ALPHA MASK: UI, SPRITES, QUADS

by Tag of Joy

This document describes usage guidelines for the **Alpha Mask** developed by **Tag of Joy**. The plugin should be purchased on the Unity Asset Store (http://u3d.as/bzJ). Sharing or distribution is not permitted.

HOW TO USE

- Create an empty Game Object as a sibling of the elements that need to be masked (the masked elements can have a deeper level of hierarchy – they will still be masked). Rename the new object, so that it's clearly identified (e. g. "Mask").
- Attach the **Alpha Mask** component to the created Game Object and assign a texture that you want to use as the mask on the **Main Tex** slot of the component (not the texture slot of the material).
- Click **Apply Mask to Siblings in Hierarchy**. It will detect all siblings (including their children) and apply the mask to them (if they use the **Default Sprite** or **Unlit/Transparent** shaders).

MASK INSPECTOR FIELDS

- **Main Tex** – the texture that will be used to mask objects.
- **Tiling** – the tiling of the texture that will be used to mask objects.
- **Offset** – the offset of the texture that will be used to mask objects.
- **Masking Enabled** – does the mask need to have the effect on the siblings? Can be enabled/disabled in the Editor, as well as in run-time.
- **Clamp Alpha Horizontally** – if the texture isn't clamped by Unity (in import settings), then you can choose to clamp it horizontally only (it will be repeated vertically, unless chosen otherwise).
- **Clamp Alpha Vertically** – if the texture isn't clamped by Unity (in import settings), then you can choose to clamp it vertically only (it will be repeated horizontally, unless chosen otherwise).
- **Clamping Border** – if one of the two above settings are enabled, you can use this variable to tweak the "edge" of clamping. Depending on the alpha texture size and its usage, you might run into texture clamping issues. In that case, try increasing (or lowering) the Clamping Border value.
- **Use Mask Alpha Channel (not RGB)** – the mask uses the texture RGB channels by default. Toggle this option to use the Alpha channel of the texture instead.
- **Display Mask** – toggle this setting to enable or disable the visibility of the mask (only the visibility, not its effect). This setting is only available in the Editor and when not running the game.

MATERIALS

The Alpha Mask component automatically assigns (and when necessary creates) the needed masked materials for the masked elements, when you click **Apply Mask to Siblings in Hierarchy**. However, if you want to have your own materials, you can create a material manually, assign one of the masked shaders to it, and assign the material to all masked elements (for a single Mask hierarchy, one masked material is usually enough).

This is especially important when using prefabs (see section "Using the Alpha Mask with prefabs").

USING THE ALPHA MASK WITH PREFABS

Some materials and necessary components are created/assigned automatically. However, when the masked objects and/or the Mask itself is used in prefabs, some additional steps might be needed.

If masked elements use only **Sprite Renderers** and **Unity UI elements**, creating and instantiating prefabs works as usual. No additional steps are necessary.

If masked elements contain **Mesh Renderers**, there are a few additional steps necessary to make them work in prefabs:

1. Create the Mask and the Meshes that you will mask (as described in the "How to Use" section, but do not click "Apply Mask to Siblings in Hierarchy" just yet).
2. Manually create a material (within the assets) and assign the "Alpha Masked/Unlit Alpha Masked - World Coords" (if you're replacing **Unlit/Transparent** material) shader.
3. Drag this material into the material slots of all the Meshes that you've created in step 1 (the ones that you want to mask).
4. If everything worked so far, create a prefab from the hierarchy. You can now instantiate the prefab, and it should work as intended.

ADDITIONAL NOTES

- **Instantiating, cloning and duplicating masked items.** If you instantiate, clone or duplicate a masked item in play mode, you need to call **ScheduleFullMaskRefresh()** on the Mask script component. Note: You do not need to do this if you are creating a copy of an entire masking hierarchy structure, including the Mask itself.
- Because of an internal Unity bug on Unity 5.2.3 and older, using sprites with animated masks might cause some performance issues.
- If you want to use the Mask with a **Particle System**, change the material of the particle system to one with the **Unlit/Transparent** shader. Then follow the regular steps of adding and applying a Mask.
- Setting Mask scale to 0 on any axis will result in an undefined behavior. Scale mask to a very small size instead.

