EaseUI

USER'S MANUAL (v1.1)

Author: Felipe Ramos
Contact: felipeberquo@gmail.com
EaseUI website: http://felipegamedev.com/easeui

SUMMARY

1. What is EaseUI?

EaseUI is a plugin for Unity3D that allows the user to easily apply custom transition effects on his game user interface (UI), like smooth or bouncing movements, for example. To achieve such effects, EaseUI makes use of easing functions, which specifies the rate that some parameter (position, angle or scale) changes over time.

With EaseUI, the user can translate, rotate and scale the UI elements in several different ways, by just adding the EaseUI component to the game object containing an UI element, and setting a few parameters of his choice. These parameters are used to customize the UI's effects, allowing the users to set initial and final points (position, rotation or scale), the duration of the effect and the easing function of his choice. These parameters can be changed via script, too.

EaseUI makes use of the awesome C# Robert Penner's easing equations (http://robertpenner.com/easing/).

NOTE: If you are an intermediary-advanced Unity's user, and you already know how to install plugins in Unity3D and use them, you can skip to the section 7, ***Quick start***.

2. Requirements

EaseUI plugin has only one requirement: Unity 4.6 or higher. This is because EaseUI is based on the new Unity's UI system that came up since the 4.6 version.

3. Installation instructions

To install EaseUI, you need to import the **Plugins** and **Editor** folders to your project with its contents. You just have to make sure that the **EaseUI.dll** file is inside the **Plugins** folder and the **EaseUIEditor.dll** file is inside the **Editor** folder. If you don't know how to do such thing, please refer to Unity's documentation website:

http://docs.unity3d.com/Manual/HOWTO-InstallStandardAssets.html

If the installation succeeds, you should see an option to add EaseUI components to your object, inside the **Component** > **UI** tab in your editor (Figure 1).

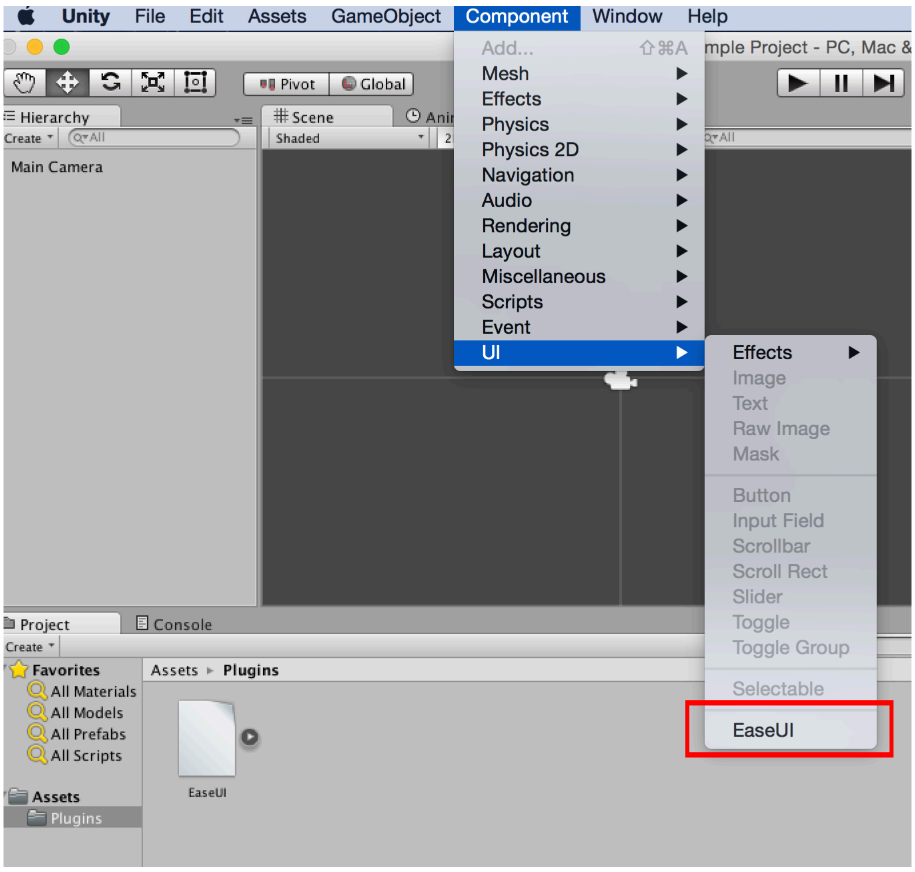


Figure 1 – If you see this option in your editor, it means that you're ready to rock! :)

4. EaseUI component overview

First, you have to add an EaseUI component to the game object, already containing an Unity's UI component (buttons, texts, etc.), so you can customize its effects parameters the way you want. There are two ways to add EaseUI to your **selected** game object: you can use the tab **Component** > **UI** in the editor's top menu bar (Figure 5), or pressing the **Add Component** button in the bottom of the game object's inspector, and then select the **UI** option (Figure 6 and 7).

