Technical Notes

Some highly specific technical details, problems and solutions

- Detecting a User About to Type on InputField
- Rebuilding UI Layout
- Reading TMP_Dropdown Value in Udon

Detecting a User About to Type on InputField

Due to VRChat not being the most consistent with sent UI events, especially between desktop and VR, it can be difficult to reliably detect when a user is about to start typing on a UI InputField.

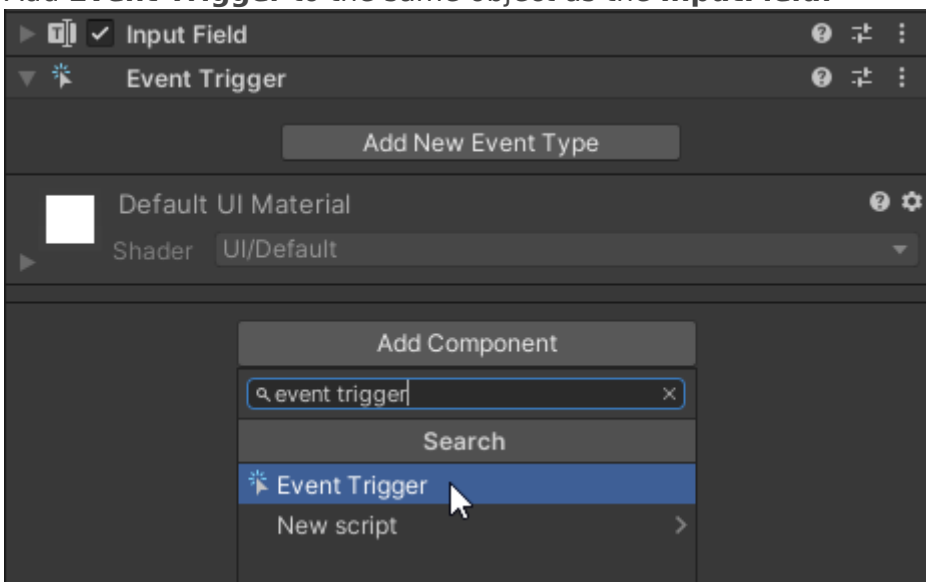
During the development of Udonity I faced this specific issue with the fields on the inspectors and editor windows where values on the input fields are being updated per frame to display accurate information about the inspected objects.

In this scenario detecting the intentions of a user about to start typing on a field is necessary in order to pause the values being brute forced in while typing, which would result in user's input being rejected.

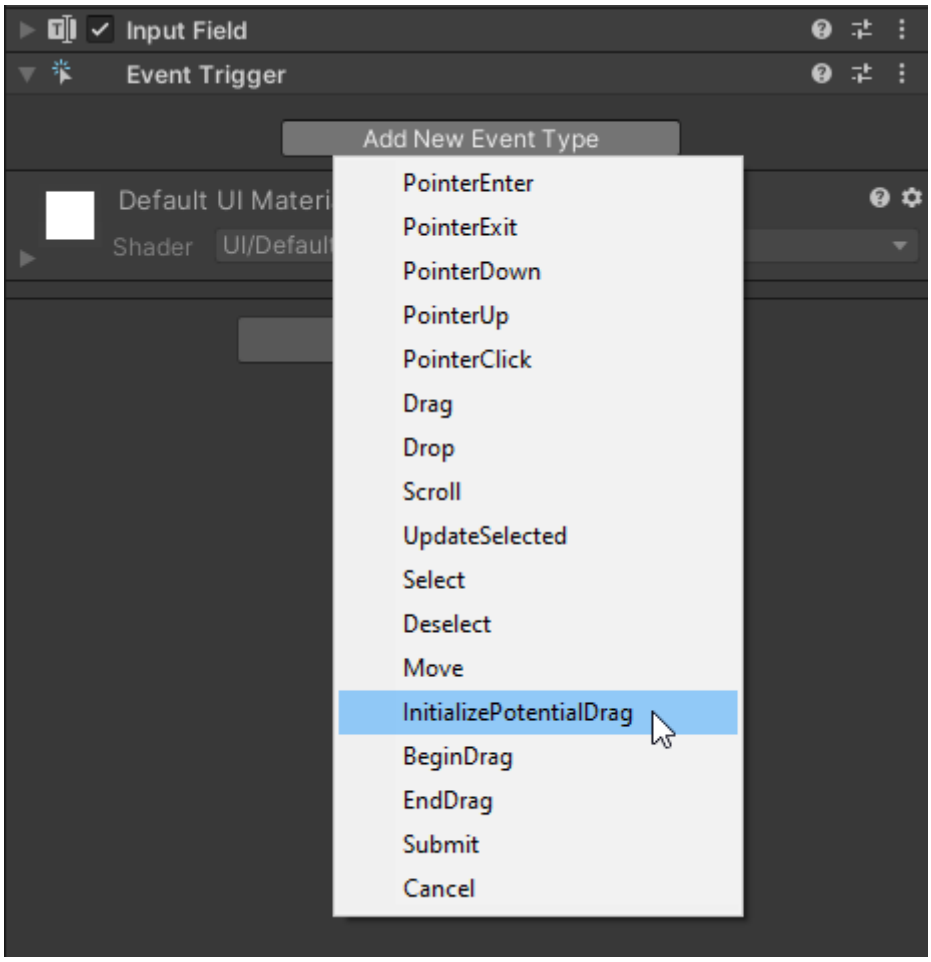
InputFields with no VRCInputFieldKeyboardOverride

