

Soft Mask

[Asset Store](#) | [Demo](#) | [Email](#) | [Forum](#)

Documentation for version 1.4

What is Soft Mask?	2
Features	3
Getting Started	3
The Use of Soft Mask	3
Applying a Soft Mask	3
Replacing a standard mask by a Soft Mask	5
Disabling a Soft Mask	5
Using Soft Mask with TextMesh Pro	5
Adding support of Soft Mask to custom shaders	5
Using Soft Mask in code	6
Instantiating Soft Mask from code	6
Reference	6
Interaction with the Graphic component	6
Integration with TextMesh Pro	7
Prefabs	9
Nested Masks	9
Properties	10
Limitations	14
Performance	14
Compatibility	14
Support and feedback	14
Change history	15

What is Soft Mask?

Soft Mask is a component for smooth masking of UI elements in Unity.

After adding to a Game Object, Soft Mask “masks” its child elements. Soft Mask gives a quite similar effect as standard Mask and Mask Rect 2D but has some advantages over them.

- Unlike [Rect Mask 2D](#), Soft Mask supports rotation of the mask.
- Unlike [Mask](#), Soft Mask supports transparency, which allows you to create smooth transitions from visible to invisible parts of an image, and also use inclined or rounded edges of UI elements.



Soft Mask



*Unity's Standard Mask
(notice visual artifacts on the rounded
corners caused by cutoff)*

Features

- *Ease of use*: just place it on a parent element in the same way as standard Mask. There's no need to assign special materials or modify the masked elements in any other way.
- *Support for Image, Raw Image, Sprite, and Texture*: use Image or Raw Image components as a mask. Or set a Sprite or Texture explicitly.
- *Support for TextMesh Pro*: mask your high-quality SDF texts along with other UI elements.
- *Custom shader support*: make your UI shaders compatible with Soft Mask by adding just a few lines to the shader code.
- *Mask Inversion*: invert the inner and outer areas of the mask separately.
- *Real-time updates*: change the configuration at runtime — create, destroy, enable, disable, move, rotate and reorder a mask or masked elements.
- *Raycast filtering*: restrict input processing to only visible parts of the masked UI.
- *Flexible adjustment of mask's color channels*: use black and white images, images with transparency or set a custom weight for each color channel.

Getting Started

Soft Mask requires no additional setup. Just import the package, and you are ready to go.

The package includes example scenes so that you can see Soft Mask in action right away. These scenes can be found in the `Samples` folder. This folder is unnecessary for the Soft Mask to work, so you can safely remove it from your project if you want.

If you are using UnityScript instead of C#, you can move the imported `SoftMask` folder to the `Plugins` folder. It allows you to use Soft Mask in UnityScript code.

The Use of Soft Mask

Applying a Soft Mask

To apply a Soft Mask, follow the steps below.

1. Add a UI/Soft Mask component to a Game Object in the same way as you add a standard Mask.
2. Set up a Soft Mask:
 - If you want the mask image to be visible (as with the *Show Mask Graphic* option of the standard Mask), ensure that the Game Object has an Image or Raw Image component. The Soft Mask will use the same image that is rendered by this component.
 - If you do not want the mask image to be visible and only want to use it as a mask, you have two ways to do this:
 - Select *Sprite* or *Texture* in the *Source* drop-down list and set a sprite or texture for the mask. In this case, the Game Object doesn't have to have a Graphic component at all.
 - Alternatively, if you already have an Image or Raw Image on the Game Object, you can just disable it. Soft Mask will continue masking. Note that this option has some limitations, for details see [Interaction with the Graphic component](#).

Replacing a standard mask by a Soft Mask

You can easily replace a standard Mask that uses an Image or Raw Image component by a Soft Mask. To do so, follow the steps below.

1. Add a UI/Soft Mask component to the Game Object which already has a standard Mask.
2. Disable the standard Mask and check that Soft Mask works properly.
3. If the *Show Mask Graphic* option has been disabled for the standard Mask, you have two options to make up for it.
 - a. Assign the same sprite or texture that was used by the Image or Raw Image to the Soft Mask and configure it accordingly by specifying *Border Mode* or *UV Rect*. After that, delete the Graphic component.

