USER GUIDE FOR UNIVERSAL RENDERING PIPELINE

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

Setup

Video instructions: https://youtu.be/OlCnEAcHJm0

Steps:

- Make sure you're using Unity 2019.3 and URP 7.1.8 or later.
- Import the package for URP in your project.
- Create or select the Universal Rendering Pipeline asset
- Create a new Forward Renderer (or use the renderer included in the package in the folder HighlightPlus/Pipelines/URP)
- If you create a new forward renderer, just click the "+" symbol below and add the Highlight Plus Rendering Pass Feature.

Demo Scene

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

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

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.

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 ("Object not highlighted!");
  }

}
```