Essential UI Pack

Version 1.1 for Unity – by Llama Software

Dependencies

To fully use this pack, please first install the free asset TextMeshPro from the Asset Store. It provides the most clean and clear text in Unity and comes with a ton of great features. The simple "Text" on the UI does not even compare.

If you do not want to use TextMeshPro, you can fairly easily update all of the prefabs and scripts to use "Text" instead of "TextMeshProUGUI", (also the dropdown and Input Fields used are from TextMeshPro) however it may take some time to do.

When using Unity 2018.2+ with Packages, sometimes there are issues with the Package version of TextMeshPro. I highly recommend using the one from the asset store instead of the package version.

What's Included

The following items are included in the Essential UI Pack

- Page Management System
    - No Coding Required to manage your UI Pages
    - Provided In/Out Animations
        - Fade
        - Slide (Left/Right/Top/Bottom)
        - Zoom
    - Customizable In/Out Animations per Page
    - Custom Events supported on PrePush, PostPush, PrePop, and PostPop on each page for ultimate control over your dynamic pages
    - Simple and Clear Custom Inspector
    - Tab Selection of elements
    - Escape to "go back"
- Multi-value Toggle Component – Allow your users to easily swap between Quality Settings
- Progress Bar Component
    - Simple
    - With Text Updating
- Slider showing current Value Component
- Slider with Input Component
- UI Gradients that can be applied to any Image
    - Linear, Weighted + Rotatable
    - Different Color per Sprite Corner
- Over 1200 Sprite Icons (Vector->PNG icons sourced and converted from Font Awesome)
- Essential Sprites:
    - 4 square variants for Panels and button backgrounds
    - 6 border variants per square.

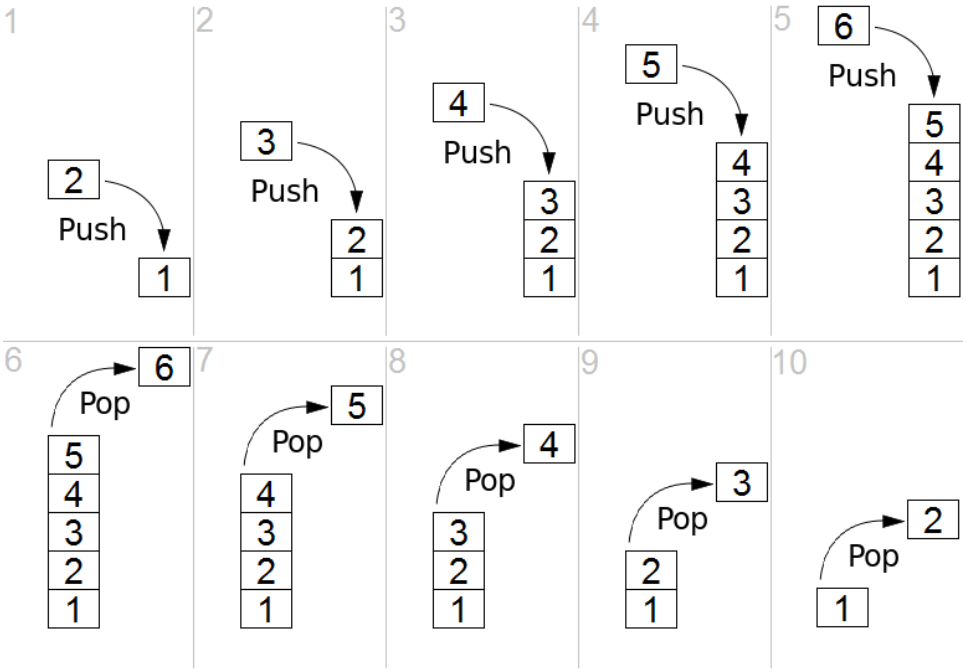
    - Circle
    - 8 circle border variants
    - 5 Indeterminate progress indicators with XCFs (GIMP project file)
    - 6 page backgrounds
    - Star Ratings (3 and 5 star variants)
- Rounded Corner Sprites
- Prefabs for:
    - A large variety of buttons
        - XCFs provided for each (GIMP project file)
    - Dropdown / Select
    - Modal
    - Toggles
    - Radio Buttons
    - Basic Slider
    - Sliders that show current value
    - Sliders with input field
    - Popover
    - Radial Progress Bars
    - Linear Progress Bars
    - Panels
- 2 Popular, Clean Fonts
- Sample Page Enter/Exit Sounds

Page Management

Concepts

The Stack

The concept of Page Management used in this pack is a **Stack**. Each new page is pushed on top, and going back removes the most recent page. If you are unfamiliar with this concept, this image from Wikipedia should help visualize the concept.



Each number (1,2,3,4,5,6) is a PAGE. As you travel deeper into the menu, the stack gets larger, and as we go back, it shrinks again.

Designing your UI

When designing your UI in this system, typically it is easiest to encapsulate each page as a full screen UI **GameObject**, and build your UI/Page within this. I typically create a parent empty object, set the anchors to: Min: 0,0. Max: 1,1 with anchored position at 0,0,0,0. Once I have constructed the page as I like within this, I move it off the screen and start on the next Page. Find what works best for you and your specific use case.

Take a look at the Demo Scene to see how each page is constructed for guidance on how I prefer to work. On larger projects you may consider making each page a Prefab to avoid scene conflicts. Hooking each page up requires minimal work and can be done fully through the Inspector. If you prefer to write code to manage the transitions, that is also supported.

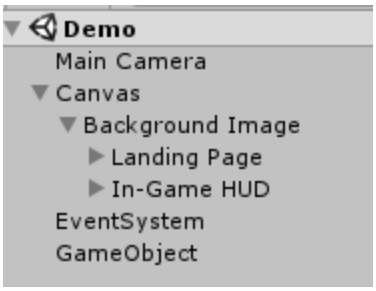
Setting Up

If you are starting from an empty Scene, add in a **Canvas**, and on this **Canvas** add the **MenuController** script. It will add an **Animator** to the **Canvas**, and depending on your need, you may want to also add a **CanvasGroup** to the **Canvas**.

