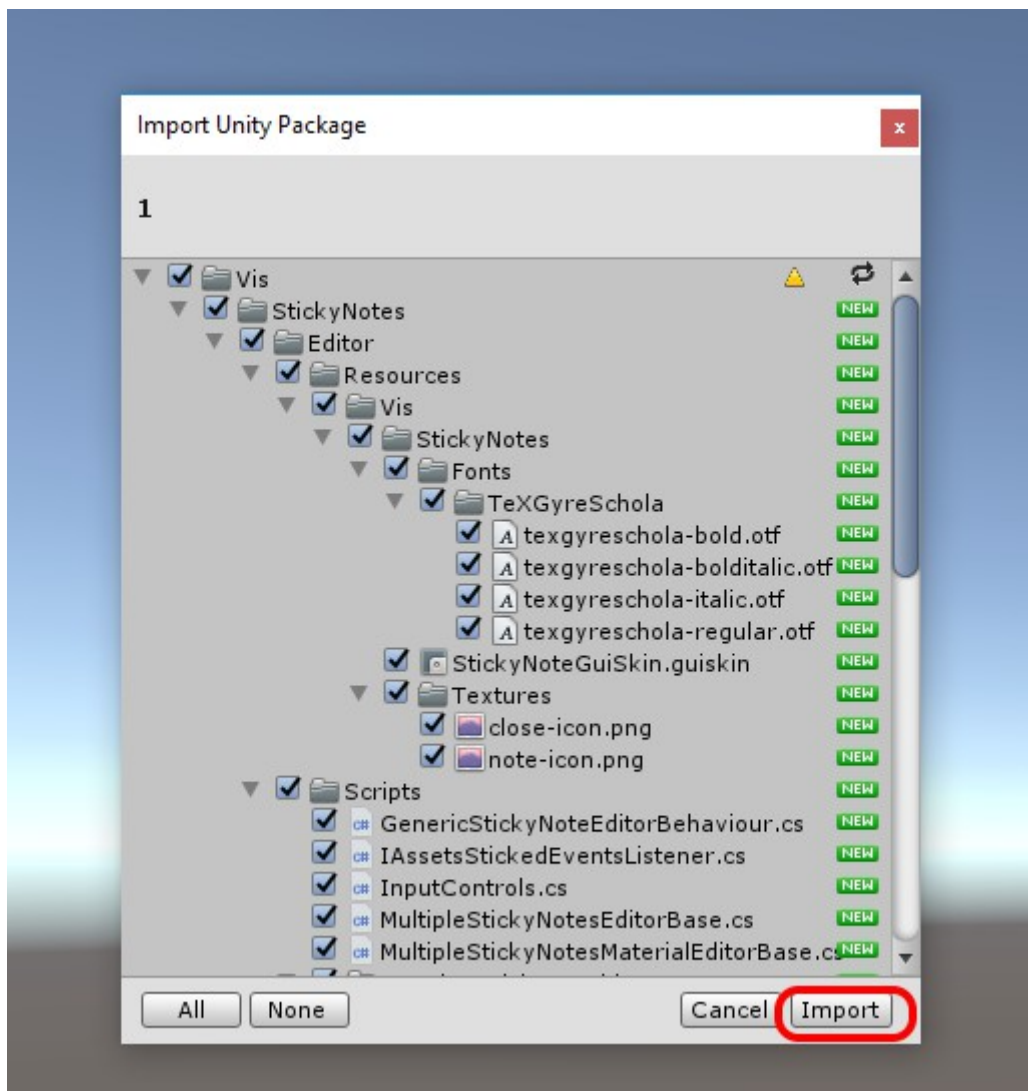


# StickyNotes user manual

In this document you will find the instructions on how to set up and use StickyNotes.

## How to set up

Step 1.



