PROGRESS BAR PRO

VERSION 1.3 - USER MANUAL

Health and Progress Bars Pro is an asset for Unity 3d to quickly generate good looking fillable linear or circular bars. These can be used as progress meters, health or energy bars among other things.

Features

- 15 pre-designed progress bars
- Smooth bar animation
- Preview of final value during animation
- Dynamic progress bar colors
- Flashing or animating bars on value change
- Easily customizable and extendable
- Full source code included

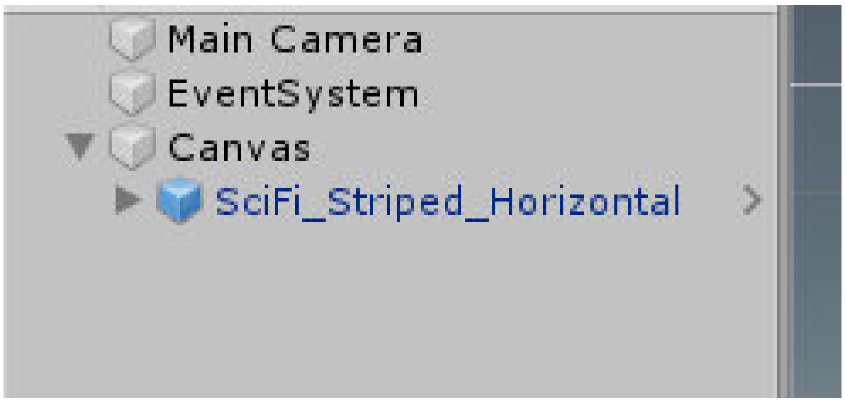
Table of Contents

SHARKBOMB

Getting Started

The asset contains multiple example scenes that demonstrate the variouos options and settings of the progress bar.

To add a progress bar to your game, the easiest is to pick one of the many example prefabs. Simply drag it into your scene. Since it the progress bars are making use of the UI system you need to make sure that the prefab is the child of an object with the Canvas component.



Then, to update the progress bar, you have to call one of the `SetValue()` methods from within your code somewhere. Here is an example:

```
public ProgressBarPro progressBar;
public int health;
public int maxHealth;

void UpdateHealth() {
   progressBar.SetValue(health, maxhealth);
}
```

Updating the Progress Bar

There are three different versions of the `SetValue()` method you can call to change the displayed value of the progress bar:

- `SetValue(float value, float maxValue, bool skipAnimation = false)`
- `SetValue(int value, int maxValue, bool skipAnimation = false)`
- `SetValue(float percentage, bool skipAnimation = false)`

Two Numbers

The two versions of the method with two numbers (two floats or two ints) are used to set the current value of the stat behind the bar and the maximum value of that stat. For example, if your progress bar is used as a health bar and that the player has a max health of 100 and has his health now reduced to 80. This means you'd call the method as `SetValue(80, 100)` to change the bar to display an 80% filled state.

One Number

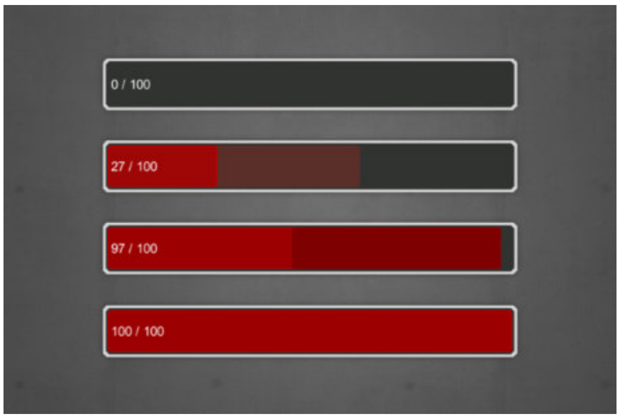
If you use the Method with the single parameter (float), then that number directly sets the fill percentage of the bar. This means the value will be clamped to a number between 0 and 1. In the example above, you'd call `SetValue(0.8f)` to set the bar to the 80% marker.

