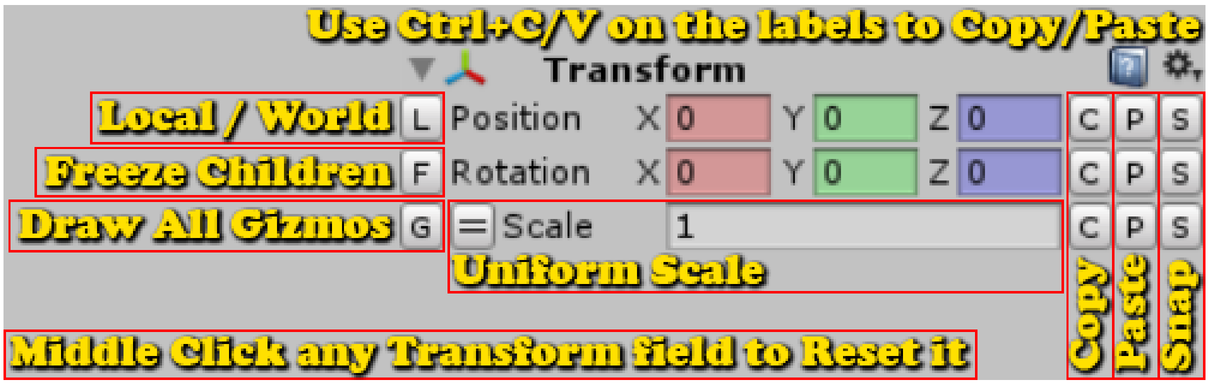
Inspector Gadgets

v5.0 Created by Kybernetik

1. Transform Inspector



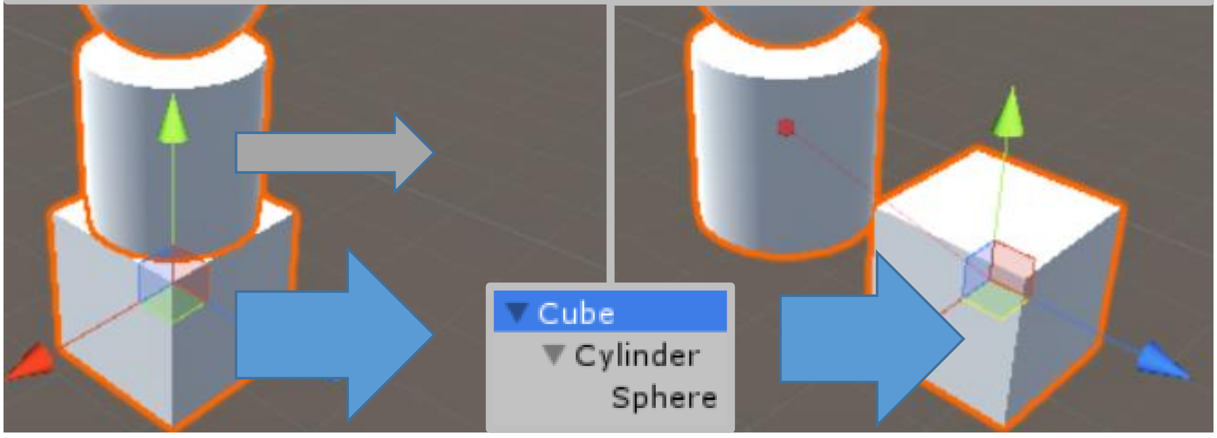
Inspector Gadgets Pro allows you to customise its settings in the *Edit/Preferences* window.

1.1. Local / World

The [**L**] button allows you to toggle the inspector between [**L**] local space and [**W**] world space.

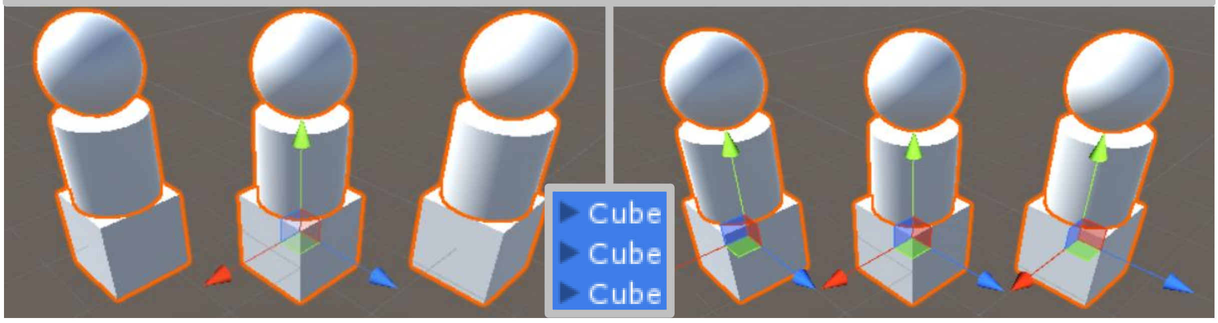
1.2. Freeze Children

Normally moving the parent would move the children as well	But with **Freeze Children** enabled moving the parent won't affect the children



1.3. Draw All Gizmos [Pro-Only]

Normally Unity only draws a scene gizmo for the first object you selected	But with **Draw All Gizmos** enabled it will draw a separate gizmo for each selected object



1.4. Uniform Scale

Most of the time when you scale an object you want to do so on all axes so that its proportions remain the same. When the object's scale is uniform (same value on all axes) it will be shown as a single value so you don't need to enter the same value multiple times.

