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.com for help and access to the latest beta releases.

Kronnect Technologies
Email: contact@kronnect.com
Support Forum: https://www.kronnect.com/support

Quick Start and Demo Scenes

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

The Demo scenes 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
e) Custom targets specified by script (see Advanced Topics & Notes)

Option 2: Highlighting/customizing ANY gameobject automatically

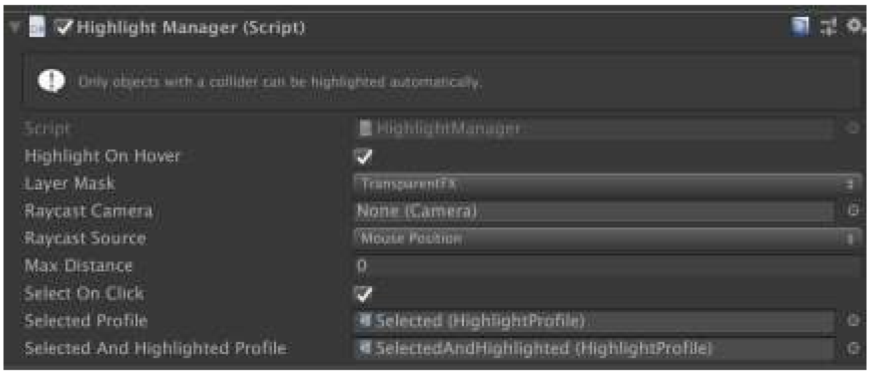
- Select top menu GameObject -> Effects -> Highlight Plus -> Create Manager.
- Customize behaviour of Highlight Manager. Those settings wil be applied to any gameobject highlighted by the manager. But if a gameobject already has a HighlightEffect component, the manager will use those settings instead.

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.

Highlighting vs Selection

Highlight Plus offers a Selection feature integrated in the Highlight Manager, which can draw objects differently depending if they're selected or not. You can select any number of objects by clicking on them.



Highlight Manager (Script)

Only objects with a collider can be highlighted automatically.

Script	HighlightManager
Highlight On Hover	✓
Layer Mask	TransparentFX
Raycast Camera	None (Camera)
Raycast Source	Mouse Position
Max Distance	0
Select On Click	✓
Selected Profile	Selected (HighlightProfile)
Selected And Highlighted Profile	SelectedAndHighlighted (HighlightProfile)

Highlighting

Refers to the temporary effect that causes an object to show an outline, glow or other kind of effect. For example, when using the Highlight Trigger or Highlight Manager components, and the "Highlight On Hover" checkbox is activated, the asset will "highlight" any target automatically when the pointer is positioned over that target. When you move the pointer out of the object, the highlight disappears.

You can also use scripting to toggle the highlight on / off by calling **SetHighlighted**() or setting the **highlighted** property of the HighlightEffect component.

The demo scene 1 provides an example of highlighting spheres.

Selection

Refers to the ability to keep the highlight visible when you click on objects. To enable the new selection feature, tick the option "Select On Click" and assign a "selected" profile and a "selected and highlighted" profile:

- Selected Profile: the profile you want to use to visualize an object that's selected only (not highlighted).
- Selected and Highlighted Profile: the profile to use when the object is both selected and highlighted.

The Highlight Effect attached to the Highlight Manager contains the default highlight settings which are used to highlight objects that are not selected.

The Highlight Manager script exposes a "selectedObjects" public property which can be used with scripting to get a list of the currently selected objects

Use the **SelectObject**, **ToggleObject** and **UnselectObject** methods of the Highlight Manager to manually control the selection state of any object in the scene using scripting.

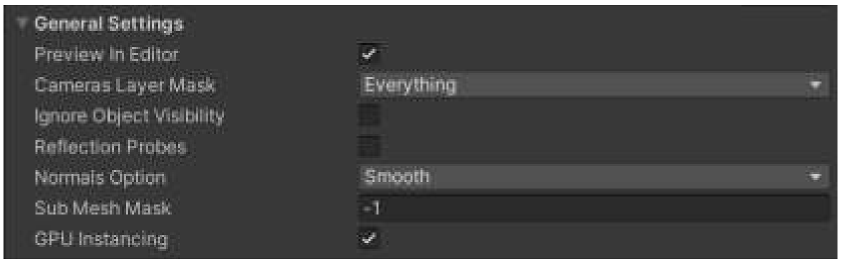
The demo scene 2 provides an example of selection by clicking on spheres.

