Unity UI Expansion:

Getting started:

The easiest way to start is by creating a UI element from the Hierarchy menu. Right-click inside the hierarchy and select UI -> SlideToggle (or any other element), for example. Additionally, you can drag a UI prefab from the Prefabs folder into your scene, or you can add a component directly to your game object.

An exception to this is the Drag component, which should be added to a game object.

Drag Components:

There are three types of drag components:

1. SimpleDrag: This is a basic drag component that should work on any object with a transform. Please note that it has limitations.

2. SmoothUIDrag: This should be used for UI elements only. It provides a smoother drag experience specifically for UI objects.

3. SnapWindowDrag: This drag component prevents a UI object from being dragged outside of the canvas.

UI elements:

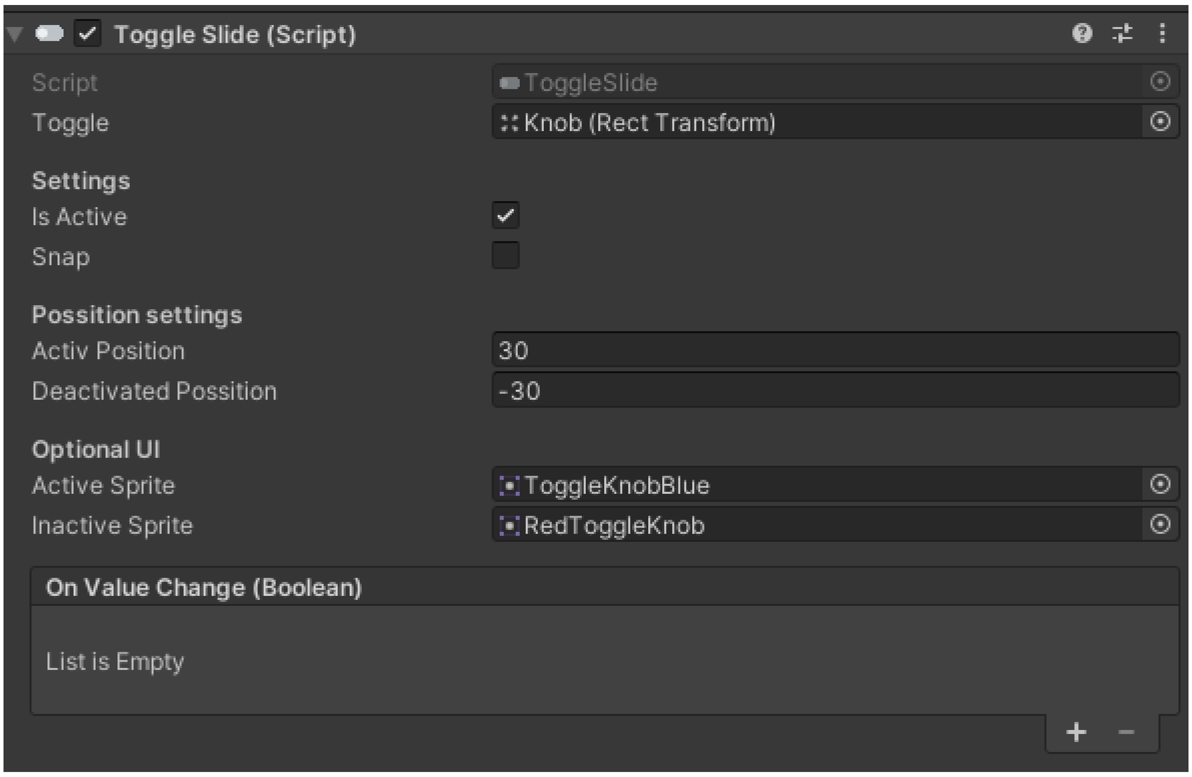
There are 6 UI elements : PopUp(TMP), SlideToggle,Carrousel(TMP), RadioBtns_Container(TMP), ProgresBar, CirclularProgresBar **(NOTE elements rhat have TMP int the name use TextMeshPro)**

PopUp:

This element is the simplest one. It basically serves as a UI placeholder for the usual popup. The only script attached to it is CloseBtn.cs, which deactivates the selected game object.

SlideToggle:

This element mostly works like a normal toggle with a slide animation. Animation is done with custom lerp function from **LerpHelper** class (containd in DIG.Tweening namespace).