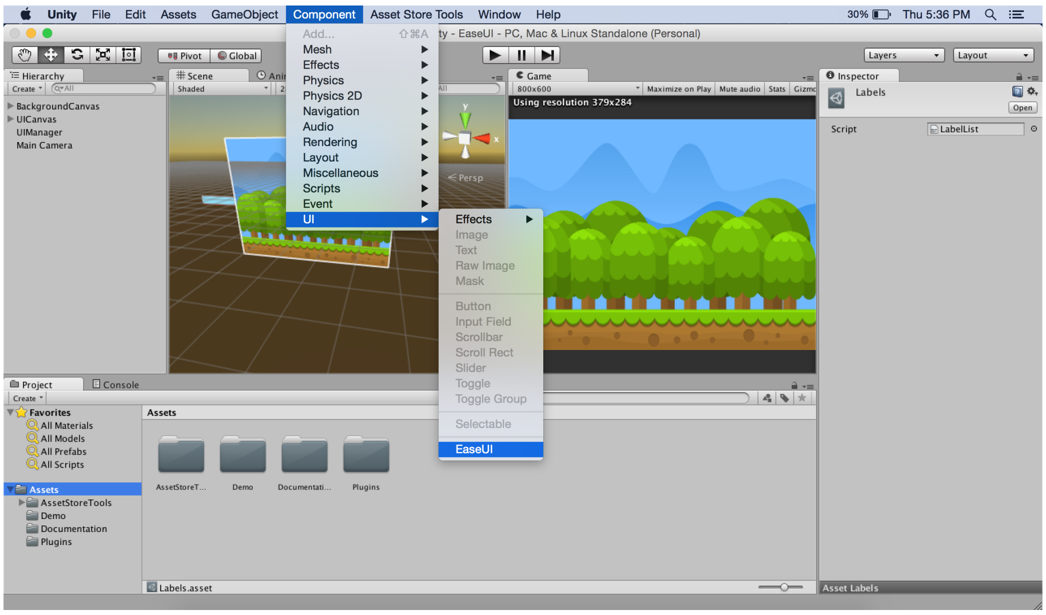


Figure 5 – Adding the EaseUI component through the editor's top menu.

NOTE: If you don't know how to create an UI component in Unity3D, please take a look at the Unity's documentation website (http://docs.unity3d.com/Manual/UISystem.html).

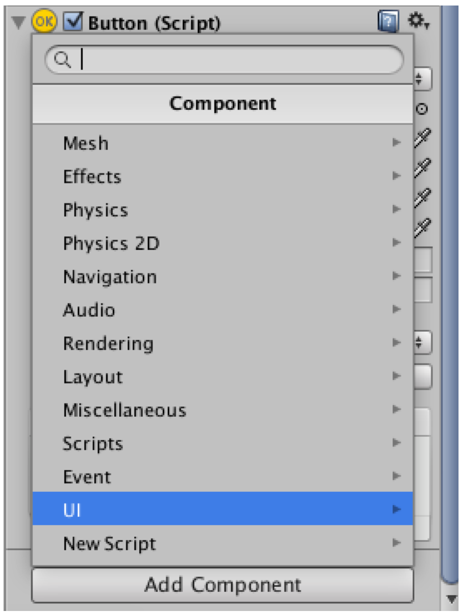


Figure 6 – To find the EaseUI's component under the inspector, you first have to press **Add Component** button and select the **UI** option.

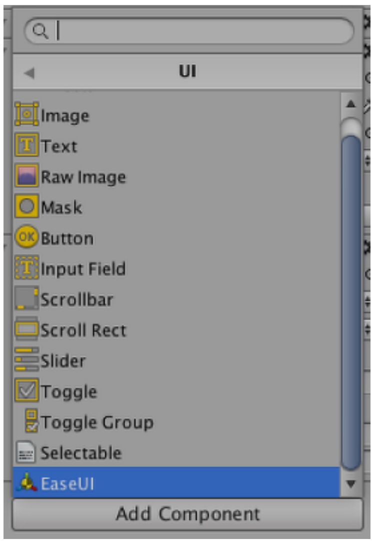
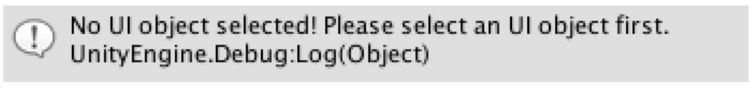


Figure 7 – You will find the **EaseUI** option in the bottom of the components list.

When you try to add EaseUI using the editor's top menu, you might have end up with two warning messages at your console output.

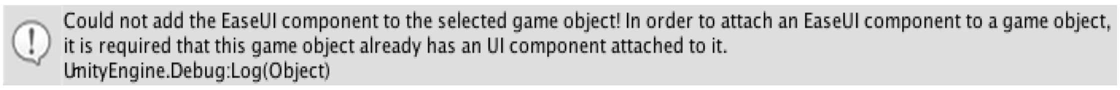
The first warning (Figure 8) shows up when there is no game object selected in your scene and you try to add an EaseUI component, so the plugin doesn't know where to attach the component.



```
No UI object selected! Please select an UI object first.
UnityEngine.Debug:Log(Object)
```

Figure 8 – Warning 1: you have to select a game object in your scene.