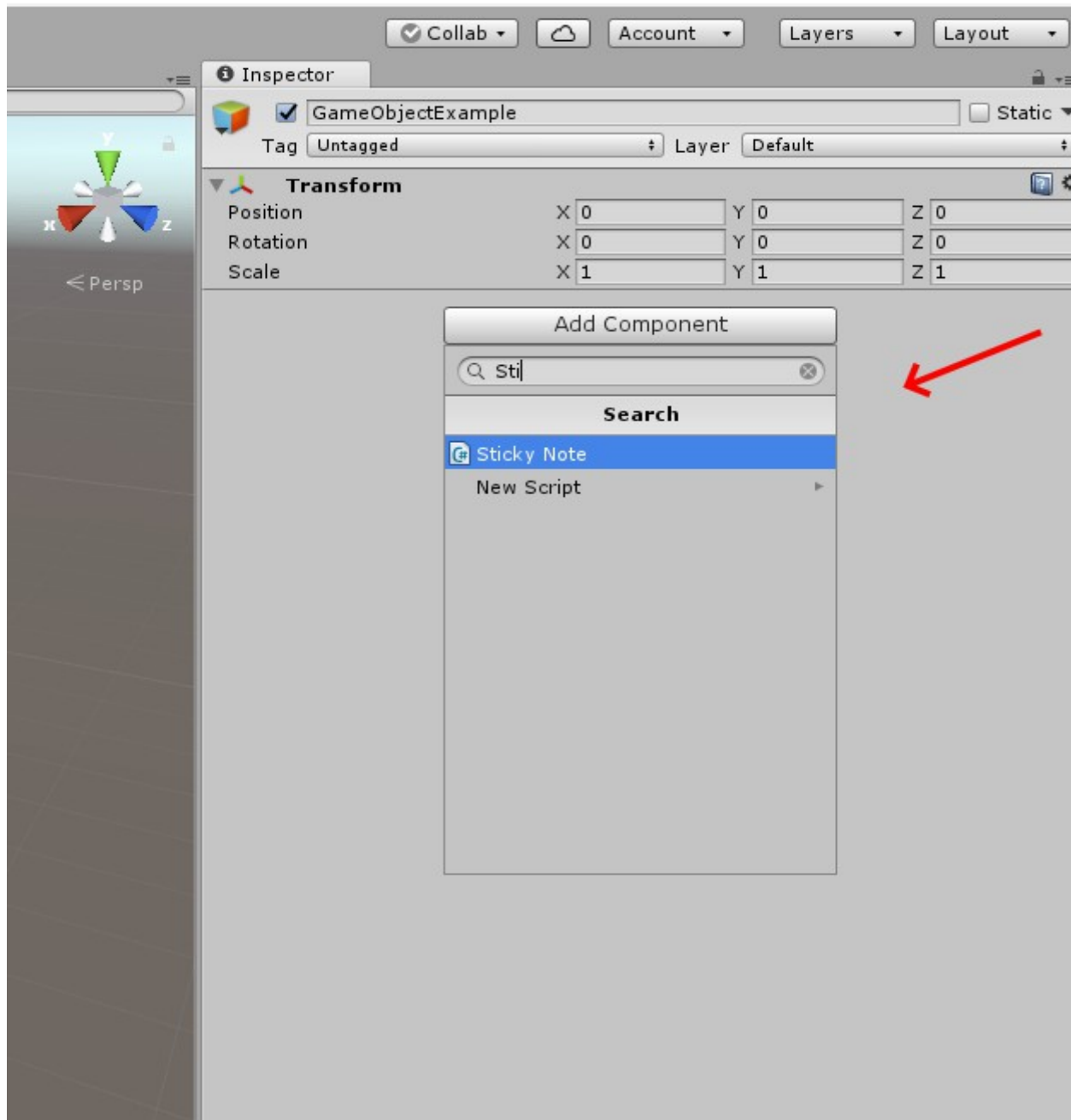
Import package from asset store.

Step 2.

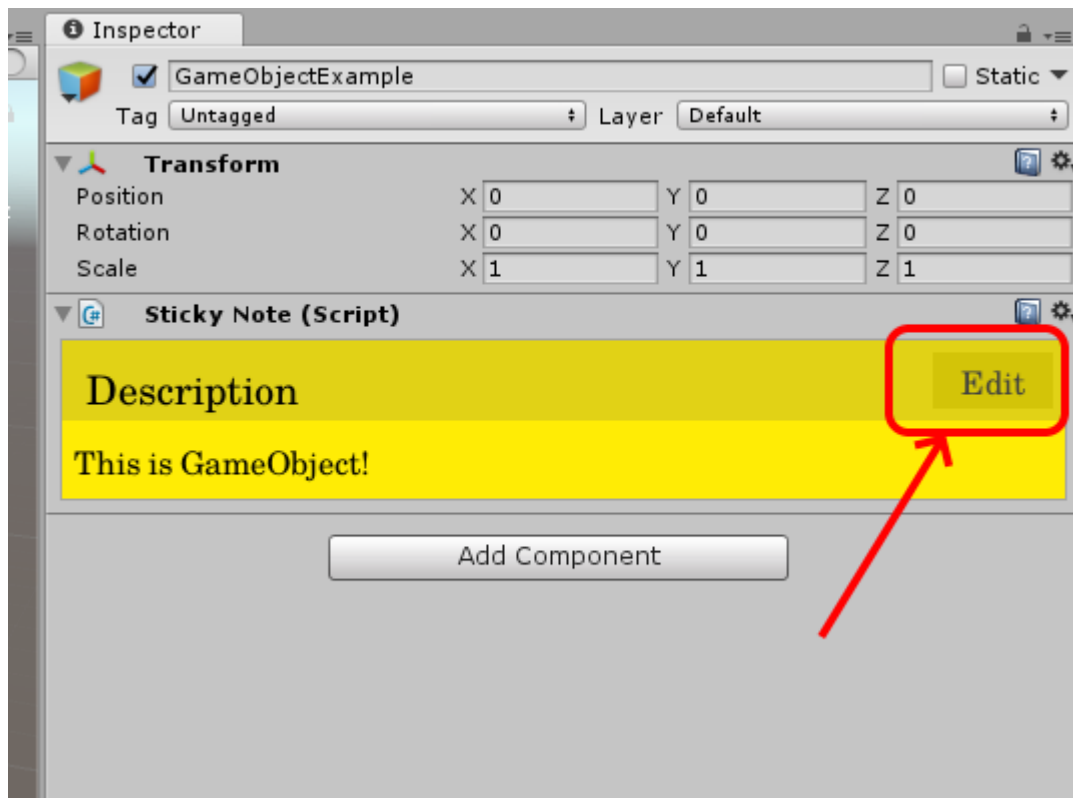
That's all! Now you can manage your sticky notes in project!

