Essential UI Kit

INTRODUCTION

About

Thanks for checking out the Essential UI Kit and we hope you find it useful for creating a great user interface for your games. We'll also try our best to keep the usage of this package as simple and straightforward as possible but to improve it further your input is most welcome. If you do like the package as it is consider leaving us a review and if you don't you would really help us by telling us about your problems. Suggestions and feature request are also most welcome.

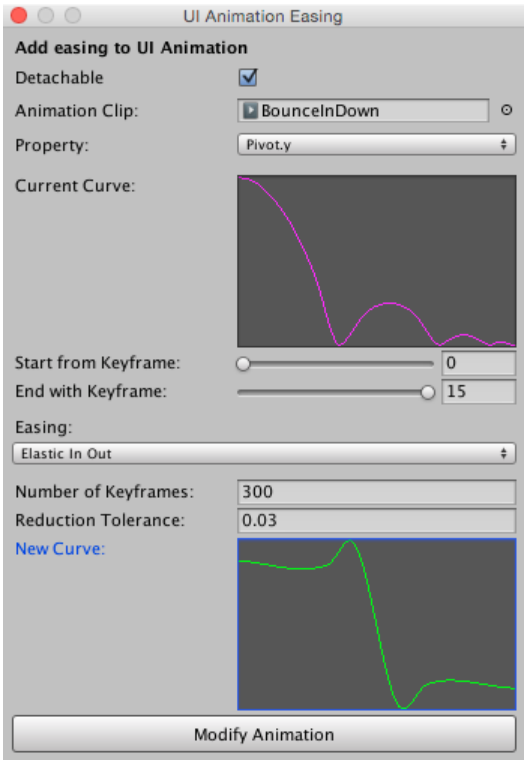
Contact

Send an email to modernalchemyapps@gmail.com or visit www.modernalchemy.de, where we'll also put short videos showing the usage of certain parts of the package.

EASING

Easing or tweening is the changing of a value, from a start point to an end point, over time. You can move the value from the start to the end linearly, meaning with the same, constant speed, but usually it looks much better if you vary it a bit (i.e. fast at first but then getting slower as you reach the end value). And to achieve that you can use easing functions, which really are key to interesting animations.
The Essential UI Kit comes with pretty much all the commonly used easing functions and below I'll explain a number of ways in which you can use them.

In the Editor



The editor window allows you to modify an existing animation with easing. It's probably the easiest & fastest way to add easing to your game. Since all the easing calculation is done in the Editor and just stored in the animation keyframes there is virtually no performance penalty here.
You'll find the easing editor under Window/Essential UI Kit/Add Easing to Animation. The options the window presents you are:

- **Detachable**: use the editor as a separate window or one that can be docked in the Editor.
- **Animation Clip:** The animation clip you want to modify
- **Property**: the clip property (i.e position.y) you want to add easing to
- **Current Curve**: the current animation curve for the selected property

- **Start from Keyframe and End with Keyframe:** The portion of the animation clip between those keyframes will have easing added and the rest will be ignored
- **Easing:** the easing function you want to use
- **Number of Keyframes and Reduction Tolerance**: Increasing the number of keyframes initially generated will make the animation more precise. A lot of the keyframes may be unnecessary so that we also provide keyframe reduction (that means removing as many keyframes as possible while leaving the curve shape as little changed as possible). The reduction tolerance value sets a limit on how strongly the curve can be changed in order to reduce keyframes (a zero will reduce no keyframes)
- **New Curve:** The new animation curve taking into account all the selected options.

From Code

It's also possible to ease from code and it's widely used within the kit. Its best for dynamic animations where the start and end values are not known before the game starts or are changing. We have prepared a (heavily commented) example where you can see how it works (EasingTest.cs under Examples/Scripts)

ELEMENTS

Canvas Element 3D (3D Model in Screen Space Canvas)

Use Cases: Weapon or Item Selection Menu, 3D coin or health symbols
Example: Overview Scene, CanvasElement3D Scene
Things to note: You can set auto-update to true if you want the 3D Model to be scaled whenever the parent rect is changed. You can also set eased to true if you want the scale adjustment to be smooth.

Page Scroller

Use Cases: Tutorials, Info screens, Loading Menu Tips
Example: Overview Scene, DraggablePage Scene
Things to note: The pages for the Page Scroller can be scrolled vertically or horizontally. (Diagonally too if you want, generally they go from one offscreen position to an active onscreen position and then either back or to another offscreen position, depending on where the user scrolls to. You can set all these positions up in the Editor.)