- You can toggle between uniform scale mode and the regular vector mode using the [**=**] button on the left.
- If the scale isn't already uniform when you enter uniform scale mode, it will be set to the average of the current values.

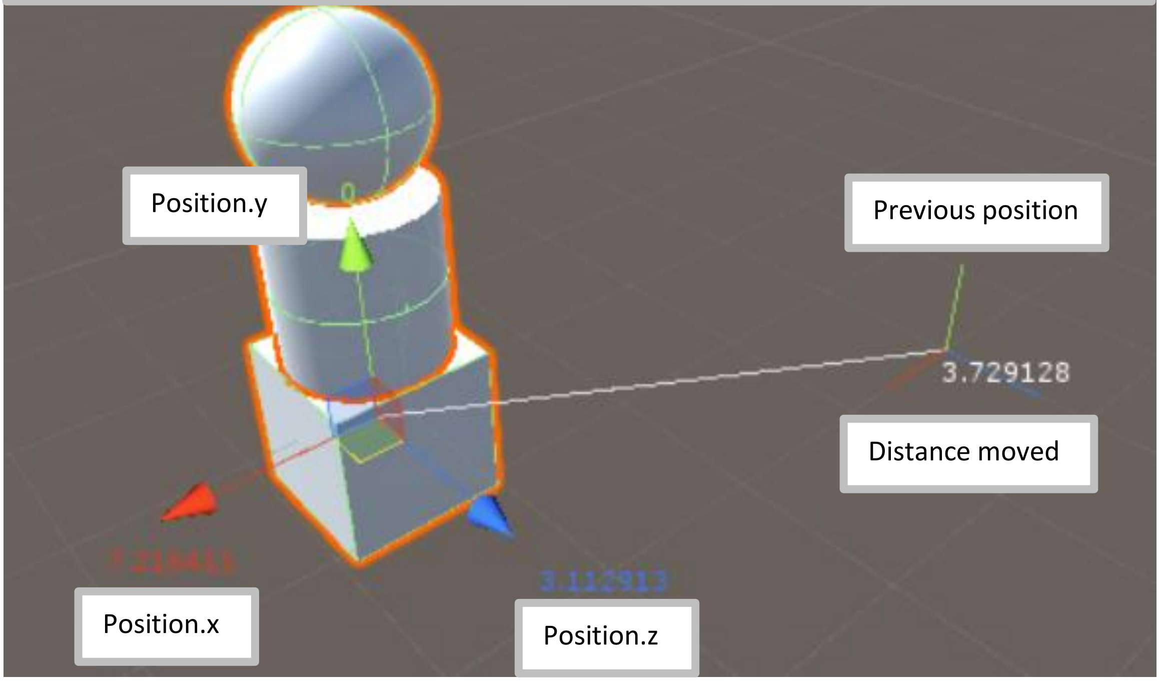
1.5. Utility Buttons

You can use the buttons on the right to [**Copy**], [**Paste**], and [**Snap**] individual transform elements (position, rotation, scale).

- The [**C**] and [**P**] buttons are integrated with the system clipboard, allowing you to copy values to and from other vector fields as well as other programs.
- Position, rotation, and scale also each have their own private clipboard, allowing you to copy a position and then a rotation at the same time without overwriting the previous value. To paste from the private clipboard, you simply right click the [**P**] button.
- You can also right click the [**C**] button to log a message containing the field's current value.
- The snap buttons use Unity's own snap settings which can be opened by right clicking them.
- These buttons are greyed out when they would do nothing. I.E. Copy and Paste are greyed out if the clipboard contains the same values as the selected object, and Snap is disabled if the object is already snapped.

1.6. Movement Guides [Pro-Only]

While you are moving an object in the scene, Inspector Gadgets will draw lines indicating the position you are moving the object from. This feature can be disabled in the *Edit/Preferences/Inspector Gadgets* menu.



1.7. Other Pro-Only Features

Inspector Gadgets Pro includes various additional features which are not available in the Lite version.

- Multi-object editing (the Lite version reverts to the default `Transform` inspector when multiple objects are selected).
- `Transform` fields which aren't at their default value will have a thicker border for emphasis.
- `Transform` fields which aren't a multiple of the snap value will be shown in italics.
- The *Edit/Selection/New Locked Inspector (Ctrl + Alt + S)* menu item opens an inspector window locked to the current selection so you can easily compare and copy values between different selection sets.

2. Auto Hide UI

Many users find it annoying having a Screen Space UI Canvas take up a massive amount of space in the scene. To work around this, the first time you select a UI object after importing Inspector Gadgets, it will ask if you want to automatically show and hide the UI layer.

