nicer animation.
arrowScale	Scale variable that overrides the scale of the arrow at runtime. This will set the localScale values of the arrow. (it will only override localScale.x and localScale.y, as localScale.z is useless for an UI).
closeSpeed	Close drawer animation speed.
container	Reference to the drawer's container. This is the part that gets animated in (when opening) and out (when closing). This also contains all of the drawer's contents (like texts, buttons, scrollers, layout groups...).
containerFixedSize	The container fixed size if the containerSize is set to FixedSize.
containerMinimumSize	The container minimum size of screen size if the containerSize is set to PercentageOfScreen.

containerPercentageOfScreenSize	The container percentage of screen size if the containerSize is set to PercentageOfScreen.
containerSize	The container size: FullScreen / PercentageOfScreen / FixedSize
customStartAnchoredPosition	The custom anchored position that this UIElement slides from or goes to when opening/closing. You can use this in code to cusomize on the fly this positon.
debugDrawer	Prints Debug Logs when the drawer is opened/closed.
debugEvents	Prints Debug Logs when the drawer invokes any of its UnityEvents.
DEFAULT_DRAWER_NAME	This is the default drawer name for any newly created drawer.
detectGestures	Enables/Disables the gesture detectors of this drawer. If disabled, this drawer will no longer react to gestures. Useful if you plan on opening/closing the drawer via a button or a script.
disableContainerWhenClosed	Disables this UIDrawer's container when it is not visible (it is closed) by setting it's active state to false. Use this only if you have scripts that you need to disable. Otherwise you don't need it as the system handles the drawcalls in an effecient manner.
disableOverlayWhenClosed	Disables this UIDrawer's overlay when it is not visible (it is closed) by setting it's active state to false. Use this only if you have scripts that you need to disable. Otherwise you don't need it as the system handles the drawcalls in an effecient manner.
dontDisableContainerCanvasWhenClosed	This will disable the optimization that disables the container's Canvas and GraphicRaycaster when the UIDrawer is closed. Do not enable this unless you know what you are doing as, when set to TRUE, this will increase your draw calls.
dontDisableOverlayCanvasWhenClosed	This will disable the optimization that disables the overlay's Canvas and GraphicRaycaster when the UIDrawer is closed. Do not enable this unless you know what you are doing as, when set to TRUE, this will increase your draw calls.
downDrawerArrowClosedPosition	Internal reference to the closed position where the arrow should move to if the drawer is set to close to the bottom (down) side of the screen.
downDrawerArrowHolder	Internal reference to the gameObject and position where the arrow should be parented and moved if the drawer is set to close to the botton (down) side of the screen.
downDrawerArrowOpenedPosition	Internal reference to the opened position where the arrow should move to if the drawer is set to close to the bottom (down) side of the screen.
drawerCloseDirection	The drawer position when closed (this also affects what gesture opens/closes the drawer).
drawerName	The name of this drawer. The name is important when opening or closing an UIDrawer using the static methods.
fadeOutContainerWhenClosed	Determines if the container should fade out when closed. This controls the fade in (when opening) and fade out (when closing) animations for the container (and its contents).
leftDrawerArrowClosedPosition	Internal reference to the closed position where the arrow should move to if the drawer is set to close to the left side of the screen.
leftDrawerArrowHolder	Internal reference to the gameObject and position where the arrow should be parented and moved if the drawer is set to close to the left side of the screen.
leftDrawerArrowOpenedPosition	Internal reference to the opened position where the arrow should move to if the drawer is set to close to the left side of the screen.
OnDrawerBeginDrag	UnityEvent invoked when the drawer started being dragged.
OnDrawerClosed	UnityEvent invoked when the drawer closed.
OnDrawerEndDrag	UnityEvent invoked when the drawer stopped being dragged.
OnDrawerIsClosing	UnityEvent invoked when the drawer started the close animations. This event does not get triggeres when the drawr is being dragged.
OnDrawerIsOpening	UnityEvent invoked when the drawer started the open animation. This event does not get triggered when the drawer is being dragged.
OnDrawerOpened	UnityEvent invoked when the drawer opened.
openSpeed	Open drawer animation speed.
overlay	Reference to the gameObject that is used as an overlay. It sohuld have the followin components attached: RectTransform, Canvas, GraphicRaycaster, CanvasGroup and Image.
overrideArrowColor	If set to true, the arrow color will get interpolated between arrowColorWhenClosed and arrowColorWhenOpened colors.
rightDrawerArrowClosedPosition	Internal reference to the closed position where the arrow should move to if the drawer is set to close to the right side of the screen.
rightDrawerArrowHolder	Internal reference to the gameObject and position where the arrow should be parented and moved if the drawer is set to close to the right side of the screen.
rightDrawerArrowOpenedPosition	Internal reference to the opened position where the arrow should move to if the drawer is set to close to the right side of the screen.