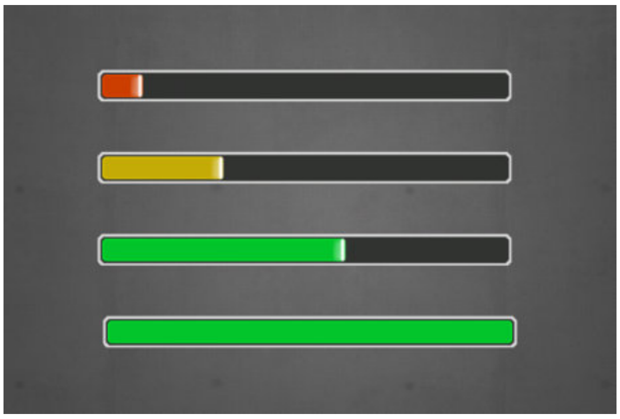
Skip Animation

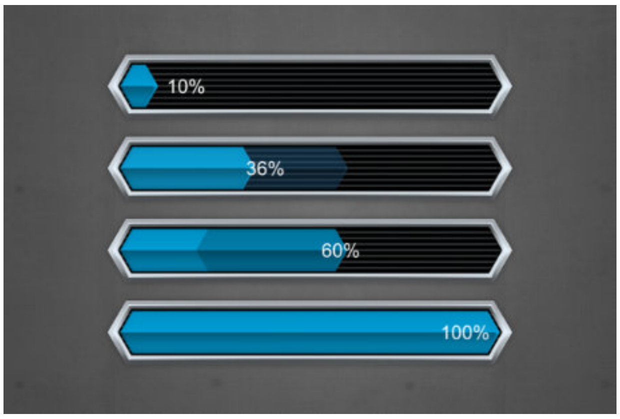
This optional parameter is `FALSE` by default. If you set it to `TRUE` when calling the method then the change you make to the progress bar will be made instantaneously, ignoring any animations the bar normally uses.

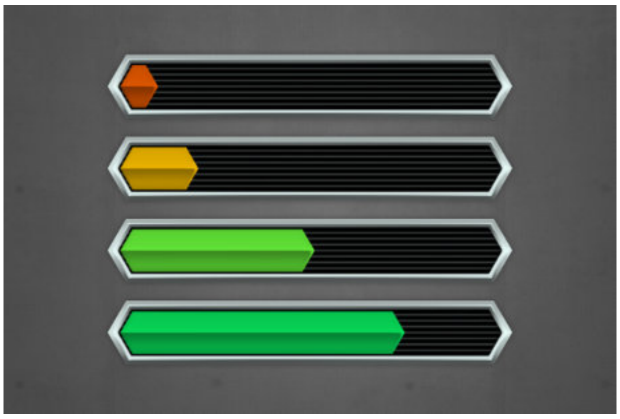
Included Progress Bars

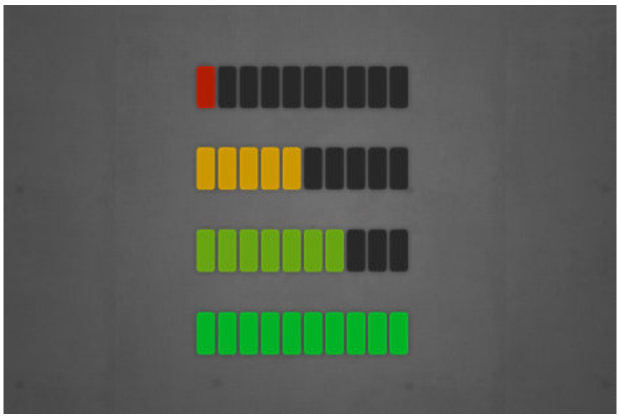
Below are some of the prefabs that are included with this asset:

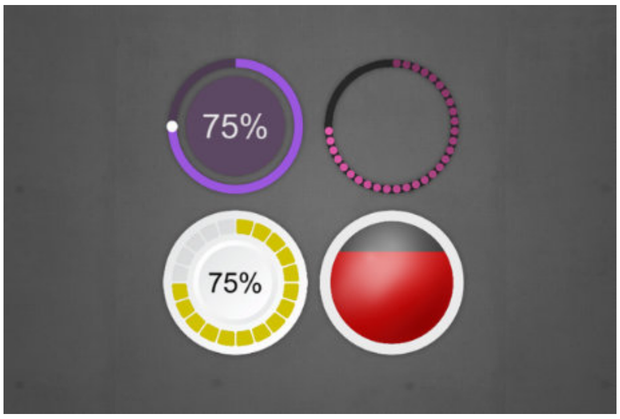












Building your own Progress Bar

At the heart of your own design is the **ProgressBarPro** script. This should be placed on the root GameObject of your design. By itself this script displays nothing, it simply takes the information given to it when `SetValue()` is called and uses it to drive other components.

These components are called **ProgressBarProView** and each of these views is updated when the value of the progress bar changes. If you're designing your bar, think of it as follows: anything that changes, be it a bar, the color, a text, is a view.

All you need to do is to apply the appropriate view components to your visual gameObjects. On the following pages you can find an overview over the included components but you can easily extend them by creating your own.

- BarViewColor
- BarViewColorWhileMoving
- BarViewPosAnchors
- BarViewPosImageFill
- BarViewScale
- BarViewSizeAnchors
- BarViewSizeAnchorsShadow
- BarViewSizeImageFill
- BarViewValueText

BarViewColor

This controls the color of the referenced UI graphic, such as text or an image. You will usually want to put this on your central bar element, unless your sprite is already colored and will not change.

- **Can Override Color**
  The progress bar has a convenience method called `SetBarColor()` that allows you to change the color of the bar. It will do so on all views that inherit from BarViewColor and have set this value to `TRUE`.
- **Default Color**
  This is the color of the bar. If your do not want to change the color then leave it white.
- **Use Gradient**
  Check this box if you want your bar to change color based on its fill state. It will use the following BarGradient instead of DefaultColor.
- **Bar Gradient**
  This is the gradient used to determine the color of the bar, based on its fill percentage. You can use the alpha of the bar gradient to have some elements only appear at certain value ranges.
- **Flash On Gain/Flash On Loss**
  If one of these values is set to `TRUE`, then the bar will be flash in a color for a short time after the bar's value was increased or lowered
- **Gain Color/Loss Color**
  These determine the color to flash the bar with when gaining or losing value.
- **FlashTime**
  This determines the length of the flash after a change.it's

BarViewColorWhileMoving

This changes the color of an image while the bar is animating or standing still. For example shis could be used to show a glow on the bar while it's moving.

- **Color Static**
  This is the color when the bar is not moving. You can set the alpha to 0 to hide the object on a static bar.
- **Color Moving**
  This is the color when the bar is animating.
- **Blend Time on Move / Blend Time On Stop**
  These determine how long it takes to change from one color to the other on starting to move or on stopping.

BarViewPosAnchors

This positions an object at the edge of a bar that is resized via its anchors (BarViewSizeAnchors). Its options are similar to BarViewSizeAnchors. It can be used to make a text move along the edge of the bar, for example.

- **Fill Type**
  This determines the axis and direction along which the bar will be resized.
- **Hide On Empty**
  If this is set to `TRUE`, then the gameObject will be disabled if the bar is at its minimum.
- **Use Discrete Steps**
  This means that the fill rate of the bar will snap to specific steps. This is useful if your image is segmented and you never want to show partial segments. See the "CircularDots" prefab for an example.
- **Num Steps**
  This determines the amount of steps used to fill the image.

BarViewPosImageFill

This allows you to position something along the edge of an image set to imageType Filled. It currently only supports Horizontal, Vertical and Radial360. This is useful if you, for example, want to position something at the edge of a circular fill progress.