- On UI selected: show UI layer, enter 2D orthographic mode, and focus the camera on the selected object.
- On UI deselected: hide UI layer and return camera to the previous state.
- The UI layer will be automatically shown when you close the Unity Editor just in case the next project you open doesn't have Inspector Gadgets.
- Inspector Gadgets Pro provides several options to control the conditions which trigger this feature and the way it works in the *Edit/Preferences/Inspector Gadgets* menu.
- Credit to Astral Byte Ltd for the original idea.

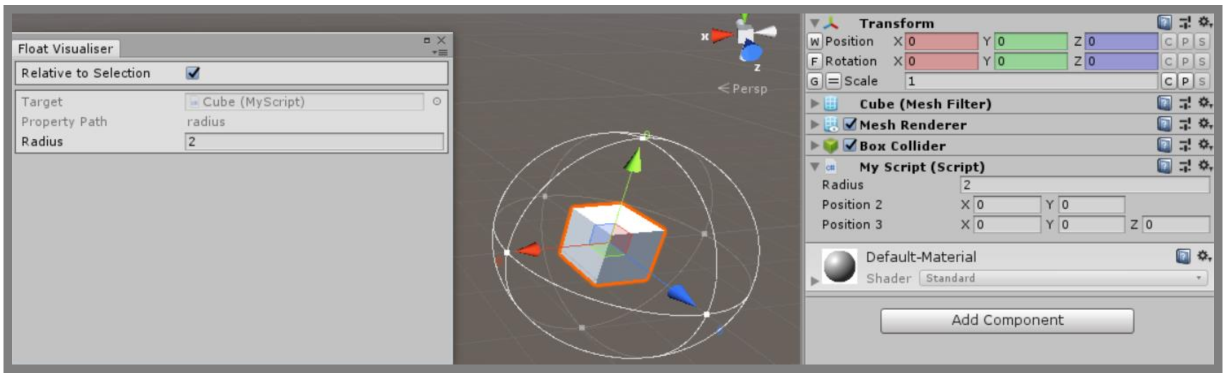
3. Context Menu Functions

Inspector Gadgets adds various useful functions to the context menu (right click menu) of each field in the inspector based on its type.

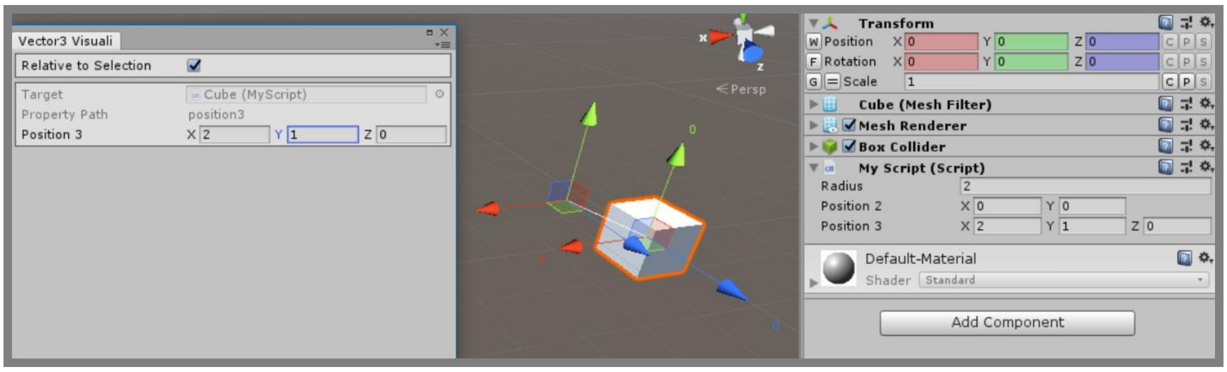
- These functions all support multi-object selection, but the Lite version disables them while multiple objects are selected.
- Most types have Copy and Paste functions which allow you to copy values between fields in Unity as well as to and from other programs.
- The fields in the `Transform` Inspector have functions to snap them to the grid, raycast down and snap to the ground, and rotate to look at another object.
- The fields in the `RectTransform` inspector have functions to square them (set the height equal to the width or vice versa) and to snap them to the edges of its siblings in a particular direction (Right/Up/Left/Down).
- Unity Object acquisition: `FindObjectOfType`, `GetComponent`, `GetComponentInChildren`, `GetComponentInParent`, `AddComponent`.
- Unity Object array acquisition: `FindObjectsOfType`, `GetComponents`, etc.
- Create new instance of any `ScriptableObject` type.
- Open Inspector to open a new inspector window showing the target object.
- Save any Unity Object as an asset (such as a procedurally generated mesh).
- Randomize within common ranges: 0-1, 0-100, 0-360 for floats, random Vector2 in a unit circle, random Vector3 on or in a unit sphere, random quaternion, random euler angles.
- Common vectors: zero, right, up, forward, one.
- Normalize vector.
- String to lower or upper case.
- Log current value.

3.1. Visualise

For floats and vectors only. Opens an editor window that displays a gizmo in the scene to visualise and manipulate the chosen field. Vectors are visualised using a position handle with a line leading back to the origin while floats are visualised using the same wireframe sphere used to display the shape of a Sphere Collider.