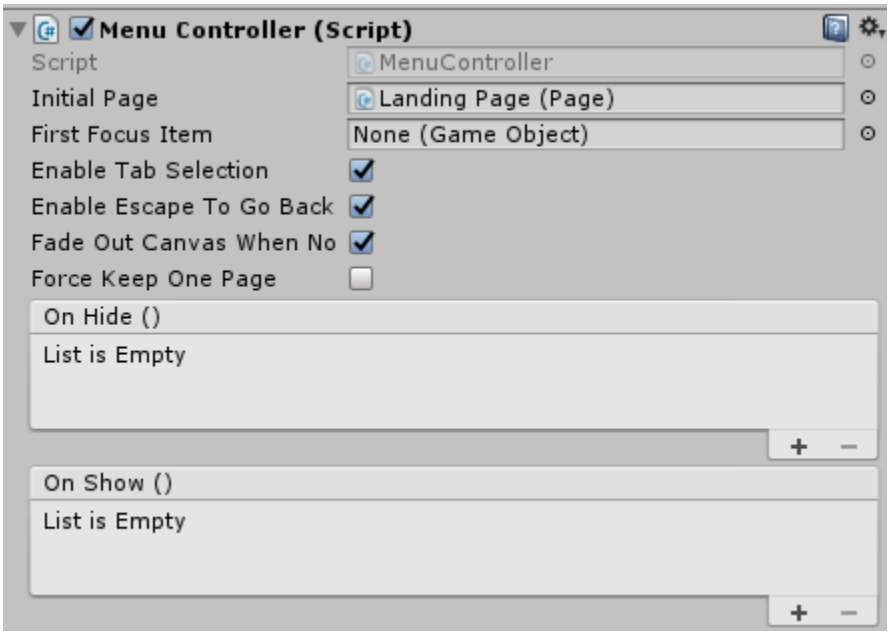
Typically, I will add a full screen background image as a child, then construct the pages as children of this.

In the Demo Scene you can see this structure.

1. Canvas – has the **MenuController**
    a. Background Image – just a full screen Image
        i. Landing Page - the first **Page** a user will see
        ii. In-Game HUD - another **Page**



MenuController

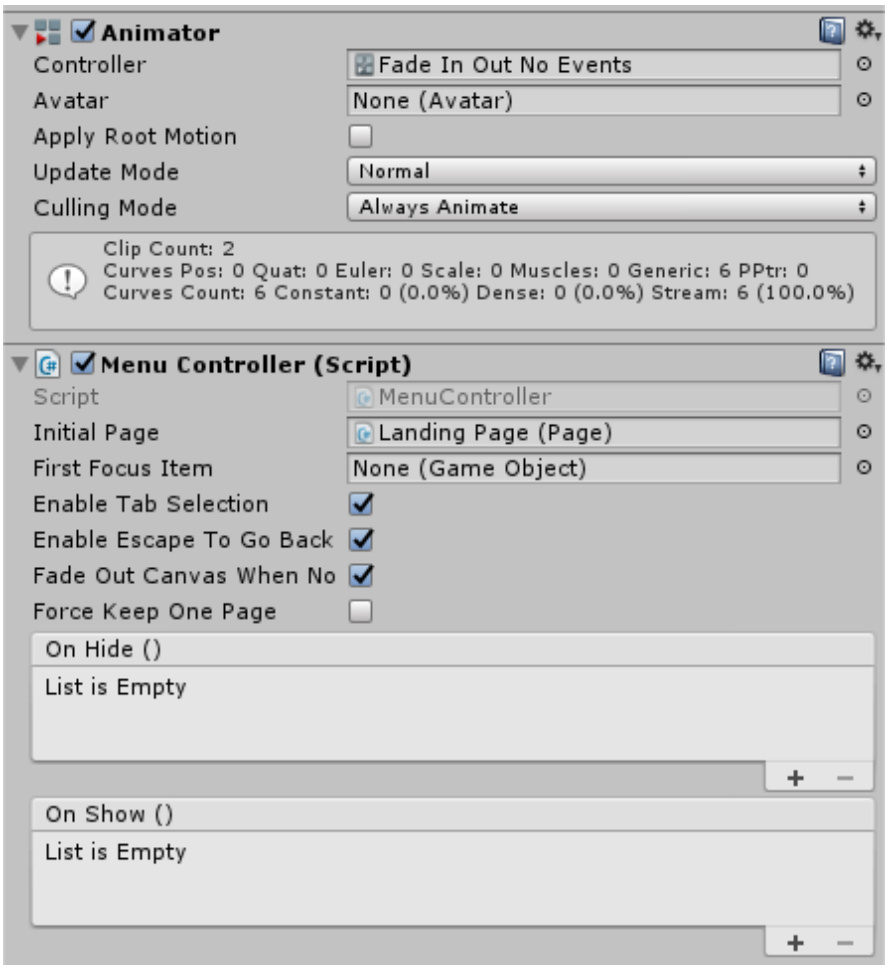


The **MenuController** is the primary worker to manage all the pages. All Stack management is done here. It also controls the Tab selection of **Selectable**s and Popping pages by pressing Escape. These options can both be turned off by un-ticking the appropriate checkbox (as seen above).

