MODERN & CLEAN

UI PACK

Documentation - 1.0

Quick Navigation

Introduction

Modern & Clean UI pack has been made with a version of unity that is **2020.2.1f1** using the native Unity UI**,** you'll open the main scene (found in /scene/MainScene.unity) where you can find a menu that has 5 different tabs each one displaying many different features all available in the prefabs folder, the first tab contains a list of 20 different button styles animated with different colors and some even have a gradient script for customizable gradient colors, the second tab comes with animated buttons that has 12 different animations, the third tab is the one that contains the most features and it has 10 rows containing each one essential ui elements that are : toggle switches, custom input fields, sliders, range sliders, tooltips, hover tooltip, dropdown, selection sliders, toggle groups and notifications, the fourth tab contains 4 rows with different loading styles that are slider loadings and radial loadings both with a loopable animation, in the last fifth tab you'll find 3 different examples of interfaces made with this asset

Modern & Clean UI pack **requires TextMeshPro**

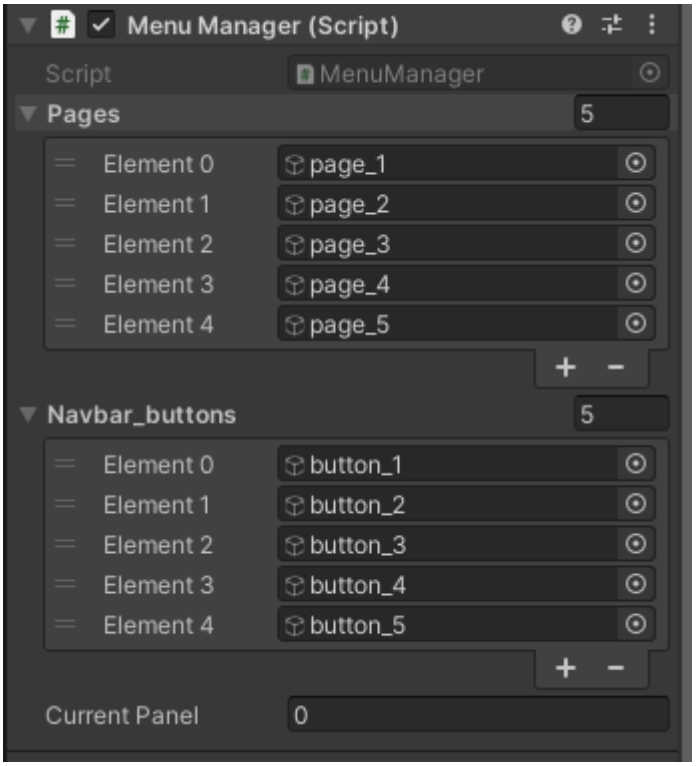
If you need help on anything regarding this asset go to the Contact section and feel free to ask me anything.