Toggle Slide (Script)

Script	ToggleSlide
Toggle	Knob (Rect Transform)

Settings

Is Active	✓
Snap	

Possition settings

Activ Position	30
Deactivated Possition	-30

Optional UI

Active Sprite	ToggleKnobBlue
Inactive Sprite	RedToggleKnob

On Value Change (Boolean)

List is Empty

Parametars:

- Toggle – Rect Tranform of a moving knob
- IsActive - represents the active or inactive state of the toggle; the starting option of the toggle will represent this boolean
- Snap – if turned on toggle will snap to possition without the animation
- Possition settings:
    1. Active position - the position that the knob goes to when the toggle is active
    2. Inactive position - the position that the knob goes to when the toggle is inactive
- Optional UI - if these fields are empty, they simply won't apply:
    1. Active sprite - the sprite that the knob uses when it is active
    2. Inactive sprite - the sprite that the knob uses when it is inactive
- OnValueChange – event that is called when value of the toggle is changed

Scripting:

Properties:

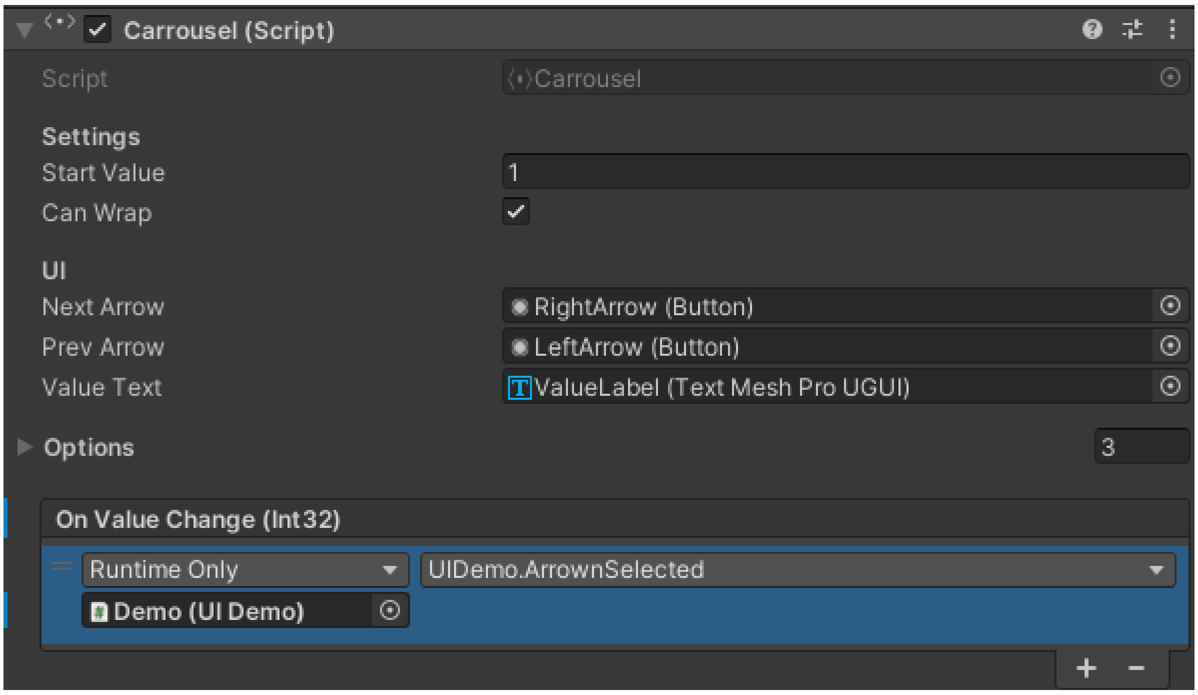
- IsActive - represents the active or inactive state of the toggle; the starting option of the toggle will represent this boolean
- AnimationDuration – duration of the knob slide animation

Methodes:

- **RefreshAndAnimate** - Refreshes toggle UI to current value (Useful if you chang value of the toggle somewhere else).
  Note this methode doesen't invoke OnValueChange!

Carrousel:

Carousel selector - a selector that allows you to choose a value from the options list. It functions similarly to a dropdown.


