MK Studio

Easy Message Box

Documentation (V1.4)

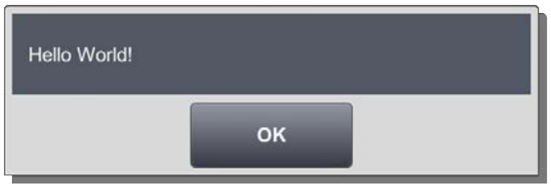
Quick Start

Thank you for buying Easy Message Box! It is a simple, versatile and highly customizable message box system based on Unity's new UI system. Let's get started!

The Setup-Free Way

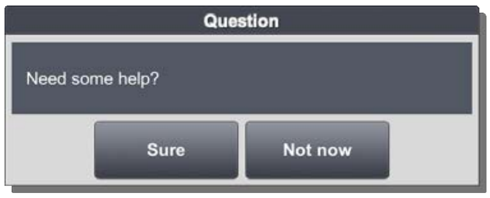
You are ready to call `EasyMessageBox.Show()` any time in your script once you have imported the package into you project, **no extra setup needed!**

```
EasyMessageBox.Show("Hello World!");
```



This method has many parameters you can specify to customize it. However, by using the named and optional arguments features of C#, you don't have to provide all of them or provide them in the order defined by the method. For example:

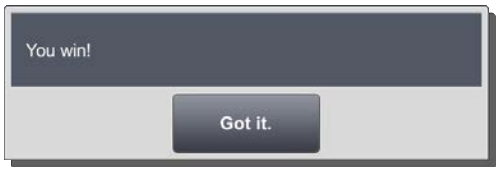
```
EasyMessageBox.Show("Need some help?",
                    messageBoxTitle:"Question",
                    button1Text:"Sure",
                    button1Action: () => { Debug.Log("Yes!"); },
                    button2Text:"Not now");
```



*Note: The method's parameters are fully commented in the source code.

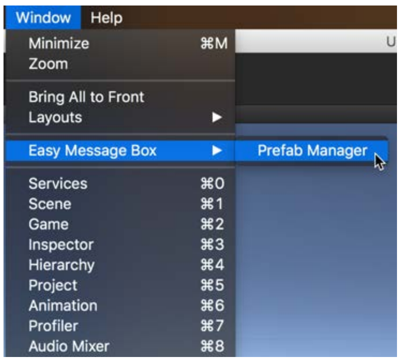
Alternatively, you can create a new instance of the `MessageBoxParams` class, set the parameters of the instance and then pass it to the `EasyMessageBox.Show()` method. For example:

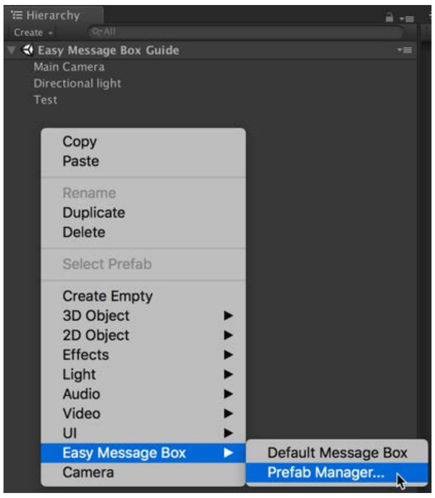
```
MessageBoxParams param = new MessageBoxParams();
param.Message = "You win!";
param.Button1Text = "Got it.";
EasyMessageBox.Show(param);
```