# How to use Components

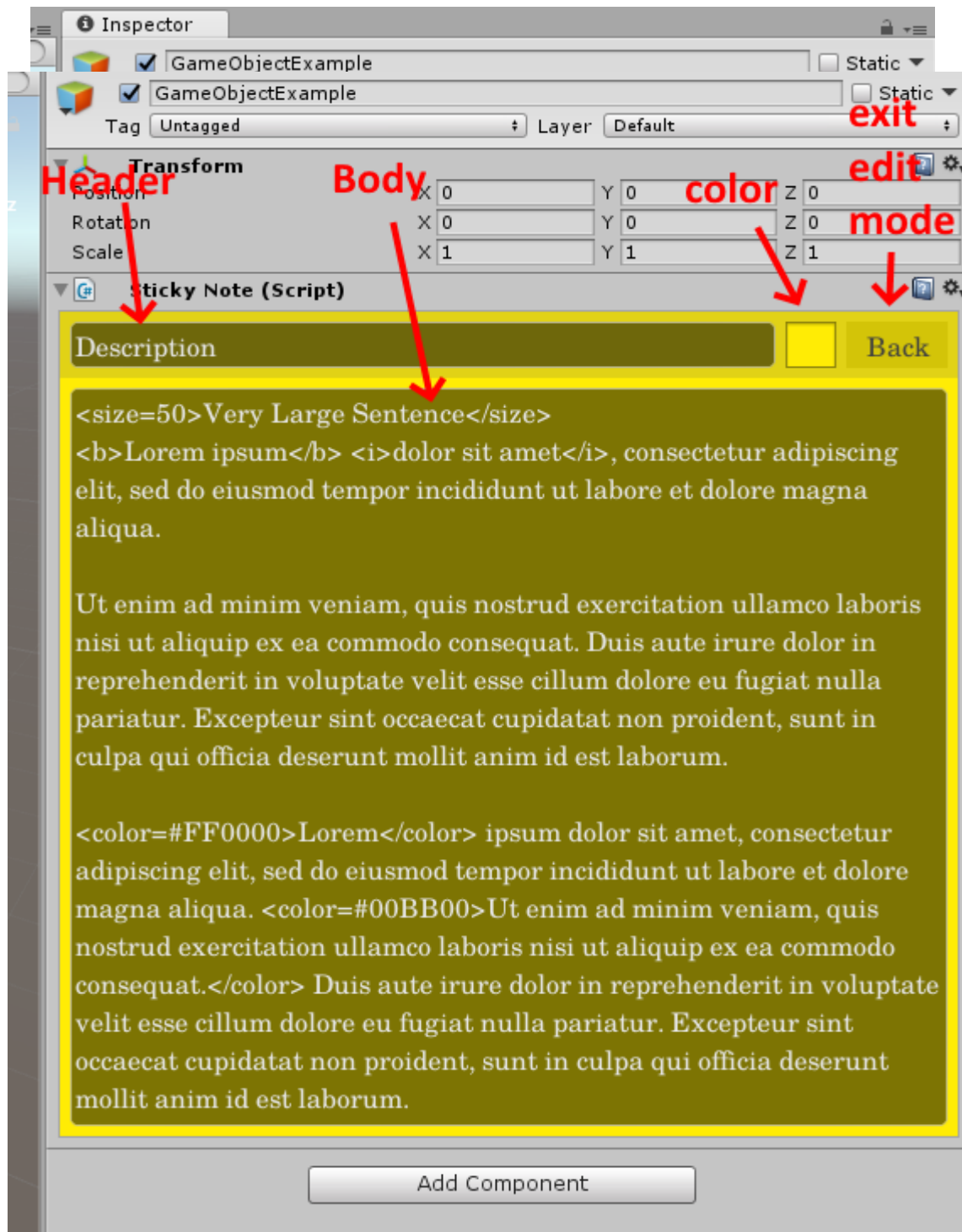
Just add *StickyNote* component to a GameObject standard way – by clicking *Add Component* button in a GameObject inspector and finding *StickyNote* component.



Now you can edit sticky note by clicking *Edit* button



Edit mode allows you to set Color of a note and any text for Header and Body



You can exit *edit mode* by clicking *Back* button.

You can use Rich Text tags when editing texts. They are:

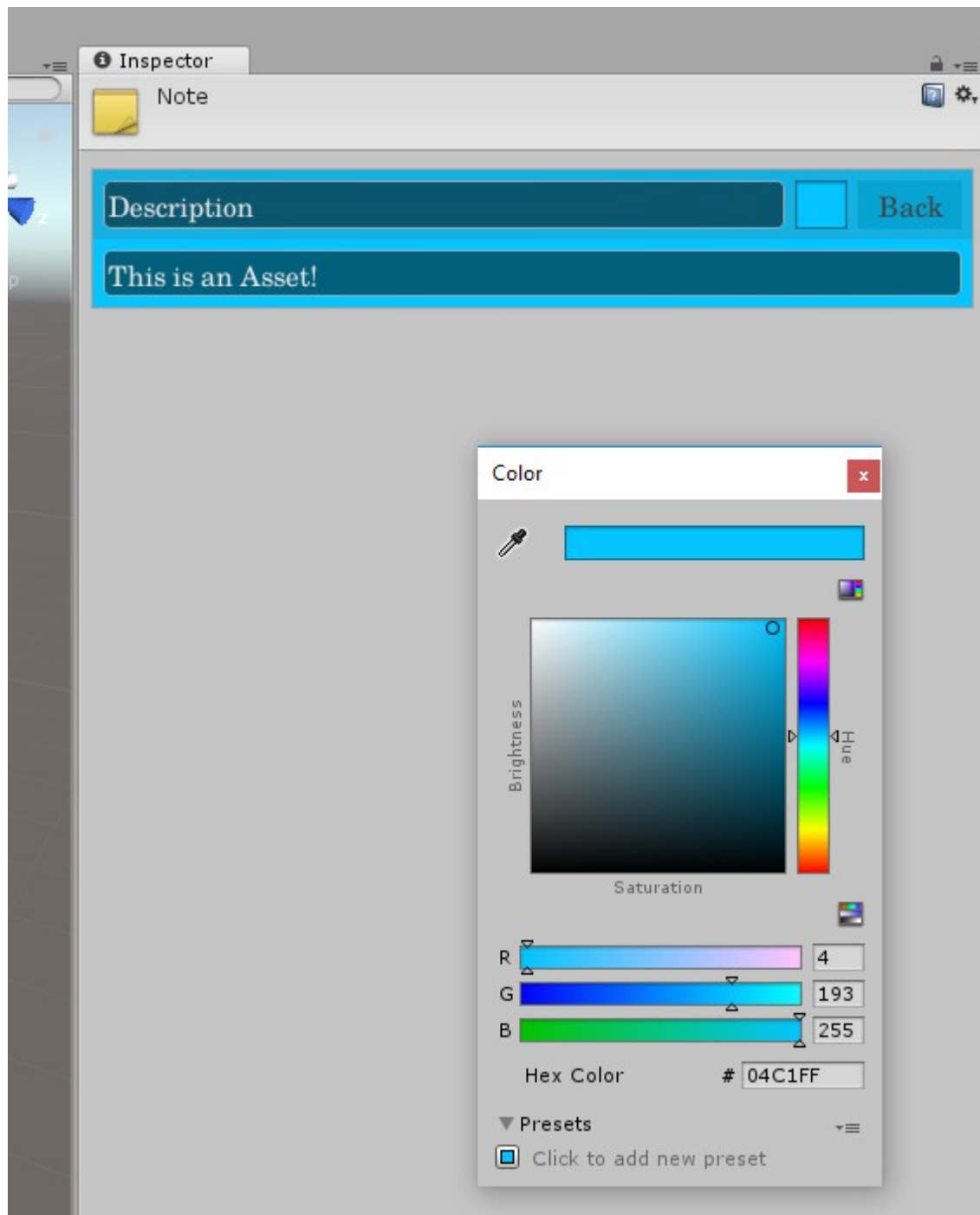
**<b>** - bold

**<i>** - italic

**<color>** - color (use hex representation for that like #00FF00)

**<size>** - set size of font

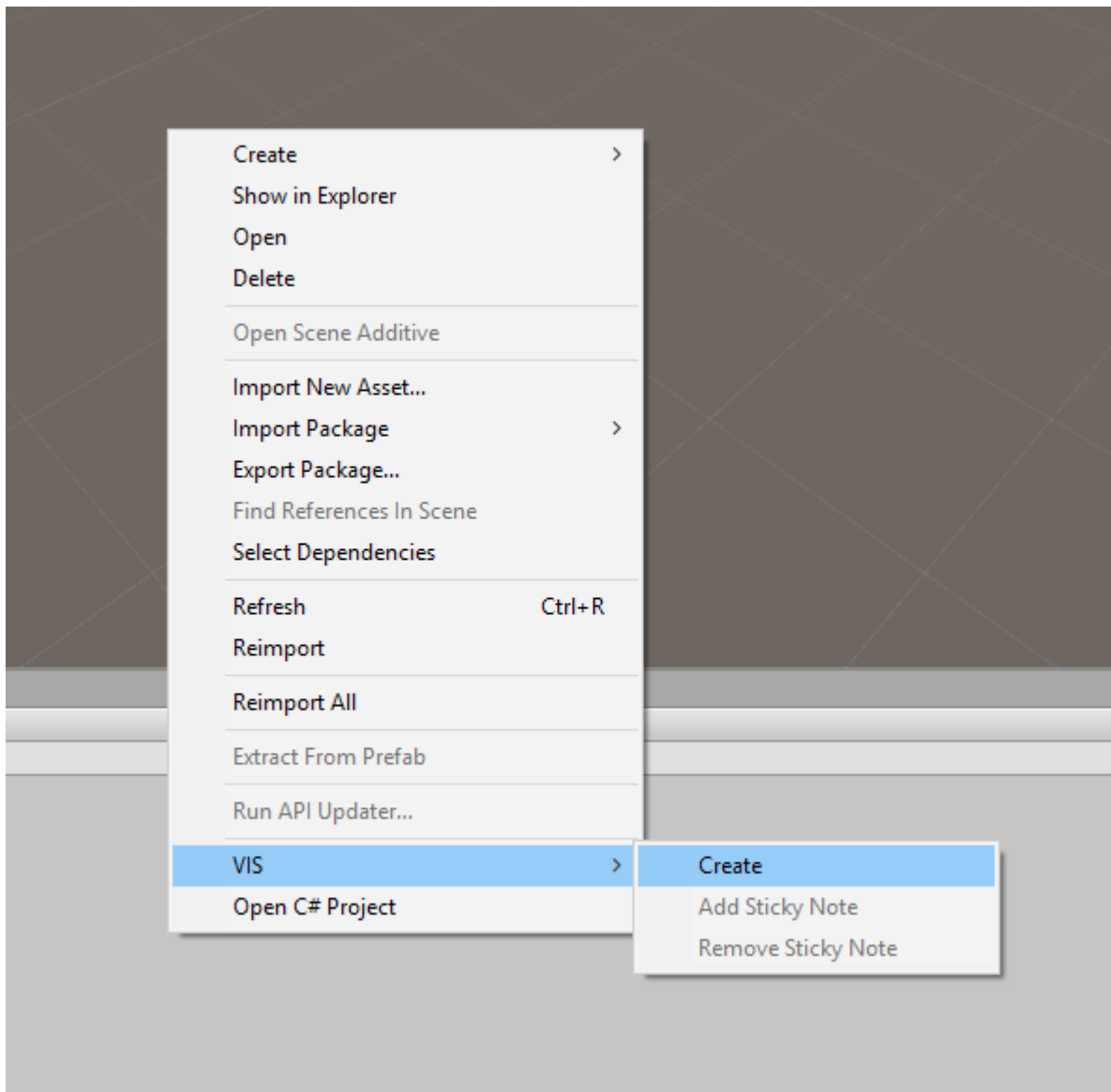
When you click *Color* button – standard Unity color picker window pops up and allow you to customize color of note.