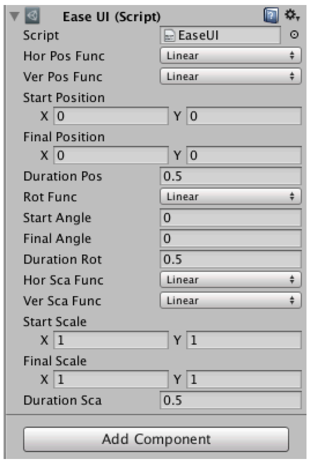
The second warning (Figure 9) is shown when there is a game object selected, but it doesn't have an Unity's UI component attached to it. **Actually, the plugin checks if there is a *Canvas Renderer* component attached to the game object that you are trying to add the EaseUI component, considering that a game object is an UI's object only if it has a *Canvas Renderer* component.**



```
Could not add the EaseUI component to the selected game object! In order to attach an EaseUI component to a game object,
it is required that this game object already has an UI component attached to it.
UnityEngine.Debug:Log(Object)
```

Figure 9 – Warning 2: the selected object you are trying to attach the EaseUI component has no ***Canvas Renderer*** component.

When you add an EaseUI component to a game object, you will notice that there are some parameters to be set in the inspector (Figure 10).



```
Ease UI (Script)
Script          EaseUI
Hor Pos Func    Linear
Ver Pos Func    Linear
Start Position
  X 0             Y 0
Final Position
  X 0             Y 0
Duration Pos    0.5
Rot Func        Linear
Start Angle     0
Final Angle     0
Duration Rot    0.5
Hor Sca Func    Linear
Ver Sca Func    Linear
Start Scale
  X 1             Y 1
Final Scale
  X 1             Y 1
Duration Sca    0.5
```

Figure 10 – Once you attach the EaseUI component to the game object, it will be easy to customize your effects just by changing the parameters in the inspector.

The first one is the **Script**, that tells us which script from the plugin is being used. You don't need to do anything with it, just leave it as it is.

As stated in Section 1, EaseUI is capable of move, rotate and scale your user's interface items, and because of that, the parameters are divided into 3 parts: translating, rotating and scaling. Each part starts with the easing function that is going to be used to apply the effects on your UI. For both translation and scaling parts, you can set 2 functions: one for the x-axis (***Hor*** prefix) and another for the y-axis (***Ver*** prefix). Note that you can set different functions to each axis, allowing you to create more customized effects. The rotation part has only one function to set, which tells how your UI component is going to rotate **along the z-axis**.

The second parameters of each part that you have to set are the start and final points. For the translation effect, these are the start and final positions; for translation, start and final angle of the rotation; and for the scaling, they are the start and final scale.

The last parameter is the same for all parts: the duration of the transition, in seconds.

5. EaseUI class overview

After the EaseUI's parameters in the inspector are set, you just have to call two methods for each type of effect you are going to use - translation, rotation or scaling – in order to control the UI component. The EaseUI's class is inside the ***EaseTools*** namespace, which means that **you have to use the "using EaseTools" statement before trying to access EaseUI's public methods and attributes.**

Basically, to control the UI's effects, you only have to use the ***In*** and the ***Out*** methods listed below:

- **Translation methods**:
    - ***MoveIn( )*** – **translates** the UI's component **from** **StartPosition** **to** **FinalPosition** within **DurationPos** seconds.
    - ***MoveOut( )*** – **translates** the UI's component **from** **FinalPosition** **to** **StartPosition** within **DurationPos** seconds.

- **Rotation methods**:
    - ***RotateIn( )*** – **rotates** the UI's component **from** **StartAngle** **to** **FinalAngle** within **DurationAng** seconds.
    - ***RotateOut( )*** – **rotates** the UI's component **from** **FinalAngle** **to** **StartAngle** within **DurationAng** seconds.

- **Scaling methods**:
    - ***ScaleIn( )*** – **scales** the UI's component **from** **StartScale** **to** **FinalScale** within **DurationSca** seconds.
    - ***ScaleOut( )*** – **scales** the UI's component **from** **FinalScale** **to** **StartScale** within **DurationSca** seconds.

All the methods mentioned above don't return any value.

You can call those methods in a script that controls your game's user interface, like a manager, for example. For that, you have to declare an EaseUI variable in this script (**UIManager.cs**, for the example below) and use its methods. The following sample code shows how to do this:

```
using UnityEngine;
using EaseTools;    // Remember this
using System.Collections;

public class UIManager : MonoBehaviour
{
        // Give whatever name you want to your variable,
        // this is just an example.
        public EaseUI easeUIComponent;

        // You can call the methods wherever you want,
        // not just inside the Update.
        void Update( )
        {
                if (" The UI's element should 'move in' ")
                        easeUIComponent.MoveIn( );
                else if (" The UI's element should 'come out' ")
                        easeUIComponent.MoveOut( );
        }
}
```

Code sample 1 – Example of how you could use the EaseUI's class methods.

Also, there are other three public methods that you can use to check if the element is moving, rotating or scaling:

- ***IsMoving( )*** - determines whether the element is moving.
- ***IsRotating( )*** - determines whether the element is rotating.
- ***IsScaling( )*** - determines whether the element is scaling.

Guess what? They all return **booleans**. :)

6. Using the demonstration scene

The EaseUI's package that you have downloaded from the Unity's Asset Store contains a demonstration scene, inside the ***Demo*** folder. This scene is intended to help you to understand how the plugin works.

