

# MANUL Toolbar

## User Guide

<b>1. Introduction.....</b>	<b>3</b>
1.1 Important information.....	3
1.2 Version History.....	3
<b>2. Quick Start.....</b>	<b>5</b>
2.1 Toolbar Sides.....	5
2.2 Shortcut Way.....	5
2.3 Manul Toolbar Settings.....	7
2.4 Override Toolbar Side.....	8
2.5 Manual Way.....	8
<b>3. Adding New Item.....</b>	<b>10</b>
3.1 Type Item Label.....	10
3.2 Select Item Type.....	11
3.2 Add Actions To Item.....	11
3.3 Toggle Item.....	12
<b>4. Settings.....</b>	<b>13</b>
4.1 Offsets.....	14
4.2 Default Styles.....	14
4.3 Other.....	14
4.4 Override.....	15
4.4.1 Use Button List Asset.....	15
4.4.2 Use Button List Set Asset.....	16
<b>5. Item Visuals.....</b>	<b>18</b>
5.1 Label Type.....	18
5.1.1 Text.....	18
5.1.2 Icon.....	18
5.1.3 Both.....	19
5.1.3 None.....	19
5.2 Option Toggles.....	19
5.2.1 Style.....	19
5.2.2 Width.....	21
5.2.3 Colors.....	21
5.2.4 Tooltip.....	22
<b>6. Item Types.....</b>	<b>23</b>
6.1 Button.....	23
6.1.1 Adding Button Actions.....	23
6.1.2 Mouse and Keyboard Buttons.....	23

6.1.3 Shortcut: Create Button Menu Item.....	25
6.1.4 Shortcut: Change Button Name.....	26
6.1.5 Shortcut: Disable Button.....	26
6.2 Label.....	27
6.3 List.....	27
6.3.1 List Editor Pref.....	27
6.3.2 List Actions.....	28
6.3.3 Creating Sub-dropdowns.....	28
6.3.4 Fill Button.....	29
6.3.5 Grouping Actions.....	30
6.3.6 Shortcut: Create List Menu Item.....	30
6.4 Value Fields.....	31
6.4.1 Editor Prefs.....	32
6.4.2 On Change Value Actions.....	32
6.4.3 Toggle.....	33
6.4.4 Popup.....	33
6.4.5 Slider.....	35
6.4.6 Number.....	36
6.4.7 Text.....	36
6.5 Other.....	36
6.5.1 Time Scale Slider.....	37
6.5.2 Frame Rate Slider.....	37
<b>7. Action Types.....</b>	<b>39</b>
7.1 Open Asset.....	39
7.2 Select Asset.....	40
7.3 Show Asset In Explorer.....	40
7.4 Properties Window.....	40
7.5 Static Method.....	41
7.6 Object of Type Method.....	41
7.7 Component Method.....	41
7.8 Open Folder.....	42
7.9 Find GameObject.....	42
7.10 Execute Menu Item.....	43
7.11 Invoke Event.....	44
7.12 Load Scene Additive.....	45
<b>8. Additional Resources.....</b>	<b>46</b>
<b>9. Conclusion.....</b>	<b>46</b>

# 1. Introduction

This document contains all the necessary information about the **MANUL Toolbar** asset.

The best way to learn about this asset is to start from the [2. Quick Start](#) section. Then, continue reading the next sections in the presented order to learn all the capabilities of this tool.

If you have any problems, questions, or suggestions, please write at [support@liquid-glass-games.com](mailto:support@liquid-glass-games.com) or reply to the topic on the **Unity Discussions Forum** [here](#).

## 1.1 Important information

There are a few things that you need to do or be aware of before you start:

- Make sure that you are using Unity 2020.2.0f1 or above, otherwise this tool will not work properly.
- The **Manul Toolbar Settings** asset has to be located in a **Resources** folder. It doesn't matter which **Resources** folder it is.
- Do not change the name of the **Manul Toolbar Settings** asset, otherwise the tool will not work. The name is **"Manul Toolbar"** in version 1.3 and above, and **"ManulToolbar"** in the previous versions.
- Make sure the setting **Player Settings / Editor / Use IMGUI Default Inspector** is set to **true**. Otherwise, there may be some problems in displaying this tool's user interface.

## 1.2 Version History

### Version 1.0.0

- Initial release

### Version 1.1.0

- New actions: **Open folder** and **Find GameObject**
- New item type: **Popup** (which uses *editor pref* of the int type)
- Shortcuts to quickly change the name of a button or disable it
- Menu items in *Project* window to quickly create a button
- You can store items' settings in an external asset

### Version 1.1.1

- Small fixes

### Version 1.1.2

- Fix in selecting asset in the Project window

### Version 1.2

- You can create a set of lists of items, and then switch between these lists with a popup
- Added menu items to open or select **Manul Toolbar Settings** asset

### Version 1.3.0

- Changed the way the settings are updated with the newer tool versions
- Various fixes

### Version 1.3.1

- Updated documentation & various fixes

### Version 1.3.2

- Added recreation of the settings file and upgraded version checking.

### Version 1.3.3

- New action: **Execute Menu Item**

### Version 1.4.0

- New action: **Invoke Event**
- New action: **Load Scene Additive**
- New item type: **List** - set of actions presented as a popup; action is performed each time the value of the popup is changed; value is stored in an *editor pref* (int type)
- New item type: **Slider** - presents float or int value as a slider, you can also configure minimum and maximum values
- New item type: **Number** - presents float or int value as a number field
- New item type: **Text** - presents string value as a text field
- **On Change Value Actions** - you can now add a set of actions for **Popup**, **Toggle**, **Slider**, **Number**, and **Text** that will be performed each time the value is changed
- You can use built-in editor icons (instead of dragging and dropping textures)

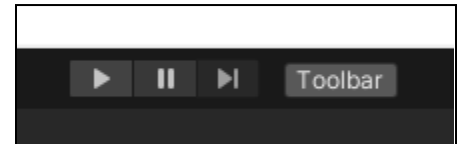
## 2. Quick Start

### 2.1 Toolbar Sides

There is a lot of empty space in the upper toolbar, just beneath the menu items. **Manul Toolbar** lets you use this space to create buttons, popups, toggles, and more, either on the **Left Side** or the **Right Side** of the toolbar.

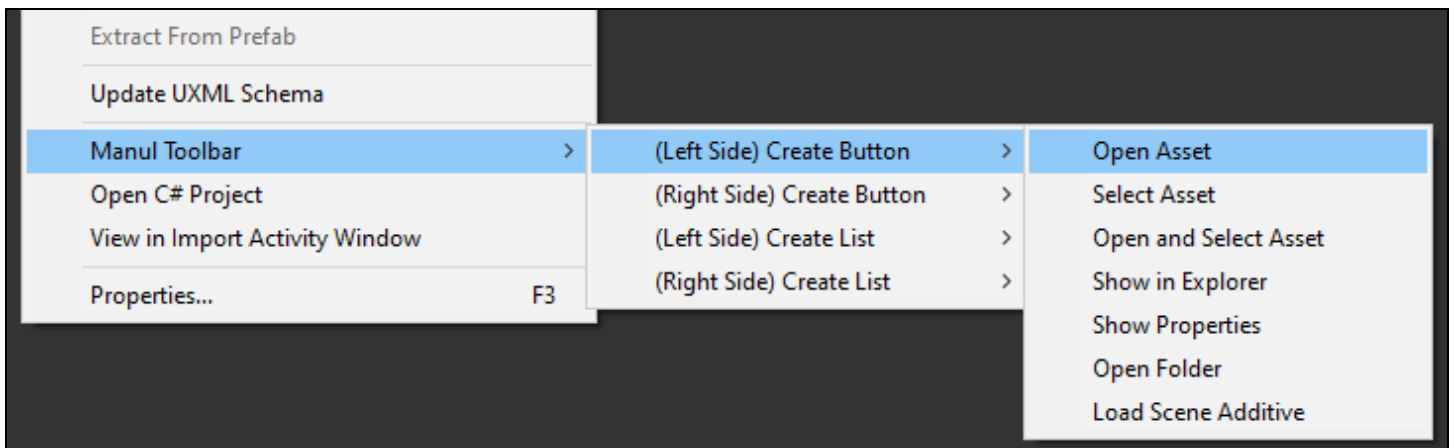


After importing the **Manul Toolbar** asset for the first time, you'll notice that a button named **Toolbar** has appeared on the right side. Leave it for now, we'll get back to it in a moment.

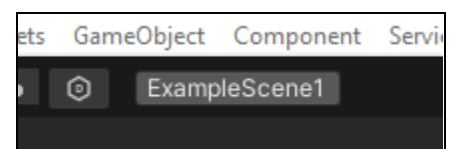


### 2.2 Shortcut Way

Let's say that you have a scene that you often open but this scene is deep down in your project's folder hierarchy. Each time you want to open the scene you have to waste your time looking for it... This is a perfect candidate for a button! To create it, simply right-click on this scene asset in the *Project* window and select **Manul Toolbar / (Left Side) Create Button / Open Asset**.



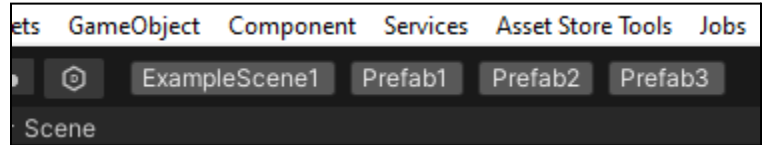
A button with the same name as the scene name will appear on the **Left Side** on the toolbar. Click on it and your scene will open. As simple as that!



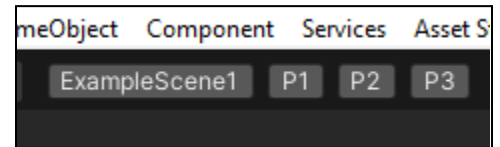
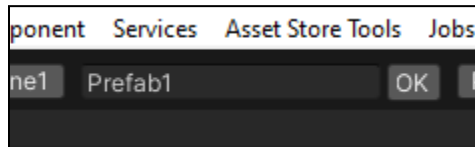
Maybe you have a couple of prefabs that you often modify, and you want to have a fast shortcut for each of them, both for opening and selecting them in the *Project* window?

Select all of them while holding down the **Shift** or **Ctrl** key, then right-click on any of them, and select **Manul Toolbar / (Left Side) Create Button / Open and Select Asset**.

New buttons will show up in the same number as the number of assets you have selected. Left-click a button to open the corresponding prefab, right-click to select this prefab in the *Project* window.

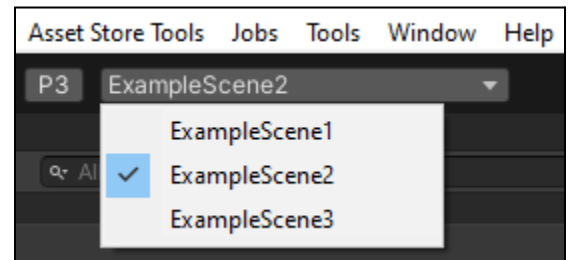


Buttons' names are too long? Left-click while holding down **Shift** and **Ctrl** keys on a button, enter a shorter name, and then click **OK**. You can repeat that with the rest of the buttons.



We've added a couple of buttons. Now let's add a different item - the popup **List**. This item lets you combine actions for different objects into one popup. For example, you can select a couple of scenes (while holding down **Shift** or **Ctrl** keys), then right-click on any of them, and select **Manul Toolbar / (Left Side) Create List / Open Asset**.

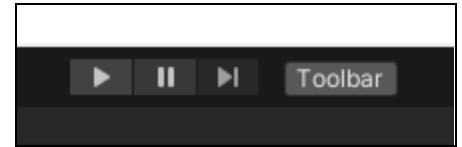
This will create a popup in the toolbar. Left-click on the popup and then again left-click on one of the popup's option (different from the current option) to load a scene associated with this option.



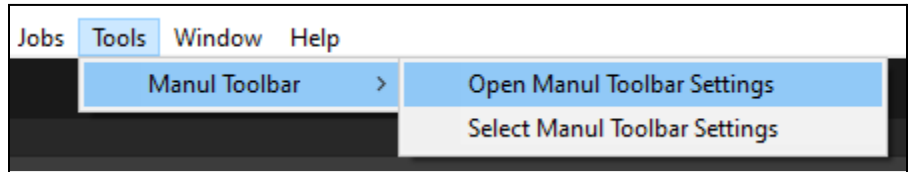
We've presented the shortcut way to do the most common things, but **Manul Toolbar** can do much more! Let's try the manual way. To do so, first you need to learn about the **Manul Toolbar Settings** asset.

## 2.3 Manul Toolbar Settings

Remember the **Toolbar** button on the **Right Side** of the upper toolbar? Left-click on it. This will open the **Manul Toolbar Settings** asset.

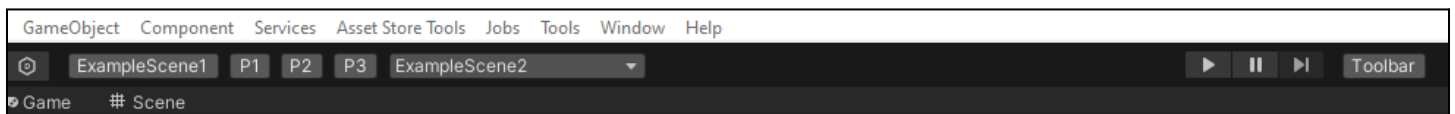
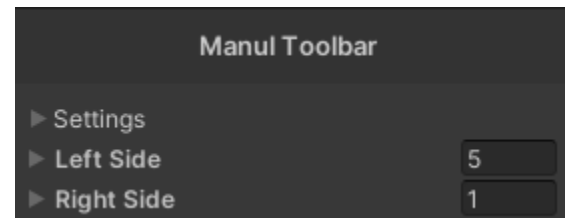


You can also open this asset by using the menu item: **Tools / Manul Toolbar / Open Manul Toolbar Settings**.



(If you are using a version of Unity older than 2021.3.1f1, that will select the asset and show its properties in the *Inspector* window.)

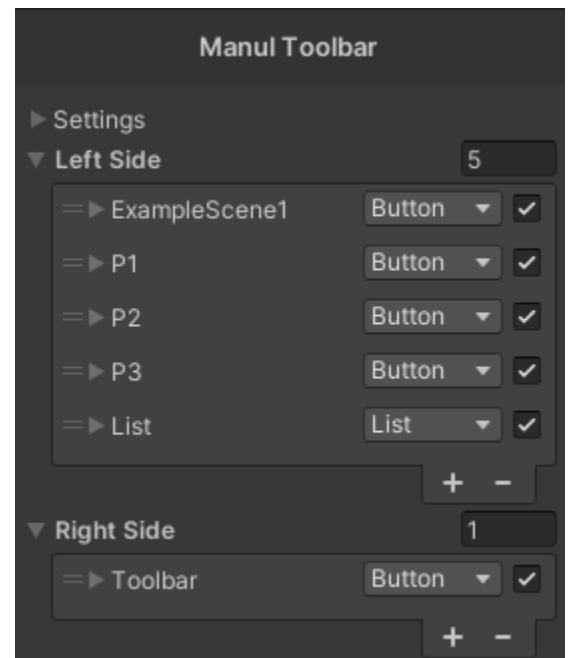
The **Manul Toolbar Settings** asset is divided into three sections. You can left-click on the heading of any section to expand it. **Settings** are general tool settings and we'll get back to them soon.



First, take a look at the **Left Side** and **Right Side** lists. Expand both of them; you'll see that their elements correspond with the items that are visible on the upper toolbar.

Each list's element is an entry that specifies the properties of the item that will appear on the left or right of the **Play, Pause, Step** buttons - depending on which list (**Left Side** or **Right Side**) the item is located on.

The buttons on the toolbar are arranged from left to right - in the exact same order as the items on the list are arranged from the top to the bottom. You can reorder the elements on the list to change the order in which the items (e.g. buttons) are arranged on the toolbar.

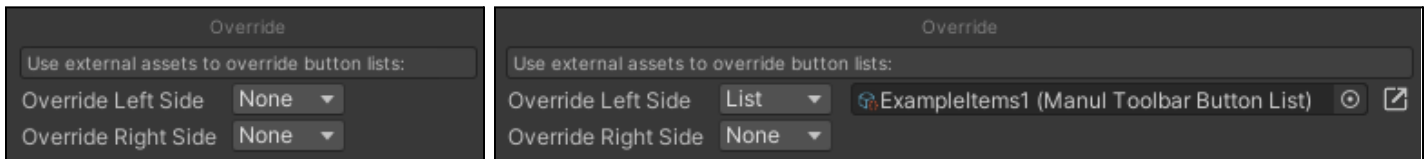


But these two lists are not the only places where you can create items for the upper toolbar. You can override the left or right side. See the next section to learn how to do it.

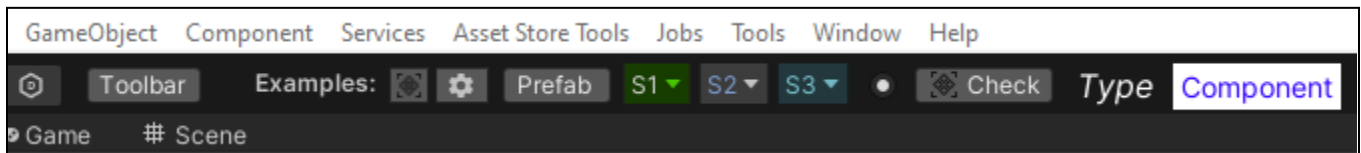
## 2.4 Override Toolbar Side

You have to learn how to override a toolbar side with an asset list because all the example buttons, popups, actions etc. in this documentation refers to items that are located in those assets. Overriding is thoroughly described in the [4.4 Override](#) section, but here we'll learn the basics.

