Ultimate Circular Health Bars Light Guide and Reference

1 CONTENTS

2 INTRODUCTION

Thank you for downloading the Ultimate Circular Health Bars Light (UCHB) Asset!

This is the free version of UCHB. For the full version with over **60** properties and many more features, check out Ultimate Circular Health Bars!

This document is a small guide to help you get started with the asset and contains some useful information for scripting.

This asset REQUIRES URP/HDRP, so if you do not have either installed, then please install it by going to Package Manager > Universal Rendering Pipeline and install and import the package into your project.

If you imported URP into a pre-existing project, some assets might not show correctly. Please refer to this video if that is the case: https://www.youtube.com/watch?v=ErsXwcb3n4c

If you encounter any bugs with this asset, please contact devorengegames@gmail.com with the details of the bug.

Check out my other assets if you like this one: https://assetstore.unity.com/publishers/49336

I also have a YouTube Channel: DevOrenge - YouTube

I hope you get everything you need out of this asset and out of this document!

3 VERSION HISTORY

3.1 V1.0

The initial version of RSHB includes seven customizable properties for the shader. These properties are: *SegmentCount, RemovedSegments, Color, Spacing, Radius, LineWidth and Rotation*

4 FILES

The base Material can be found in the **Resources** folder of this asset. There you will also find a blank sprite. This sprite is used to ensure any SpriteRenderer you are trying to assign this asset to is able to be rendered correctly.

The **ShaderGraphs** folder contains the primary ShaderGraph and SubGraphs.

The **Scripts** folder contains the scripts used to populate a given GameObject with the correct Material and which allow you to animate the properties of the RSHB Material.

The **Prefabs** folder is where you'll find finished GameObjects for UI (Image) or non-UI (SpriteRenderer) purposes which can simply be added to your scene.

The **DemoScene** folder contains the demo scene and its corresponding assets.

5 USING THIS ASSET

This asset comes with an easy-to-use script which can be added to any GameObject which has a **SpriteRenderer** or an **Image** component attached to it.

Simply add the script **UltimateCircularHealthBar** to your desired GameObject and you are good to go!

If something goes wrong, you'll get an error message in the console and the script will remove itself from the GameObject. Everything should work fine, provided you have a SpriteRenderer or an Image attached to your GameObject and you haven't removed any essential files from the asset.

It currently isn't possible to duplicate a GameObject with this script attached, since the Material will be the same across duplicated GameObjects, meaning that when changing a value on one Material, the other Materials will change too. This is not the case in the full version.

The script has **7 public properties** which can be used for animation or as an easy way to set the RSHB Material properties. To use these, you simply need a reference to the **UltimateCircularHealthBar** component and then you can access them like:

```
public UltimateCircularHealthBar healthBar;

…

healthBar.<Property> = <value>;
```

There are some public methods to make setting the SegmentCount and RemovedSegments a little easier:

`healthBar.SetSegmentCount(<float value>);`

Set the number of segments in this health bar. Minimum for this is 0.

`healthBar.SetRemovedSegments(<float value>);`

Sets the absolute count of removed segments. The result is clamped to 0 and the Maximum value.

`healthBar.SetPercent(<float value>);`

Sets the absolute percentage (0 to 1) of the health bar. The input is clamped to 0 and 1.

`healthBar.AddRemoveSegments(<float value>);`

Add or remove (+ or -) a certain amount of segments from/to the health bar.
This does not alter the segment count. The result is clamped to 0 and the Maximum value.

`healthBar.AddRemovePercent(<float value>);`

Add or remove (+ or -) a certain percent (0 to 1) of the health bar.
This does not alter the maximum value. The result is clamped to 0 and the Maximum value.

The shader has **7 public properties**. The **properties** can be set like so:

`material.SetFloat(Shader.PropertyToID("<ShaderProperty>"), <value>);`

`material.SetColor(Shader.PropertyToID("<ShaderProperty>"), <value>);`

`material.SetVector(Shader.PropertyToID("<ShaderProperty>"), <value>);`

Following Notation:

<ScriptProperty> -> <**ShaderProperty**>

SegmentCount -> **_SegmentCount**

This is used to define how many segments the health bar has. This is a float value to allow for more flexibility, however, it is recommended to only use integers for this field to avoid funky looking health bars. Use a value of 1 to define a monolithic health bar with only one segment. Don't use 0…

RemovedSegments -> **_RemoveSegments**

How many segments have been removed from the health bar's total segment count. This is also a float value, here you can input something like 1.5 to remove 1.5 segments. It is also worth noting that if you have 1 set for the segment count this value acts like a percentage with 0 being 0% removed and 1 being 100% removed.

Color -> **_Color**

The color for the value part of the health bar.

Spacing -> **_SegmentSpacing**

The amount of spacing between each segment.

Radius -> **_Radius**

The relative size of the health bar. Values greater than 0.5 usually make the health bar exceed the bounds of the sprite.

Line Width -> **_LineWidth**

The thickness of the health bar. Setting a low value for the radius and cranking up the line width results in a pie looking health bar.

Rotation -> **_Rotation**

The rotation of the health bar in degrees.