- b. Alternatively, you can just disable the Graphic component. Soft Mask will continue masking. Note that this option has some limitations, for details see [Interaction with the Graphic component](#).
4. Delete the standard Mask from the Game Object.

Disabling a Soft Mask

If you want to turn off masking, disable the Soft Mask component on the Game Object. Unlike the standard Mask, when the Graphic component of a Game Object is disabled, Soft Mask still keeps masking.

Using Soft Mask with TextMesh Pro

To enable Soft Mask to mask [TextMesh Pro](#) texts in your project, follow the steps below.

1. Import Soft Mask and TextMesh Pro packages into the project.
2. If you're using the Package version of TextMesh Pro (available starting from 2018.2), import TextMesh Pro essentials by clicking *Window / TextMesh Pro / Import TMP Essential Resources*.
3. Click the menu *Tools / Soft Mask / Update TextMesh Pro Integration*. This will generate versions of TextMesh Pro shaders with Soft Mask support.

For more information on using SoftMask with TextMesh Pro, see [Integration with TextMesh Pro](#).

Adding support of Soft Mask to custom shaders

Soft Mask is able to mask all Graphic components which use the standard Unity UI shader. If you want to use custom shaders for rendering UI elements (your own or from another package from Asset Store), you can add the support of Soft Mask to them.

To add the support of Soft Mask to a custom shader, you have to insert some declarations and instructions into the shader code. As an example of what exactly should be done, see the shader `SoftMask/Samples/Materials/WaveWithSoftMask.shader`. This shader is used by the *04-CustomShaders* example. All the instructions required for Soft Mask support have the comment `// Soft Mask Support`

For a more detailed step-by-step guide, see [Custom Shader Tutorial](#).

Using Soft Mask in code

Everything related to Soft Mask is contained in the `SoftMasking` namespace. All the properties described in the [Properties](#) section are also available from the code. The public interface of `SoftMask` is documented via XML comments.

If you are interested in how Soft Mask is implemented, you can find an overall description at the beginning of the `SoftMask` class.

Instantiating Soft Mask from code

Soft Mask replaces the standard Unity UI shader during the rendering of child objects. When you add a Soft Mask component in the editor, it gets a reference to the replacement shader and works “out

of the box”. But if you add a Soft Mask from the code by `AddComponent()` you have to specify the shader manually. To do so, set the `defaultShader` property:

```
public class InstantiateSoftMaskExample : MonoBehaviour {
    public Shader defaultMaskShader;

    void Start() {
        var mask = gameObject.AddComponent<SoftMask>();
        mask.defaultShader = defaultMaskShader;
    }
}
```

Using the example above, you can just assign `SoftMask/Shader/SoftMask.shader` to the `defaultMaskShader` in Inspector. If none of the masked objects use the default UI shader, you don’t have to set the `defaultShader` property.

If you’re going to publish the project on the mobile platforms and use ETC1 compressed textures in UI, you should also set the `defaultETC1Shader` property.

You may wonder why Soft Mask references shaders explicitly and doesn’t use dynamic resource loading as many packages do. The main reason is that explicit referencing is more optimal. And it still ease to use when instantiating Soft Mask in the Editor which is, presumably, the most popular case.

Dynamic resources approach is used for TextMesh Pro shaders as this package uses them anyway.

Reference

Interaction with the Graphic component

Unlike Unity's standard Mask, Soft Mask doesn't rely on the rendering of the Graphic component. When you set *Source* to *Graphic*, Soft Mask just picks up some properties (texture and border mode or UV rect) from the Graphic component but doesn't really use it in itself.

This approach has some limitations compared to a standard Mask:

- You can use only Image or Raw Image of all Graphic components as a mask source. Text and any other Graphics aren't supported as a mask.
- Soft Mask doesn't support the Filled type and Fill Center option of Image.
- The color set in the properties of Image or Raw Image isn't taken into account by Soft Mask. Only the texture is used.