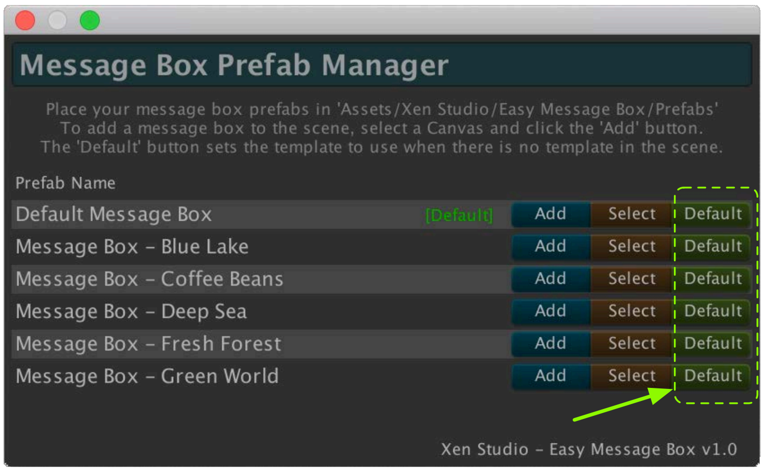


Changing the Default Message Box

By default, Easy Message Box will use the `'Default Message Box.prefab'` under the folder `'Assets/MKStudio/Easy Message Box/Resources/'` as the template when using the 'setup-free way'. To use another prefab, open the `Prefab Manager` (`'Window->Easy Message Box->Prefab Manager'` or right-click in the Hierarchy then `'Easy Message Box->Prefab Manager'`) and set the default prefab by its `'Default'` button.







Message Box Prefab Manager

Place your message box prefabs in 'Assets/Xen Studio/Easy Message Box/Prefabs'
To add a message box to the scene, select a Canvas and click the 'Add' button.
The 'Default' button sets the template to use when there is no template in the scene.

Prefab Name				
Default Message Box	[Default]	Add	Select	Default
Message Box – Blue Lake		Add	Select	Default
Message Box – Coffee Beans		Add	Select	Default
Message Box – Deep Sea		Add	Select	Default
Message Box – Fresh Forest		Add	Select	Default
Message Box – Green World		Add	Select	Default

Xen Studio – Easy Message Box v1.0

Using Easy Message Box without setting up the scene is the quickest way to show a message box in any scene. However if you need more customization or using multiple templates in the same scene, it is recommended that you do some setup before using it. Now let's look at the second way of using Easy Message Box.

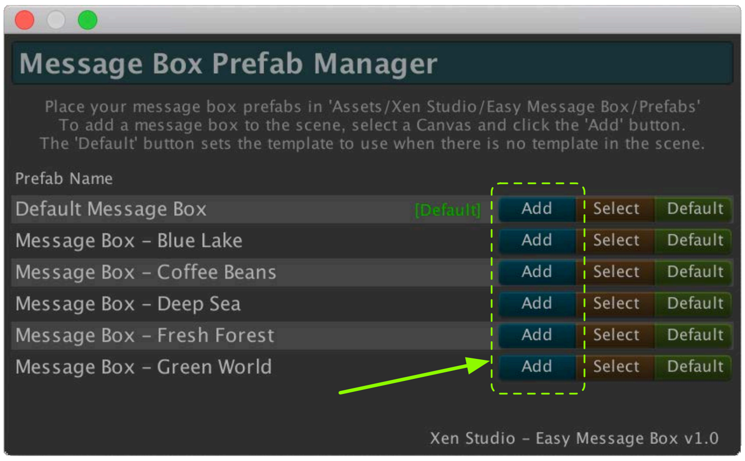
The Setup-Before-Use Way

The second way of using Easy Message Box requires some setup. However it allows you to easily customize the message box as well as using multiple templates in the same scene.

1. In the scene you want to display the message box, select the Canvas you want the message box to be displayed on.

*Note: If there is no Canvas in the scene, add one by the menu '*GameObject->UI->Canvas*' or right-click the Hierarchy then select '*UI->Canvas*'.

2. Open the Prefab Manager (`Window->Easy Message Box->Prefab Manager` or right-click the Hierarchy and choose `'Easy Message Box->Prefab Manager'`), choose a prefab and click the "Add" button.


Message Box Prefab Manager

Place your message box prefabs in 'Assets/Xen Studio/Easy Message Box/Prefabs'
To add a message box to the scene, select a Canvas and click the 'Add' button.
The 'Default' button sets the template to use when there is no template in the scene.

Prefab Name				
Default Message Box [Default]	Add	Select	Default
Message Box - Blue Lake	Add	Select	Default
Message Box - Coffee Beans	Add	Select	Default
Message Box - Deep Sea	Add	Select	Default
Message Box - Fresh Forest	Add	Select	Default
Message Box - Green World	Add	Select	Default

Xen Studio - Easy Message Box v1.0

3. Customize the added message box to meet your requirement. For example, it's size, background, font, etc. A detailed guide on how to customize the message box is provided in the later part of this document.

*Note: To change the message box's size, you can use the *Message Box Scaler* slider on the *BoxController* component or use it's parent Canvas's *Canvas Scaler* (for details, see this section).

4. Once you are happy with the result, simply disable the message box (the one with the `BoxController` component.) and that's it! This customized message box will be used the next time you call **EasyMessageBox.Show();**.

