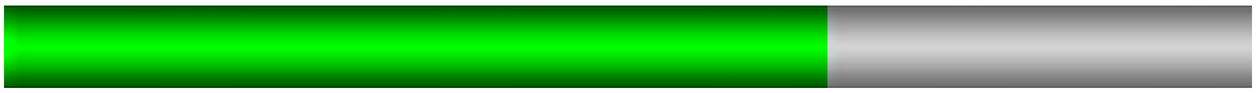
Progress Bar Pack

Manual



1 Overview

The Progress Bar Pack contains a selection of textures, shaders and scripts for creating different ways to visualize linear data, such as health, level load progress, points or player's progress towards the level end.

The textures and materials included are intented more as templates and examples than as final assetts to be used in a project, but of course if they happen to suit your needs as such without modification, great!

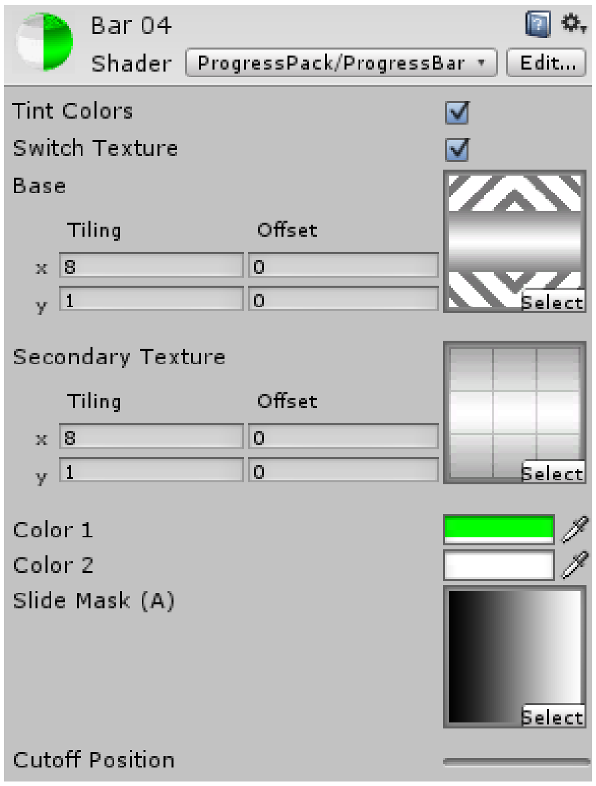
A demo scene is included (ProgressBars/Demo/DemoScene), you can play with it to find how out the various effects are made, or you can read this manual. Your choice.

A webplayer build of the demo scene can be found at
http://ifelsemedia.com/assetstore/progressbars/progressdemo.html

2 Shaders

2.1 ProgressBar / ProgressBar Transparent

ProgressBar shaders are generic progress bar shaders for drawing an unlighted image made of two separate parts that can have different color tint and different texture.



Bar 04
Shader ProgressPack/ProgressBar Edit...

Tint Colors ✓
Switch Texture ✓
Base
Tiling x 8, y 1; Offset x 0, y 0
Secondary Texture
Tiling x 8, y 1; Offset x 0, y 0
Color 1 🟩
Color 2 ⬜
Slide Mask (A)
Cutoff Position

The shader takes a mask texture (you can use the main texture here, if you wish to save memory and have the mask built in to the main texture's transparency), and uses it with a cutoff position propery to decide where to apply the two colors/textures.

The ProgressBar shaders have two options that can be set per-material: **Tint Colors** and **Switch Texture**. You need to have at least one of the options enabled for the shader to do anything sensible.

Tint Colors options opens up two colors on the material. Color 1 (“_Color”) is applied to the filled portion of the progress bar, and Color 2 (“_Color2”) to the empty portion.

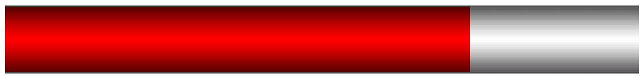
Switch Texture option opens secondary texture (“_SecondaryTex”) selection on the material. This texture will replace the main texture on the empty portion of the progress bar.

In addition, the shaders have the following properties:

- Base Texture (“_MainTex”) : Texture (RGB / RGBA)
- Slide Mask (“_SlideControl”) : Texture (A)
- Cutoff Position (“_Cutoff”) : Float (Range 0-1)

The transparency of the mask texture defines how the progression effect moves. A gradient that goes from fully transparent to fully opaque from left to right (as in Textures/SlideMasks/Horizontal) creates an effect that moves from left to right as the progress value (ie. the Cutoff Position property) increases.

