MANUL Toolbar

User Guide

1. Introduction..... 3

1.1 Important informations..... 3

1.2 Version History..... 3

2. Quick Start..... 4

3. Adding New Button..... 5

3.1 Enable Button..... 5

3.2 Button Label..... 6

3.3 Button Type..... 7

4. Settings..... 8

4.1 Offsets..... 8

4.2 Default Styles..... 8

4.3 Other..... 9

4.4 Override..... 9

4.4.1 Use Button List Asset..... 9

4.4.2 Use Button List Set Asset..... 10

5. Button Actions..... 12

5.1 Adding New Action..... 12

5.2 Mouse and Keyboard Buttons..... 12

5.3 Button Action Types..... 13

5.3.1 Open Asset..... 14

5.3.2 Select Asset..... 14

5.3.3 Show Asset In Explorer..... 14

5.3.4 Properties Window..... 15

5.3.5 Static Method..... 15

5.3.6 Object of Type Method..... 15

5.3.7 Component Method..... 16

5.3.8 Open Folder..... 16

5.3.9 Find GameObject On Scene..... 17

5.3.10 Execute Menu Item..... 17

6. Button Visuals.....19
6.1 Label, Icon, or Both.....19
6.2 Option Toggles.....19
6.2 Style.....20
6.3 Width.....21
6.4 Colors.....21
6.5 Tooltip.....22

7. Toggle.....23

8. Popup.....24

9. Shortcuts.....26
9.1 Opening MANUL Toolbar Settings.....26
9.1 Menu Items in Project Window.....26
9.2 Change Button Name.....27
9.3 Disable Button.....27

10. Conclusion.....28

1. Introduction

This document contains all the necessary information about the **MANUL Toolbar** asset. It is recommended that you read all sections in the presented order to learn all the capabilities of this tool. Also, make sure that you are using Unity 2020.2.0f1 or above, otherwise this tool will not work properly.

If you have any problems, questions, or suggestions, please write at support@liquid-glass-games.com or create a post on the **Unity Discussions Forum** here.

1.1 Important informations

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

- 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 the 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
- Initial release

Version 1.1
- New button actions: **Open folder** and **Find GameObject in Scene**
- New button type: **Popup** (which uses editor pref of the int type)
- Shortcuts to quickly change the name of the button or disable it
- Menu items in *Project* window to quickly create a button
- Possibility of storing button settings in an external asset

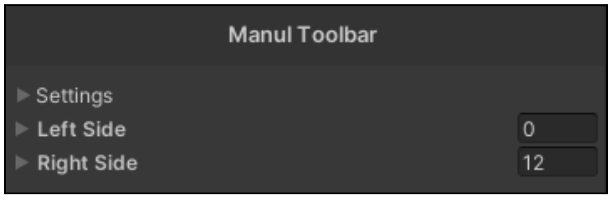
Version 1.2
- Possibility of creating a set of lists of buttons and then switching between these lists with a popup
- Added menu items to open or select Manul Toolbar Settings asset

Version 1.3
- Changed the way the settings are updated with the newer tool versions
- New button action: **Execute Menu Item**

2. Quick Start

After importing the **Manul Toolbar** asset for the first time and after recompiling the scripts, you'll notice that a few more buttons have appeared on the top bar in Unity (the one with the **Play**, **Pause**, and **Step** buttons).

They look quite random but don't worry, these are just examples. This documentation refers to them so it is recommended not to delete them. You can delete or disable them after reading all the sections of the documentation and learning all the capabilities of the **Manul Toolbar** asset. Also, the last button on the right side can be truncated, depending on which Unity version you are currently using.



Left-click on the **Toolbar** button. If you are using Unity 2021.3.1f1 or above, this will open a window with the properties of the **Manul Toolbar Settings** asset (otherwise it will select this asset and its properties will be seen in the *Inspector* window). This is where you will be creating new buttons and defining their appearance and functionality. The **Manul Toolbar Settings** asset is divided into three sections. Left-click on the heading of any section to expand it:

- **Settings** are general asset settings.

- **Left Side** is a list of buttons that are located to the left of the **Play**, **Pause**, **Step** buttons (marked by green color on the image below). This list is empty for now, but we will add new buttons to it soon.

- **Right Side** is a list of buttons that are located to the right of the **Play**, **Pause**, and **Step** buttons (marked by blue color on the image below). This list has several sample items, including an item that determines the properties of the **Toolbar** button you have just clicked (the first item on the list).

Each item in the **Left Side** and **Right Side** list is an entry that specifies the properties of the button that will appear to 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 the left to the right - in the exact same order as the items on the list are arranged from the top to the bottom. Reorder the items on the list to change the order in which the buttons are arranged on the toolbar.

3. Adding New Button

This section will show you how to add new buttons. First, add a new item to the **Left Side** list. To add a new item, click the **plus icon** or change the list **Size** to 1.

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

3.1 Enable Button

Each newly added button is disabled by default, so the color of its options is semi-transparent. Disabled buttons will not appear on the toolbar. You can change the color of disabled buttons in **Settings: Disabled Color.**

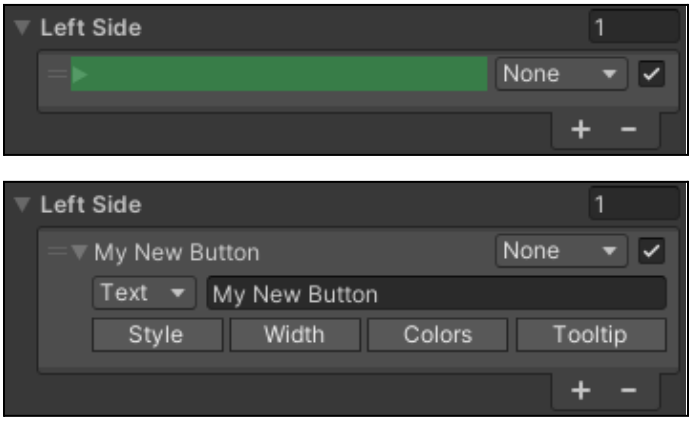
Tip: *As you're working on your project, you will be adding new buttons and others will no longer be needed. You can, of course, delete the latter, but you can also just disable them. If you need them again, simply enable them. It will take you less time than creating them from scratch.*

To enable the button, click on the **toggle** at the end of the first line. The color of the option will change (it will no longer be semi-transparent), and the value of the **toggle** will set to true, which means that the button is active.

3.2 Button Label

Each item on the list with items (which describe the appearance and functionality of the buttons) begins with a header, and in each newly added item this header is empty.

The header is also a foldout button that expands the item options. Left-click on the header (green box on the image) to expand its options. 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 item.



This text will also (optionally) appear on the toolbar button that we are creating (you will see that in the next section).

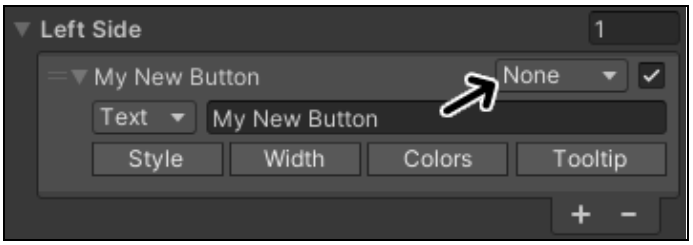
The dropdown menu on the left of the text field allows you to specify what the toolbar button will display: **Text**, **Icon**, **Both** or **None**. More information on this topic can be found in section ***6. Button Presentation***. For now leave the default option (**Text**).

