

Soft Mask

Documentation for version 1.2.1

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 the image, as well as use inclined or rounded edges of UI elements.



Soft Mask



*Standard Unity's 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 the Image or Raw Image components as a mask. Or set the Sprite or Texture directly to the Soft Mask.
- *Support for TextMesh Pro*: mask your high-quality SDF texts by just importing the special integration package.
- *Custom shader support*: support Soft Mask in your shaders by adding just a few lines to the shader code.

- *Separation from the masked hierarchy*: organize your hierarchies how you want, masked elements doesn't have to be children of the masking rectangle.
- *Real-time updates*: change the configuration at runtime — create, destroy, enable, disable, move, rotate and reorder the 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, you can 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 the Soft Mask, follow the steps below.

1. Add a UI/Soft Mask component to the Game Object in the same way as you add a standard Mask.
2. Set up a Soft Mask:
 - a. If you want to display the mask image in the scene, 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.
 - b. If you want to use an image as a mask only and do not want to see it in the scene, you have two ways to do this:
 - i. 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.
 - ii. 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, see the section [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 the 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 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, see the section [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. Download and import the integration package — [Soft Mask for TextMesh Pro](#).
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 the section [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 ones 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 the 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 example *04-CustomShaders*. All instructions required for Soft Mask support have the comment `// Soft Mask Support`

If you want a more detailed step-by-step example check out the [Custom Shader Tutorial](#).