Here’s an example of a simple health bar:



You can set the cutoff position either from the Position Cutoff property in the material editor, or from a script:

```
myHealthBarMat.SetFloat(“_Cutoff”, 0.7f);
```

Replace myHealthMat with a reference to your material (such as renderer.material), and 0.7f with desired value (0 being completely empty bar and 1 being completely full). The colors can also be set from a script, for example

```
myHealthBarMat.color = new Color(0,1,0,1);
```

would replace the first tint color with green, and

```
myHealthBarMat.SetColor(“_Color2”, new Color(0, 0, 1, 0.5f));
```

would set the second tint color to semi-transparent blue. The textures can also be set from script:

```
myHealthBarMat.mainTexture = myNewMainTexture;
myHealthBarMat.SetTexture(“_SecondaryTex”, myNewBackgroundTexture);
myHealthBarMat.SetTexture(“_SlideControl”, myNewMaskTexture);
```

So, for example, for the above health bar, inside a function for handling receiving damage you could do this:

```
float healthPercentage = currentHealth / maxHealth;
healthBarRenderer.material.SetFloat(“_Cutoff”, healthPercentage );
if (healthPercentage < 0.25f)
      healthBarRenderer.material.color = Color.red;
else
      healthBarRenderer.material.color = Color.green;
```

Setting the Tint Color and Switch Texture options can be set from script in the following way:

```
myHealthBarMat.shaderKeywords=new string[]{“TINT_ON”, “TEXTURE2_ON”};
```

replace _ON with _OFF to disable an option.

NOTE ON A BUG IN UNITY 4.1:

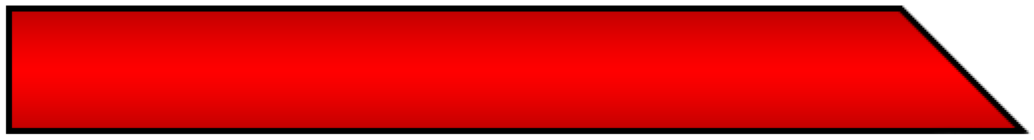
Sadly, there's a bug (or a "feature") in Unity 4.1, which in effect resets the "Tint Color" and "Switch Texture" options set in the editor if the properties in the material are modified at runtime. This is not a problem when you have both options enabled, as both options default to True. If this becomes a problem, the *most straightforward solution is to have both options enabled, and if secondary texture is not needed, simply assign the Base texture to it, and if color tint is not needed, just leave them white.*

All the included scripts that could be affected implement a workaround for this.

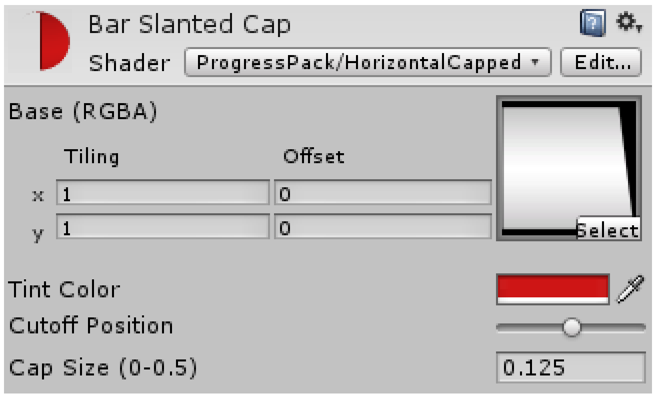
Why this happens: In Unity, when you modify a material at runtime, an instance of the material is created. Otherwise the instance is an exact copy of the original, with all your modifications applied to it, but for some reason shaderKeywords, which is used for storing the material options, does not get copied to the new material, as would be the expected behaviour. Hopefully this will be fixed soon.

2.2 HorizontalCapped / VerticalCapped

Capped shaders works much in the same way as the ProgressBar shaders, but are specialized in creating straigth horizontal or vertical progress bars with a cap at the end.



Capped progress bar shaders don't take a separate slide mask, but assume a direct left to right or bottom to up progression. The cap is included in the main texture, and it's proportional size is defined in the Cap Size ("_Cap") property of the shader. In the example below, the cap is ⅛ (or 12.5%) of the image width.



