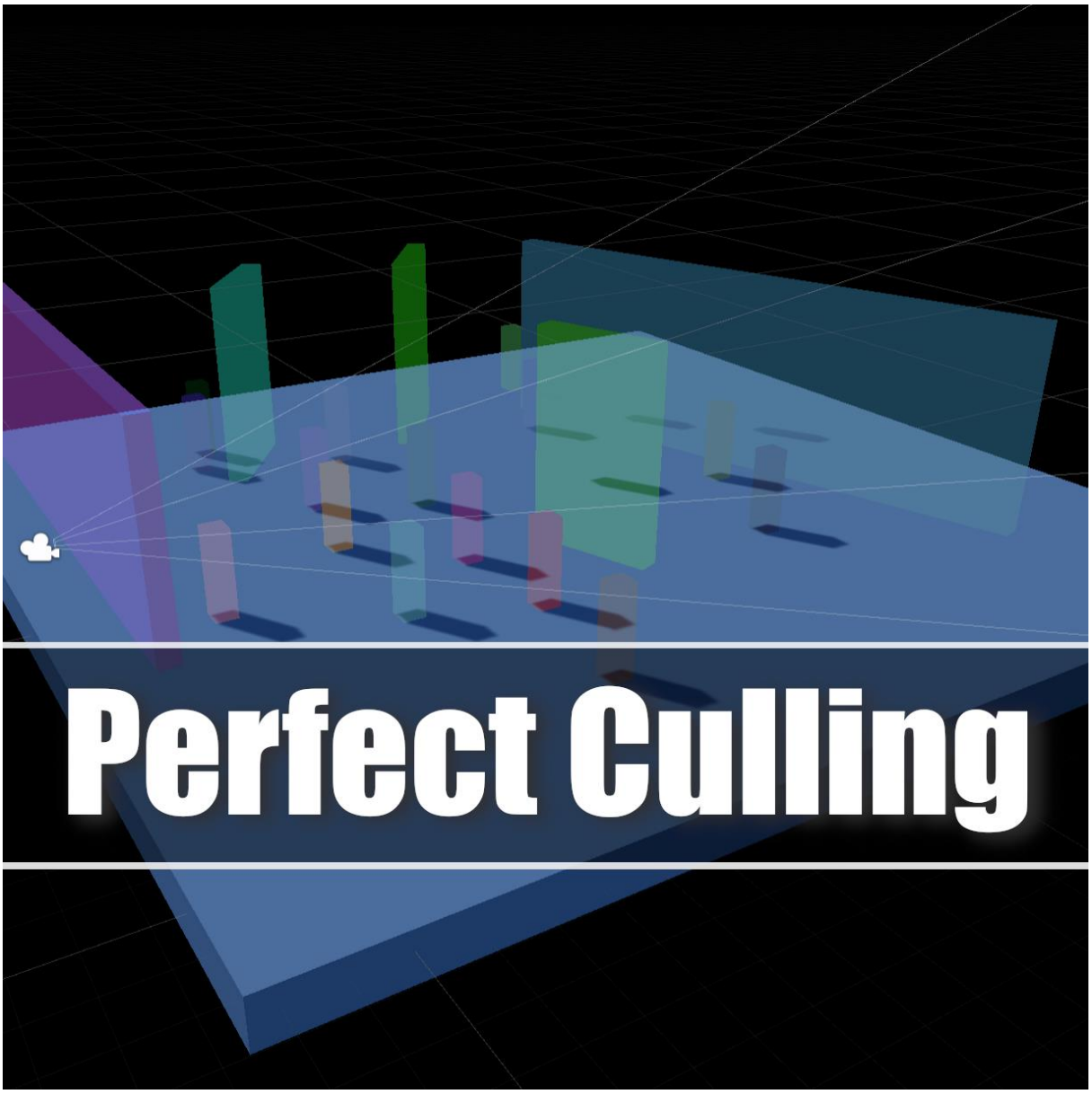
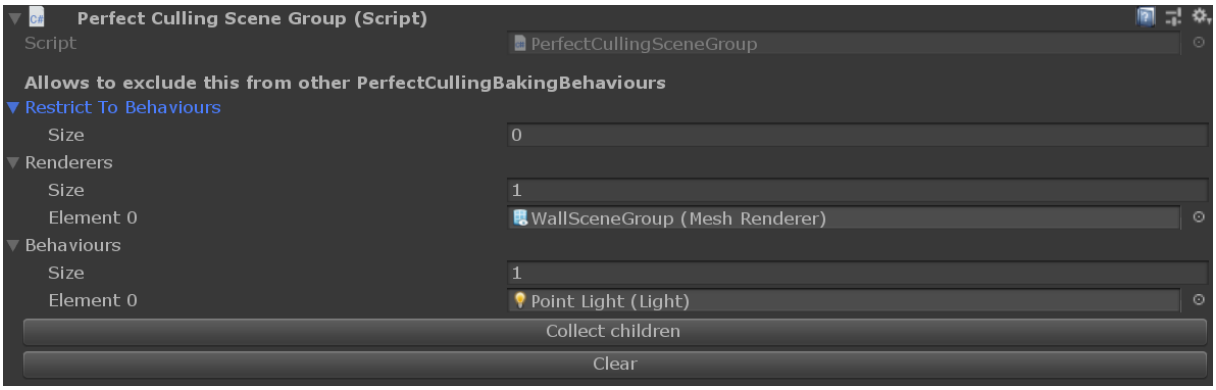