Full list of options in Highlight Effect component

The Highlight Effect component provides many options. This section covers each one of them:

- **Profile**: allows you to assign a profile with custom settings. You can create a profile by clicking "Create", customize that profile and reuse it on any number of highlight effect components.

General Settings

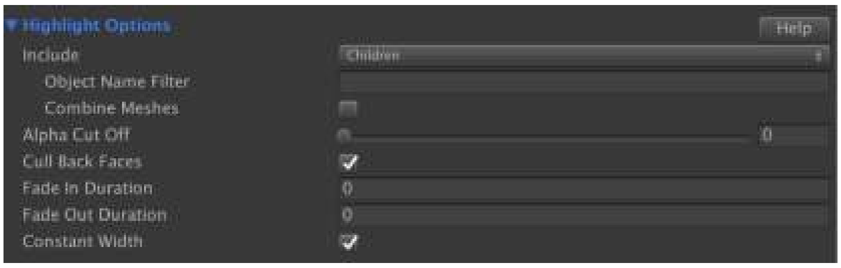


General Settings

Preview In Editor	✓
Cameras Layer Mask	Everything
Ignore Object Visibility	
Reflection Probes	
Normals Option	Smooth
Sub Mesh Mask	-1
GPU Instancing	✓

- **Preview In Editor**: enabling this option will render the effects while in Edit mode. Note that most effects won't be visible unless the "**Highlighted**" checkbox is enabled.
- **Cameras Layer Mask**: let you specify which cameras can render the effects. By default, every camera.
- **Ignore Object Visibility**: by default, Highlight Plus won't render the effects if the object is not visible in screen. If you're using some tool that uses GPU instancing to render the object directly on the GPU and not using the regular renderer component, enable this option to force the effects to be rendered.
- **Reflection Probes**: enable to render the effects on reflection probe cameras.
- **Normals Option**: several options to smooth or reorient normals of the geometry of the object. Only used in outline and glow effects when rendering in Fast, Average or High quality mode (mesh-based effects, not used in Highest quality modes or screen-space mode).
- **Sub Mesh Mask**: bitwise mask that filters which sub-mesh will be highlighted. Useful for objects that have multiple sub-meshes.
- **GPU Instancing**: enabled by default, improves rendering performance of outline and glow effects.

Highlight Options



Highlight Options

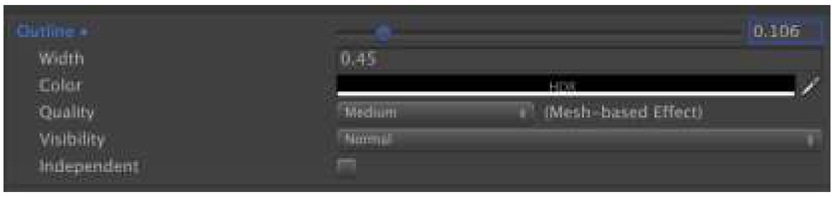
Include	Children
Object Name Filter	
Combine Meshes	
Alpha Cut Off	0
Cull Back Faces	✓
Fade In Duration	0
Fade Out Duration	0
Constant Width	✓

- **Ignore**: let you disable any effect on this object. If you use the Highlight Manager, it will still use this highlight effect component values.
- **Include**: this option let you specify which other objects will also be highlighted. It's extremely useful as you can using a single highlight effect on the root of a group of objects for example, and include all the children. The outline and glow effects will be combined and show a single contour around the entire group if they overlap.

