SKRPTR

1. Intro

Hello and thank you for buying Skrptr – it means probably more to me than you realize :).

I would like to recommend to read this document entirely even if it might become a bit too technical, as it will hopefully explain every little detail that will give you a cutting edge advantage to structuring your UI.

Skrptr is meant to allow any kind of developer, especially designers, to implement a rapid prototype of an UI that then they can pass along to a software engineer and implement the functionality without altering structure / design / layout.

2. Components

In this section I will take the abstract components and explain the logic behind them, please bear with me, I am terrible with explanations.

2.1. Skrptr Element / Radio Button / Checkbox

The main component that drives the behavior is the Skrptr Element, by adding this component to a UI game object will make the gameobject interactable with the Skrptr System. This means, that the object will start to receive events for Hover, Selection, Enable, Locking, Hiding, Looping, Long press, and their counterpart events.

Skrptr Actions, Animations, Triggers and more can be set to react to every event, allowing you to set anything whenever you want. More on this in the Action and Animation sections.

The Checkbox element is nothing more than a Skrptr Element, with the functionality of a ‘check’ state. The check state is triggered by its respective events: Check and Uncheck, based on the elements state, one of the two will be fired whenever this changes.

The Radio button is once again nothing more than a Skrptr Checkbox, but it will search for Radio buttons in the same hierarchy level (siblings) and uncheck them if they are checked.

Every SkrptrElement has a list of events to be called on Start, you can call as many as you wish to set the scene right on start / initialize other properties / panels etc.

2.2. Skrptr Main

This is the class that holds the current information, which element is selected, hovered, clicked, visited, and perhaps more with time. All interaction systems use this static class in order to retrieve or store relevant information.

2.3. Skrptr Events

Hopefully the next table will explain exactly when a event is fired, based on the interaction system. More systems can always be added to expand the functionality. In the Touch / Mouse / Keyboard interaction it is assumed that the respective action is happening on a gameobject with a Skrptr Element component.

Skrptr Event	Touch Interaction	Mouse Interaction	Keyboard Interaction
Click	When finger is lifted off	Left Mouse Click Release	Default on Enter Press
Select	When finger started pressing	Left Mouse Click Down	Any Directional Input that moves selection on a new element
Deselect	On Finger liftoff / selection of another element	When mouse leaves selection / selects another element	Right after selection, last selected element will be deselected
Hover Enter	Not Triggered	Should be obvious	Not Triggered
Hover Exit	Not Triggered	Should be obvious	Not Triggered
Enable	When gameobject is enabled *	When gameobject is enabled *	When gameobject is enabled *
Disable	When gameobject is disabled *	When gameobject is disabled *	When gameobject is disabled *
Hide	Custom event for user to setup for intro / outro animation, can be fired only using Triggers	Custom event for user to setup for intro / outro animation, can be fired only using Triggers	Custom event for user to setup for intro / outro animation, can be fired only using Triggers
Show	Same as above	Same as above	Same as above
Lock	Same as above, but it turns off raycast on target	Same as above, but it turns off raycast on target	Same as above, but it turns off raycast on target
Unlock	Same as above, but it turns on raycast on target	Same as above, but it turns on raycast on target	Same as above, but it turns on raycast on target
Check	Event fired on click if the element is a radio button / checkbox	Event fired on click if the element is a radio button / checkbox	Event fired on click if the element is a radio button / checkbox
Uncheck	Event fired on click if the element is a radio button / checkbox	Event fired on click if the element is a radio button / checkbox	Event fired on click if the element is a radio button / checkbox
Long Press	Fired after 1.5 seconds long press selection, on liftoff CLICK event will NOT FIRE	Fired after 1.5 seconds long press selection, on left mouse button release CLICK event will NOT FIRE	Not Triggered
Loop	Event that will loop, can be triggered only by SkrptrStartLoop/SkrptrEndLoop	Event that will loop, can be triggered only by SkrptrStartLoop/SkrptrEndLoop	Event that will loop, can be triggered only by SkrptrStartLoop/SkrptrEndLoop

Right, that's a lot of information, but hopefully most of it makes some sort of sense.

