USER GUIDE

Contents

Introduction

Thank you for purchasing!
Highlight Plus is a simple yet powerful package for adding outline, glow and other effects to your gameobjects.

We hope you find the asset easy and fun to use. Feel free to contact us for any enquiry.
Visit our Support Forum on https://kronnect.me for help and access to the latest beta releases.

Kronnect Games
Email: contact@kronnect.me
Kronnect Support Forum: http://www.kronnect.me

Quick Start and Demo Scene

1. Import the asset into your project or create an empty project.
2. Go to Demo folder and run the demo scene to quickly test the asset effects.
3. Examine the code behind the script attached to the Demo game object.

The Demo scene contains 3 spheres with a Highlight Effect and Highlight Trigger scripts attached to each one. Each sphere has:

- *The **Highlight Effect** script contains all the settings and appearance properties for the effects. If you activate the "Highlighted" checkbox, the effects will be rendered immediately.*

- *The **Highlight Trigger** component checks the position of the pointer and detected when it passes over the gameobject. When this occurs, it activates the "Highlighted" checkbox of the previous component and disables it when the pointer exits the gameobject.*

Alternatively, you can create a "**Highlight Manager**" from the top menu GameObject -> Effects -> Highlight Plus -> Create Manager. This command will create a gameobject with the Highlight Manager script attached, responsible for detecting mouse interaction with any gameobject that matches the layer and other settings in the manager and highlight it accordingly.

How to use the asset in your project

Option 1: Highlighting/customizing gameobjects

- Add HighlightEffect.cs script to any gameobject. Customize the appearance options.
- Optionally add HighlightTrigger.cs script to the gameobject. It will activate highlight on the gameobject when mouse pass over it. A collider must be present on the gameobject. Note: adding a HighlightTrigger.cs script to a gameobject will automatically add a HighlightEffect component.

In the Highlight Effect inspector, you can specify which objects, in addition to the parent, are also affected by the effects. The “**Include**” property in the inspector allows:

a) Only this object
b) This object and its children
c) All objects from the root to the children
d) All objects belonging to a layer

Option 2: Highlighting/customizing ANY gameobject automatically

- Select top menu GameObject -> Effects -> Highlight Plus -> Create Manager.
- Customize behaviour of Highlight Manager. Those settings wil be applied to any gameobject highlighted by the manager. But if a gameobject already has a HighlightEffect component, the manager will use those settings instead.

Ignoring specific gameobjects from highlighting

In addition to the “Include” options in the inspector, you can add a “Highlight Effect” component to the gameobject that you don’t want to be highlighted and activate the “Ignore” checkbox.
If you’re using the Highlight Manager, it also provides some filter options like Layer Mask.

Advanced Topics and Notes

Quick help & tips

Highlight Plus has been designed to be used without having to read a long manual. Hover the mouse over the label of any option in the inspector to reveal a tooltip with a short explanation of that option.

This section contains specific instructions or notes about certain features.

Preserve Original Mesh

Highlight Plus automatically smoothes normal of meshes to improve the outline effect when using the Fast or High-quality level (not with Highest). It won't affect your gameobject mesh as it creates a copy of that mesh and caches it so this operation only occurs once. However, in some circumstances you may want to modify your mesh dynamically and also want Highlight Plus to avoid caching that mesh – enable "Preserve Original Mesh" to force Highlight Plus to use the original mesh always.

Highlight when entering a volume

It's possible to automatically enable/disable highlight effects when object enters/exits a volume. Add a HighlightTrigger component to the object and select "Volume" as trigger mode.

The script uses the OnTriggerEnter / OnTriggerExit events. The volume must have a collider marked as IsTrigger and static. Also, the object entering the volume must have a rigidbody in order for the events to trigger.

Depth Clip option

When using Outline or Outer Glow in high quality mode, the asset relies on the depth buffer to perform proper depth cull or depth clipping. However, when MSAA (integrated antialias) is enabled, the depth buffer is not available at the stage Highlight Plus renders. If you need MSAA in your project and wants depth clipping/culling, you can enable the "Depth Clip" option in the Highlight inspector. This option only is used when Outline or Outer Glow is set to High Quality (the other quality modes work differently and do not require special treatment).