Bar Slanted Cap
Shader ProgressPack/HorizontalCapped Edit...

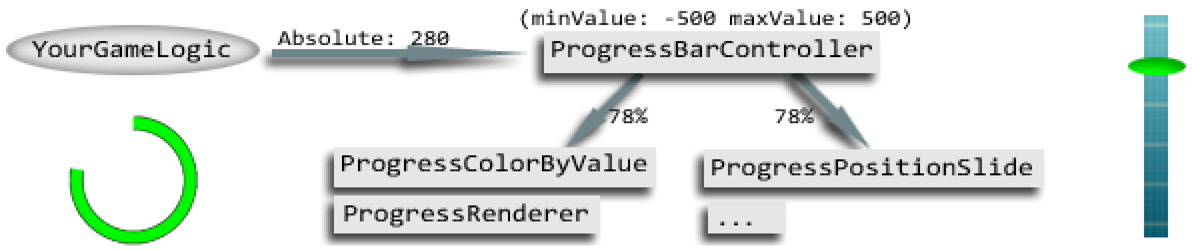
Base (RGBA)

	Tiling	Offset
x	1	0
y	1	0

Tint Color
Cutoff Position
Cap Size (0-0.5) 0.125

3 Scripts

A collection of helper scripts are included to help integrating the progress bars as smoothly as possible to your project. They follow the following usage pattern: a **Progress Bar Controller** is set up to handle the variable you wish to track, and one or more scripts are then set up to use that controller. You can have several progress bars getting their data from the same controller if you need to display the same data in several different places.



Simple setup procedure:

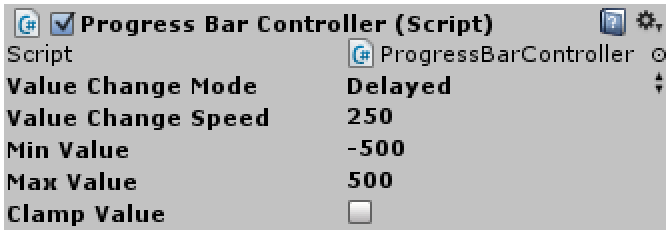
1. Add ProgressController01 prefab to the scene
2. Set the min and max values to match your data
3. Add ProgressBar prefab to the scene
4. Drag a reference to the ProgressController01 to the “Progression” field in the ProgressBar’s ProgressRenderer component
5. In your script holding the data you wish to visualize, set the Percentile or Absolute value of ProgressController01 to match your value.

   Example:

   ```
   public ProgressBarController progress;
   public float myValue;
   void Update()
   {
           progress.Absolute = myValue;
   }
   ```

3.1 Progress Bar Controller

Generic controller component for passing your data to the actual progress bar scripts. Using Progress Bar Controller you can have your scripts send the appropriate data to the Progress Bar Controller as either absolute or percentile values, and be absolutely agnostic about where and how the data is displayed.



Progress Bar Controller (Script)	
Script	ProgressBarController
Value Change Mode	Delayed
Value Change Speed	250
Min Value	-500
Max Value	500
Clamp Value	☐

Members exposed to the editor:

- valueChangeMode : enum ProgressBarController.ValueChangeMode
    - Whether the value is realtime or it changes over time.
- valueChangeSpeed : float
    - If valueChangeMode is Delayed, how many points per second will the value change towards the target number.
- minValue : float
    - The zero point for the tracked value. Any progress bars listening to this controller will be at their zero position when this value is reached.
- maxValue : float
    - The maximum point for the tracked value. Any progress bars listening to this controller will be at their maximum position when

this value is reached.

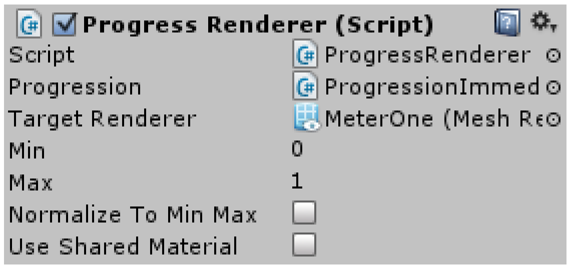
- clampValue : bool
    - Whether the tracked value is clamped between minValue and maxValue.

