2D LAYER ORGANIZER

Thank you for getting the asset! Be sure to

leave a heart if you like it.

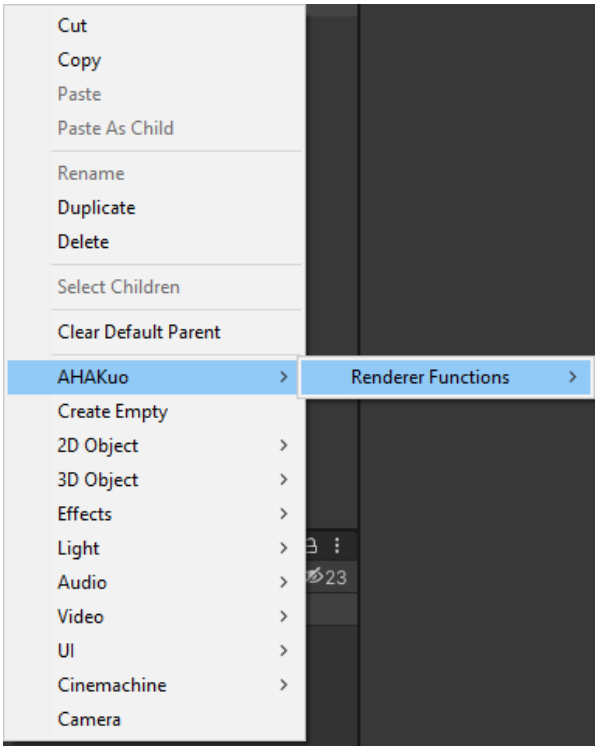
This document will set you right up.

Contents of documentation

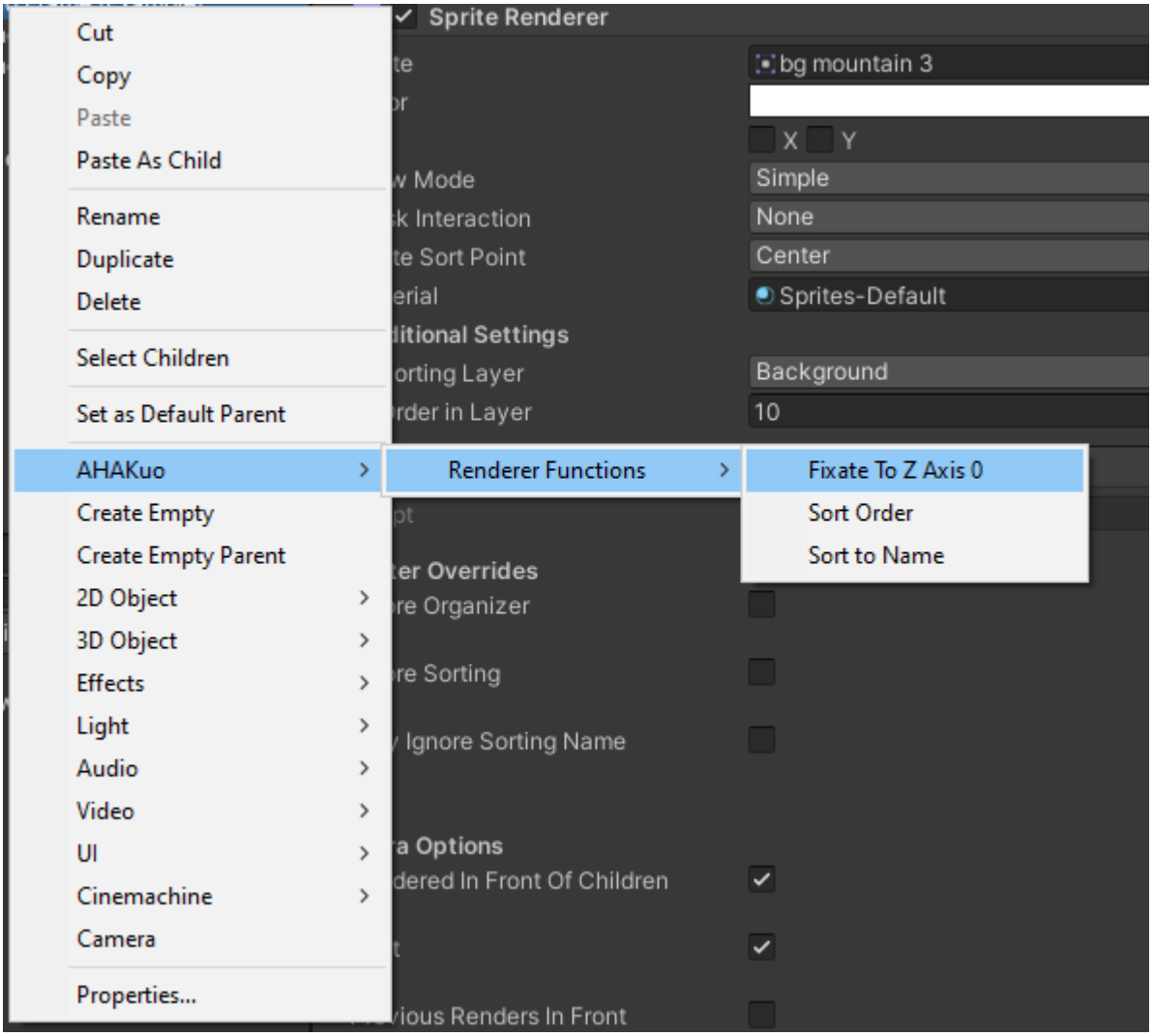
Quick Setup:

One of the best parts of setting up this layer organizer system is that there is no setup. You just download it, import it, and it's ready!