Tip: *Even if your button displays only an icon, without text, it is still worth entering some text in the text field so that the header of the item is not empty on the list. The first button after the* ***"Example:"*** *label, which displays only an icon, is an example. This button doesn't display text but it still has a header with the* ***Mouse Buttons & Keys*** *text (the third item on the* ***Right Side*** *list).*

In order for the button to finally appear on the toolbar, you must complete the last step described in the next section.

3.3 Button Type

The default type of a newly added button is **None**. Buttons of the **None** type will not appear on the toolbar. You can change the button type using the dropdown menu, on the left of the enable / disable toggle. There are four options here:



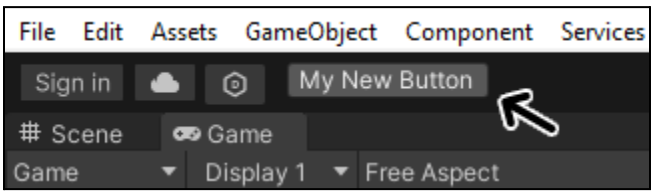
- **Button** - as the name suggests, it's a button. Something will happen if you click it. If you select this type, a list of **Button Actions** will appear at the bottom of the item. In this list, you define the actions that will be performed when you click the button. You also specify which mouse button you need to press, and, optionally, which of the keys you need to hold while clicking to perform the action. More information on this topic can be found in section ***5. Button Actions***.

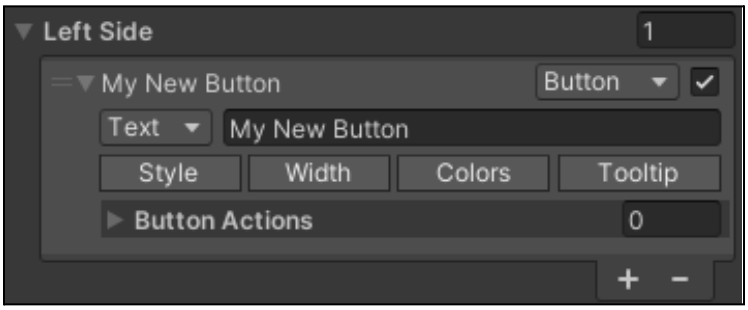
- **Toggle** - a toggle that can have one of two values: on or off (true or false). If you select this type, the **Editor Pref Bool Name** field will appear as the second line. More information on this topic can be found in section ***7. Toggle.***

- **Label** - label displays only text, only an icon, or both, and it is for informational purposes only. Nothing will happen if you click the **Label**.

- **Popup** - this creates a popup button which shows a dropdown menu after left-click. If you select this type, the **Editor Pref Int Name** field will appear as the second line along with a string list below. More information on this topic can be found in section ***8. Popup.***

Change the type of the new button to **Button**. Your new button should appear on the toolbar.



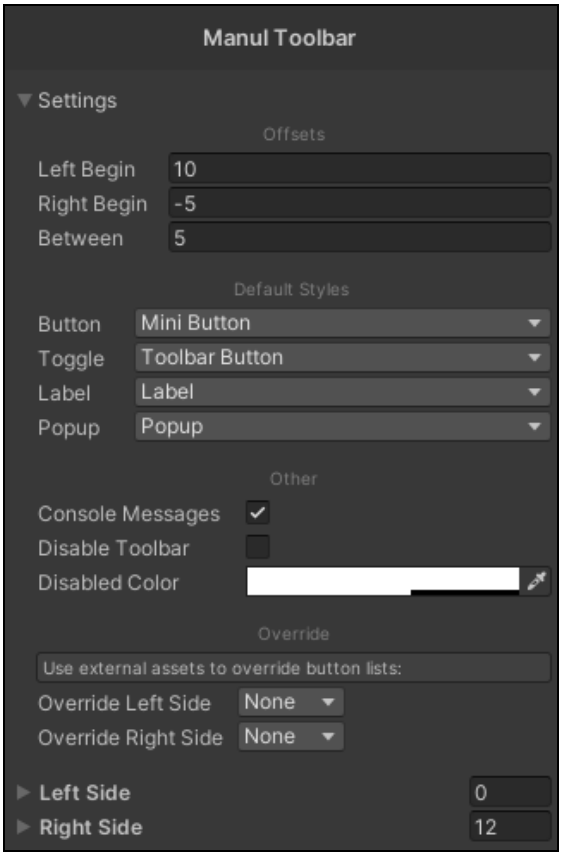


Congratulations! You have successfully added your first button. What's next?

- If you want to learn more about configuring button actions, see section ***5. Button Actions***.
- If you want to learn more about button presentation, see section ***6. Button Presentation***.

4. Settings

The **Settings** section in the **Manul Toolbar Settings** asset allows you to modify several general tool settings. The asset with these settings 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 **Resources** folder. The copy is named **"Manul Toolbar Settings (Default)"** and all its settings are set to default.)

The section is divided into four parts: **Offsets**, **Default Styles**, **Other,** and **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 buttons. 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 buttons. It is marked in blue in the image below.

- **Between** - the horizontal distance between buttons on the toolbar. The value applies to the buttons 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 each of the **Button**, **Toggle**, and **Label** 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 ***6.2 Style***.

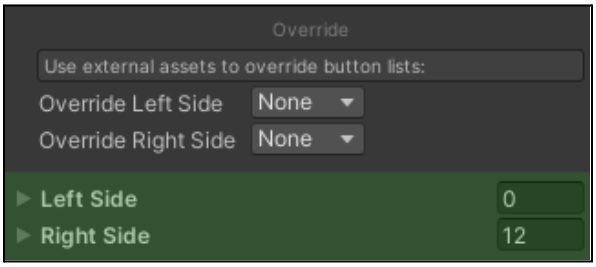
- **Button** - default style for the **Button** type (*Mini Button* style).
- **Toggle** - default style for the **Toggle** type (*Toolbar Button* style).
- **Label** - default style for the **Label** type (*Label* style).
- **Popup** - default style for the **Popup** type (*Popup* style).

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 Unity console window. All messages from MANUL Toolbar have the ***"MANUL Toolbar message"*** prefix.
- **Disable Toolbar** - check this option to disable the display of buttons on the toolbar.
- **Disabled Color** - this color field determines the tint that will be applied to the options of an item if it is disabled on the list. See section ***3.1 Enable Button*** for more information.