When you open the demo, it should look like this:

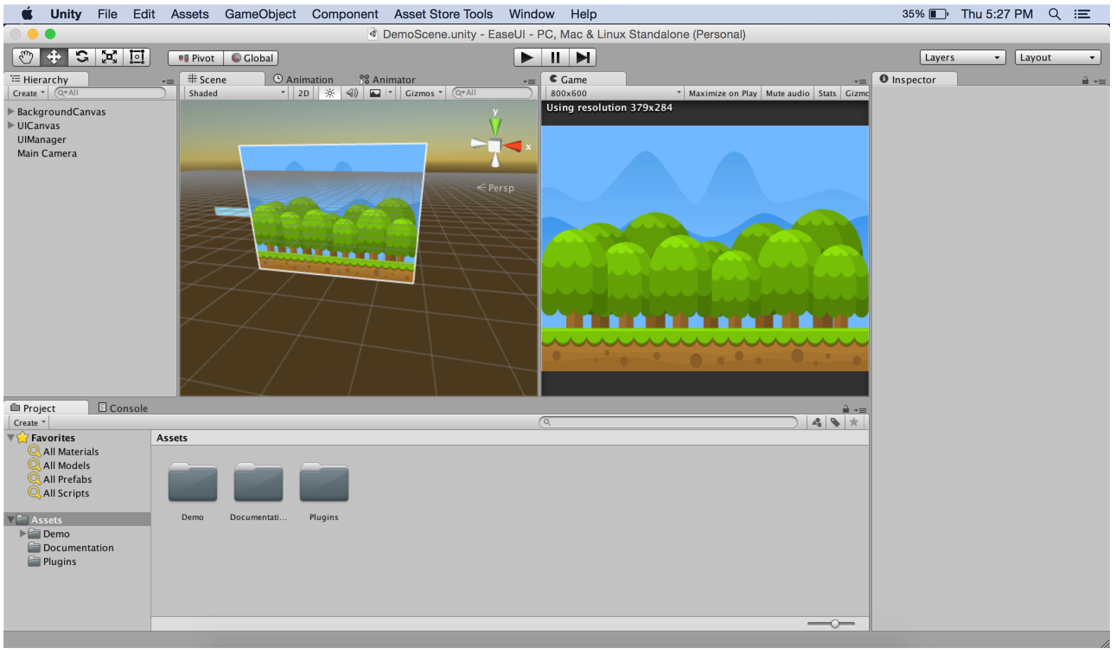


Figure 11 – Demonstration scene overview.

If you look at the scene's hierarchy, you will see a game object called ***UICanvas***, which contains the canvas for rendering the user interface (UI) of the scene.

"The Canvas is the root component for rendering all UI objects in Unity."

Source: Unity's official documentation site

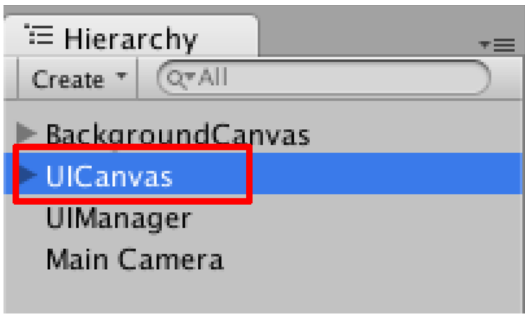


Figure 12 – The user interface canvas.

Inside the ***UICanvas*** object, you have another game object, ***Button***, which is an UI element of the UI's canvas.

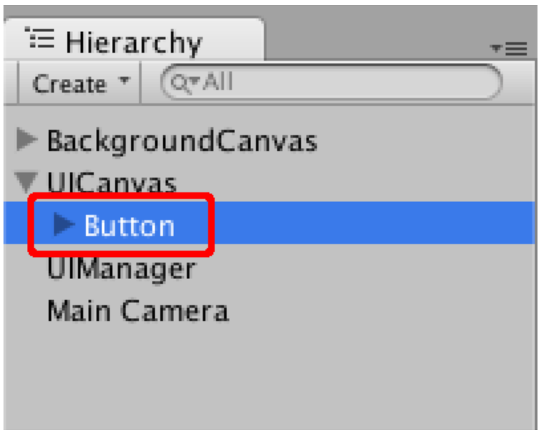
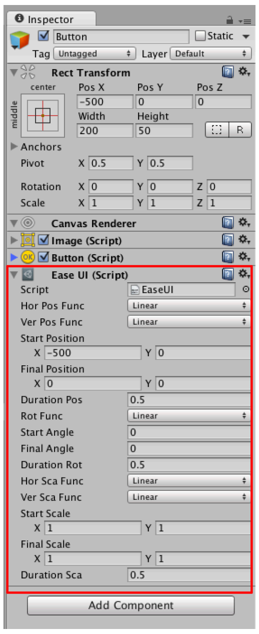


Figure 13 – The UI's elements are placed inside the ***UICanvas*** object.

Although we have just one button inside the UI canvas, you can have several elements in your UI, with each one having an EaseUI component to control them individually.