    - **Object Name Filter**: let's you specify which objects will be included (according to the Include option) by filtering their names. For example, if you have 5 children, you could use a prefix or suffix to specify which of them will be highlighted.
    - **Combine Meshes**: this option is useful for objects that do not rotate individually. Highlight Plus will automatically combine the meshes of the group, reducing the number of draw calls dramatically.
- **Alpha Cut Off**: applies an alpha cutoff to the effects, useful when you're highlighting semi-transparent objects or objects that exhibit holes or transparent parts in their textures like leaves, fences, etc.
- **Cull Back Faces**: do not apply the effects to back faces. Enabled by defauflt.
- **Fade In/Out Duration**: applies a "lerping" effect for the given duration.
- **Constant Width**: keeps the outline or glow width constant regardless of the distance of the object to the camera.
- **Depth Clip**: performs a depth buffer clipping so effects are correctly hidden behind solid geometry.

Outline Effect Options

The outline shows a usually thin colored line around the object.

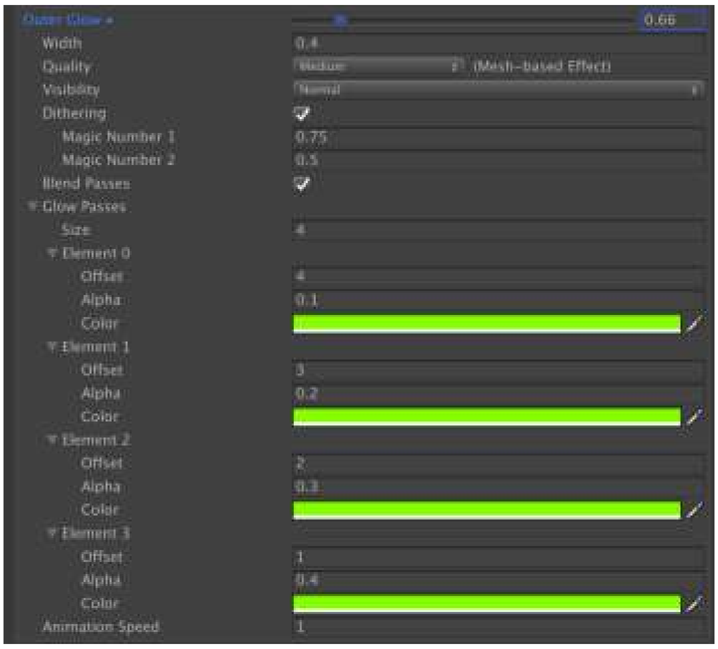


Outline	0.106
Width	0.45
Color	HDR
Quality	Medium (Mesh-based Effect)
Visibility	Normal
Independent	☐

- **Outline**: controls the transparency of the outline. A value of 0 deactivates completely this effect.
- **Width**: the width of the outline. The "Constant Width" option in the previous section can affect the actual visible width.
- **Color**: the color for the outline.
- **Quality**: there're 4 quality modes: fast, average, high and highest. The first 3 quality modes use a mesh-based rendering technique which is usually faster. The Highest quality mode uses a screen-space effect which provides a smoother result. The quality modes and the actual result can vary depending on the characteristic of the highlighted object so don't hesitate to use the mode that looks or perform better in your case, regardless of the name. For example, the "Fast" quality mode usually works best for simple objects.
- **Visibility**: controls the z-buffer testing for the outline. Let you specify if the outline should be visible on top of everything, only when the object is occluded (check also see-through effect for more options) or normal, which means the outline will be visible when the object is visible as well.
- **Optimal Blit**: when in Highest Quality mode, performs a blit over the affected region of the screen, instead of using a full-screen blit. This option will improve performance.
- **Independent**: this option forces the outline to show completely ignoring other highlighted objects that could overlap on the screen. By default, Highlight Effect will merge the outlines of overlapping objects, showing a continuous outline around the mixed group. By enabling the Independent option you ensure that the outline is visible around each group separately.

Outer Glow Effect Options

The outer glow effect shows a bright glow or bloom-like effect, usually wider than the outline, around the object.