When the Graphic component is disabled, Soft Mask continues to work, but it isn't updated accordingly when properties of the Graphic change. Despite the limitation, this operating mode may be useful when you transit from a standard Mask with the *Show Mask Graphic* option enabled to Soft Mask.

Integration with TextMesh Pro

[TextMesh Pro](#) is a package that implements high-quality text rendering. Starting from 2018.2, it's included in Unity by default and available via Package Manager.

Soft Mask supports TextMesh Pro and can be used to mask TMPro texts. It works with Source, Binary and, the newest one, Package versions of TextMesh Pro. To make it possible, Soft Mask includes a script that generates specializations of TMPro's shaders with special Soft Mask instructions. This generation process is executed via the *Tools / Soft Mask / Update TextMesh Pro* menu.

Samples

There are two example scenes included in Soft Mask which demonstrate usage of Soft Mask with TextMesh Pro. You can find these scenes in the *Samples/Scenes/TextMeshPro* folder. These scenes are provided in three variants, depending on which version of TextMesh Pro you're using:

- PackageTMPPro — for the [Package Manager version](#) of TextMesh Pro,
- BinaryTMPPro — for the version of TextMesh Pro [available on Asset Store](#),
- SourceTMPPro — for the previous (paid) version of TextMesh Pro which is distributed via the [TextMesh Pro forum](#)

If you don't know which version of TextMesh Pro you're using, it's probably the binary one. If you open a sample scene from a wrong subfolder, text objects will contain "missing script" errors and will not be rendered. So you can just try different folders until you meet a working example.

Shader Generation

To activate integration the menu *Tools / Soft Mask / Update TextMesh Pro Integration* should be executed. It runs a script that generates specializations of TextMesh Pro shaders with Soft Mask

support. The script does this by copying original TextMesh Pro shaders and patching them accordingly.

Generated shaders are stored in the `SoftMask/Shaders/Generated/Resources` folder. Soft Mask automatically picks up these shaders and use them as replacements of standard TextMesh Pro shaders in the case when a TextMesh Pro object is nested into a Soft Mask. If you want to know more about these mechanics or want to implement something like this for your own shaders, see the `SoftMask/Scripts/MaterialReplacements.cs` file.

Shader generation can be executed at any moment. If generated shaders already exist, they will be overridden. The generated shaders are referenced by a name, not by GUIDs, so, it's completely safe to remove the generated shaders folder and re-run the *Update TextMesh Pro Integration* menu.

Generated shaders contain absolute paths to TextMesh Pro and Soft Mask shader include files. If you have moved these packages, you need to regenerate shaders by executing *Tools / Soft Mask / Update TextMesh Pro Integration*.

Known Issue The current version of the TextMesh Pro integration couldn't detect which shader variants are used in your scenes. This may lead to Unity removing necessary shaders from the build. If your texts look simpler in the build than they should, you probably faced this issue. To bypass it, you can add the patched shaders to the *Always Included Shaders* list on the *Graphics Settings* window. For more information, see [this post](#).

Updating the Packages

Any update of TextMesh Pro could potentially break Soft Mask integration. Please follow the guidelines below to be sure that your project will not be broken during an update of these packages.

- When a new version of TextMesh Pro is available it's strongly recommended to test the integration on an empty project first. To do so:
 - a. Create a new project.
 - b. Import the updated TextMesh Pro and Soft Mask.
If you're using a Package version of TMP, import essential resources first.
 - c. Execute the *Tools / Soft Mask / Update TextMesh Pro Integration* menu.
 - d. Open an example scene from the `SoftMask/Samples/TextMeshPro` folder. Ensure that it works as expected and the *Console* window doesn't contain any errors.
- If the integration isn't working in an empty project, please [contact](#) the developer and wait for a Soft Mask update before updating TextMesh Pro.

After updating either TextMesh Pro or Soft Mask, the *Tools / Soft Mask / Update TextMesh Pro Integration* menu should be executed again.

Limitations

The integration with TextMesh Pro has some limitations. Most of them are caused by that Soft Mask is a UI-only solution while TextMesh Pro also can display texts in 3D scenes.