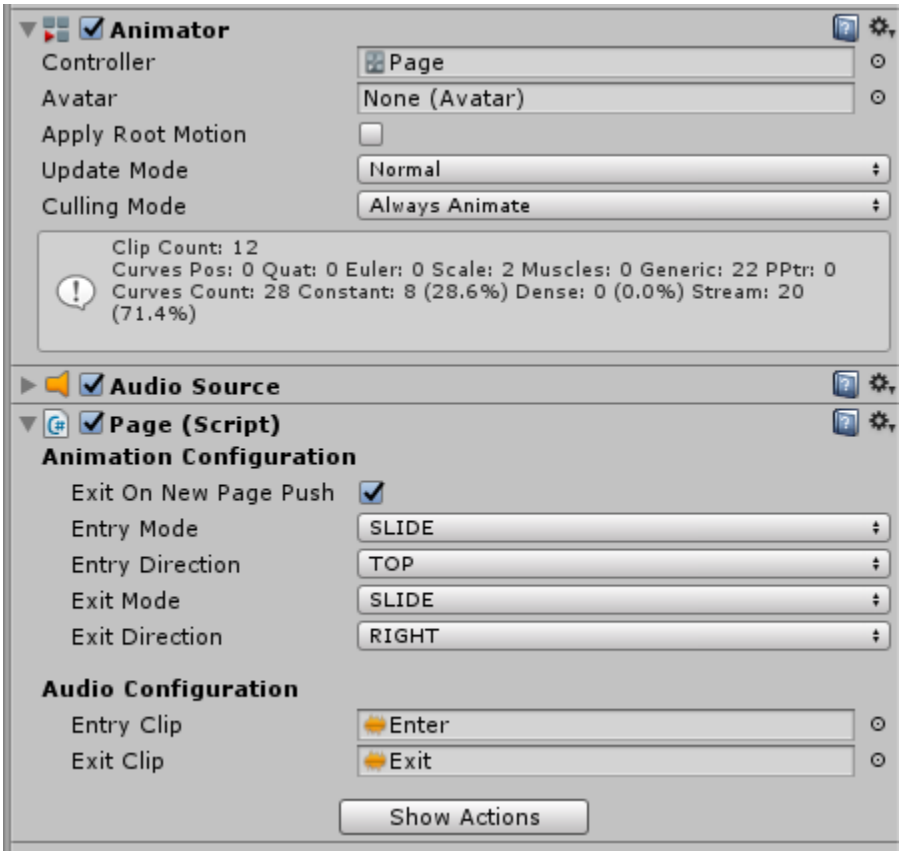
Initial Page is the first **Page** it will automatically Push when the scene starts.
First Focus Item allows you to pre-focus a specific element.

If you have the Use Case that the menu should fade out when all pages are popped, simply check **FadeOutCanvasWhenNoPagesRemaining**, add a **CanvasGroup** to the **Canvas**, and attach the **Fade (No Events) Controller** to the **Animator**. The user can bring it back up by pressing Escape again.

If your game is fully 2D, or you always want the Canvas visible, *do not* enable **FadeOutCanvasWhenNoPagesRemaining**. This option allows the Canvas to fade out and become inactive. When disabled, all pages can be removed by pressing Escape. To prevent this, check **ForceKeepOnePage**. This option will make the MenuController always keep 1 page visible. This option overrides **FadeOutCanvasWhenNoPagesRemaining**. Meaning if you have both checked, the Canvas will *never fade out and never call OnHide events*.


Animator
Controller
Fade In Out No Events
Avatar
None (Avatar)
Apply Root Motion
Update Mode
Normal
Culling Mode
Always Animate
Clip Count: 2
Curves Pos: 0 Quat: 0 Euler: 0 Scale: 0 Muscles: 0 Generic: 6 PPtr: 0
Curves Count: 6 Constant: 0 (0.0%) Dense: 0 (0.0%) Stream: 6 (100.0%)
Menu Controller (Script)
Script
MenuController
Initial Page
Landing Page (Page)
First Focus Item
None (Game Object)
Enable Tab Selection
Enable Escape To Go Back
Fade Out Canvas When No
Force Keep One Page
On Hide ()
List is Empty
On Show ()
List is Empty

Page



The **Page** component has an important default behavior to be aware of. Each **Selectable** as a child of this **Page** will be disabled until **the** Page is pushed. This is done to prevent improper keyboard navigation via tab or otherwise.

Under Animation Configuration you can see both Entry and Exit Mode are configurable independently, meaning a Page can ZOOM *in* but SLIDE DOWN *out*. SLIDE is the only Entry/Exit Mode with a Direction, since ZOOM typically originates from the center of the screen, and Fade is a full screen effect.
“Exit On New Page Push” will play the Exit animation when a new page is pushed on top of this one. This is useful when you have no background on your page and want to reuse the primary background (as done on the Demo). If you have unique backgrounds per page, or want to see the previous page still, you can un-tick this.

Under Audio Configuration there is an option to play a sound. If you do not want a sound to play, then simply leave it unassigned. The Entry and Exit Clips will play every time a new page is pushed or popped respectively – except in the case of a page coming back in due to “Exit On New Page Push”.

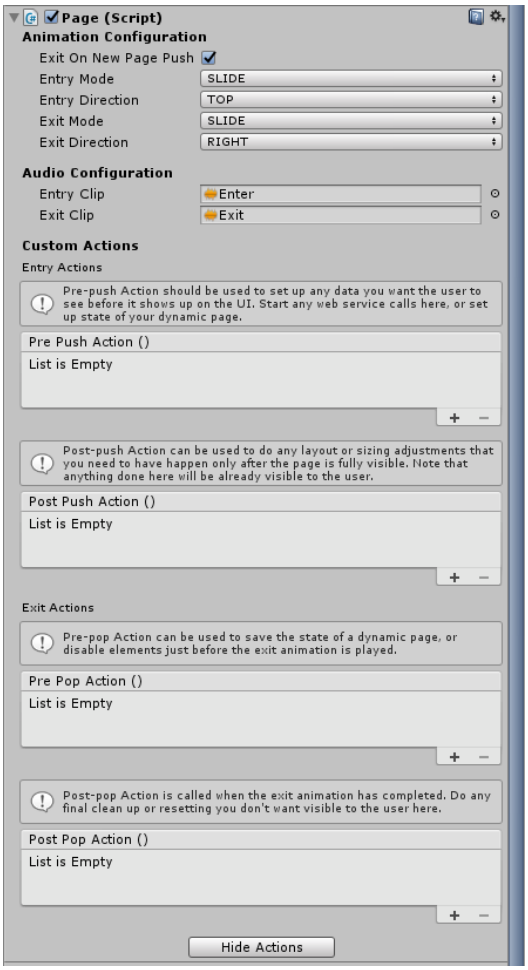
For example, Referring to our Stack image (square 10) above. When page 2 is popped, Page 2 exits and plays a sound. Page 1 also re-enters but does not play a sound.