1. **First, go to a parent object that contains the sprites you want to sort.**
2. **Right-click on the parent object then select the following:**



3. **Select your preferred type of sorting. This won't matter much as you can change the pre-determined settings after the sorter is added:**



- **Fixate to Z Axis 0:** This tool simply makes the selected object not movable on the Z axis. This is

helpful to place on objects like the player or enemies as they may accidently be moved on the z axis. This script is pretty straightforward.

- **Sort Order:** This will add a layer organizer to the selected object, and will then sort all the renderers inside it and its children. It will only sort the order of renderers and not their sorting layer name.
- **Sort to Name:** This will add a layer organizer to the selected object, and will then sort all the renderers inside it and its children by order and by name. You can change the name field inside the script settings itself. Simply type out the sorting layer name you want to sort to, if it exists, it will sort.

Learn how to add more options to the context screen for your specific layers later in the doc.

You are done! Basic sorting should be working instantly.

How the sorting system works:

Higher objects will be sorted farther away from the eye. So, the first object in the hierarchy will be sorted to 0 or the start index you selected.

The system sorts all renderer types (sprites, meshes, particles, etc...) and simplifies the process of level design. It adds a RendererMarker script to all renderers in the object, and this renderer marker script is responsible for handling the object.

The design of the system works per-one-object basis, this allows you to handle the sorting system in large scenes that contain a lot of sprites that are divided into rooms.

Since having only one place for sorting may not be easy to handle when your scene has many sprites, this system sorts within the hierarchy of objects itself.

All you have to do to change the order of sorting is moving the object up or down in the hierarchy.

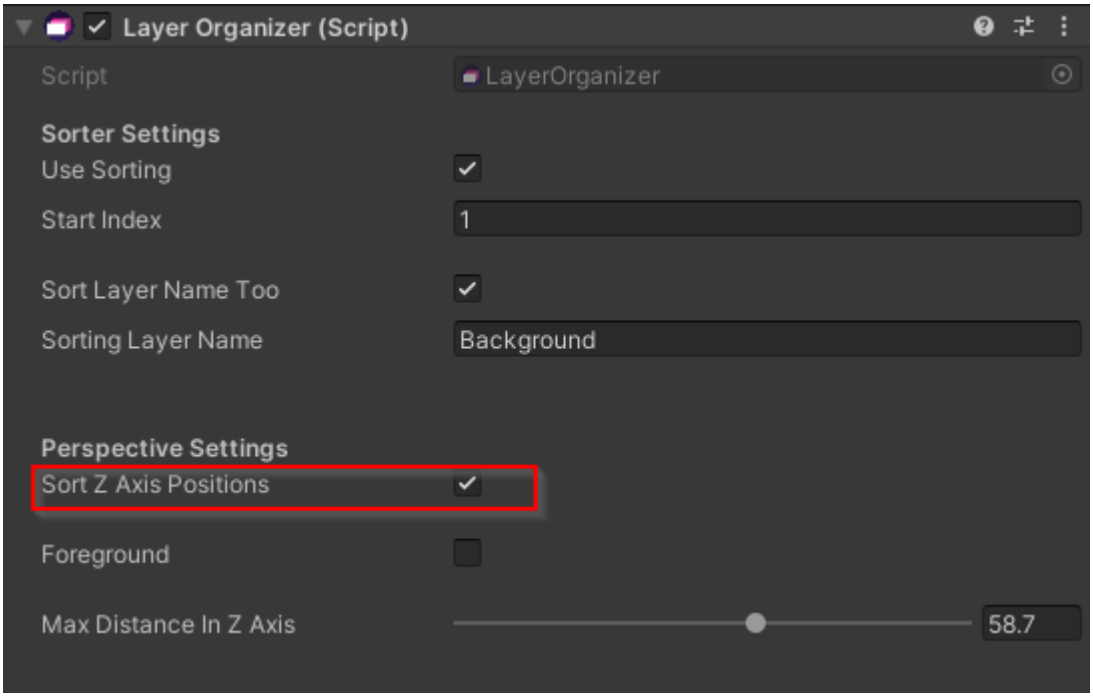
How to use perspective Z axis sorting:

Layer organizer comes with a really useful tool in case you were building a 2D scene for a perspective camera (i.e., a camera not set for orthographic view).