Public members accessible from script:

- OnValueChange : Event (float)
    - Listen to this event if you wish to react to the value changing.
- Absolute : float
    - The tracked value. If you write to this value, Percentile is changed accordingly, and OnValueChange event is fired.
- Percentile : float
    - The tracked value as a percentage between minValue and maxValue. If you write to this value, Absolute is changed accordingly, and OnValueChange event is fired.
- SetMinMax(float min, float max) : void
    - Call this if you need to change minValue or maxValue at runtime.

3.2 Progress Renderer

Modifies the _Cutoff property in the target renderer has one of the included shaders, and is connected to a Progress Bar Controller. Used for the typical progress bars.


Progress Renderer (Script)

Script	ProgressRenderer
Progression	ProgressionImmed
Target Renderer	MeterOne (Mesh Re
Min	0
Max	1
Normalize To Min Max	☐
Use Shared Material	☐

Members exposed to the editor:

- progression : ProgressBarController
    - The controller from which the component gets its values.
- targetRenderer : Renderer
    - The renderer to affect. If empty, uses renderer component on the same gameobject, if it exists.
- min : float
    - The _Cutoff wil not go below this.
- max : float
    - The _Cutoff will not go above this.
- normalizeToMinMax : bool
    - Whether the input value will be normalized between the min and mix values, or simply clamped.
- useShaderMaterial : bool
    - Whether to affect the renderers sharedMaterial, which will affect all objects using the same material. If left false, will create an instance of the material, and affect only that.

3.3 Progress Dial

Rotates a transform according to Progress Bar Controller's Percentile value. Used for dials, such as a speedometers.

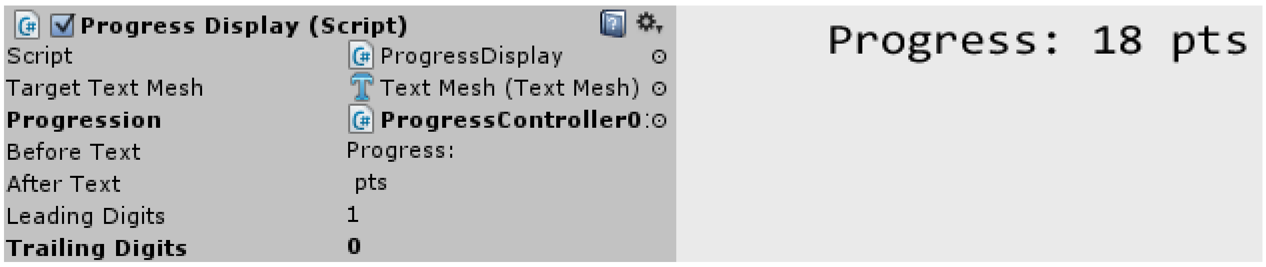
Members exposed to the editor:

- progression : ProgressBarController
    - The controller from which the component gets its values.
- needle : Transform
    - The target transform to rotate.
- rotationAxis : Vector3

    - Over which (local) axis is the rotation done.
- range : float
    - Range of the rotation in Euler angles.
- zeroPos : float
    - The Euler angle which represents 0% position. 100% position is zeroPos + range.

3.4 Progress Display

Displays the absolute value of a Progress Bar Controller on a textmesh with optional before and after strings. Useful for, among other things, debugging systems using Progress Bar Controller.



Progress Display (Script)

Script	ProgressDisplay
Target Text Mesh	Text Mesh (Text Mesh)
Progression	**ProgressController0**
Before Text	Progress:
After Text	pts
Leading Digits	1
Trailing Digits	**0**

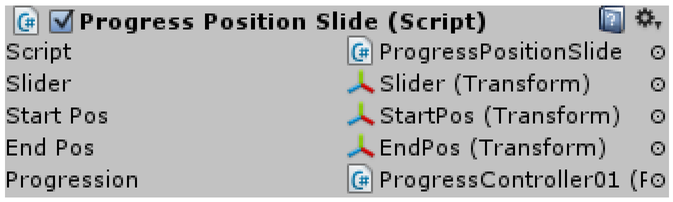
```
Progress: 18 pts
```

Members exposed to the editor:

- progression : ProgressBarController
  - The controller from which the component gets its values.
- beforeText : string
  - The text that is rendered before the value.
- afterText : string
  - The text that is rendered after the value.
- leadingDigits : int
  - The number of digits *before* the decimal point. For example, 3 leading digits will print out such values as 001, 010 and 100. Set leading digits to 1 if you don’t want any extra zeroes to appear.
- trailingDigits : int
  - The number of digits *after* the decimal point. For example, 3 trailing digits will print out such values as 0.001, 0.100 and 0.000. Set trailing digits to 0 if you don’t want any trailing zeroes.