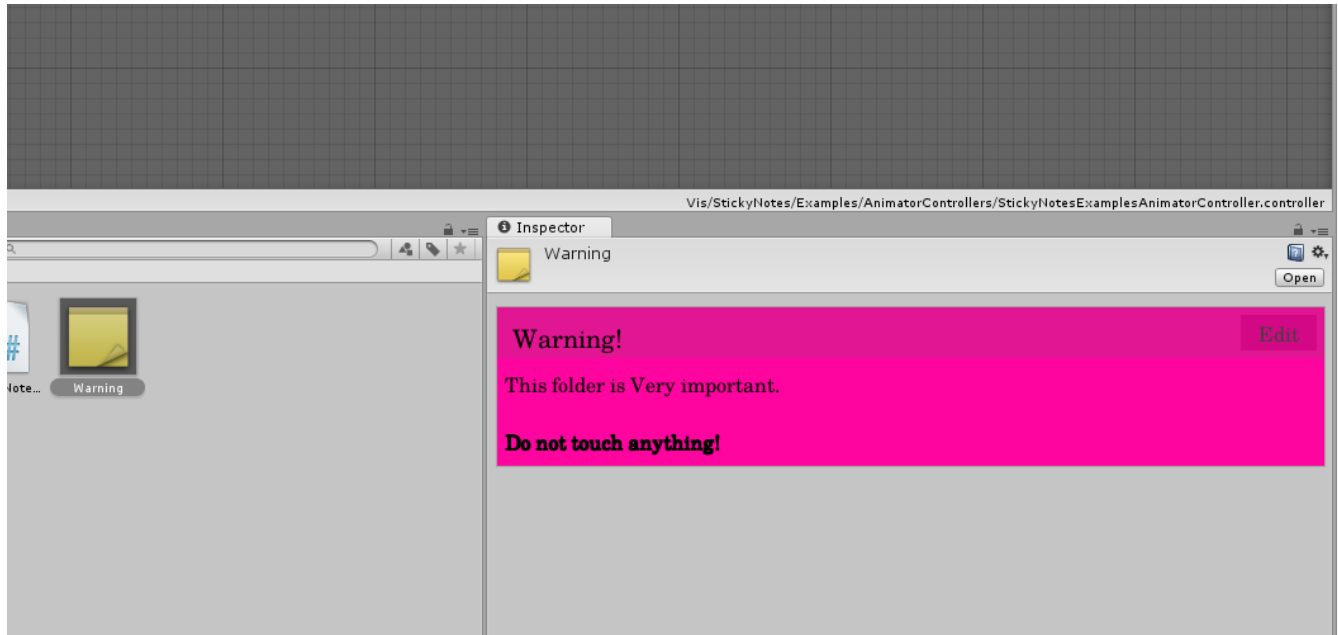
# Assets

You can create standalone Note-assets and embedded ones.

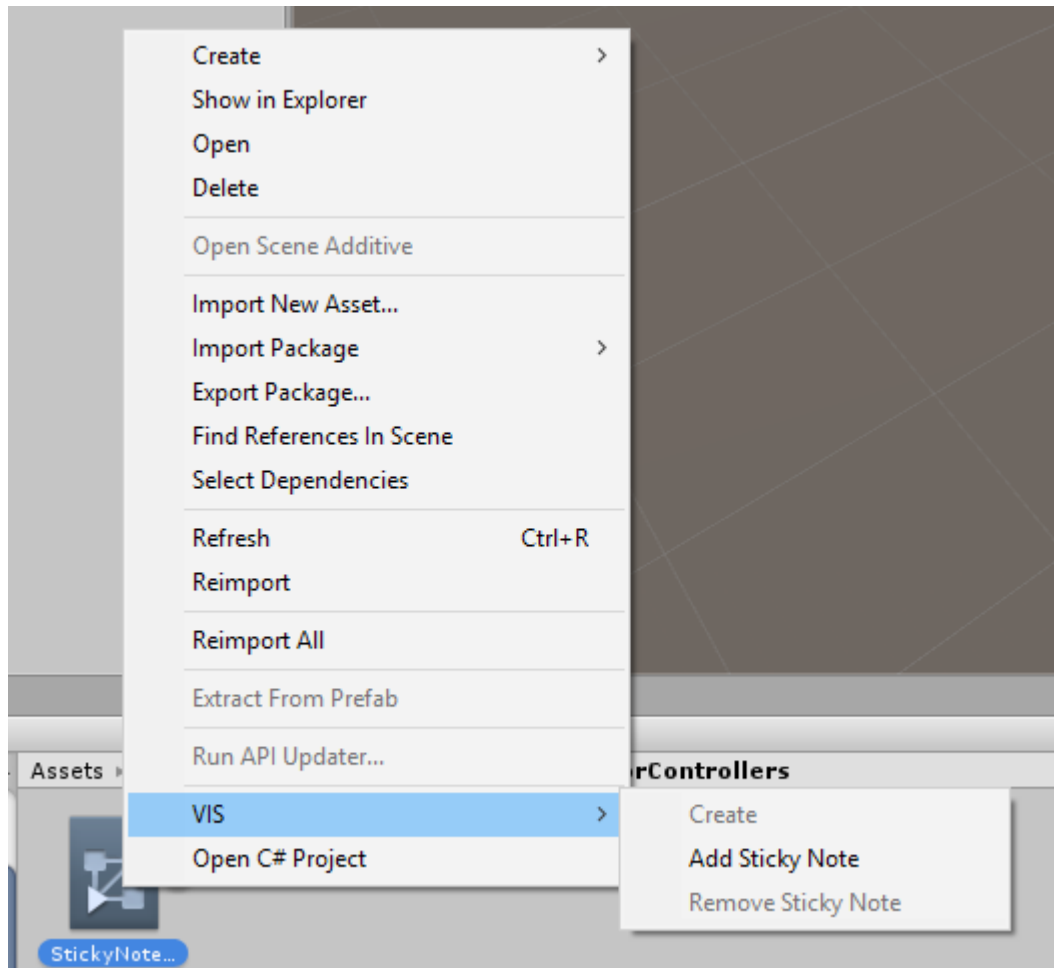
To create Standalone asset, right-click on the empty directory area in *Project* window and choose *Vis* → *Create*