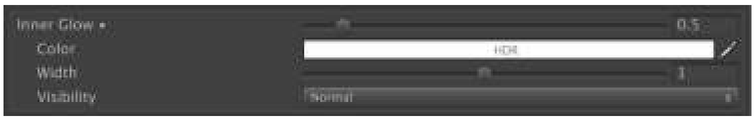
Outer Glow	0.66
Width	0.4
Quality	Medium (Mesh-based Effect)
Visibility	Normal
Dithering	✓
Magic Number 1	0.75
Magic Number 2	0.5
Blend Passes	✓
Glow Passes	
Size	4
Element 0	
Offset	4
Alpha	0.1
Color	🟩
Element 1	
Offset	3
Alpha	0.2
Color	🟩
Element 2	
Offset	2
Alpha	0.3
Color	🟩
Element 3	
Offset	1
Alpha	0.4
Color	🟩
Animation Speed	1

- **Outer Glow**: controls the transparency of the outer glow. A value of 0 disables this effect.
- **Width**: the width of the glow. Check the "Constant Width" setting in Highlight Options section because it can affect the actual width displayed.
- **Visibility**: controls the z-buffer testing for the effect. Let you specify if the glow should be visible on top of everything, only when the object is occluded or normal, which means the glow will be visible when the object is visible as well.
- **Dithering**: only used in mesh-based quality modes, let you control the style of the dithering used in the glow. This is a pure and optional artist-driven effect. You can disable it to show a solid glow.
- **Blend Passes**: in mesh-based quality modes, the outer glow is achieved by rendering the object mesh using several custom passes. When this option enabled, each pass will be blended with the previous one. If the option is disabled, each pass is rendered additively.
- **Glow Passes**: let you configure the number and characteristic of each glow pass including color, offset and transparency (alpha).
- **Animation Speed**: the outer glow will animate if this value is greater than 0.

To quickly change the glow color using scripting, you can call **SetGlowColor(color)** method on the HighlightEffect component. Check demo scene 1 for an example.

Inner Glow Effect Options

The Inner Glow renders a rim-like effect over the object.

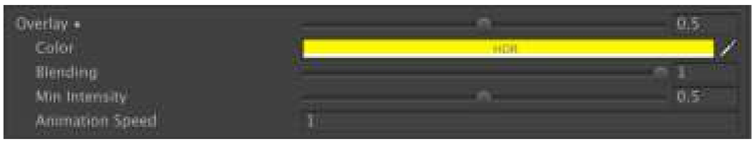


Inner Glow •	0.5
Color	HDR
Width	1
Visibility	Normal

- **Inner Glow**: controls the transparency of the inner glow effect which renders inside the object, like a rim lighting effect. A value of 0 disables this effect.
- **Color**: the color for this effect.
- **Width**: the amount of inner glow.
- **Visibility**: controls the z-buffer testing for the effect. Let you specify if the glow should be visible on top of everything, only when the object is occluded or normal, which means the glow will be visible when the object is visible as well.

Overlay Effect Options

The Overlay effect adds a solid color over the object with custom transparency.

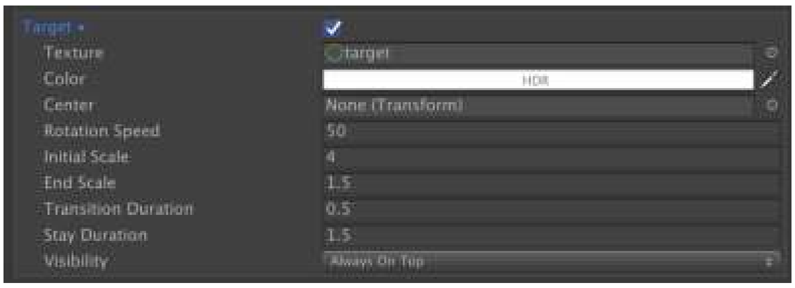


Overlay •	0.5
Color	HDR
Blending	1
Min Intensity	0.5
Animation Speed	1

- **Overlay**: controls the transparency of the effect. A value of 0 disables this effect.
- **Color**: the color of the effect.
- **Blending**: determines how much the color is blended with the original color/texture of the object.
- **Min Intensity**: minimum intensity for the blending.
- **Animation Speed**: the speed of the overlay "pulse". Enter 0 to disable animation.