Lists

Use Cases: Option Selection, showing additional Information on Elements
Example: ShopMenue Scene, Map Scene
Things to note: A list element is just a basic Selectable that can expand (it can also fire of custom events) when selected. The selected element for any list can be queried from the list controller.

UI Menu (General Menu Panel)

Use Cases: classic 1-Panel Menus (such as options menues etc.)
Example:
Things to note: For lower level menus (like Graphic Effects Options) you can fill in the higher Menu field (Example: Graphic Options → Options → Main Menu). This can then be used to travel back up that chain with hotkeys (like Escape)

Tooltips

Use Cases: Tooltips for UI Elements
Example: Tooltip Scene
Things to note: a tooltip consists of at least 2 components. A number of tooltip targets (objects that will show object specific information when hovered over) and 1 or more tooltips (components that will display that information, you only need more than one if you want to show differently styled tooltips).

Modals & Notification

Use Cases: Warnings, Yes-No-Options, Pause Menues, Achievements (Notifications), Power-Ups (Notifications)
Example: Overview Scene
Things to note: Notifications can either be stacked (using a layout group) or replaced when the next new notification comes in.

Scroll Rect Controller (Maps & More)

Use Cases: Maps, very long lists
Example: Map Scene, Shopmenu Scene
Things to note: the Scroll Rect Controller can jump either to a point inside the Scroll Rect or make sure that the Scroll Rect is centered on a given Rect Transform. It respects the Scroll Rects Moving Type (i.e. if unrestricted it will center on the rect transform even if it shows parts outside the Scroll Rect's content and if if's clamped it will only go to the content's border, meaning the target Rect Transform will not be completely centered)

Dials

Use Cases: Progress, Loading Bars, Health & Energy Bars
Example: GUIElements Scene
Things to note: A shows a value with between min and max. It can also show the current value as a formatted string with a given number of zeroes behind the comma.

Slider Pop-Up

Use Cases: Sliders where the exact value should be visible
Example: GUIElements Scene

Things to note: The slider popup will show up when you drag the slider handle and show the sliders current value. Like with the Dial you can also choose how the shown value will be formatted (How many zeroes behind the comma).

Input Validation

Use Cases: Input that you want in a certain form and give feedback in case it's incorrect
Example: GUIElements Scene
Things to note: Put a InputValidation script on any of your InputFields and add as many Validators as you like. The user input must pass all validators in order for the InputValidation Script to consider the input valid. By far the most versatile Validator is the RegexValidator which uses regular expressions to check the Input. There is no easy introduction into regular expressions but there are some useful regular expressions (Domain, E-mail etc…) included that you can select from a drop-down in the Regex Validator Editor. There are also lots of resources on the internet that can help you find regular expressions that suit your needs. A good starting point

Inventory

Use Cases: Weapons Inventory, Items Inventory
Example: Inventory Scene
Things to note: The Inventory is basic but gives you Drag & Drop between Inventories of the same type, notifies you when a new Item has been added or an old one has been removed and of course access to all the items that are currently in the Inventory,

EDITOR HELPER

Color Schemes

Create a Color Schema (Asset/Create/Color Scheme) give it a name and add some Colors. You should also name the Colors appropriately to make it easier for you to find your Colors later. You can use the ***DisplayablePaletteColor*** class as a property in your Scripts and you'll see a dropdown with all the palette colors in the editor (using the names you have given them earlier). ***DisplayablePaletteColor*** is also implicitly convertible to **Color** meaning that you can assign it to a Color property and it will have the value of your selected Color.
Tip:
Use the Editor Color Setter Script to assign Colors from your palette to Materials, UI Images and Cameras in the Editor.

Spritesheet from PNG-Sequence

Just select the folder with the sprite sequence and specify how many sprites you want to be in a row (on the x-axis). Remember to set the PNG-files to advanced and read/write enabled otherwise they can't be accessed.

Attributes

The kit also includes some useful property attributes such as:

- **GetOrAddComponent**: this either gets the Component of the property field type on the GameObject or adds a new one in case none is present. It also directly initializes the field to that Component. Optionally you can also set lookinChildren or lookinParents to true and it will also look among the GameObjects children or parent for the Component before adding it. Note: this is a helper script that should provide a good default for a property, it is however not a hard guarantee that the Component is really present at Runtime.
- **RequireDependencies**: If you have a GameObject field on one of your scripts and pass RequireDepencies one or more types it will check whether components with this type are present on the GameObject or Prefab that you have selected for that field. If they're not present a warning will appear, telling you which Components are missing on the GameObject