showArrow	Variable used to determine if the arrow should be shown or not.
upDrawerArrowClosedPosition	Internal reference to the closed position where the arrow should move to if the drawer is set to close to the top (up) side of the screen.
upDrawerArrowHolder	Internal reference to the gameObject and position where the arrow should be parented and moved if the drawer is set to close to the top (up) side of the screen.
upDrawerArrowOpenedPosition	Internal reference to the opened position where the arrow should move to if the drawer is set to close to the top (up) side of the screen.
useCustomStartAnchoredPosition	Should this UIDrawer slide from or go to a set custom position. Default is set to true.

Properties

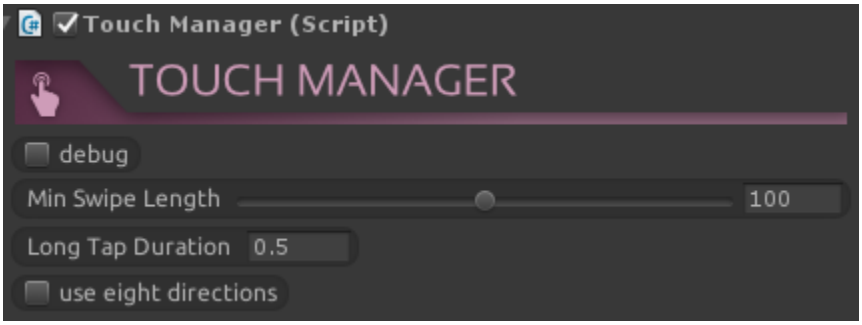
Name	*Description*
Canvas	Returns the Canvas component.
Closed	Returns true if the drawer is closed.
ContainerCanvas	Returns the Canvas component of the container.
ContainerCanvasGroup	Returns the CanvasGroup component of the container.
ContainerClosedAnchoredPosition	Returns the calculated value of the container when the drawer is closed.
ContainerGraphicRaycaster	Returns the GraphicRaycaster component of the container.
ContainerOpenedAnchoredPosition	Retruns hte initial position of the container. This is the position of the container when the drawer is open.
ContainerVelocity	Used by the DrawerArrow live animation.
CurerntDrawerState	Returns the current state of the drawer. Opened - IsOpening - Closed - IsClosing
DraggedDrawer	Gets the reference to the dragged UIDrawer.
DrawerDatabase	Returns a registry of all the registered UIDrawers.
HasOverlay	Returns true if an overlay has been references.
IsAnyDrawerOpened	Retruns true if there is an OpenedDrawer. There can be only one drawer opened at a time.
IsClosing	Returns true if the drawer is currently closing (the close animation is running).
IsDragged	Returns true if this drawer is currently begin dragged.
IsOpening	Returns true if the drawer is currently opening (the open animation is running).
IsVisible	Retruns true if the drawer is open, thus the Visibility value is 1.
Opened	Returns true if the drawer is opened.
OpenedDrawer	Returns the reference to the currently open drawer. If no drawer is opened, it will return null. There can be only one drawer opened at a time.
OverlayCanvas	Returns the Canvas component of the overlay.
OverlayCanvasGroup	Returns the CanvasGroup component of the overlay.
OverlayGraphicRaycaster	Returns the GraphicRaycaster component of the overlay.
RectTransform	Returns the RectTransform component.
TouchManager	Returns the TouchManager reference.
Visibility	Returns the open state percentage the drawer (how much of the container is visible on screen). (0 = 0% = Closed and 1 = 100% = Open)

