Impossible Odds - Mouse Events

In some types of games, the mouse cursor plays a vital part in controlling the game's mechanics. The Unity game engine provides a simple but bare interface of working with certain mouse events. The Impossible Odds Mouse Events tool provides a robust way in dealing with clicking and dragging events in your game.

This tool offers you the following features:

- Single and double clicking,
- Dragging and drag completion,
- Events and callbacks for these operations, and
- Tracking over several mouse buttons at once.

Installation

You can add this tool to your project using Unity's package manager. Add the following git-tracked URL to your project:

> https://github.com/juniordiscart/ImpossibleOdds-MouseEvents.git?path=/Assets/Impossible%20Odds/MouseEvents

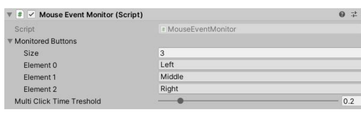
Quick Setup

Attach the `MouseEventMonitor` script to a game object in your scene, and set which mouse buttons it should monitor. Additionally, adjust the time it takes to distinguish between single clicks and multi-clicks of a mouse button.

When your game is running, you can listen for each of the following events:

- `onSingleClick` : when a single click is registered.
- `onDoubleClick` : when a double click is registered.
- `OnDragStart` : when a drag operation is started.
- `onDragOngoing` : when the mouse pointer is being dragged while one of the mouse buttons is being held down.
- `onDragCompleted` : when the drag operation has ended.
- `onEvent` : called right after when any of the above events are called.

Each of these events contain a `MouseButtonEvent` parameter, which contains all information about the event such as which mouse button it pertains to, whether it's a click or drag event, the click count, the relevant mouse position(s) and any additional key modifiers that were active.



Mouse Event Monitor Inspector

Additionally, you can also query the state of a specific mouse button on the mouse event monitor by calling the `CurrentEvent` method. It takes a single mouse button as parameter, and will return you the current state of that button.

Advanced

The main point of entry is the `MouseEventMonitor` script. It requires to be placed on a game object in your scene and will monitor the mouse inputs you set it up to be: left, right and/or middle mouse buttons. Apart from which mouse buttons it should monitor, you can also adjust the time threshold for registering multi-clicks (double click). Unity does not allow to transparently distinguish between single click and double click without always invoking the single click event. This multi-click time threshold is the time limit it will delay a single click event while listening in for a secondary click or other event. Additionally, you can also set a drag distance threshold for when you want to let small or accidental drag operations to still count as a click event instead.

You can listen for events of the registered mouse buttons as well as querying the current state of a particular button using the `CurrentEvent` method. When a new mouse button requires monitoring, you can add it using the `StartMonitoring` method. Conversely, you can also stop monitoring events for a specific mouse button by calling the `StopMonitoring` method.

This script primarily works using Unity's `Update` and `FixedUpdate` methods (whichever runs first that frame). The `MouseEventMonitor` script is also placed at the lowest *script execution order* value automatically, so that it always runs first. This guarantees that the input states are updated and available for all other scripts that may potentially need them.

Internally, the event monitor employs a `MouseButtonStateTracker` object per mouse button that's registered for monitoring. It's basically a small state machine which keeps track of what's happening with a particular mouse button. When it changes state, it will let interested parties know, e.g. the mouse event monitor.

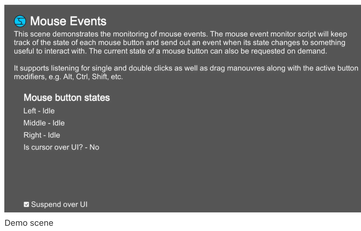
Code Example

The following code example is a showcase of a naive target and selection system using the `MouseEventMonitor`. In its `Update` it will check for click events to perform a single target selection or dispatch a move command. It's also subscribed to events related to dragging the mouse for showing a selection box.

```
public class MouseEventsDemo : MonoBehaviour
{
    [SerializeField]
    private MouseEventMonitor monitor = null;

    private void Start()
    {
        monitor.onDragOngoing += OnDragging;
        monitor.onDragCompleted += OnDragComplete;
    }

    private void Update()
    {
        MouseButtonEvent mouseEvent = monitor.CurrentEvent(MouseButton.Left);
        if (mouseEvent.IsSingleClick)
        {
            if (Physics.Raycast(Camera.main.ScreenPointToRay(mouseEvent.MousePosition), out
            {
                // Select single target.
            }
        }
        else if (mouseEvent.IsDoubleClick)
        {
            // Move selected objects to target position.
        }
    }

    private void OnDragging(MouseButtonEvent mouseEvent)
    {
        // Show box selection on screen.
        Rect screenRect = new Rect(mouseEvent.DragStartPosition, mouseEvent.DragDelta);
    }

    private void OnDragComplete(MouseButtonEvent mouseEvent)
    {
        Rect screenRect = new Rect(mouseEvent.DragStartPosition, mouseEvent.DragDelta);
        if (mouseEvent.Modifiers == EventModifiers.Shift)
        {
            // Expand the current selection with the targets in the selection box.
        }
        else
        {
            // Set the current selection to the target objects in the selection box.
        }
    }
}
```

Demo

The package comes with a demo scene that shows the real-time state of the left, right and middle mouse buttons. This allows you to test the behaviour and events of this package.



Demo scene

Unity Version

Developed and tested on Unity 2019.4 LTS.

License

This package is provided under the MIT license.

Changelog

v1.1.0

- Moved away from working in the `OnGUI` loop because it seems to there's a bug in its cycle, which skips a frame and can't reliably poll for inputs. The `MouseEventMonitor` will now work very early in the regular `Update` cycle.
- Added the `onDragStart` event which is called whenever the mouse events monitor detects that the user starts dragging the mouse while holding one of the tracked mouse buttons.
- Added the `onNewFrame` event which can be useful to clear any mouse event caches.
- Added the `IsTerminalEvent` property to the `MouseButtonEvent` struct to detect whether the mouse event can evolve further into different events or not.
- Updated the `MouseEventMonitor` to allow suspension of operations when it detects the cursor is over UI elements.
- Removed the TextMeshPro dependency.

v1.2.0

- Fixed a runtime issue where clicks would be invalidated due to using Unity's built-in `Event` struct. This structure would put the event type to `Ignore` while it was actually a mouse button up/down event.

v1.3.0

- Added a `DragDistanceThreshold` to the the `MouseEventMonitor` to allow small/accidental drag operations to still register as a click.
- Updated the sample scene to also log the performed actions to the console.