Target Effect Options

The Target effect adds an animated object floating over the highlighted target.



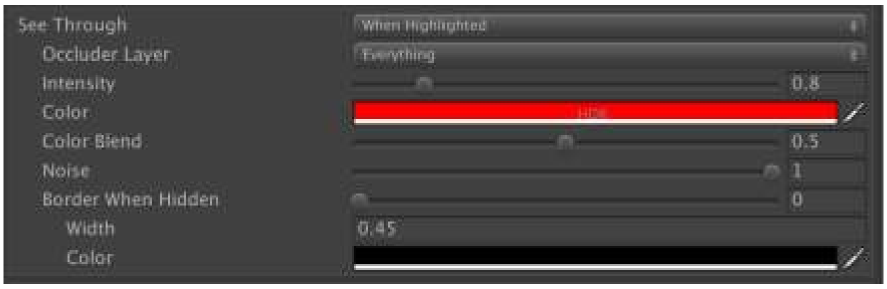
Target •	✓
Texture	target
Color	HDR
Center	None (Transform)
Rotation Speed	50
Initial Scale	4
End Scale	1.5
Transition Duration	0.5
Stay Duration	1.5
Visibility	Always On Top

- **Texture**: the overlay texture to be used for this effect.
- **Color**: the tint color for such texture.
- **Center**: where the target should be positioned. If not set, the center of the highlighted object will be used. However, if the object has multiple parts or children, you could assign a transform of a child instead (ie. part of the body of a monster).
- **Rotation Speed**: the speed at which the object rotates. Enter 0 to disable rotation.
- **Initial / End Scale**: used to create a zoom / pulsating effect.

- **Scale To Object Bound**: the scale of the target image will be relative to the object screen bounds (if object grows, the target effect will also expand). Note that if object rotates and its bounds change, the target effect will also change its size. This can produce a strange result so you may prefer to have this option disabled for rotating characters/objects.
- **Transition Duration**: the duration of the transparency transition when the overlay texture appears/disappears.
- **Align To Ground**: projects the target sprite on the ground (as a decal). In this mode, vertical walls will be detected by using the camera depth texture and reconstructed normal of the scene so the decal won’t appear stretched on the wall. Use the Fade Power option to attenuate the target sprite with elevation.
- **Stay Duration**: how long the overlay texture will be displayed when the object is no longer highlighted until it fades out. A value of 0 keeps the effect visible until it’s deactivated.
- **Visibility**: controls the z-buffer testing for the effect. Let you specify if the glow should be visible on top of everything, only when the object is occluded or normal, which means the glow will be visible when the object is visible as well.

See-Through Effect Options

The see-through effect, also known as x-ray effect, shows when the object is occluded by other elements, like walls.