Light and script culling

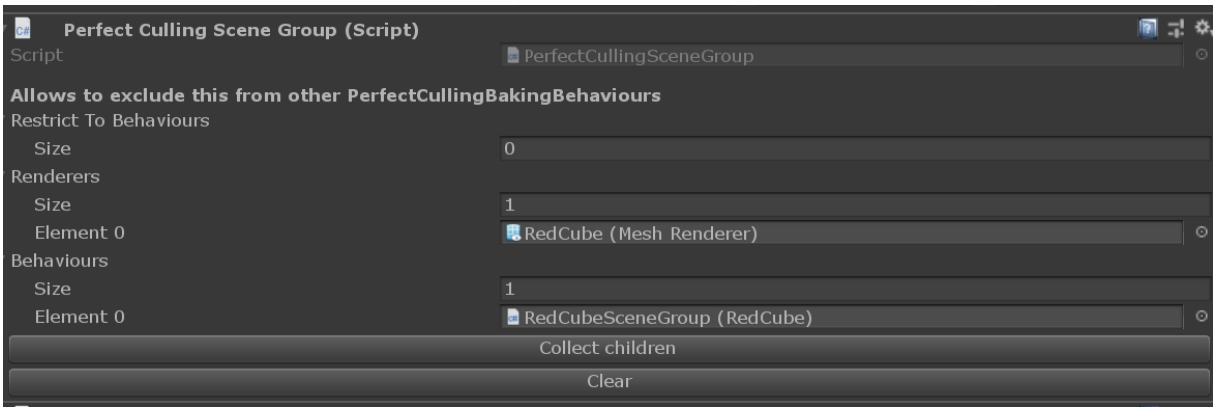
Perfect Culling also allows you to cull any UnityBehaviour. This includes pretty much everything the Unity engine has to offer such as lights, projectors, scripts, etc.

In order to make use of this you will need to add a **PerfectCullingSceneGroup** component. You also need a renderer that will be used as a reference to toggle the state of the assigned behaviours.

For instance, for a projector, it would make sense to add the renderer that is being projected on. The behaviour would of course be the projector. This way every time the renderer that is being projected on becomes visible/occluded the projector will become enabled/disabled as well.



Example setup that will tie the state of "Point Light" to the renderer "WallSceneGroup"



RedCube script is enabled/disabled with RedCube mesh

You can also use this concept to get a callback. Just create a script and add it to behaviours. For your script you will notice that **OnEnable** is called when the renderer is visible and **OnDisable** is called when the renderer is occluded thus allowing you to execute custom logic.

You can also see this in the following demo scene:

- Demo_UnityBehaviours

Notice that this might become a bit messy and inefficient for multi-camera setups (multiple **PerfectCullingCamera** scripts) because one camera might be able to see the object whereas the other one does not.