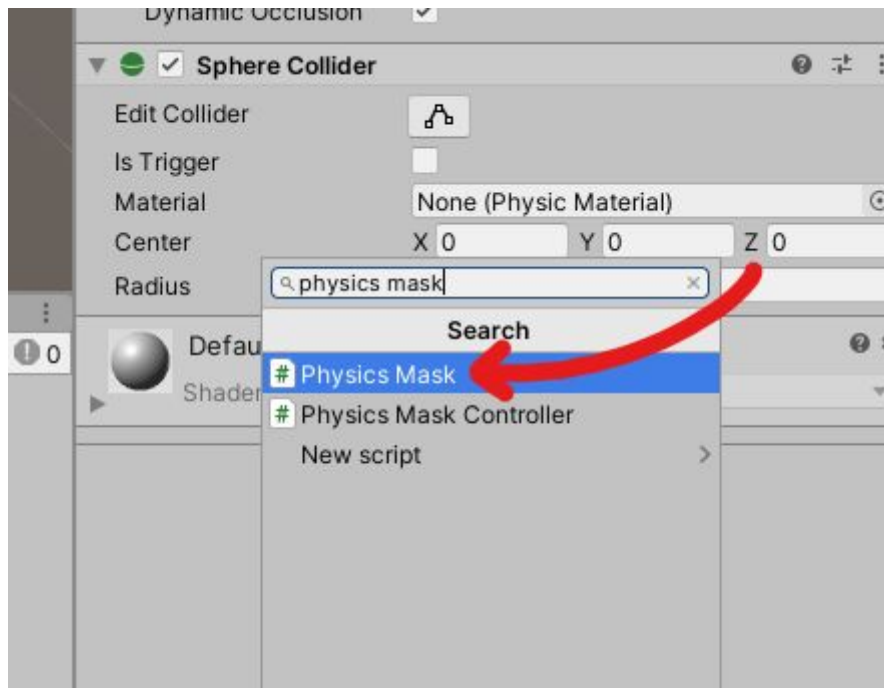# Better Physics Layers 1.0.0
Documentation

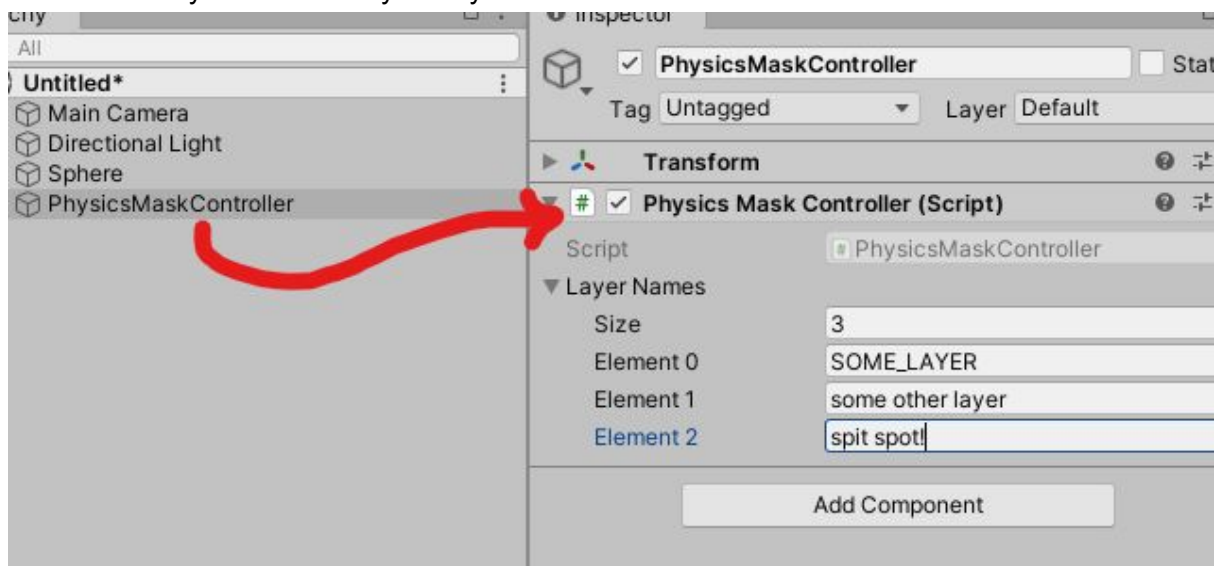
Thanks for supporting me with your purchase! If you have issues, don't hesitate to shoot questions my way: forum thread, discord or email 🙂

Quick-start guide:
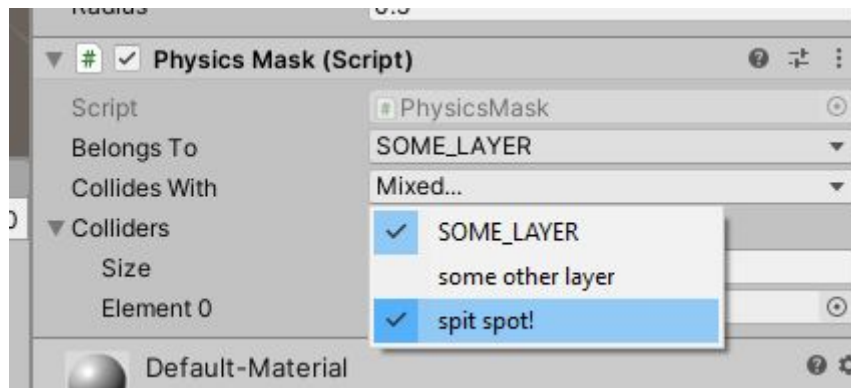
1) Add the **Physics Mask** component to your rigid body.

2) A **Physics Mask Controller** object will automatically be created in the scene. This is where you can name your layers.

3) You can now decide which layers your object should belong to and collide with! Just use the dropdowns on the **Physics Mask** component.

4) For two objects to collide, each object needs to *belong to* at least one layer that the other object *collides with*. For example, if object 1 *belongs to* **SOME_LAYER** and *collides with* **spit spot!**, and object 2 *belongs to* **spit spot!** and *collides with* **SOME_LAYER**, the objects will hit each other.

More information

Colliders

Since **Physics.IgnoreCollision()** acts on individual colliders, it's possible to have the colliders of child objects all controlled by a single **PhysicsMask** component. By default the **PhysicsMask** component will find all colliders on the same object, but you can manually adjust the list to include colliders of child objects or even other objects entirely.

PhysicsMaskController

The **PhysicsMaskController** object is optional. Most of the methods of the **PhysicsMaskController** class are static; the only feature that requires an instance of the class is named layers. If you prefer to manage layers yourself and don't want the object to be created automatically, just change the **GENERATE_MISSING_INSTANCE** constant in **Singleton.cs** to false.

When the **PhysicsMaskController** object does not exist, the **Belongs To** and **Collides With** dropdowns will show all possible layers from [layer 0] to [layer 63].

PhysicsMask.SetLayer/GetLayer

SetLayer and **GetLayer** are just convenience functions, and nothing special. All they do is to combine **SetBelongsTo/SetCollidesWith** and **GetBelongsTo/GetCollidesWith**. For many situations you may not need the added flexibility of individually controlling **belongsTo/collidesWith**, and would rather just have an enhanced version of the standard Unity layer functionality. In this case, **SetLayer** and **GetLayer** are for you.