SDF Robo Progress Bars 1.1

[DOCUMENTATION]

RELEASE VERSION 1.1

Robo Bars is a powerful master material usable for creating a wide range of customized progress bars. Package contains 12 predefined excellent styles easy to use for various types of games. Material supports advanced customization that covers most of the possible design requirements. Thanks to SDF functions created progress bars stays always sharp.

[Marketplace]
[Showcase/Tutorial]
[Forum]

Features:

- Signed distance functions based rendering. The shape is generated from mathematical functions that give quality of 'Vectorized' graphics quality without using huge resolution textures.
- Configurable for all resolutions.
- Fully scalable and extendable for more shapes.
- Can be mixed with SDF and standard textures.
- Rounded corners of shape colors patterns and progress bar showing.
- GPU friendly, one draw call.
- 12 example different themes that can be adjusted to requirements.
- Unlimited user-defined themes can be created easily using the available properties.
- Multiple advanced effects of progress bar burning/fade.
- Easy to integrate witch UMG

Roadmap:

- SDF shape used as bar progress.
- Point bar.
- Full mobile support.
- Example how to attach bar in 3d space.
- Example how to add second buff bar.

Release history:

Version 1.1 - Circle bar added. New 10 templates.

Version 1.0 - First release. Rect bar.

1. Basics (UMG vs Surface)

Robo Bar system is divided into two separate material instances that allow using this system on the surface as well as on UMG widgets.

All bars used in the game should be created as Material instance that derives from:

MI_SurfaceRectBar - material used on regular meshes (planes in 3d space). Check ExamplesSurface folder to see all examples prepared for surface bars.

UseInterface switch set false (for optimization.

MI_UmgRectBar - material used on hud controls (widget elements). Check ExamplesUmg folder to see all examples prepared for surface bars.

2. Transfer parameters between UMG and Surface

Sometimes there is need to change Surface material into UMG to render on hud without loss of all parameters set in the material. This operation is well supported by UE4. Open material that needs to be changed to support UMG/Surface mode, find option **General->Parent** and change to **MI_Umg*Bar/MI_Surface*Bar** depends on your needs.

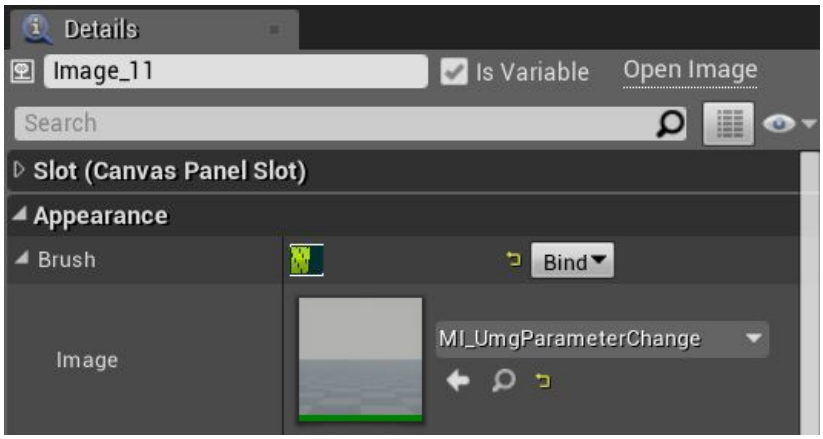
Explanation of UE4 mechanism. There are two differences between UMG and Surface materials:

	MI_SurfaceRectBar **MI_SurfaceCircleBar**	**MI_UmgRectBar** **MI_UmgCircleBar**
UseInterface	false	true
MaterialDomain	Surface	User Interface

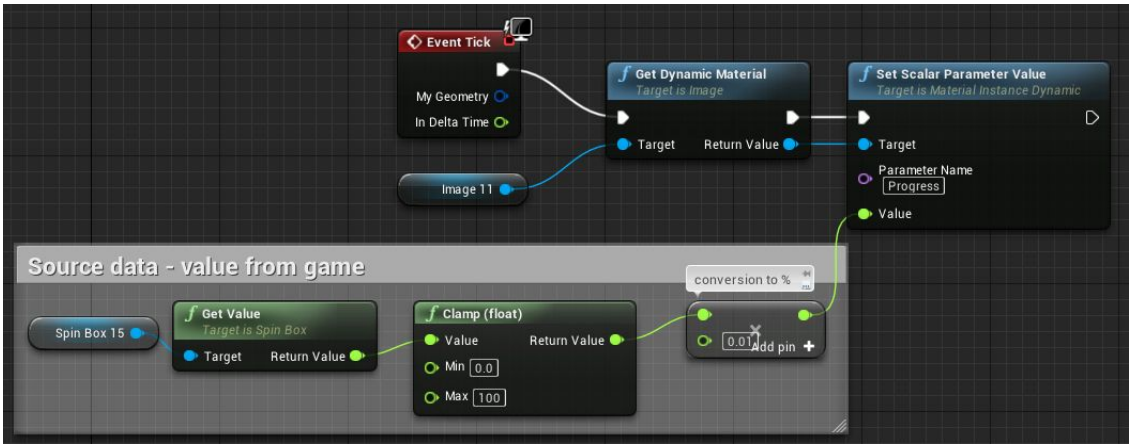
UMG does not support vertex interpolators so **UseInterface** need to be set false in materials used in UMG. Surface allows for vertex interpolators so it is used as optimization.