- **Reference Image**
  This is the image that the object will stick to.
- **Offset**
  This determines how much the object will be offset along the fill line. Particularly useful if you want to align something to a circular bar.

BarViewScale

This component changes the scaling of the bar. It can either be used to scale the bar dynamically as its value shrinks or grows or it can animate the object's scale whenever the bar's value is changed.

- **Graphic**
  This is the RectTransform that will be scaled.
- **AnimatedOnChange**
  If `TRUE`, then the bar will animate whenever it's value is updated.
- **MinSize / MaxSize**
  This defines the minimum and maximum scaling values of the bar.
- **Scale Anim**
  This animationCurve defines how the bar scales based on its current value or the animation time. A value of 0 on the bar means it will be at its minScale, a value of 1 means it will be at its maxScale.
- **Anim Duration**
  If the bar animates on change then this value determines how long this animation takes.

BarViewSizeAnchors

This adjusts the anchors of the element to reflect the size of the bar. At 100% the object will fill out its parent on either the X or Y axis (based on your settings). This is good for bars that use sliced or tiled images.

- **Fill Type**
  This determines the axis and direction along which the bar will be resized.
- **Hide On Empty**
  If this is set to `TRUE`, then the gameObject will be disabled if the bar is at its minimum.
- **Use Discrete Steps**
  This means that the fill rate of the bar will snap to specific steps. This is useful if your image is segmented and you never want to show partial segments. See the "CircularDots" prefab for an example.
- **Num Steps**
  This determines the amount of steps used to fill the image.

BarViewSizeAnchorsShadow

This is a variation of the SizeAnchors build specifically for progress bar shadows: that show the target value of the bar while it smoothly animates toward that target. It has the same options as BarViewSizeAnchors with the following extra option:

- **Shadow Type**
  This can either be gaining or falling and determines how the anchors are picked and when the bar is shown or hidden. Gaining bars are shown when the value is increased, falling bars are shown when the value is lowered. Note that generally the gaining bar should be behind the actual visual bar, the losing bar should be in front.

BarViewSizeImageFill

This component is used in conjunction with a UI Image set to image type "Filled". The bar will change the fill amount to match its current value.

- **Hide On Empty**
  This disables the gameObject when the bar is empty. Note that this may cause some issues if the gameObject has child objects that would still have to be displayed.
- **Use Discrete Steps**
  This means that the fill rate will clamp to specific steps and not move smoothly between each step. This is useful if your image is segmented and you never want to show partial segments. See the "CircularDots" prefab for an example.
- **Num Steps**
  This determines the amount of steps used to fill the image.

BarViewValueText

This allows you update a Text with the value of the progress bar.

- **Prefix**
  This text is always shown at the start of the text field.
- **Min Value / Max Value**
  These are the min and max values between which the bar moves. If you use the bar as a health indicator and your maximum health changes, you may have to adjust max Value accordingly.
- **Num Decimals**
  This determines to how many decimals the values will be rounded.
- **Show Max Value**
  If this is `TRUE`, then the maximum possible value will be displayed.
- **Number Unit**
  If text is given, then the unit is added to both the current and the max value (if applicable).
- **Suffix**
  This text is always shown at the end of the text field.

Troubleshooting

I'm using SetValue but the bar isn't updating! Or: the bar is only updating slowly!

Are you're setting the value of the bar at a high frequency? For example you are calling `SetValue()` in every frame (from within an Update method)? If that's the case then you need to make sure to disable the animation on the bar.

Why? Because with animation enabled the bar eases and gradually moves towards its target value over time. If you set the value every frame the bar is stuck at the start of the movement over and over again making little to no progress.

Miscellaneous

Thank you for purchasing this asset and supporting future development.

Support

For additional support please turn to one of these channels:

- **Twitter**: @sharkbombs
- **Discord**: discord.sharkbombs.com
- **E-Mail**: support@sharkbombs.com