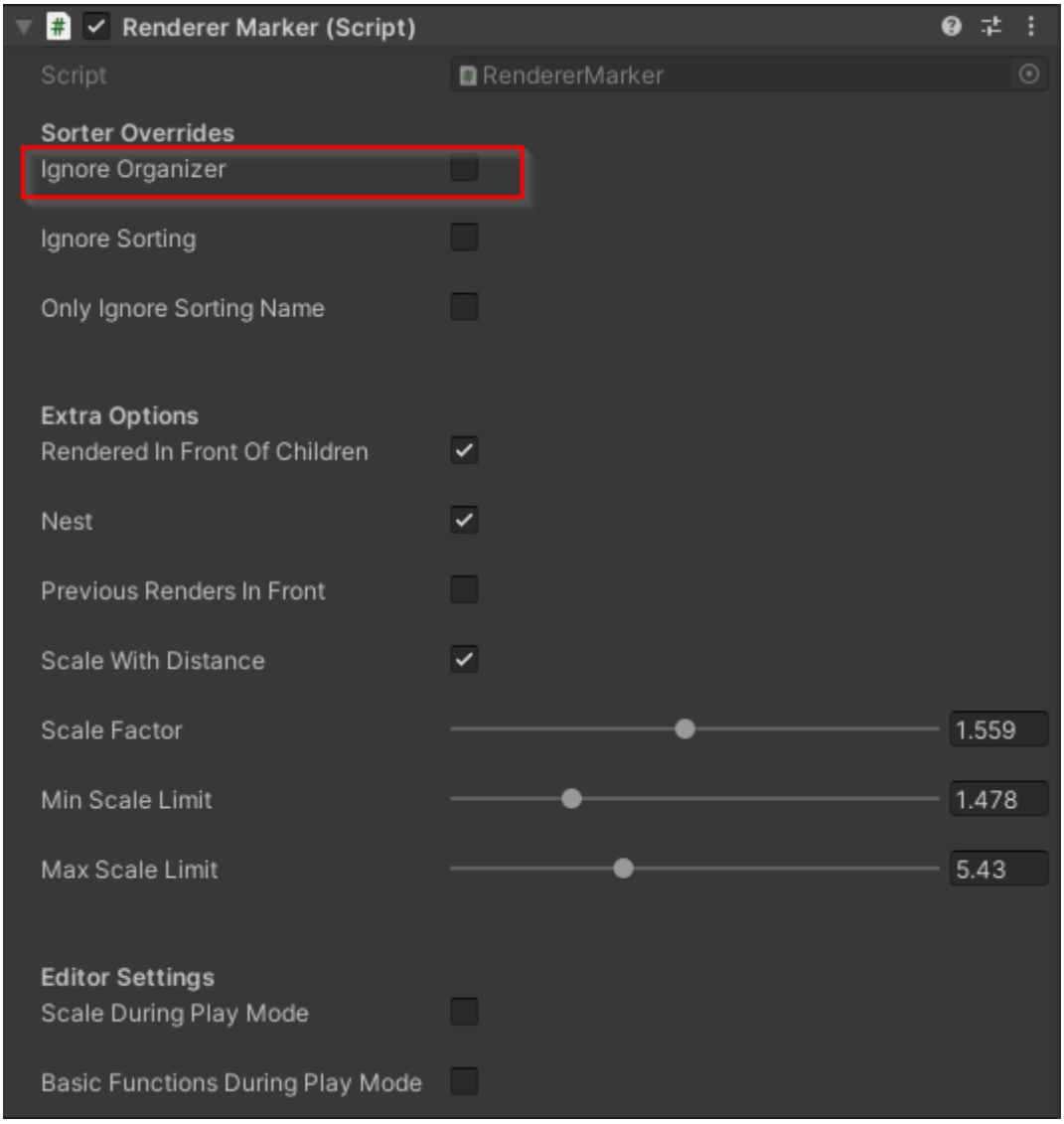
However, in order to achieve a parallax effect for this type of camera view, you will need to constantly position renderers on the z axis yourself, and that is definitely not great for large scenes with multiple assets.

So, with this system, you can immediately sort the images on the z axis depending on a max distance factor and whether or not the renderers are a foreground element.

1. **Add a layer organizer to a parent object that contains all the renderers you want to sort.**
2. **Then, enable the check mark for Sort Z Axis Positions:**