Carrousel (Script)

Script: Carrousel

Settings

Start Value: 1

Can Wrap: ✓

UI

Next Arrow: RightArrow (Button)

Prev Arrow: LeftArrow (Button)

Value Text: ValueLabel (Text Mesh Pro UGUI)

Options: 3

On Value Change (Int32)

Runtime Only — UIDemo.ArrownSelected

Demo (UI Demo)

Parametars:

- Start value – a int value of a Options list starting index (0 is the first option)

- Can Wrap – If set to true, the carousel values will wrap around, meaning that if you reach the last value and click forward, you will go to the first value. If this setting is turned off, after reaching the last option, you won't be able to go forward anymore.
- The UI section represents the user interface related to the selector. It is mostly self-explanatory.
- Options – list of string options inside the selector
- OnValueChange – event that is called when value of the carrousel is changed

Scripting:

Properties:

- Value – index of a scelected option
- Start value – a value that will be displayed at start
- Options – list of string options inside the selector

Methodes:

- **RefreshAndAnimate** - Refreshes arrow value without invoking an event
- **ClearOptions –** Clears options list

ProgresBar

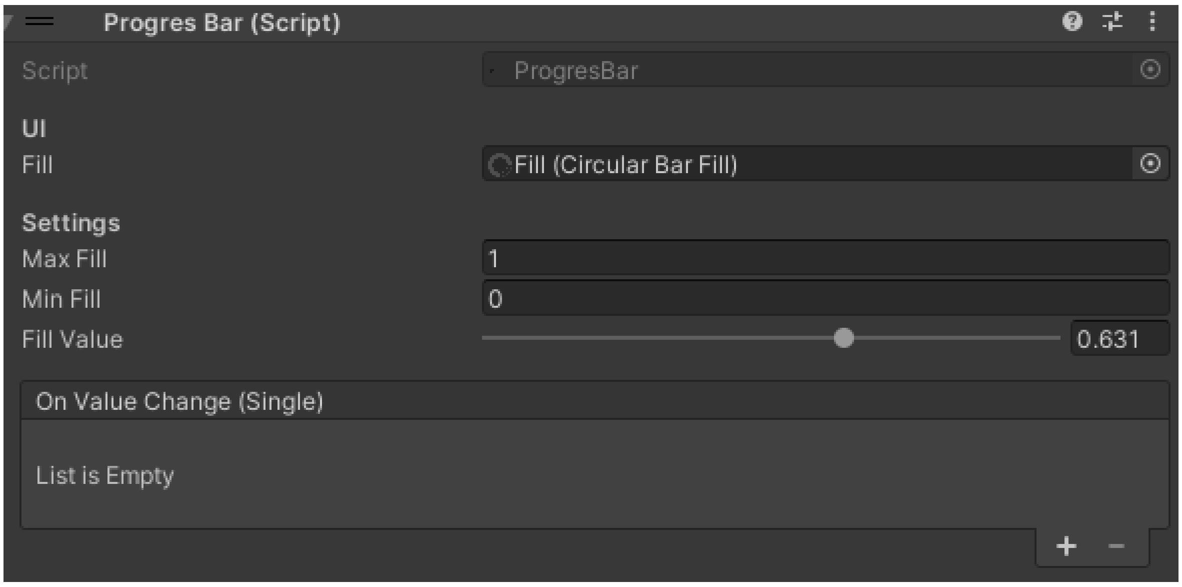
The progress bar is designed to replace a slider in use cases such as progress bars and health bars. It is essentially an image set to fill, but similar to a slider, it has a value that can be changed and variables for minimum and maximum values.

While there is only one progress bar component, there are two types that can be created from the hierarchy menu:

1. Standard Progress Bar: This is simply called "ProgressBar."

2. Circular Progress Bar: In this type, the fill method is set to radial 360, creating a circular appearance.

Note that image type and fill methodes are set in the coresponding components!


Progres Bar (Script)

Script: ProgresBar

UI

Fill: Fill (Circular Bar Fill)

Settings

Max Fill: 1

Min Fill: 0

Fill Value: 0.631

On Value Change (Single)

List is Empty

Parametars:

- Fill - refers to the type of fill that the bar uses. As mentioned, there are currently two types available. However, if want, you can also create your own fill type.

  To do so your script should inherit from **BarFill.cs** script and implement **InitilizeSliderBar()** methode.