3. Using Robo Bars in UMG

3.1. The easiest way to use Robo Bars on UMG is to apply progress bar material onto **Widget Image** in parameter **Brush->Image** as presented below:

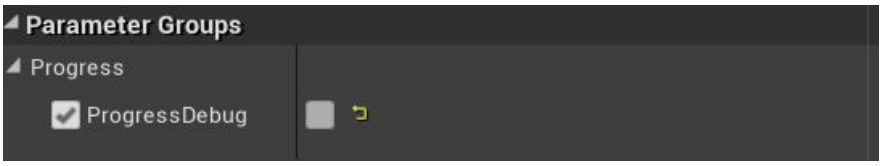


3.2. Next step is to setup parameter "**Progress**" of progress in blueprint graph. To do this you need to read material from the widget and set parameter value.



3.3. Progress is basic value parameter in master material used for setting how much progress bar is filled. Debug options override behaviour of progress bar by default to visualize the effect of changes in the preview. Find material that you used in your progress bar and disable the debug option by setting.

Progress->ProgressDebug = false.

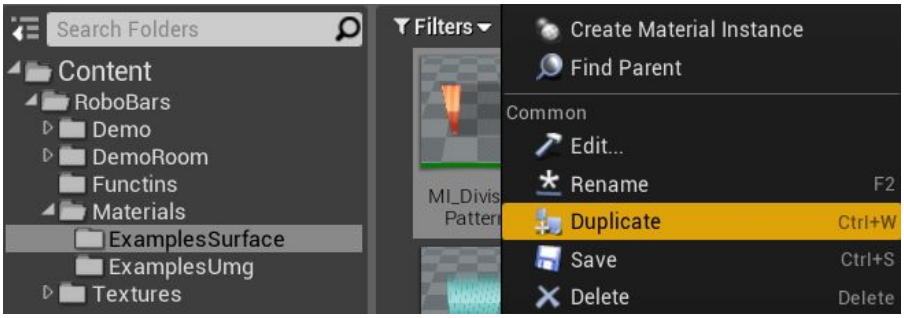


Simple example of using Robo Bars is presented in demo (Demo/Blueprints/BP_Hud)

4. Creating new bar

4.1. Select bar from group **ExamplesSurface/ExamplesUmg**. Find a bar that looks closest visual to effect that you would like to achieve (all bars share the same functionalities so can be modified one to another but it's faster to start from some point).

4.2. Create duplicate or material instance from selected bar.



4.3. Set the name for the newly created instance of material. It's good practice to start name from **MI_Surface or MI_Umg.** Open material and modify parameters.

5. Signed Distance Fields

Signed Distance Field is an image where each pixel contains the distance to the nearest point on the boundary. An additional sign of distance allows determining if the pixel is inside or outside rendered shape. SDF image that looks like gradient can be loaded from a file or generated by mathematical functions called Signed Distance Functions.

The same method is used to render outlined/smooth high-quality fonts in Unreal Engine 4 based on low-resolution textures.

6. Master material parameters

The progress bar is divided into 4 basic layers that are combined into one by translucent blending. Each of layer is configurable and implements some additional effects.