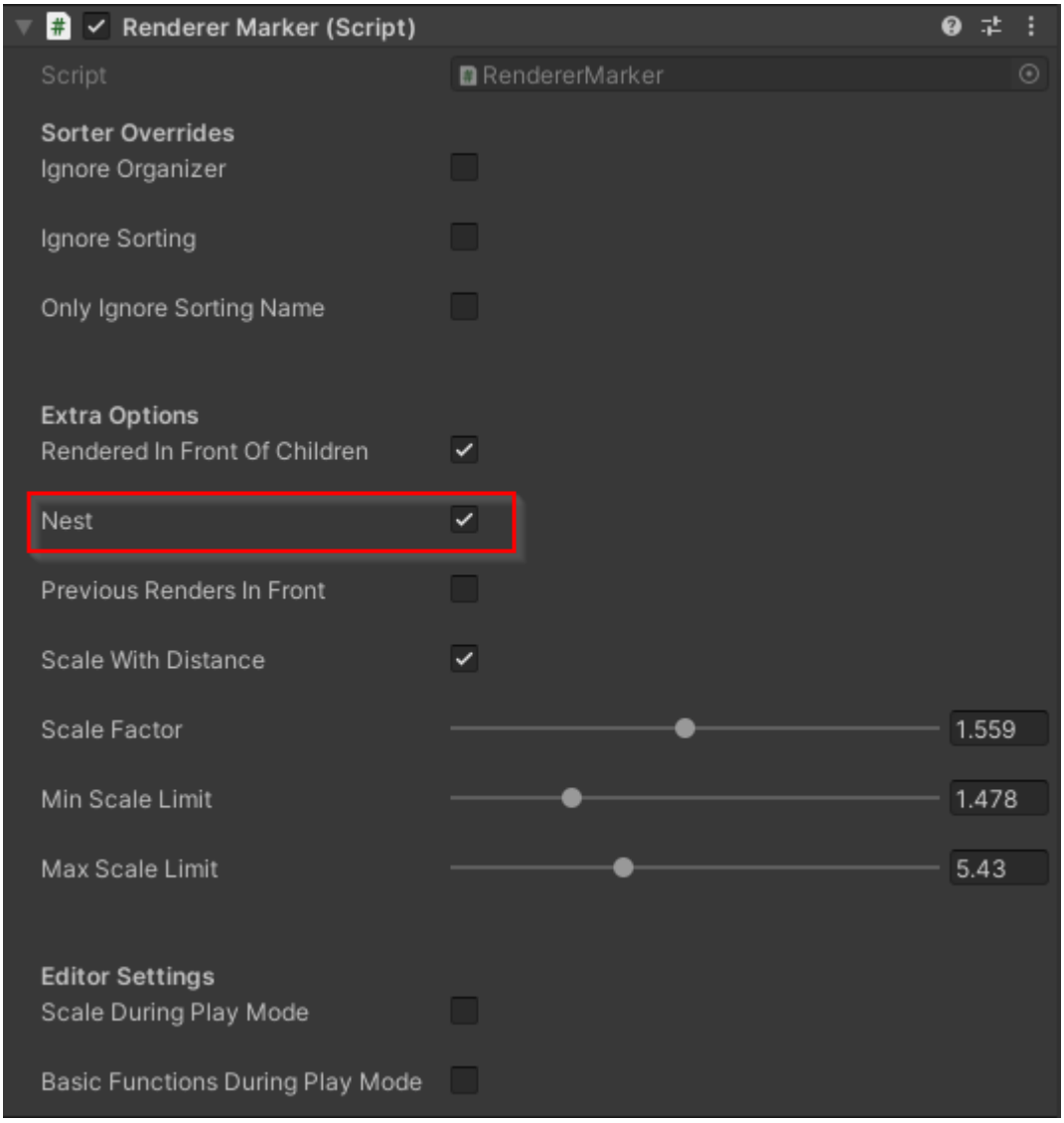


3. **Then, since not all children should be moved in the z axis (e.g., a window which is parented to a building), you should Uncheck Ignore Organizer from the renderer markers on the objects you want to move in the z axis:**



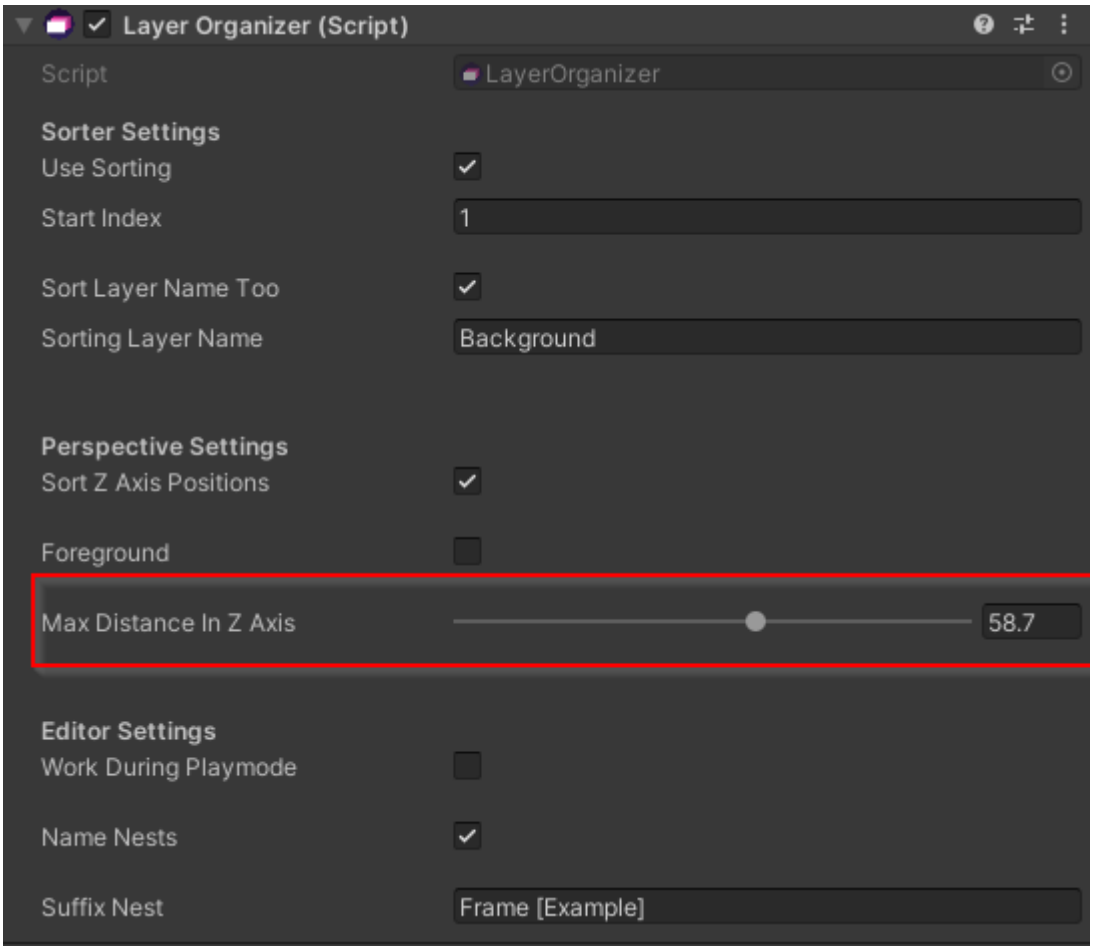
Those objects would be things like big landscapes, or buildings, or things that contain other children (i.e. a nest of renderers). Check the example scene for a better understanding of what this means.

4. **It is good practice to check the Nest option on the object that will be sorted in z axis and has children:**



5. **By now, you'll notice that the objects are already sorted in the z axis, however, the default max**

distance is 100, this means that as soon as the sorter is working, all your objects may disappear far away. Simply go to the LayerOrganizer script that is on the parent object of the children, and control the distance from there:



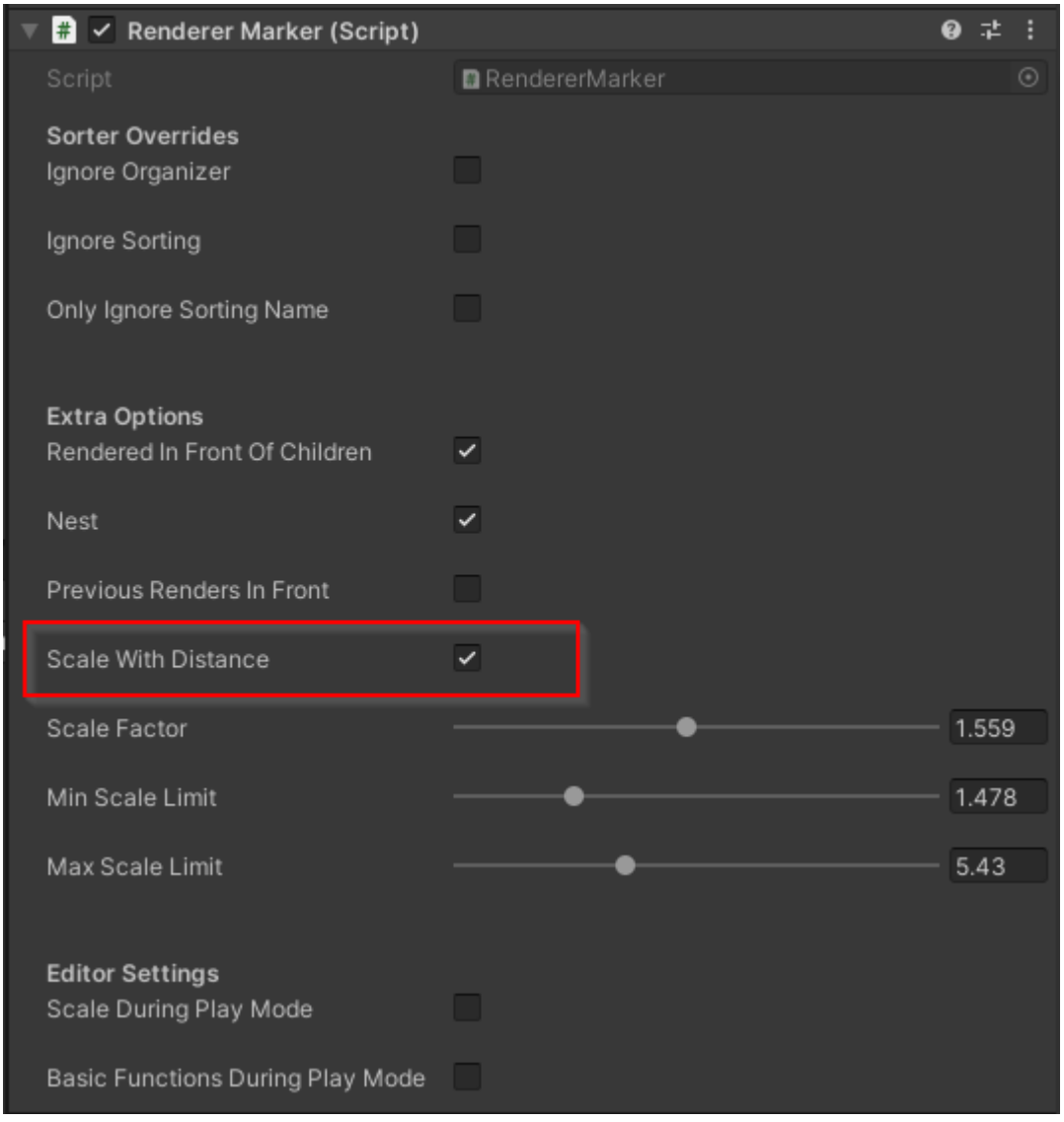
You can also check Foreground if this will be sorted in the opposite side of the camera.

How to scale renderers with distance (Hide Edges):

This issue will face you if you have sprites or renderers with defines edges on the screen. This will be the same with sprites that have edges on the upper side too, since we are only moving the sprites away and not scaling them to complement the distance.

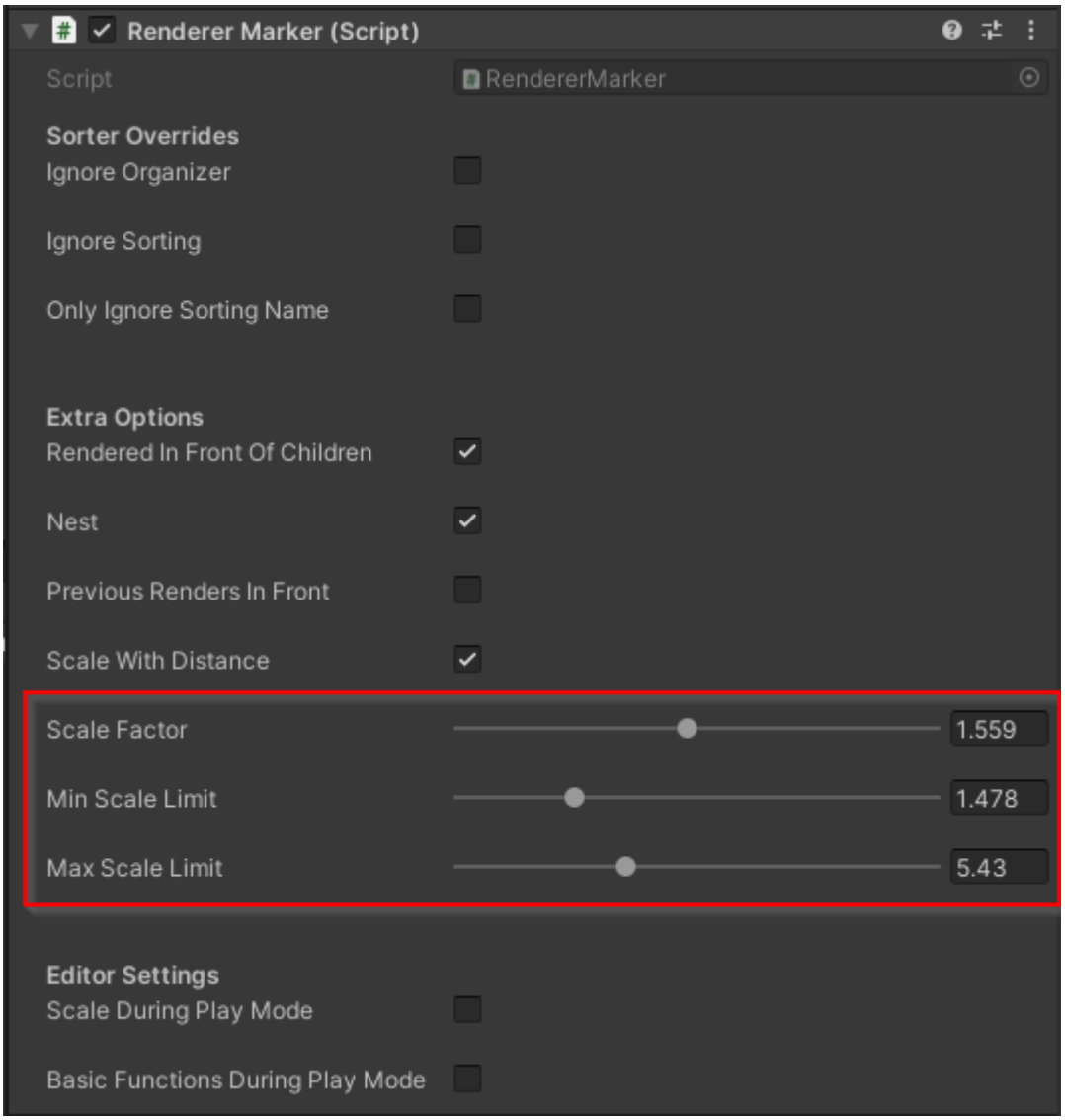
In order to fix this, you need to use the scale feature of the renderer marker script.

