And

Online technical documentation ▪ Discord

Getting Started Guide

Sam Swicegood, Lead Developer
Updated 3/14/2024

Introduction to EventsPro

EventsPro is a versatile and powerful extension for Unity, designed to enhance both UnityEvents and the Unity User Interface (UI). It offers a comprehensive suite of tools and features that streamline and augment the event handling and UI capabilities within the Unity engine. Key features of EventsPro include:

1. **Extended UnityEvents**: EventsPro expands on the standard UnityEvents, providing more flexibility and functionality. This makes it easier for developers to manage complex event-driven systems in their games or applications.
2. **UI Enhancements**: With its focus on the Unity UI, EventsPro offers additional components and attributes that enrich the UI experience. This includes advanced UI elements and controls that can be customized and integrated seamlessly into your projects.
3. **Custom Attributes and Utilities**: EventsPro includes a range of custom attributes and utility classes, designed to simplify common development tasks and add new capabilities to the Unity editor.
4. **Event Triggers and Listeners**: It provides sophisticated event triggers and listeners, allowing for intricate event management and response mechanisms.
5. **Integration with Unity Systems**: Seamlessly integrating with Unity’s existing systems, EventsPro enhances the overall workflow and allows developers to leverage Unity's full potential more effectively.

EventsPro is ideal for both beginners and intermediate Unity users, offering an intuitive interface and easy-to-understand documentation. It's particularly useful for those looking to create more interactive and dynamic UI elements or to implement complex event-driven behaviors in their Unity projects.

Setting up EventsPro

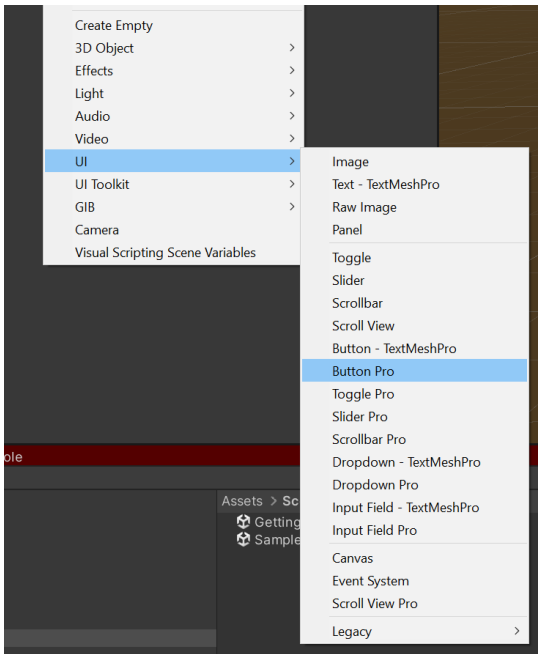
Setting up EventsPro in your Unity project is a straightforward process, as the extension is designed to work "out-of-the-box" without requiring extensive configuration. Here are the simple steps to get started:

1. **Importing EventsPro**: First, you need to import the EventsPro package into your Unity project. This is usually done through the Unity Asset Store or by importing a package file if you have it.
2. **Accessing EventsPro Features**: Once imported, EventsPro integrates with Unity's existing systems. You can access its features through the Unity Editor, where it enhances UnityEvents and UI components with its extended functionalities.
3. **No Additional Setup Required**: EventsPro is designed to be ready to use immediately after import. There are no additional steps or complex configurations needed to start using its features.
4. **Exploring EventsPro Components**: You can explore various EventsPro components and attributes in the Unity Editor. These components will be available in the component menu and can be added to your GameObjects just like any standard Unity component.
5. **Utilizing Enhanced UnityEvents and UI**: With EventsPro, you can now leverage enhanced event handling and UI capabilities in your projects. This includes using advanced event triggers, listeners, and custom UI elements that EventsPro provides.

Remember, if you're new to Unity or have specific requirements, you may want to familiarize yourself with the Unity Editor and basic Unity concepts to make the most out of EventsPro. The extension is designed to be user-friendly and enhance your development experience with Unity.

The EventsPro Component

An EventsPro component works by enabling event handling in Unity in an enhanced and flexible way. Users have the option to use the included LiteTrigger or TriggerAgent components or integrate EventPro into their own scripts. The EventTriggerPro component, specifically, is designed to receive events from Unity's EventSystem and call registered functions for each event.