You can also view this documentation through this link

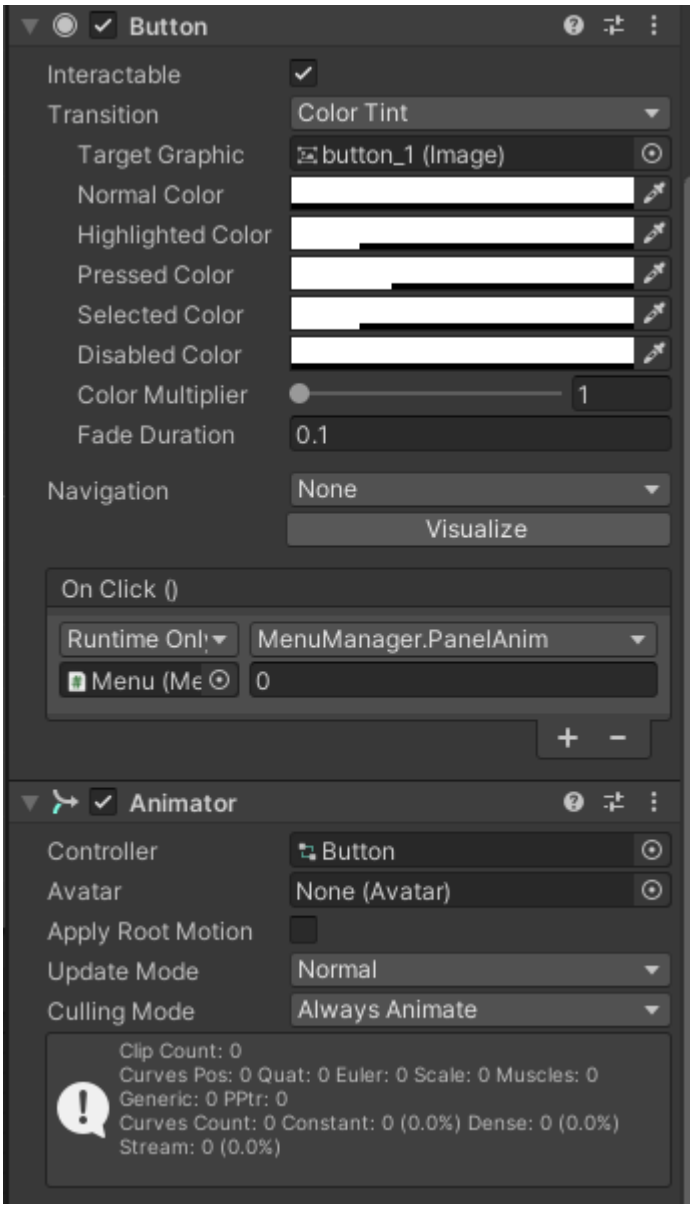
Elements and Prefabs

Menu Manager

To build a menu with pages and buttons that open them you can use Script/MenuManager.cs script, just drag the file under any GameObject then you will see 2 lists : **Pages** and **Navbar_Buttons**, enter the number of pages and buttons, after this add the pages and buttons prefabs like in this image

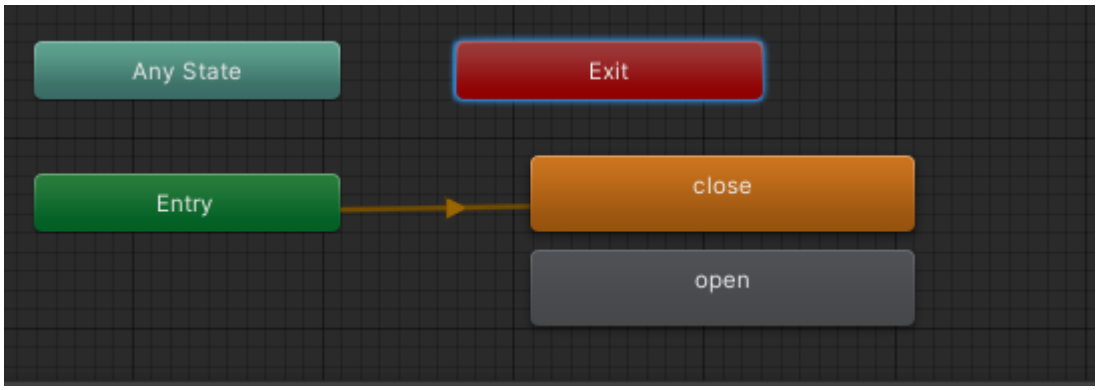


Each button to open the page must have an animator with 2 animations called “open” and “close” (you can take the animations i already made by copying them from the main scene) then a onclick event which is PanelAnim() that opens a page by its list index like this image below