1. **Go to the render marker that is handling the image which is showing its edge, and enable Scale with Distance:**



This will allow the object to scale depending on how far it is in the scene, NOT depending on the distance of the camera.

Then, the object should scale normally using the three below options:



These options just control how much scaling does the object go through. The factor is how much scale we are adding, and the minimum/maximum limits simply prevent it from getting any bigger or smaller respectively.

You can use the “Scale During Play Mode” option to set up the scale while in play mode. Keep in mind that this script does not save play mode edits, so you should use something like (Play Mode Save by Plugin Master) which is a really powerful tool that I use to retain play mode edits.

What are the extra options?

- **(Name Nests) in Layer Organizer:**

  This tools simply names all the objects that have been marked as a nest. The naming adds a prefix to the object's name.

- **(Rendered In Front of Children) in Renderer Marker:**

  This makes the object's sorting order change so that it becomes in front of its children. This maybe used if you want the children of the object to renderer behind it, since sorting is handled by Hierarchy order basis.

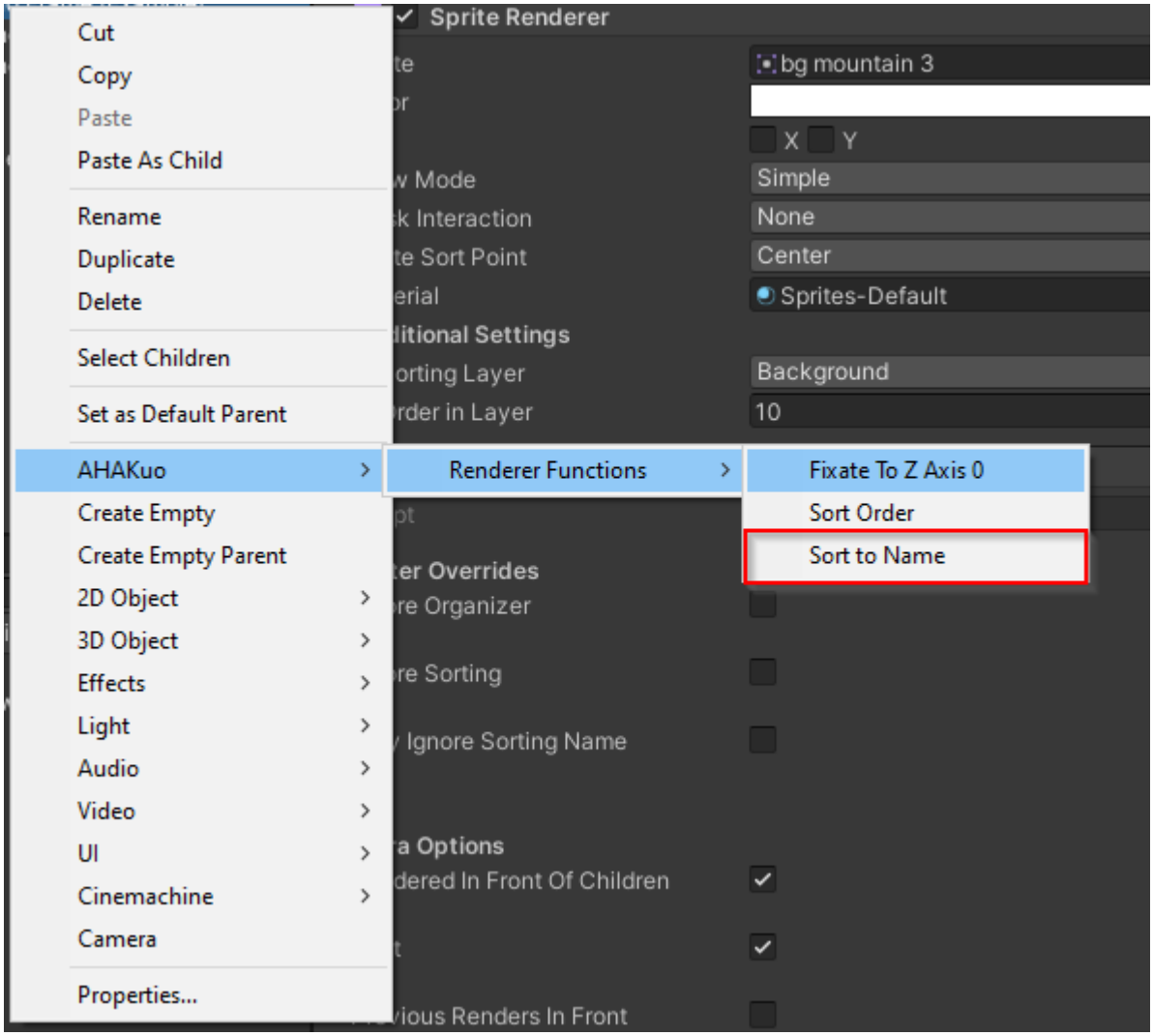
- **(Previous Renders in Front) in Renderer Marker:**

