Input Icons for TextMeshPro

Requirements (from package manager. Window → Package Manager):

- **TextMeshPro 2.1.6 or higher**
- **Input System 1.2.0 or higher** (**1.3.0**, **1.4.4** or higher recommended)

Input Icons allows easy display of input bindings in TextMeshPro texts using the style tag **(<style=NameOfActionMap/NameOfAction>**). It updates all TMP texts dynamically if the input device changes and also updates the displayed sprites whenever input bindings change.

The package includes sprites for keyboard/mouse and XBox, PlayStation, and Nintendo controllers. If using a different controller, the XBox sprites will be displayed by default (based on Steam statistics showing most PC players use XBox controllers).

For splitscreen games using gamepads we can set a generic "Overwrite Gamepad IconSet" in the Input Icon Set Configurator to always display a specific icon set for gamepads.

Video guide: https://youtu.be/pbm0UvPGCag

Overview

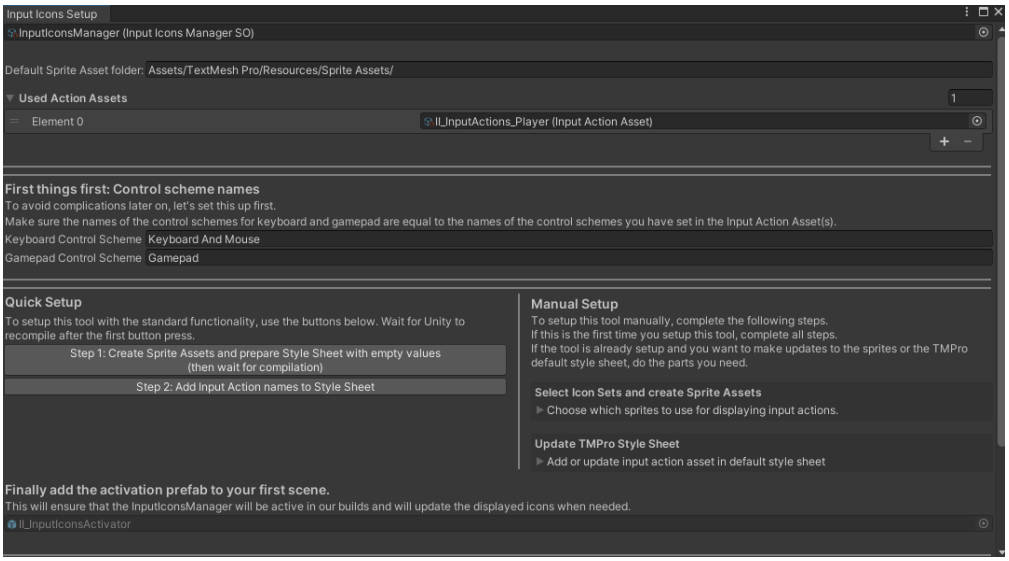
Updating From An Earlier Version

Updating to a new version may cause missing input icons in scenes**. Do the Quick Setup in "Tools" → "Input Icons Setup"** to fix. Check the "Default Sprite Assets folder" path, "Used Action Assets" list, and "Control Scheme Names" in the setup window or in the Input Icons Manager, as they may have reset.

Getting Started

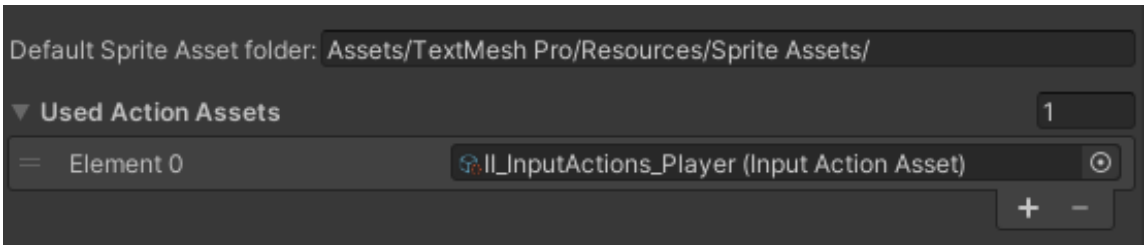
To start, you can use the included sprites or substitute them with your own. This package includes sprites for keyboard/mouse and controllers (PlayStation, Nintendo Switch, and Xbox) and has them pre-configured. The Scriptable Objects storing sprite data can be found in the "Assets/InputIcons" folder.

From the toolbar, open “**Tools/Input Icons Setup”**.



To begin, we need an Input Action Asset with our input actions. If not already done, import the Input System package from the Package Manager (Window > Package Manager) - it requires **Input System 1.2.0** or higher for saving/loading input binding changes. Create a new Input Action Asset or copy the one provided in the package and customize as needed. Add the Input Action Asset to the "Used Input Action Assets" list.