INTEGRATION WITH CUSTOM TRANSPARENT VERTEX/FRAGMENT SHADERS

1. Open the shader you wish to modify
2. Copy this inside the properties field of the shader:

```
[HideInInspector][PerRendererData][Toggle] _Enabled("Mask Enabled", Float) = 1
[HideInInspector][PerRendererData][Toggle] _ClampHoriz("Clamp Alpha Horizontally", Float) = 0
[HideInInspector][PerRendererData][Toggle] _ClampVert("Clamp Alpha Vertically", Float) = 0
[HideInInspector][PerRendererData][Toggle] _UseAlphaChannel("Use Mask Alpha Channel (not RGB)", Float) = 0
[HideInInspector][PerRendererData][Toggle] _ScreenSpaceUI ("Is this screen space ui element?", Float) = 0
[HideInInspector][PerRendererData]_AlphaTex("Alpha Mask", 2D) = "white" {}
[HideInInspector][PerRendererData]_ClampBorder("Clamping Border", Float) = 0.01
```

3. Add: "ToJMasked" = "True" to subshader tags
4. Add #include "Assets/Alpha Masking/ToJAlphaMasking.cginc" to the top of CGPROGRAM
5. Add TOJ_MASK_COORDS(<texcoord index>) to "vertex to fragment" struct, where <texcoord index> is an unused TEXCOORD number (similar to fog integration) e.g.: TOJ_MASK_COORDS(1)
6. Add TOJ_TRANSFER_MASK(<output>, <input vertex>); to vertex shader function, where <output> is

"vertex to fragment" struct that will be passed forward to fragment shader, and <input vertex> is a float4 of vertex coordinates that will be passed to fragment (v.vertex most of the time)

7. Add TOJ_APPLY_MASK(<input>, <color alpha>); to fragment shader, where <input> is the incoming "vertex to fragment" struct and <color alpha> is the alpha channel of the color that you intend to return at the end.
8. Create regular mask hiearchy and follow the usual standard steps of applying a mask.

INTEGRATION WITH CUSTOM TRANSPARENT SURFACE SHADERS

1. Your surface shader will have to include vertex function for custom vertex calculations. Open the shader you wish to modify.
2. Copy this inside the properties field of the shader:

```
[HideInInspector][PerRendererData][Toggle] _Enabled("Mask Enabled", Float) = 1
[HideInInspector][PerRendererData][Toggle] _ClampHoriz("Clamp Alpha Horizontally", Float) = 0
[HideInInspector][PerRendererData][Toggle] _ClampVert("Clamp Alpha Vertically", Float) = 0
[HideInInspector][PerRendererData][Toggle] _UseAlphaChannel("Use Mask Alpha Channel (not RGB)", Float) = 0
[HideInInspector][PerRendererData][Toggle] _ScreenSpaceUI ("Is this screen space ui element?", Float) = 0
[HideInInspector][PerRendererData]_AlphaTex("Alpha Mask", 2D) = "white" {}
[HideInInspector][PerRendererData]_ClampBorder("Clamping Border", Float) = 0.01
```

3. Add "ToJMasked" = "True" to subshader tags
4. Add #include "Assets/Alpha Masking/ToJAlphaMasking.cginc" to the start of CGPROGRAM
5. Add "float2 tojAlphaUV;" to the struct that will be transferred from vertex to surface function
6. Add TOJ_TRANSFER_MASK(<output>, <input vertex>); to the end of your vertex function
7. Add TOJ_APPLY_MASK(<input>, <color alpha>); to your Surface function, where <color alpha> is the variable which will be assigned to output.Alpha at the end.
8. Create regular mask hiearchy and follow the usual standard steps of applying a mask.

TEXT MESH PRO INTEGRATION (SURFACE SHADERS)

1. Open TMPro_Surface.cginc and add #include "Assets/Alpha Masking/ToJAlphaMasking.cginc" at the start of the file.
2. Add TOJ_TRANSFER_MASK(data, v.vertex); to the end of VertShader function
3. Add TOJ_APPLY_MASK(input, faceColor.a); to the PixShader function, just before the "// Set Standard output structure" comment.
4. Open both TMP_SDF-Surface-Mobile and TMP_SDF-Surface shaders and add this to their properties field:

```
[HideInInspector][PerRendererData][Toggle] _Enabled("Mask Enabled", Float) = 1
```

[HideInInspector][PerRendererData][Toggle] _ClampHoriz("Clamp Alpha Horizontally", Float) = 0
[HideInInspector][PerRendererData][Toggle] _ClampVert("Clamp Alpha Vertically", Float) = 0
[HideInInspector][PerRendererData][Toggle] _UseAlphaChannel("Use Mask Alpha Channel (not RGB)", Float) = 0
[HideInInspector][PerRendererData][Toggle] _ScreenSpaceUI ("Is this screen space ui element?", Float) = 0
[HideInInspector][PerRendererData]_AlphaTex("Alpha Mask", 2D) = "white" {}
[HideInInspector][PerRendererData]_ClampBorder("Clamping Border", Float) = 0.01

5. Add “float2 tojAlphaUV;” to their Input struct
6. Add "ToJMasked" = "True" to their subshader tags
7. Create regular mask hiearchy and follow the usual standard steps of applying a mask.

CONTACT INFORMATION

You can contact the developers of the Alpha Mask by e-mail: info@tagofjoy.lt