First, expand the **Settings** section by left-clicking on it, and search for the last part with the **Override** header. Find the **Override Left Side** label, set the dropdown next to it from **None** to **List**, and then, in the **Examples** folder, find an asset called **ExampleItems1**. Drag this asset into the field next to the dropdown (images below shows the settings before and after the changes).



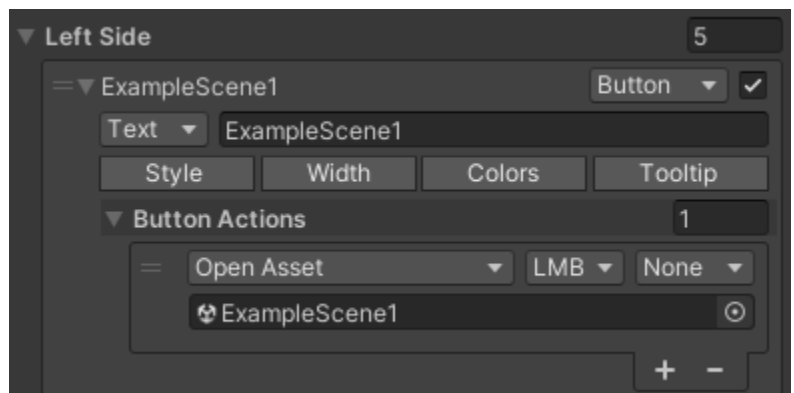
You will notice that your previous buttons and popup on the left side of the toolbar have disappeared, and a new set of items has appeared in their place. They look quite random but don't worry. These are just examples, and **ExampleItems1** asset is one of many example assets that this documentation refers to.



Now you know how to override a toolbar side with a **Manul Toolbar Button List** asset. You can set the **Override Left Side** dropdown back from **List** to **None** option. Let's get back to the items.

## 2.5 Manual Way

We have added buttons and a popup using the shortcut way but the manual way gives you a lot more customization options, actions types, and item types. Expand any element on the **Left Side** or **Right Side** list to see some of those (e.g. **ExampleScene1** in the image on the right).



This subsection will be a guidepost that will tell you where you can find information about certain topics in this document.



First of all, we won't describe how to configure an item from scratch using the manual way, because this topic is described in detail in the [3. Adding New Item](#) section. It is also a good idea to learn about the tool's settings by reading the [4. Settings](#) section.

You can add icons to your items, and also change style, width, and colors. You can even add a tooltip. All these things are described in the [5. Item Visuals](#) section.

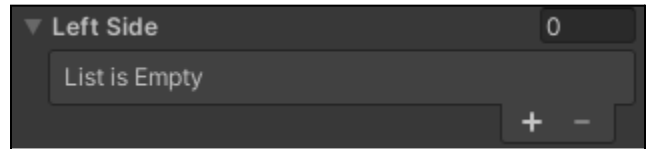
**Button** and **List** are not the only types of items that you can add to the upper toolbar. You can also add **Toggles**, **Labels**, **Popups**, **Sliders** etc. Section [6. Item Types](#) thoroughly describes all available item types and how to use and configure them.

The same goes for action types. Opening and selecting assets are only two of the 12 available action types. You can also use actions to invoke methods and events, to open folders or properties windows, or even to execute menu items. See the [7. Action Types](#) section to learn about all of them.

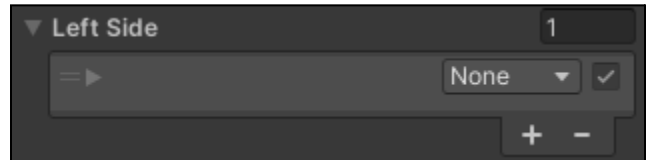
### 3. Adding New Item

This section will show you how to add new items to the toolbar (without using shortcuts) and how to modify them. First, left-click on the **Toolbar** button on the upper toolbar or select the **Tools / Manul Toolbar / Open Manul Toolbar Settings** to open a properties window of the **Manul Toolbar Settings** asset.

Then add a new element to the **Left Side** list (to add an element, click the **plus icon** or change the list **Size** to 1, assuming that this list is empty).

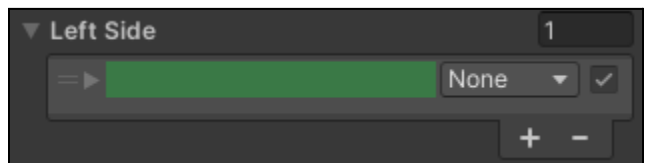


After you add a new element to the list, the new item will not appear on the toolbar immediately. To make it happen, you need to make several steps described in the sections below.

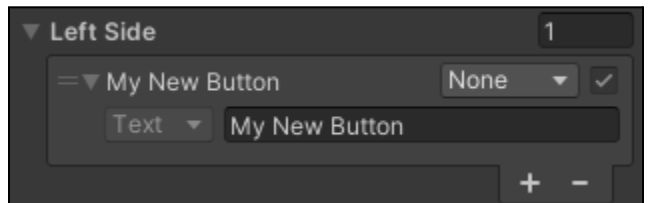


#### 3.1 Type Item Label

Each element on the list begins with a header (green box in the image), and in each newly added element this header is empty. The header is also a foldout button. Left-click on it to expand the element options.



Then enter some text in the field in the second line, as shown in the image ("My New Button"). The text you enter will appear as the header of the element. This text will also (optionally) appear on the toolbar item that we are creating (you will see that in the next section).

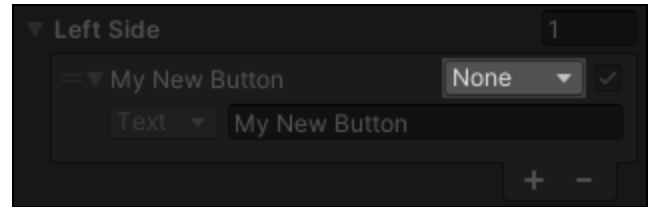


The dropdown menu on the left of the text field allows you to specify what the toolbar item will display: **Text**, **Icon**, **Both**, or **None**. More information on this topic can be found in [Section 5. Item Visuals](#). The dropdown is disabled because the item type is **None**. We will change that in the next section.

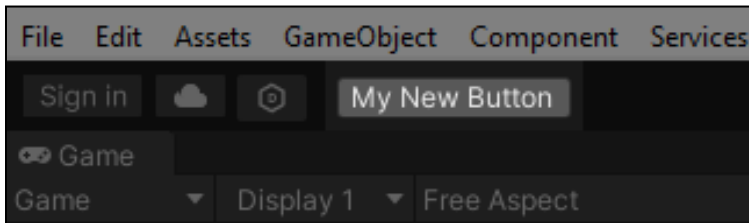
***Tip:** Even if your item displays only an icon, without text, it is still worth entering some text in the text field so that the header of the element is not empty on the list.*

## 3.2 Select Item Type

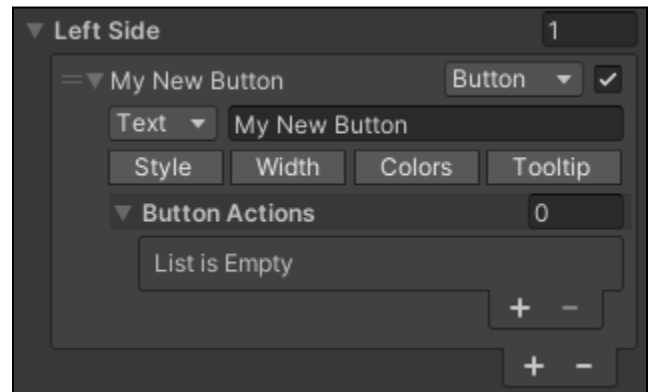
The default type of a newly added element is **None**. Items of the **None** type will not appear on the toolbar. You can change the item type using the dropdown menu shown in the image.



There are several options here. All have been described in the [6. Item Types](#) section. For now, just select the **Button** option. Your new item should appear on the left side of the toolbar (look at the left image below).



You will also notice that some new options have appeared. The set of four toggles under the label field (**Style**, **Width**, **Colors**, **Tooltip**) lets you configure how your new item will look like. You can find more information about this topic in the [5. Item Visuals](#) section.

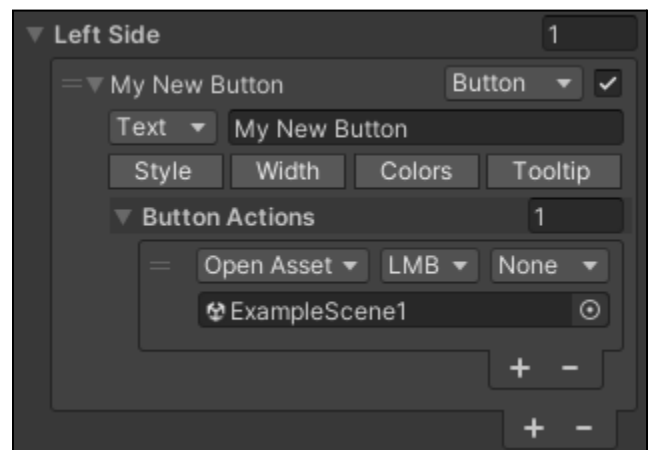


## 3.2 Add Actions To Item

Another thing that has appeared after you change the item type to **Button** is the **Button Actions** list. Here you can define what actions will be performed after the button is clicked. You can find more information about this topic in the [7. Action Types](#) section.

For now just add a new element to the actions list, set the type of the new action to **Open Asset** in the action type popup (the default type is **None**), and then drag and drop any asset file to the field under the action type popup (for example, **ExampleScene1** as shown in the image).

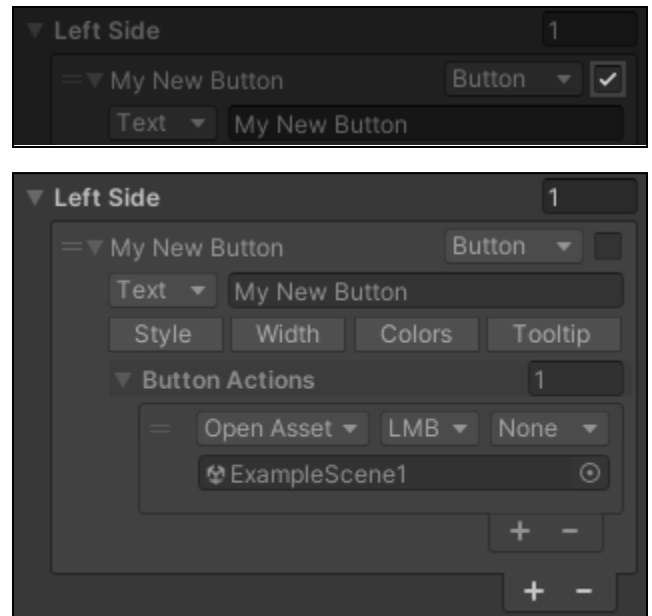
You can test your button now. Left-click on it (on the left side of the upper toolbar) and the asset you have dragged and dropped should be opened.



### 3.3 Toggle Item

There is a toggle next to the item type popup that you can use to enable or disable any item. Items of the **None** type cannot be disabled, and any new added item is enabled by default. If you disable this toggle, two things will happen:

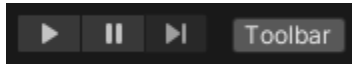
- The disabled item will not be drawn on the upper toolbar (you will notice how your new button has disappeared).
- The color of the item options will be semi-transparent (you can change the color of the disabled buttons in **Settings: Disabled Color**) as you can see in the image on the left.



**Tip:** As you're working on your project, you will be adding new items and others will no longer be needed. You can, of course, delete the latter, but you can also just disable them. If you need them again, simply enable them. It will take you less time than creating them from scratch.

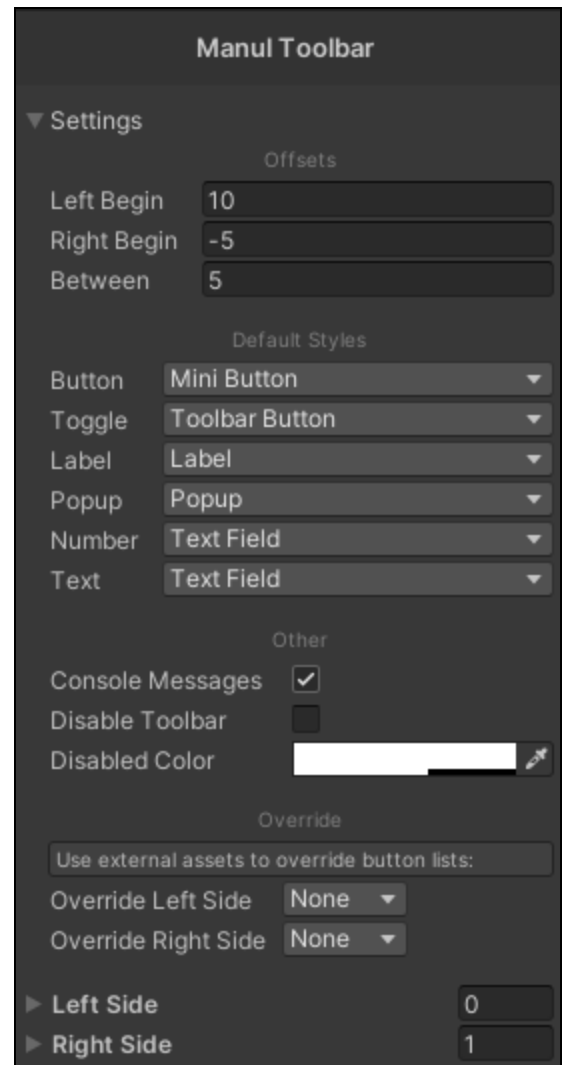
## 4. Settings

The **Settings** section in the **Manul Toolbar Settings** asset allows you to modify several general tool settings. You can access this asset in four ways:



- Left-click on the **Toolbar** button in the upper toolbar, next to the **Play**, **Pause**, **Step** buttons (recommended).
- Use **Tools / Manul Toolbar / Open Manul Toolbar Settings Asset** menu item to open a property window.
- Use **Tools / Manul Toolbar / Select Manul Toolbar Settings Asset** to show asset's properties in the *Inspector* window.
- Manually find and select this asset in the *Project* window. The asset can be found in this default location: **Assets / Liquid Glass Games / Manul Inspector / Resources / Manul Toolbar.asset**.

(In version 1.3 and above, you can find a copy of this asset in the **Assets / Liquid Glass Games / Manul Inspector / Resources** folder. The copy is named **“Manul Toolbar Settings (Default)”** and all its settings are set to default.)

A screenshot of the 'Manul Toolbar' settings window. It has a dark theme. The title bar says 'Manul Toolbar'. Below it is a 'Settings' section with a dropdown arrow. The settings are organized into four sections: 'Offsets' with 'Left Begin' (10), 'Right Begin' (-5), and 'Between' (5); 'Default Styles' with dropdowns for 'Button' (Mini Button), 'Toggle' (Toolbar Button), 'Label' (Label), 'Popup' (Popup), 'Number' (Text Field), and 'Text' (Text Field); 'Other' with 'Console Messages' (checked), 'Disable Toolbar' (unchecked), and 'Disabled Color' (a color picker showing white); and 'Override' with a button 'Use external assets to override button lists:', 'Override Left Side' (None), 'Override Right Side' (None), and two buttons at the bottom: 'Left Side' (0) and 'Right Side' (1).

The **Settings** section is divided into four parts which are described in the next sections:

- **Offsets**
- **Default Styles**
- **Other**
- **Override**

## 4.1 Offsets

**Offsets** are different distances on the toolbar.

- **Left Begin** - the horizontal distance between the last Unity button on the left (**Version Control**) and the beginning of the left row with items. It is marked in green in the image below.
- **Right Begin** - the horizontal distance between the Unity button (**Step**) and the beginning of the right row with items. It is marked in blue in the image below.
- **Between** - the horizontal distance between items on the toolbar. The value applies to the items on both the left and right rows. These distances are marked in cyan in the images below.



## 4.2 Default Styles

**Default Styles** define the default styles for most of the item types. The default style will be used if, either the **Style** option is disabled, or - if it is enabled - the style for the button is set to **Default**. More on this topic can be found in Section [5.2.1 Style](#).

- **Button** - default style for the **Button** type (*Mini Button* style).
- **Toggle** - default style for the **Toggle** type (*Toolbar Button* style).
- **Label** - default style for the **Label** type (*Label* style).
- **Popup** - default style for the **Popup** type (*Popup* style).
- **Number** - default style for the **Number** type (*Text Field* style).
- **Text** - default style for the **Text** type (*Text Field* style).

You cannot change the style for some item types (**None**, **List**, **Other**), that is why there is no default style for them.

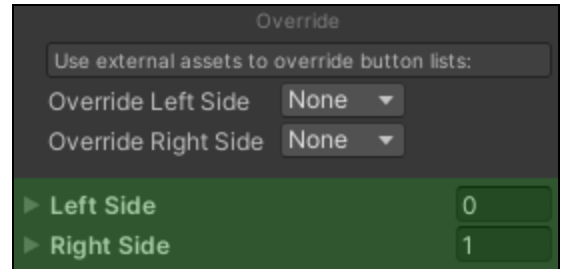
## 4.3 Other

**Other** is the section which contains three unrelated options.

- **Console Messages** - enable this option to enable log, warning, and error messages in the *Console* window. All messages from **Manul Toolbar** have the “[**MANUL Toolbar message**]” prefix.
- **Disable Toolbar** - check this option to disable the display of items on the toolbar.
- **Disabled Color** - this color field determines the tint that will be applied to the options of an element if it is disabled on the items list. See Section [3.3 Toggle Item](#) for more information.