Using Soft Mask in code

Everything related to Soft Mask is contained in the namespace `SoftMasking`. Any properties that are described in the section [Properties](#) are available from code too. 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 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;
    }
}
```

If none of 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.

Reference

Interaction with the Graphic component

Unlike Unity’s standard Mask, Soft Mask doesn’t rely on 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 itself.

This approach has some limitations comparing to 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 Filled type and Fill Center option of Image.
- The color set in the properties of Image or Raw Image isn’t accounted by the Soft Mask. Only texture is used.

When the Graphic component is disabled, Soft Mask continues to work, but 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 a Soft Mask.

Integration with TextMesh Pro

[TextMesh Pro](#) is a free package that implements high-quality text rendering. Out of the box, Soft Mask doesn't mask TextMesh Pro texts and graphics because they use specialized shaders for rendering. Of course, you can adapt these shaders by yourself, but you don't need to because this is already done for you.

The integration is implemented in [a separate package](#) which can be freely downloaded from Asset Store. It contains patches to TextMesh Pro shaders, a script that generates specializations of shaders with Soft Mask support and also some examples.

Samples

The example scenes in the integration package are provided in two variants which are stored in the following subfolders:

- BinaryTMPPro — for the current (free) version of TextMesh Pro,
- SourceTMPPro — for the previous (paid) version of TextMesh Pro (which is distributed via the [TextMesh Pro forum](#) instead of Asset Store)

If you don't know which version of TextMesh Pro you use, 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 be not rendered.

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 applying to them patches from the integration package.

Generated shaders are stored in the Shaders/Generated/Resources subfolder of the package root. 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 implement something like this for your own shaders, see the `MaterialReplacements.cs` file in the Soft Mask package.

Shader generation can be executed at any moment. If generated shaders already exist, they will be replaced.

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

Updating the Packages

Any update of TextMesh Pro could potentially break Soft Mask integration. Please, follow the guidelines below to be sure that you will not break your project while updating 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:
 - Create a new project.
 - Import the updated TextMesh Pro, Soft Mask, and the integration package.
 - Execute the menu *Tools / Soft Mask / Update TextMesh Pro Integration*.
 - Open an example scene from the integration package. Ensure that it works as expected and the console doesn't contain any errors.
- If the integration isn't working in an empty project, please wait for an update of the integration package before updating TextMesh Pro.
- When a new version of Soft Mask is available, you can find out whether the integration package should be updated too or not in [the support thread](#). If the integration package should be updated and an update isn't available on the Asset Store yet, please wait for it.

After updating either TextMesh Pro or the integration package, the menu *Tools / Soft Mask / Update TextMesh Pro Integration* 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.
- As standard Text, TextMesh Pro can't be used as a mask.

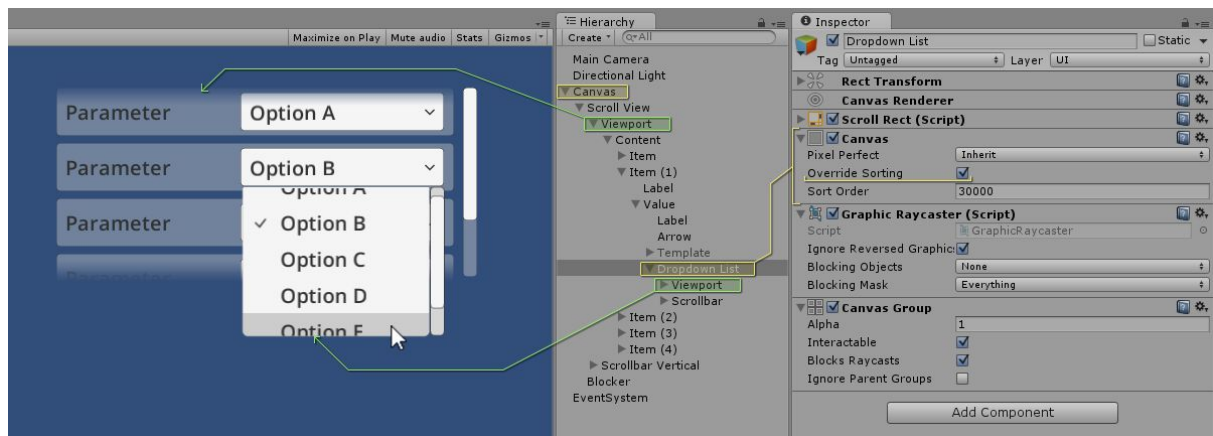
Nested Masks

Soft Mask doesn't allow 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 nested freely.

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

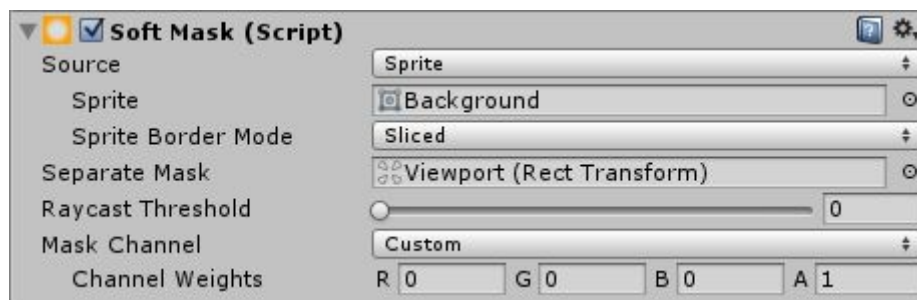
If you need nested masked areas, consider using a standard Unity's Mask instead of either the parent or child Soft Mask.

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. This 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:	Function:
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 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</p>

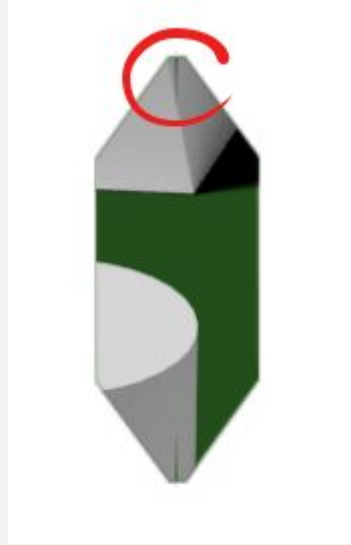
	<p>used (this is the way standard Unity's Mask works). The default value is None.</p>
Raycast Threshold	<p>Specifies the minimum mask value that the point should have for an input event to pass. Accepts values in range [0..1].</p> <p>If the value sampled from the mask is greater or equal this value, 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 found in the sample <i>02-Minimap</i>.</p>
Mask Channel	<p>Defines what color channels of the image are used as a mask. Possible values:</p> <p>Alpha (default) The alpha channel (transparency). Coincides with <i>Channel Weights</i> = (0, 0, 0, 1).</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> <div> <p>If you notice visual artefacts 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 artefacts is that when a child element is rendered there are no separate vertices between different parts of a sliced sprite, all mapping is performed in the shader. In 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 artefacts. In many cases both anisotropy and mipmaps aren't needed for UI textures.</p>  </div>

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 of 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 you lack features of standard Mask.

Compatibility

Soft Mask 1.2.1 is compatible with Unity versions 5.3 - 2017.2.

Tested platforms:

- standalone (Windows/Mac OS),
- WebGL,
- Android,
- Windows Phone.

Change history

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. Unity doesn't upgrade Soft Mask shaders and warnings aren't popped up during import anymore.
- Nested Masks aren't disabled automatically now. Instead, each child is masked by the nearest mask only, which gives more predictable behaviour. 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.

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](#).