4.4 Override



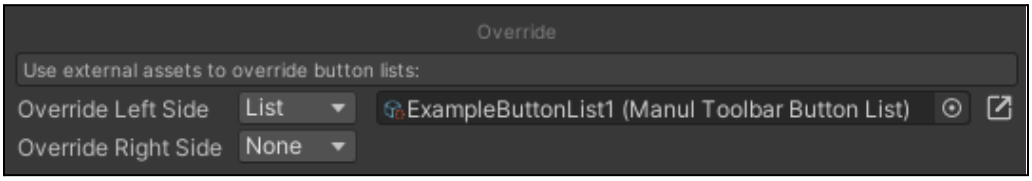
By default, the buttons that appear on the left side of the toolbar are defined in the **Left Side** list (under the **Settings** section) and the buttons that appear on the right side of the toolbar are defined in the **Right Side** list (both lists are marked with green color on the image on the right).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

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

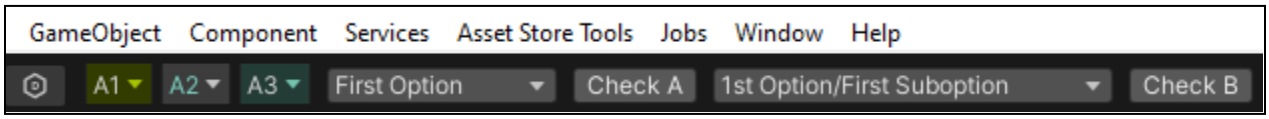
4.4.1 Use Button List Asset

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 buttons' 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 is also an example asset of this type in the **Examples** folder, named **ExampleButtonList1**. You can use it 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 the before-mentioned asset 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 this asset.



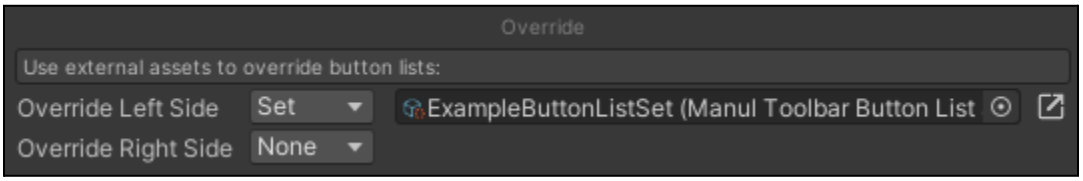
You should notice that some buttons and popups have appeared on the left side of the toolbar (**A1**, **A2**, **A3**, **First Option**, etc.). These buttons and popups are defined in the **ExampleButtonList1** asset. If you want to override the right side of the toolbar, simply create a new asset and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

*If you are new to **Manul Toolbar** and you have created your first button by reading the **3. Adding New Button** section, be sure to set the **Override Left Side** popup to **None** (in the **Settings** section) before reading the next sections!*

4.4.2 Use Button List Set Asset

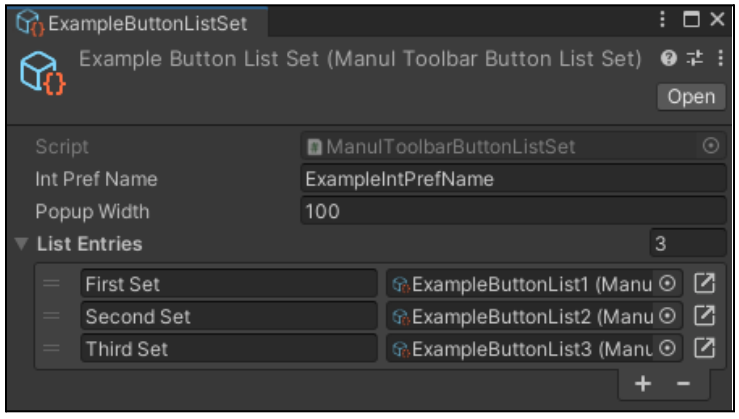
If you want to have a set of lists of buttons and switch between them using a popup to show a different button group each time, use the **Manul Toolbar Button List Set** asset.

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 properties of this asset. You will see two fields and a list.

To define which list of buttons should be shown, **MANUL Toolbar** uses a **Editor Pref** of the *int* type. Type a name in the **Int Pref Name** field (see the example name **"ExampleIntPrefName"** on the image on the right).

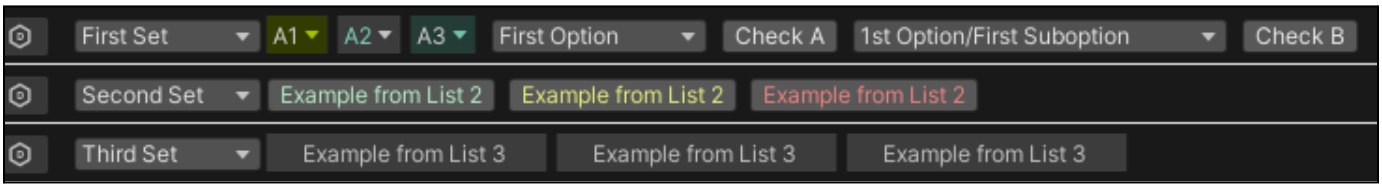


To change the value of the pref (and thus, change the button set that it is being shown), you will use a popup. This popup is located at the beginning of the row with buttons, which you can see on the second image on this page. The popup has a fixed width that you can change in the **Popup With** field (or just leave the default value of **100**).

The last thing to do is to create a list of entries in the **List Entries** list. Each item 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 was 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 item row to open a new window with properties of this asset.

So, for example, if the user chooses the item with the **"Third Set"** name, **MANUL Toolbar** will show the buttons 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 buttons 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 buttons (i.e. have a different value of the* ***Editor Pref****).*

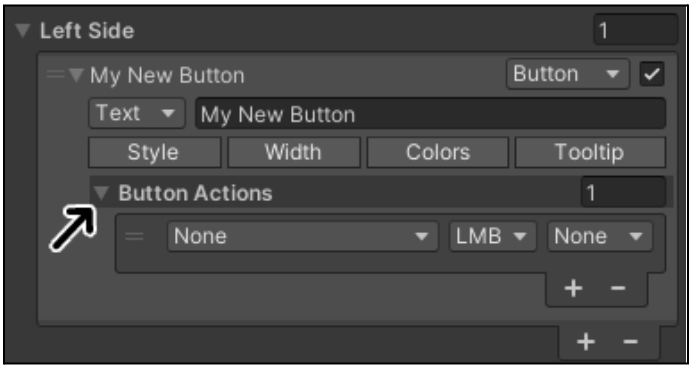
If you are new to ***Manul Toolbar*** *and you have created your first button by reading the* ***3. Adding New Button*** *section, be sure to set the* ***Override Left Side*** *popup to* ***None*** *(in the* ***Settings*** *section) before reading the next sections!*

5. Button Actions

5.1 Adding New Action

If you select the **Button** type, a list of **Button Actions** will appear at the end of the item. Add a new element to it. For the action to be performed after clicking the toolbar button, you need to do two things.

First, determine which mouse (and optionally keyboard) buttons will activate it (you will find more about this in section ***5.2 Mouse and Keyboard Buttons***).



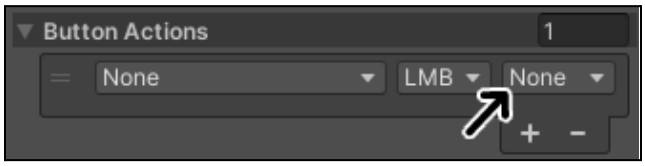
Secondly, you need to specify the type of the action, and, depending on the type, fill text or object fields that will appear (you will find more about this in section ***5.3 Button Action Types***).

