Blend Modes

Documentation for version 2.0

Thank you for purchasing Blend Modes! You now have the ability to apply 21 different blend modes to UI, sprites, 3D meshes and particle systems. This documentation will help you to get started using the plugin.

Introduction

Blend modes (or mixing modes) are used to determine how two layers are blended into each other. The default blend mode used for most of the objects in Unity is simply to hide the lower layer with whatever is present in the top layer (so called normal blend mode). Using different blend modes, you can achieve a wide range of effects and create a truly unique look for you game.

Blend Modes is a plugin for Unity, which consists of shaders pack and editor extension that provides an easy way to apply them to the supported object types.

List of all the available blend modes

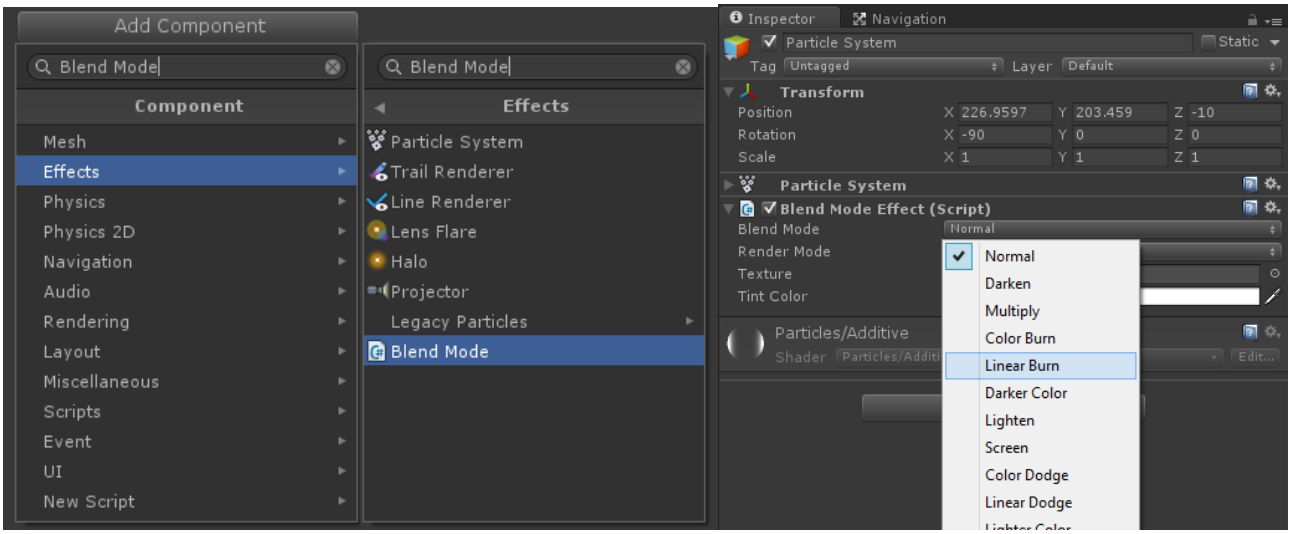
The plugin provides access to 21 different blend modes:

- Darken
- Multiply
- Color Burn
- Linear Burn
- Darker Color
- Lighten
- Screen
- Color Dodge
- Linear Dodge
- Lighter Color
- Overlay
- Soft Light
- Hard Light
- Vivid Light
- Linear Light
- Pin Light
- Hard Mix
- Difference
- Exclusion
- Subtract
- Divide

Getting started

The plugin does not require any special setup: just import the package and you are ready to go! You are also free to move the plugin folder (BlendModes) to any place in the Assets directory.

To apply blending effect to the supported object, just add the “Blend Mode” component and select the desired blend mode. Supported objects are: UI graphic and text (Image, Raw Image or Text components), 2D sprites (Sprite Renderer component), 3D meshes (Mesh Renderer component) and particle systems (Particle System component).



Render mode

The plugin provides three different modes to apply blend effect:

- Grab
- Unified grab
- Framebuffer

Grab mode will execute a grab pass for each object with blend mode effect, which may be too heavy for mobile devices but will work on all platforms. This is the preferred mode for standalone platforms and will be selected by default.

Unified Grab also uses a grab pass, but objects with this mode will not execute a separate one for themselves. Instead, they will share the grab texture of the first performed pass. This will work much faster in case there are multiple objects with blend mode effect, but these objects will not blend with each other.

Framebuffer mode will not use grab pass at all, which is extremely faster and will work smoothly on mobile devices. The drawback is that its support is limited to:

- Apple devices running iOS version 6 or newer
- Android devices based on Nvidia Tegra architecture