If you moved the TextMesh Pro folder, update the corresponding field in the setup window to ensure assets are created in the correct folder.

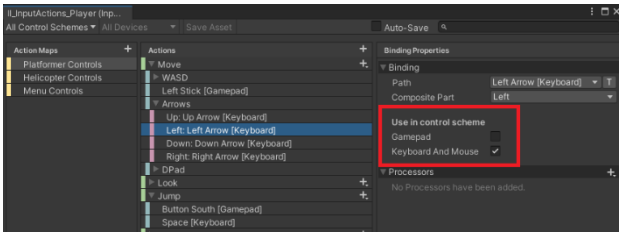


Setting Up The Input Action Asset

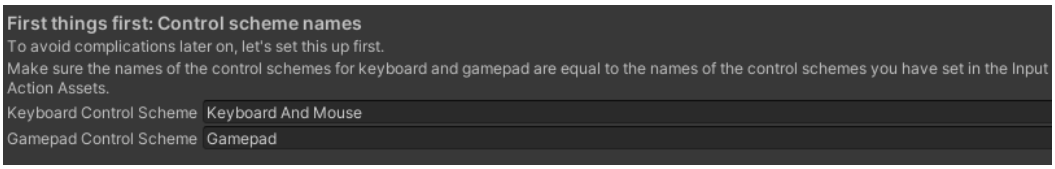
Our Input Action Assets need control schemes for keyboard and gamepads. We have to add devices to the control schemes so the Input Icons Manager can decide which icons to display whenever we switch to gamepad or keyboard.

Verifying Control Scheme Names

Let's be sure that the Input Action Asset has **control schemes** set up **for keyboard and gamepad** control. Or only one of them, depending which device(s) we want to support.

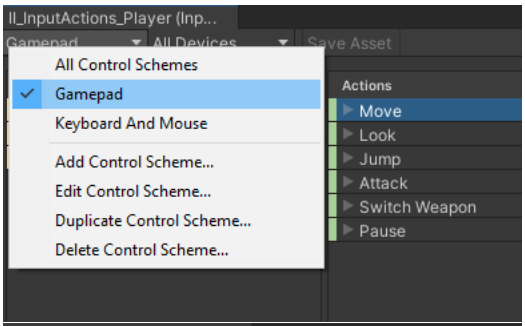


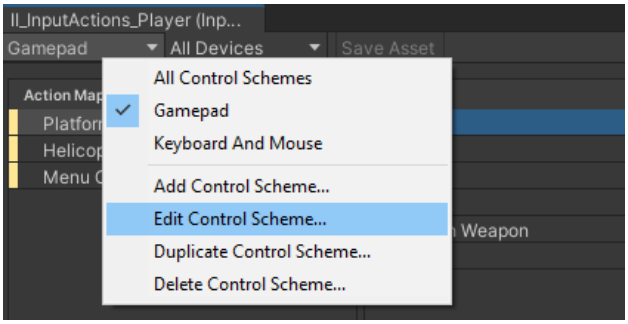
The InputIconsManager needs to know about these names. Fill in the names in the required fields in the setup window or in the Input Icons Manager scriptable object.

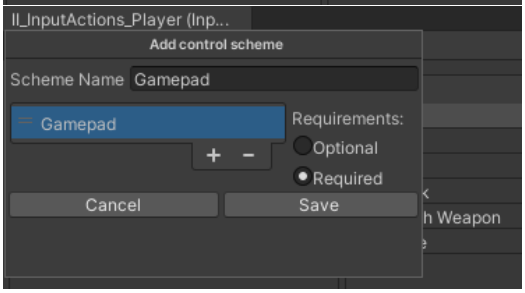


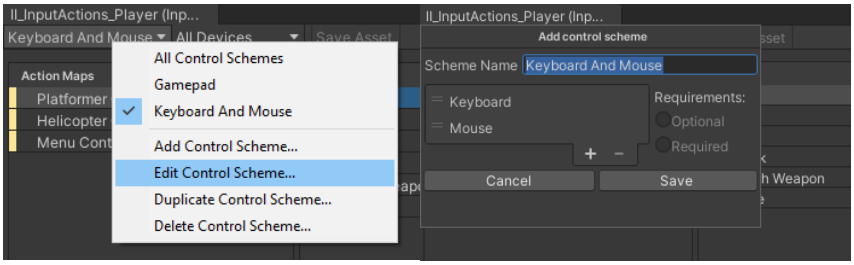
Adding Devices To Control Schemes

We should now have two control schemes. One for gamepads, one for keyboard and mouse. We need to add devices to these control schemes in order for the manager to know which icons to display when the user switches to a different device.