5.2 Mouse and Keyboard Buttons

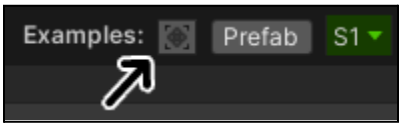
In the first line of the item that defines the action, two dropdowns are responsible for assigning mouse and keyboard buttons to perform this action.

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.

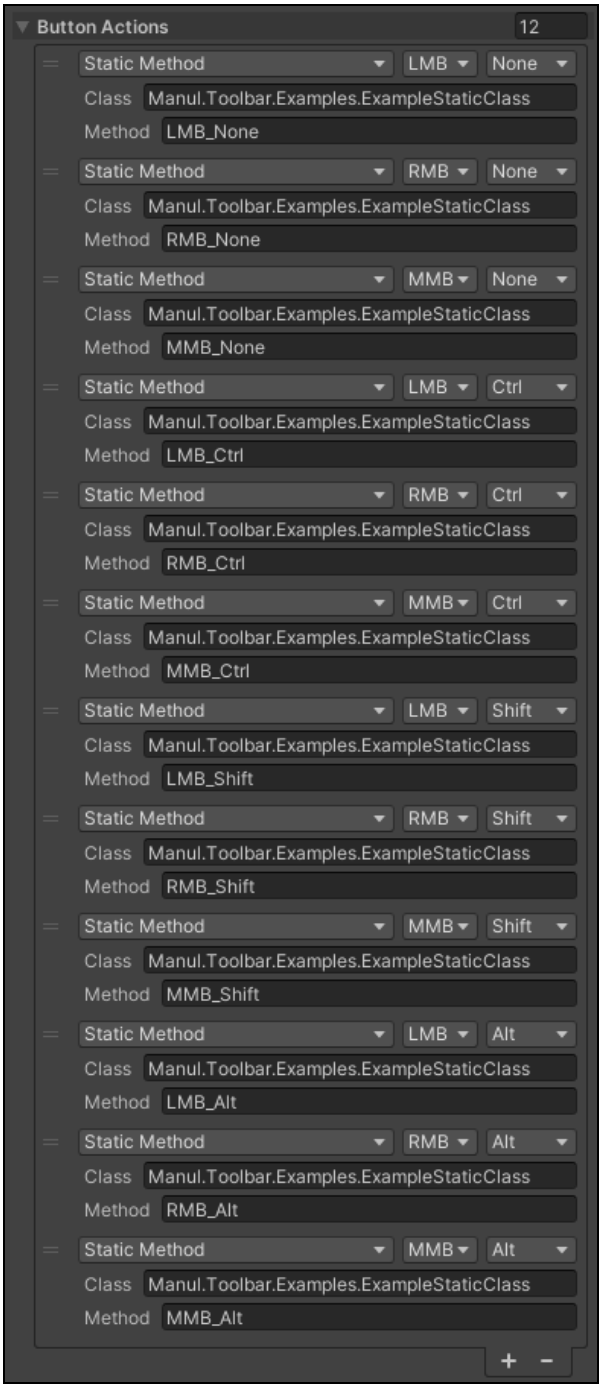
The right dropdown specifies an optional key that must be held during the click for the action to take place. By default, it is **None**, which means you don't have to hold down any keys, but you can change it to **Ctrl**, **Shift**, or **Alt**.



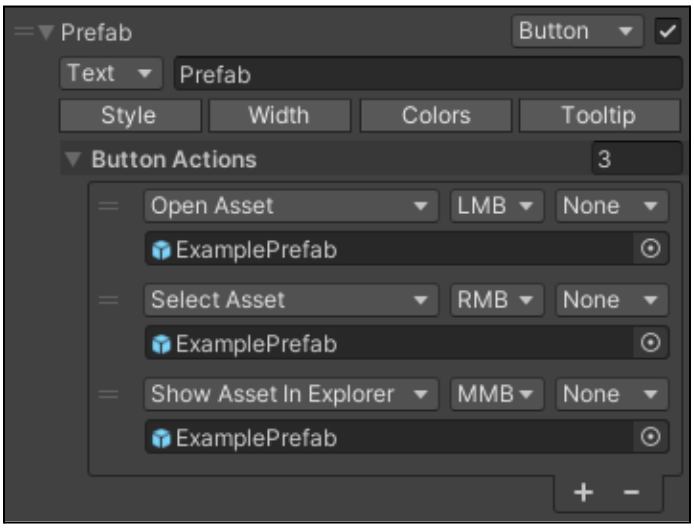
By using the mouse and keyboard buttons you can create as many as twelve actions that can be triggered in various ways - and one button on the toolbar contains them all!



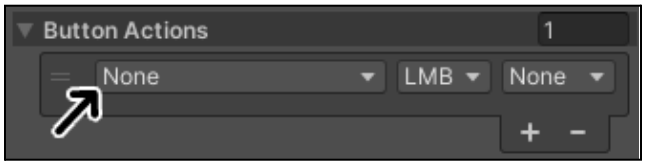
This is shown by the **first example button**, pointed by the arrow on the image to the left. Click on this button with different mouse buttons (**left**, **right**, **middle**) while, optionally, holding down the **Ctrl**, **Alt**, or **Shift** keys to see various messages appear in the console. This button is called **Mouse & Keyboard Buttons** on the **Right Side** list, you can see its twelve actions on 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.



5.3 Button Action Types

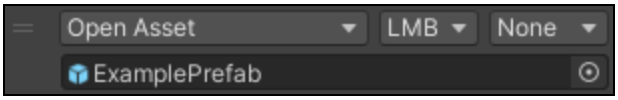


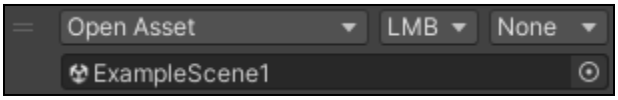
The first dropdown in the first line of the item that defines the action is responsible for determining the action type. After selecting the type of action, fields that need to be filled with objects or text will appear below.

In addition to the default **None** type (which does nothing), you can choose from the following action types:

5.3.1 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 the ***image*** will be opened in your default image editing tool.



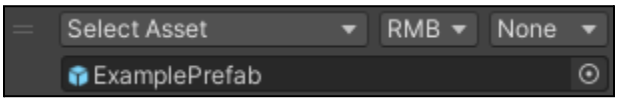


These are not all the types of assets that can be opened, but there is no point in listing them all. In short, the **Open Asset** action is the equivalent of double-clicking on the asset in the *Project* window.

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

5.3.2 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.

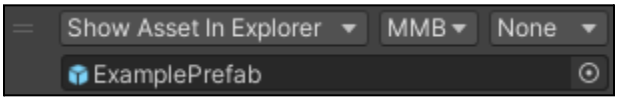


An example of using this action is the **Prefab** button. Right-click on it to select the ***ExamplePrefab*** in the *Project* window.

5.3.3 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.

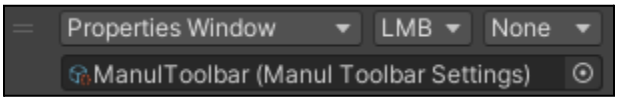
An example of using this action is the **Prefab** button. Middle-click on it to open the folder containing this asset.



5.3.4 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 top toolbar.

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.