Make sure to assign the Page Animation Controller for the animations to work. You can also create your own custom animations / Animation Controller and use that. If you need to use the advanced feature of

Custom Actions, please read the next section for some warnings on how your animation needs to be set up for them to still work.

Advanced Usage

Pages also support custom actions.



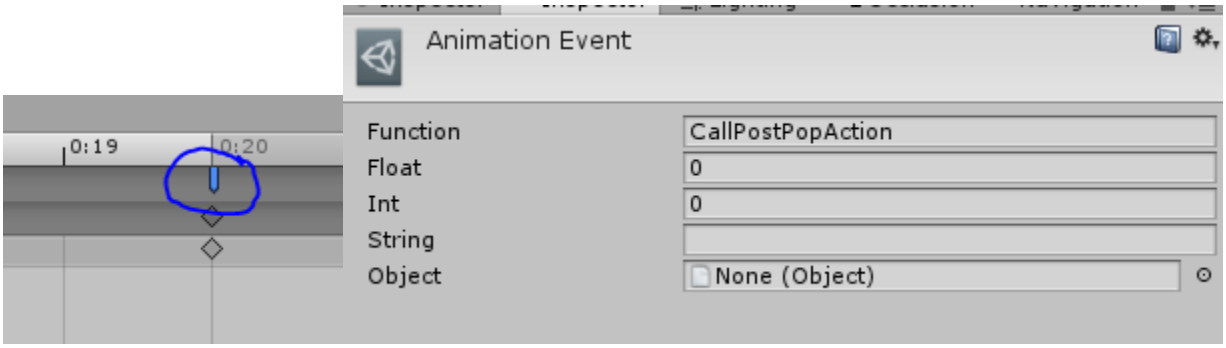
This most likely requires some code from you to do something. In the inspector there is some help about exactly when each action will be called. There are 4 timings on the events:

1. PrePush – before the push happens, call these actions.
    - In general, this is a good place to start any web service calls (if you didn't do it on Start), do dynamic element creation, or do any state setting.
2. PostPush – after the push is complete (animation is done), call these actions.
    - This is a good place to put any post-render layout adjustments. For example: resizing a ScrollRect Content area to fit dynamically generated content, and scroll to the top. Be aware that anything done here is done **while the user can see the page**!
3. PrePop – before the pop happens, call these actions
    - This is a good place to save the state of a page before it goes off screen. Be aware that anything done here is partially visible to the user since it will **start happening** while the page is still visible.

4. PostPop – after the pop is complete (animation is done), call these actions.
    - This is a good place to reset the state of a page while the user cannot see it, or do any garbage cleanup.

Custom Animations

Since we want a generic way to call **PostPopAction** and **PostPushAction** after an animation has completed, and don't want to make you input "the animation is 0.235 seconds long" on each page, we raise an **AnimationEvent** to handle calling these methods. This means when you create an Animation you want to use, it is ***highly recommended*** you add an **AnimationEvent** on the last frame.

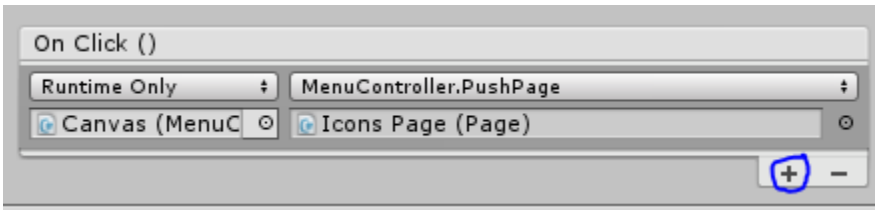


You must set the Function on *OUT* animations to be "**CallPoPopAction**" and set the Function on *IN* animations to "**CallPostPushAction**".

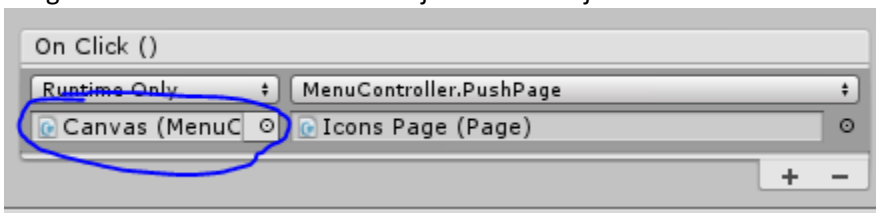
Making it All Work Together

From the Inspector you can fully define all animations from your menu with absolutely no coding. For the common use case of clicking a button causing page navigation, we can do the following:

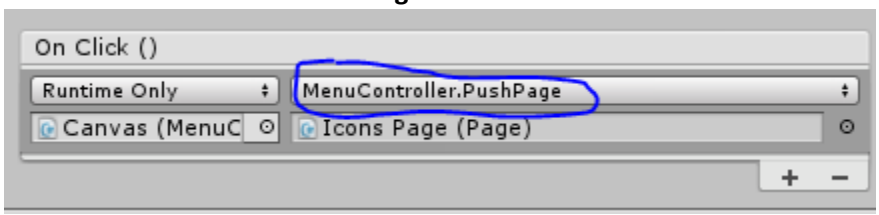
1. Add an **OnClick** event to the **Button**



