Documentation

Table of Content:

Introduction

Modular Bars package provides with tools to easily create progress bars for your projects without having to spend extra time coding for testing different animations for your bars. with 10+ different bar prefabs ready to use out of the box, it takes only minutes to add and set up a bar.

Creating your own bar with your design is also super easy with dragging and dropping scripts in inspector and setting them up right there. Code is performant and works even when not in Play Mode! Which makes it so convenient to test if your bar looks good and is working as your expect it to be.

> **Note:** This isn't limited to UI or even 2D. The Bars can be 3D or use shaders.
>
> **Note:** Moreover, Bars are not the only thing you can make with this package. if used creatively, it can be used to make anything that needs to be seeked within a range, like Day Night cycle, or Weather perhaps.

Basics of Modular Bars

Each Bar is divided into 3 parts.

1. Choosing Bar with either Normalized or Custom Max value
2. **What** the Bar will be effecting. using different Mods(Image fill, scale, Image color).
3. **How** the value of Bar translates for each Mod. Value Modifiers.

Lets learn about each of these step one by one.

---

1. Normalized vs Max Value Bar.

You can add Either `ProgressBar` (A bar with normalizedValue form 0 to 1) or `MaxBar` (A Bar with custom positive Max Value) Component to any gameObject to create a new Bar that you can interact with using their properties. Inspector shows a slider to change value for testing and for `MaxBar` an additional float variable to adjsut max value. if you try changing value of Slider, you will find out it doesn't do anything. why? we will know about that in next part of this documentation.

They will be the Base of the Bars so we will refer to both of them as `BarBase`

For a Normalized Bar(`ProgressBar`):

```
TryGetComponent(ProgressBar bar){
    bar.NormalizedValue=0.5f;
}
```

For a custom Max Value Bar(`MaxBar`):

```
TryGetComponent(MaxBar bar){
    bar.MaxValue=250;
    bar.NormalizedValue=0.5f;   //Sets value to 125.
    bar.Value = 130;        //Sets value to 130.
}
```

`MaxBar` can basically do everything a `ProgressBar` can, but not every bar needs to have a custom MaxValue. Still, its upto you to decide.

> *There are some special mods exclusive to Max Bar, which are explained in 2nd step.*
>
> *MaxBars can't be used as fragment for fragmentBarMod

---

2. Adding Bar Mods.

Adding the Bar script is pretty useless on its own because Bars Scripts don't know what to update when value is changed. That is where `BarMods` enter the scene. ~~Get it? unity scene?~~

Anyways, On the same object as `BarBase` (Either `ProgressBar` or `MaxBar`), You can add `BarMods` by either drag and drop the scripts or AddComponent *(ctrl+ shift + A)* then searching from inspector.

Each Mod controls some Component or GameObject and has different settings accodingly. Once you add the Mod it might try to autofill with components it needs, so you might want to provide it with right components from inspector before you can use the `BarBase`'s slider and see the changes happening in edit mode! yes. No need to go in Play mode to see how its looking at each value of bar.

> **Note:** AnimatorMod(Controls animator's exposed parameter) doesn't work in EditMode. Let me know if you know how to. Thanks.

`BarMods` must be added on same gameObject as `BarBase` but you can give them references to anywhere in the scene.

Example: `ImageFillMod`, It needs a reference to any UI Image Component in the scene with `ImageType` Set to `Filled`. *It will automatically Add first Image found in children if reference is null.*

You may add multiple `BarMods` and they all will be updated along with `BarBase`'s Value. Each of it has two variable in common which are `ValueModifier` and `UpdateWhen`. `VM(ValueModifier)` has its own section for explaination, for now Lets discuss `UpdateWhen` dropdown.

Bars can be set to only run when value is either increasing or decreasing using `UpdateWhen` dropdown which is available on all Mods.

There are a couple of `BarMods` exclusive to `MaxBar`:

- `FragmentMod`
- `DynamicFragmentMod`
- `RawNumbersMod`

These just require `BarBase` to have a `MaxValue` and updates accordingly when MaxValue is Changed.

---

3. Adding Value Modifiers.

`ValueModifiers` are scriptable Object instances that can alter the bar `Value` provided by `BarBase` to work Async or differntly than actual progress. (Like in discrete steps)

Value Modifier Instances can be added to assets by `Right-Clicking in Project Window > Create > ValueModifiers> Modifier`.

Using `ValueModifiers`, Mods can be interpolated, reversed, delayed etc; independently from each other.

Example: For a `ProgressBar` with an `RectWidthHeightMod` (Changes Width/Height of a RectTransform) and a `GradientImageColorMod`, gradient one can have its value delayed and Width Mod can have its value interpolated. When the bar Value is changed, the width will change linearly instead of snapping to the new value but color will snap to the new color after the set Delay.

We can divide Value Modifiers into 2 catagories: 1. Active. 2. Passive.

1. Active

Some value modifiers works Async to update Mods over time(or after a delay). They starts a Coroutine behind the scenes so they ***does not work in EditMode***.

