Procedural UI Image

Documentation

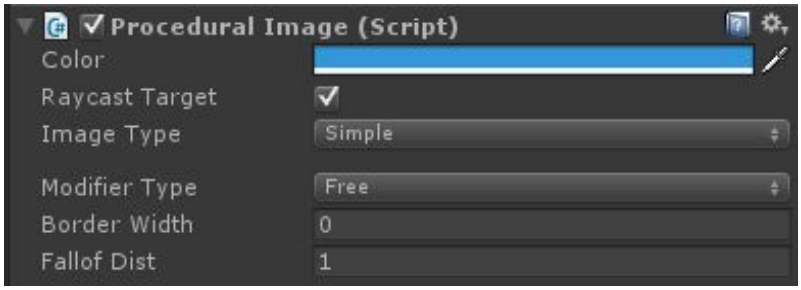
Product Overview

Procedural UI Image is an extension of the standard Image component of Unity UI. Procedural Image component does not use an image sprite, but instead renders a sprite in real time with given parameters. It is inspired by the styling possibilities of CSS & HTML for boxes. It is perfect for creating flat design styled UI without all the sprites created in an Image Editor outside of Unity. Another great thing about **Procedural UI Image**: You can animate the Properties like Border-Width or Border-Radius to create cool looking effects for an interactive UI.

Components:

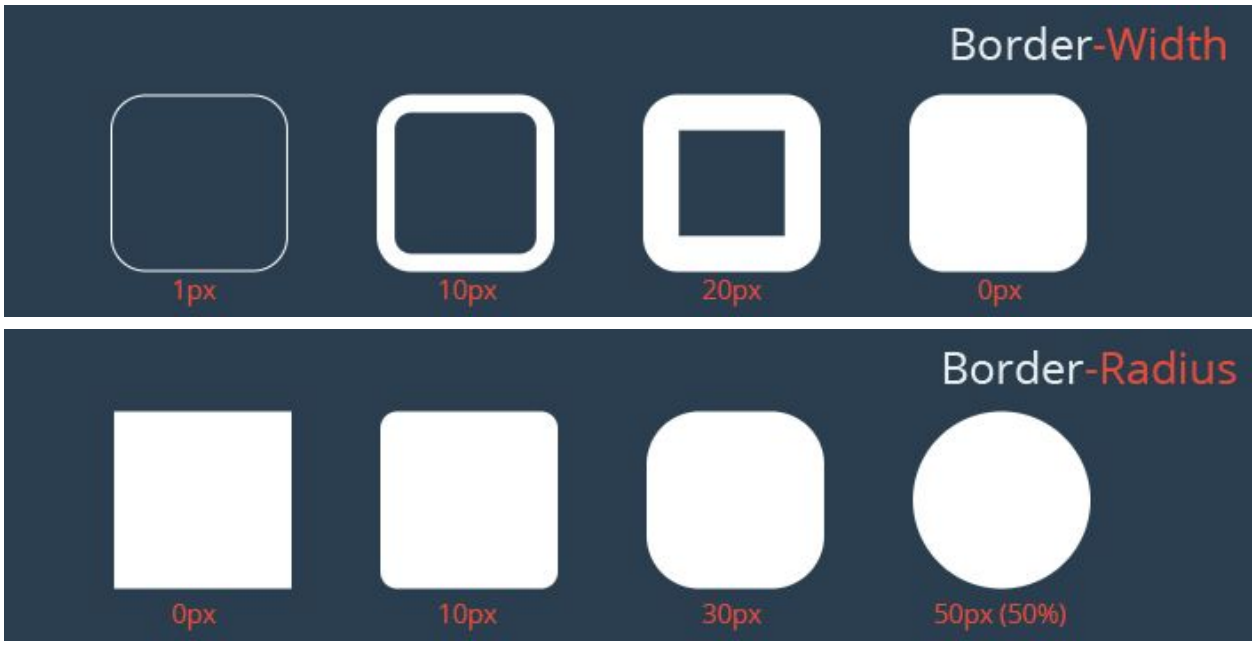
To use **Procedural UI Image** simply import the **Procedural UI Image** package into your Project. All files of this package can now be found in *"Assets/ProceduralUIImage"* folder. To add a procedural Image right-click in the Hierarchy window and choose *"UI/Procedural Image"*.