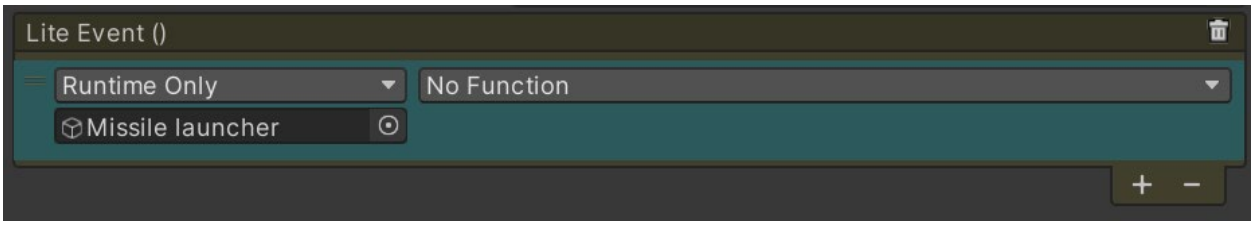
To utilize EventPro in a custom script, users need to include the namespace using `UnityEngine.Events` in their script, just as they would when using Unity's standard UnityEvent. This makes the advanced features of EventsPro accessible within the script.

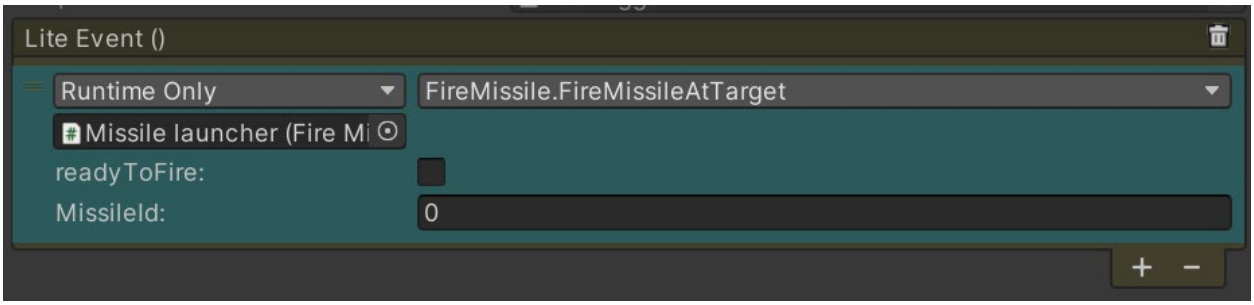
EventsPro UI components

Users can add EventsPro UI components from the UI objects menu as well.

Adding an Event

The setup process for using EventsPro is quite similar to that of a standard UnityEvent. Users can drag an object – which could be a static script, a scene object, or a ScriptableObject – into the object field of the event. They then select the relevant method they wish to call from the dropdown menu. This interface design makes EventsPro intuitive for those already familiar with UnityEvents, while offering the extended capabilities of EventsPro.





Calling Events

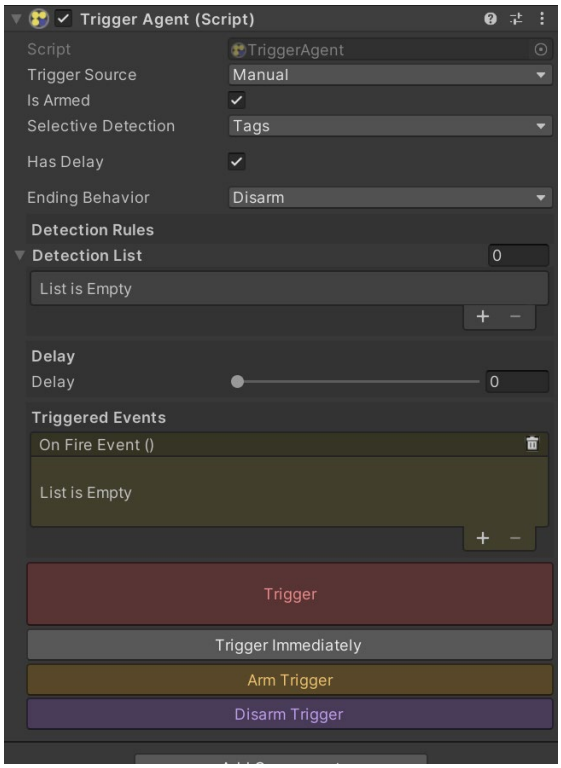
By default, any EventPro can be called in the exact same way as a UnityEvent: using the **`Invoke()`** method.

Any script deriving from EventTrigger (including the Trigger Agent) can be triggered using the method **`Trigger()`**.

Trigger Agent, specifically, can also be called by clicking the relevant button in Inspector.

Trigger Agent

Trigger Agent is a component made to handle triggers, build behaviors, and invoke events. You can use Trigger Agent to set the exact circumstances required for an event to trigger.



The Trigger Agent contains a EventPro called OnFireEvent, which is the container for the component's logic. The rest of the component is managed through several options.

Properties

Trigger Source

This is what determines the trigger to fire.

- **Manual** (default): The trigger will only be fired when Trigger() is called on the component.
- **Trigger Enter:** The trigger will attempt to fire when an object enters its trigger (using **`OnTriggerEnter()`**)
- **Trigger Exit:** The trigger will attempt to fire when an object exits its trigger (using **`OnTriggerExit()`**)
- **Start:** The trigger will attempt to fire on the first frame it is active (using **`Start()`**)
- **Collision:** The trigger will attempt to fire when an object exits its trigger (using **`OnCollisionEnter()`**)
- **Event:** The trigger will attempt to fire when a specific event is called from the EventController.