Standalone Sticky Note asset will be created in that directory and it can serve as a Note for the whole directory:

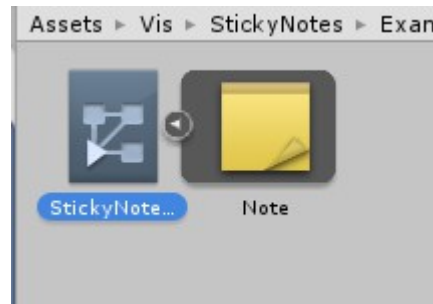


You can add Sticky Notes as sub-assets to existing assets also. For example, you can right click on *AnimatorController* asset and choose *Vis* → *Add Sticky Note*

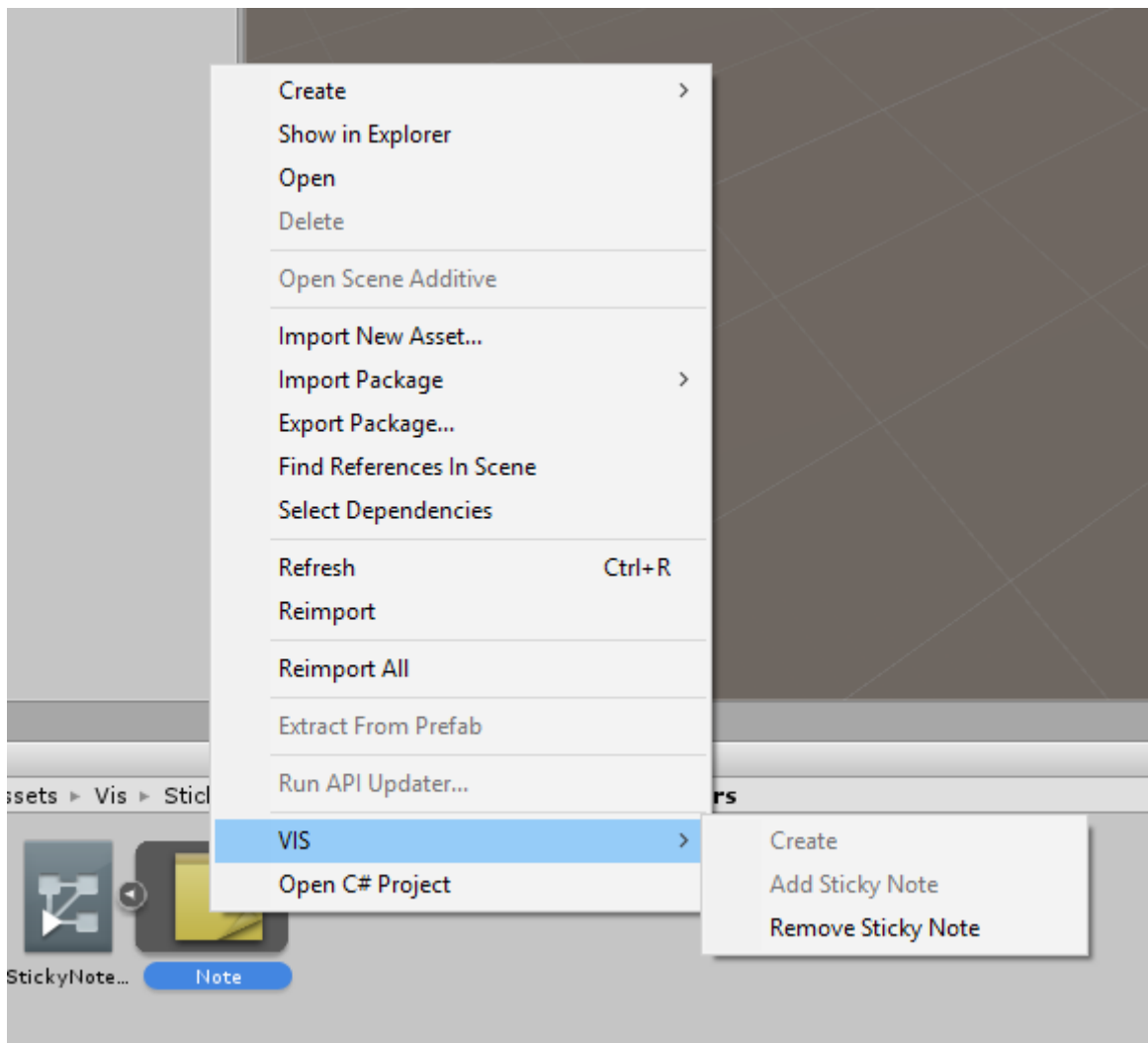




You'll find all sub-assets by unfolding parent asset with a little triangle button.