- Only *TextMeshProUGUI* component can be masked.

- Surface and Overlay versions of shaders aren't supported by Soft Mask.
- Like standard Text, TextMeshProUGUI can't be used as a mask.

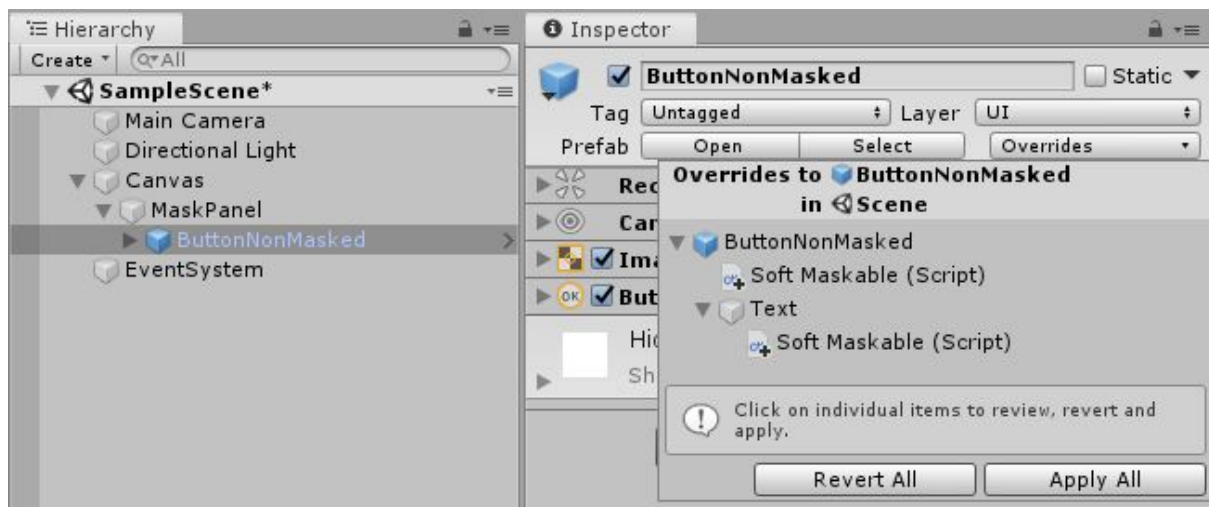
Prefabs

Prefabs may be used freely with Soft Mask. But due to some internals of Soft Mask the use experience with prefabs may be less convenient than expected. This section explains these inconveniences in details.

Soft Mask creates an invisible script named *Soft Maskable* on each of its children. This script dynamically replaces the actual material that will be used for rendering. During usual work with a Scene, these scripts are completely unnoticeable. They are created and destroyed automatically as objects get under a Soft Mask and back.

But if you create a prefab from an object that is masked by a Soft Mask, that invisible script is saved into a prefab. If you later place an instance of this prefab not inside a Soft Mask, the attached *Soft Maskable* will be destroyed. Unity will treat this situation as a modification of a prefab instance and show it on the UI. If you revert properties back to the prefab defaults, the invisible script will be re-instantiated again and then immediately destroyed.

So, instances of prefabs that have been created from a masked object will behave a bit strange when they're not masked by a Soft Mask. The opposite is true as well: an instance of a non-masked prefab that's put inside a Soft Mask will display addition of a new component.



The ButtonNonMasked is an instance of a non-masked prefab that is placed into a mask. As you can see, Unity Inspector shows addition of the new components.

This may make your workflow a bit inconvenient but it doesn't incur any errors.

A general recommendation on whether to create a prefab from a masked or a non-masked object is as follows: if you use this prefab mostly inside masks, create it from a masked object (with a *Soft Maskable* script). If you use it mostly without masks, create it from a non-masked object (without a *Soft Maskable* script). This will minimize the number of runtime operations.