*Note: To use multiple templates in the same scene, see this section.

Adding Prefabs to the Prefab Manager

Easy Message Box ships with several message box prefabs. You can easily add your own prefabs to the Prefab Manager and reuse them across the scenes or in other projects.

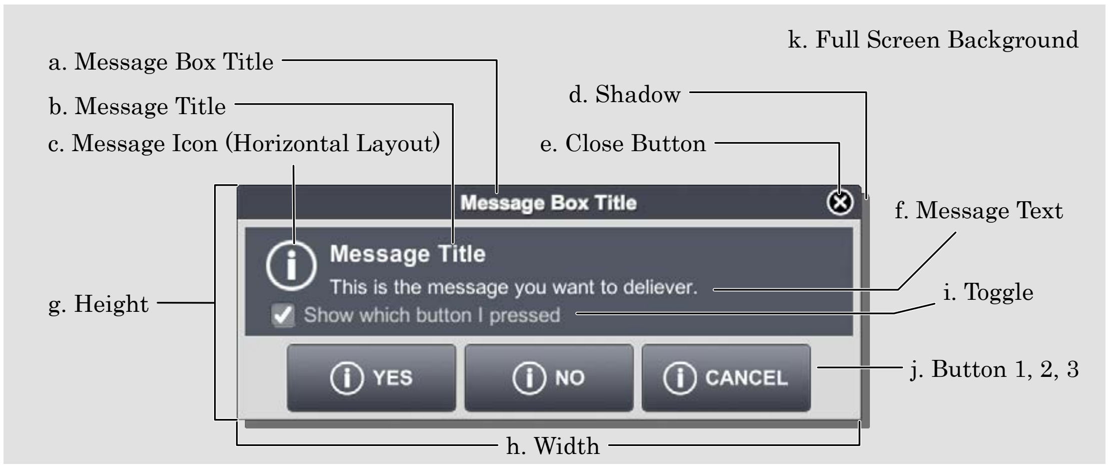
To add your own message box prefabs to the Prefab Manager, simply drag your customized message box (Do Not include the Canvas) from the Hierarchy to the folder `'Assets/MKStudio/Easy Message Box/Prefabs'` to create a prefab. Then they will be displayed in the list and ready to be added just like built-in prefabs.

Customization Guide

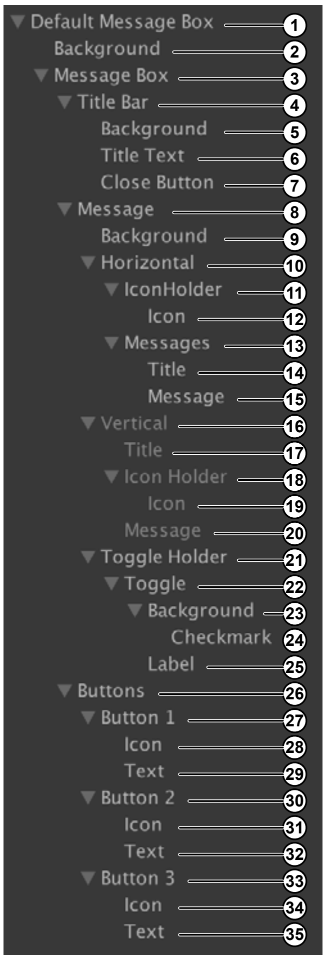
Because modifying UI elements can be messy sometimes, a detailed guide on how to modify each part is provided. Refer to this section if you encounter problems when customizing the looks.

Customizing UI Elements

Here is how a message box template is structured:



The following list shows which part in the hierarchy to modify if you want to change a specific element in the above message box:



```
▼ Default Message Box  1
    Background  2
  ▼ Message Box  3
    ▼ Title Bar  4
        Background  5
        Title Text  6
        Close Button  7
    ▼ Message  8
        Background  9
      ▼ Horizontal  10
        ▼ IconHolder  11
            Icon  12
        ▼ Messages  13
            Title  14
            Message  15
      ▼ Vertical  16
          Title  17
        ▼ Icon Holder  18
            Icon  19
          Message  20
      ▼ Toggle Holder  21
        ▼ Toggle  22
          ▼ Background  23
              Checkmark  24
            Label  25
    ▼ Buttons  26
      ▼ Button 1  27
          Icon  28
          Text  29
      ▼ Button 2  30
          Icon  31
          Text  32
      ▼ Button 3  33
          Icon  34
          Text  35
```

a. **Message Box Title**. The Text component on ⑥. There is also a Shadow and an Outline component on ⑥ to achieve these effects.