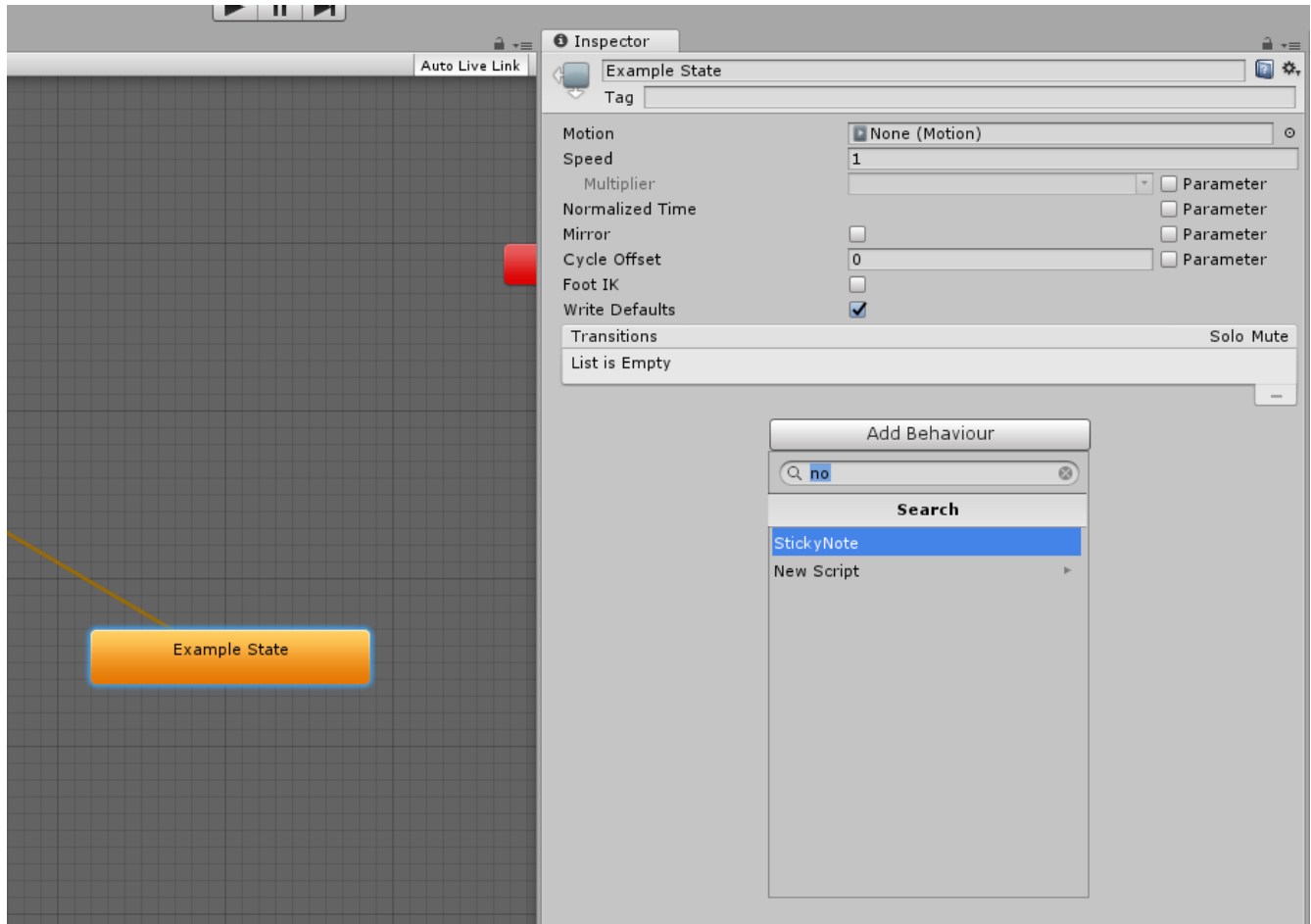


To remove sub-asset right click on it and choose *Vis* → *Remove Sticky Note*

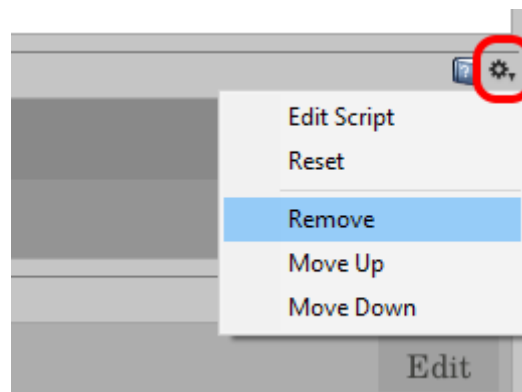


# StateMachineBehaviours

To add a Note to AnimatorState click on *Add Behaviour* button, just like adding regular StateMachineBehaviour script, and choose *StickyNote*.



To remove – click on little gear button and choose *Remove*



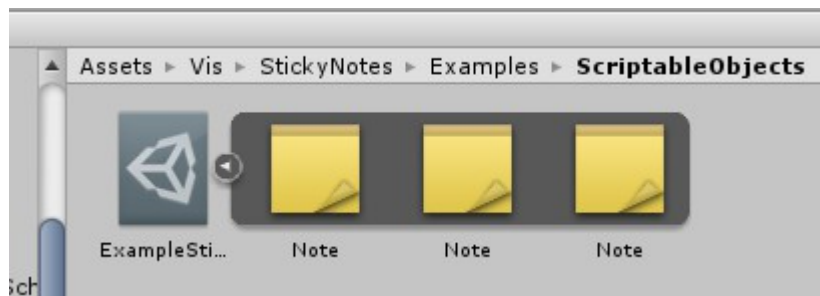
# Extending Your Custom Asset Classes with StickyNotes scripting API

To extend your asset class with StickyNotes sub-assets drawing, create custom editor class for the class you wish to extend (add `CustomEditor` attribute with your target class' type as a parameter) but inherit it from *MultipleStickyNotesEditorBase* class instead of regular *Editor* class. You'll find *MultipleStickyNotesEditorBase* class in *VIS.StickyNotes.Editor* namespace.