Since VRChat 2022.2.2p3, clicking any field will display VRChat's own in-game keyboard for typing, and *so far* I've found one reliable solution for detecting when a user is about to start typing on a field:

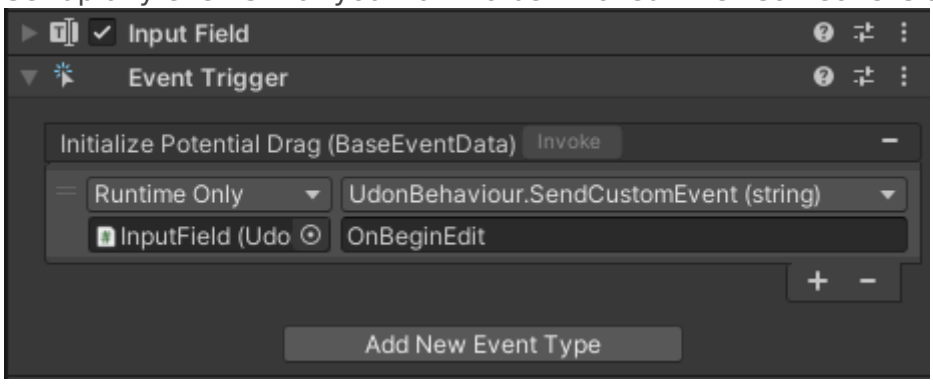
1. Add **Event Trigger** to the same object as the **InputField:**



2. Click **Add New Event Type** > **InitializePotentialDrag**:



3. Set up any events that you want to be invoked when someone is about to start typing:



Rebuilding UI Layout

If you are using Unity UI Auto Layout components on your UI, you may face problematic situations where the elements are getting stuck in weird positions due to layout not being rebuilt when necessary.

In these cases you can simply call **LayoutRebuilder.ForceRebuildLayoutImmediate**, which should be able to fix the problem in many scenarios:

```
// Call this static method with the root of the layout as a parameter. E.g:
UnityEngine.UI.LayoutRebuilder.ForceRebuildLayoutImmediate(GetComponent<RectTransform>());
```