This option should be checked if the object has another renderer before it in the hierarchy that renders in front of its children. You don’t need to check both Previous Renders in Front and Rendered in Front of Children. If Rendered in Front of Children is checked, do not check Previous Renders in Front even if there is such an object before it.

This will make the object correct its order.

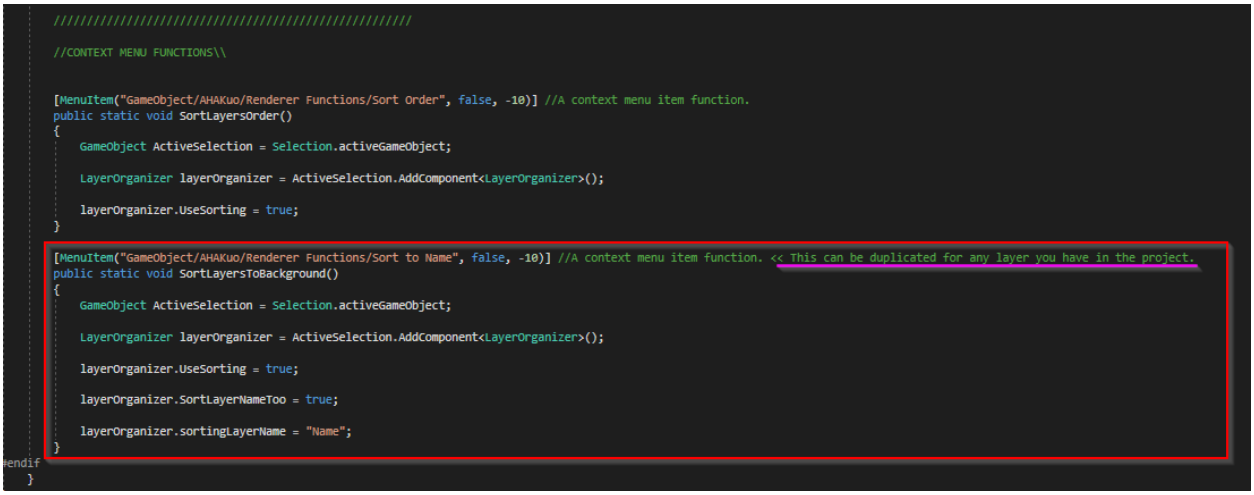
How to add your own context option?

A good way to speed up your workflow so you don’t have to keep inputting the name of the layer you want to sort to is to add it within the context menu as such:



That option, as explained earlier, will make it so that the layer organizer you add will be pre-set to sort to a layer name of choice. Here in this example option, it will sort to the layer name “Name”, but obviously that may not exist in your project. So, a good way to do this is to add a new context option or change this one to make it set to a name you want.

1. **Open the LayerOrganizer.cs script, and navigate below to this:**



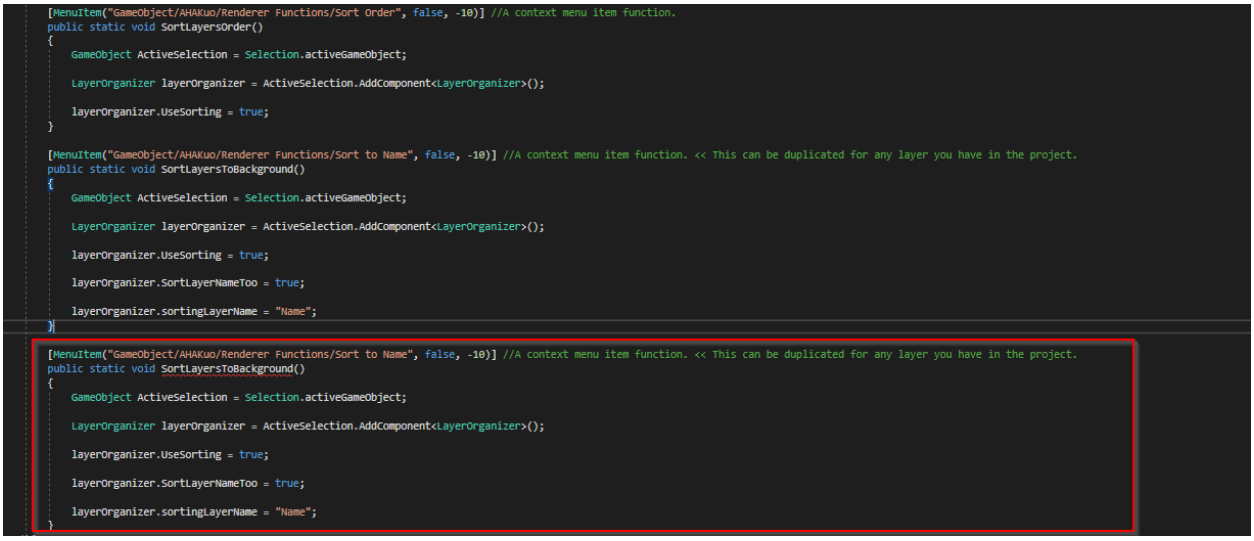
```
    ////////////////////////////////////////////////////

    //CONTEXT MENU FUNCTIONS\\


    [MenuItem("GameObject/AHAKuo/Renderer Functions/Sort Order", false, -10)] //A context menu item function.
    public static void SortLayersOrder()
    {
        GameObject ActiveSelection = Selection.activeGameObject;

        LayerOrganizer layerOrganizer = ActiveSelection.AddComponent<LayerOrganizer>();

        layerOrganizer.UseSorting = true;
    }

    [MenuItem("GameObject/AHAKuo/Renderer Functions/Sort to Name", false, -10)] //A context menu item function. << This can be duplicated for any layer you have in the project.
    public static void SortLayersToBackground()
    {
        GameObject ActiveSelection = Selection.activeGameObject;

        LayerOrganizer layerOrganizer = ActiveSelection.AddComponent<LayerOrganizer>();

        layerOrganizer.UseSorting = true;

        layerOrganizer.SortLayerNameToo = true;

        layerOrganizer.sortingLayerName = "Name";
    }
#endif
}
```