*Off note: There is no beautiful way to hook an event when a gameobject is enabled, therefore if you wish to enable / disable an object, call the Enable / Disable event, this will also enable / disable the gameobject that holds the Skrptr Element. (More on how to call this on Skrptr Trigger)

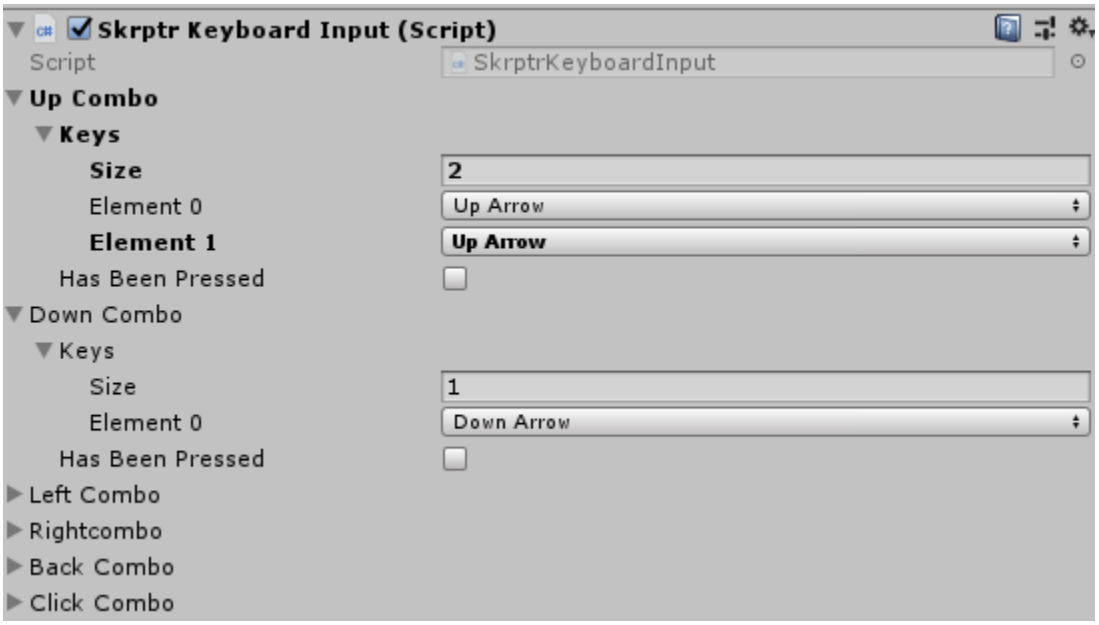
2.4. Skrptr Mouse / Touch Input

This drives the input event system and calls the respective, above mentioned, events for the mouse and touch interaction systems. Can always be altered by simply calling different events if you wish to do so.