If you select the ***Button*** object in the scene's hierarchy and take a look in the inspector, you will notice that the EaseUI's component is already attached to the button. So now, it's left to you to change the EaseUI's parameters and see the effects you can get. There are a lot of available easing functions to select, each one with its own singularities. The supported functions by EaseUI are listed in section **8.**



```
Button                         Static
Tag Untagged         Layer Default
Rect Transform
center      Pos X     Pos Y     Pos Z
middle      -500      0         0
            Width     Height
            200       50
Anchors
Pivot       X 0.5     Y 0.5
Rotation    X 0       Y 0       Z 0
Scale       X 1       Y 1       Z 1
Canvas Renderer
Image (Script)
Button (Script)
Ease UI (Script)
Script              EaseUI
Hor Pos Func        Linear
Ver Pos Func        Linear
Start Position
   X -500           Y 0
Final Position
   X 0              Y 0
Duration Pos        0.5
Rot Func            Linear
Start Angle         0
Final Angle         0
Duration Rot        0.5
Hor Sca Func        Linear
Ver Sca Func        Linear
Start Scale
   X 1              Y 1
Final Scale
   X 1              Y 1
Duration Sca        0.5
```

Figure 14 – With the EaseUI component already attached to the UI element, you can change some parameters to obtain several transition effects.

For this particular demonstration scene, you will see that the **StartPosition** variable already has a preset value, **-500** at its horizontal (**X**) component and **0** at the vertical (**Y**). That's because the button will **start outside** the screen. When you call the ***MoveIn( )*** method, the button will start moving into the screen, going from **StartPosition** to **FinalPosition**. To hide the button, just call the ***MoveOut( )*** method, making the transition from **FinalPosition** to **StartPosition** to start. To call these methods, you need to have a script with the EaseUI's component reference of the button. As this is a demonstration, this script is already created for you and its attached to a game object called ***UIManager***. See figure 15.

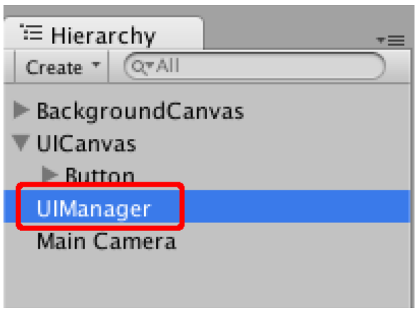


Figure 15 – The ***UIManager*** object contains the **UIManager.cs** script, which is used to control the button.

When you select this object, you are going to see the script attached to it in the inspector and the **EaseUIComponent** variable, which is the reference to the button's EaseUI component and is already set. With this reference, you can call all the EaseUI's methods to control the button.

```
using UnityEngine;
using EaseTools;
using System.Collections;

public class UIManager : MonoBehaviour
{
        // Declaring this variable as public
        // allows the player to see it in
        // the inspector.
        public EaseUI easeUIComponent;

        // Update is called once per frame
        void Update( )
        {
                // Translation
                if (Input.GetKeyDown(KeyCode.Q))
                        easeUIComponent.MoveIn( );
                else if (Input.GetKeyDown(KeyCode.W))
                        easeUIComponent.MoveOut( );

                // Rotation
                if (Input.GetKeyDown(KeyCode.A))
                        easeUIComponent.RotateIn( );
                else if (Input.GetKeyDown(KeyCode.S))
                        easeUIComponent.RotateOut( );

                // Scaling
                if (Input.GetKeyDown(KeyCode.Z))
                        easeUIComponent.ScaleIn( );
                else if (Input.GetKeyDown(KeyCode.X))
                        easeUIComponent.ScaleOut( );
        }
}
```

Code sample 2 – This is how **UIManager.cs** looks like. It won't be much different than this when you write your own code for your own project.

Now, run your game and try the ***Q*** and ***W*** keys of your keyboard. When you press ***Q***, you will call the ***MoveIn( )*** method and is going to see the button coming into the screen. You could use this when the player pauses the game, and then the pause menu would come into the screen in a more exciting way than just appearing/disappearing. So, your button is already in the screen and now you want to hide it again. As the ***W*** key is set to call the ***MoveOut( )*** method, press this key to take the button out of the screen again.

Try changing the EaseUI's component parameters: functions and/or duration, for example. Make some transition effects combinations using the translation, rotation and scaling. See the different effects you can get. **Try it**! :)

7. Quick start

This section is intended to those who already have a good familiarity with Unity3D (intermediary and advanced users) and need only a quick overview of the plugin to start using it right away.

After the plugins (**EaseUI.dll** and **EaseUIEditor.dll**) are installed successfully, you just have to add an EaseUI component to an UI's element (buttons, texts, images, etc.) to control it. You can add EaseUI to a game object by using the editor's top menu or using the **Add Component** button in the inspector.

