Flash Like Events

Shot Overview

Flash Like Events this is fast and flexible event system for Unity, builded on C# delegates. It’s similar to well known AS3 event system. Just read the method description and you will understand how it work and how it can help you with your game.

Event system documentation

There Is two main classes (EventDispatcher and EventDispatcherBase) with allow you to implement eventing tou your game. They do the same, but EventDispatcher is inherited from MonoBehaviour and EventDispatcherBase is just a System.Object.

if you want your script to be able to dispatch events you should inherit it from EventDispatcher or EventDispatcherBase.

Methods description

Here is methods description of EventDispatcher and EventDispatcherBase classes.

```
// Adds events listener do the dispatcher. You may use two types of functions
(with or without additional data) and two types of event identification
(string of int)
```

```
public void addEventListener(string eventName, EventHandlerFunction handler)
public void addEventListener(int eventID, EventHandlerFunctio handler)
public void addEventListener(string eventName, DataEventHandlerFunction handler)
public void addEventListener(int eventID, DataEventHandlerFunction handler)
```

```
// Removes events listener from the dispatcher.
```

```
public void removeEventListener(string eventName, EventHandlerFunction handler)
public void removeEventListener(int eventID, EventHandlerFunction handler)
```

```
public void removeEventListener(string eventName, DataEventHandlerFunction handler)
public void removeEventListener(int eventID, DataEventHandlerFunction handler)
// Dispatches the event
public void dispatchEvent(string eventName)
public void dispatchEvent(string eventName, object data)
public void dispatchEvent(int eventID)
public void dispatchEvent(int eventID, object data)

public void removeEventListener(string eventName)
public void dispatch(string eventName, object data)
public void dispatch(int eventID)
public void dispatch(int eventID, object data)

// Remove all listners from the dispatcher
public void clearEvents()
```

If you need to send some data via your events. You should use dispatcher function with contains data filed. In order to receive this data, your data function should have CEvent as parameter.

Note: If you dispatch event with additional data, and you have some listeners that do not have CEvent as parameter thay will be called any way.

Note: Because the EventDispatcher is inherited from MonoBehaviour it can detect what gamobject is destroyed, and clean up all listeners on destroy. This feature can save you from dangerous situations when you forgot to remove listener. But I do not recommend to

rely on this feature, it only designed for hedging. Besides EventDispatcherBase not able to do that.

Here is properties and methods description of CEvent class.

```
// Event id, if you are using int event identification. And if you use string event identification it will contain HashCode of string identificator.
public int id {get;}

// Name of the event if you use string event identification. Contains string.Empty if you are using int event identification
public string name {get;}


// Event data, that you have sended via dispatcher function. Null if you use dispatcher function without data parametr.
public object data {get;}

// Event target. Contains link to the dispatcher class of this event
public IDispatcher target {get;}
public IDispatcher dispatcher {get;}
```

```
// Contains current event target

public object curentTarget  {get;}
```

Note: curentTarget contains link to the class with currently receive the event. target contains like to the class who has dispatched the event.

You can read more this post to understand difference between them.

```
// Will stop event propagation, but listener functions of class with has received last event still be able to get event.

public void stopPropagation()

// Stop Event propagation immediately

public void stopImmediatePropagation()
```

Note: You can also read this and this post, to better understand difference between stopPropagation and stopImmediatePropagation.

Use example

Here is small example of button that dispatchers click with click time as data.

To create our button we should create simple mono behavior script. But instead inheriting our button class from MonoBehaviour we will inherit it from EventDispatcher to give our button ability dispatch the events. Here is full code of out Button.cs script

```
using UnityEngine;
using System.Collections;

public class Button : EventDispatcher {

	private float w = 150;
	private float h = 50;

	private Rect buttonRect;

	void Awake() {
	    buttonRect = new Rect((Screen.width - w) / 2, (Screen.height - h) / 2, w, h);
	}

	void OnGUI() {
		if(GUI.Button(buttonRect, "click me")) {
			dispatch(BaseEvent.CLICK,  Time.time);
		}
	}

}
```

When we click GUI button it will dispatch click event as we see if use dispatch function with data and sending Time.time variable as event data.

Next we create Class with will create this button and will listen for it event. Lets create our

example listener class and attach it to any GameObject on scene. Here is full code of ExampleListner script.

```
using UnityEngine;
using System.Collections;

public class ExampleListner : MonoBehaviour {
        void Start () {
                //creating our button
                Button btn = gameObject.AddComponent<Button>();
                //listening for the event
                btn.addEventListener(BaseEvent.CLICK, onButtonClick);
        }

        private void onButtonClick(CEvent e) {
                //getting click time from event
                float clickTimde = (float) e.data;

                //printing result
                Debug.Log("Button click time is: " + clickTimde.ToString());
        }
}
```

This is basic example of building event based solution. You will find a lot more under Plugin Example folder.

Support

I hope this extension will help you to build and analyze your game architecture.

If you have any questions, problems or suggestions, please contact me anytime via E-mail: lacost.st@gmail.com.