## 4.4 Override

By default, the items that appear on the left side of the toolbar are defined in the **Left Side** list (under the **Settings** section) and the buttons that appear on the right side of the toolbar are defined in the **Right Side** list (both lists are marked with green color in the image on the right). You can override this behavior in two ways:



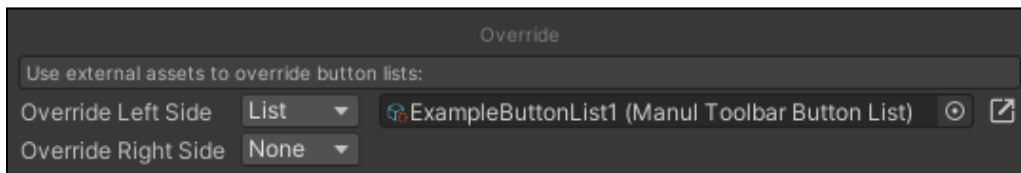
- Use an external asset which contains a list of items' settings.
- Use an asset with a set of lists of items, and then use a popup to switch between these lists.

### 4.4.1 Use Button List Asset

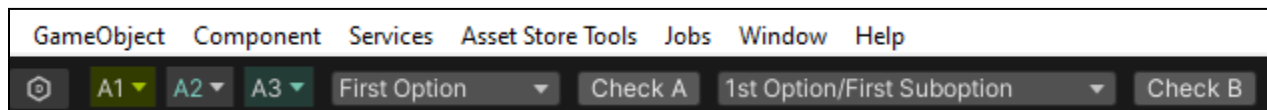
**Important!** All example items that this documentation refers to are stored in external assets. That is why this subsection is highly recommended or even mandatory to read because it shows you how to override the toolbar sides with those assets.

If you want to override one side of the toolbar with an external asset, use the **Manul Toolbar Button List** asset which contains one list with the items' settings. You can create this type of asset by right-clicking in the *Project* window and selecting **Manul Tools > Manul Toolbar > Manul Toolbar Button List**.

There are also some example assets of this type in the **Examples** folder (*ExampleItems1*, *ExampleItems2*, *ExampleItems3*, *ExampleButtonList1*, *ExampleButtonList2*, *ExampleButtonList3*). You can use one of them to test this functionality.



Use the popup next to the **Override Left Side** label and change it from **None** to **List** option. Then drag and drop one of the before-mentioned assets in the field next to the **Override Left Side**. You can also use the **open icon** at the end of the row to open a new window with the properties of the selected asset.

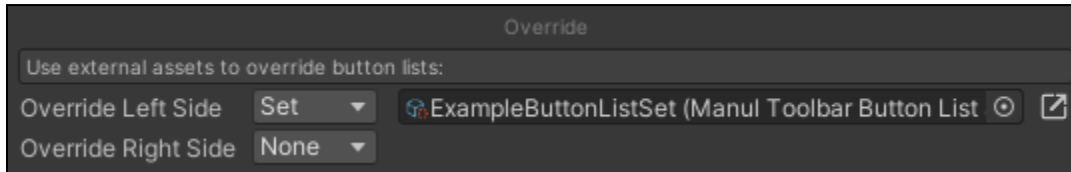


You should notice that some buttons and popups have appeared on the left side of the toolbar (**A1**, **A2**, **A3**, **First Option**, etc.). These buttons and popups are defined in the **ExampleButtonList1** asset. If you want to override the right side of the toolbar, simply create a new asset and do the rest with the right side exactly in the same way you did it with the left side.

#### 4.4.2 Use Button List Set Asset

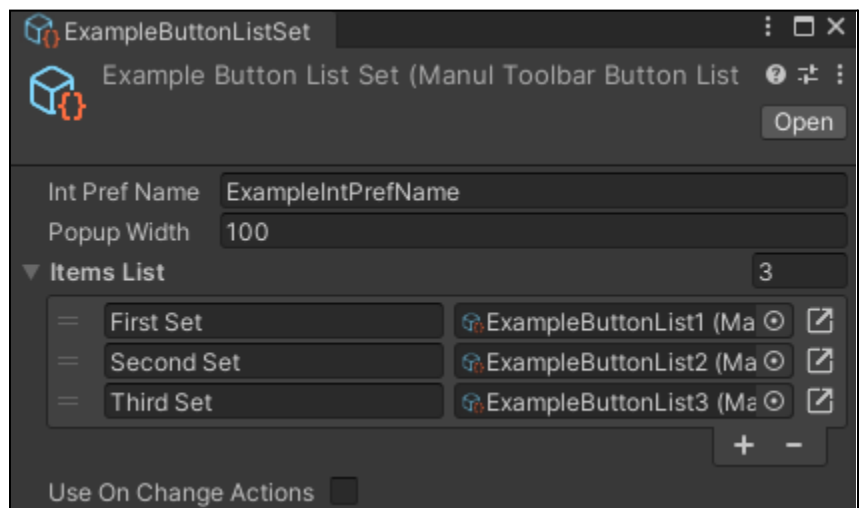
If you want to have a set of lists of items and switch between those lists using a popup to show a different item group each time, use the **Manul Toolbar Button List Set** asset. You can create this type of asset by right-clicking in the *Project* window and selecting **Manul Tools > Manul Toolbar > Manul Toolbar Button List Set**.

There is also an example asset of this type in the **Examples** folder, named **ExampleButtonListSet**. You can use it to test this functionality.



Use the popup next to the **Override Left Side** label and change it to **Set** option. Then drag and drop the before-mentioned asset in the field next to the **Override Left Side**. Left-click on the **open icon** at the end of the row to open a new window with the properties of this asset. You will see two fields and a list.

To define which list of items should be shown, **Manul Toolbar** uses an *editor pref* of the *int* type. Type a name in the **Int Pref Name** field (see the example name **“ExampleIntPrefName”** in the image on the right). To change the value of the pref (and thus, changing the item set that it is being shown), you will use a popup. This popup is located at the beginning of the row with items, which you can see in the images on the next page.

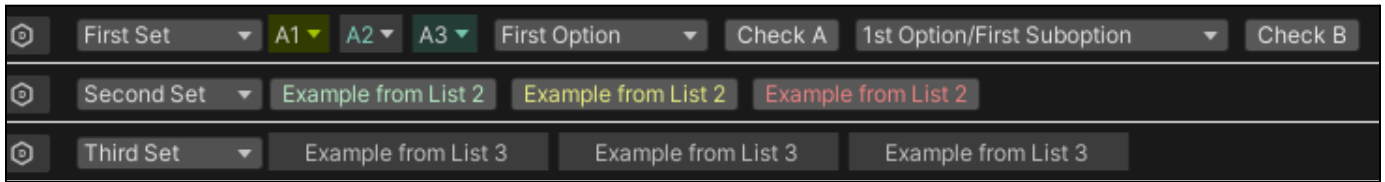


The popup has a fixed width that you can change in the **Popup Width** field (or just leave the default value of **100**).

The next thing to do is to create a list of entries in the **Items List**. Each element in the list corresponds to one option that can be chosen by using the popup. It has two fields and a button. The first field is for putting a name in it, for example, the second item has a **“Second Set”** name. The second field is for **Manul Toolbar Button List** asset, which is described in the previous section ([4.4.1 Use Button List Asset](#)). You can also use the **open icon button** at the end of the row to open a new window with the properties of this asset.



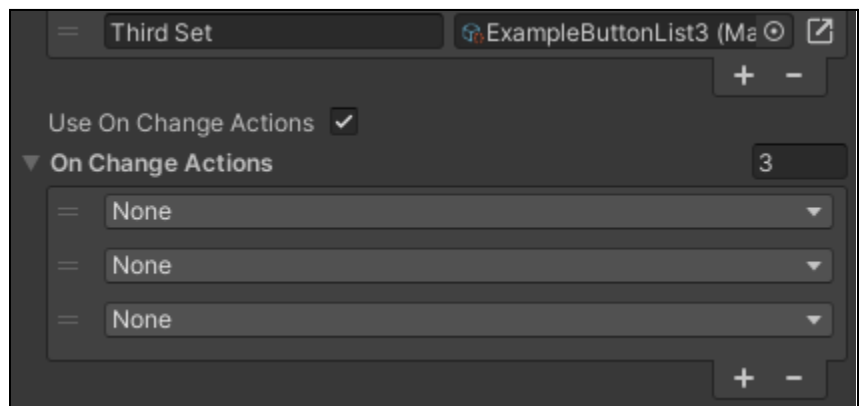
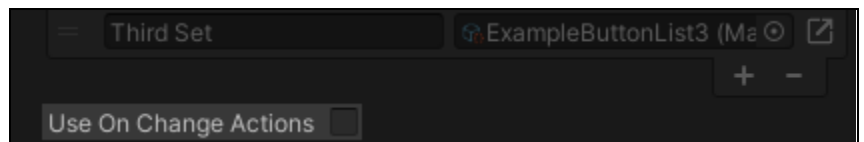
So, for example, if the user chooses the item with the **“Third Set”** name, **Manul Toolbar** will show the items from the list that is defined in the **ExampleButtonList3** asset. Look at the images below to see how changing the popup at the beginning of the row changes which list of items is shown.



**Tip:** This is extremely useful when working with a team and using some sort of version control. Changing the **editor prefs** will not make any changes to the repository, so each member of the team can work with their own list of items (i.e. have a different value of the **editor pref**).

You can also add actions which will be performed each time the value of the popup is changed. Look at the **Use On Change Actions** toggle. By default, it is disabled.

When you enable it, the **On Change Actions** list will appear under the toggle. Populate this list with elements. For each element you need to specify the type of the action, and, depending on the type, you need to fill text, object, or other fields that will appear.



You can find more information about actions in the [7. Action Types](#) section.

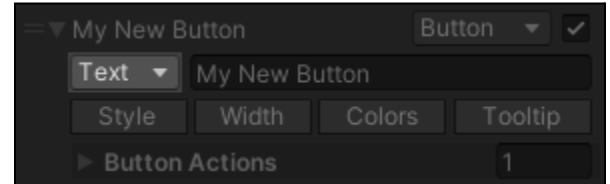
## 5. Item Visuals

There are many options regarding the visual layer of the item. In this section we will describe each of them.

All example items which this section refers to are in the **ExampleItems1** asset in the **Examples** folder. See the [4.4.1 Use Button List Asset](#) section to learn how to override a toolbar side with items in this asset.

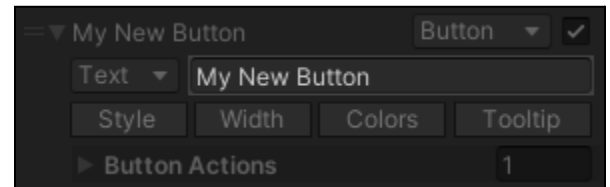
### 5.1 Label Type

This dropdown allows you to specify what the item's label will consist of: **Text**, **Icon**, **Both**, or **None**. The way that label looks also depends on the selected style (more about this in the [5.2.1 Style](#) section). Please note that some item types (**Popup**, **Slider**, **Number**, **Text**, **List**) will not show the label.



#### 5.1.1 Text

The item will display only the text entered in the field next to the dropdown. For example, you can see a label **“My New Button”** in the image to the right. Additional information about the **Text** type can be found in the [3.1 Type Item Label](#) section.

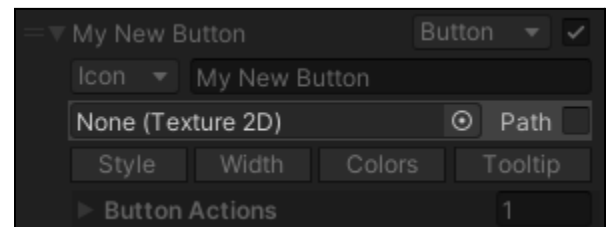


#### 5.1.2 Icon

The item will display only the icon. There are two ways that you can use to provide an image for the item's icon. It depends on whether the **Path** toggle is enabled or not.

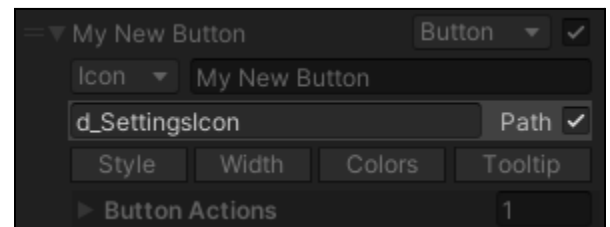
If the **Path** toggle is disabled, the image of the icon must be dragged and dropped in the **Texture2D** field next to the toggle.

*Tip:* [Google Fonts](#) is a nice source of free icon images, which you can use even in commercial projects.

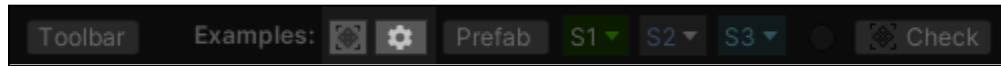


If you want to use built-in editor icons, enable the **Path** toggle and type a path to the icon in the text field next to the toggle (for example, **“d\_SettingsIcon”**).

*Tip:* You can find a list containing all editor icons [here](#).

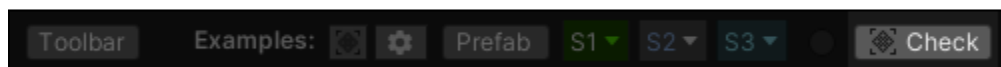


In the **ExampleItems1** asset you can find examples of items which show only an icon. The item named **“Mouse & Keyboard Buttons”** (fourth element on the list) uses an image file while the item named **“Project Settings”** (fifth element on the list) uses a built-in editor gear icon. Both are shown in the image below.



### 5.1.3 Both

The item will display both text and icon. You enter the text and provide the icon in the same ways that were described in two previous sections. In the **ExampleItems1** asset you can find an example of an item that displays text and icon. It's called **“Check”** (11th element on the list) and is visible in the image below.



### 5.1.3 None

The item will not display anything. It's useful for creating empty spaces. In the **ExampleItems1** asset you can find an example of an item that does not display anything. It's called **“Empty Space”** (second element on the list).

## 5.2 Option Toggles



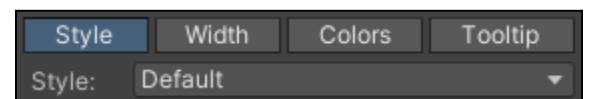
The next row has four toggles: **Style**, **Width**, **Colors**, and **Tooltip**. Each of them is responsible for different functionalities, which are described in the following sections. Please note that for a given functionality to work, the corresponding toggle must be enabled (highlighted).

For example, if you enable the **Colors** toggle and make changes to the colors, you will notice that the item on the toolbar has changed colors. But if you turn off the **Colors** toggle, the colors of the item will return to the default ones (even though the colors will still be changed in the item properties).

### 5.2.1 Style

If the **Style** toggle is disabled, the item will use one of the default styles, set separately in the **Default Styles** group in the **Settings** (see the [4.2 Default Styles](#) section for more information).

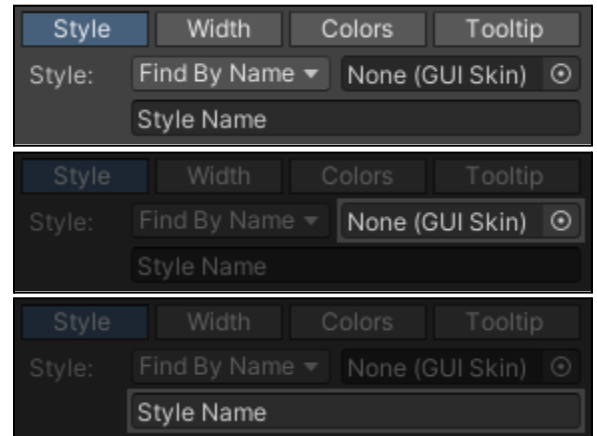
Enable the **Style** toggle to be able to set a style for an item other than the default one. A new row with a dropdown labeled **“Style:”** will appear. This dropdown lets you choose a style for the item.



You have following options here:

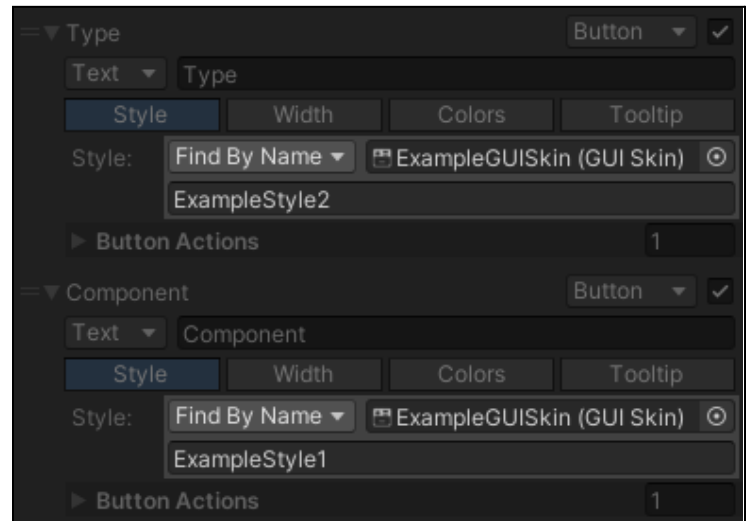
- **Default** - the item will use the default style which can be set in the **Default Styles** group in the **Settings** (see the [4.2 Default Styles](#) section for more information).
- **Find By Name** - after selecting this option two new fields will appear, one on the right of the dropdown and the second below it.