b. **Message Title.** The Text component on ⑭. To adjust the distance between the Message Title and the Message Text, modify the Spacing of the Vertical Layout Group component on ⑬.

c. **Message Icon.**

- To change the size of the icon area, modify the Min/Preferred Width/Height of the Layout Element component on ⑪.
- To change the size and position of the icon itself, modify the rect transform of ⑫.
- To adjust the distance between the icon and the messages, modify the Spacing of the Horizontal Layout Group component on ⑩.

d. **Shadow.** Enable/disable the shadow by enabling/disabling the Shadow component on ③.

e. **Close Button.** In the hierarchy its ⑦. No need to disable it manually if you want to hide it, it's controlled via script.

f. **Message Text.** The Text component on ⑮.

g. **Height.** The message box's height is adaptive to its content (including the title bar, the message and the buttons), no need to change it manually.

h. **Width.** The message box's width has a preferred value which can be adjusted by changing the Preferred Width of the Layout Element component on ⑭ and ⑮. It uses the larger value of the two, so you may need to change both.

i. **Toggle.** You can custom it through ㉑ to ㉕. Note that you don't need to disable/enable it manually since it's controlled via script.

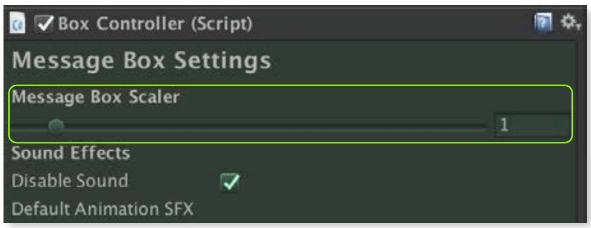
j. **Buttons.** The message box contains three buttons that you can customize through ㉖ to ㉟. No need to disable/enable any of them since they are controlled via script.

k.**Full Screen Background / Blocking other UI elements.** This is the background image that will fill the full screen when the message box is shown. It is also responsible for blocking other UI elements' interaction while a message box is on the screen. By default it has a dark translucent color, you can change it by changing the Color value of the Image component on ② in the hierarchy. *Note: Setting the Color's alpha to 0 will make it completely transparent while still being able to block other UI. If you don't want the message box to block other UI elements (i.e. being a non-modal message box), simply uncheck the Raycast Target toggle.*

Changing the Overall Size of the Message Box

To change the overall size of the message box (scaling it up or down), you can use one of the following two methods:

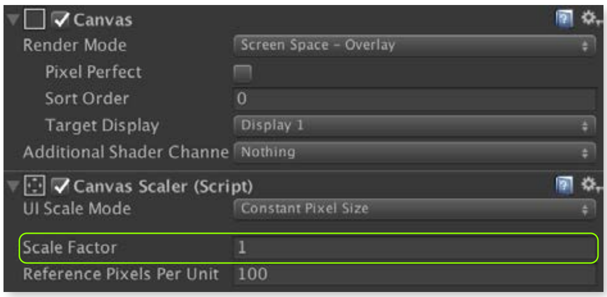
1. Use the `Message Box Scaler` slider in the Box Controller component's inspector of ①.



Box Controller (Script)
Message Box Settings
Message Box Scaler 1
Sound Effects
Disable Sound ✓
Default Animation SFX

*Note: Manually setting the message box panel's scale will not work because there is an Animator that takes control of the local scale of the actual message box panel.

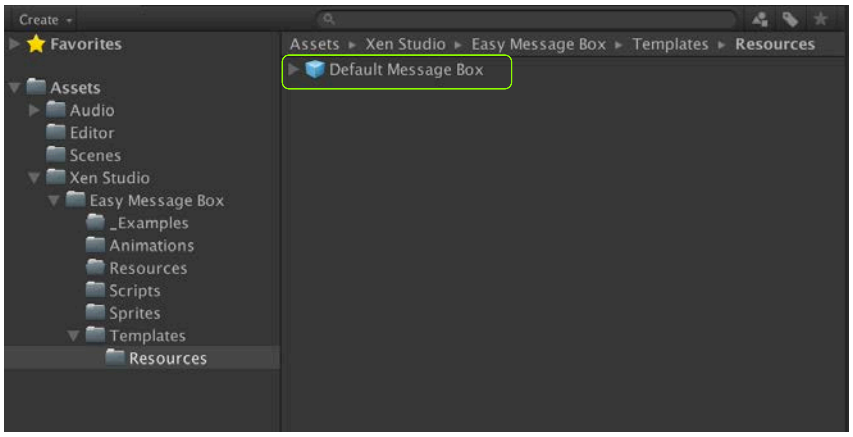
2. If there is a `Canvas Scaler` on the Canvas on which the message box will be displayed, you can use its Scale Factor value to scale the message box. Note that this will also affect other UI elements on the same Canvas.