Float Visualiser
Relative to Selection ✓
Target: Cube (MyScript)
Property Path: radius
Radius: 2

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale 1
Cube (Mesh Filter)
✓ Mesh Renderer
✓ Box Collider
My Script (Script)
Radius 2
Position 2 X 0 Y 0
Position 3 X 0 Y 0 Z 0
Default-Material
Shader: Standard



Vector3 Visuali

Relative to Selection ✓

Target: Cube (MyScript)

Property Path: position3

Position 3: X 2 Y 1 Z 0

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale 1

Cube (Mesh Filter)

✓ Mesh Renderer

✓ Box Collider

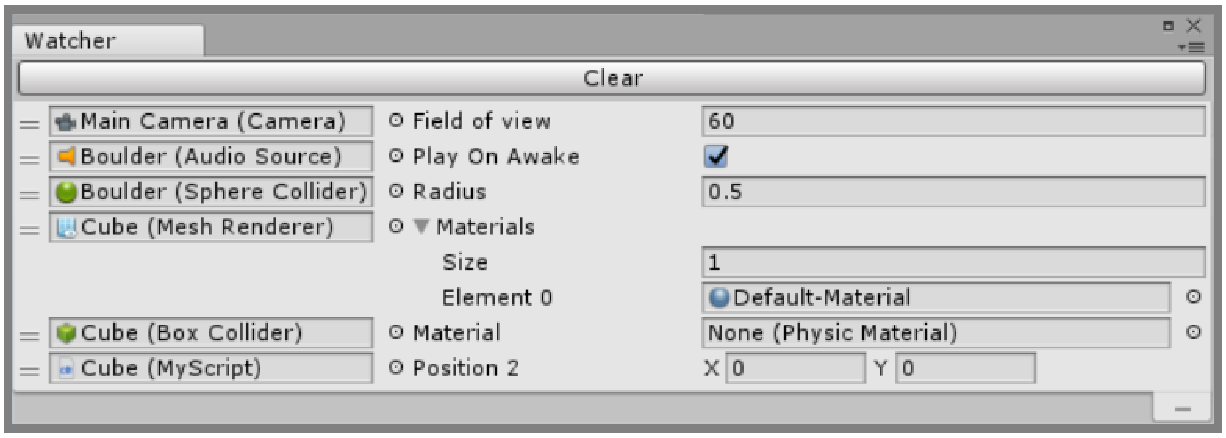
My Script (Script)
Radius 2
Position 2 X 0 Y 0
Position 3 X 2 Y 1 Z 0

Default-Material
Shader Standard

Add Component

3.2. Watch

Opens an editor window that displays a list of all fields you have used the function on so you can view (and edit) them all at the same time. The Lite version only allows you to watch up to 3 fields at a time.



Watcher

Clear

Main Camera (Camera)	Field of view	60
Boulder (Audio Source)	Play On Awake	✓
Boulder (Sphere Collider)	Radius	0.5
Cube (Mesh Renderer)	Materials	
	Size	1
	Element 0	Default-Material
Cube (Box Collider)	Material	None (Physic Material)
Cube (MyScript)	Position 2	X 0 Y 0

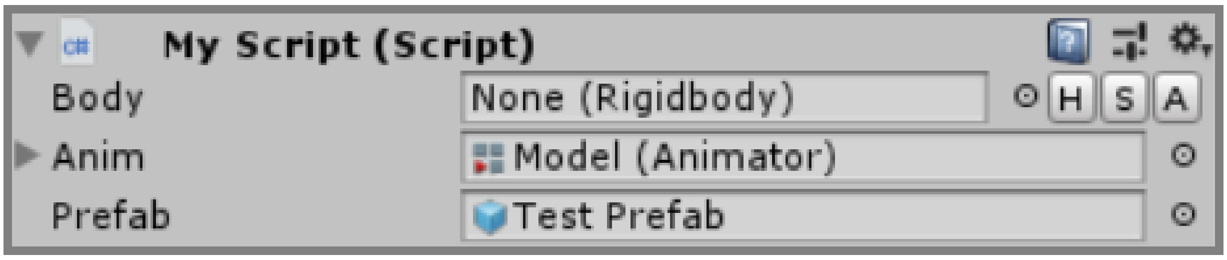
4. Script Inspector [Pro-Only]

This section applies to `MonoBehaviour`, `ScriptableObject`, and `StateMachineBehaviour`.

4.1. Object Reference Fields

While an `Object` reference field has no reference it will be shown with several buttons to find one:

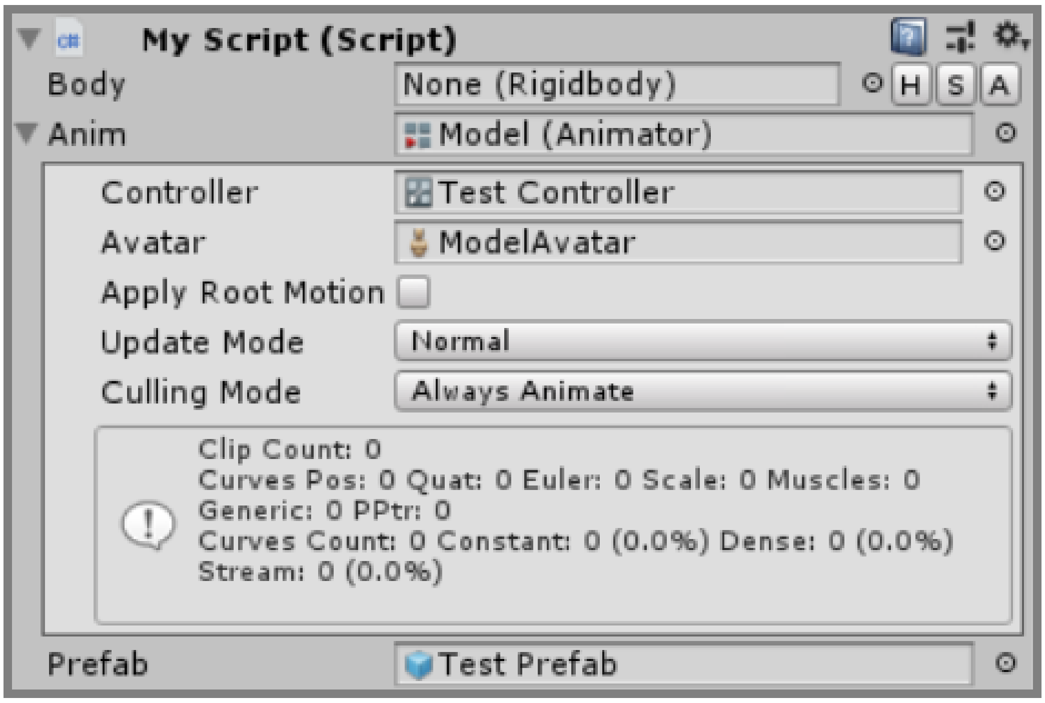
- [**H**] - Find in Hierarchy (`Component` references only) - Searches all child and parent objects.
- [**S**] - Find in Scene - Searches all objects in the scene.
- [**A**] - Find in Assets - Searches all assets in the project.
- These buttons search all potential candidates to find the one with a name closest to that of the field in question.
- The context menus for these fields as well as for all components also contain similar functions.



Body: None (Rigidbody)
Anim: Model (Animator)
Prefab: Test Prefab

While a reference is assigned, it will be shown with a foldout arrow to show the referenced object's editor nested below the field.

- Unfortunately, due to the way the nested editor is drawn, arrays and lists of references will draw their editors beneath the list rather than beneath each individual field.
- If you make a custom class, you can add a `private const bool NestedObjectDrawers = false;` field to is to disable nested object drawers for it in case they don't fit within the layout of your custom editor.



Body: None (Rigidbody)
Anim: Model (Animator)
Controller: Test Controller
Avatar: ModelAvatar
Apply Root Motion: ☐
Update Mode: Normal
Culling Mode: Always Animate

```
Clip Count: 0
Curves Pos: 0 Quat: 0 Euler: 0 Scale: 0 Muscles: 0
Generic: 0 PPtr: 0
Curves Count: 0 Constant: 0 (0.0%) Dense: 0 (0.0%)
Stream: 0 (0.0%)
```

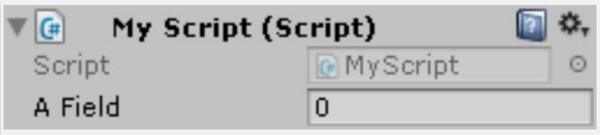
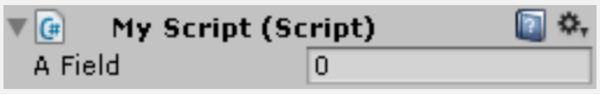
Prefab: Test Prefab

4.2. Script Field

By default, Inspector Gadgets hides the *Script* field at the top of Unity's script inspector to save space and provides the following extra functions:

- *Middle Click* in a script’s inspector to open that script.
- *Ctrl + Middle Click* in a script’s inspector to open its custom inspector script (or create a one if it doesn't exist, see Custom Inspectors).
- *Shift + Middle Click* in a script’s inspector to ping the script asset in the Project window.
- This feature can be disabled via the component’s context menu or the Inspector Gadgets tab of the Edit/Preferences menu.

```
// Example.
class MyScript : MonoBehaviour
{
    public int aField;
}
```

Unity Default	**Inspector Gadgets**
 My Script (Script) Script: MyScript A Field: 0	 My Script (Script) A Field: 0

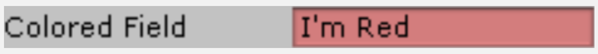
4.3. Decorator Attributes

Unity has several attributes which can be used to easily customise the appearance of the inspector without needing to write a custom inspector class: Header, HideInInspector, Multiline, Range, SerializeField, Space, TextArea, Tooltip. The `InspectorGadgets` namespace contains some additional attributes to allow for even greater customisation.

4.3.1. Color

The `[Color]` attribute simply changes the color of a field, allowing you to highlight it or group multiple fields together visually.

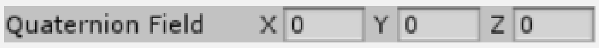
```
[InspectorGadgets.Color(1, 0.5f, 0.5f)]
public string coloredField = "I'm Red";
```


Colored Field: I'm Red

4.3.2. Euler

The `[Euler]` attribute displays the Euler angles of a `Quaternion` field so they can be more easily edited by people without an intimate understanding of quaternions (which is everyone).