You can drag and drop your own *GUI Skin* asset in the first field. If you do not do this, the style will be searched for in a standard skin - *DarkSkin* or *LightSkin* - depending on which *Editor Theme* you have selected in Unity editor. Enter the style name in the second field ("**Style Name**" in the image).



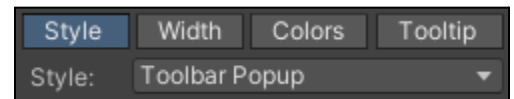
In the **ExampleItems1** asset you can find examples of items that use custom skin and custom styles. They are named "**Type**" (12th element on the list) and "**Component**" (13th element on the list).

The **ExampleGUISkin** skin, dragged and dropped into the field next to the style dropdown (in both elements), contains two **Custom Styles**: "**ExampleStyle2**" (used by the **Type** button) and "**ExampleStyle1**" (used by the **Component** button).



More information about *GUISkin* can be found [here](#).

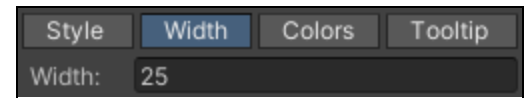
- The remaining options of the dropdown are various Unity styles. Select one of them to change the style in which the item is presented. For example, the **S1**, **S2**, and **S3** buttons use the *Toolbar Popup* style.



Remember that some styles have a specific purpose and may not look good or display text or icons. Moreover, the *Icon Button* and *Selection Rect* styles only work in Unity version **2021.3.1f1** or higher. Please also note that the **Style** toggle is disabled for **Slider**, **Other**, **Number**, and **Text** item types. Therefore, the style for those types cannot be changed (but you can change the default style for **Number** and **Text** styles).

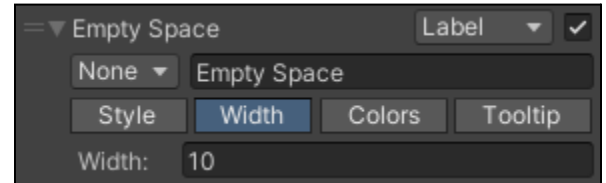
### 5.2.2 Width

If the **Width** toggle is disabled, the width of the item will be determined automatically, based on the length of the text the item displays. After enabling the toggle a new row labeled “**Width:**” will appear. It contains a field where you can enter the width of the item.



If the button displays only the icon or nothing, or the item is **Popup** or **List** type, then the **Width** toggle should definitely be enabled.

In the **ExampleItems1** asset you can find examples of items with fixed width. They are named “**Mouse & Keyboard Buttons**” (2nd element on the list) and “**Empty Space**” (3rd element on the list). The latter is shown in the image on the right.



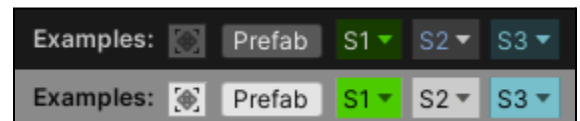
### 5.2.3 Colors

If the **Colors** toggle is disabled, the item colors will have no tint. Enable the **Colors** toggle to modify the tint. After enabling the **Colors** toggle, three color fields will appear in the row below:



- **First color:** modifies the tint of the background and content (text) colors. Equivalent of *GUI.color*. Example: **S1** button (7th element on the list in the **ExampleItems1** asset).
- **Second color:** modifies only the tint of the content (text) color. Equivalent of *GUI.contentColor*. Example: **S2** button (8th element on the list in the **ExampleItems1** asset).
- **Third color:** modifies only the tint of the background color. Equivalent of *GUI.backgroundColor*. Example: **S3** button (9th element on the list in the **ExampleItems1** asset).

Remember that you tint a color, not change it. This is equivalent to multiplying color by color. For example, in the *Light* skin in Unity the text is black (so the color value is zero), therefore you will not be able to change its tint using the **Colors** option (because multiplying by zero gives you zero).

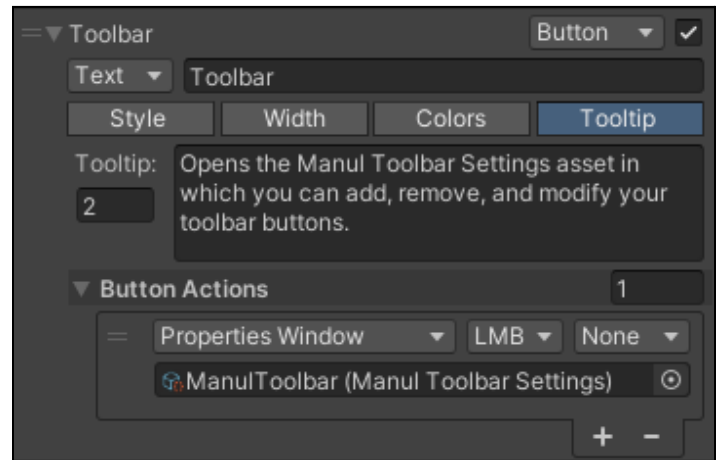
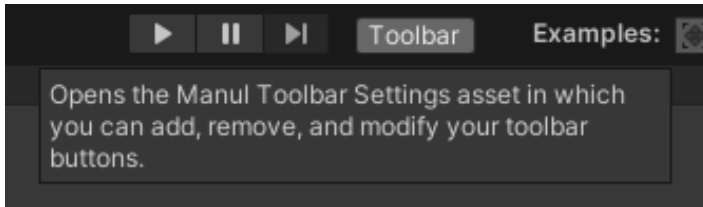


*Tip: The original text color can be changed by using **GUIStyles**. More information can be found [here](#).*

## 5.2.4 Tooltip

To see additional information about an item when you hover the cursor over it, enable the **Tooltip** toggle.

A large text field will appear below, in which you can enter the tooltip text, and a smaller one will appear on the left (below **“Tooltip:”** label), specifying the height (in lines) of the first field.



So if you want to enter an extremely long tooltip, change the number of lines to, for example, 10 or more. The field cannot be smaller than 2 lines.

In the **ExampleItems1** asset you can find examples of items using tooltips. They are named **“Toolbar”** (1st element on the list) and **“S3”** (9th element on the list).

## 6. Item Types

After creating a new item, which is described in the [3. Adding New Item](#) section, you must select its type. There are nine types. Some of them use actions or *editor prefs*, others do not. Some can be fully configured, others have predefined settings that cannot be changed. This section describes all of the item types.

Five of them (**Toggle**, **Popup**, **Slider**, **Number**, and **Text**) were grouped into the [6.4 Value Fields](#) subsection because they share overall behavior and some options.

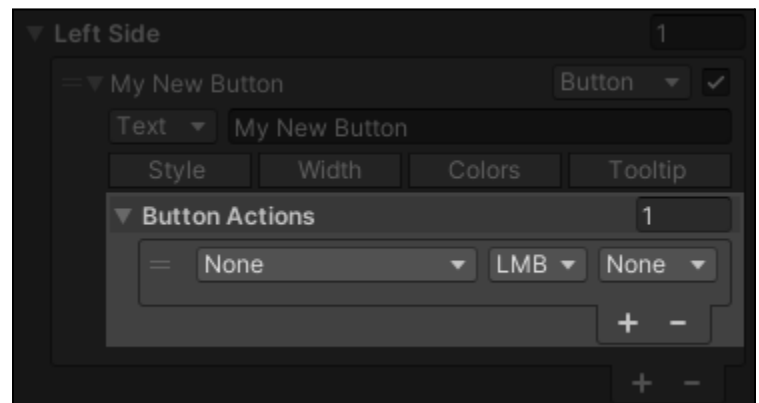
### 6.1 Button

#### 6.1.1 Adding Button Actions

If you select the **Button** type, a list of **Button Actions** will appear. All of the actions from this list will be performed after clicking the toolbar button (of course, you can attach more than one action to a single button). Each action needs to be configured by doing two things (to see this, add at least one element to the **Button Actions** list).

First, determine which mouse (and optionally keyboard) buttons will activate the action (you will find out more about this in the next section).

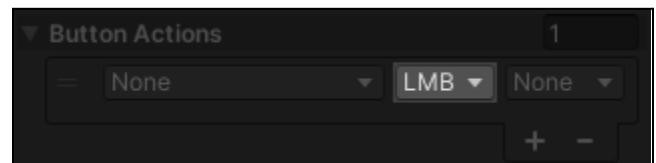
Secondly, you need to specify the type of the action, and, depending on the type, fill text, object, or other fields that will appear (you will find more about this in the [7. Action Types](#) section).



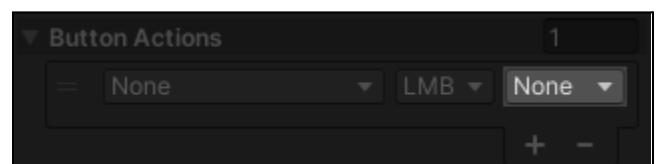
#### 6.1.2 Mouse and Keyboard Buttons

In the first line of the element that defines the action, two dropdowns are responsible for assigning mouse and keyboard buttons to perform this action.

The middle one determines the mouse button (**LMB** - **Left Mouse Button**, **RMB** - **Right Mouse Button**, **MMB** - **Middle Mouse Button**) that must be pressed after hovering the cursor over the button for the action to be performed.

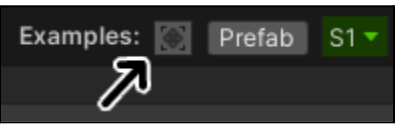


The right dropdown specifies an optional key that must be held down during the click for the action to take place. By default, it is **None**, which means you don't have to hold down any keys, but you can change it to **Ctrl**, **Shift**, or **Alt**.



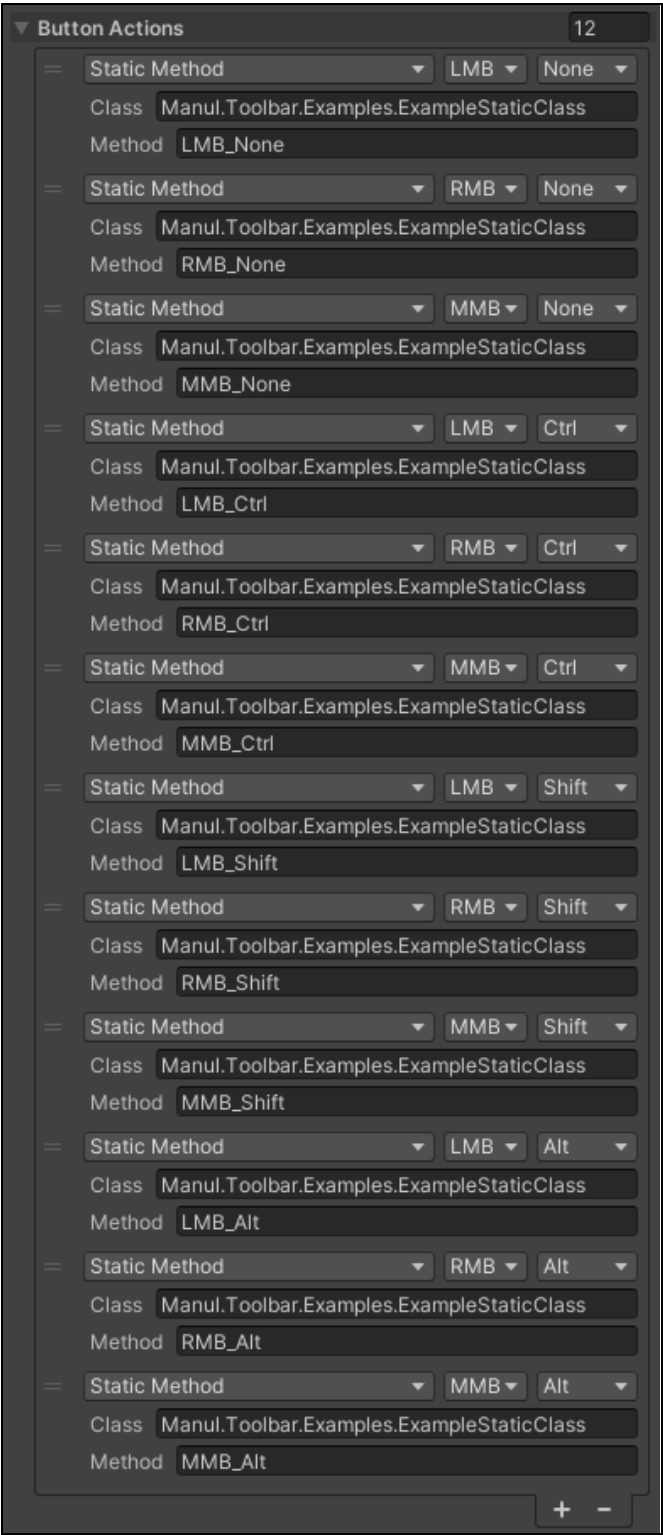
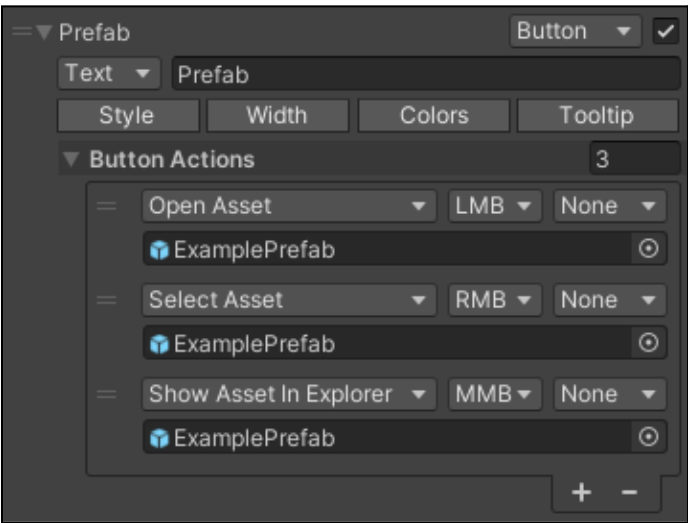
By using the mouse and keyboard buttons, you can create as many as twelve actions that can be triggered in various ways - and one button on the toolbar contains them all!

Example buttons, which this section refers to, are in the **ExampleItems1** asset in the **Examples** folder. See Section [4.4.1 Use Button List Asset](#) to learn how to override a toolbar side with items in this asset.



This is shown by the **first example button**, pointed by the arrow in the image to the left.

Click on this button with different mouse buttons (**left, right, middle**) while, optionally, holding down the **Ctrl, Alt**, or **Shift** keys to see various messages appear in the *Console* window. This button is called **“Mouse & Keyboard Buttons”** in the **ExampleItems1** asset; you can see its twelve configured actions in the image to the right.



A good practical example of using several actions assigned to one button is an example button called **“Prefab”**. Left-click on it to open the prefab in the prefab window, right-click on it to select this prefab in the *Project* window, and middle-click on it to open the folder containing this prefab in explorer.

You can find more about action types used in these examples (**Static Method, Open Asset, Select Asset, Show Asset In Explorer**) in the [7. Action Types](#) section.

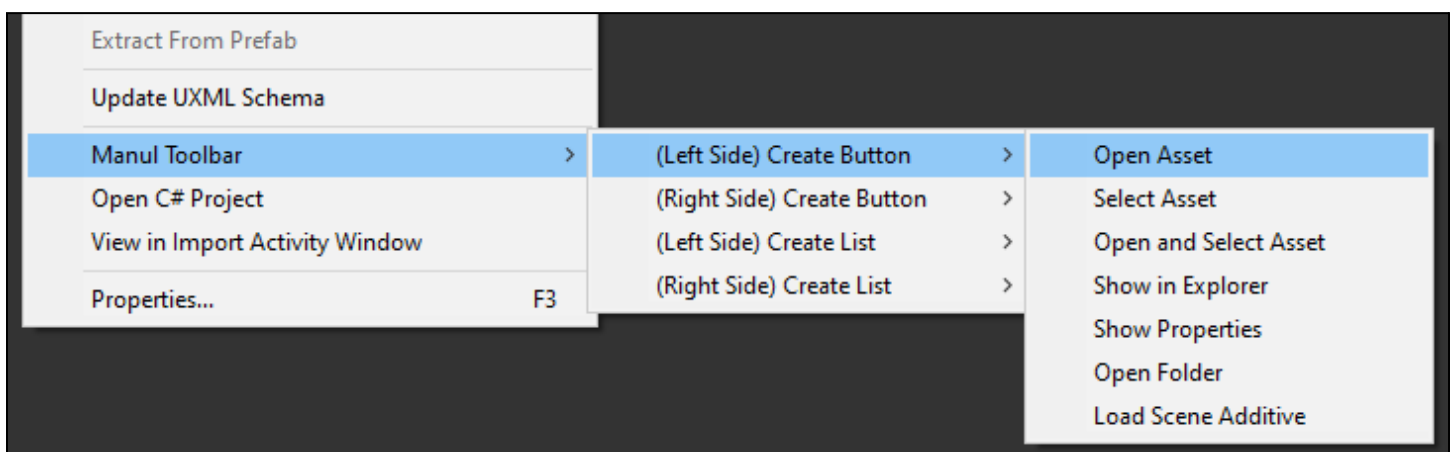


### 6.1.3 Shortcut: Create Button Menu Item

You can quickly create a **Button** corresponding with any asset in the *Project* window by using menu items. If you select more than one asset (by holding down **Shift** or **Ctrl** keys), this shortcut will create a button for each selected asset, in the same order in which you were selecting those assets.