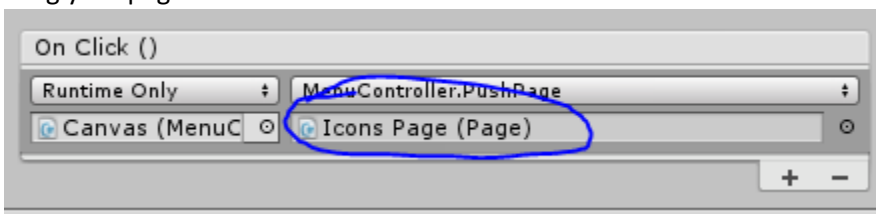
2. Assign the **OnClick** event
    a. Drag the **MenuController** GameObject to the Object field



    b. Select **MenuController** > **PushPage** from the function list



c. Drag your page to the **Value** field

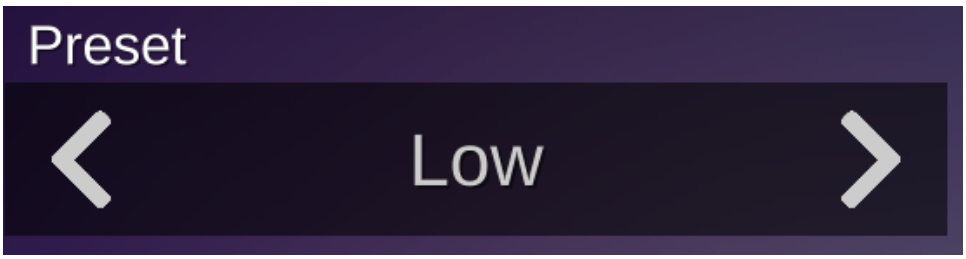


3. Set up your **Page** animations as described above in the Page section

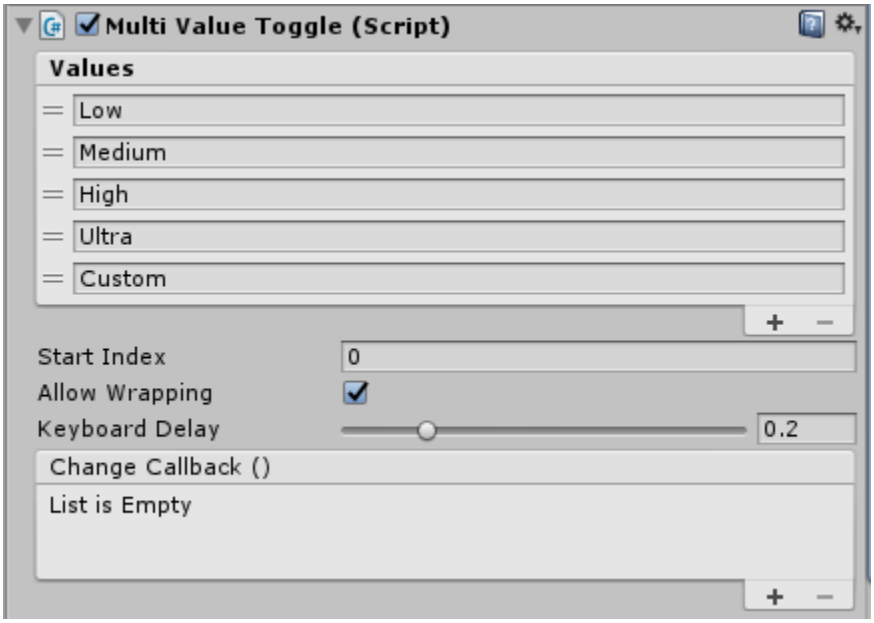
Components

Multi-Value Toggle

Every PC game has Graphics options, and how do you set them? Usually via this control:



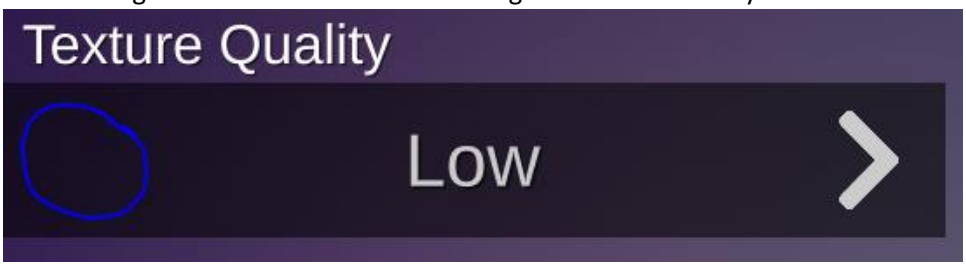
This is what we call a Multi-Value Toggle.



You can see the following configuration options:

1. **Values** – the list of available options
2. **Start Index** – the index of the value to start on (“Low” in this case)
3. **Allow Wrapping** – if ticked, allows user to press the left arrow from the first value (“Low” in this case) and go to the last value (“Custom” in this case). Also allows pressing the right arrow when on the last value (“Custom”) and it will wrap to the first value (“Low”). Unticking it results in the

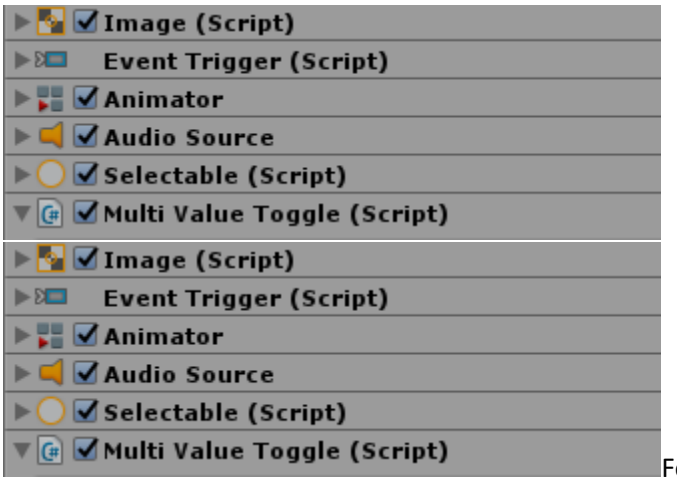
arrow being hidden when the user cannot go that direction any more:



4. **Keyboard Delay** – the left and right arrows will navigate to the previous/next value. This delay is the delay between "swaps". A too low value will result in accidental "double taps" and a too high value will make the user feel this control is unresponsive.
5. **Change Callback** – an action to call whenever the option has changed. Useful to apply modifications on the fly instead of waiting for user to press an "Apply" button.

