This product contains a lot of powerful attributes plus some new types.

What is an Attribute?

An attribute is a tag in code. In this case they add tags to fields to change their appearance in Unity's Inspector.

For example, even though [HorizontalLine] simply adds a line, it is tied to a field and in fact only changes the field below to have a line above it.

How to use an Attribute?

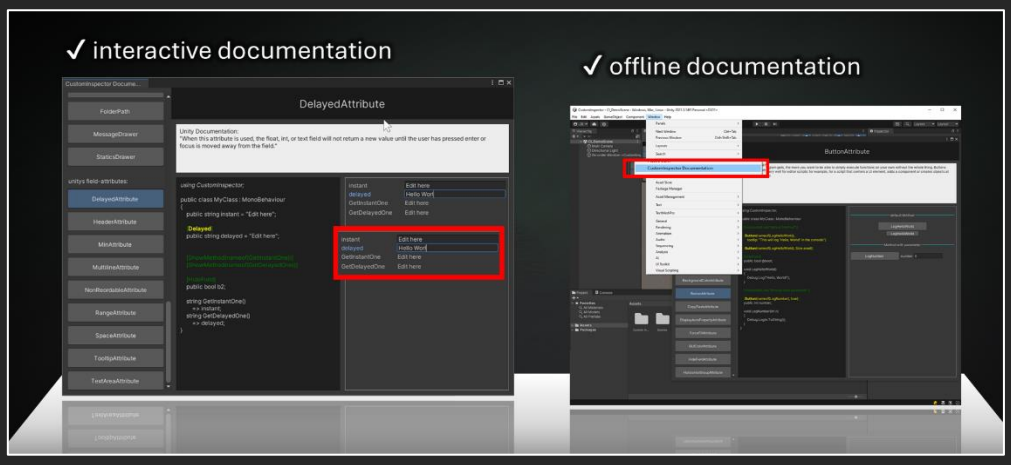
As you already know, fields are displayed in the Inspector if the types are serializable and also have public or the [SerializeField] attribute. In order to be able to use the attribute now and to be able to change the display in the inspector, you must write the attribute in square brackets in front of the desired fields, e.g. [MyAttributeName] public int myField.

How to use the serialized types?

Generally, you can use the types just like other C# types. However, it may be that you have to write the name of the type as an attribute in front of the field declaration so that it is evaluated correctly in the inspector (see editor window examples).

How do I find the right attribute?

As soon as you have imported the asset, you can view all attributes in an extra editor window. To do this, go to the toolbar at the top of your unity editor.

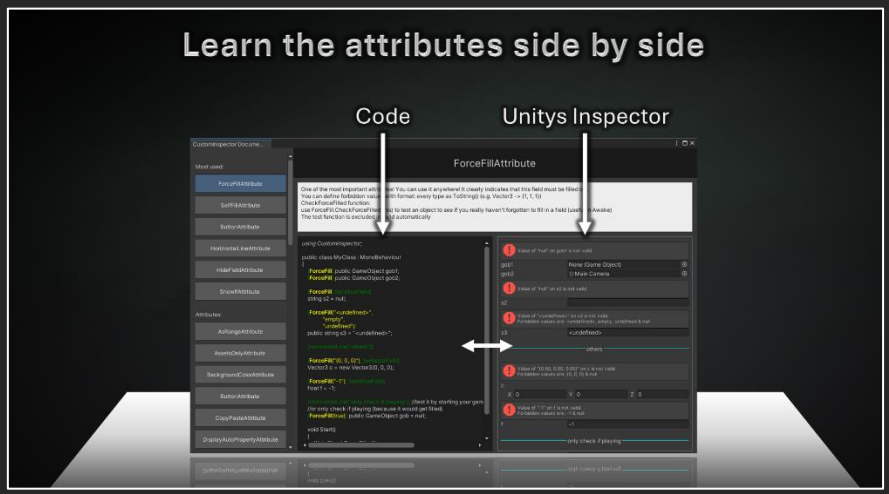


There you will see the items 'File', 'Edit', [...], 'Window' [...]

Click on 'Window' and select 'CustomInspector Documentation'.

Here you can see all the attributes that are included. You can click there around and see which attribute appeals to you on the right-hand side. You can even interact with the ui on the right.

You always see the code on the left that you write in a c# file and on the right you can see what this c# code would look like in the unity inspector.



Compatibility between different attributes

The new attributes are mostly compatible with each other with only a few exceptions.

For example, you can write

```
[Min(0), Max(10)] public int myInt
```

Or

```
[SerializeField, RequireType(typeof(IMyInterface)), ForceFill,
Indent(1), ShowIf(nameof(someBool))] GameObject myGob.
```

A problem is when you write attributes before the new types. Then just add the type's name in square brackets at the front so the inspector doesn't forget that this is a special type (see editor window examples).

Something unexpected happened?

Write me an email or leave me a comment on my website. I look forward to suggestions for improvement. If you have ideas about a missing attribute or needed serialized type then contact me and it might already be in the next update :)