2. That block of code can be copied and pasted.

Copy this section, and paste it below:

```
    ////////////////////////////////////////////////

    //CONTEXT MENU FUNCTIONS\\


    [MenuItem("GameObject/AHAKuo/Renderer Functions/Sort Order", false, -10)] //A context menu item function.
    public static void SortLayersOrder()
    {
        GameObject ActiveSelection = Selection.activeGameObject;

        LayerOrganizer layerOrganizer = ActiveSelection.AddComponent<LayerOrganizer>();

        layerOrganizer.UseSorting = true;
    }

    [MenuItem("GameObject/AHAKuo/Renderer Functions/Sort to Name", false, -10)] //A context menu item function. << This can be duplicated for any layer you have in the project.
    public static void SortLayersToBackground()
    {
        GameObject ActiveSelection = Selection.activeGameObject;

        LayerOrganizer layerOrganizer = ActiveSelection.AddComponent<LayerOrganizer>();

        layerOrganizer.UseSorting = true;

        layerOrganizer.SortLayerNameToo = true;

        layerOrganizer.sortingLayerName = "Name";
    }
#endif
}
```

Paste it below like so:



```
    [MenuItem("GameObject/AHAKuo/Renderer Functions/Sort Order", false, -10)] //A context menu item function.
    public static void SortLayersOrder()
    {
        GameObject ActiveSelection = Selection.activeGameObject;

        LayerOrganizer layerOrganizer = ActiveSelection.AddComponent<LayerOrganizer>();

        layerOrganizer.UseSorting = true;
    }

    [MenuItem("GameObject/AHAKuo/Renderer Functions/Sort to Name", false, -10)] //A context menu item function. << This can be duplicated for any layer you have in the project.
    public static void SortLayersToBackground()
    {
        GameObject ActiveSelection = Selection.activeGameObject;

        LayerOrganizer layerOrganizer = ActiveSelection.AddComponent<LayerOrganizer>();

        layerOrganizer.UseSorting = true;

        layerOrganizer.SortLayerNameToo = true;

        layerOrganizer.sortingLayerName = "Name";
    }

    [MenuItem("GameObject/AHAKuo/Renderer Functions/Sort to Name", false, -10)] //A context menu item function. << This can be duplicated for any layer you have in the project.
    public static void SortLayersToBackground()
    {
        GameObject ActiveSelection = Selection.activeGameObject;

        LayerOrganizer layerOrganizer = ActiveSelection.AddComponent<LayerOrganizer>();

        layerOrganizer.UseSorting = true;

        layerOrganizer.SortLayerNameToo = true;

        layerOrganizer.sortingLayerName = "Name";
    }
```

3. **Then, rename these 3 fields to your liking, and add the layer name exactly as it is in your unity project into the field marked with the pink line:**

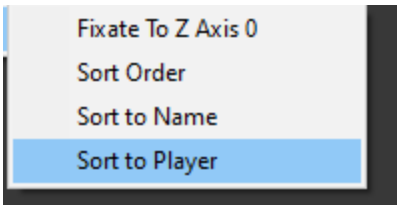
```
[MenuItem("GameObject/AHAKuo/Renderer Functions/Sort to Name", false, -10)] //A cont
public static void SortLayersToBackground()
{
    GameObject ActiveSelection = Selection.activeGameObject;

    LayerOrganizer layerOrganizer = ActiveSelection.AddComponent<LayerOrganizer>();

    layerOrganizer.UseSorting = true;

    layerOrganizer.SortLayerNameToo = true;

    layerOrganizer.sortingLayerName = "Name";
}
```

Then, it will appear in the context menu.

Here is an example I made from the same thing we copied:

```
[MenuItem("GameObject/AHAKuo/Renderer Functions/Sort to Player", false, -10)] //A cont
public static void SortLayersToPlayer()
{
    GameObject ActiveSelection = Selection.activeGameObject;

    LayerOrganizer layerOrganizer = ActiveSelection.AddComponent<LayerOrganizer>();

    layerOrganizer.UseSorting = true;

    layerOrganizer.SortLayerNameToo = true;

    layerOrganizer.sortingLayerName = "Player";
}
```

And here it is in the context menu:



If you have any questions, direct them through here and you'll get a response very soon.

https://www.ahakuo.com/contact-page