Canvas
Render Mode: Screen Space - Overlay
Pixel Perfect
Sort Order: 0
Target Display: Display 1
Additional Shader Channe: Nothing
Canvas Scaler (Script)
UI Scale Mode: Constant Pixel Size
Scale Factor: 1
Reference Pixels Per Unit: 100

Customizing the Default Message Box

The default message box prefab is inside the folder `'Assets/MKStudio/Easy Message Box/Resources'` instead of the prefab folder. You can customize it like other prefabs, just make sure it's file name is exactly `"Default Message Box.prefab"`, otherwise it will not be loaded correctly.



Assets ▸ Xen Studio ▸ Easy Message Box ▸ Templates ▸ Resources
Default Message Box

*Note: This prefab is used when you choose '*GameObject->Easy Message Box->Default Message Box*' to create a default message box template, it is highly recommended that you **Backup This Prefab Before Modifying Or Replacing It**. Otherwise you have to reimport the package to get a fresh one if it's broken.

Other Features

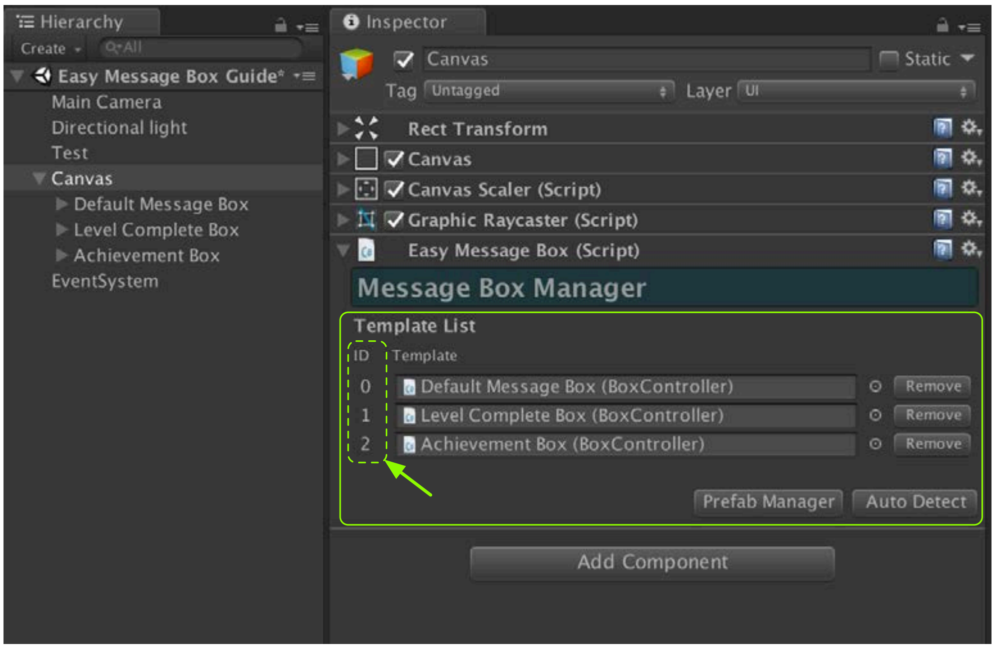
Using Multiple Templates in the Same Scene

Easy Message Box allows you to use multiple templates in the same scene. This feature is useful when you want to use your message box for different purposes. For example you can config a message box template as the regular message box to show information or let the player make simple choices, another template as the 'Level Complete Screen' and another template to show achievements and let the player claim the rewards.
*Note: For an example of this, check the scene '*Example_WithSetup*' included in the package.