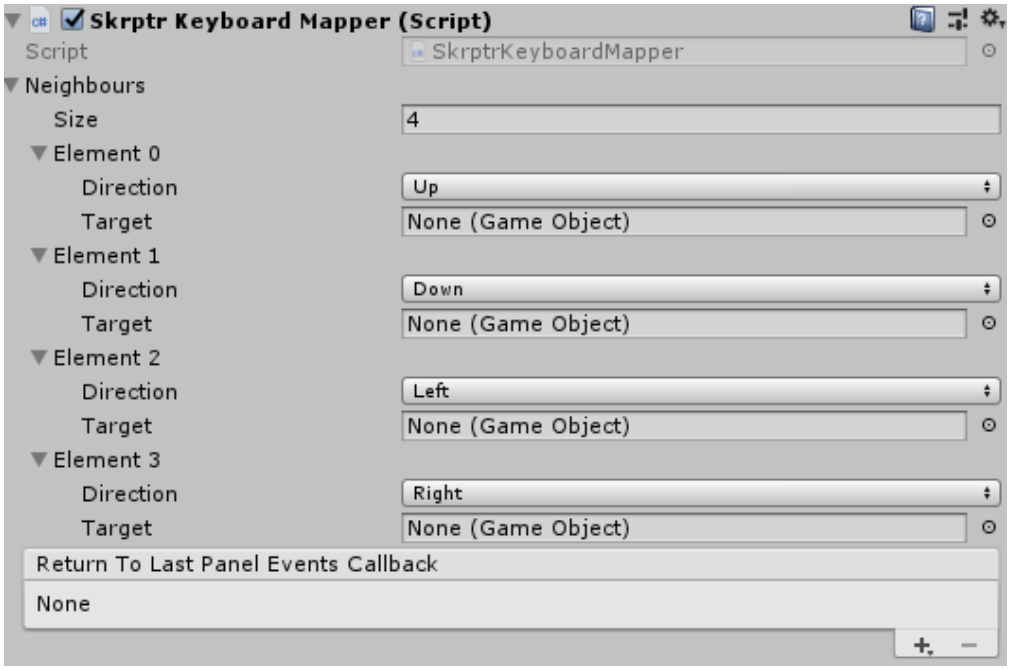
In this component you can also set the delta time after which the long press event is fired instead of a click.

2.5. Skrptr Keyboard Input & Keyboard Mapper

The Keyboard Input system can always be remapped, and it even supports key combinations. If you require another directional navigation system mapped, my advise is to fake and simulate input so that Skrptr consider it a keypress. (Example, you have swipe pads / joysticks / other system that sends you some input, instead of writing a whole new Skrptr Input system, simply forward the Input from your system under the form of Keypresses that are mapped in the Keyboard Input – hope I didn't overcomplicate the explanation)



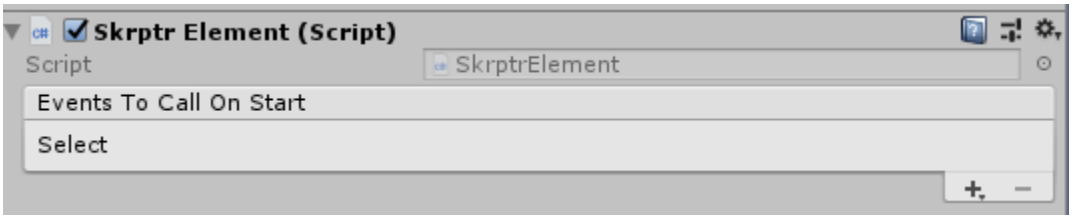
Unlike the mouse / touch system, the keyboard input requires another component that 'links' elements between each other, for that we have the keyboard mapper.



It is pretty straight forward: If this element is selected, and directional input is registered, this element will become unselected, and the new one (the target here in the mapper) will be selected.

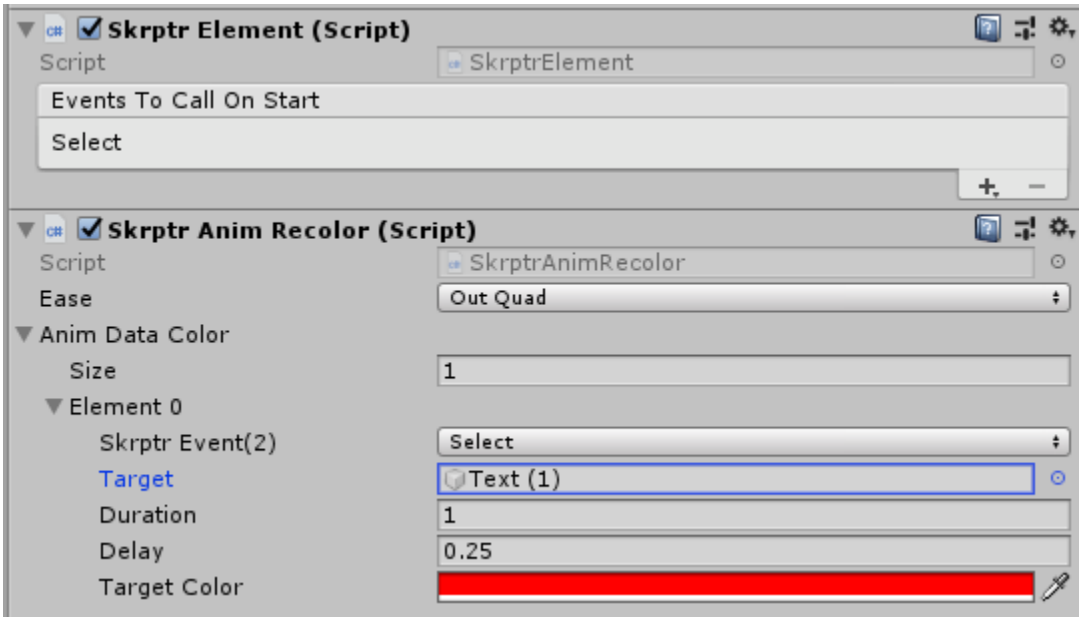
The “Return To Last Panel Events Callback”, let’s ignore this for now, I’ll explain later how to use this feature in the Skrptr Back Functionality.