Texture and tint color

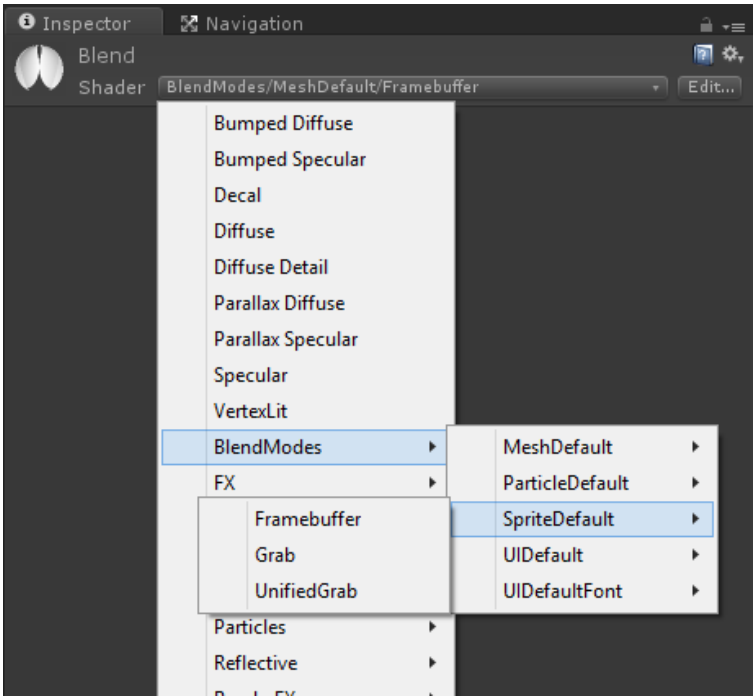
When applying blend effect to the meshes and particle systems, you will see two additional options for setting texture and tint color. This will work just like setting texture and color to the default materials and should not bring any difficulties.

Texture and tint color for UI objects, text and 2D sprites should be set in their respective rendering components.

Using custom blend materials

If you need to apply blend effect to one of the not natively supported objects (say, skinned mesh, trail renderer or some third party GUI solution) you have an option to do so by creating a custom blend material and manually assigning it to the object.

To create blend material, just create a standard one and assign on of the blend shaders to it:

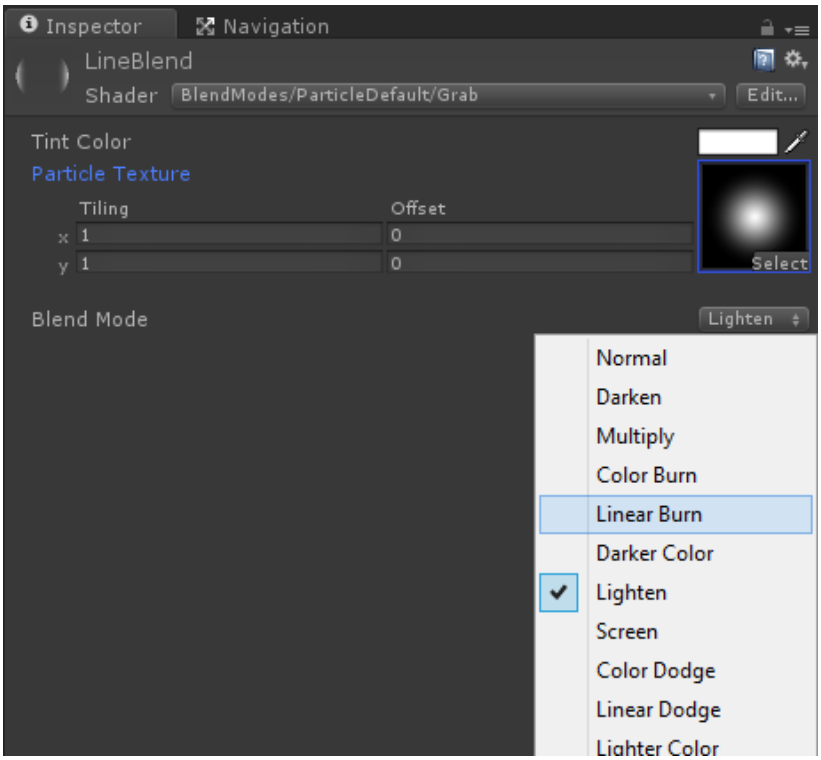


All the blend shaders are split into five categories by the generic type of the renderer they will be used with:

Category	When to use
MeshDefault	Any 3D objects with meshes.
ParticleDefault	Transparent plane objects, usually VFX like line/trail renderers.
SpriteDefault	Sprite objects that need to support pixel snapping.
UIDefault	Any UI graphic.
UIDefaultText	Any UI text (fonts).

In every shader category, there are three shaders for each render mode. Therefore, if you need a trail renderer blend material and you want it to work in Grab mode — you should choose the “BlendModes/ParticleDefault/Grab” shader.

After that, you may choose specific blend mode in the material options:



Setting Blend Mode at runtime

All the code with the plugin is sealed under the BlendModes namespace to prevent conflicts. If you wish to access the Blend Mode scripts at runtime, you will have to include the namespace or use fully qualified names:

```
3   using UnityEngine;
4   // Include BlendMode namespace.
5   using BlendModes;
6
7   public class SomeObject : MonoBehaviour
8   {
9       private void Awake ()
10      {
11          // Get the BlendModeEffect component attached to the object.
12          var blendEffect = GetComponent<BlendModeEffect>();
13
14          // Change blend mode of the object.
15          blendEffect.BlendMode = BlendMode.VividLight;
16
17          // Set render mode.
18          blendEffect.RenderMode = RenderMode.Framebuffer;
19
20          // You may also use this method to set blend and render modes.
21          blendEffect.SetBlendMode(BlendMode.VividLight, RenderMode.Framebuffer);
22      }
23
24  }
25
```

Support and feedback

If you need support for this product or wish to provide suggestions, please feel free to email me at elringus@gmail.com or post at the support forum.

Version changes:

Version 2.0:

- Shaders and editor code overhauled
- Added support for particle systems, sprite and mesh objects
- Added Framebuffer optimization level
- Improved shaders general performance
- Implemented custom blend material editor
- Implemented blend materials auto caching
- Changed package name to "Blend Modes" and store category to Editor Extensions / Effects

Version 1.1:

- Added support for transparent sprites

Version 1.0:

- Initial version