To use multiple templates in the same scene, just use 'The Setup-Before-Use Way' and add the prefabs you need to the same Canvas using the Prefab Manager. After that, you can show a message box based on a certain template by specifying its corresponding template id. For example:

```
EasyMessageBox.Show("Level Complete!",
                     button1Text:"Home",
                     button2Text:"Restart",
                     button3Text:"Share",
                     templateId: 1);
```

You can check the `'ID'` of each template in the `'Template List'` of the `Easy Message Box component` on the Canvas.



*Note: If you have added a message box template manually instead of using the Prefab Manager (for example duplicating an existing message box in the Hierarchy), you can include it in the Template List by pressing the *Auto Detect* button.

Automatic Queuing

By default, a new instance of a message box will be displayed instantly when you call `EasyMessageBox.Show()`. However, it is sometimes very handy to use a message box mechanism that can queue itself when additional calls to show a message box happen before the player finishes the interaction of the current one. For example when the player completed a level and earned more than one achievement where each achievement has a reward the player can claim, it may be more exciting to let the player know and claim the rewards one by one rather all at once. With the auto-queue feature of Easy Message Box, you don't need to detect whether the player has completed previous message box's interaction, just call them all at once and let Easy Message Box do the queuing for you. The queued message box will popup once the previous one is closed.

To enable the auto-queue feature, simply provide the `multipleCallBehaviour` parameter, set its value to `MultipleCallBehaviours.Queue` and that's it.

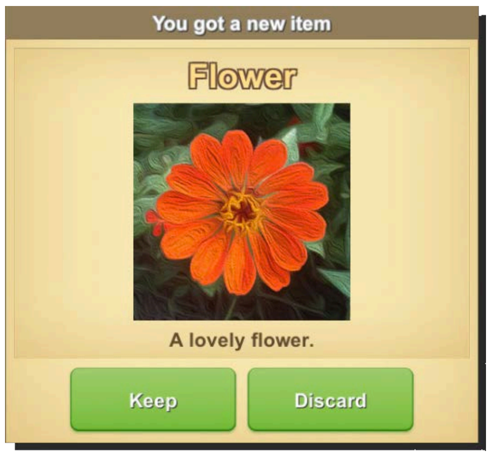
For the above example, you can do something like this:

```
foreach (var achievement in achievementList){
            if (achievement.condition.IsMet)
            {
                EasyMessageBox.Show("Achieved: " + achievement.name,
                       button1Icon: diamondIcon,
                       button1Text: "100",
                       multipleCallBehaviour: MultipleCallBehaviours.Queue);
            }
        }
```

Vertical Layout

Easy Message Box has a built-in vertical layout to display an 'icon-focus' style dialog box. Just provide the parameter `'layout'` and set its value to `'MessageLayout.Vertical'`. For example:

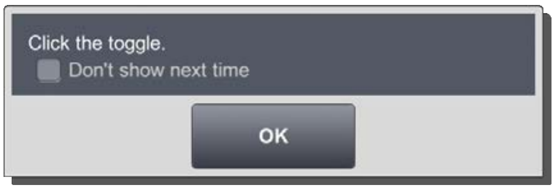
```
EasyMessageBox.Show("A lovely flower.", messageTitle: "Flower",
                     layout: MessageLayout.Vertical,
                     messageBoxTitle: "You got a new item",
                     button1Text: "Keep", button2Text: "Discard",
                     messageIcon: flowerIcon);
```



Built-in Toggle

Easy Message Box has a built-in toggle that can be optionally displayed. To show the toggle, simply provide the 'toggleText' parameter as the toggle's label and 'toggleAction' to handle the toggle's 'OnValueChanged' event. For example:

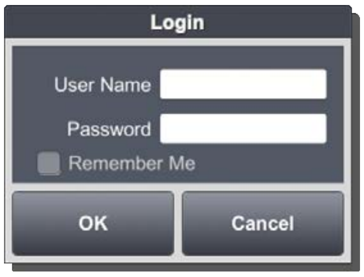
```
EasyMessageBox.Show("Click the toggle.", toggleText: "Don't show next time",
                   toggleAction: (x) =>
                      {
                            if (x) { EasyMessageBox.Show("Checked."); }
                            else { EasyMessageBox.Show("Unchecked."); }
                      });
```



Input Fields

Easy Message Box has implemented **two** built-in input fields that can be optionally displayed. This is very convenient if you want to display a message box that can collect the player's input, for example a Login window. For example:

```
  EasyMessageBox.Show("", button2Text: "Cancel",
          inputField1Label: "User Name", inputField2Label: "Password",
          inputFieldAction: (x1, x2) =>
          { EasyMessageBox.Show("User name: " + x1 + " Password: " + x2); },
          inputFieldActionsFireOnButton: Buttons.Button1,
          toggleText: "Remember Me", messageBoxTitle: "Login");
```



Displaying or Hiding the Input Fields

The two built-in input fields can be individually controlled by providing the parameter 'inputField1Label' and 'inputField2Label'. If the string you provide is not 'null' or an empty string, the corresponding input field will be displayed, otherwise it will not be displayed.