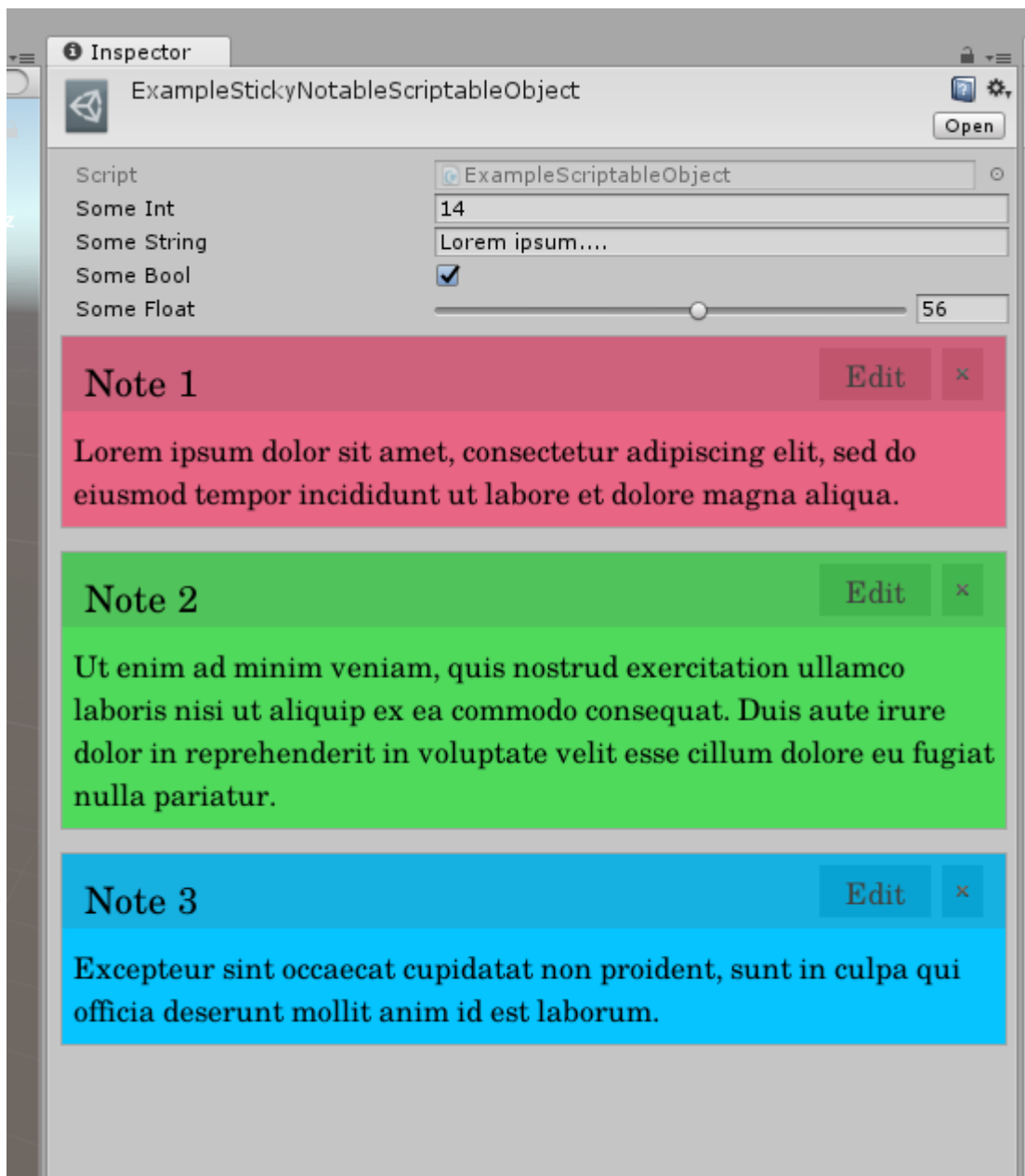
```
ExampleScriptableObjectEditor.cs
ObjectDescriptor.Editor
VIS.StickyNotes.Examples.Editor.ExampleScriptableObjectEditor

1 using UnityEditor;
2 using VIS.StickyNotes.Editor;
3
4 namespace VIS.StickyNotes.Examples.Editor
5 {
6     [CustomEditor(typeof(ExampleScriptableObject))]
7     public class ExampleScriptableObjectEditor : MultipleStickyNotesEditorBase { }
8 }
9
```

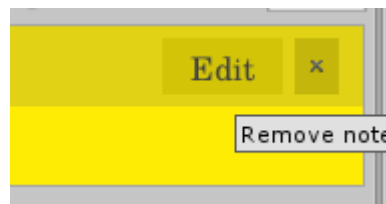
That's it! Now your asset will draw notes if there would be any note subasset inside it.



In other ways *MultipleStickyNotesEditorBase* will behave as regular *Editor* class, i.e. you can build your custom editors based on this class.



When StickyNotes embedded in such way, another notes-removing option will appear – with the *Close* button.



Support: [celtic.gua@gmail.com](mailto:celtic.gua@gmail.com)