Nested Masks

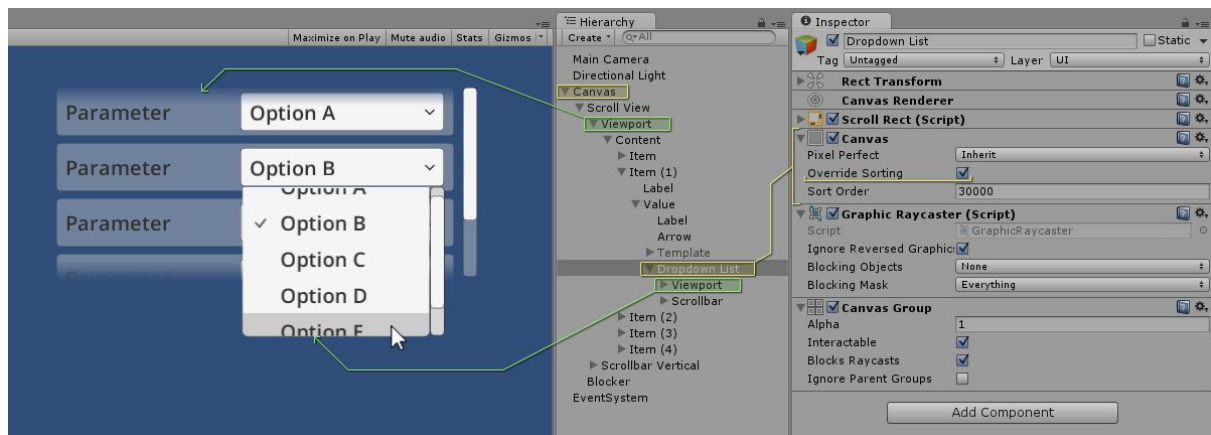
Soft Mask doesn't support nesting of masks. That is, you can't create a soft masked area inside another soft masked area. This is contrary to Unity's standard masking components—*Mask* and *Rect Mask 2D*, which can be freely nested.

If you need nested masked areas, consider using a standard Unity's Mask instead of either a parent or child Soft Mask.

Each UI element below Soft Mask is masked by the closest parent Soft Mask. If you place a Soft Mask inside another Soft Mask, each of them will mask only its own children, and the Inspector window will display a warning message for both masks. Note that if a nested mask has a Graphic component, it will be masked by the parent mask, which may not be what you expect.

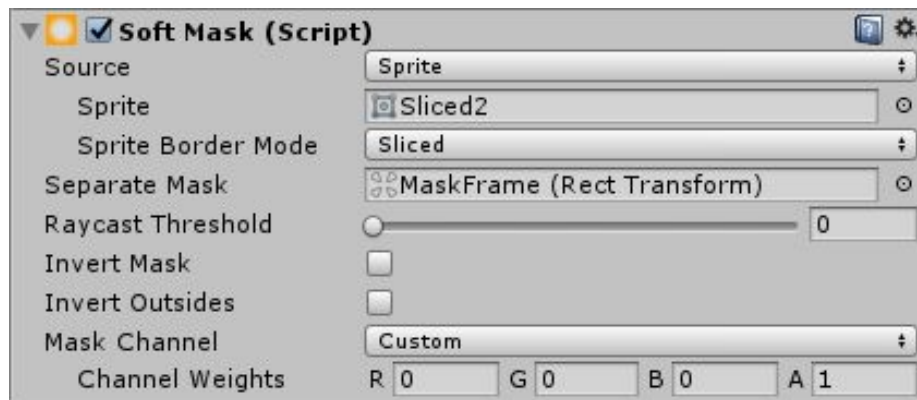
Known Issue Due to specifics of how standard Mask works with other components on the same Game Object, a standard Mask nested into a Soft Mask may work incorrectly in case if it was added after Soft Mask had been activated. For more information, see [this post](#).

The exception from this limitation is nested canvases with *Override Sorting* enabled. The effect of standard Unity masking components doesn't cross the borders between nested canvases with *Override Sorting* set. It is used by, for example, the standard *Dropdown* component: it spawns its drop-down list in a separate canvas. If you place a Dropdown inside a masked area, the drop-down list will be not clipped, as expected. Soft Mask works in the same way:



An example scene with nested canvases (yellow) and Soft Masks (green). Note that the nested canvas has *Override Sorting* enabled.