To use this shortcut, right-click on the asset, select **Manul Toolbar**, and choose on which side of the toolbar your new button will appear (**(Left Side) Create Button** or **(Right Side) Create Button**). Lastly, select an item with the name of the action type (e.g. **Open Asset**).

When using the **Open Folder** action, do not right-click on the folder but right-click on any file that is inside this folder.



All these menu items will create a **Button** item and add its settings to the left or right side list of elements (in the **Manul Toolbar Settings** asset).

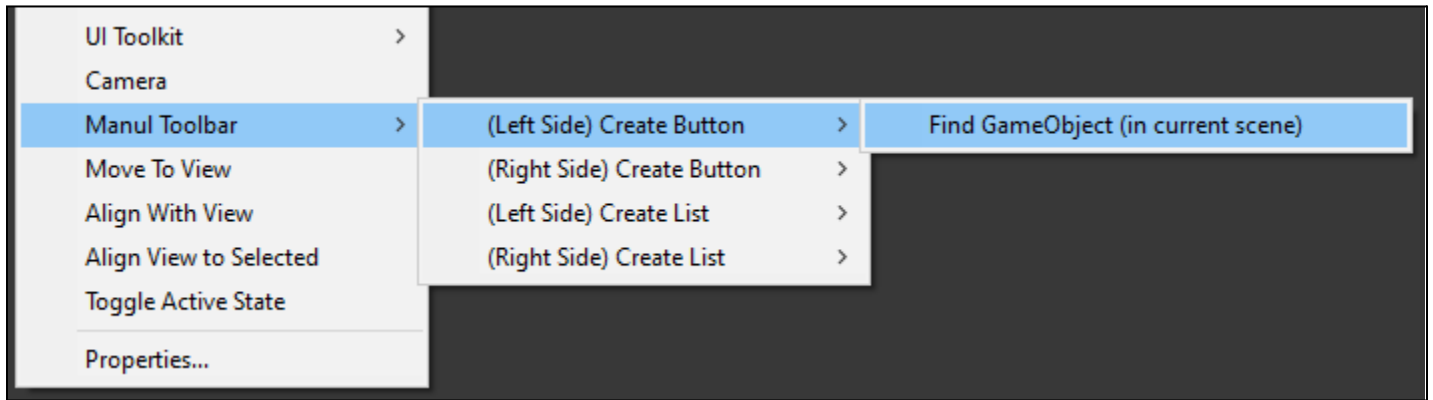
If you have overridden a side with a **Manul Toolbar Buttons List** asset, your new **Button** item will be added to this asset (see the [4.4.1 Use Button List Asset](#) section for details).

If you have overridden a side with a **Manul Toolbar Buttons List Set** asset, your new **Button** item will be added to the list asset that is currently selected by the popup (see the [4.4.2 Use Button List Set Asset](#) section for details).

Each item will create a button which will have one action, invoked with a left-click, but there is one exception. **Open and Select Asset** will create a button with two actions: open the asset (invoked with a left-click) and select the asset in the *Project* window (invoked with a right-click).

As you can see, you can use this shortcut to create a button only with those actions, which use an asset (**Open Asset, Select Asset, Show In Explorer, Show Properties, Open Folder, Load Scene Additive**).

There is one exception. If you right-click on any *gameObject* in the **Hierarchy** window, you can create a button with the **Find GameObject** action (look at the image below). Of course you can select more than one *gameObject* to create a **Button** item for each of the selected *gameObjects*.



*Tip: If you want to get rid of the menu items in the Project window, simply comment out all the contents of the `ManulToolbarMenuItems.cs` file.*

#### 6.1.4 Shortcut: Change Button Name

You can quickly change the name of a button by left-clicking on it on the toolbar while holding down the **Ctrl** and **Shift** keys.

For example, **Ctrl + Shift + Left-click** on the **Prefab** button. You will notice that the button will disappear, and a text field with the **OK** button will appear in its place.



Enter the new name of the button (e.g. **"New Button Name"**) and left-click on the **OK** button. Your button will reappear with a new name. Remember that this will only work with the **Button** type. It will not work with other types.

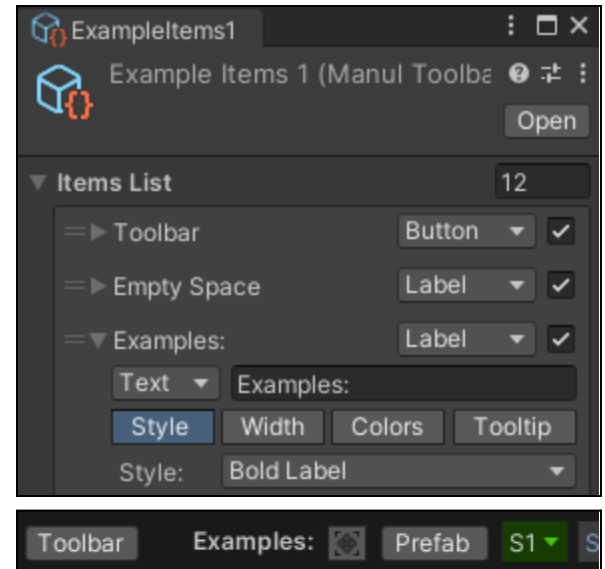
#### 6.1.5 Shortcut: Disable Button

You can quickly disable a button. Just **Ctrl + Shift + Alt + Left-click** on it to do it. Remember that this will only work with the **Button** type. It will not work with other types.

## 6.2 Label

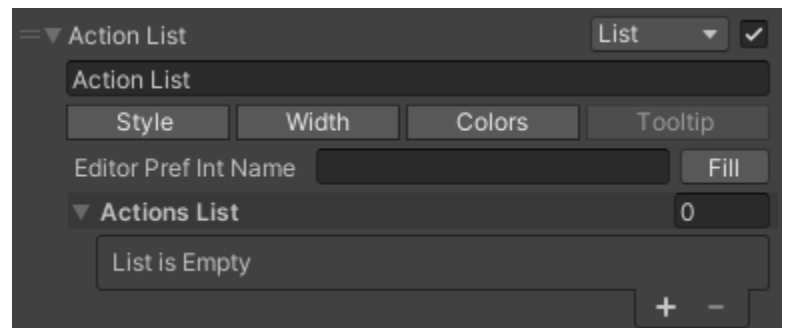
**Label** displays only text, only an icon, or both, and it is for informational purposes only. Nothing will happen if you click on it, but you can change its style, width, colors, or add a tooltip. You can find more information about this topic in the [5. Item Visuals](#) section.

An example label can be found in the **ExampleItems1** asset in the **Examples** folder (see Section [4.4.1 Use Button List Asset](#) to learn how to override a toolbar side with items in this asset). It is the third element on the **Items List**, and it shows the “**Examples:**” word in the upper toolbar.



## 6.3 List

**List** is a special type, you can think of it as a “multi selection button”. It will not show any label on the upper toolbar. Instead, it will create a popup there. Each time you click on the popup and then select one of the options, the actions associated with this option will be performed.

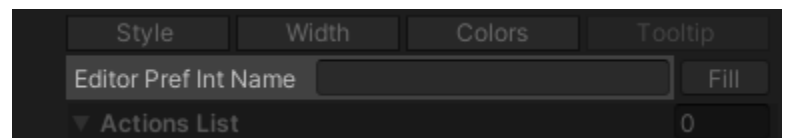


You cannot create a **Tooltip** for the **List** type, and this type doesn’t have a label. If you want to create a label for a **List** item, use another item of the **Label** type. Also, it is recommended to enable the **Width** toggle and type a fixed width for this item (see the [5.2.2 Width](#) section for more information).

### 6.3.1 List Editor Pref

Under the hood the last selected option is stored as an *editor pref* of the *int* type. When you select a new option, its index is compared with the current value of this *pref*. If it differs, action(s) associated with this option are performed, and the previous index is overwritten with the new one.

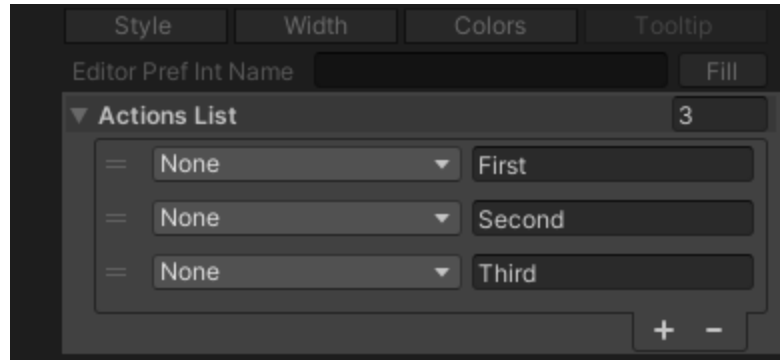
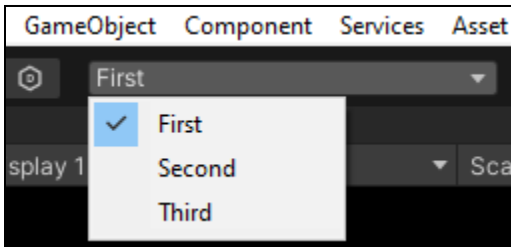
That is why the first thing you need to do is to specify the name of this *pref*. Type it in the **Editor Pref Int Name** field. Make sure it’s unique and it’s not used by any other item.



More information about *editor prefs* can be found [here](#).

### 6.3.2 List Actions

Next, add some elements to the **Actions List**. Each action needs to be configured by doing two things. First, in each element enter a name that will be associated with this action (for example: *“First”*, *“Second”*, *“Third”* names shown in the image on the right). These names will be shown in the popup (see image below).

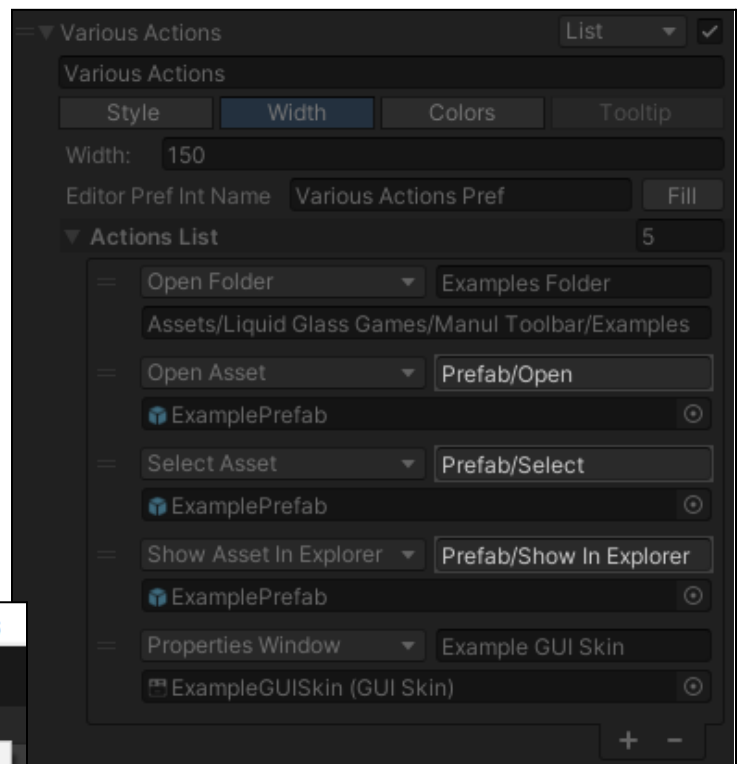
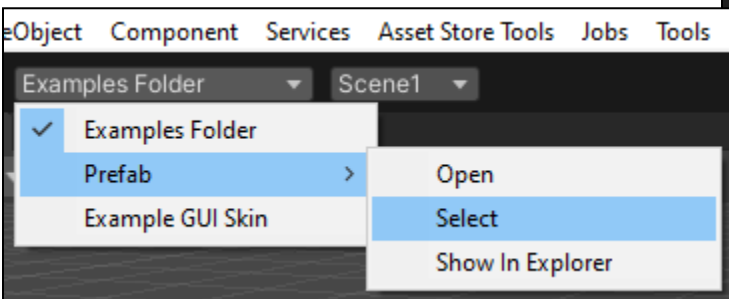


Secondly, in each element you need to specify the type of the action, and, depending on the type, fill text, object, or other fields that will appear (you will find more about this in the [7. Action Types](#) section).

### 6.3.3 Creating Sub-dropdowns

You can also use the *“/”* character (forward slash) to create more complex lists of actions. Each *“/”* character will create another sub-dropdown.

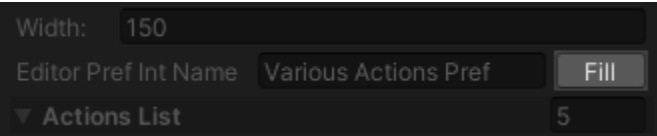
You can see how it works by overriding one of the toolbar sides with **ExmapleItems2** (see the [4.4.1 Use Button List Asset](#) section for more information). Look at the first element in the asset's list, named *“Various Actions”*. Three from five entries with actions, highlighted in the image, have names with *“/”* character.



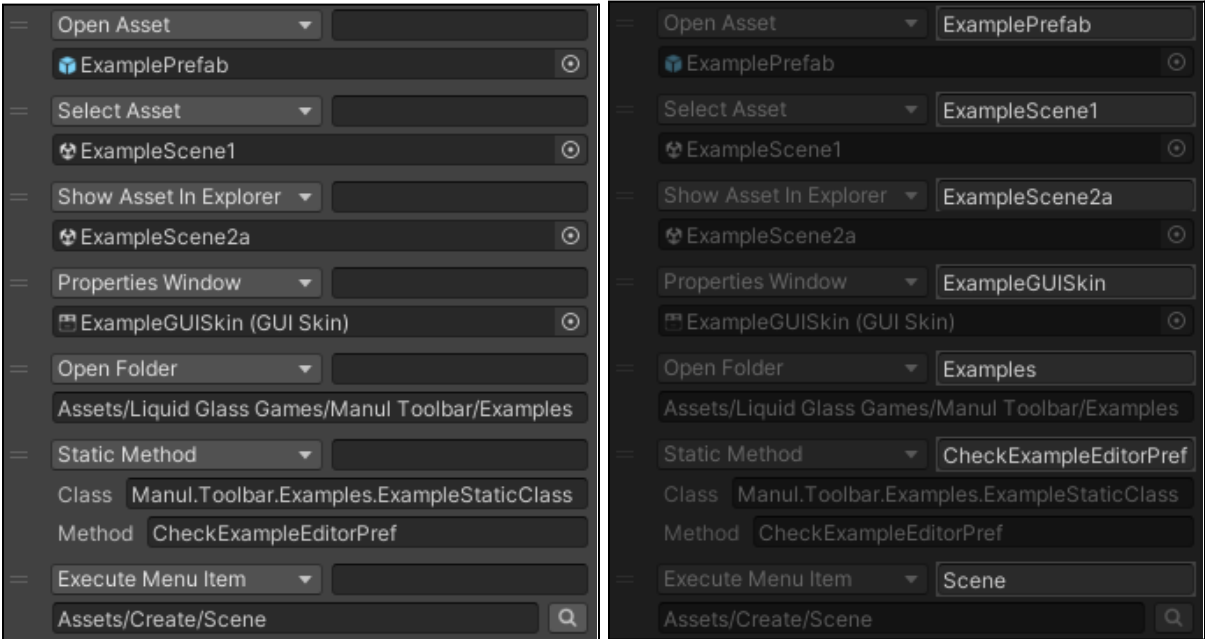
The *“Prefab”* texts before the *“/”* character will form a group (assuming the text in each action is the same). You can see how it works on the upper toolbar in the left image above.

6.3.4 Fill Button

There is another way to quickly set names for options in the **List** item. **Fill** button will create names automatically. The name that is created depends on the type of the action (see the table below). Furthermore, the name will be created only if the field for the name in the action element is empty.



Action Type	Name Creation
<ul style="list-style-type: none"><li>Open Asset</li><li>Select Asset</li><li>Show Asset In Explorer</li><li>Properties Window</li><li>Load Scene Additive</li></ul>	Name of the option will be the same as the name of the asset that was dragged and dropped into the object field.
<ul style="list-style-type: none"><li>Static Method</li><li>Object Of Type Method</li><li>Component Method</li></ul>	Name of the option will be the same as the name of the method.
<ul style="list-style-type: none"><li>Open Folder</li></ul>	Name of the option will be the same as the name of the folder.
<ul style="list-style-type: none"><li>Find GameObject</li></ul>	Name of the option will be the same as the name of a <i>gameObject</i> that needs to be found by this action.
<ul style="list-style-type: none"><li>Execute Menu Item</li></ul>	Name of the option will be the same as the last item from the path (i.e. menu item that is clicked as last).
<ul style="list-style-type: none"><li>Invoke Event</li></ul>	Name will not be created.

