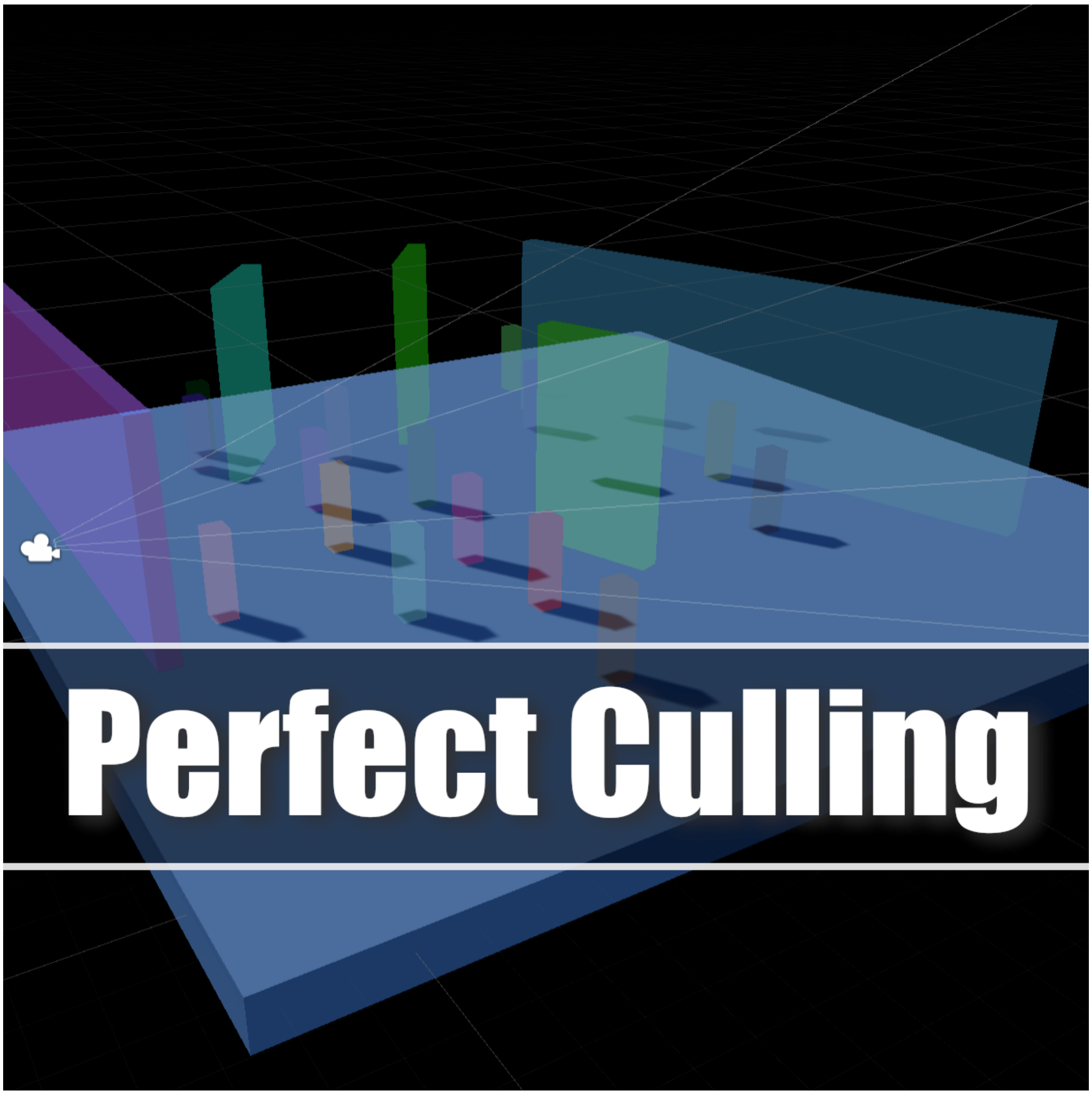
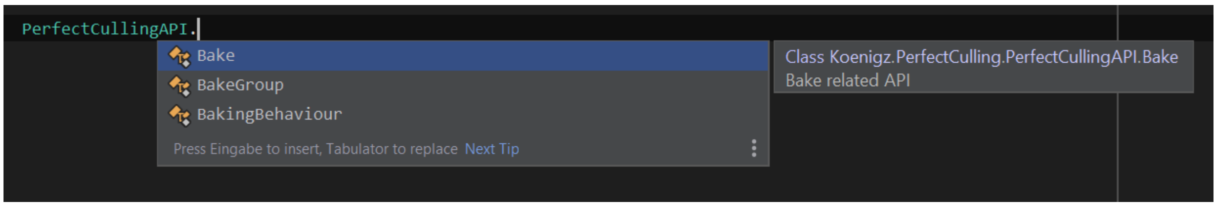


API

The Perfect Culling API can be found in the static class **Koenigz.PerfectCulling.PerfectCullingAPI** that contains additional static sub-classes to further organize the API. For instance **Bake** related API functionality can be found in **Koenigz.PerfectCulling.PerfectCullingAPI.Bake.**



```
PerfectCullingAPI.
    Bake
    BakeGroup
    BakingBehaviour
    Press Eingabe to insert, Tabulator to replace  Next Tip

Class Koenigz.PerfectCulling.PerfectCullingAPI.Bake
Bake related API
```

Bake API

You might want to make the occlusion calculations part of your build pipeline.

You can use **ScheduleBake** (followed by **BakeAllScheduled**) or use **BakeNow** to bake immediately.

Here is an example how to bake all volumes in a scene:

```
PerfectCullingBakingBehaviour[] bakingBehaviours =
PerfectCullingAPI.Bake.FindAllBakingBehaviours();

PerfectCullingAPI.Bake.BakeNow(bakingBehaviours);
```

You can also schedule bakes first and then bake all scheduled bakes:

```
PerfectCullingBakingBehaviour[] bakingBehaviours =
PerfectCullingAPI.Bake.FindAllBakingBehaviours();

foreach (PerfectCullingBakingBehaviour behaviour in bakingBehaviours)
{
    PerfectCullingAPI.Bake.ScheduleBake(new
     BakeInformation()
     {
         BakingBehaviour = behaviour,
         AdditionalOccluders = null
     }
    );
}

PerfectCullingAPI.Bake.BakeAllScheduled();
```

The **AdditionalOccluders** argument allows you to pass in Renderers that are not culled by the asset but can occlude renderers during the baking process. This can be useful if you are using multiple scenes but you want to make sure culling also works when you cross over into a different scene.

In order for this to work you absolutely need to make sure that you increase the size of your volume so it overlaps the content of the other scenes. Ideally your scene is setup in such a way that good culling opportunity is given and only small overlaps are required.

Since the baking process happens asynchronously you can make use of the following events to get information when the bake finished:

```
// Called when a single bake finished
PerfectCullingAPI.Bake.OnBakeFinished

// Called when all bakes finished
PerfectCullingAPI.Bake.OnAllBakesFinished
```

<u>Bake Group API</u>

You might want to dynamically add and remove renderers from a Bake Group at run-time to manage visibility on your own for a period of time.

You can find the API that allows you to do that in **PerfectCullingAPI.BakeGroup.**

<u>Baking Behaviour API</u>

The API can be found in **PerfectCullingAPI.BakingBehaviour**.