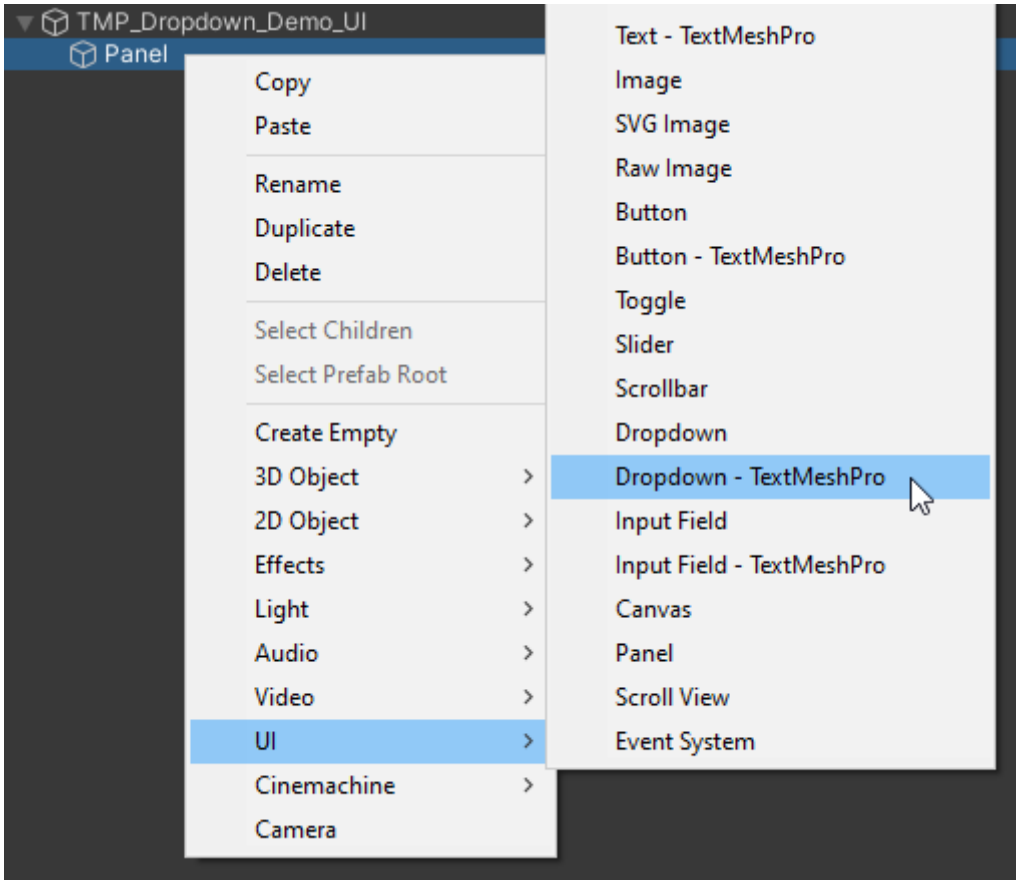
If your UI consists of complex nested layout groups, sometimes calling the method above more than once have been able to solve the problem.

Reading TMP_Dropdown Value in Udon

TMP_Dropdown isn't exposed in Udon, so we're unable to read its value by default. Here is a workaround to receive an event in Udon when a player selects an item on the dropdown, after which we can read the value from a proxy.

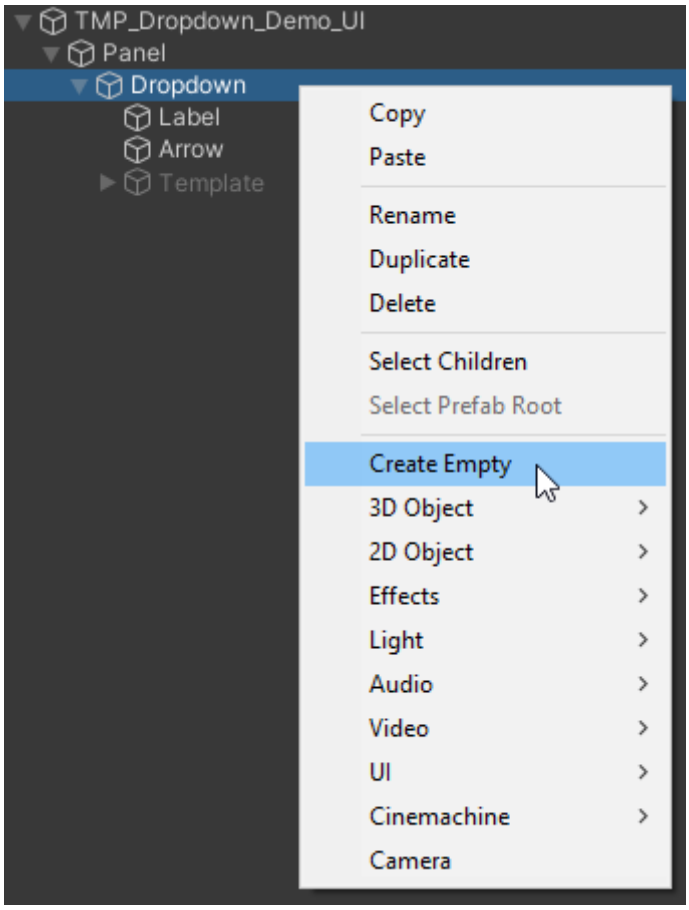
1) Create new TextMeshPro Dropdown

Right click on the element you wish to add the dropdown to, and navigate to `UI` > `Dropdown - TextMeshPro`