There is also a public API for this component so you can dynamically populate their values and adjust all of these options in code.

You will notice this component has many dependencies as it is a fully managed component including animations and focus.

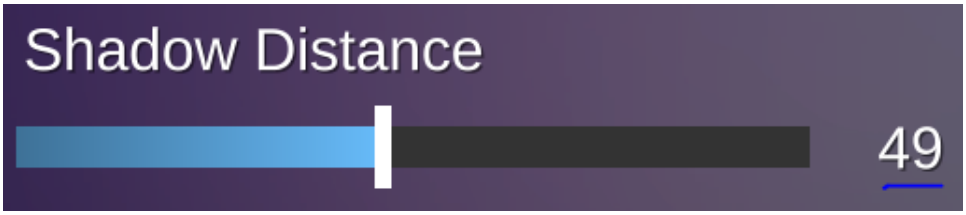


For this reason, it is recommended to use the provided prefab and modify it rather than manually just attaching the MultiValueToggle to a component and trying to rebuild it yourself.

Sliders

There are 2 types of customized sliders provided. Both use the default Unity Slider and extend the capability.

Slider with Text



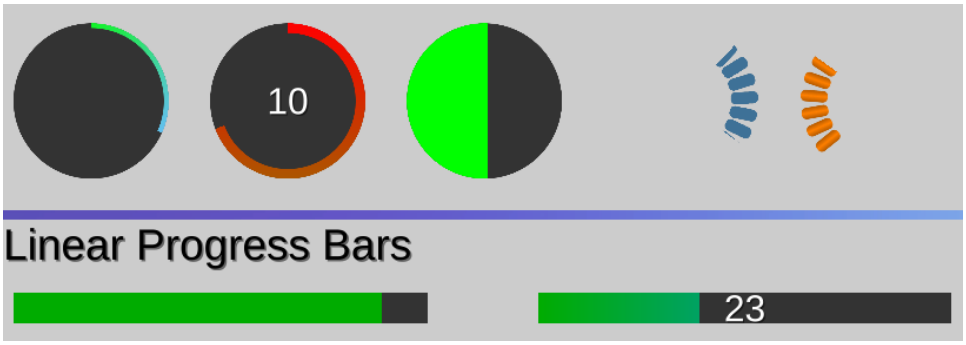
A prefab is provided for this. It will, on change, set the text to what the Slider value is. This is achieved by setting the OnChange event on the Slider to call the script SliderValueDisplay.SetText. The number of decimals to show is configurable on this script.

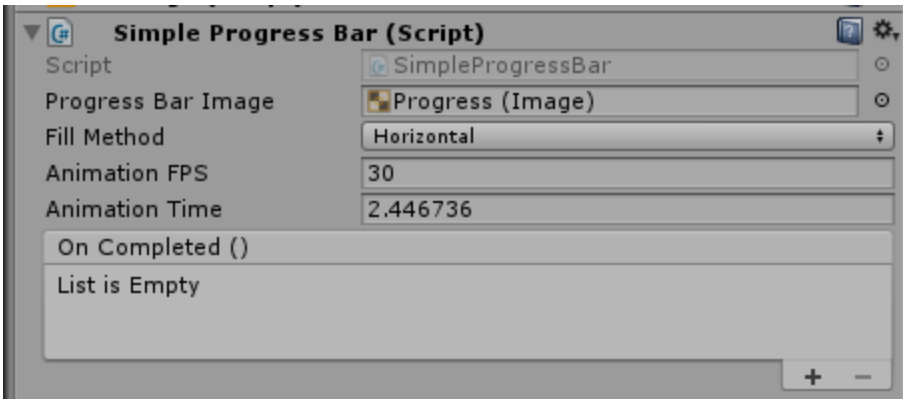
Slider with Input Field

A prefab is provided for this. On Change of the slider it will set the Input Field value to the Slider's value. When a user completes typing in the input field (via pressing Enter or making the Input Field lose focus), the Slider will update to that value. This is achieved by double binding onEndEdit on the InputField and onChange on the Slider to InputFieldSliderValueDisplay.SetValue and SliderValueSetter.SetValue respectively. The number of decimals to show is configurable on InputFieldSliderValueDisplay.

Progress Bar

Progress Bars animate the fill of an image over a configurable duration and FPS. They will work on any Image, any Fill type, and have an optional OnCompleted action. You can use this for things like displaying an alert when a user levels up, for example.



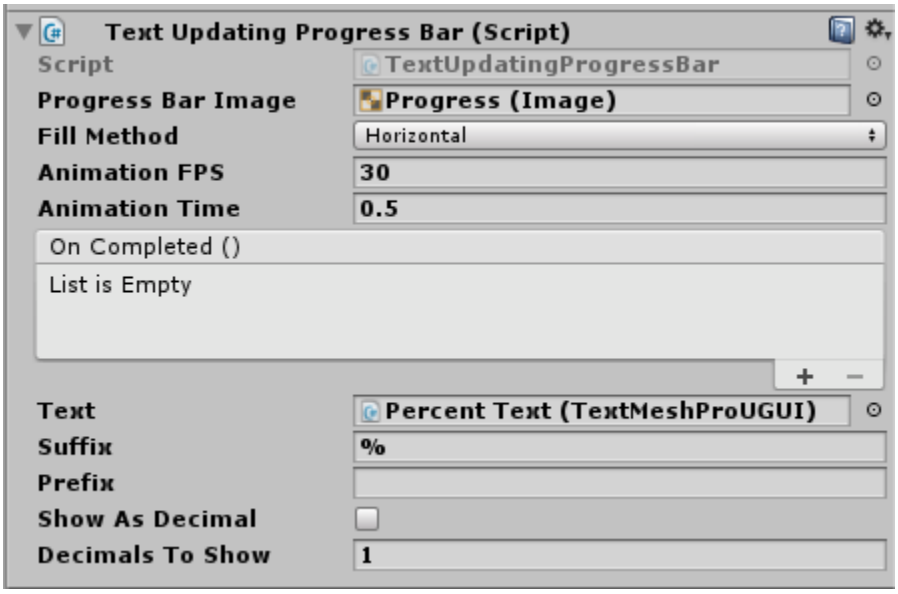


The following configuration options are available:

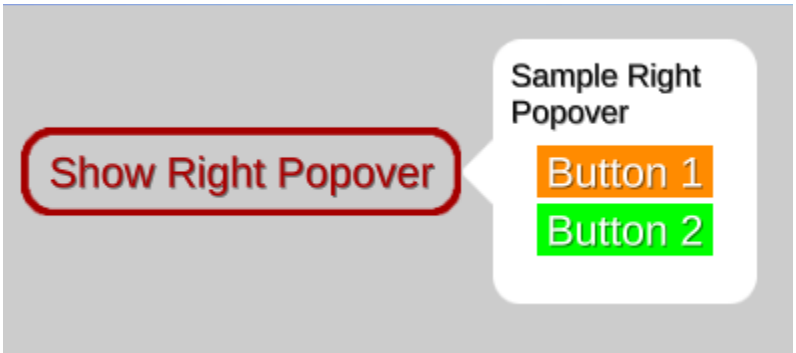
1. Progress Bar Image – the image to animate, usually a child of the Progress Bar Game Object
2. Fill Method – the method to fill the image, all options are supported.
3. Animation FPS – how many frames per second to animate the image at. 30 is a good starting point for smoothness versus cpu intensity.
4. Animation Time – the amount of time it should take to animate.
5. OnCompleted – an Action to call when the progress bar reaches 100%. Note that this is **not** called when the animation completes, and the progress bar has not reached 100%.

Progress Bars are fully interactable via Script as well. Call SetProgress(float percent) to animate it via code. SetProgressImmediate(float percent) will not animate. The second is useful when resetting to 0, for example.

A utility subclass TextUpdatingProgressBar will show the percentage the progress bar is currently at as it animates.



Popovers

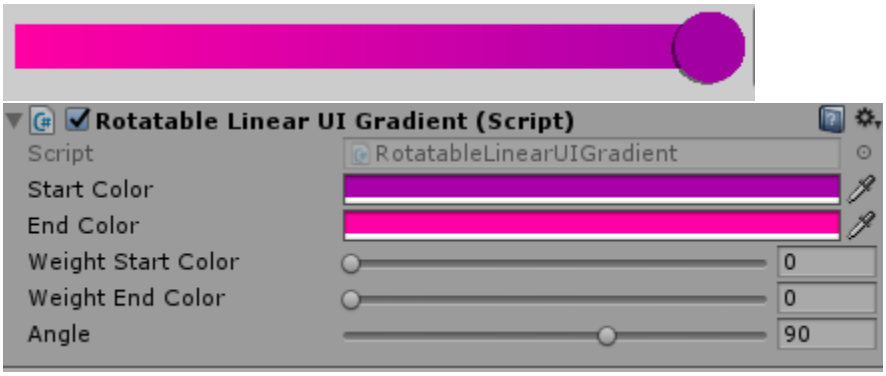


Sometimes also called a “callout”. Prefabs provided for Up / Right / Down / Left variations. Resize, Recolor, and put in whatever content you like. There is a basic script included to manage visibility.

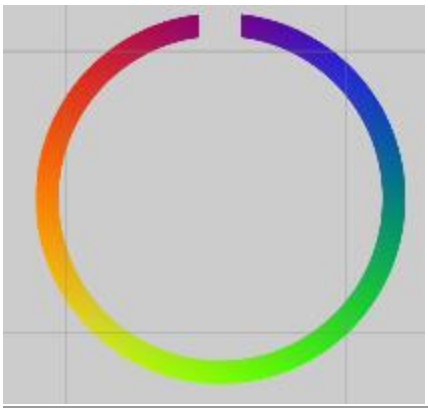
Gradients

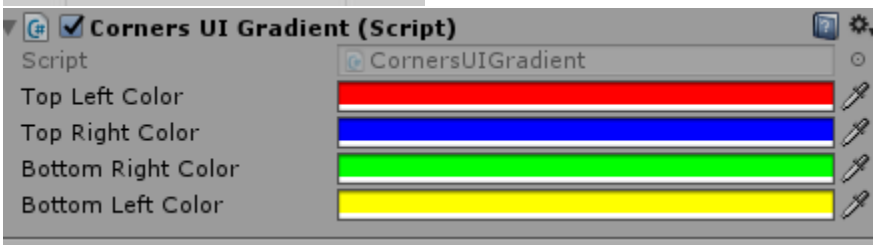
This is a Vertex Modification script (no custom shader work). Setting the main image color as White will give you the best results, but it will blend the color of the image with the colors provided on the gradient.
There are 2 kinds of gradients provided:

1. Linear, Rotatable – Linearly goes from 1 color to the second. You can weight it towards one color or the other, and also rotate it.