> **Note:** Active Modifiers(Interpolation, Tween and Delay) does not work in Edit mode but still updates the Mods for a preview. So smash that Play button to see that in Action.

2. Passive

Others only does some maths and returns the updated value like reversing, reading a AnimationCurve or returns the value in descrete steps.

Chaining of ValueModifiers

Each Modifier has a serialized reference to another `ValueModifier` called `NextMod` so they can be Daisy-chained together.

> **Note:** Chaining to itself or creating loop in the chain isn't allowed. You can try, It won't work.

The updated value from the `ValueModifier` is passed to its `NextMod` and then `NextMod`'s `NextMod` and so on, until its `null` and passed back to the `BarMod` where it updates the Target Component.

Example: Having a `ReverseValueModifier` instance added as `NextMod` of `InterpolatedValueModifier` instance, reverses the interpolated result before giving it back to `BarMod`. Resulting, if `NormalizedValue` of `BarBase` is 1, the Mod will interpolate towards what it would be at 0(with its `ValueModifier` set to `null` (Left unset)).

Setting Up A Fancy Bar

Setting up a complex Bar for practice will make things clear.

First we need a Canvas all set up to put our Bar in. I like my canvas's Size Fitter component set to Full HD(1920 x 1080)

so you can change too if you'd like.

> All the prefabs are made with canvas fitter set to full HD, but you can change their width and height to fit your need.

1. Adding Image Fill and Gradient

Add an empty GameObject called "Bar" and add an UI Image as its child and adjust it to be where we need it, for our case, lets move Bar to top left corner. Then Change image to be Filled type and Name it "Fill Image".

Once Image is set Add `ProgressBar`, `ImageFillMod` and `GradientImageColorMod` component to it. Now if you scroll slider, you can see it already in action. You might want to change the gradient on `GradientImageColorMod` though.

Create a folder named "Value Modifiers" in Project window and Add an `InterpolatiedValueModifier` by *Right-Clicking in Project Window > Create > ValueModifiers> InterpolatedValueModifier* to the folder. Drag and drop this value modifier to `ImageFillMod`'s Value Modifier field.

It won't do anything in edit Mode, if you use slider, you will see its still the same but when you hit Play Mode, it will interpolates to new value.

Now the Image fill gets interpolated to its new value But Gradient color snaps to its new value. you can add interpolation VM to Gradient Mod as well to smooth color transition.

---

2. Adding An Fill Edge Image

The bar still pretty basic, and our goal is to make a Fancy one. Lets add some more Fancy effects.

1. Make the Fill Image to fill its parent size by setting anchors to:

anchor	x	y
Min	0	0
Max	1	1
pivot	0.5	0.5

Or you can do this:

2. Add a new Image as child of our Fill Image, Name it "Edge".
3. fit the image vertically setting its anchors to

anchor	x	y
Min	1	0
Max	1	1
pivot	0.5	0.5

and set the Width to Lets say 30.

4. then Drink some water, because thats also important.
5. Add 2 components to our Bar GameObject
    - `AnchorPositionMod`
    - `PivotShiftMod`

then set the new Edge Image obect as Rect Target for both of them.

What `AnchorPositionMod` does is move an object to it's parent's min to max. default is horizontal direction, and thats what we need.And `PivotShiftMod` changes the pivot from min to max, which we need so the edge's pivot is on left when `Bar.Value=0` and on right when Its on right edge of the image at `Bar.Value=1`.

6. Add another `GradientColorImageMod` and turn alpha to 0 on the edges like as follows.
7. Add interpolation VM to All `BarMods`.

Done. Check your new Bar, looking Fancier than before.

List of Mods

1. ImageFillMod
2. GradientImageColorMod
3. GradientImagesColorMod
4. ArcFillMod
5. CanvasGroupAlphaMod
6. ImageFlashMod
7. PercentageMod
8. AnchorFillMod
9. AnchorPositionMod
10. RectWidthHeightMod
11. PivotShiftMod
12. TranslationMod
13. ScaleMod
14. RotationMod
15. ImageSpriteListMod
16. BarEventsMod
17. FractionEventsMod
18. RendererShaderMod
19. CanvasShaderMod
20. FragmentBarMod
21. DynamicFragmentBarMod
22. RawNumbersMod

---

1. ImageFillMod

Controls Fill amount of target Image.

2. GradientImageColorMod

Changes Image color based on gradient provided in the mod.

3. GradientImagesColorMod

Changes color of list of Images based on gradient provided in the mod.

4. ArcFillMod

Radial fill Mod with limited Arc to be filed. starting angle rotates the whole object to start from a certain angle and `Arc Angle` defines how many degrees to fill.

5. CanvasGroupAlphaMod

Changes Alpha of target CanvasGroup from 0 to 1.

6. ImageFlashMod

Flashes target image, *frequency* times in *duration* seconds, before gradually turning to original color. Changes Canvas Renderer color instead of image so color multiplies together. Use a different overlay image to have white Flash.

> `ValueModifier` doesn't work with this mod.