Retrieving the Entered Strings

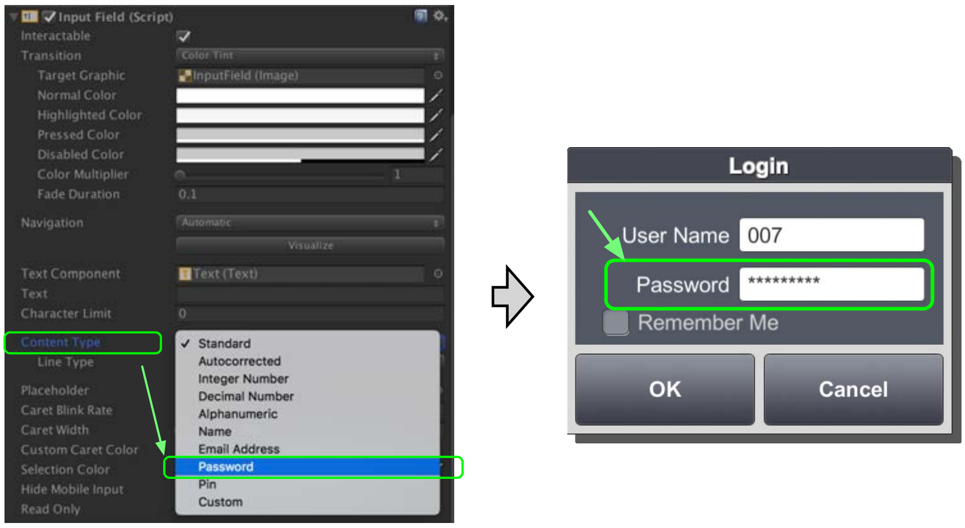
The texts entered by the player can be retrieved by the 'inputFieldAction' provided as the parameter. Note that this action always requires two string parameters (the first one for Input Field 1, the second one for Input Field 2) regardless of the number of input fields actually being showed.

Specifying the Button to Trigger the Input Field Action

The inputFieldAction provided will be fired once a main button is pressed by the player. You can specify which button will be responsible for firing this button by providing the '`inputFieldActionsFireOnButton`' parameter. For instance in the above example, only the 'OK' button (Button 1) will fire the action, and the 'Cancel' button (Button 2) will not.

Displaying the Input Text as Password

To display the input field's text as password, set the '`Content Type`' property of the input field itself. Note that you may need to use the 'setup-before-use' method.



Input Field (Script)
Interactable ✓
Transition: Color Tint
Target Graphic: InputField (Image)
Color Multiplier: 1
Fade Duration: 0.1
Navigation: Automatic
Text Component: Text (Text)
Character Limit: 0
Content Type: ✓ Standard, Autocorrected, Integer Number, Decimal Number, Alphanumeric, Name, Email Address, Password, Pin, Custom

Login
User Name: 007
Password: *********
Remember Me

Animations

Easy Message Box includes several built in animations when a message box is appearing and disappearing. These animations are made with Unity's own Animator and Animation system thus requires no 3rd party tween systems. You can simply provide the `'inAnimation'` and `'outAnimation'` parameters to activate corresponding animation. For example:

```
EasyMessageBox.Show("This is a message box with animation.",
                                   inAnimation: InAnimationTypes.Spin,
                                      outAnimation: OutAnimationTypes.Fade);
```

Blocking or Unblocking other UI Elements

To block or unblock other UI elements while a message box is showing, manipulate its full screen background image's `Color` and `Raycast Target` property. The detail is described in this section.

Force Close All the Message Boxes

If you want to close all the on-screen Message Boxes immediately, you can call

```
EasyMessageBox.ForceCloseAllMessageBoxes();
```

Note that this will also clear the queue if automatic queuing is used.

Thank you for choosing Easy Message Box!

- *For questions, suggestions and feature requests, send email to lx84@outlook.com.*
- *Please give us a good review if you like it! It means a lot to us, thank you!*