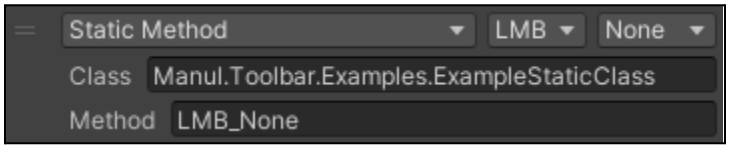


Tip: *The ability to open a new window with object properties is a great and often overlooked option. It's a good idea to assign a key shortcut to it. When we need to open a window with the properties of more than one object, pressing this key is a much faster way than creating a new Inspector window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

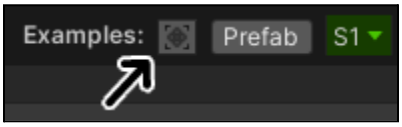
This action works only on **Unity 2021.3.1f1 or above**. If you are working on a lower version, this action will behave in the same way as the **Select Asset** action.

5.3.5 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).

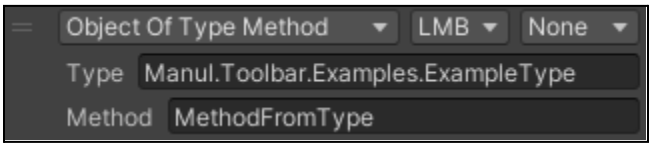


An example of using this type of action is the first example button, marked with an arrow on the image. This button is called **Mouse & Keyboard Buttons** on the **Right Side** list.



5.3.6 Object of Type Method

Searches for an object with the type entered in the **Type** field (**including namespace**, see the image) 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**. 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.

5.3.7 Component Method

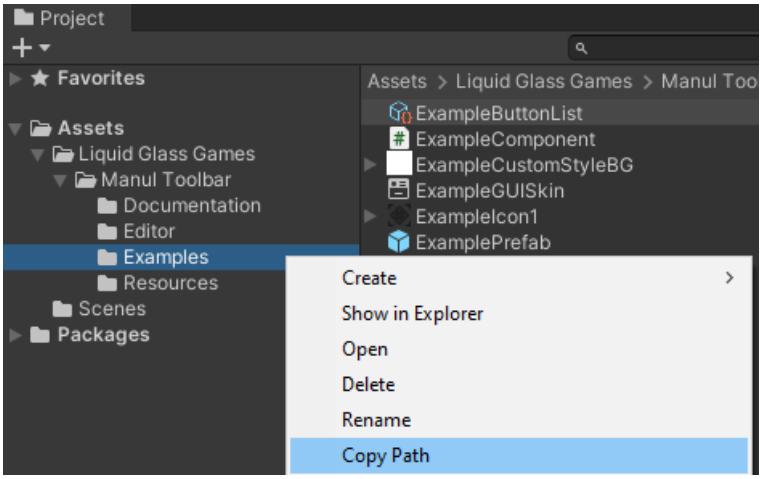
Searches for a *GameObject* with the name entered in the **GameObject** field in the current scene. Then it looks for a component (connected to this *GameObject*) of the type whose name you entered in the **Component** field (**including namespace**). Finally, it calls the method with the name given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

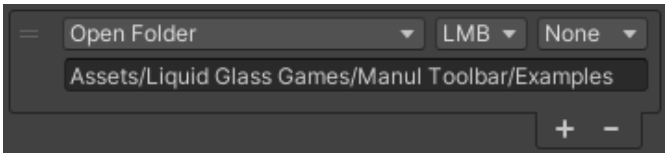
For the action to be performed, there must be exactly one *GameObject* in the current scene, with the name that you entered in the **GameObject** field. The search is done by using the ***GameObject.Find*** method.

An example of using this type of action is an example button called **Component**. 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.

*In order to see the examples for the next types of actions (**Open Folder** and **Find GameObject On Scene**), see section **4.4 Override** and follow the instructions to override the left toolbar side with the **ExampleButtonList** asset. The settings of these example buttons are defined in this very asset.*

5.3.8 Open Folder



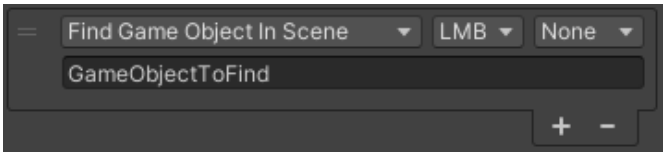


Opens a folder whose path was entered in the text field (see the image above). It is easier to copy and paste the path rather than type it manually. In order to copy the path of the folder, right-click it in the *Project* window and select **Copy Path** (see the image on the left) and then paste it in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

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

5.3.9 Find GameObject On Scene

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.

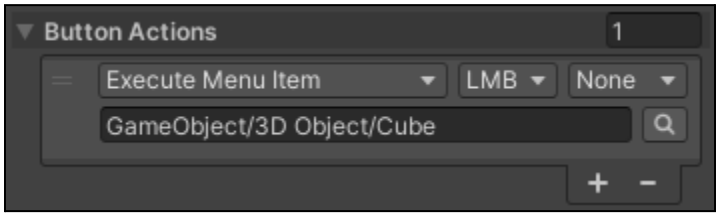


An example of using this type of action is an example button called **A2** (on the buttons list in the **ExampleButtonList** asset). Open the **ExampleScene1** scene (located in the **Examples** folder) and then left-click on the **A2** button. A *gameObject* with a name “**GameObjectToFind**” should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

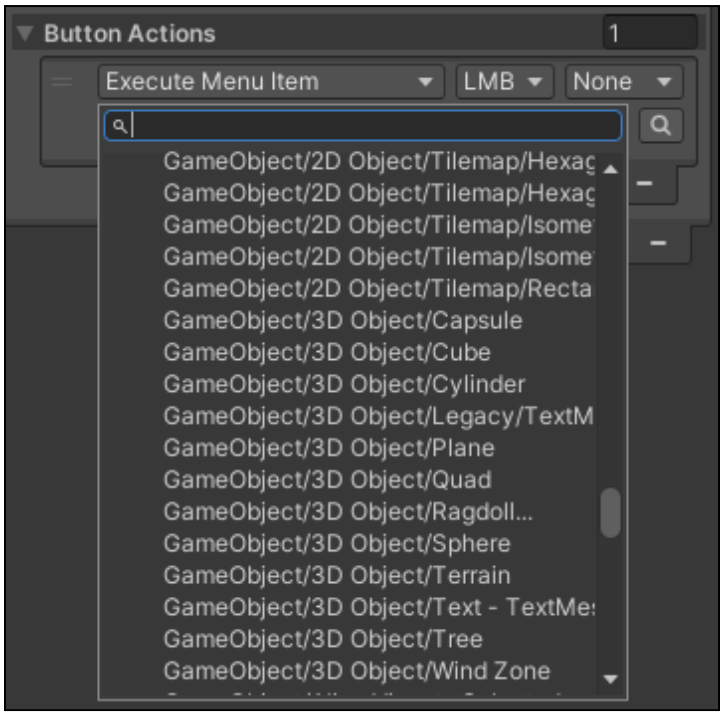
5.3.10 Execute Menu Item

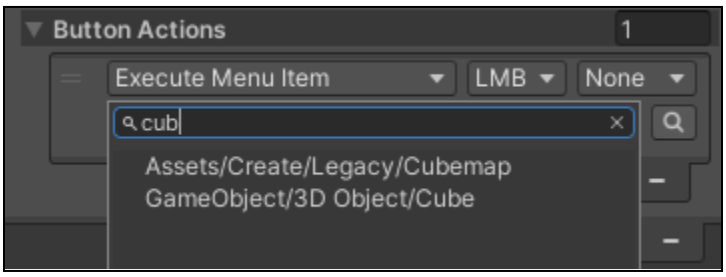
Executes any menu item. Menu items are lists of various actions grouped above the upper toolbar (File, Edit, Create, etc.). For example, you can use such an item to open a **Project Settings** window (**Edit / Project Settings…**) or to create an empty gameObject (**GameObject / Create Empty**). You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

With **Execute Menu Item** action you can invoke any of these actions with a button. 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 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 (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.