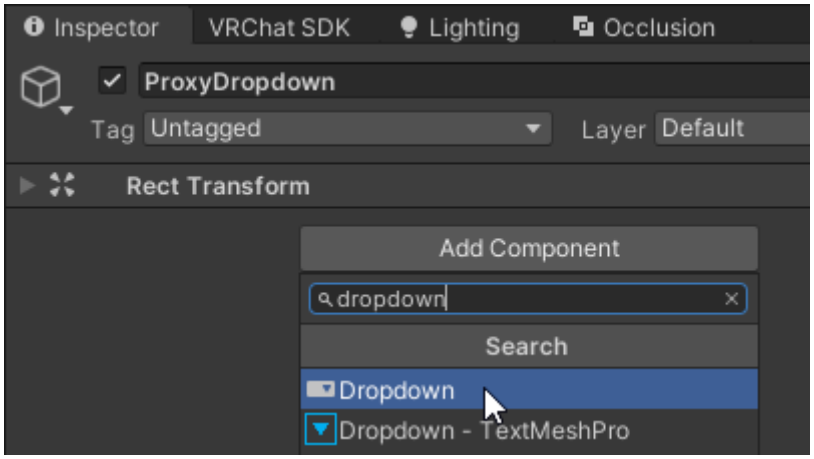


2) Create proxy dropdown

Right click on the TPM dropdown object and add new empty GameObject

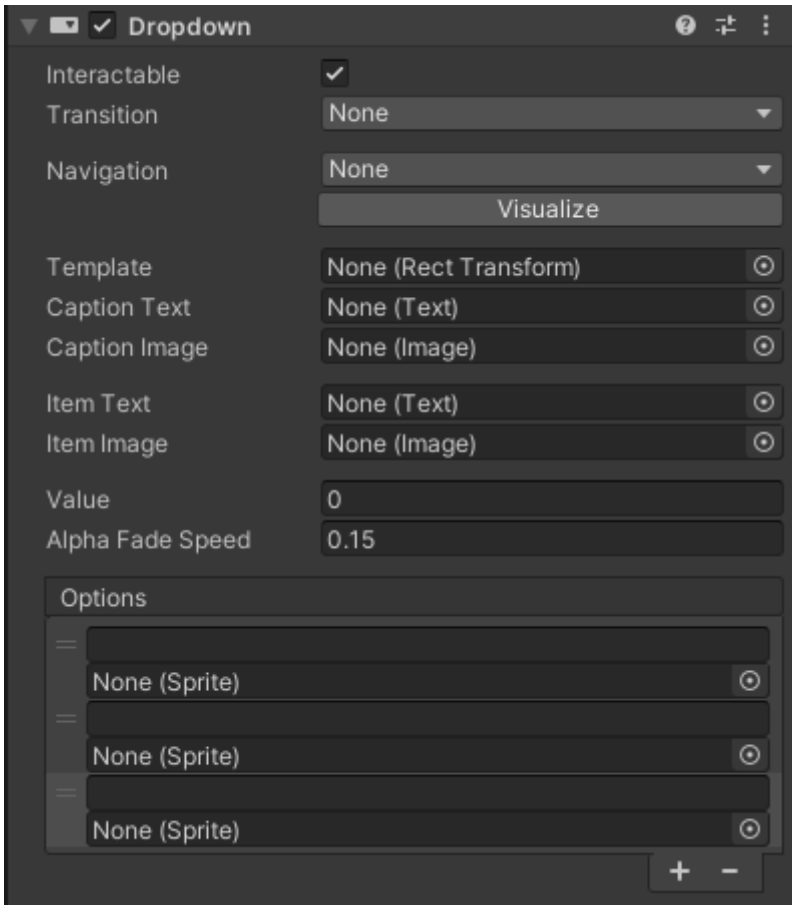


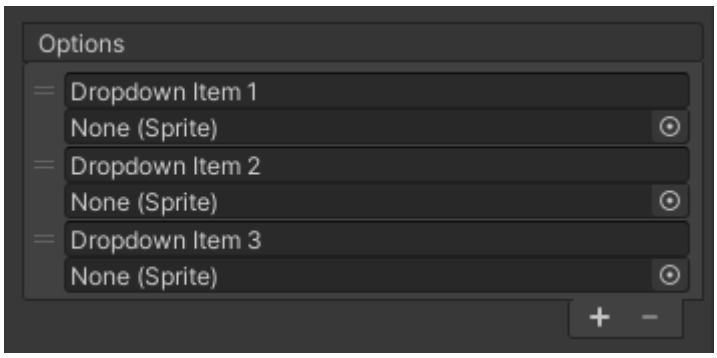
Add Dropdown component on the created proxy GameObject