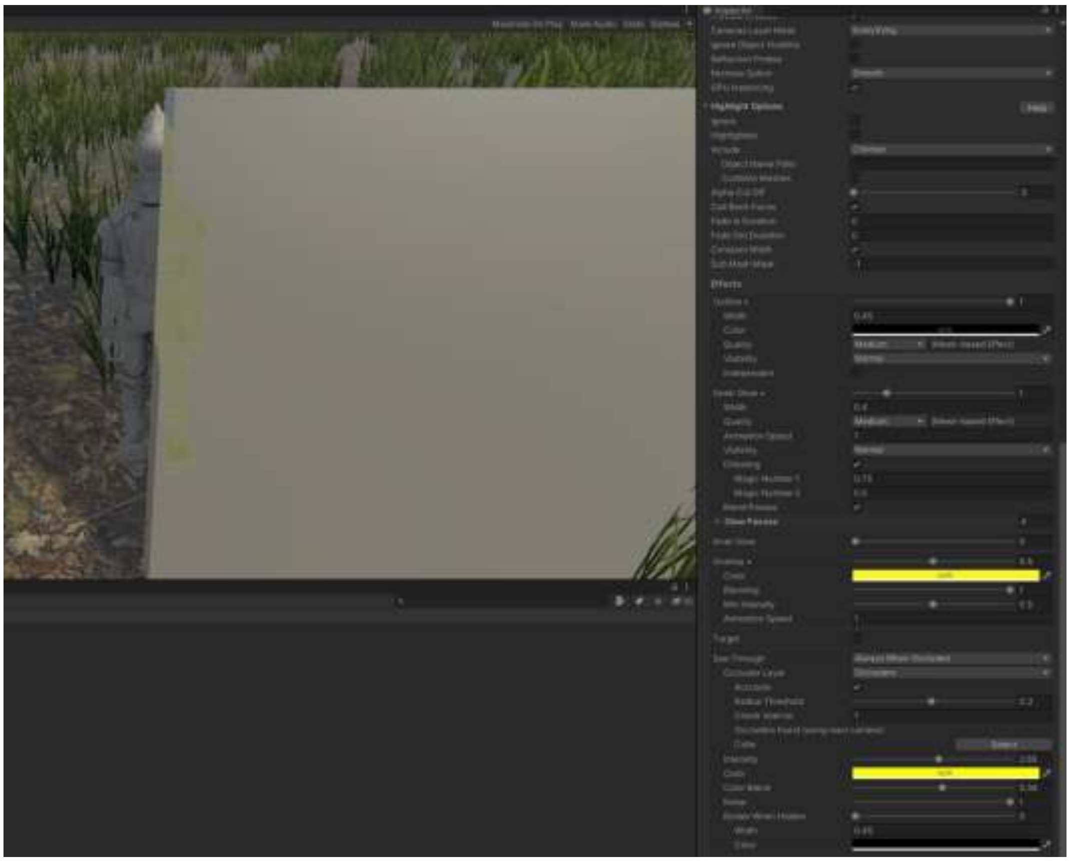


See Through	When Highlighted
Occluder Layer	Everything
Intensity	0.8
Color	🟥 HDR
Color Blend	0.5
Noise	1
Border When Hidden	0
Width	0.45
Color	⬛

- **See Through**: determines if the see-through will be visible only when the object is highlighted, always when it’s occluded or never. Use “Never” to completely disable this effect.
- **Occlude Layer**: if this value is different to Everything, the asset will use raycasting to try to determine if an object belonging to a matching layer is occluding the highlighted object. If it hits an occluding object, the see-through effect will be activated.
  - **Threshold**: when using the occlude layer, the object bound size is multiplied by this value when testing occlusion. You can use this value to avoid false positives and only show the see-through effect when the object is really behind another solid object.
  - **Check Individual Objects**: by default, Highlight Plus takes all children affected by the see-through effect of this component and combines the bounds before testing occlusion. If this option is enabled, the occlusion test is performed per each individual child.
  - **Depth Offset / Max Depth**: provides distance limitations to the effect (distance referred to the occluder). A value of 0 will not use these options.
  - **Accurate**: when using the occlude layer, this option enables pixel-perfect occlusion.
- **Intensity**: intensity of brightness of the effect.
- **Color & Color Blend**: controls the color and intensity of the blending of the color with the original color/texture of the object.
- **Noise**: optional, artist-driven, effect.
- **Border When Hidden**: this option let you show an additional outline but only when see-through is active.

- **Ordered**: when enabled, it will respect the relative distance to camera when rendering see-through effect.

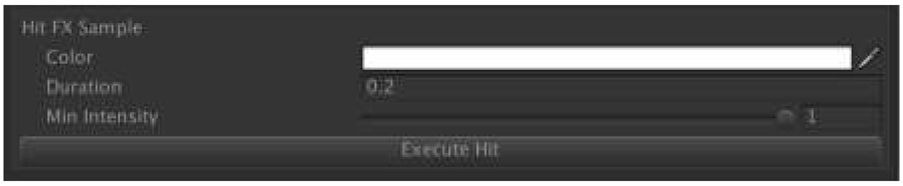
A common scenario when using the See-Through effect is when you want your character to be occluded by walls and not ground or ceilings. A solution involves assigning your walls to a different layer than the ground/ceilings and specify that layer in the Occluder Layer of the See-Through effect. Enable the "Accurate" option for best results:



In the screenshot above, you can see the see-through effect is visible only over the wall but not over the grass or terrain.