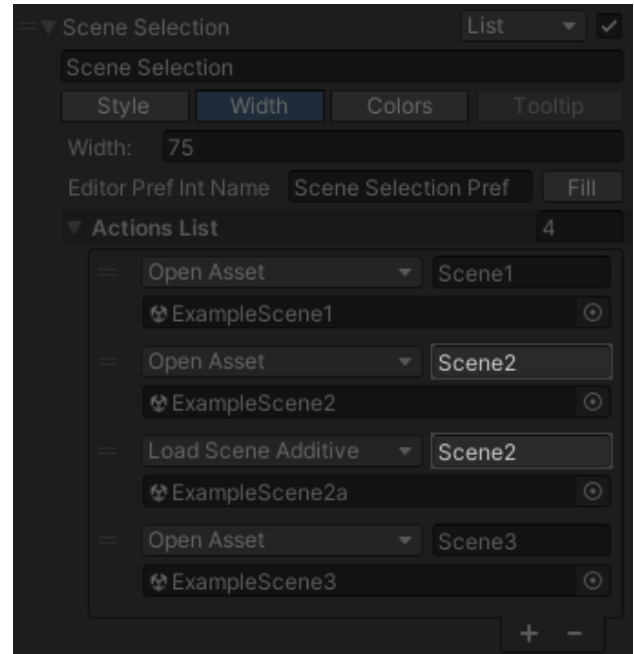
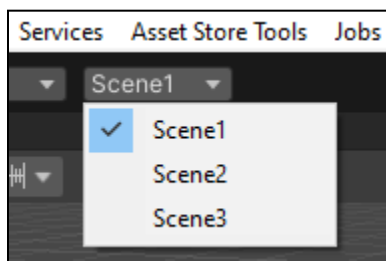


The image on the left shows **Actions List** with empty option names and the image on the right shows names after clicking the **Fill** button.

### 6.3.5 Grouping Actions

If you want to perform more than one action after changing the value of the popup, just use the same name in each action entry.

You can see how it works by overriding one of the toolbar sides with **ExampleItems2** (see the [4.4.1 Use Button List Asset](#) section for more information). Look at the second element in the asset's list, named **"Scene Selection"**. The second and third action have the same name: **"Scene 2"**. If you select this option in the popup, both actions will be performed.

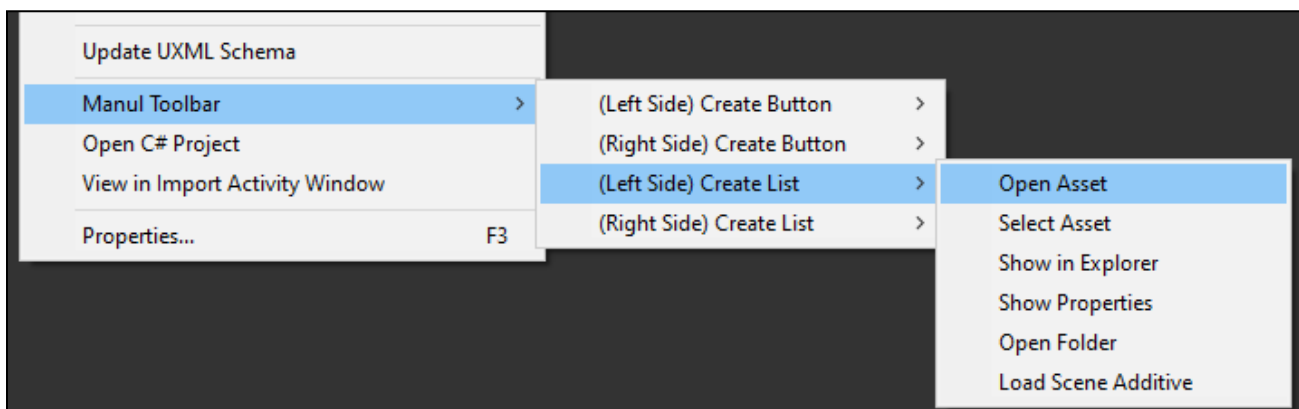


So, even if the **Actions List** has four elements, the popup will show three options (as you can see in the image on the left).

### 6.3.6 Shortcut: Create List Menu Item

You can quickly create a **List** action popup by selecting multiple assets in the *Project* window (while holding down **Shift** or **Ctrl** keys). The order of the actions in the **List** will be the same as the order in which you have selected assets in the *Project* window.

Select one or more assets, right-click on any of them, select **Manul Toolbar**, and choose on which side of the toolbar your new list will appear (**(Left Side) Create List** or **(Right Side) Create List**). Lastly, select an item with the name of the action type (e.g. **Open Asset**).



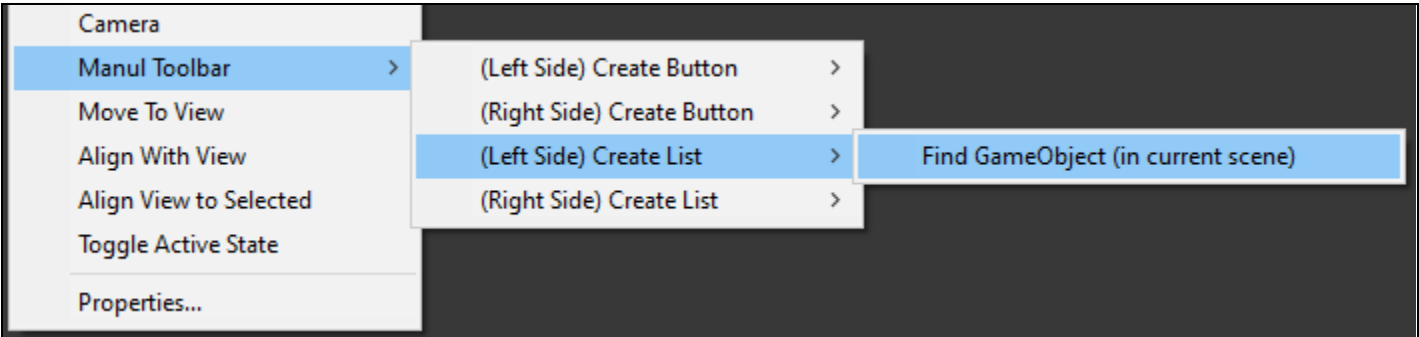
All these menu items will create a **List** item and add its settings to the left or right side list of elements (in the **Manul Toolbar Settings** asset).

If you have overridden a side with a **Manul Toolbar Buttons List** asset, your new **List** item will be added to this asset (see the [4.4.1 Use Button List Asset](#) section for details).

If you have overridden a side with a **Manul Toolbar Buttons List Set** asset, your new **List** item will be added to the list asset that is currently selected by the popup (see the [4.4.2 Use Button List Set Asset](#) section for details).

As you can see, you can use this shortcut to create a **List** item only with those actions which use an asset (**Open Asset, Select Asset, Show In Explorer, Show Properties, Open Folder, Load Scene Additive**).

There is one exception. If you right-click on any *gameObject* in the **Hierarchy** window, you can create a **List** item with the **Find GameObject** action (look at the image below). Of course, you can select more than one *gameObject* to create a **List** item from the selected *gameObjects*.



*Tip: If you want to get rid of the menu items in the Project window, simply comment out all the contents of the `ManulToolbarMenuItems.cs` file.*

## 6.4 Value Fields

There are five item types (**Toggle, Popup, Slider, Number, Text**) that share most of the options and functionalities. In order not to repeat the same information, these shared functionalities are described in two subsections: 6.4.1 Editor Prefs and [6.4.2 On Change Value Actions](#).

<i>bool</i>	<b>Toggle</b>
<i>int</i>	<b>Popup, Number, Slider</b>
<i>float</i>	<b>Number, Slider</b>
<i>string</i>	<b>Text</b>

These items are called **Value Fields** because each of them creates a value field on the upper toolbar. Users can interact with this field and change its value. Each item type from this group is associated with a value type. Look at the table to see which item type you can use to create a field for a specific value type.

*All example assets which this section refers to are in the **Examples** folder. See the [4.4.1 Use Button List Asset](#) section to learn how to override a toolbar side with items in such assets.*

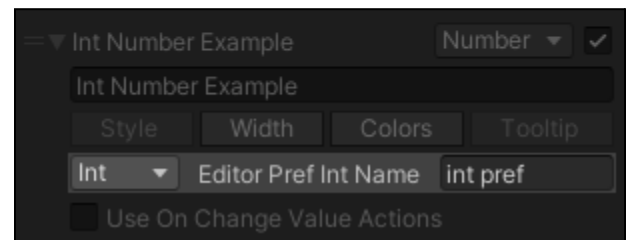
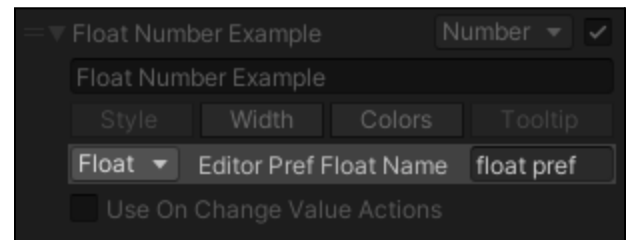
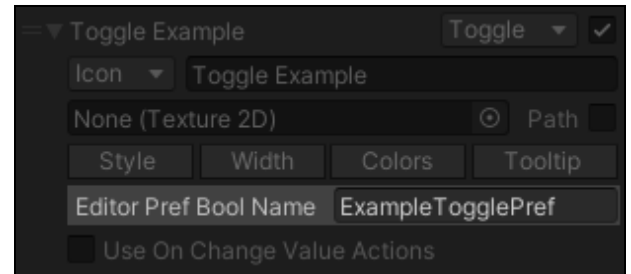
### 6.4.1 Editor Prefs

Each item from the **Value Fields** group uses an *editor pref*, and each time the value is changed it will be saved in this *editor pref*. That's why the first thing you need to do is to specify the name of this *pref*. Make sure it's unique and it's not used by any other item. Type the name in the text field with a label which contains **"Editor Pref"** words.

The label for this field will be different for various items and it depends on the value type. For **Toggle** it will be **"Editor Pref Bool Name"** (you can see this field in the first image on the right), for **Popup** it will be **"Editor Pref Int Name"**, and so on.

Additionally, **Number** and **Slider** types have a dropdown that lets you choose between *float* and *int* value type. This dropdown is located before the text field for the *pref* name (see the images on the right). So for the **Float** the label will be **"Editor Pref Float Name"**, and for the **Int** the label will be **"Editor Pref Int Name"**.

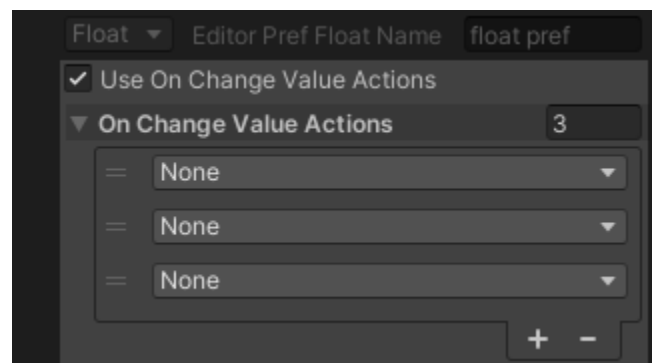
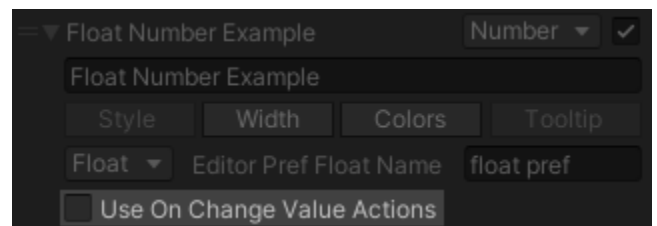
More information about *editor prefs* can be found [here](#).



### 6.4.2 On Change Value Actions

You can add actions to every item type from the **Value Fields** group which will be performed each time the value of the item is changed. Look at the **Use On Change Value Actions** toggle. By default, it is disabled.

When you enable it, the **On Change Value Actions** list will appear under the toggle. Populate this list with elements. For each element you need to specify the type of the action, and, depending on the type, you need to fill text, object, or other fields that will appear.





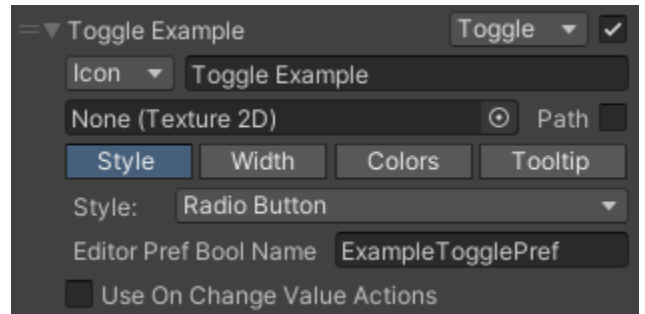
Of course it doesn't make sense to create a slider that will open a prefab each time the slider's value is changed. That is why some actions will be more useful for adding them to the **On Change Value Actions** list than others. The most useful types of actions for this purpose are those which invoke methods (**Static Method**, **Object Of Type Method**, **Component Method**) and events (**Invoke Event**). You will find out more about this in the [7. Action Types](#) section.

Also, you can check the **ExampleItems3** asset. The asset's items named *"Example Number (float)"*, *"Example Number (int)"*, *"Example Slider (float)"* and *"Example Slider (int)"* show how to use **On Change Value Actions**.

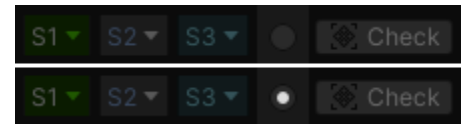
### 6.4.3 Toggle

**Toggle** is a switch that can have one of two values: on or off (true or false). It uses a *bool* value type, and upon a value change, it also saves value in the associated *editor pref* (true or false) of the *bool* type.

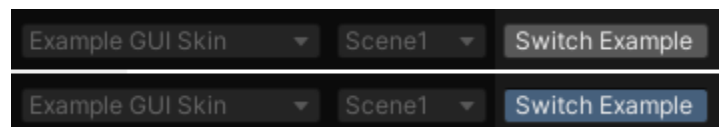
The first example of a **Toggle** type can be found in the **ExampleItems1** asset. It is named *"Toggle Example"* (10th element on the list).



This item creates a small empty circle - a toggle - visible in the image on the right. You can click on it to change its value. To check if everything works properly, you can click the **Check** button next to it. A message will appear in the console informing you about the current value of the *bool editor pref* associated with this toggle.



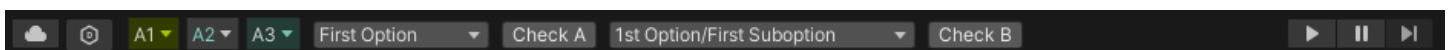
Another example of a **Toggle** type can be found in the **ExampleItems2** asset. This item is named *"Switch Example"* and its label is visible on the toolbar.



The label visibility depends on the style you choose for this item. Some styles will show it, others will not. More information about styles can be found in the [5.2.1 Style](#) section.

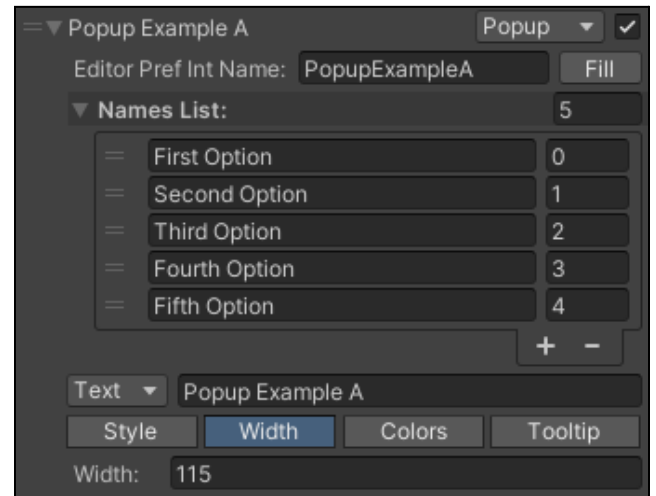
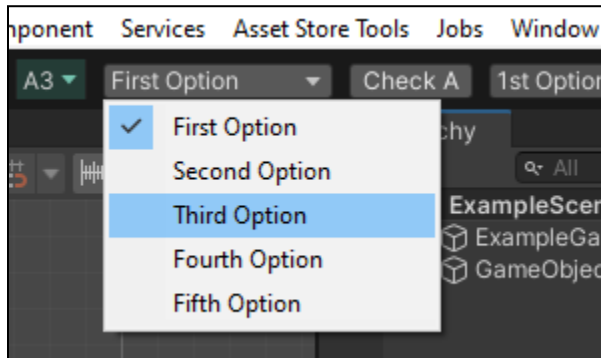
### 6.4.4 Popup

**Popup** is a dropdown menu that will set and save an integer value as an *editor pref* upon changing its value. You can see two examples of a **Popup** type in the image below: *"First Option"* and *"1st Option / First Suboption"*. You can find these items in the **ExampleButtonList1** example asset (the first one is named *"Popup Example A"*, and the second is named *"Popup Example B"*).



After selecting the **Popup** option as an item type, enter the name of the *editor pref* of the *int* type (e.g. **“PopupExampleA”**), which the popup will be associated with. For now, ignore the **Fill** button.