Properties



Soft Mask Inspector

Property	Description
Source	<p>Defines the way to configure a mask image. Possible values:</p> <p>Graphic The mask image is taken from the Graphic component of the Game Object. Soft Mask supports Image and Raw Image components only. If there is no appropriate Graphic on the GameObject, a solid rectangle of the RectTransform dimensions will be used.</p> <p>Sprite Mask image is taken from an explicitly specified Sprite. When this mode is used, <i>Sprite Border Mode</i> can be also set to determine how to process Sprite's borders. If the sprite isn't set, a solid rectangle of the RectTransform dimensions will be used. This mode is similar to using an Image with according <i>Sprite</i> and <i>Type</i> set.</p> <p>Texture The mask image is taken from the explicitly specified Texture2D. When this mode is used, <i>Texture UV Rect</i> can be also set to determine what part of the texture should be used. If the texture isn't set, a solid rectangle of the RectTransform dimensions will be used. This mode is similar to using a RawImage with according <i>Texture</i> and <i>UV Rect</i> set.</p>
Separate Mask	<p>Allows to specify a separate Rect Transform that should be used as a mask. If you set this property, the Soft Mask will still mask only its own children but the masking window will match the specified Rect Transform. If <i>Source</i> mode is set to <i>Graphic</i>, the Graphic component of the specified object will be used.</p> <p>This property simplifies the way to make masks that are moving relative to</p>


	<p>the masked content. An example of this you may see in the sample <i>04-CustomShaders</i>.</p> <p>When the value is set to None, the Rect Transform of the Soft Mask will be used (this is the way standard Unity's Mask works).</p> <p>The default value is None.</p>
Raycast Threshold	<p>Specifies the minimum mask value that the point should have for an input event to pass. Accepts values in range [0..1].</p> <p>If the value sampled from the mask is greater or equal this value, the input event is considered to be 'hit'. The default value is 0, which means that any input event inside the RectTransform is passed further. If you specify the value greater than 0, the mask's texture should be readable.</p> <p>An example usage of this property you can find in the sample <i>02-Minimap</i>.</p>
Invert Mask	<p>If checked, the mask inside the masking rect will be inverted. That is, points where mask is transparent will be opaque and vice versa. The masking rect is the Rect Transform of the Game Object that Soft Mask is attached to or the transform specified by the <i>Separate Mask</i> property if it's not None.</p> <p>Mask inversion is applied after <i>Mask Channels</i>.</p>
Invert Outsides	<p>If checked, the mask outside the masking rect will be inverted. By default, the outsides of a mask are fully transparent. When this option is enabled, the outsides of a mask will be fully opaque. The masking rect is the Rect Transform of the Game Object that Soft Mask is attached to or the transform specified by <i>Separate Mask</i> if it's not None.</p> <p>Combining this option with <i>Invert Mask</i> you may achieve effect of cutting a hole in the underlying objects.</p> <p>Unlike <i>Invert Mask</i>, this option isn't affected by <i>Mask Channels</i> because the texture isn't used at all for the outside area of a mask.</p>
Mask Channel	<p>Defines what color channels of the image are used as a mask. Possible values:</p> <p>Alpha The alpha channel (transparency). Coincides with <i>Channel Weights</i> = (0, 0, 0, 1). (default)</p> <p>Red The red channel. Coincides with <i>Channel Weights</i> = (1, 0, 0, 0).</p>

	<p>Green The green channel. Coincides with <i>Channel Weights</i> = (0, 1, 0, 0).</p> <p>Blue The blue channel. Coincides with <i>Channel Weights</i> = (0, 0, 1, 0).</p> <p>Gray The average value of the red, green and blue channels. Coincides with <i>Channel Weights</i> = (0.33, 0.33, 0.33, 0).</p> <p>Custom Allows to set a specific weight for each color channel of the image.</p>
Channel Weights	<p>Displayed if <i>Mask Channel</i> is Custom.</p> <p>You can set the weight from 0 to 1 for each color channel. Given the value sampled from the mask texture is <i>mask</i> and the specified property value is <i>weights</i>, the resulted mask value is calculated as:</p> $\sum_{i \in \{r,g,b,a\}} mask_i \cdot weights_i$

Additional properties for Texture-based masks:

Texture	Defines a texture to use as a mask.
Texture UV Rect	Defines coordinates and size of the texture fragment to display the image, given in normalized coordinates (range 0.0 to 1.0). This property works in the same way as UV Rect property of <i>Raw Image</i> .