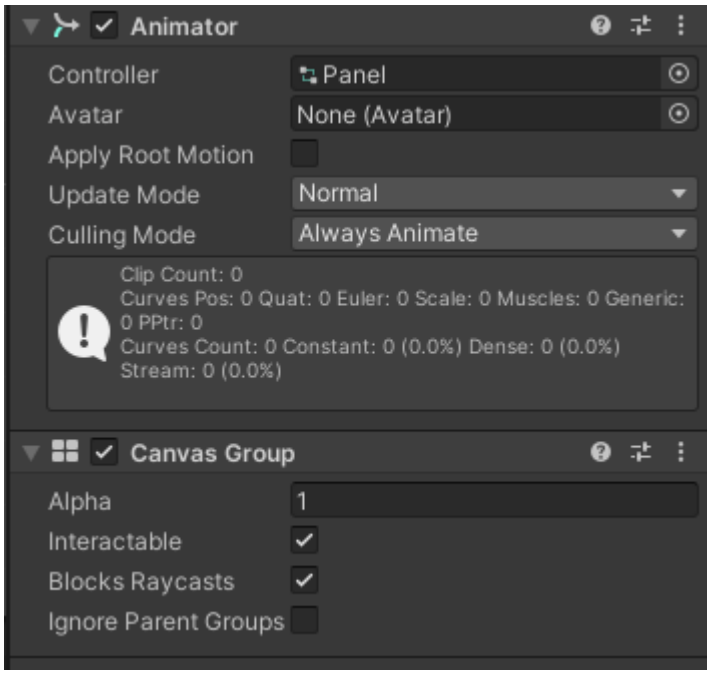


For example this button is going to open the page_1 because it has the index 0 on the MenuManager.Pages list, the last thing you'll have to set up is the animator for the pages that should contain 2 animations called "open" and "close" (you can also here take the animations i already made by copying them from the animator of the main scene or just customize them),

Animator :



Page inspector:



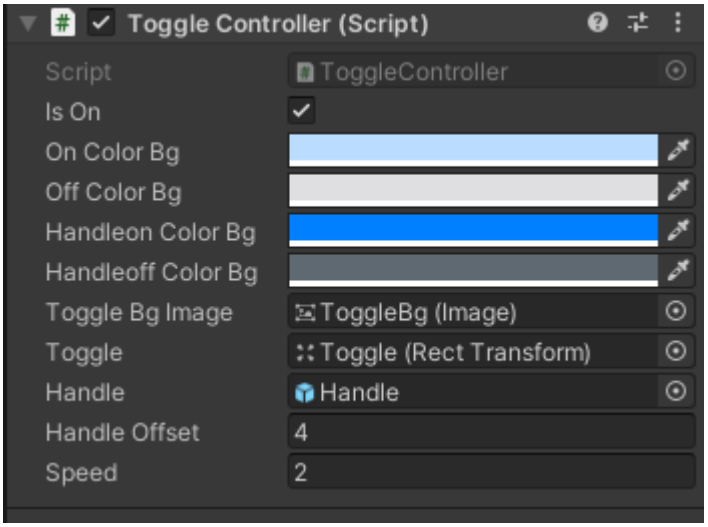
If you use the default Animator of the asset you **must** have the canvas group component on the page.

Switch

Switches are basically animated toggles, they don't depend on the default Unity UI package and you can customize them as you want, to access their values you can fetch the isOn variable on the ToggleController.cs script, like this example :

```
public ToggleController MySwitch; // Your switch

void function()
{
  bool status = MySwitch.isOn; //getting the value of the switch
  // You can also access or change other values
  MySwitch.speed = 3;
  MySwitch.onColorBg = Color.red;
}
```

you can also customize the colors as you like on the inspector.



You can find many different switch examples under the /Prefabs/Switches folder.

Input fields

Input fields work with the “TextMeshPro - Input field” component and are animated by a script (/Scripts/InputFieldAnimation.cs) that requires an animator with 2 animations called “active” and disable (you can copy the default animator on the main scene or just customize it or make a new animator with those 2 animations) and you can just interact them with TextMeshPro component.

Sliders

Sliders are made with the default native Unity UI package so you can interact with them like normal sliders.