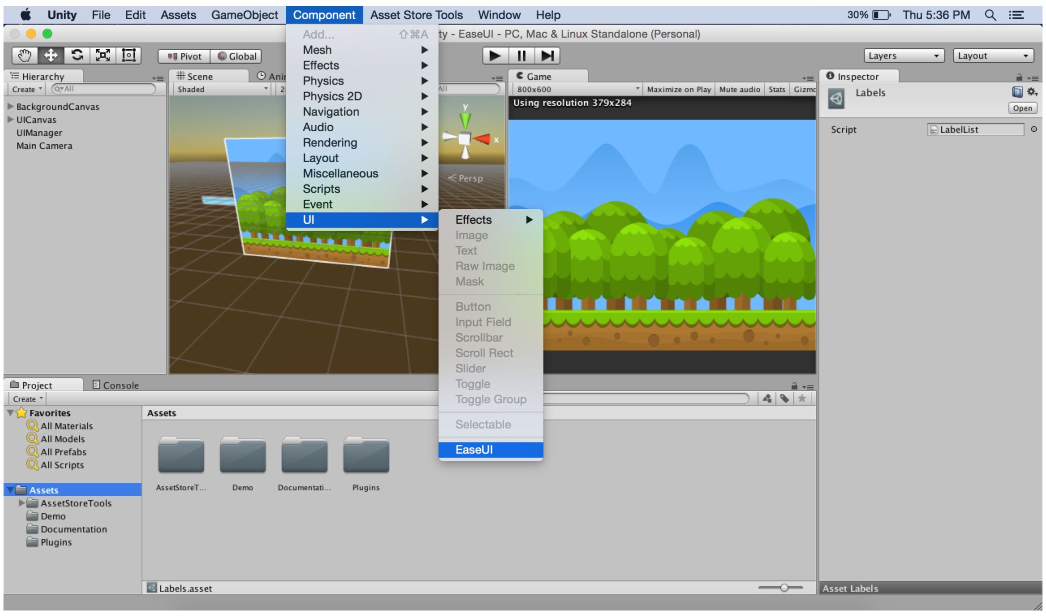


Figure 16 – The EaseUI component is accessible from the editor's top menu.

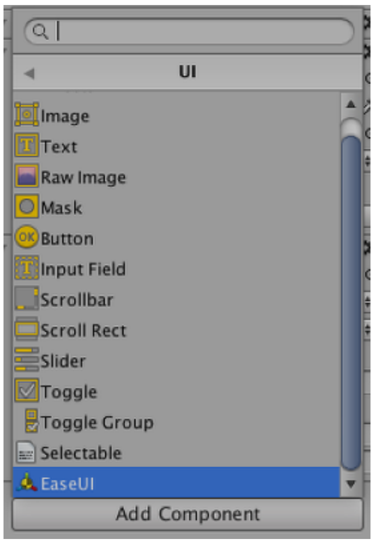
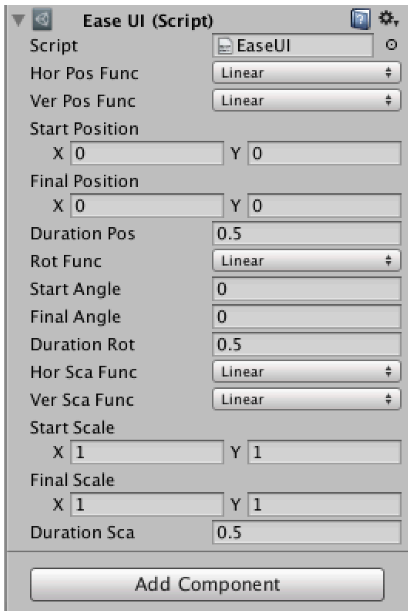


Figure 17 – Inside the **UI** menu, you can add an **EaseUI** component to your game object, pressing the **Add Component** button in the inspector.

With the EaseUI component already attached to the object you need to control, you can now set the transition effects parameters in the inspector.



Ease UI (Script)

Script	EaseUI
Hor Pos Func	Linear
Ver Pos Func	Linear
Start Position	X 0 Y 0
Final Position	X 0 Y 0
Duration Pos	0.5
Rot Func	Linear
Start Angle	0
Final Angle	0
Duration Rot	0.5
Hor Sca Func	Linear
Ver Sca Func	Linear
Start Scale	X 1 Y 1
Final Scale	X 1 Y 1
Duration Sca	0.5

Figure 18 – EaseUI component's parameters are shown in the inspector. Change them to customize your transition effects.

After the EaseUI's parameters in the inspector are set, you just have to call two methods for each type of effect you are going to use - translation, rotation or scaling – in order to control the UI component. **The EaseUI's class is inside the *EaseTools* namespace.** Of course, to call these methods, you need to have a script with references of the EaseUI's components you want to control (a manager script, for example).

Basically, to control the UI's effects, you only have to use the ***In*** and the ***Out*** methods listed below:

- **Translation methods**:
    - ***MoveIn( )*** – **translates** the UI's component **from** **StartPosition** **to** **FinalPosition** within **DurationPos** seconds.
    - ***MoveOut( )*** – **translates** the UI's component **from** **FinalPosition** **to** **StartPosition** within **DurationPos** seconds.

- **Rotation methods**:
    - ***RotateIn( )*** – **rotates** the UI's component **from** **StartAngle** **to** **FinalAngle** within **DurationAng** seconds.
    - ***RotateOut( )*** – **rotates** the UI's component **from** **FinalAngle** **to** **StartAngle** within **DurationAng** seconds.

- **Scaling methods**:
    - ***ScaleIn( )*** – **scales** the UI's component **from** **StartScale** **to** **FinalScale** within **DurationSca** seconds.
    - ***ScaleOut( )*** – **scales** the UI's component **from** **FinalScale** **to** **StartScale** within **DurationSca** seconds.