7. PercentageMod

outputs current value in range of 0 to 100 on a TMP_Text(TextMeshPro). format of the output can be customized as following:

```
format = "HP ={0} %"
where {0} will be replaced by the value, giving output as "HP =30%" for NormalizedValue = 0.3f.
```

8. AnchorFillMod

Controls Anchors of a Rect to Simulate Fill. Lets you use sliced images without cutting off the edges as fill decreases.

9. AnchorPositionMod

Controls position of target rect from min to max of parent of rect's Anchors.

10. RectWidthHeightMod

Controls the height or/and the width of target rect.

11. PivotShiftMod

Shifts pivot from min to max for x or/and y.

12. TranslationMod

Lerps target transform's localPosition from `startPosition` to `endPosition` based on current bar Value.

Having same starting and ending value on an axis makes it not alter the axis so another TranslationMod can be added with different `VM` for different effect on other Axises.

13. ScaleMod

Scales target transform from startScale to endScale based on current bar Value.

14. RotationMod

Lerps target transform's localRotation from startRotation to endRotation based on current bar Value.

15. ImageSpriteListMod

Replaces UI Image sprite from given list of sprites based on current bar Value. where 0 index means full Bar. and last index means empty.

16. BarEventsMod

Unity events to trigger when given condition is met from the drop down menu.

Event	Triggers when

Event	Triggers when
OnEmpty	bar reaches 0.
NotEmpty	bar isn't 0 anymore.
OnFull	bar reaches 1.
NotFUll	bar isn't full anymore.

17. FractionEventsMod

Unity events to trigger when given condition is met. `Fraction` is compared to bar's current value and triggers the event once it has met, until the condition fails and then meets again. Choose `Comparitor` from drop down menu and set `Fraction` to compare against `NormalizedValue` of bar.

Comparitor	Triggers when bar's `NormalizedValue` is
LessThan	Less than `fraction`.
GreaterThan	greater than `fraction`.
Equals	equals to `fraction` now.
NotEqual	not equal to `fraction` anymore.

Example: If `Compartitor` is set to `LessThan` and `Fraction` is 0.4f, Event will only be triggered when bar value lowers below 0.4f from a greater value. If bar value was already under 0.4f, it won't trigger.

> To trigger the event(s), Condition must be false in last frame and must be true this frame.

18. RendererShaderMod

controls float value shader parameter of first material found in target Renderer. Name of parameter can be set in `exposedParameter` string in the mod.

> **Warning:** To avoid material leaking, it works on shared material in EditMode. It would change values for all objects with that material on. *Works just fine in Play mode though.*

19. CanvasShaderMod

Controls float value of a shader material taken from target CanvasRenderer. Name of parameter can be set in `exposedParameter` string in the mod.

20. FragmentBarMod

Takes a List of `ProgressBar`s(not MaxBar) and controls their values to be filled in sequence as a fragmented bar.

> ***Exclusive to `MaxBar`***

21. DynamicFragmentBarMod

Creates `count` Number of instances of `ProgressBar` Prefab under its GameObject, and updates them in sequence as fragment of same bar. count can be updated anytime.

> ***Exclusive to `MaxBar`***

22. RawNumbersMod

Works like `PercentageMod` but has MaxValue available in formatting as {1}.

for format = "{0}/{1} HP", where {0} will be replaced by the current Value and {1} by MaxValue, giving output as "30/100 HP" for `MaxBar` with `Value` of 30 and `MaxValue` of 100.

> ***Exclusive to*** `MaxBar`

List of Value Modifiers

1. **DelayValueModifier**

   Delays the mods functionality by *delay* seconds. if bar is updated again before *delay* ends, bar will snap to its current value before waiting *delay* seconds to update to new value.

2. **InterpolatedValueModifier**

   Linearly Interpolates value based on `speed`.

   if bool `relative` is set true, the magnitude of change in value will effect speed, making every change take same duration. while if `relative` is set to *false*, small changes will take small time and larger will take more time.

   > while using a `MaxBar`, if `MaxValue` is 100 and Interpolation(with speed 1) isn't set to `relative`(=false), then it will take 100 seconds to reach form 0 to 100. but if it is set to relative, then it will always take 1 second to reach either 0 to 100, or 0 to 1.

3. **GraphValueModifier**

   Translates the input Value to animationCurve's y axis, taking input on x axis.

   > Interpolation + Graph can give Tween like effects. and Graphs can even reverse Values.

4. **ReverseValueModifier**

   returns reverse of progress to the mod. i.e. MaxValue - CurrentValue;

   e.g. if the input is 0.1f, it returns 0.9f on `ProgressBar` and for `MaxBar` with `MaxValue = 10` it will return 9.9f.

5. **StepsValueModifier**

   Ceils the result to closest discrete step. `NumberOfSteps` defines the number of steps. ~~no need to thank me.~~

   e.g. if it's set to 5 steps and current value is 0.24f it will Ceil to 0.4f.

6. **~~TweenValueModifier (Requires DOTween)~~**

   > To Unlock, uncomment the top line of the script.

   Tweens value of Mod. Provides dropdown menu of DOTween Eases and duration of animation.

   Power of good ol' DOTween at your hands.