Additional properties for Sprite-based masks:

Sprite	Defines a sprite to use as a mask.
Sprite Border Mode	<p>Specifies how to render the sprite borders. Possible values: Simple, Sliced, Tiled. They all work in the same way as according values of Type property of <i>Image</i>.</p> <p>If you notice visual artifacts in <i>Sliced</i> or <i>Tiled</i> mode when the central part of a mask is too shrunk, you may try to reduce the anisotropy level of the mask texture or even disable it at all. Also, disabling of mipmapping may help. The cause of these artifacts is that when a child element is rendered there are no</p> 

separate vertices between different parts of a sliced sprite, all mapping is performed in the shader. In the case when the central part is collapsed, the texture coordinate changes are too steep which causes the hardware to perform texture filtering in a way that produces artifacts. In many cases, both anisotropy and mipmaps aren't needed for UI textures.

Limitations

- Only textures and sprites can be used as a mask. Currently, Soft Mask doesn't support masking by Text or any other Graphic components except Image and Raw Image.
- Soft Mask doesn't support nested masks. To work around this limitation, use a standard Mask or Rect Mask 2D in combination with a Soft Mask. See [Nested Masks](#) section for more information.
- Sprites that are packed in *Tight Packing Mode* aren't supported as a mask. Disable packing for the mask sprite or use *Rectangle Packing Mode*.
- Sprites with an alpha split texture aren't supported as a mask. Disable compression for the mask texture or use another compression type.
- Soft Mask is intended to work with Unity UI and doesn't support any other 2D or 3D graphics "out of the box".

Performance

- Soft Mask uses 3 Shader Keywords, while Unity 5 provides 128. Before using Soft Mask, make sure that you have enough free slots in the project. For details about Shader Keywords, see [Multiple Program Variants](#) section in the Unity Documentation.
- Support of Soft Mask adds to a shader from 7 to 10 constants and 1 sampler, depending on Soft Mask settings.
- All child elements of a Soft Mask are rendered using the special shader. This shader is more complex and a bit slower than the standard shader. Execution speed of the shader depends mostly on the *Border Mode* value. The fastest mode is Simple. It's recommended to use Soft Mask only when the capabilities of standard Mask are insufficient for your task.

Compatibility

Soft Mask 1.4 is compatible with Unity versions 5.6 - 2019.1.

Soft Mask doesn't contain any platform-specific code and should work on all the platforms supported by Unity.

Support and feedback

If you need support for this product or wish to provide feedback or suggestions, feel free to email me at knyazev751@gmail.com or post on [the forum thread](#).

Change history

1.4

Starting from 1.4, Soft Mask requires Unity 5.6 or higher.

Improvements

- Added the *Invert Mask* and *Invert Outsides* options which allow to separately invert the inner and outer areas of the mask.
- Added a new example featuring mask inversion.
- Added tooltips for the Inspector UI.

Bug fixes

- Fixed a bug with incorrect display of a mask when a scaled sprite atlas was used.
- Fixed an incorrect calculation of tile repeat count in tiled mode that might occur for some sprites.
- Fixed a bug with inconsistent Inspector state after reverting the *Channel Weights* property to prefab defaults.

1.3.1

Improvements

- Improved the error message that is shown in the case when an unreadable texture is used with the *Raycast Threshold* value greater than zero. Also, this error is now displayed in Inspector window at edit time, not only in the Console window at runtime.
- Removed an unnecessary garbage allocation that might be happening because of the `GetComponent<>` call.