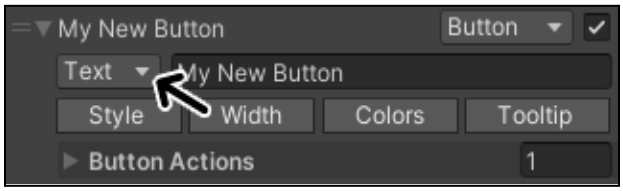


An example of using this type of action is an example button called **A3** (on the buttons list, in the **ExampleButtonList** asset). This button will execute a menu item named **“GameObject / 3D Object / Cube”**, which will create a *gameObject* with *Box Collider* and *Mesh Renderer* components. To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

6. Button Visuals

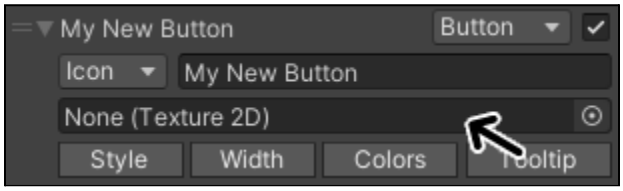
There are many options regarding the visual layer of the button. In this section we will describe each of them.

6.1 Label, Icon, or Both



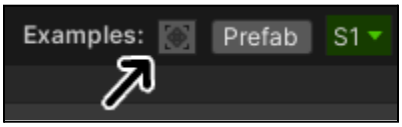
First, decide what will be displayed on your button. You specify this in the first dropdown in the second row. You have the following options here:

- **Text** - the button will display only the text entered in the field next to the dropdown (more on this in section ***3.2 Button Label***). Most example buttons display only text.



- **Icon** - only an icon will be displayed on the button. The image of the icon must be dragged and dropped in the *Texture2D* field below the dropdown and text field.

  An example of a button displaying only an icon is the first example button, marked with an arrow on the image. This button is called **Mouse & Keyboard Buttons** on the **Right Side** list.



- **Both** - the button will display the text entered in the field next to the dropdown and an icon, whose image must be dragged and dropped in the *Texture2D* field (below the dropdown and text field). An example of a button that displays text and an icon is the **Check** button.

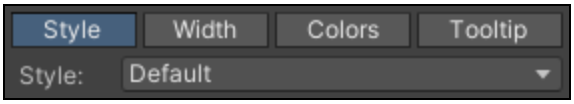
- **None** - nothing will be displayed on the button. Useful for creating empty spaces. The second item in the **Right List**, called **Empty Space**, is an example of using this type.

6.2 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 button on the toolbar has changed colors. But if you turn off the **Colors** toggle, the colors of the button will return to the default ones (even though the colors will still be changed in the button properties).

6.2 Style

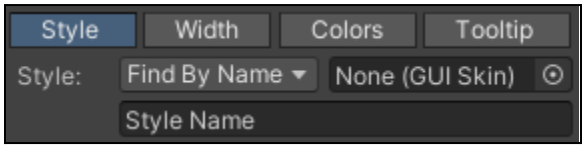


If the **Style** toggle is disabled, the button will use one of the default styles, set separately for the **Button**, **Toggle**, and **Label** types (see section ***4. Settings***). Enable the **Style** toggle to be able to set a style for a button other than the default one.

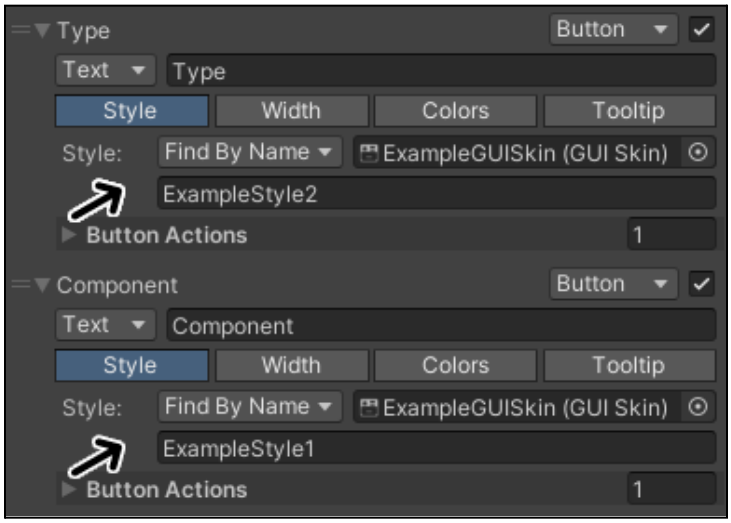
After enabling this toggle, a new row with a dropdown will appear. This dropdown lets you choose a style for the button. The following options are available:

- **Default** - the button will use the default style which can be set in settings (see ***4. Settings*** section).

- **Find By Name** - the button will use the style whose name is entered in the text field that will appear under the dropdown (*"Style Name"* on the image below). Also, after selecting this option, another field will appear next to the dropdown.

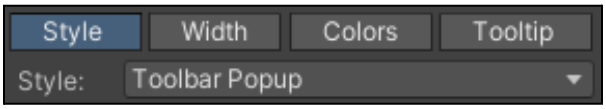
  

  In this field, you can drag and drop your own *GUI Skin* asset (if you do not do this, the style will be searched for in a standard skin - *DarkSkin* or *LightSkin* - depending on which *Editor Theme* you have selected in Unity).

  The **Type** and **Component** buttons are examples of using your own skin and custom styles. The ***ExampleGUISkin*** skin, dragged and dropped into the field next to the style dropdown, 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.



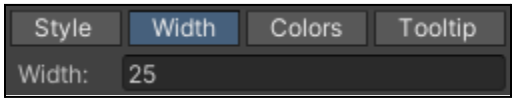
The remaining options of the dropdown are Unity styles. Select one of them to change the style in which the button is presented. For example, the **S1**, **S2**, and **S3** buttons use the *Toolbar Popup* style.

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.

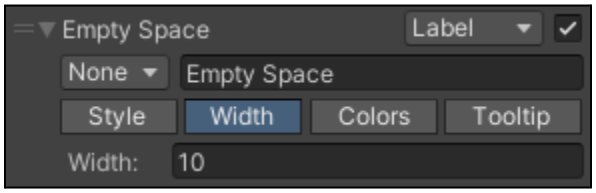
6.3 Width

If the **Width** toggle is disabled, the width of the button will be determined automatically, based on the length of the text the button displays.