Methods

Name	*Description*
Close(bool)	Closes the drawer.
Close(string, bool)	Closes the specified drawer name.
CopyRectTransform(RectTransform, RectTransform)	Copies a RectTransform properties from the source to the target.
GetDrawer(string, bool)	Returns the UIDrawer reference to the drawer registered under the given drawerName. If no drawer is found, it will return null.
GetDrawerName()	Gets the name of the drawer. If the drawer name was auto-generated it will retrun 'drawerCloseDirection.ToString() + "Drawer"', otherwise it will retrun the drawerName. This method is used by the editor.
MatchRectTransform(RectTransform, RectTransform)	Matches the target rect transform to the source rect transform.
Open(string, bool)	Opens the specified drawer name.

Open(bool)	Opens the drawer.
RenameDrawer(string)	Renames the UIDrawer and also re-registers it to the DrawersDatabase.
ResetArrowHolder(RectTransform, SimpleSwipe)	Resets the arrow holder to the default position.
ResetClosedArrow(RectTransform, SimpleSwipe)	Resets the closed arrow position to its default value.
ResetOpenedArrow(RectTransform, SimpleSwipe)	Resets the opened arrow position to its default value.
UpdateArrowContainer()	Updates the arrow container, copying the container's RectTransfrom properties to itself.
UpdateContainer()	Updates the container to the set settings. This method is mostly used by the editor button to help setup and design the container.
UpdateContainerSize(float, float)	Updates the size of the container. Use this method only to set it to PercentageOfScreen.
UpdateContainerSize(float)	Updates the size of the container. Use this method only to set it to FixedSize.
UpdateContainerSize()	Updates the size of the container. Use this method only to set it to FullScreen.
UpdateDrawerCloseDirection(SimpleSwipe)	Updates the drawer's close direction

Touch Manager



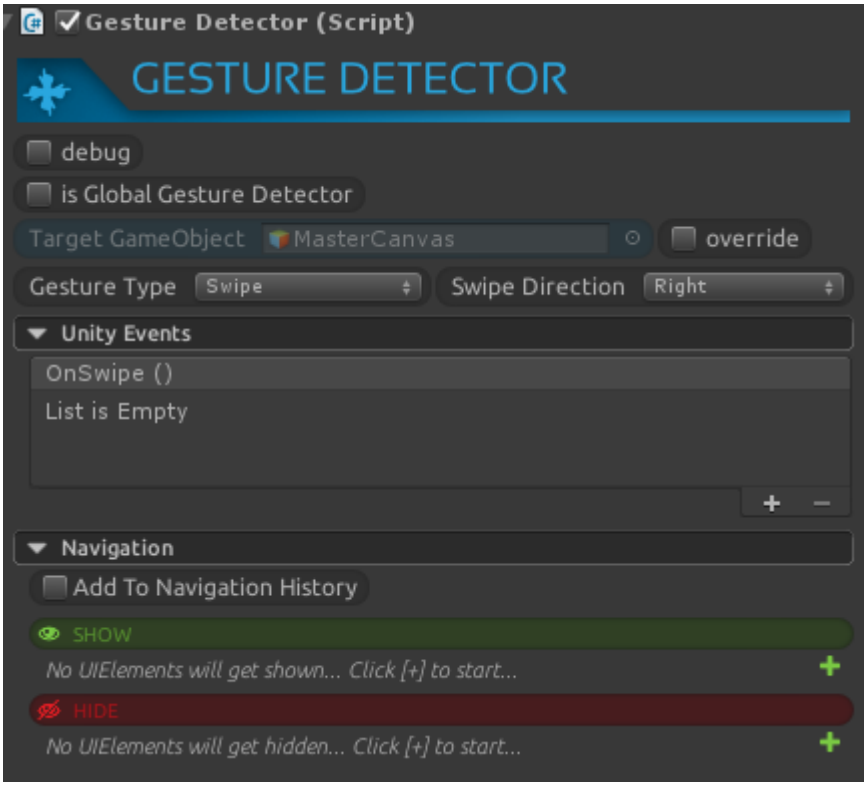
The Touch Manager is responsible for detecting touch gestures made on the target device. It will also simulate them, if in the Editor, by reacting to the mouse gestures.