Bug fixes

- Fixed a [bug](#) caused the mask to lag behind when being moved. In particular, it was causing the lagging of the mask during the automatic movement inside a Scroll Rect.
- Fixed a compilation error when targeting .NET Standard 2.0.

1.3

Improvements

- Integration with TextMesh Pro, which was previously implemented in a separate integration package, now included in the main package. It corresponds to Unity's decision to include TextMesh Pro in standard Unity installation.
- Removed unnecessary [garbage allocation](#) which was taking place when non-Graphic mask source was used.

Bug fixes

- Fixed an [assertion failure](#) which was introduced in the previous version.
- Fixed a bug caused the masked UI to entirely disappear sometimes in Editor (after scene saving, after code reload, etc.).
- Fixed a bug caused *Update TextMesh Pro Integration* to not work on some Unity versions in the case when the path to the project contained “Soft Mask” text.

1.2.4

Bug fixes

- Fixed a [bug](#) that led the mask dimensions to not update when a separate mask resizes.

1.2.3

Improvements

- Improved compatibility of Soft Mask shaders with 2018 lineup. Now they support UV transformations as standard 2018’s UI shader do.
- Removed a use of an obsolete name which caused Unity 2017.3 and later versions to run script upgrade on Soft Mask import.

Bug fixes

- Fixed a [bug](#) caused Soft Mask to not work on some AOT platforms.
- Fixed an [issue](#) caused masked UI elements to flicker sometimes.

1.2.2

Improvements

- Improved compatibility of Soft Mask shaders with 2017.2. Now they perform rectangular clipping only when UNITY_UI_CLIP_RECT is defined.

Bug fixes

- Fixed a bug causing Soft Mask to not work when the project contained dynamic assemblies with an unfinished process of type building i.e. when [TypeBuilder.CreateType\(\)](#) hasn’t been called.

1.2.1

Bug fixes

- Fixed a bug causing Soft Mask to not work when the project contained dynamic assemblies.

1.2

Improvements

- Added support of TextMesh Pro in a separate integration package — [Soft Mask for TextMesh Pro](#). The package can be found in Asset Store or in [the support thread](#).
- Added an ability to inject custom logic into the process of shader replacement. This feature is used by the new TextMesh Pro integration package and also can be used to better integrate Soft Mask with your own UI shaders.

- Added [a tutorial](#) on how to add support of Soft Mask into your own shader.

Bug fixes

- Removed anti-aliasing from rectangular clipping. Images whose content is close to the texture borders are displayed correctly now. If you used Soft Mask without texture set and want to achieve the previous anti-aliased look, you can use a special texture with a one-pixel gap on the borders.
- Removed a visual artifact that sometimes could appear on the edge of the masked image in Tiled sprite mode.

1.1.1

Improvements

- Improved usage of Soft Mask in Unity version 5.6 or later. Now Unity doesn't upgrade Soft Mask shaders and warnings aren't popped up during import.
- Nested masks aren't disabled automatically now. Instead, each child is masked by the nearest mask only, which gives more predictable behavior. See [Nested Masks](#) section.

Bug fixes

- Fixed the way Soft Mask interacts with nested canvases with Sorting Override flag enabled. It doesn't mask those canvases anymore which corresponds to the way standard Mask does work.

1.1

Improvements

- Added the *Separate Mask* parameter that allows to separate mask from the masked elements. The sample *04-CustomShaders* has been reworked to show this feature in use.
- Optimized real-time performance, especially on huge hierarchies. Soft Mask now has almost the same performance cost on CPU as standard Unity Mask (which is very close to zero).
- Improved mapping of Sliced and Tiled masks in the case when the central part is collapsed.
- Reworked samples: they now demonstrate more features of Soft Mask and also look better.

Bug fixes

- Fixed a bug causing Soft Mask to work incorrectly on nested canvases. The partial consequence of this was the inability to use Soft Mask in a standard Dropdown control which uses nested canvas for the drop-down list. Now it is possible.
- Fixed a bug preventing Soft Mask from updating after moving it from one canvas to another.
- Fixed a bug preventing rendering of the child elements when a mask used a sprite with one of its borders set to zero in Sliced or Tiled mode.