If the button displays only the icon or nothing, then the **Width** toggle should be enabled. After enabling it, a new row labeled *Width* will appear. It contains a field where you can enter the width of the button.



This option is used in two example buttons - the one with only an icon (labeled as **Mouse & Keyboard Buttons** on the **Right Side** list), and the one which shows nothing and is used as a space between the **Toolbar** button and the **"Examples:"** label (second item on the **Right Side** list, called **Empty Space**, shown in the image).



6.4 Colors

If the **Colors** toggle is disabled, the button 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:

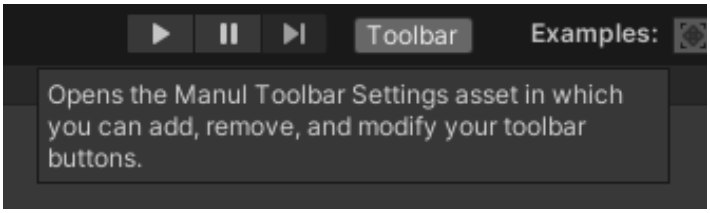
- **First color:** modifies the overall tint of the background and text colors. Equivalent of *GUI.color* (example: **S1** button).
- **Second color:** modifies the tint of the text color. Equivalent of *GUI.contentColor* (example: **S2** button).
- **Third color:** modifies the tint of the background color. Equivalent of *GUI.backgroundColor* (example: **S3** button).

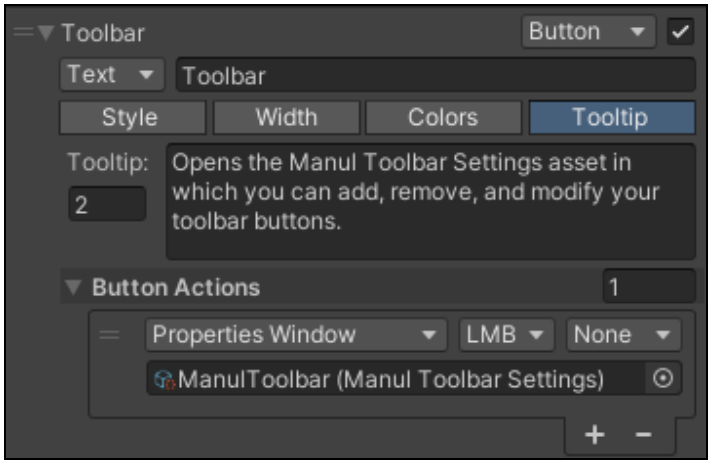
Remember that you tint a color, not change it. This is equivalent to multiplying color by color. For example, in the *Light* skin in Unity the text is black, therefore you will not be able to change its tint using the **Colors** option.

Tip: *The original text color can be changed by using* ***GUIStyles****. More information can be found here.*

6.5 Tooltip

To see additional information about a button when you hover the cursor over it, enable the **Tooltip** toggle. A large text field will appear below, in which you should enter the tooltip text, and a smaller one will appear on the left, specifying the height (in lines) of the first field.

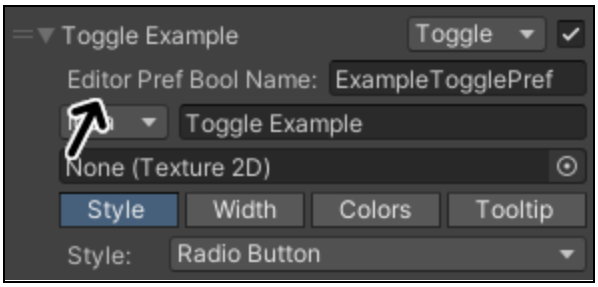




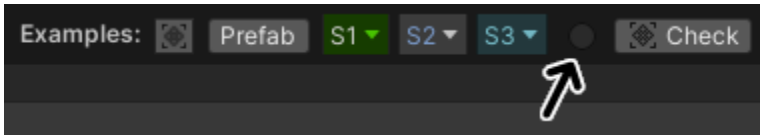
So if you want to enter an extremely long tooltip, change the number of lines to, for example, 10 or more. The field cannot be smaller than 2 lines. An example of using a tooltip is the **Toolbar** button and the example **S3** button.

7. Toggle

Toggle is a switch that can have one of two values: on or off (true or false). If you set the button type to **Toggle** (you change the type in the first line of the button properties item), a line with the **Editor Pref Bool Name** field will appear below it. In this field, enter the name of the **EditorPref** of the *bool* type, which the toggle will be associated with.



The toggle example is the sixth example button, a small empty circle, to the right of **S3** (in the **Right Side** list, this button is called **Toggle Example**).



If you click on it, it will change its value to the opposite value to the one it had before clicking - **on** will change to **off**, and **off** will change to **on**. Moreover, clicking will set the bool value of *EditorPref* called **ExampleTogglePref** to the same value as toggle - **on** to **true**, **off** to **false**.

To check if everything works properly, click the **Check** button next to it. A message will appear in the console informing you about the current value of the **ExampleTogglePref** *bool*.

More information about *EditorPrefs* can be found here.

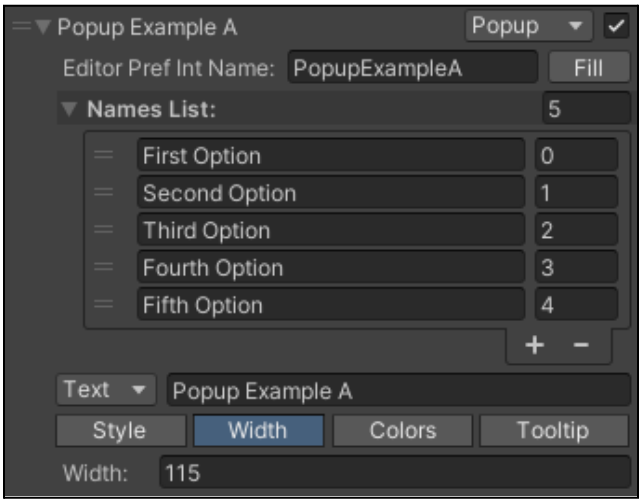
8. Popup

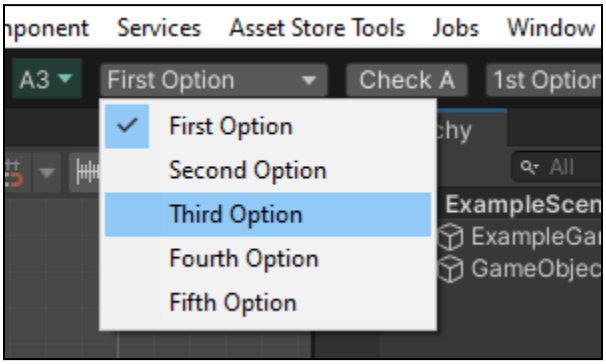
*In order to see the examples for the **Popup** button type, see section **4.4.1 Use Button List Asset** and follow the instructions to override the left toolbar side with the **ExampleButtonList1** asset.*

Popup is a dropdown menu that will set and save an integer value as an *editor pref*. You can see two examples of a popup type on the image below: **First Option** and **1st Option / First Suboption**. You can find their settings on the buttons list in the ***ExampleButtonList*** *asset* (the first one is named **Popup Example A**, and the second is named **Popup Example B**).