Is Armed

Trigger Agent can be armed or disarmed. If the trigger is armed, it may be able to fire when **`Trigger()`** is called. A Trigger Agent that is disarmed will only fire if **`TriggerImmediately()`** is called.

Selective Detection

This only applies to Trigger Agents set to detect collision or trigger event/exit.

Determines if a trigger is seeking a particular target. For example, a Trigger Agent triggered by *Trigger Enter* will attempt to fire any time any time **`OnTriggerEnter`**() is called **.** Options are:

- **None** (default)**:** This Trigger Agent does not use selective detection.
- **Player:** The target has the *Player* tag.
- **Tags:** The target has a tag that matches a tag in the list.
- **Name:** The target object has a name that is in the list.
- **Collider:** The target object has a collider that is in the list.
- **Trigger Target:** The target object has the TriggerTarget component.
- **Target ID:** The target object has the TriggerTarget component, and that target ID is in the list.

Has Delay

When Trigger is called, this Agent will wait a specified number of seconds (up to 60) before firing.

Ending Behavior

After the trigger has fired, the Agent will do one of the following things.

- **Disarm** (default): The trigger will disarm and not activate until it is re-armed.
- **Arm:** The trigger will remain armed and be able to be triggered again.
- **Destroy:** The TriggerAgent component will be destroyed.
- **Destroy Object:** The entire object this Trigger Agent is attached to will be destroyed.

Using the Agent

Arming and Disarming the Trigger Agent

The Trigger Agent can be armed and disarmed to prevent it from being triggered. To arm the Agent, use :

- **`SetTriggerState(bool state)`** – true is armed, false is disarmed.
- **`ArmTrigger()`** – arms the trigger.
- **`DisarmTrigger()`** – disarms the trigger.

Triggering the Agent

Trigger Agents may be triggered in the following ways:

- Trigger/Collider interactions that meet the conditions of the Trigger Source and Delective Detection (see below)

- Calling the **`Trigger()`** method (if the Agent is armed, see below).
- Calling **`TriggerImmediately()`** (this will bypass conditions and arm state).
- Calling **`OnFireEvent.Invoke()`** will trigger the EventPro as a UnityEvent.

Other Useful Components

Lite Trigger

An EventPro on a single Monobehavior. No mess, no fuss.

Event Trigger

Like a Lite Trigger, but used for collecting UI events.

Trigger Target

A target script used for selective detection for Trigger Agents.

Event Announcer

Announces a target string to the Event Controller. You can also have it call a custom string.

Event Controller

Receives and sends events to the scene and can be called with the static *EventController*.

Trigger Listener

Invokes an EventPro when a specific event is called from the Event Controller.

GIB Toolkit

The GIB Toolkit can be opened by going to Tools > GIB Toolkit.

Menu Items

Tools

Physics Simulation

Simulate physics in-editor. Note: make sure to turn off simulation when done!

Tickrate - The rate at which to calculate a physics step. **Default is 33.**

Speed - The rate at which to run the simulation. **Default is 1.**

Gravity - The direction of gravity to run the simulation. **Default is (0, -9.8, 0).**

Simulate/Stop Simulate - Start/Stop the simulation of selected objects.

Find Missing References

Allows you to find missing references. You can search is the current scene, all scenes, or all assets.

Find Materials By Shader

Searches through materials and returns materials that use the given shader.

Remove Missing Scripts

Finds and removes missing scripts in the selected GameObjects.

Replace Missing Materials

Finds missing materials and replaces them with the target material.

Replace Objects with Prefab

Replace all of the selected objects with the target prefab. The object will retain its transform position, rotation and scale.

Destroy EditorOnly on Play

If checked, then when you enter Play Mode in Unity, objects with the EditorOnly tag will be destroyed on Awake().

IMGUI Debugger

Opens the Unity Immediate Mode GUI (IMGUI) debugger window by calling **`EditorWindow.GetWindow(type).Show()`**.

Request Script Compilation

Forces Unity editor to call a script compilation by calling **`CompilationPipeline.RequestScriptCompilation()`**.

Toggle Inspector Lock

Toggles the Inspector Lock, and also provides the shortcut key *Alt+Shift+Q to do so.*

Components

Some useful components!

Debug Find Collider

Each time a collision occurs on the GameObject, outputs data to the console.

Ignore Colliders

Set target colliders to be ignored by this GameObject.

Lock Local Position

Locks an object in local space. Doesn't require a rigidbody.

LookAtPlayer

Looks at the first object found tagged "Player".

LookAtCamera

Looks at the Main Camera.

DeveloperComments

Just a big TextArea for you to write comments visible in Inspector.

DisableOnAwake

Disables an object on Awake().

DontDestroyOnLoad

Adds the object to Unity's DontDestroyOnLoad metalist.