- Max Fill – maximum value of the fill
- Min Fill – minimum value for the fill **(NOTE that min and max value always scale to 0 and 1 in the actual fill of an image)**
- Fill value – value of a progres bar that also fill the fill image, it can be change from editor or at runtime

- OnValueChange - event that is called when fill calue is changed

Scripting:

Properties:

- Value – value of a progres bar that translates to fiill of fill image.
  This value can be set to anything but it will always scale to 0 and 1 when translated to a fill of an image (for example if the value is 2 and max value is set to 2 than image fill will be set to 1 and if value is 1 and max value is set to 2 image fill will be 0.5)
- MaxFill – maximum value that Value property can be set to
- MinFill – minimum value that Value property can be set to

Methodes:

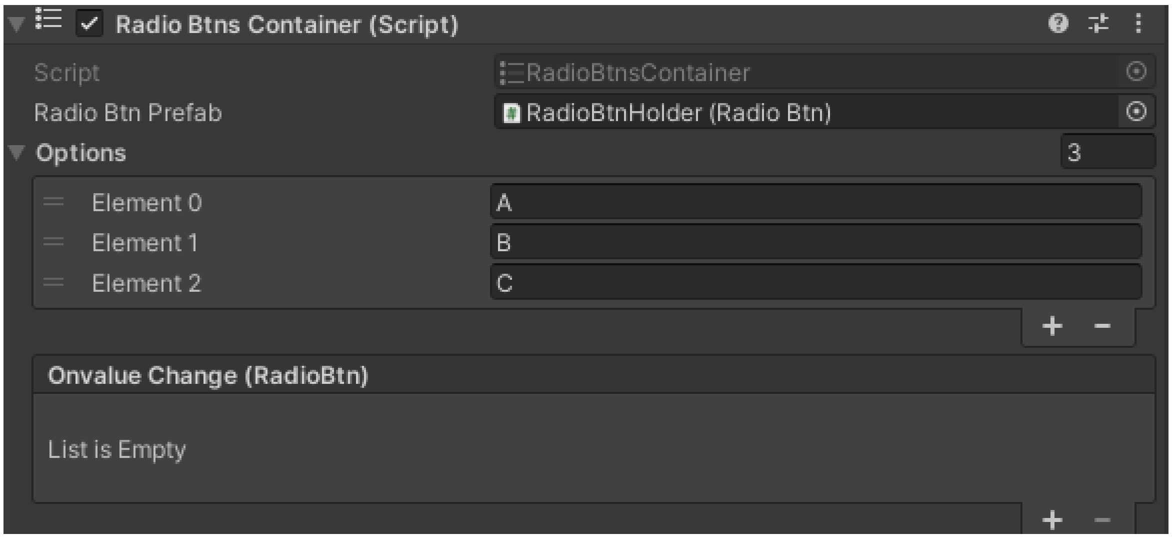
- No public methodes in the **ProgresBar.cs**
- **BarFill.cs –** a class that any new fill component should inherit from.
  1. FillImage – an image that will be filled by the ProgresBar
  2. InitilizeSliderBar() - abstract method that any class that inherits **BarFill** class needs to implement. Here you shoudl put fill settings of an component Like setting image type to fill and setting fill methode

Radio Buttons Container

This element consists of mainly two components: the container component and the radio button component.

The container component tracks the currently selected button and handles the OnValueChange event. It keeps track of which button is currently selected.

The radio button component contains properties associated with the button, such as button index and button label. It also provides set methods to update the button index and label.


Radio Btns Container (Script)

Script	RadioBtnsContainer
Radio Btn Prefab	RadioBtnHolder (Radio Btn)
Options	3
Element 0	A
Element 1	B
Element 2	C

Onvalue Change (RadioBtn)

List is Empty

Parametars:

- Options - refers to a list of string options that represent the labels of the buttons. The number of options in the list determines the number of buttons. Regardless of how many buttons you created in the edit mode, they will be destroyed and replaced by the updated buttons based on the provided options.
- OnValueChange - is an event that gets triggered when you select a button. This event takes in a **RadioBtn.cs** class, which contains all the relevant information related to the selected button.

Scripting:

Properties:

Properties for the Radio Button are located in **RadioBtn.cs** script but are accesed in **RadioBtnsContainer.cs**

RadioBtn.cs:

- LabelValue – value of the radio button text label
- Index – current button index
- IsActive – active status of the button

RadioBtnsContainer.cs:

- OnValueChange – an event that is called when a different button is selected. This event takes a **RadioBtn** as a parameter because that is where most of the relevant properties are.

Methodes:

RadioBtn.cs:

- SetIndex(int index) – sets current button index
- SetActiveStatus(bool status) – sets button status

RadioBtnsContainer.cs:

- SelectBtn(RadioBtn rb) – manages selection of the buttons, sets the activ one and invokes the OnValueChange.