Slider range

Slider range are a different type of sliders where you choose your values between a range of numbers, to make a slider range you’ll have to create a GameObject with the slider range script (found in /Scripts/RangeSliders.cs) and then fill all the needed values.

To fetch the values of the Slider range you just have to access the public values variables, like this example :

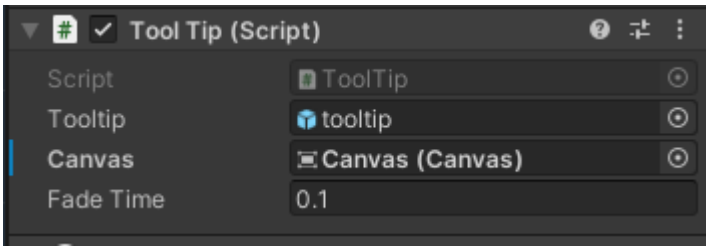
```
public RangeSlider rs; // Your range slider

void function()
{
  var max = rs.maxValue; // Access the maximum value
  var max = rs.maxValue; // Access the minimum value
}
```

Hover tooltip

The Hover tooltip script must be dragged on the latest layer of the button or component which is the layer over everything else or latest on the hierarchy of the parent, like in this image.

Hierarchy:

TooltipTrigger inspector :



where tooltip is the gameObject that is going to appear once the mouse goes over the TooltipTrigger, the tooltip must have the “Canvas Group” component.

Dropdown

Dropdowns are made with the TMP package, so you can just interact with them normally.

Selection sliders

Selection sliders are basically like a dropdown, they work with a script (/Script/HorizontalChoose.cs) that you have to drag on a GameObject with an animator that animates the transitions when you change values, you can take the prefabs under the folder /Prefabs/SelectionSliders and customize them as you like.

to access the value of the selection slider you can fetch the public variable called value, like in this example :

```
public HorizontalChoose hc; // Your selection slider
```

```
void function()
{
  var value = hc.value; // Access the selection slider value
  var index = hc.index; // Access the selection slider index
}
```

Toggle group

Toggle groups are made with the native Unity UI, the only thing that changes is that they have different styles using the sprites of this asset.

Notifications

Notifications can be customized and spawned as and when you like, notifications require 2 scripts which are “NotificationManager.cs” and “Notification.cs”. Each notification prefab requires the Notification.cs script where you’ll have to add a few variables which are : Title (from TMP), Description (again from TMP) and Icon (which is an Image). This Notification.cs script is needed if you want to generate different notifications with the NotificationManager.cs script.
You’ll have to drag the NotificationManager.cs on any GameObject and then fill the required variables which are : Notification prefab (that must contain the Notification.cs script i talked about previously), Title, Description and Icon (Sprite not Image).
Now you are ready to spawn the notification with the function

```
NotificationManager.SpawnNotification()
```

as i said before you can also customize the notifications as you like using the NotificationManager variables, like in this example :

```
public NotificationManager nm; // Your manager
public Sprite Bell; // Any sprite to change the icon
public GameObject Notification2; // if you want other animations

void function()
{
  nm.Icon = Bell;
  nm.Title = “title”
  nm.Description = “hey, this is a notification”
  // if you want to use a single animation just skip this line
  nm.NotificationPrefab = Notification2;

 nm.SpawnNotification(); // Spawns the notification
}
```

Buttons

All the buttons are made with the native Unity UI package the only difference is that they are animated and they have 20 different styles, some also have gradient colors done by a script located in /Scripts/UIGradient.cs, you can also customize them as you like

Edit animations

All the animations both for the buttons or any other object are done using the animator so if you want to change or edit anything you can do that using the Animator window.

Contact

you can feel free to contact me through these links below if you have any issues regarding this asset:

Email : yessou.rayan@gmail.com
Discord : R4yan#1337
Website : https://r4yan.cf
Youtube : youtube.com

Licence