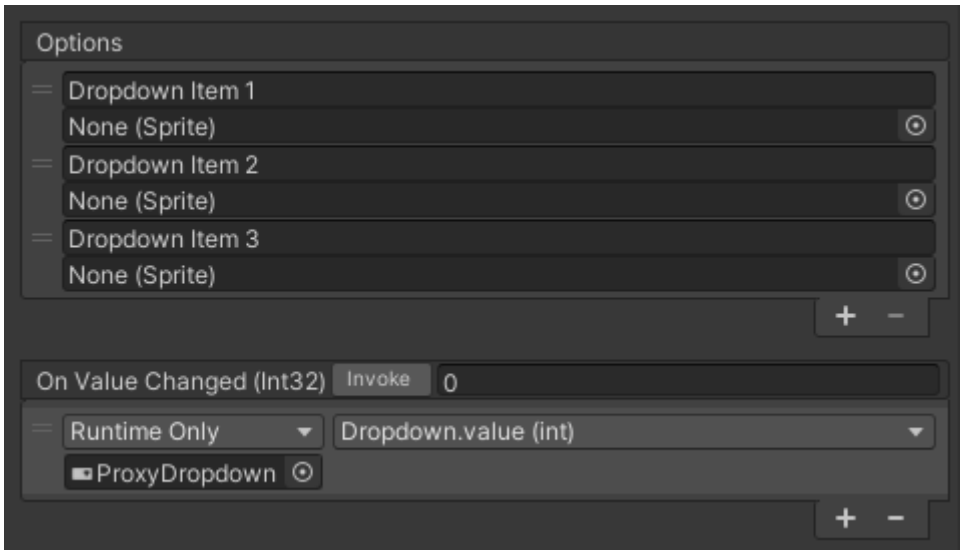
Set up the proxy dropdown

- Transition: `None`
- Navigation: `None`
- Make sure proxy dropdown has the same number of options as the TMP dropdown





3) Add OnValueChanged event on TMP dropdown to set the value of the proxy dropdown



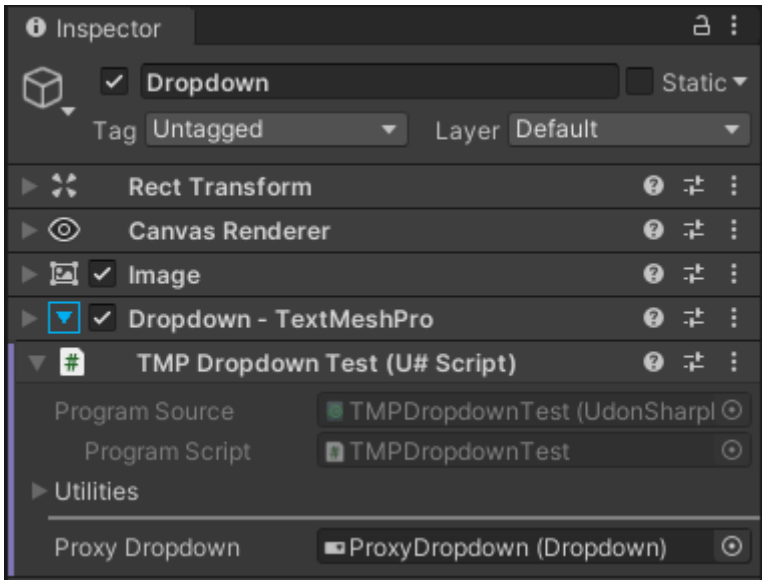
4) Create a dropdown handler UdonBehaviour

```
using UdonSharp;
using UnityEngine;
using UnityEngine.UI;

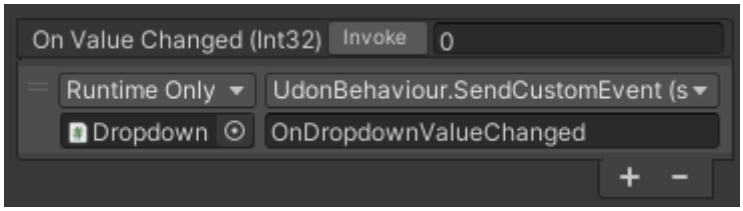
namespace Varneon.Examples
{
    [UdonBehaviourSyncMode(BehaviourSyncMode.None)]
    public class TMPDropdownTest : UdonSharpBehaviour
    {
        [SerializeField]
        private Dropdown proxyDropdown;

        public void OnDropdownValueChanged()
        {
            // value is the index of the item selected in the dropdown
            int value = proxyDropdown.value;
        }
    }
}
```

Add the script above to the TPM dropdown object and assign the proxy dropdown to the field



Add OnValueChanged event on the proxy dropdown to invoke the public method on the UdonBehaviour



Done! - You should now be able to execute any code you want based on the value from the proxy dropdown