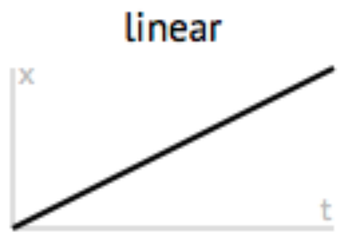
You can also check if the element is moving, rotating or scaling:

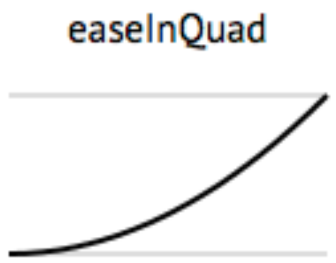
- ***IsMoving( )*** - determines whether the element is moving.
- ***IsRotating( )*** - determines whether the element is rotating.
- ***IsScaling( )*** - determines whether the element is scaling.

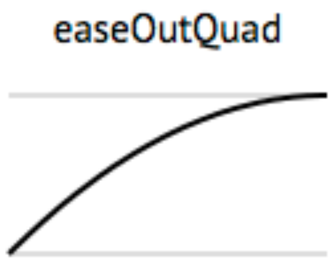
8. Available functions

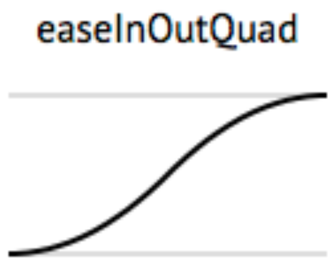
This section lists almost all the available easing functions in EaseUI, missing only the ***OutIn*** functions. Notice that the ***InOut*** functions are the combination of ***In*** and ***Out*** functions. An ***OutIn*** function acts the same way, but the functions are combined at the inverse order.

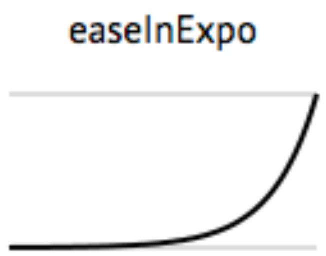
Each function has a particular behavior curve, so when you select different easing functions, you will also get different transition effects. The following curves might help you when you want to achieve a specific transition.