Layers

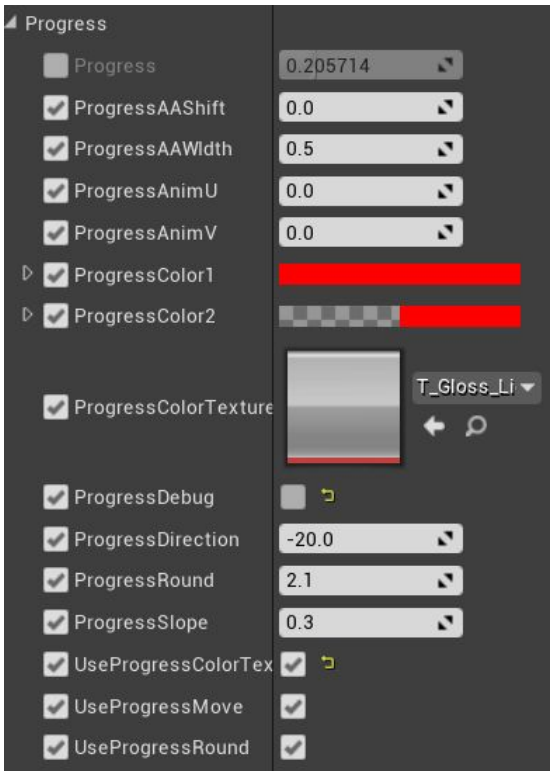
1. **Outline layer**- base background shape allows to setup outline color of the shape.

UseOutline - Swith determinates weather the bakground outline should be rendered. When disabled then background is filled by one clolor.

OutlineColor - Color of outline/background layer. To make it fully inviisble change color alpha to 0.

2. **Shape layer**- shape of bar is represented by combined signed distance functions.

3. **Progress layer** - layer used for drawing current progress (how much shape is filled by color/texture/effect).



Progress - parameter used to steering fill of the progress bar.

ProgressAnimU/V - sets animated UV of progress bar texture.

ProgressColor1/2 - sets separate colors of the pattern used on the bar.

Use Progress Collor texture- Allow to use progress color texture.

ProgressColorTexture - Texture of progress bar sampled based on ProgressAnimU/V. The final color of the bar is calculated by the multiplication of Colors and Texture. This option is useful for an additional effect like gloss/shadow.

Progress Debug - Debug options override the behaviour of progress bar by default to visualize the effect of changes in the preview.

Progress Direction - Two functionalities in one parameter. Allow changing the direction of progression by setting positive-negative value. The negative value means that bar will be filled from left to right and the positive from right to left. The absolute value of this value represents the blending of progress bar into the shape layer. Big value = hard edge/small value = soft edge.

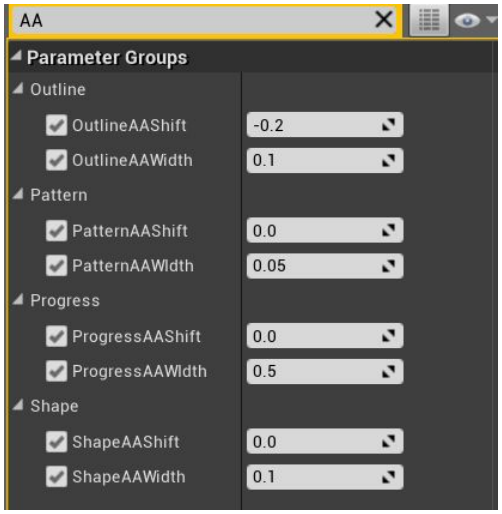
Use Progress Round - Allow to use round corners on the progress bar.

Progress Round - Represents the roundness of corners.

Use Progress Move - If true then progress texture will be moved when Progress parameter changes.

4. **Pattern layer-** divisions layer describe scheme of division the bar on two colors.

Edges Anti Aliasing



Each of layer contains parameters of edge smoothing that allow to set up hard or soft blending between lower and higher layer. List of parameters:

Width - Range of smoothness. Bigger value allows making smooth edges of the layer.

Shift - Shift related to zero position on SFD. Determinates the edge size of the layer. Can be used to set up outlines.

7. FAQ and known issues

[Would be nice to have a second layer of progress behind existing one controlled by another parameter. Something usually called 'buffer'.]

The easiest way to do this in my system is by creating two bars:

- Background bar with buff.
- Foreground top layer bar where: Outline color and shape Colors are set to fully transparent. (only progress will be rendered) I will think about something more convenient to merge this two bars in one :)

[Just bought your pack and i can't find the updated content]

Please report about this to me at krystian.komisarek@gmail.com. Probably something went wrong with update and i didnt notice this. I have to contact with epic to solve this problem.

[I Have a question ...]
Cool! You can always ask!. I'm ready to support and add some new useful features to my product. Feel free to comment and rate my package too :)

Forum topic:
https://forums.unrealengine.com/unreal-engine/marketplace/1412672-sdf-robo-progress-bars
Support mail:
krystian.komisarek@gmail.com