3.5 Progress Position Slide

Linearly interpolates a transforms position and rotation between two transforms according to percentile value of a Progress Bar Controller. Used for example for showing player’s position along a linear track or moving a particle effect alongside a linear progress bar.



Progress Position Slide (Script)

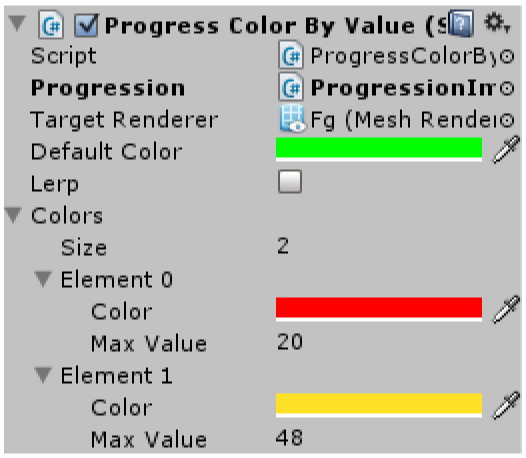
Script	ProgressPositionSlide
Slider	Slider (Transform)
Start Pos	StartPos (Transform)
End Pos	EndPos (Transform)
Progression	ProgressController01 (F

Members exposed to the editor:

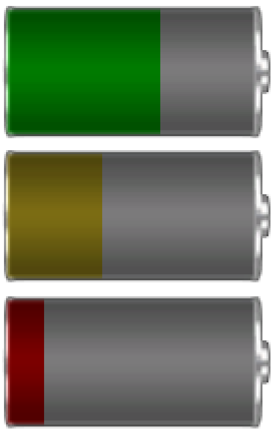
- progression : ProgressBarController
  - The controller from which the component gets its values.
- slider : Transform
  - The transform to affect. If left empty, will affect this gameobject.
- startPos, endPos : Transform
  - The positions between which to interpolate.

3.6 Progress Color By Value

Changes the color on a material based on the absolute value of a Progress Bar Controller. Used for example indicating when a value is dangerously low (like fuel bar turning red), or just by itself for visual effects that depend on some linear variable.



```
Progress Color By Value (S
Script              ProgressColorBy
Progression         ProgressionIn
Target Renderer     Fg (Mesh Rende
Default Color       🟩
Lerp                ☐
Colors
  Size              2
  Element 0
    Color           🟥
    Max Value       20
  Element 1
    Color           🟨
    Max Value       48
```



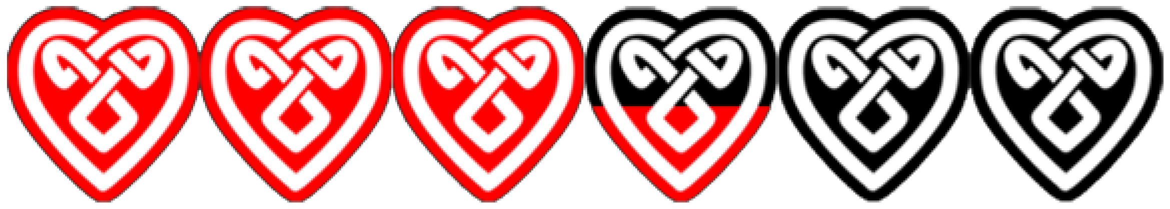
Members exposed to the editor:

- progression : ProgressBarController
  - The controller from which the component gets its values.
- targetRenderer : Renderer
  - The renderer to affect. If left empty, will affect this gameobject.
- defaultColor : Color
  - The color to use for the highest values.
- lerp : bool
  - Wheter to interpolate between color. If false, colors will change suddenly at the thresholds.
- colors : ColorValueData
  - The array of colors and the associated threshold values that is used. The first in the array is the color for the lowest values, and the rest are the colors in the ascending orders. Take care to set appropriate

values. DefaultColor is used when the value exceeds the maxValue of the last in this array.