Hit-FX Effect Options

This effect is only usable in play mode. It let you show a "punch" effect by changing the color of the hit object briefly. In play mode you can test it:



Hit FX Sample
Color
Duration 0.2
Min Intensity 1
Execute Hit

Please refer to the "Executing a Hit-FX effect" in the next section for sample code.

Scripting support

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.SetGlowColor(Color.yellow);
effect.UpdateMaterialProperties();
```

To control the state of the highlight, use:
`effect.SetHighlighted(true/false)` or `effect.highlighted = true/false`.

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.

Setting profile at runtime

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

Changing properties at runtime

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

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);
```

Starting the Target FX on demand

Call `effect.StartTargetFX()` method to start the target FX whenever you wish.
Note: if you change target fx properties using scripting, you may need to call `UpdateMaterialProperties()` so the new values get reflected in new target fx executions.

Changing the target of highlight using scripting

The objects to be highlighted by the script are those configured by the “Include” option in the Highlight Options section. However, you can override this selection using scripting in two ways:

`effect.SetTarget(transform)` will instruct the script to execute the effects on the given object.
`effect.SetTarget(transform, Renderer[] additionalObjects)` will do the same but also will include any number of other renderers. Useful when you want to highlight a set of objects defined by a script.

Events / reacting to highlighting

Note: check **SphereHighlightEventExample.cs** script in the demo scene.
The Highlight Effect script exposes two events (OnObjectHighlightStart and OnObjectHighlightEnd) which fire when the object highlight starts or ends.

Example:

```
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;
   }

}
```

These two events are also exposed by the HighlightManager and the HighlightTrigger.
For example, when using the HighlightManager, you can also hook into its OnObjectHighlightStart, OnObjectHighlightEnd, OnObjectSelected, OnObjectUnselected events:

```
HighlightManager.instance.OnObjectHighlightStart += MyEventHandler;
```

Events / reacting to selection

Selection events **OnObjectSelected** and **OnObjectUnSelected** are provided by the HighlightManager or HighlightTrigger components. Check demo scene 2 for a working sample (using the code below).

Using the Highlight Manager

```
using UnityEngine;
using HighlightPlus;

public class SelectionEventExample : MonoBehaviour {

   void Start() {
       HighlightManager.instance.OnObjectSelected += SelectObject;
       HighlightManager.instance.OnObjectUnSelected += UnSelectObject;
   }


   bool SelectObject (GameObject obj) {
       // Used to fine-control if the object can be selected; return false to cancel
selection
       return true;
   }

   bool UnSelectObject (GameObject obj) {
       // Used to fine-control if the object can be unselected; return false to cancel
unselection
       return true;
   }
}
```

Using the Highlight Trigger

```
using UnityEngine;
using HighlightPlus;

public class SelectionEventExample : MonoBehaviour {

   void Start() {
      HighlightTrigger ht = GetComponent<HighlightTrigger> ();
       ht.OnObjectSelected += SelectObject;
       ht.OnObjectUnSelected += SelectObject;
   }


   bool SelectObject (GameObject obj) {
       // Used to fine-control if the object can be selected; return false to cancel
selection
       return true;
   }

   bool UnSelectObject (GameObject obj) {
       // Used to fine-control if the object can be unselected; return false to cancel
unselection
       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!");
  }

}
```

Manually selecting/toggling/unselecting objects with Highlight Manager

The Highlight Manager provides a “selection” state functionality which you can invoke using the methods SelectObject, ToggleObject and UnselectObject of the Highlight Manager. The following code is extracted from demo scene 2:

```
using UnityEngine;
using HighlightPlus;

namespace HighlightPlus.Demos {

    public class ManualSelectionDemo : MonoBehaviour {

        HighlightManager hm;

        public Transform objectToSelect;

        void Start() {
            hm = FindObjectOfType<HighlightManager>();
        }

        void Update() {
            if (Input.GetKeyDown(KeyCode.Alpha1)) {
                hm.SelectObject(objectToSelect);
            }
            if (Input.GetKeyDown(KeyCode.Alpha2)) {
                hm.ToggleObject(objectToSelect);
            }
            if (Input.GetKeyDown(KeyCode.Alpha3)) {
                hm.UnselectObject(objectToSelect);
            }
        }
    }
}
```

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 specific features or issues.