To create a popup first select the **Popup** option as a button type (you change the type in the first line of the button properties item). A line with the **Editor Pref Int Name** field will appear below it. In this field, enter the name of the **EditorPref** of the *int* type (e.g. **"PopupExampleA"**), with which the popup will be associated. 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 item 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).

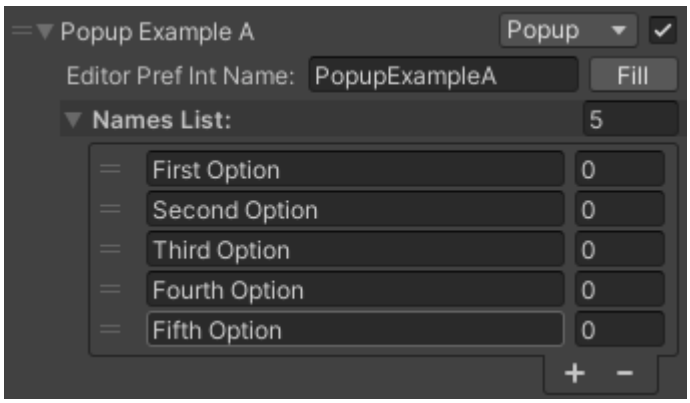


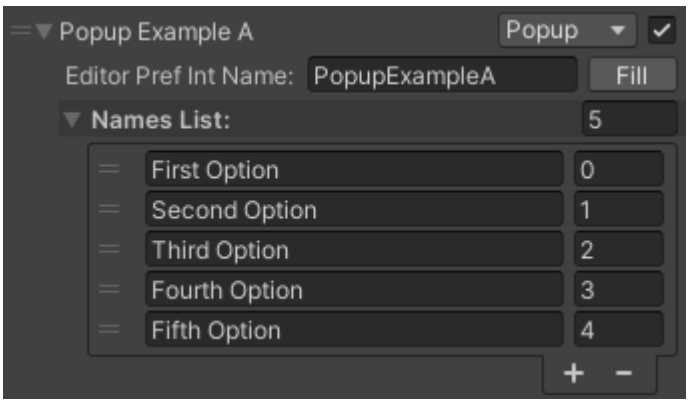


So, if, for example, you select the **Third Option** from the popup dropdown menu and left-click on it, it will set and save a value of 2 for the *editor pref* named **PopupExampleA**.

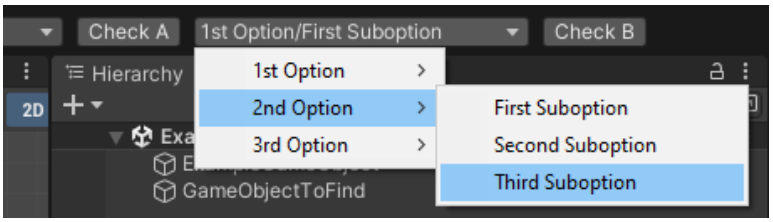
You can check this value by clicking the **Check A** button (next to the first popup dropdown). A message will appear in the console informing you about the current value of the **PopupExampleA** (it should be 2).

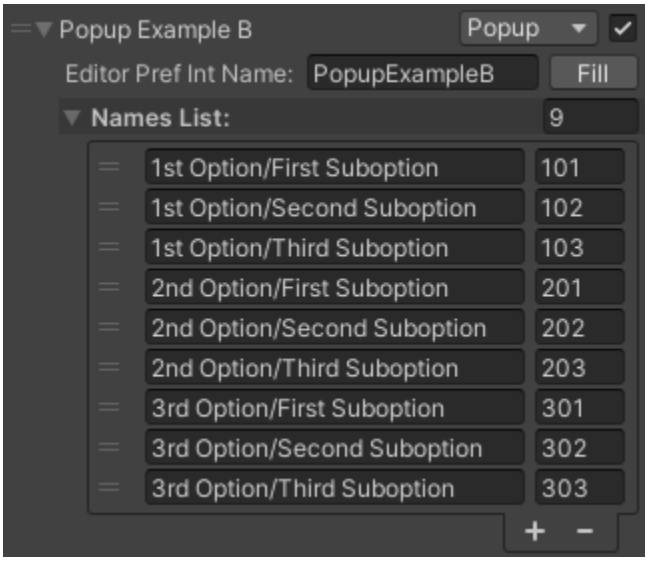
When you create a list of items for the dropdown you have to specify an integer value for each item. You can do this manually or click on the **Fill** button. It will automatically put a number in each item (starting with index 0). To see how this button works, first manually change all numbers to 0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





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 button with **Label** type.

More information about *EditorPrefs* can be found here.

9. Shortcuts

9.1 Opening MANUL Toolbar Settings

You can find the **Manul Toolbar Settings** asset in this default location: **Assets / Liquid Glass Games / Manul Toolbar / Resources / ManulToolbar.asset**. You can also use menu items to open it:

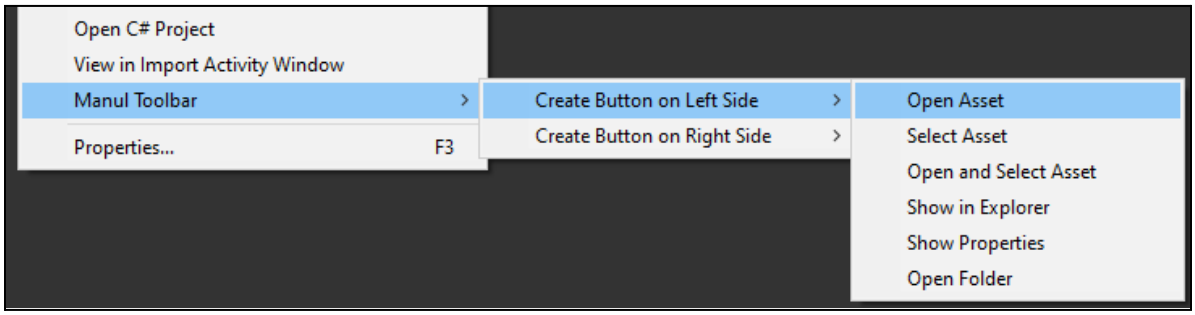
- **Tools / Manul Toolbar / Open Manul Toolbar Settings** - opens the **MANUL Toolbar** properties in the new property window.

- **Tools / Manul Toolbar / Select Manul Toolbar Settings** - opens the **MANUL Toolbar** properties in the *Inspector* window by selecting this asset in the *Project* window.

9.1 Menu Items in Project Window

You can quickly create a button corresponding with any asset in the *Project* window by using menu items. Right-click on the asset, select **Manul Toolbar** and choose on which side of the toolbar your new button will appear (**Create Button on Left Side** or **Create Button on Right Side**). 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 items will create a button and add its settings to the left or right side list of buttons (in the **Manul Toolbar Settings** asset).

If you have overridden a list with a **Manul Toolbar Buttons List**, your new button will be added to this asset (see section ***4.4.1 Use Button List Asset*** for details).

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

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

9.2 Change Button Name

You can quickly change the name of a button by left-clicking on it while holding 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 **Label**, **Toggle**, and **Popup** types.

9.3 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 **Label**, **Toggle**, and **Popup** types.

10. 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 or create a post on the Unity Discussions Forum here.