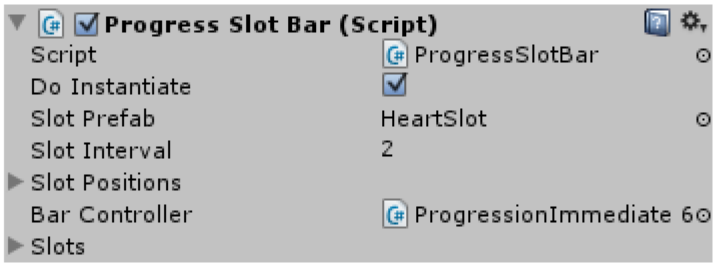
3.7 Progress Slot Bar

System for showing bars made of individual gameobjects. Each slot should have the ProgressSlot component, and the entire bar is controlled by a ProgressSlotBar component. Used for example for showing health or stars attained on a level. Each individual slot can have a substate value - in the example below, the input value is 3.5 (of maximum of 6), and the 4th slot currently has a value of 0.5.



There are three ways to set up the individual slots:

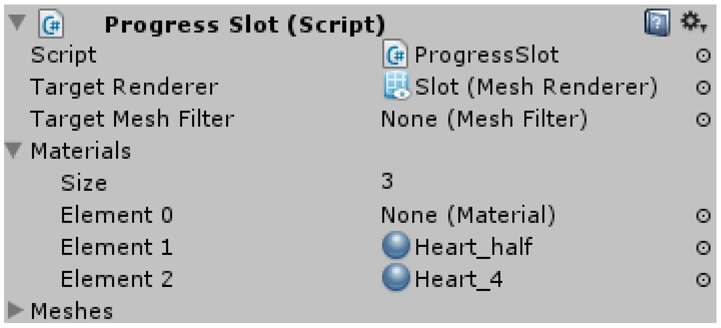
1. **Manually**. Uncheck the doInstantiate checkbox, create the slots in the editor, and assign them to the ‘slots‘ field list in the ProgressSlotBar.
2. **Instantiate from prefab**. Check the doInstantiate checkbox, set the ‘slotPrefab’ field, and set the ‘slotInterval’ field to a value that corresponds to the distance you wish the spawn positions to be apart from each other. The slots will be spawned along the ProgressSlotBar’s positive local x-axis.
3. **Instantiate from prefab to preset positions.** Check the doInstantiate checkbox, set the ‘slotPrefab’ field, and assign Transforms corresponding to the spawn locations to the ‘slotPositions’ field in the ProgressSlotBar. Make sure to have to correct number of positions defined.



```
Progress Slot Bar (Script)
Script              ProgressSlotBar
Do Instantiate      ✓
Slot Prefab         HeartSlot
Slot Interval       2
Slot Positions
Bar Controller      ProgressionImmediate 6
Slots
```

3.8 Progress Slot

Controller for an individual slot in a ProgressSlotBar system. Can be set to switch either materials or meshes to correspond to input value, and if the targetRenderer's material has _Cutoff property (ie. it uses Progress Bar Pack shaders), the property is properly animated.



```
Progress Slot (Script)
Script              ProgressSlot
Target Renderer     Slot (Mesh Renderer)
Target Mesh Filter  None (Mesh Filter)
Materials
  Size              3
  Element 0         None (Material)
  Element 1         Heart_half
  Element 2         Heart_4
Meshes
```

Members exposed to the editor:

- targetRenderer : Renderer
  - The renderer component that is manipulated. You can leave this empty if you do not intend to affect the slot's material in any way.
- targetMeshFilter : MeshFilter
  - The mesh filter whose mesh will be switched. Leave empty if you only manipulate the renderer.
- materials : Material[]
  - If targetRenderer is set, the material will be selected from these at

regular intervals. For example, if you have 2 materials defined, the first is used when the value is below 0.5, and the second when it's above.

- meshed : Mesh[]
    - If targetMeshFilter is set, the mesh will be selected from these at regular intervals. For example, if you have 4 meshes defined, the first is used when the value is below 0.25, and the second when it's below 0.5, third if it's below 0.75 and the last if it's greater than or equal to 0.75.

3.9 ProgressOnGUI

A class for drawing direct horizontal or vertical progress bars with or without caps on OnGUI calls.