2. Corners – Color each corner of the image any color you like.

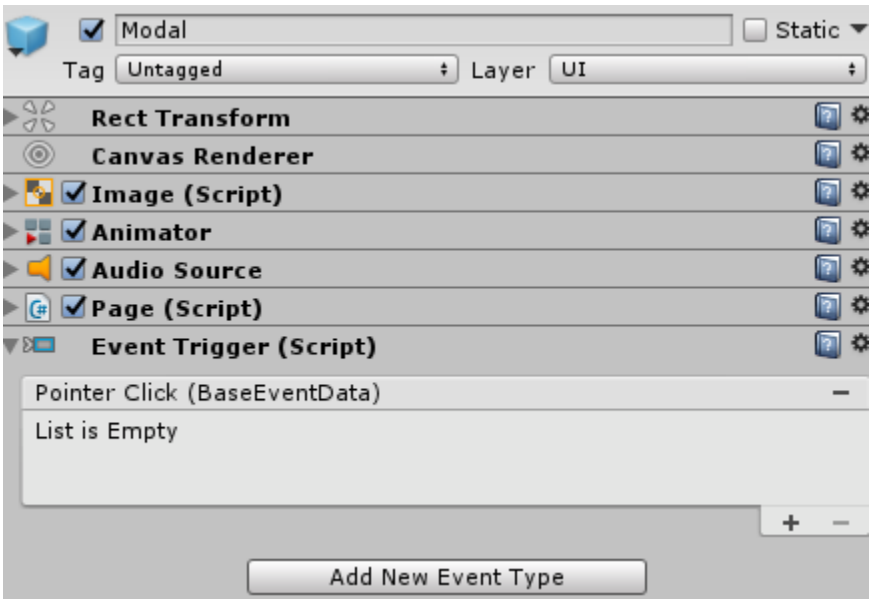




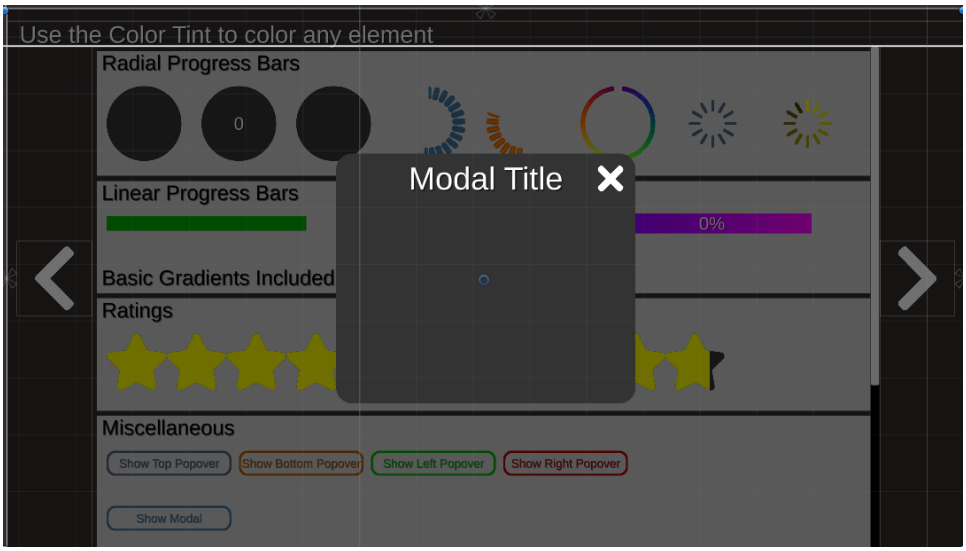
These can be applied to any Image and have been applied to many in the demo.

Modal

Pretty much everything has modals (aka popups) nowadays. There's a Prefab that you can use as the base for all of yours!



To make it where a user can click on the mask and it dismisses, bind this onClick to MenuController.PopPage



You can of course size it and put in as much (or little) content as you like.

The Overlay component has an EventTrigger added as a convenience to allow you to add a click to the background overlay to dismiss the Modal. Simply bind the PopPage method to the OnClick action here.

Radio Button / Toggle / Dropdown

These elements are all pretty much provided by Unity, but not necessarily in an easily accessible way. There is a prefab for each for drag & drop better looking options (using TextMeshPro and higher quality images).

Buttons

There are 10 stylized buttons provided, each with a prefab.

Additionally, there are 4 flat buttons, and 6 outline variants for each button.

You can combine the stylized buttons with outline flat buttons for a huge number of custom buttons.

All buttons are provided in greyscale so you can Color Tint them to the color of your choice. This reduces your total download size and increases your color customization options. This also means that you can use the built-in Color Tint on buttons to color them for the different states, without having to sprite swap for each one.

Provided Animations

The provided animations were designed to work with a 1920x1080 resolution Canvas (with Canvas Scaler) with UI Scale Mode set to Scale with Screen Size with Screen Match Mode SHRINK. This scales the down to lower resolutions and prevents it from showing off-screen content on higher resolution Canvases.

You can of course use any Canvas scaling mechanism you like. With the provided animations you may see pieces of pages that are offscreen if you do not follow this design pattern.

Please see the Unity Canvas Scaler documentation for full details on what happens with the various modes.

Code Reference

Please see the **Help/Documentation.chm** file provided in the Asset for fully documented Scripting API reference.

FAQ

Q: How do I set up a new page?

A: Watch this video! Basically, you need to bind a user event, such as a button click to call MenuController > PushPage(Your New Page Here)

Q: Using your animations I can see parts of other pages!

A: Make sure to set your Canvas Scaler to 1920x1080 with mode "SHRINK" to use the provided animations. This is recommended to ensure your pages will never show up overlapping. You can set up your canvas however you need, but you will need to recreate these animations for other scaling modes to ensure this issue doesn't happen.

Q: I need dynamically set up content on a page before it shows up, how do I do that?

A: You will need to set it up in either a PrePush or PostPush action. These are available by pressing "Show Actions" button in the inspector for a Page. You'll have to write the "set up content" code then just bind the function in either PrePush or PostPush section of the Page (see above section "Page > Advanced Usage" for details on how these work)

Q: How do I turn off the built-in keyboard shortcuts?

A: Un-tick "Enable Tab Selection" and "Enable Escape To Go Back" in the Menu Controller.

Q: How do I create my own page enter/exit animations?

A: Add a new page to your Canvas and position it off screen. Go to the Animation Panel and create a new animation, then set it up as you would any other animation. Then to make sure the PostPop and PostPush actions work, add an Animation Event to the event and have it call "CallPostPushAction" or "CallPostPopAction" depending on if it's an In or Out animation. You can look at the provided Page Animations to see exactly how this is done.

Q: I have another issue that's not listed here and I can't figure it out!

A: Please send me an email at support@llama.software and I'll be happy to help you!

If you find this asset helpful, please leave a 5 star rating on the Asset Store so others can more easily find it!