Beautifiers

Introduction

This package is a collection of generally useful Property Drawers for unity. Each user which buys the asset can email us a request for a new drawer and chances are high that it will be made available unless it is too complex or undoable with our knowledge or in general. Below is the list of all drawers and their usage.

LowerCaseString

It will make the string lower case so even if the user types upper case letters in the inspector, the drawer will turn it into lower case.

UpperCaseString

This is like the above but will make the string upper case

ReplaceCharacter

This works on strings as well, it takes two arguments so if you put [ReplaceCharacter(' ','_')] on top of a variable, All spaces typed in ti will be turned to _ characters.

PopUp

This works on strings too. You put a list of desired values in the Popup and then the user can choose one of them as the value for the string so PouUp("Unity","UE4")] will show Unity and UE4 as values for the string in a popup and user can only chooose one of them for the value.

Password

If you want to hide the value of a string when the user types it in inspector, this attribute is for you. Useful for passwords and secrets and ...

Colorize

This works on all types of public variables, you put it on top of the variable and give it R, G and B values and it shows the variable in taht color to emphasise it. So [Colorize(0,0,255)] will make the variable blue and so on. Usage of Color is not possible here.

DisplayBasedOnBool

You can put this on top of any field which you want to show/hide when a bool is true. As parameters it takes name of a bool variable which should be in the same script and a boolean which says when the specified bool is true, we should show or hide this field. Default is that we show the field if the bool is true.

DsipayBasedOnEnum

You can put this on top of any field which you want to show/hide based on an enum in the same script. The attribute takes 3 parameters, the first is name of the enum field in the same script, second is the int representation of the enum value which will cause showing/hiding of this field and the third is a bool indicating if we should show/hide the field. Let's say we have an enum QuestType {ZoneBased, Global} and we want to show a string zoneName when ZoneBased is selected. We define a public field named QuestType type and a string zoneName with an attribute like this
[DispalyBasedOnEnum("type",(int)QuestType.ZoneBased,true)]

The cast to int is because we wanted this to be generic and usable with any enum typ. Now the zoneName is shown only when ZoneBased is selected in type, if we set the show property to false, it would only be hidden when ZoneBased was selected in type.

ScriptableObjectSelector

You put this on top of variables which should take a ScriptableObject and mention the type, If you want a user to select from the list of weapons for example and have a ScriptableObject class names Weapon you put like this [ScriptableObjectSelector(typeof(Weapon)] on top of it and the user can choose one of the available weapons from the drop down. The Weapon ScriptableObjects should be created before of course.

TexturePreview

You put this on top of a Texture or Texture2d to show a preview of it below the variable.

TimeMinutes

If you want to put time in a int variable in seconds but want the user to be able to type it in minutes and seconds, this is for you. It will show a minutes and seconds field but the result in seconds will be stored in the variable.

TimeHours

This is like the above but has a field for specifying hours as well.

DescriptiveToggle

This is used for bool values, If you want to show a text for on and off states of the checkbox, this is for you. If you put [DescriptiveToggle("Yes","No")] on top of your bool value, then Yes and No will be shown beside the checkbox in the inspector for on and off states respectively.