IMPORTANT NOTE: At least one Skrptr Element must call on start the “SELECT” event for the system to have a starting point if you plan to use only the keyboard as input. Example:



2.6. Skrptr Animation (Abstract class)

Here’s the key feature that makes Skrptr useful: Once a gameobject has a Skrptr element, this will receive all the events listed in this documentation. The Skrptr element will then forward all the events fired on itself to all the Skrptr Animation components (Basically every component under Skrptr/Scripts/SkrptrAnim) which are added to this game object. Example below, On start the gameobject will select itself and turn the text (1) red over a duration of 1 second and with a .25 delay .



This allows us to hook any kind of animation to any event. (I feel somewhat proud of myself, perhaps I shouldn’t).

Most animations will have a duration, delay, and a target/tag to run the respective action on. Other properties might be available for more specific animations, such as move outside of screen, where direction is required.

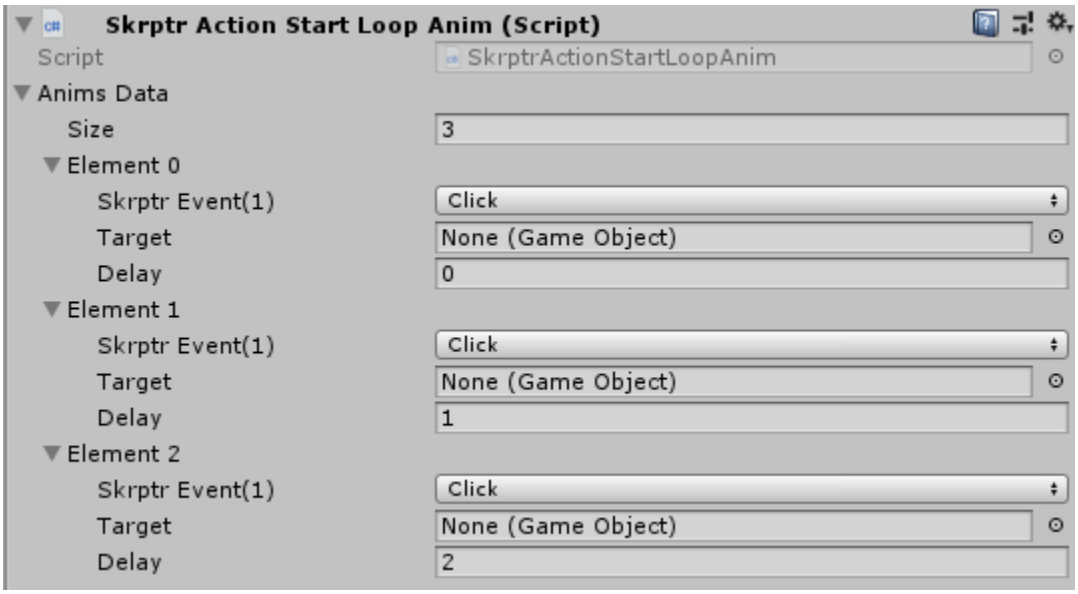
If the already implemented animations are not enough for your needs, feel free to drop me a mail at fairy_mental@yahoo.com or implement it yourself by creating a new script, inheriting from SkrptrAnim and overriding the methods. Tutorial on this will be coming soon.