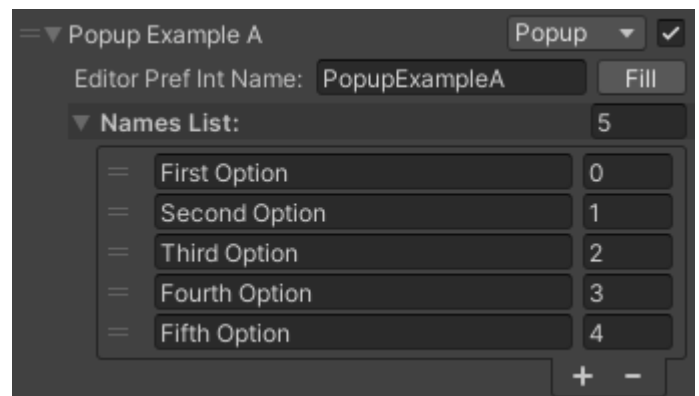
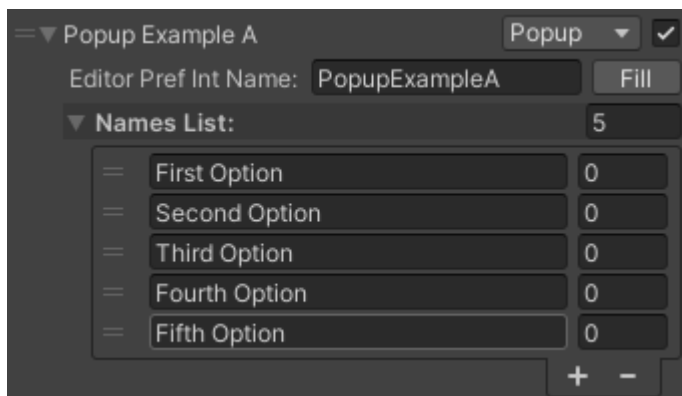
Next, you must create a list of items which will be shown after left-clicking on the popup in the upper toolbar. Each element consists of a name (left text field) and an index associated with this name (right integer field). Look at the image on the right. The example list has five items (**First Option** has a value of **0**, **Second Option** has a value of **1**, **Third Option** has a value of **2**, and so on).



So, if, for example, you select the **Third Option** from the popup dropdown menu and left-click on it, it will set and save a value of **2** in the *editor pref* named **“PopupExampleA”**.

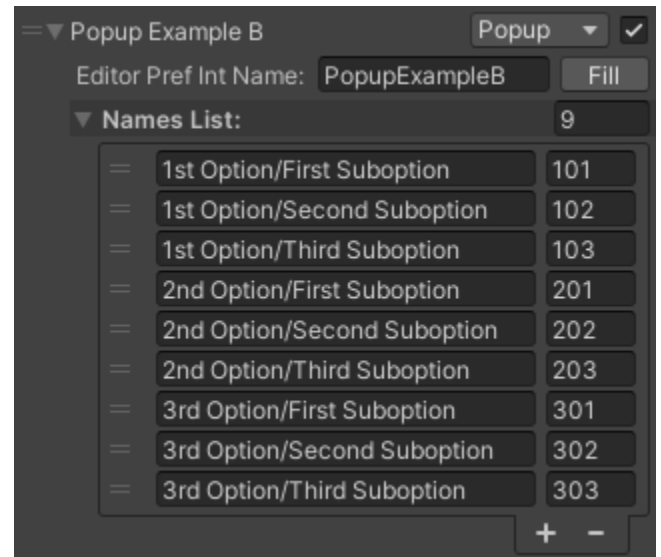
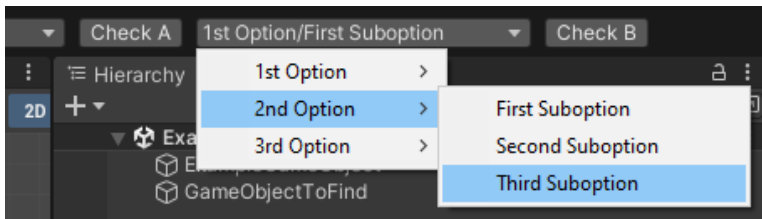
You can check this value by clicking the **Check A** button (next to the first popup dropdown). A message will appear in the console informing you about the current value of the **“PopupExampleA”** (it should be 2).

When you create a list of items for the dropdown, you must specify an integer value for each item. You can do this manually or click on the **Fill** button. It will automatically put a number in each element (starting with index 0). To see how this button works, first manually change all numbers to **0** (left image below) and then click on the **Fill** button. You will see how the numbers changed from only **zeros** to **0, 1, 2, 3, 4** (the right image below).



You can also use the **“/”** character (forward slash) to create more complex lists of items. Each **“/”** character will create another sub-dropdown.

You can see how it works in the upper toolbar on the left image below. The right image below shows how the setup of this popup looks like. Also, the elements' indices don't have to be incremented by one, nor ascend. You can enter various integer numbers, just make sure each option has an index with a different value.



You will notice that the **Popup** type is the only type whose name is not shown on the upper toolbar. If you want to create a label for a popup, use another item of **Label** type. Also, you cannot add a tooltip for the **Popup** item type.

It is recommended to set the fixed width for a **Popup** type (see the [5.2.2 Width](#) section for more information).

#### 6.4.5 Slider

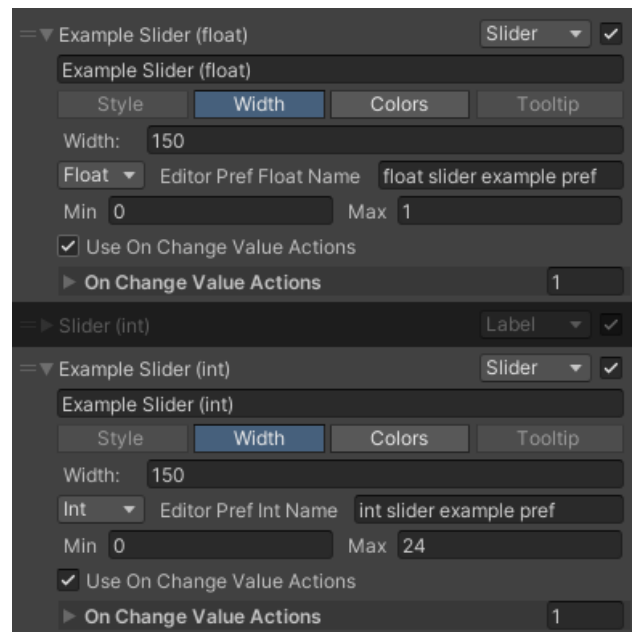
**Slider** type creates a slider on the upper toolbar. You can click and drag the handle of this slider to change its value. This will also change the value of the *editor pref* associated with the slider. Two example sliders can be found in the **ExampleItems3** asset. They are named “*Example Slider (float)*” and “*Example Slider (int)*” (labels visible in the image below are not part of these items).



After selecting the **Slider** type, you need to decide if you want to create a slider based on a *float* type or on an *int* type. You can do that by selecting the **Float** or **Int** option in the dropdown next to the field for *editor pref*.

Then type the name of the *editor pref* (that will be associated with the slider) in the **Editor Pref Int Name** or **Editor Pref Float Name** field. Make sure it's unique and it's not used by any other item.

The last thing to do is to specify the minimum and maximum value of the slider by typing appropriate values in the **Min** and **Max** fields. Users won't be able to set the slider to a value that is not clamped by these two values.

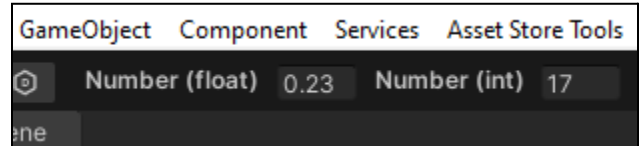


It is recommended to set the fixed width for a **Slider** type (see section [5.2.2 Width](#) for more information). Remember that in order for a slider to appear it must have a width that is no less than 105. Also remember that you cannot choose a style for a **Slider** type and you cannot add a tooltip to it.

#### 6.4.6 Number

**Number** type creates a number field in which the user can type a numerical value. When the value is changed and this change is accepted (by pressing **Enter** or **Tab** keys, or clicking anywhere in the editor), a new value will be saved in the associated *editor pref*.

Two example **Number** items can be found in the **ExampleItems3** asset. They are named “**Example Number (float)**” and “**Example Number (int)**” (labels visible in the image are not part of these items).



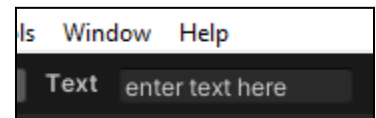
After selecting the **Number** type, you need to decide if you want to create a number field based on a *float* type, or on an *int* type. You can do that by selecting the **Float** or **Int** option in the dropdown next to the field for *editor pref*. Then type the name of the *editor pref* (that will be associated with the slider) in the **Editor Pref Int Name** or **Editor Pref Float Name** field. Make sure it's unique and it's not used by any other item.

Also, take note that you cannot change the style of the **Number** item, nor add a tooltip to it.

#### 6.4.7 Text

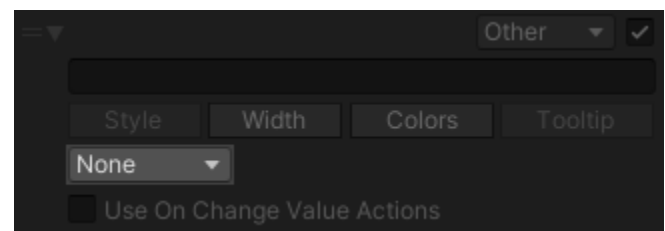
**Text** type creates a text field in which the user can type a text value. When the value is changed and this change is accepted (by pressing *Enter* or *Tab* keys or clicking anywhere in the editor), a new value will be saved in the associated *editor pref*. Type the pref name in the **Editor Pref String Name** field.

An example **Text** type item can be found in the **ExampleItems2** asset. It is named “**Example Text**” (label “**Text**” in the image is not part of the item). Keep in mind that you cannot change the style of the **Text** item, nor add a tooltip to it.



### 6.5 Other

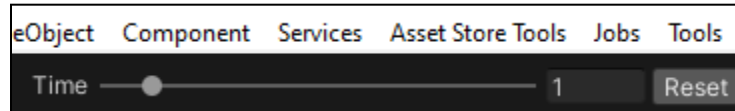
**Other** is a type that contains various predefined items. After selecting this type, you will notice that a new dropdown has appeared with a default option **None**. Use this dropdown to select a subtype. For now, there are two subtypes: **Time Scale** slider and **Frame Rate** slider. More will be added in the future versions of the tool.



You can find examples of both subtypes in the **ExampleItems4** asset. For the both subtypes you cannot change their style, nor create a tooltip. However, you can use **On Change Value Actions** in the same way they work with **Value Field** types, which is described in the [6.4.2 On Change Value Actions](#) section.

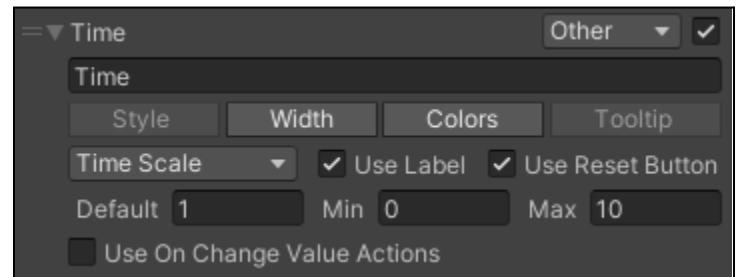
### 6.5.1 Time Scale Slider

**Time Scale** creates a slider on the upper toolbar that lets you change the *Time.timeScale*, i.e. slow down or speed up the time scale in the *Play Mode* (more about *Time.timeScale* can be found [here](#)). This is very useful for testing and debugging. After selecting this subtype, some new options will appear next to and beneath the subtype dropdown.



**Min** and **Max** fields let you specify the minimum (no less than 0.0) and maximum (no more than 100.0) value of the slider.

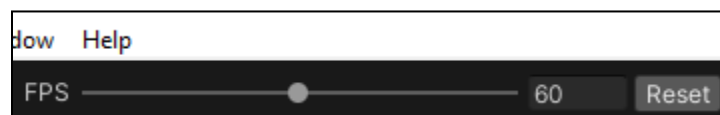
If the **Use Label** toggle is enabled, it will show a label on the upper toolbar with a text that was typed in the first field in this element (e.g. *"Time"*, see the images above).



If the **Use Reset Button** toggle is enabled, it will show the **Reset** button on the upper toolbar (see the first image on this page). When the user clicks this button, it will set the slider to a value that is typed in the **Default** field.

### 6.5.2 Frame Rate Slider

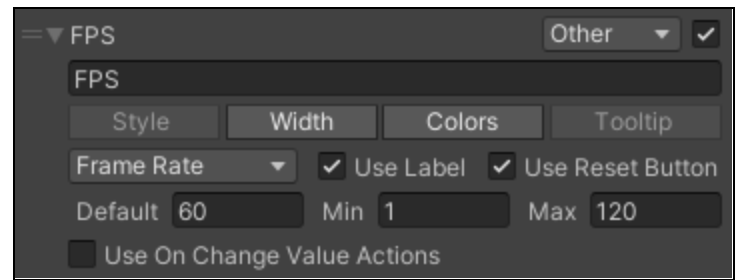
**Frame Rate** creates an *int* slider on the upper toolbar that lets you set the *Application.targetFrameRate* value, namely the target frame rate at which Unity tries to render your game (more about *Application.targetFrameRate* can be found [here](#)).



After selecting this subtype, some new options will appear next to and beneath the subtype dropdown.

**Min** and **Max** fields let you specify the minimum (no less than 0) and maximum value of the slider.

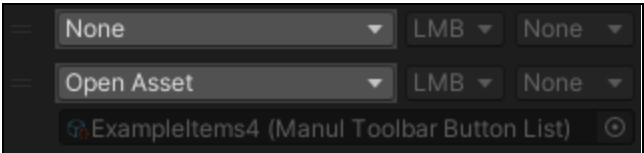
If the **Use Label** toggle is enabled, it will show a label on the upper toolbar with a text that was typed in the first field in this element (e.g. ***"FPS"***, see the image to the right).



If the **Use Reset Button** toggle is enabled, it will show the **Reset** button on the upper toolbar (see the image above). When the user clicks this button, it will set the slider to a value that is typed in the **Default** field.

## 7. Action Types

Actions are used by various item types. You add them to lists which are always located at the end of the item's element. You start the action configuration by selecting its type (the default type is **None**) in the first dropdown in the first row of the element. Dropdowns of two actions are shown in the image on the right. More information can be found in the table below, which shows differences in the action configuration between various item types.



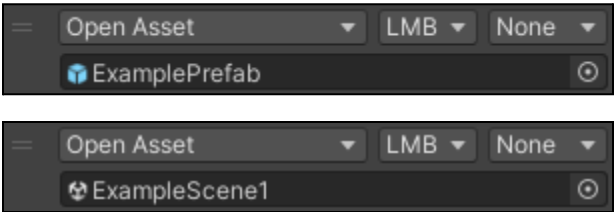
Item type:	Button	List	Toggle, Popup, Slider, Number, Text, Other
Add actions to:	<i>Button Actions</i>	<i>Actions List</i>	<i>On Change Value Actions</i>
Additional configuration:	For each action you need to specify the mouse button (and optional keyboard button) which will be used by the user to perform this action (see section <a href="#">6.1.2 Mouse and Keyboard Buttons</a> ).	For each action you need to type its name that will be visible in the list popup (see section <a href="#">6.3.2 List Actions</a> ).	None.

After selecting the action type, new rows with new fields will appear. You need to fill them with text or objects, depending on the action's type. Each action type is described in the following sections.

All example assets which this section refers to are in the **Examples** folder. See the [4.4.1 Use Button List Asset](#) section to learn how to override a toolbar side with items in such assets.

### 7.1 Open Asset

Opens the asset that was dragged and dropped in the field in the second line. The way the asset is opened depends on its type. For example, a **prefab** will be opened in the prefab edit window, a **scene** will be loaded in the editor, and an **image** will be opened in your default image editing tool.

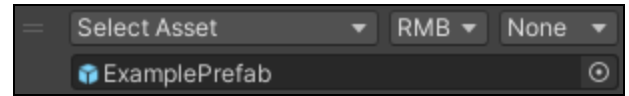


These are not all the types of assets that can be opened, but there is no point in listing them all. In short, the **Open Asset** action is the equivalent of double-clicking on the asset in the *Project* window.

An example of using this action is the **Prefab** button in the **ExampleItems1** asset. Left-click it to open the **ExamplePrefab** in the prefab editing window. Other examples are the **S1**, **S2**, **S3** buttons (in the same example asset) which, upon left-clicking on them, open scenes: **ExampleScene1**, **ExampleScene2**, and **ExampleScene3**.

## 7.2 Select Asset

Selects the asset that was dragged and dropped in the field in the second line. It is equivalent to a single left-click on an asset in the *Project* window.

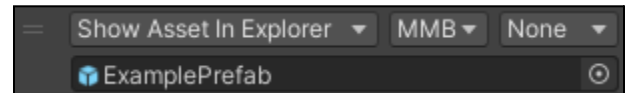


An example of using this action is the **Prefab** button in the **ExampleItems1** asset. Right-click on it to select the **ExamplePrefab** in the *Project* window.

## 7.3 Show Asset In Explorer

Opens the folder that contains the asset (which was dragged and dropped in the field in the second line) in explorer. It is equivalent to right-clicking on the asset in the *Project* window and selecting the *Show In Explorer* option.

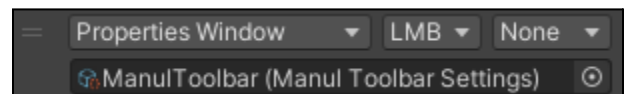
An example of using this action is the **Prefab** button in the **ExampleItems1** asset. Middle-click on it to open the folder containing this asset.



## 7.4 Properties Window

Opens a window with the properties of the asset that was dragged and dropped in the field in the second line (the same asset properties that will appear in the *Inspector* window after selecting the asset). This is the equivalent of selecting an object or asset and selecting **Assets > Properties** from the upper menu items.

An example of using this action is the **Toolbar** button. Left-click on it to open a window with the properties of the **Manul Toolbar Settings** asset.