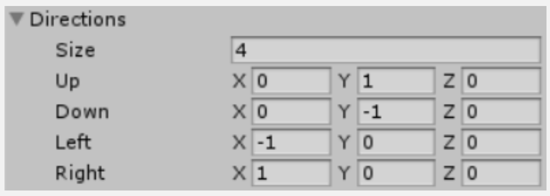
```
[InspectorGadgets.Euler]
public Quaternion quaternionField;
```


Quaternion Field X 0 Y 0 Z 0

4.3.3. Labelled Collection

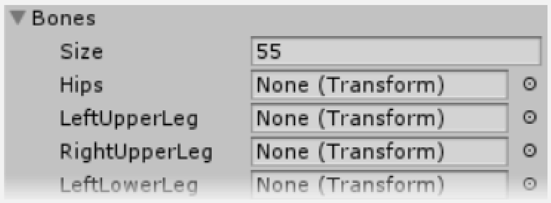
The [LabelledCollection] attribute provides labels for the elements of a collection field to use instead of just calling them Element X.

```
// Manually specify labels.
[InspectorGadgets.LabelledCollection(
    "Up", "Down", "Left", "Right")]
public Vector3[] directions;
```



Directions
Size 4
Up X 0 Y 1 Z 0
Down X 0 Y -1 Z 0
Left X -1 Y 0 Z 0
Right X 1 Y 0 Z 0

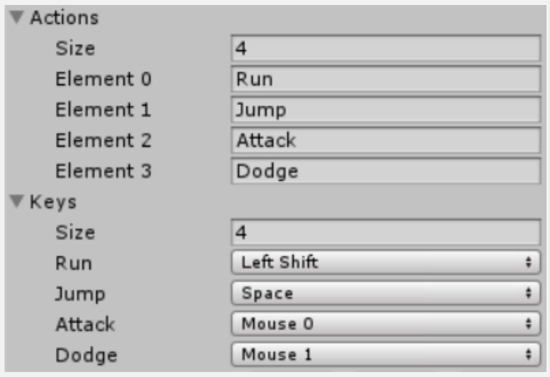
```
// Use names from an Enum.
[InspectorGadgets.LabelledCollection(
    typeof(HumanBodyBones))]
public Transform[] bones;
```



Bones
Size 55
Hips None (Transform)
LeftUpperLeg None (Transform)
RightUpperLeg None (Transform)
LeftLowerLeg None (Transform)

```
// Use names from another field.
public string[] actions;

[InspectorGadgets.LabelledCollection(
    "actions")]
public KeyCode[] keys;
```

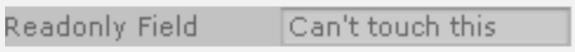


Actions
Size 4
Element 0 Run
Element 1 Jump
Element 2 Attack
Element 3 Dodge
Keys
Size 4
Run Left Shift
Jump Space
Attack Mouse 0
Dodge Mouse 1

4.3.4. Readonly

The [Readonly] attribute causes a field to be greyed out in the inspector so the user can't edit it.

```
[InspectorGadgets.Readonly]
public string readonlyField = "Can't touch this";
```

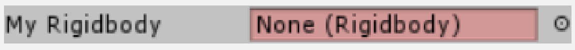


Readonly Field Can't touch this

4.3.5. Required

The [Required] attribute causes a field to be highlighted red in the inspector if it hasn't been given a value so the user knows that they should assign something to it. This attribute is very useful on object reference fields, but can be used on any type such as int or string.

```
[InspectorGadgets.Required]
public Rigidbody myRigidbody;
```

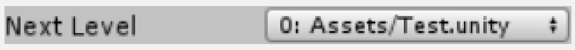


My Rigidbody None (Rigidbody)

4.3.6. Scene

The [Scene] attribute can be placed on an int or string field to have it drawn as a dropdown box for selecting scene indices or names from the build settings.

```
[InspectorGadgets.Scene]
public int nextLevel;
```



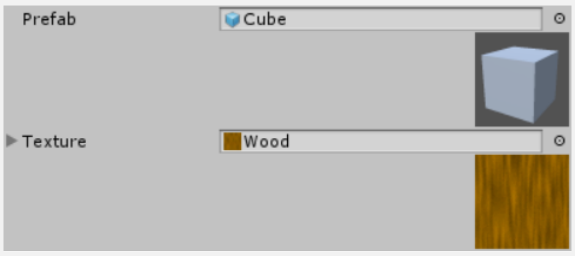
Next Level 0: Assets/Test.unity

4.3.7. Show Preview

The [ShowPreview] attribute can be placed on an Object reference field to show a preview image of the referenced object.

```
[InspectorGadgets.ShowPreview]
public GameObject prefab;

[InspectorGadgets.ShowPreview]
public Texture texture;
```

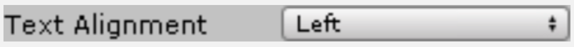


4.3.8. Toolbar

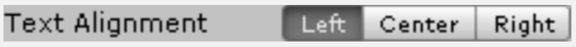
The [Toolbar] attribute replaces the attributed field with a set of buttons to let you easily select from among a specific set of options without opening a dropdown menu. It can be used on various different field types:

- bool - Allows you to specify user friendly labels such as yes/no or on/off.
- Enum - Automatically uses the enum's values as the button labels. If the enum has a [System.Flags] attribute the buttons will act as toggles to be activated and deactivated in any combination.
- string - Allows you to specify your own set of options for the field.
- If the toolbar has too many values to fit into the available space, you can scroll across using your scroll wheel while the mouse cursor is pointing at it or you can drag the field label (just like a number field).

```
public TextAlignment textAlignment;
```



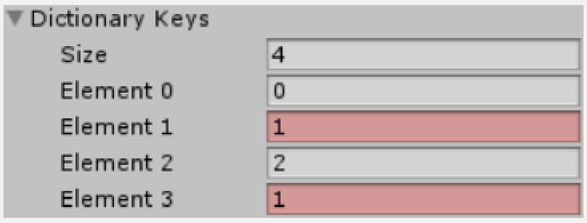
```
[InspectorGadgets.Toolbar]
public TextAlignment textAlignment;
```



4.3.9. Unique Collection

The [UniqueCollection] attribute causes the elements of a collection to be highlighted in red if they are duplicates of other elements.

```
[InspectorGadgets.UniqueCollection]
public int[] dictionaryKeys;
```



4.3.10. Validator Attributes

These attributes enforce specific rules on the attributed fields:

- [MinValue] Sets a minimum which the value of a numeric field cannot fall below.
- [MaxValue] Sets a maximum which the value of a numeric field cannot rise above.
- [ClampValue] Sets both a minimum and maximum for the value of a numeric field.
- [HasComponent] Placed on a GameObject field to deny any references to objects which don't have a specific component type attached.

4.4. Inspectable Attributes

Unlike the above attributes which inherit from `PropertyAttribute` and alter the way Unity shows a field in the inspector, the following attributes inherit from `InspectableAttribute` and add extra elements at the bottom of the inspector.

These attributes have several properties which can be set in the constructor:

- `Label` and `Tooltip` - the text and tooltip to show in the inspector.
- `DisplayIndex` - determines the order in which the inspectable is to be drawn amongst the regular serialized fields, starting with 0 at the top of the inspector.
- `When` - determines when the inspectable is drawn: always (default), in play mode only, or in edit mode only.

4.4.1. Inspectable

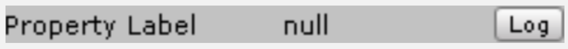
The `[Inspectable]` attribute adds the attributed field or property to the inspector as if it were serialized. Works on static members as well.

4.4.2. Label

The `[Label]` attribute adds a read-only label to display the value of any field, property (with a getter), or method (with no parameters), even `static` ones.

- If the value will change constantly, you can set the `ConstantlyRepaint` property to true.
- If the value is likely to be long, you can set the `Multiline` property to true.

```
[InspectorGadgets.Label]
string PropertyLabel { get; set; }
```



4.4.3. Button

The `[Button]` attribute adds a button in the inspector which the user can click to invoke the attributed method.

- The will automatically be given the method's name (with spaces between words) unless you specify a different `Label` in its constructor.
- If it is an instance method and multiple objects are selected, the method will be called once for each selected object.
- If the method has a non-void return type, the returned value will be logged.
- If you assign `SetDirty = true` in the attribute's constructor, it will automatically call `UnityEditor.EditorUtility.SetDirty` on the target after invoking the method to make sure changes are saved properly.

```
[InspectorGadgets.Button]
void InspectorButton() { }
```

4.5. Shader Attributes

These attributes can be placed on properties in shaders to modify their appearance and behaviour in the inspector much like Decorator Attributes do in regular scripts. Unity comes with several of these attributes listed in the Unity Manual (look under the "Property attributes and drawers" headding).

Currently Inspector Gadgets only has one of these attributes, so if you have any suggestions for new ones please email kybernetikgames@gmail.com.

4.5.1. Tex Define

The `[TexDefine]` attribute can be placed on texture properties to specify a keyword that will be added to the material whenever a texture is assigned to the property. This allows you to more easily create Shader Variants.

4.6. Inspector Events

If Decorator and Inspectable Attributes aren't able to achieve the level of customisation you want, you can simply add a method called `AfterInspectorGUI` to your script and Inspector Gadgets will automatically call it after drawing the script inspector (so you can draw additional GUI elements below it). Or you can call the method `OnInspectorGUI` to replace the regular inspector entirely.

