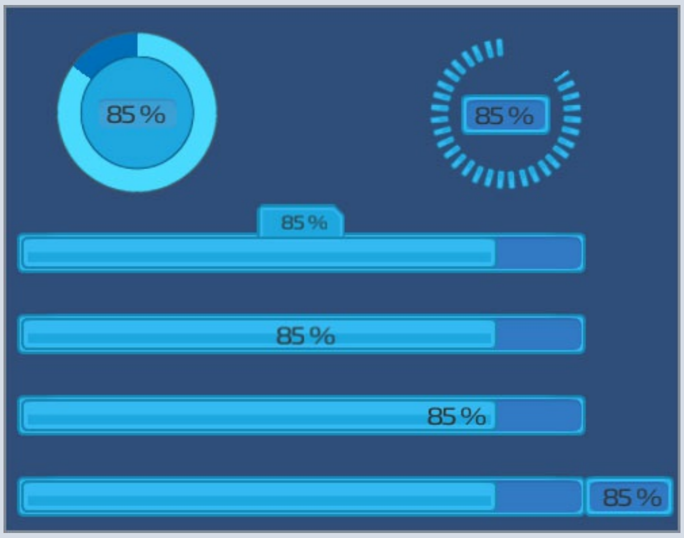
ProgressBar Scripts

Demo

The ProgressBar asset is made of two scripts dedicated to linear and radial progress Elements.
Using only three functions and the Unity GUI elements the result are powerful but easy to use UI Elements.

Art by Kenney

ProgressBarBehaviour.cs
ProgressRadialBehaviour.cs
Utils.cs



Functions

You do not need to use values that are not percentages (between 0 and 100) with these two scripts. These three first functions all need to take percents as parameters.
By default the percent value displayed is an integer, you can modify this easily to show decimals.

Value Property: this property will return the current ProgressBar value as a **percentage**. Setting it is the same as using *SetFillerSizeAsPercentage(percent)*.

```
//Get
float CurrentValue = Script.Value;
//Set
Script.Value = 45.5f;
```

IncrementValue: use this method to increment by X percent

```
//Value = 5f
IncrementValue(25f);
//Value = 30f
```

DecrementValue: use this method to decrement by X percent

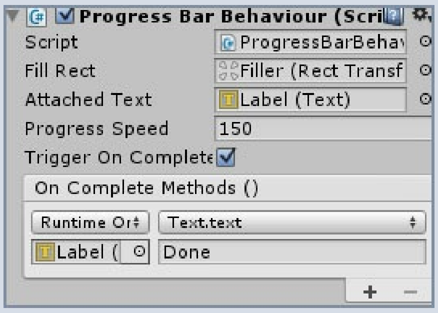
```
//Value = 30f
DecrementValue(25f);
//Value = 5f
```

OnComplete: if *TriggerOnComplete* is **True** the OnComplete methods chosen will be triggered when the ProgressBar reaches 100%.

IsDone and IsPaused: both boolean properties, IsDone returns true if the ProgressBar is complete (100%); IsPaused returns true if the transitory value is equal to the real value, meaning that the ProgressBar animation is not playing.

Unity Editor

Within the Unity Editor you may tweak three elements:



Progress Bar Behaviour (Scri

Script	ProgressBarBehav
Fill Rect	Filler (Rect Transf
Attached Text	Label (Text)
Progress Speed	150
Trigger On Complete	✓

On Complete Methods ()

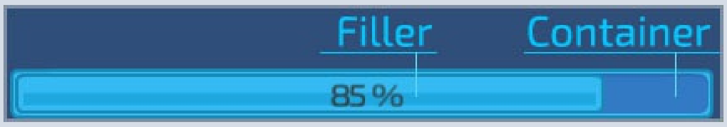
Runtime Onl	Text.text
Label (	Done

Attached Text: the Text element used to display the current ProgressBar's value. If null no text is displayed.
Progress Speed: the speed at which the filling animation is played. For the linear script it is the number of pixels per seconds, so it is relative to your element's width. On the contrary for the radial script, it is the percentage of the image filled per second. In the examples provided 150 for linear is similar to 0.35 for radial.
TriggerOnComplete and OnCompleteMethods: You can choose to trigger a method when your ProgressBar hits 100%, in the demo you can see the labels reading "Done" when finished.

Linear

How To

The ProgressBarBehaviour.cs script must be attached to the UI panel containing the Filler panel. This is because we are using the method **SetInsetAndSizeFromParentEdge** and we need to know the offset between the container and the filler.



Important: by default this offset places the filler in the middle of the container horizontally. The X offset is thus calculated has the difference in width between the two panels divided by two.
Before Runtime your filler must be full (at 100%), because its max width is set at runtime. The progress bar is set to 0% in the Start method, so there is no worrying about starting at 100%.

Prefab Examples

Four prefabs examples are provided for the ProgressBar script.

Radial

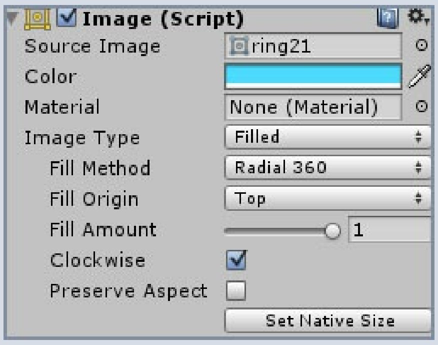


Image (Script)

Source Image	ring21
Color	
Material	None (Material)
Image Type	Filled
Fill Method	Radial 360
Fill Origin	Top
Fill Amount	1
Clockwise	✓
Preserve Aspect	

Set Native Size

How To

The ProgressRadialBehaviour script must be attached to the UI element containing the Image component that will be modified.
The point here is quite simply to set the FillMethod as Radial360 and to use the FillAmount to set how many percents of the image we want to display.
We don't care about any other properties this UI Element may have, the FillAmount is all we need. Just be sure that the image used has a design suitable for radial progress (is ring shaped).

Prefab Examples

Two prefabs examples are provided for the ProgressRadial script.

TOP