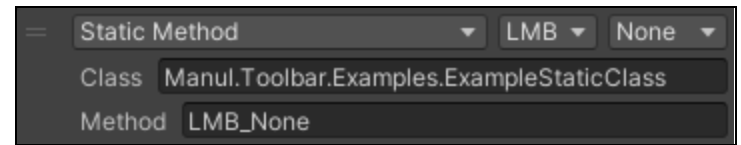
***Tip:** The ability to open a new window with object properties is a great and often overlooked option. It's a good idea to assign a key shortcut to it. When we need to open a window with the properties of more than one object, pressing this key is a much faster way than creating a new Inspector window and locking it with the padlock icon. Moreover, it is also a great way to view and change options in Scriptable Object assets.*

This action works only in **Unity 2021.3.1f1 or above**. If you are working in a lower version, this action will behave in the same way as the **Select Asset** action.



## 7.5 Static Method

Calls a static method with the name entered in the **Method** field from a static class with the name entered in the **Class** field (including namespace, see the image on the right).

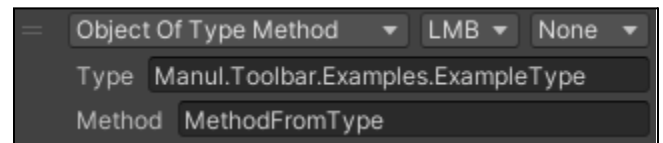


An example of using this type of action is the first example button, marked with an arrow in the image. This button is called **“Mouse & Keyboard Buttons”** and you can find it in the **ExampleItems1** asset.



## 7.6 Object of Type Method

Searches for an object of the type entered in the **Type** field (including namespace, see the image on the right) in the current scene, and calls a method with the name entered in the **Method** field from the object that was found. For the action to be performed, there must be exactly one object of this type in the current scene. The search is done by using the *Object.FindObjectOfType<T>* method.

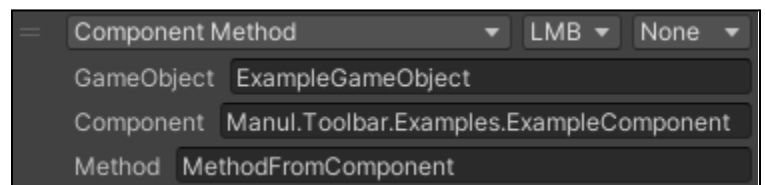


An example of using this type of action is the example button called **Type** in the **ExampleItems1** asset. First, left-click on the **S2** button to open the scene called **ExampleScene2**. It contains a *gameObject* with the **ExampleType** component. Then left-click on the **Type** button to invoke a method from that component.

## 7.7 Component Method

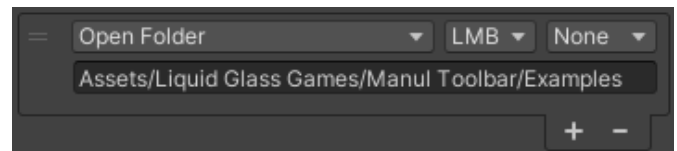
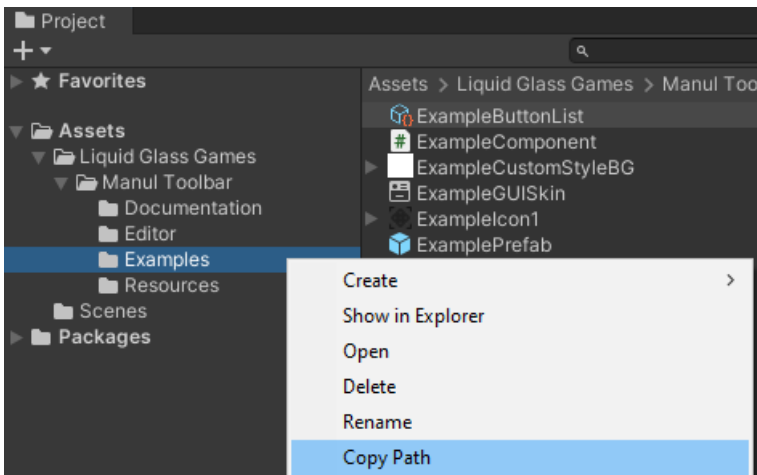
Searches for a *gameObject* with the name entered in the **GameObject** field in the current scene. Then, it looks for a component (added to this *gameObject*) of the type whose name you entered in the **Component** field (including namespace). Finally, it calls the method with the name given in the **Method** field from the first component found.

For the action to be performed, there must be exactly one *gameObject* in the current scene, with the name that you entered in the **GameObject** field. The search is done by using the *GameObject.Find* method.



An example of using this type of action is an example button called **Component** in the **ExampleItems1** asset. First, left-click on the **S1** button to open one of the sample scenes - **ExampleScene1**. It contains a *gameObject* with the **ExampleComponent** component. Then left-click on the **Component** button to invoke a method from that component.

## 7.8 Open Folder

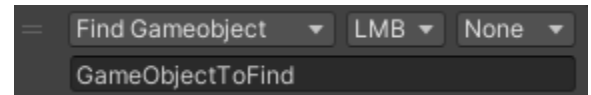


Opens a folder whose path was entered in the text field (see the image above). It is easier to copy and paste the path of the folder rather than type it manually. In order to copy the path of the folder, right-click on it in the *Project* window and select **Copy Path** (see the image on the left) and then paste the copied path in the text field.

An example of using this type of action is an example button called **A1** in the **ExampleButtonList1** asset. To test it, in the *Project* window, first select any folder except the **Examples** folder. Then left-click on the **A1** button on the upper toolbar. The **Examples** folder should be selected and its contents should appear in the right column of the *Project* window.

## 7.9 Find GameObject

Searches for a *gameObject* with the name entered in the text field (e.g. **"GameObjectToFind"**, see the image on the right) in the current scene and selects it. This action will work only if there is exactly one *gameObject* with the given name in the current scene.



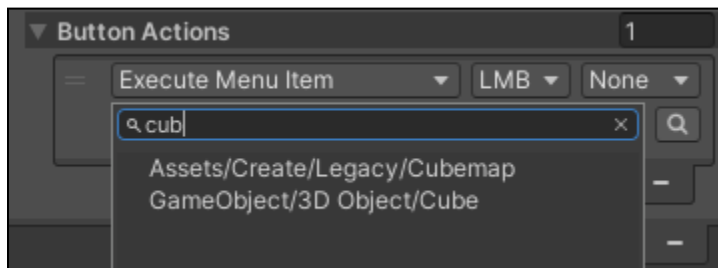
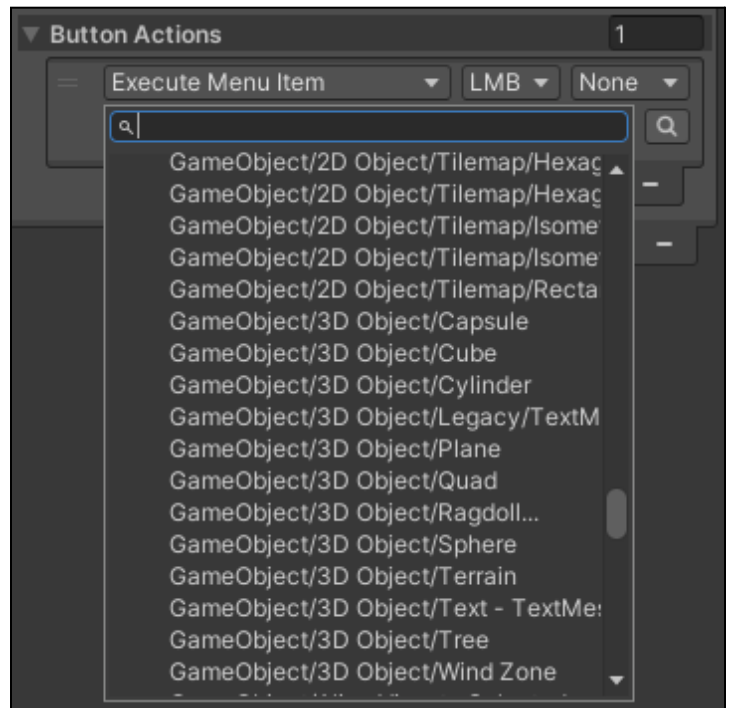
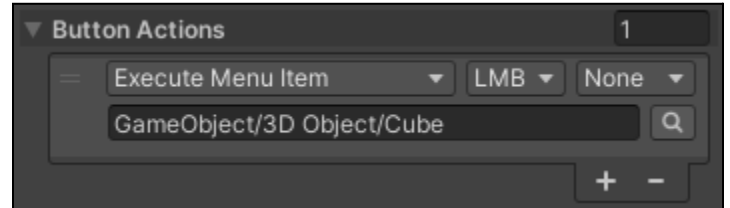
An example of using this type of action is an example button called **A2** (in the **ExampleButtonList1** asset). Open the **ExampleScene1** scene (located in the **Examples** folder), and then left-click on the **A2** button. A *gameObject* with a name **"GameObjectToFind"** should be selected, and its properties should be shown in the *Inspector* window.

## 7.10 Execute Menu Item

Executes any menu item. Menu items are lists of various actions grouped above the upper toolbar (*File, Edit, Create*, etc.). For example, you can use such an item to open a **Project Settings** window (**Edit / Project Settings...**) or to create an empty *gameObject* (**GameObject / Create Empty**). You must have definitely used some of them before.

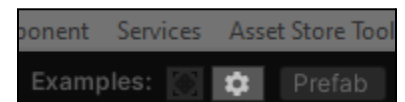
With **Execute Menu Item** action you can execute any of these items. Just type the full path of the item in the field beneath the action type popup (e.g. **“GameObject/3D Object/Cube”**, see the image on the right). Remember not to use spaces before and after backslashes.

You can also click the **search icon** (with the magnifying glass image) to open a window with a list of all menu items. You can type something in the upper text field to narrow your search down (for example, **“cub”** in the image below), and then double-click on an item that you want to choose. The window will close and the path of the selected item will appear in the text field.



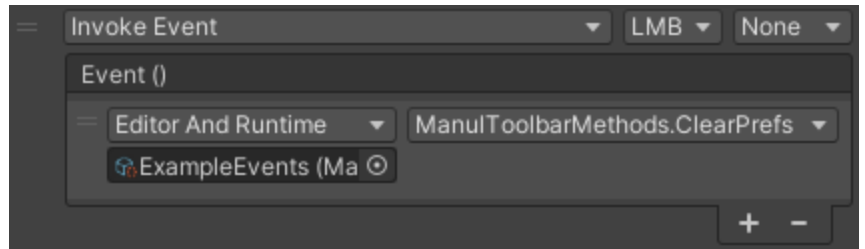
An example of using this type of action is an example button called **“A3”** (in the **ExampleButtonList1** asset). This button will execute a menu item named **“GameObject / 3D Object / Cube”**, which will create a *gameObject* with *Box Collider* and *Mesh Renderer* components. To test this action, simply left-click on the **A3** button, and the cube *gameObject* will be created in the current scene.

Another example is in the **ExampleItems1** asset and it's called **“Project Settings”**. This item is a button with a gear icon that will open a *Project Settings* window (using an **Execute Menu Item** action) when you click it.



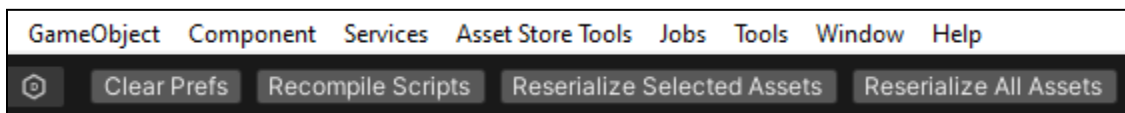
## 7.11 Invoke Event

This action invokes an *Unity Event* (more information about *Unity Events* can be found [here](#)). After selecting this type a standard *Unity Event* user interface will appear.



First of all, make sure that the dropdown above the object field is set to **Editor And Runtime** (as you can see in the image above). By default, this dropdown is set to **Runtime Only**, which means that this event can be invoked only in the *Play Mode*. So if you don't change this option to **Editor And Runtime**, your action will not invoke the event in the *Edit Mode*.

Next, drag and drop an object to the field in the second row (e.g. **ExampleEvents** *scriptable object* in the image above). Remember that you cannot drag objects from the scene. The object has to exist in your project as a *prefab* or *scriptable object*. Then select a method (e.g. **ManulToolbarMethods.ClearPrefs**) and optionally set the *bool*, *int*, *float*, or *string* parameter, if the method has it.

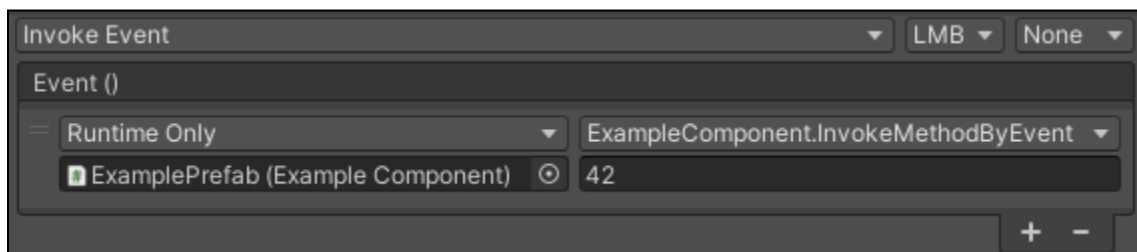


Examples of using this action type are in the **ExampleEventsButtons** asset. All four items (**Clear Prefs**, **Recompile Scripts**, **Reserialize Selected Assets**, and **Reserialize All Assets**) are buttons that use an **Invoke Event** action with a *scriptable object* file named **ExampleEvents**. In this *scriptable object* there are four methods and each button uses a method from this object (see section [8. Additional Resources](#) for more information about what these methods do).

Another example can be found in the **ExampleItems2** asset. This item is named **"Prefab Event"**, and it's a



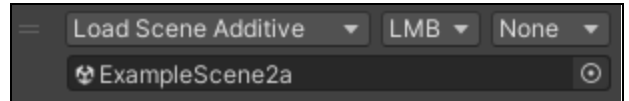
button that also performs an **Invoke Event** action. But instead of using a *scriptable object*, this event has a prefab (named **"ExamplePrefab"**) dragged and dropped into the event's object field. The action uses a method (named **"InvokeMethodByEvent"**) from a component (named **"ExampleComponent"**) that is added to the aforementioned prefab. The method has also an example *int* parameter. You can change it and then click the **Prefab Event** button to see a console message with the current value of the parameter.



## 7.12 Load Scene Additive

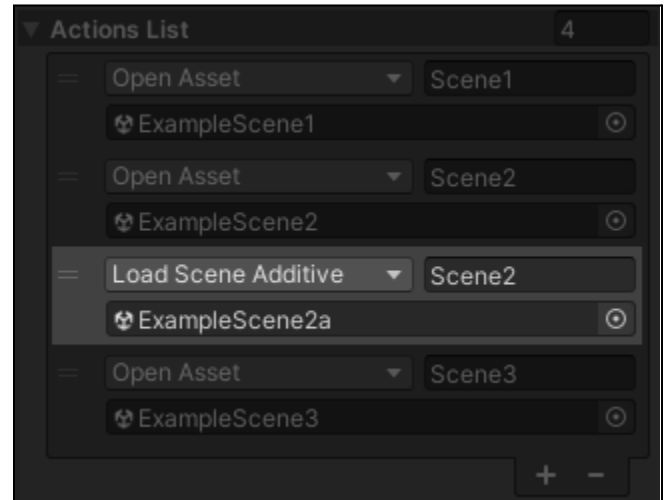
By default, opening a new scene in the editor using the **Open Asset** action will close the current scene. If you want to load

a scene in additive mode, namely without closing the current scene, use the **Load Scene Additive** action. It has one object field in which you can drag and drop a *scene asset* (this action will not work with other types of assets).



You can find an example of this action in the **ExampleItems2** asset. The item is named **“Load Scene (Additive)”** and it creates a button (look at the image above) which, upon clicking, will load the **ExampleScene2a** scene without closing the currently opened scene.

Another example is the **List** type named **“Scene Selection”** (in the same example asset). In this item, the **Load Scene Additive** action is used in the third element in the **Actions List** of this item (look at the image on the right).



## 8. Additional Resources

In the **Editor** folder you can find a script named **ManulToolbarMethods.cs** that contains four methods. They can be helpful during the development of your project.

- **Clear Prefs** - deletes all *PlayerPrefs*. Make sure you know what you're doing before using this method. More information about *PlayerPrefs* can be found [here](#).
- **Recompile Scripts** - recompiles all scripts. Scripts are automatically recompiled each time you make a change in any script so this method could be seen as unnecessary. However, if your project is using an external text file asset to store texts (subtitles, descriptions, item names etc.), then when you make a change in this file, this change will not be visible in the editor. You have to recompile scripts to make editor update scripts (including text files) in order to see the aforementioned change.
- **Reserialize Selected Assets** - this method will use ***AssetDatabase.ForceReserializeAssets*** method to force reserialize selected assets. This can be helpful (for example, if your assets have errors after upgrading Unity editor to a newer version), but make sure you know what you're doing before using this method. More information about force re-serializing assets can be found [here](#).
- **Reserialize All Assets** - same as above but this method will force reserialize all assets.

## 9. Conclusion

We hope that the **Manul Toolbar** asset will speed up your work significantly!

If you have any problems, questions, or suggestions, please write at: [support@liquid-glass-games.com](mailto:support@liquid-glass-games.com) or create a post on the Unity Discussions Forum [here](#).