4.7. Custom Inspectors

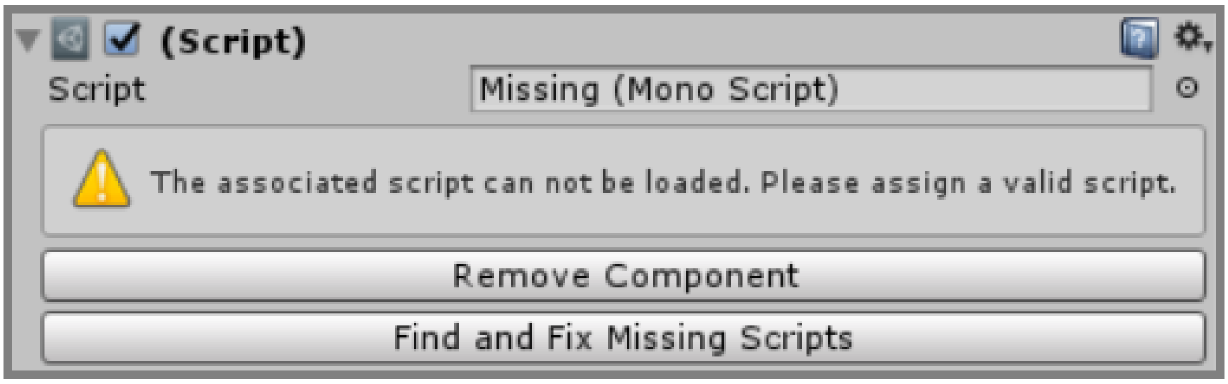
If you require more customisation than you can achieve using the available events and decorator attributes, you will need to create your own custom editor class. You can do this quickly using *Ctrl + Middle Click* in the target script's inspector or using the *Create Editor Script* context menu command.

By inheriting from `InspectorGadgets.Editor<T>` instead of `UnityEditor.Editor` your custom editor will inherit the features described above (hide Script field, Middle Click to open script, support for Inspectable Attributes and Inspector Events). You will also be able to access the inspected objects directly using the `Target` and `Targets` properties (note the capitalisation) rather than needing to type-cast the regular `target` and `targets` properties yourself.

4.8. Missing Scripts

Inspector Gadgets improves the inspector for missing scripts be adding a button to easily remove the component in question as well as one to open a window that will search through all assets and scenes to find any more missing scripts. This window can also be opened from the *Edit/Preferences/Inspector Gadgets* window.

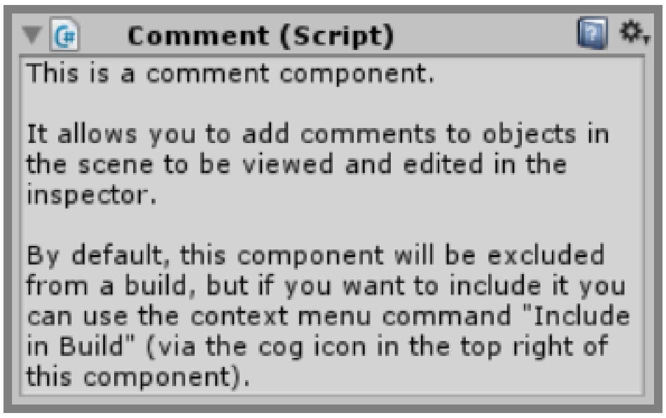
The Scripty Hunter window tries to suggest other possible alternatives for each missing script based on its name and the names and types of each of its serialized fields. Unfortunately, Unity tends to lose this data after scripts go missing so it isn't always possible to offer any suggestions.



The associated script can not be loaded. Please assign a valid script.

5. Misc

5.1. Comment Component



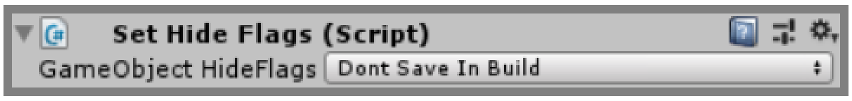
This is a comment component.

It allows you to add comments to objects in the scene to be viewed and edited in the inspector.

By default, this component will be excluded from a build, but if you want to include it you can use the context menu command "Include in Build" (via the cog icon in the top right of this component).

5.2. Set Hide Flags Component

This component can be added to a scene object to modify its hide flags in the inspector. Adding the component automatically sets the `DontSaveInBuild` flag.



GameObject HideFlags: Dont Save In Build

5.3. Auto Prefs

The `InspectorGadgets.AutoPrefs` class contains a group of nested classes which simplify the way you can store and retrieve values in `PlayerPrefs` and `EditorPrefs`.

```
// First you declare your pref with the key and default value (optional).
public static readonly AutoPrefs.Bool MyPref = new AutoPrefs.Bool("MyPref", true);

// Then you can get and set the value without using the key everywhere.
if (MyPref)// Or MyPref.Value.
{
    MyPref.Value = false;
}
```

- `AutoPrefs.Bool` stores its value in `PlayerPrefs` while `AutoPrefs.EditorBool` stores its value in `EditorPrefs`.
- Other pref types are also available: `float`, `int`, `string`, `Vector2`, `Vector3`, `Vector4`, `Quaternion`.
- You can create your own Auto Pref types by inheriting from `AutoPref<T>`.

5.4. Editor Events

- The `[InspectorGadgets.OnWillUnloadAssemblies]` attribute registers a static parameterless method to be called immediately before the Unity Editor unloads assemblies, such as when preparing to reload after compiling modified scripts.
- The `[InspectorGadgets.OnEditorQuit]` attribute registers a static parameterless method to be called when the Unity Editor is closed (but not each time assemblies are reloaded, such as when scripts recompile).

Questions, feedback, feature requests, etc: kybernetikgames@gmail.com.