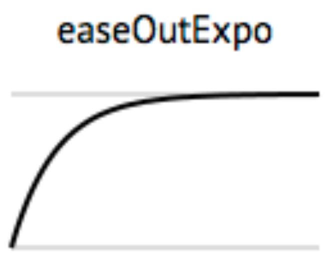


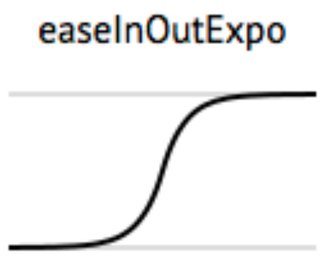


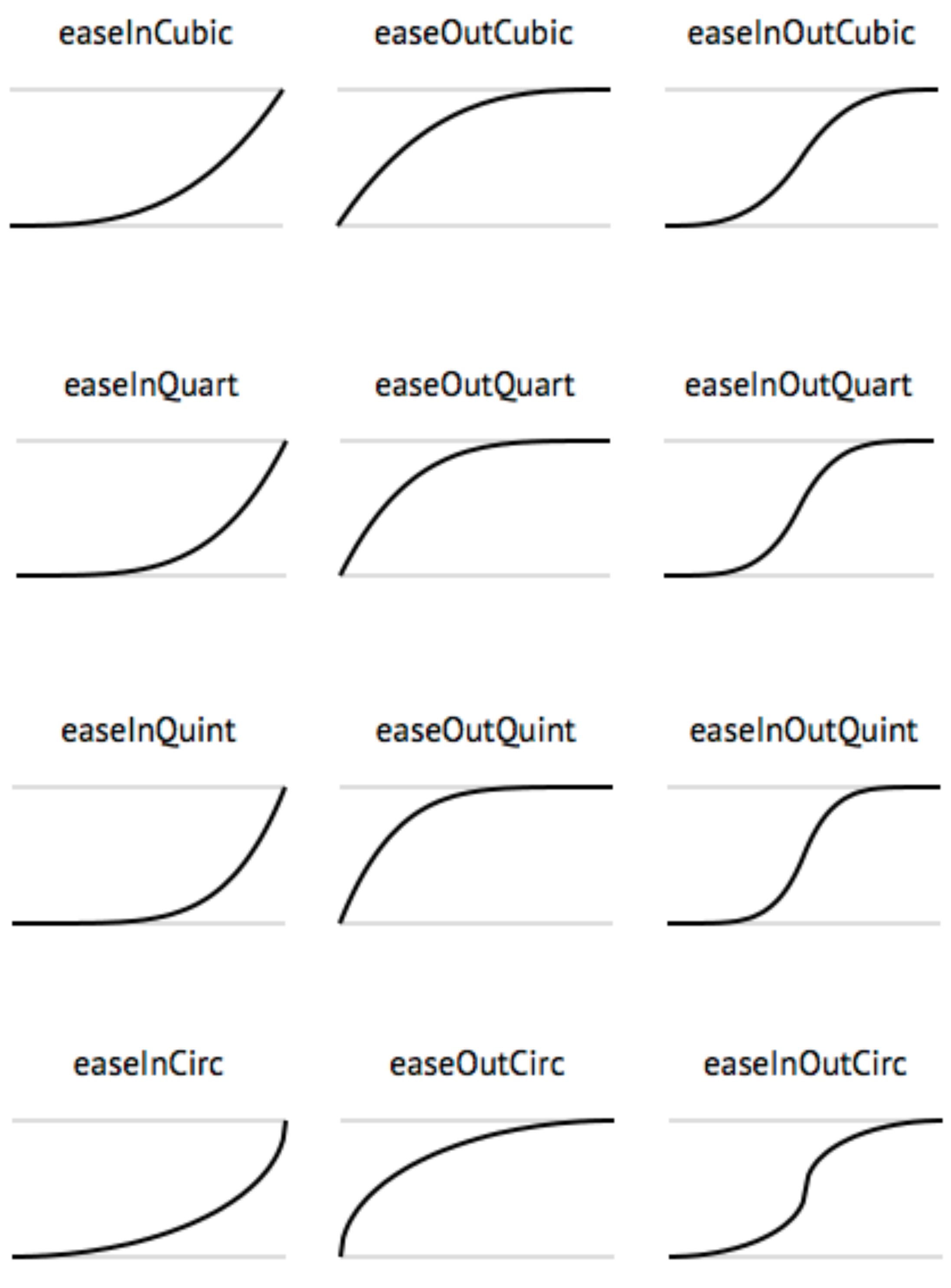


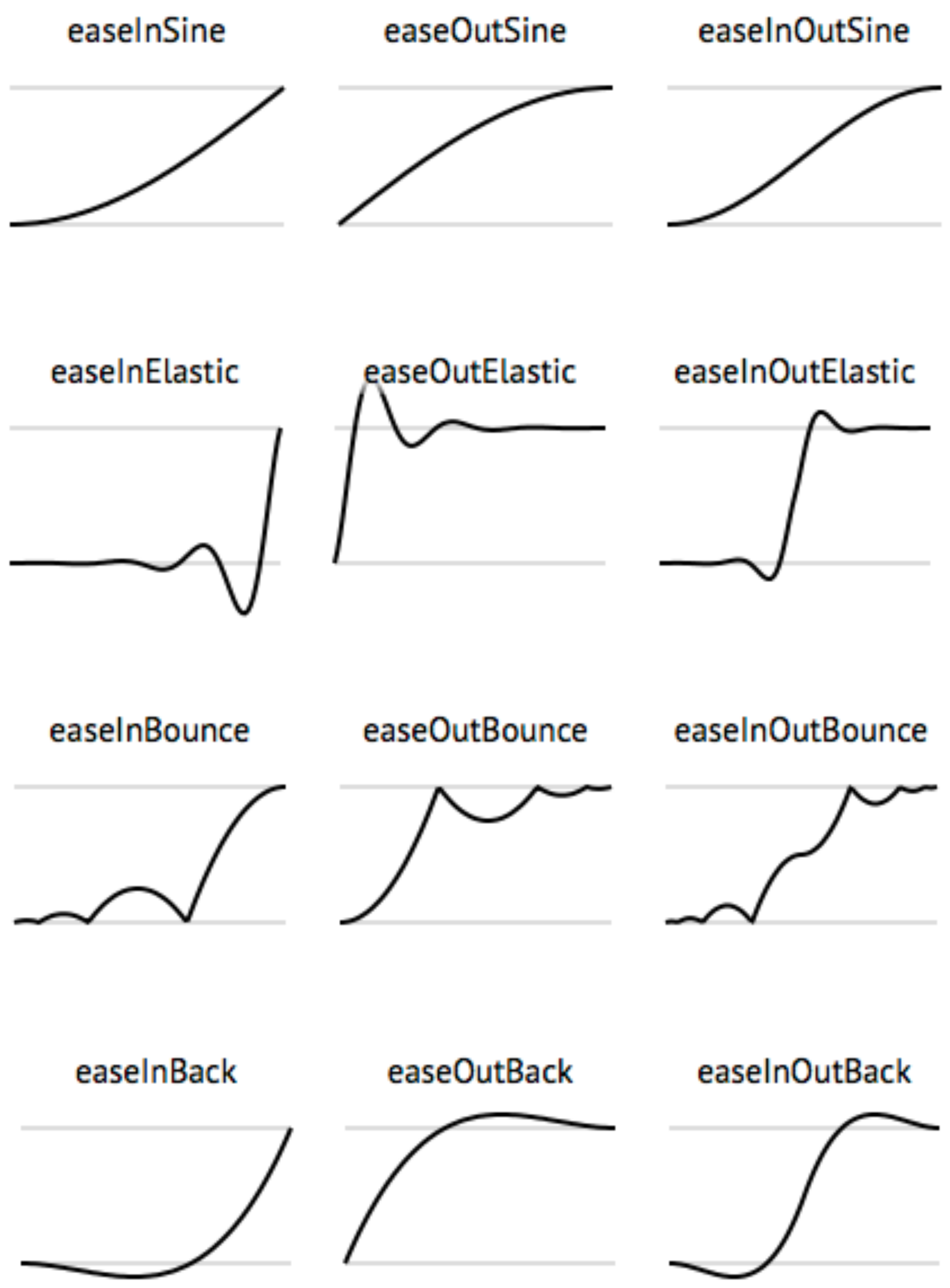









easeInCubic
easeOutCubic
easeInOutCubic
easeInQuart
easeOutQuart
easeInOutQuart
easeInQuint
easeOutQuint
easeInOutQuint
easeInCirc
easeOutCirc
easeInOutCirc



All these images were taken from http://easings.net.