Compatibility of transparent objects with depth clip option

The Depth Clip option is only available for Outline and Outer Glow in High Quality modes. When enabled, the special _CameraDepthTexture buffer is used to clip the effect. This option is useful if you need to use MSAA since enabling MSAA disables depth checking in high quality mode.
Transparent objects do not write to this special texture unless you use this option in GameObject -> Effects -> Highlight Plus -> "Make Transparent Object Compatible With Depth Clip".
Once you add this feature to your transparent object, make sure you also have "Depth Clip" option enabled in the objects to be highlighted using Outline or Outer Glow in HQ mode.

Compatibility of see-through effect with transparent shaders

If you want the See-Through effect be seen through other transparent objects, their shaders need to be modified so they write to depth buffer (by default transparent objects do not write to z-buffer). To do so, select top menu GameObject -> Effects -> Highlight Plus -> “Add Depth To Transparent Object”.
Note that forcing a transparent object to write to depth buffer will cause issues with transparency.

Static batching

Objects marked as "static" need a MeshCollider in order to be highlighted (other collider types won’t work). This required because Unity combines the meshes of static objects so it's not possible to access to the individual meshes of non-batched objects.
Note: the MeshCollider can be disabled.

Using scripting to add effects

Use GetComponent<HighlightEffect>() to get a reference to the component of your gameobject.
Most properties shown in the inspector can be accessed through code, for example:

```
using HighlightPlus;
…

HighlightEffect effect = myGameObject.GetComponetn<HighlightEffect>();
effect.outline = true;
effect.outlineColor = Color.blue;
effect.UpdateMaterialProperties();
```

Important! If you change the hierarchy of your object (change its parent or attach it to another object), you need to call `effect.Refresh()` to make Highlight Plus update its internal data.

Executing a Hit FX effect

The HitFX effect is a fast flash overlay effect which is used from code. Just call HitFX and pass the desired parameters:

```
using HighlightPlus;
…

HighlightEffect effect = myGameObject.GetComponetn<HighlightEffect>();
effect.HitFX(color, duration, initial_intensity);
```

Changing properties at runtime

When changing specific script properties at runtime, call UpdateMaterialProperties() to ensure those changes are applied immediately.

Cancelling see-through effect behind certain objects

Add HighlightSeeThroughOccluder script to the object you want to block see-through effects.

Setting profile at runtime

Call ProfileLoad() or ProfileReload() methods of the HighlightEffect component and pass your highlight profile object.

Excluding submeshes

Use the SubMesh Mash property to specify which submeshes will be affected by the effect.

By default, a value of -1 means all submeshes. This is a component mask field. The effect does the following test to determine if the submesh will be included:

(1<<subMeshIndex) & SubMeshMash != 0

Examples:
If you want to only include submesh 1 (index 1), set it to 1 (1<<1).
If you want to include only submesh 2, set it to 2 (1<<2).
For submesh 3, set it to 4 (1<<3).
For submeshes 2 and 3, set it to 6 (1<<2 + 1<<3).
In general, this works like the layer mask or culling mask in rest of Unity.

Events / reacting to selection

Note: check SphereHighlightEventExample.cs script in the demo scene.

Example code:

```
using UnityEngine;
using HighlightPlus;

public class SphereHighlightEventExample : MonoBehaviour {

   void Start() {
       HighlightEffect effect = GetComponent<HighlightEffect> ();
       effect.OnObjectHighlightStart += ValidateHighlightObject;
   }


   bool ValidateHighlightObject(GameObject obj) {
       // Used to fine-control if the object can be highlighted; return false to
cancel highlight
       return true;
   }

}
```

Messages

Highlight Effect will send the “HighlightStart” and “HighlightEnd” to any script attached to the gameobject when its highlighted (or highlight ends). You can get those messages using the following code:

```
using UnityEngine;
using HighlightPlus;

public class MyBehaviour : MonoBehaviour {

  void HighlightStart () {
      Debug.Log ("Object highlighted!");
  }

  void HighlightEnd () {
      Debug.Log ("Oject not highlighted!");
  }

}
```

Universal Rendering Pipeline compatibility

URP support is limited in this package. A version of Highlight Plus designed for URP can be downloaded from our support forum on https://kronnect.me. Please sign up and send us an email with your username to get access to the private board.