Usage:

1. Create a ProgressOnGUI object, for example
   ```
   ProgressOnGUI myBar = new ProgressOnGUI();
   ```
2. Supply at least mainTex and fullArea values to the object
   If you want to connect to ProgressBarController, do so in the constructor.
   ```
   ProgressOnGUI myBar = new ProgressOnGUI(myController);
   myBar.mainTex = myBarTexture;
   myBar.fullArea = new Rect(10, 10, 100, 20);
   ```
3. Inside your OnGUI function, call myBar.Draw().
4. If not connected to a ProgressBarController, call myBar.Update(yourValueAsPercentage) whenever your value has changed.

The main texture is not stretched, but cut. If you want to render the progress bar so that the filled portion is tinted with one color and the empty portion with another, point both the mainTex and secondaryTex to your bar texture. If you only want to show the filled portion, leave secondaryTex empty. If your texture includes a cap (such as a rounded or slanted end), enter the width (or height if the bar is vertical) of the cap in the capSize field.

The public variables of ProgressOnGUI, in the order they appear in the 'full' constructor.

Variable	Type	Explanation
mainTex	Texture	The bar texture
secondaryTex	Texture	Secondary texture
capSize	float	Size of the cap in pixels
color1	Color	Bar tint color
color2	Color	Secondary texture tint color
progress	ProgressBarController	Controller to listen to
fullArea	Rext	Where to draw the bar
isVertical	bool	true=Vertical, **false**=horizontal
minValue	float	0-1, smaller than maxValue
maxValue	float	0-1, larger than maxValue
normalizeToMinMax	bool	whether to normalize input

Available constructors:

```
new ProgressOnGUI();
new ProgressOnGUI(ProgressBarController progress);
new ProgressOnGUI(Texture mainTex, Rect fullArea);
new ProgressOnGUI(
     ProgressBarController progress, Texture mainTex, Rect fullArea);
new ProgressOnGUI(
     Texture mainTex, Texture secondaryTex, float capSize,
     Color color1, Color color2, ProgressBarController progress,
     Rect fullArea, bool isVertical, float minValue,
     float maxValue, bool normalizeToMinMax
     );
```

4 Creating Custom Progress Bars

The most important factors in creating a nice looking progress bar are the texture(s) and the slide mask. A bundle of textures and masks has been included, but you might want to make your own, to accomodate your own unique visual style. The trickiest part is to get the slide masks right, so here's a couple of pointers:

- The slide control textures are gradients that go from opaque to transparent. The progression of the bar follows the gradient.
- The easiest way to make a mask is to make an image with greyscale gradient and check "Alpha from Grayscale" in the import options.
- Don't make the gradient go from pure opaque to pure transparent; instead, go from 99% opaque to 1% opaque (or from (254, 254, 254, 254) to (1,1,1,1)) or so.
- Import settings:
    - Texture type: Advanced
    - Generate Mip Maps: false (evaluate this on case-by-case basis)
    - Wrap Mode: Clamp
    - Format: Alpha 8

A demo scene is included, study it to see how the various effects were done.

5 FAQ

Q: Why use the ProgressBarController instead of setting the value directly?
A: It adds a nice layer of abstraction, allowing you to decouple your business logic from your visuals. Among other benefits, this approach makes code maintenance easier, and allows you to easily modify or change parts of your program later without it affecting other parts.

Q: Does this work for mobile?
A: Yes, it does. It has been tested on a 1st gen iPad and iPhone 4, where the demo scene ran smoothly and without problems, and there is no reason why it wouldn't run on other platforms as well. If you encounter mobile-specific problems, please send us email and we'll try to fix it ASAP. And as always, watch your draw calls and don't use transparency where you don't need it.

6 Contact

Progress Bar Pack is released for Unity Asset Store and maintained by Ifelse Media Ltd. We are happy to provide support for our products and to answer any questions.

Support email: support@ifelsemedia.com

Website: http://ifelsemedia.com/
Development blog: http://ifelsemedia.tumblr.com/
Facebook: https://www.facebook.com/IfelseMedia
Twitter: https://twitter.com/IfelseMedia
YouTube Channel: http://www.youtube.com/user/IfelseMedia

The developer responsible for this product is Timo Moisio, and can be contacted directly at timo.moisio@ifelsemedia.com

Do you think this pack is missing something? Don't hesitate to send us email, we'd love to hear from you on how we could improve this package!