Procedural Image component



Properties:

- **Color, Raycast Target:** Work the same as in UI Image component.
- **Image Type:** *Filled* and *Simple* are identical to UI Image. *Sliced* and *Tiled* have no effect and will result in the same behaviour as *Simple*.
- **Modifier Type:** Current modifier type. For more Information on modifiers see below.
- **Border Width:** Controls the thickness of Outline in pixels. A value of 0 results in a solid filled graphic.
- **Falloff Distance:** The distance of linear falloff around the edges. The default value of 1 gives a nice sharp but antialiased look. Values greater than 1 can be used to create blurred shadow effects.

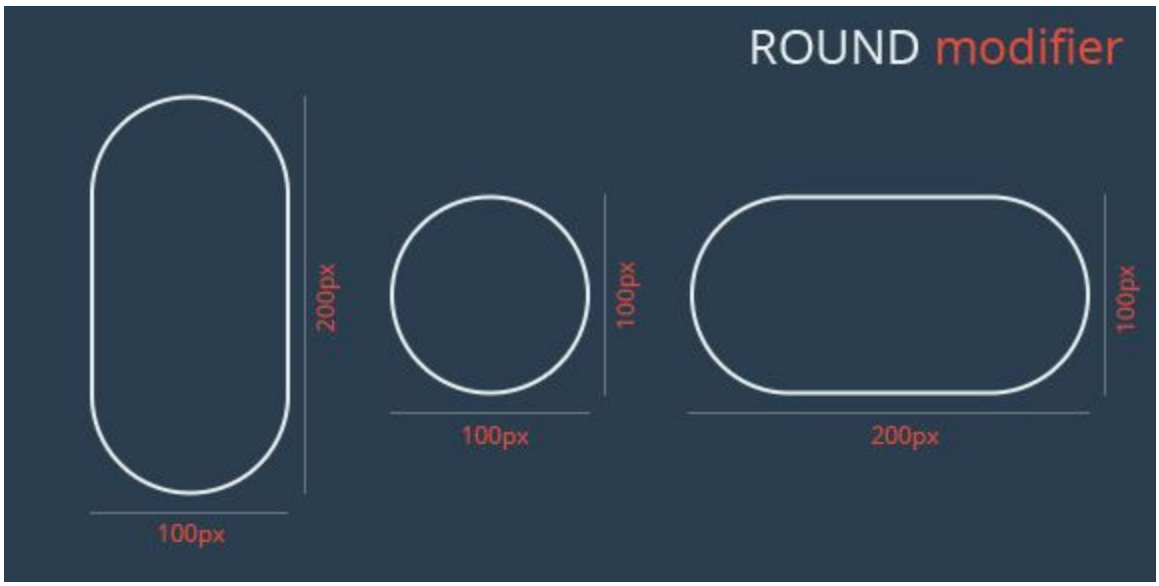


Modifier component

Modifiers are a flexible and powerful way to control the **border radius.** The most basic and default Modifier is the Free-Modifier. If you want to define your own Modifiers, you can! See in “Create custom modifiers” below.

The built-in Modifiers are:

- **Free:** It simply lets you control all 4 border-radiuses individually.
- **Only One Side:** Sets same radius for two corners of same edge.
- **Round**: Uses half width or half height as the border-radius. The Image will always have round caps. If Image is square shaped, Round modifier will give a perfect circle.
- **Uniform**: Sets the same border-radius for all four corners.



Create custom modifiers

[At least some beginner knowledge at programming needed]
Extending the functionality of **Procedural UI Image** by scripting new modifiers is easy to do.

```
using UnityEngine;
using System.Collections;

[ModifierID("Modifier Name")]
public class CustomModifier : ProceduralImageModifier {
	#region implemented abstract members of ProceduralImageModifier
	public override Vector4 CalculateRadius (Rect imageRect){
		//Do whatever math you want
		//Return some Vector4 with the border radiuses.
	}
	#endregion
}
```

A Modifier is a class that derives from **ProceduralImageModifier** and must implement a public method **CalculateRadius(Rect** *imageRect***)**. The parameter *imageRect* contains the position and size of the Procedural Image. You can use that information to calculate a radius relative to the image size (as Round-Modifier does). **ProceduralImageModifier** derives from **MonoBehaviour** so you can use the standard methods like Update() and Start().
Important: The returned Vector4 (x,y,z,w) contains the four radiuses in pixels, where **x** is the upper left, **y** the upper right, **z** the lower right, **w** the lower left corner (clockwise).

Premade example code:

To get started with your custom modifier you can open "*ProceduralUIImage/Scripts/Modifiers/CustomPremadeModifier.cs*". Uncomment the code and modify it to your needs.

Feedback | Support | Feature requests

Feel free to contact me and let me know what you think or need.
Have fun with Procedural UI Image!

Josh H
assetstore.joshh@gmail.com