2.7. Skrptr Action

Just like a Skrptr Animation, same functionality, just no looping possibility and no ease type. Use this if you require more technical functionality, such as sending a message, saving stuff in a database, saving / loading stuff.

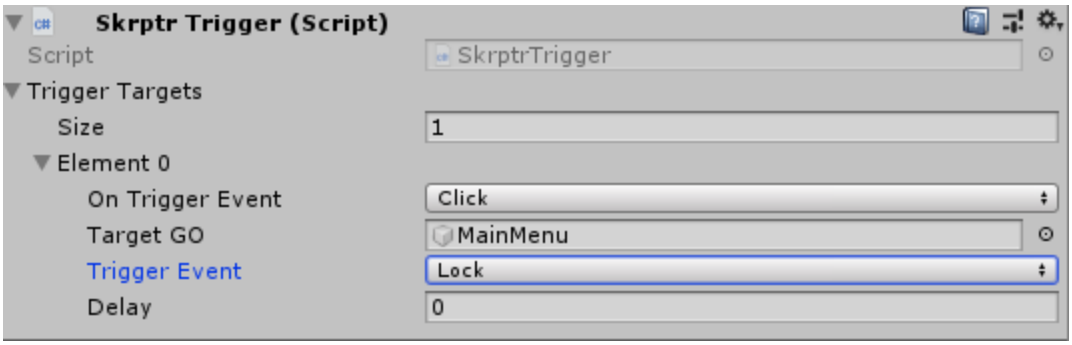
Theoretically, if one designer builds a whole UI using the components already available in Skrptr, he can pass the whole module to a software engineer. This will be able to hook actions to their respective UI elements just by writing new Skrptr Action classes.

Example: You can build a button on a gameobject that only has an image component by adding a Skrptr Element and a SkrptrAction. Whenever the SkrptrElement is clicked on, the action will trigger.



2.8. Skrptr Trigger

Skrptr Trigger is a Skrptr Action that allows you to chain events. For example, if the user presses the Close button, we will use a Trigger that will chain on Click the Lock event on the panel that it belongs to. Result: The panel closes whenever you close the button.



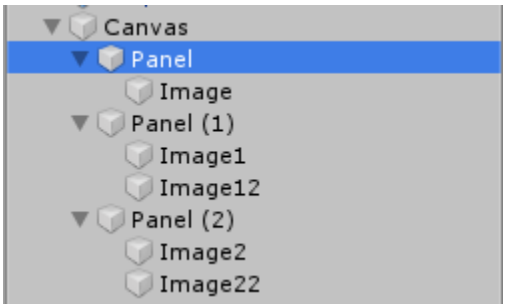
You can also set a delay for smoother transitions.

2.9. Skrptr Return to Last Panel

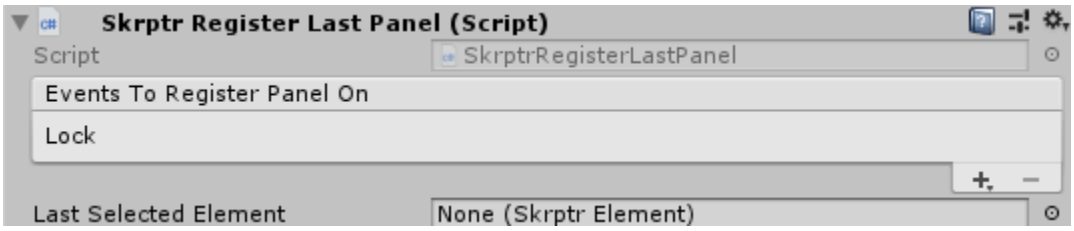
Hold on tight, this is slightly complicated.