It can detect the following gestures: Tap, Long Tap and Swipe

Tap Gesture: *is similar to a click, but it can be detected on any UI component, 2D object or 3D object.*
Long Tap Gesture: *is similar to a long click (or long press), but it can be detected on any UI component, 2D object or 3D object*
Swipe: *it can be detected in 4 directions (Left, Right, Up and Down) or it can be detected in 8 directions (Left, UpLeft, Up, UpRight, Right, DownRight, Down, DownLeft), on any UI component, 2D object or 3D object.*

debug: prints debug messages related to detected gestures at runtime
Min Swipe Length: the minimum swipe distance to be considered a swipe
Long Tap Duration: time period for a finger to be touching the target device, in order for the tap to be considered a long tap (long press)
use eight directions: sets if the TouchManager should detect swipes on eight or on four cardinal directions

Gesture Detector



The Gesture Detector is an extension to the TouchManager. The TouchManager is the one that detects the touches (the gestures), but the Gesture Detector is the one that reacts to them, depending on its settings.

To be clear, the TouchManager detects a gesture and the GestureDetector reacts to it. Because the listening for gestures needs to happen in the Update() method, by having only the TouchManager as the only listener and all the Gesture Detectors getting fired by the TouchManager, we have an efficient setup.

debug: prints debug messages related to detected gestures at runtime
is Global Gesture Detector: if true, it will trigger this Gesture Detector regardless of the targetGameObject
Target GameObject: only gestures performed on the target game object will trigger this Gesture Detector
override target: allows you to set another targetGameObject reference (in the Inspector); other than the gameObject this component is attached to
Gesture Type: the gesture type this Gesture Detector will react to
Swipe Direction: the swipe direction this Gesture Detector will react to; this works only if the gestureType is set to GestureType.Swipe

Only the 'active' UnityEvent is visible in the Inspector (in order to reduce clutter)
OnTap: UnityEvent invoked on tap
OnLongTap: UnityEvent invoked on long tap
OnSwipe: UnityEvent invoked on swipe (it has to be the proper swipe direction)

onTapAction: action invoked on tap
onLongTapAction: action invoked on long tap
onSwipeAction: action invoked on swipe (it has to be the proper swipe direction)

Available only if DoozyUI is installed
- and only if the UINavigation is enabled, the Navigation options will be available.
Add to Navigation History: if there are any UIElements set to be shown or hidden, the action will be added to the Navigation History
SHOW: a list of pairs of element category and element name that will get shown when this Gesture Detector gets triggered
HIDE: a list of pairs of element category and element name that will get hidden when this Gesture Detector gets triggered

Final Words

YouTube Channel

You can find all the videos related to DoozyUI, that were created by Doozy Entertainment, at youtube.com/c/DoozyEntertainment

Twitter Feed

The Twitter handle used by Doozy Entertainment is **'doozyplay'**. You can view our Twitter feed by going to twitter.com/doozyplay

Facebook Page

The Facebook page used by Doozy Entertainment is **'doozyentertainment'**. You can view our Facebook page by going to facebook.com/doozyentertainment

Help Center

The Help Center for DoozyUI can be found at doozyentertainment.zendesk.com.

There you will be able to access an online (searchable) **Documentation**, **Tutorials** and **FAQ** sections.

Also, there is a Community area where you can submit **Feature Requests** or open **General Discussion** topics at doozyentertainment.zendesk.com/hc/en-us/community/topics

Support Ticket

You can open a **support ticket** by sending an email to support@doozyentertainment.com