Normals Option

Highlight Plus can automatically optimize mesh normal to provide a better result. The available options are:

- **Smooth**: this will improve the outline effect when using the Fast or High-quality level (not with Highest).
- **Preserve Original**: 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.
- **Reorient**: this will replace existing normal with a vector pointing out from the center of the object. Useful for some 2D geometry or objects without normals.

Note that Highlight Plus won't never change the original mesh of your objects so these are safe operations.

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.

Masking UI

To avoid Highlight Plus effects show on top of any UI or sprite element, there’re a few options:

a) Use “Screen Space Overlay” mode for canvas rendering. This mode renders the UI at the end of the render loop so the UI appears on top of any highlight plus effect.
b) Use a second camera to render your world-space UI. Set the culling mask of your main camera and the “UI camera” accordingly (so the UI camera only renders the UI) and set the UI Camera clear flag to “Depth only”. Make sure the “Depth” value of the UI camera is greater than the regular camera so the UI camera renders after the main camera. This setup is easy and ensures the world-space UI renders on top of any highlight plus effect.
c) Create a panel on your UI and assign the material HighlightUIMask found in HighlightPlus/Resources/Materials. This special material will write to stencil buffers and will prevent Highlight Plus effects from rendering over that panel. This panel won’t be visible, it will just serve the purpose of masking that portion of screen.

Static batching

Objects marked as "static" need a MeshCollider 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. The only purpose of this collider is to allow Highlight Effect to get access to the original mesh before the merge performed by Unity when scene starts.

Cancelling see-through effect behind certain objects

Add **HighlightSeeThroughOccluder** script to the object you want to block see-through effects.
The *HighlightEffect.isSeeThroughOccluded* will return true if any occluder using raycast mode is covering the highlighted 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.

Custom Vertex Transformations

You can apply your own vertex transformations inside the "CustomVertexTransform.cginc" file. The function "ComputeVertexPosition" is used by all Highlight Plus shader so this is a convenient centralized place where to include your custom vertex transforms.

Please note that any addition or change to this file will be lost if you upgrade Highlight Plus.

Controlling which objects can be selected

The Highlight Manager exposes two events, OnObjectSelected and OnObjectUnSelected, which are fired when user selects/unselects objects. These events can be used to cancel a selection or deselection depending on the value returned by the even handler (a Boolean):

```
using UnityEngine;
using HighlightPlus;

public class ControlExample : MonoBehaviour {

   void Start() {
       HighlightManager manager = FindObjectOfType<HighlightManager> ();
       manager.OnObjectSelected += ValidateSelection;
   }


   bool ValidateSelection(GameObject obj) {
       // Used to fine-control if the object can be selected; return false to cancel
selection
       return true;
   }

}
```

Custom sorting

Highlight Plus effects render after transparent queue by default in any order. However, you can control ordering of the effects by adding this script to the scene:

```
using UnityEngine;
using HighlightPlus;

[ExecuteAlways]
public class CustomRendering : MonoBehaviour {

    void OnEnable() {
        HighlightEffect.customSorting = true;
    }

    void Update() {
        HighlightEffect.effects.Sort(Comparer);
    }

    int Comparer(HighlightEffect effect1, HighlightEffect effect2) {
        float dist1 = effect1.transform.position.z;
        float dist2 = effect2.transform.position.z;
        return dist2.CompareTo(dist1);
    }
}
```

In this case, the effects will be sorted by the z position. Feel free to sort the "effects" list above according to your needs.

Universal Rendering Pipeline compatibility

URP support is limited in this package. A version of Highlight Plus designed for URP can be found in the same asset (just import it from the URP folder).