If I were to implement a fully automated solution for “Back” it will be a complete chaos, and everything would break apart. Therefore, I decided to let the user decide which panels should register as a back possibility.

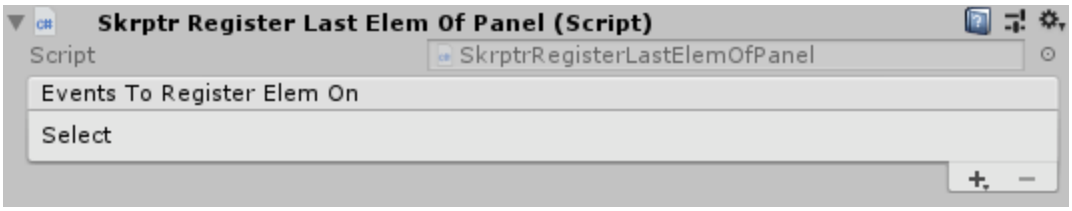
I will attempt to explain it based on an example: We have 3 panels, with some buttons in each. If we click any buttons in a panel, the next one should open, lock the previous, and if we press “Back” (Escape Key is default) it should lock the current, return to the previous. This is the hierarchy structure:



Every Panel has the “Skrptr Register Last Panel” component set to register itself as a Back Option whenever it is locked.



Every Element of the panels has the “Register Last Element of Panel” component, which will register itself as the last element selected of a panel, so that they keyboard mapper knows which element should be reselected on Back Input.



Result is something like this: https://gyazo.com/8448dceac1d27afb0806e012eaf53304

Input was: Enter -> Down -> Enter -> Back - > Back. As you can see in the GIF, we have returned to the initial panel.

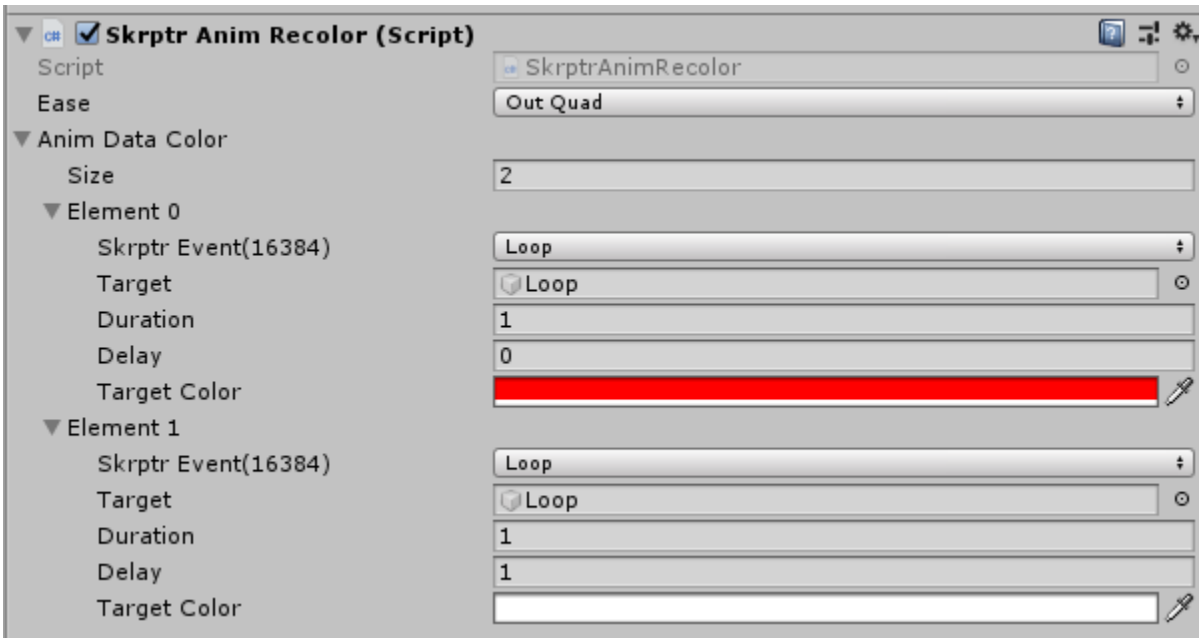
I will consider making a more in-depth tutorial on how to abuse this feature.

2.10. Skrptr Loop – Start & End

Skrptr Elements will check on start if it has any animations set on the Loop event, if they do, they will be indexed for efficiency reasons. Unlike other events, this must be called using the "SkrptrActionStartLoop", and "SkrptrActionStopLoop".

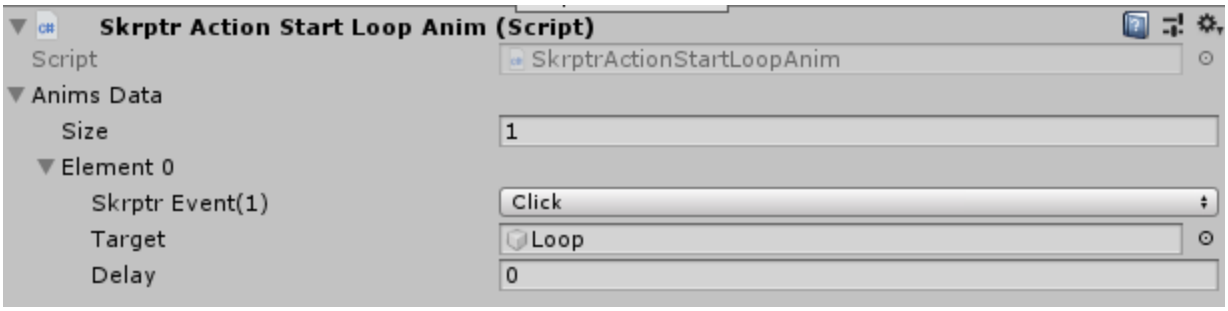
Most animations are loopable as long as you set them correctly (size, recolor, position, audio and even video).

Here is a simple example of a recolor that loops:

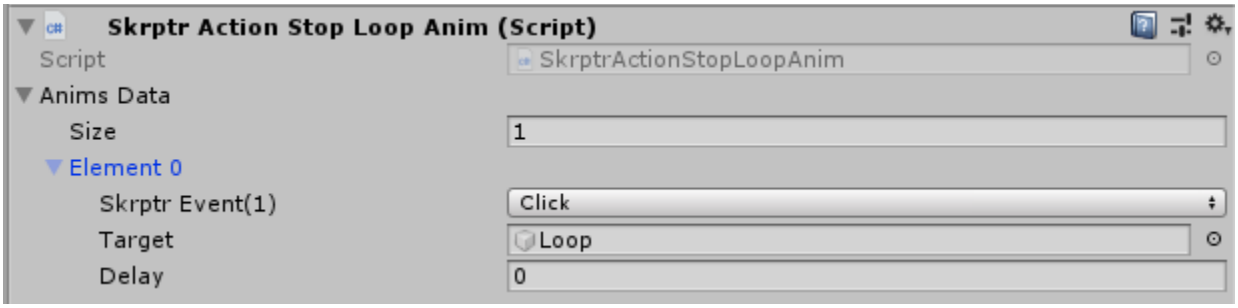


The Loop Image component will fade to red and back to white over a total duration of 2 seconds. Don't forget to set the delay if you wish this to not overlap and stay on the same color :).

On another SkrptrElement, we can add the SkrptrActionStartLoop and hook it up to whichever event we wish, this case, on click. If you want to start this right away as the scene starts, just add the Click Event to be called on Start up of this SkrptrElement.



If you want to stop it, add the Stop to any SkrptrElement and hook it to whatever event you want.



In our case, whenever this SkrptrElement is clicked, our looping animation will end.

3. Change log

Nothing much here, this will be the first release.

4. Contact

If you have any issues, advise or simply want to chat, hit me up with an e-mail at fairy_mental@yahoo.com or on discord: FairyMental#2558.

Best of luck further with your development! :)