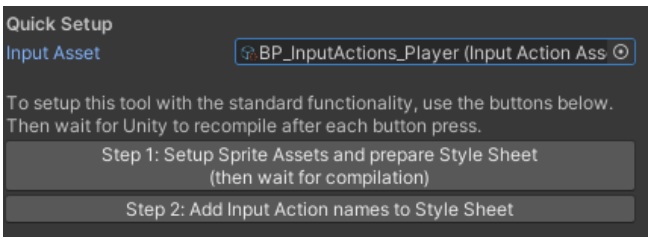


Once we have set up the Input Action Asset and the control scheme names + devices, we can continue with either the Quick Setup (recommended) or the Manual Setup which breaks up the setup process into smaller steps.

Quick Setup

With the Input Icons Setup window open (in the toolbar, select "Tools" and "Input Icons Setup"), use the buttons to set up Input Icons. Here's a detailed guide:

Step 1: With the Input Action Asset ready, create Sprite Assets filled with keyboard and gamepad sprites. They'll be located in the default Sprite Assets folder (usually "Assets/TextMesh Pro/Resources/Sprite Assets"). Also, prepare the TMP Style Sheet with empty values to be filled with the input action names from the Input Action Asset in the next step. (We can't insert action names yet due to limited access to the TMP Style Sheet - hopefully this will change, but for now this is the process).



Step 2: After preparing the style sheet with empty entries, press the second button to insert the input action names from the Input Action Asset.

And that's it! You can now display actions by typing **<style=NameOfActionMap/NameOfAction>** (e.g. to display the move controls for the platformer control scheme, type <style=Platformer Controls/Move>). When the game starts, the InputIconManager will update the default style sheet to display the correct sprites.

Manual Setup

Open "**Tools/Input Icons Setup"** from the toolbar to find the custom setup section. This section comes in handy if at some point we

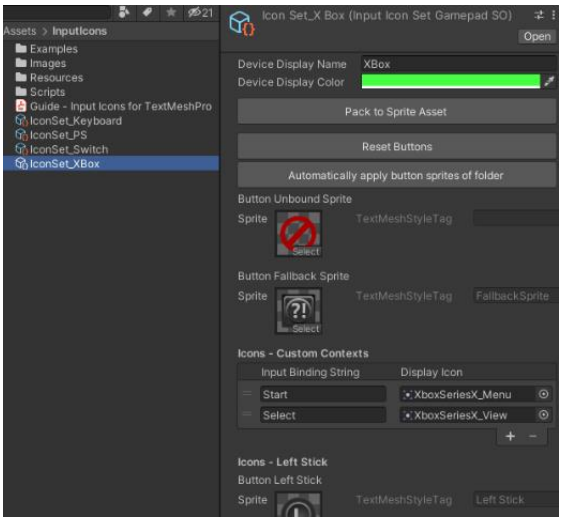
- decide we want to use a different set of sprites
- want to add more Input Action Assets to display more actions (be aware that action names must be unique for each action per action map as it only makes sense to save one instance of an action name into the default style sheet)

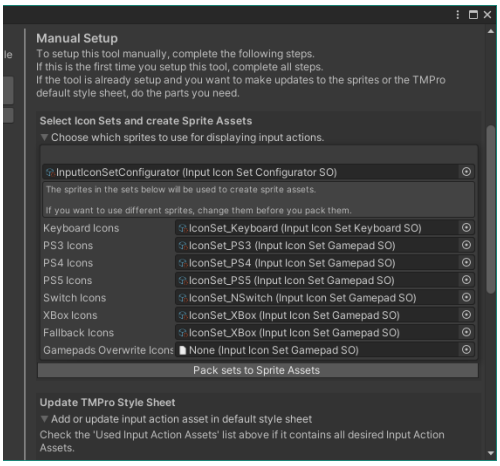
Using Different Sprites

The following steps describe how to use different sets of sprites:

1. In the **Assets/InputIcons folder**, there are 2 types of Scriptable Objects that store the sprites used by the tool: one for keyboard sprites (InputIconSetKeyboardSO) and one for gamepad sprites (InputIconSetGamepadSO).

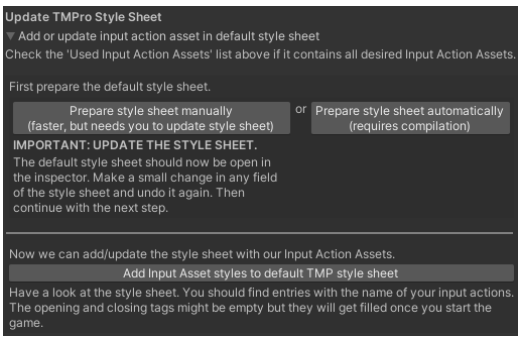
2. To use a different set of sprites, we can duplicate an IconSet and assign our desired sprites to the new Scriptable Object. One efficient way to do this is to drag the Scriptable Object into a folder containing our sprites and then click on “Automatically apply button sprites of folder” in the inspector of the Scriptable Object. This will try to assign as many sprites as possible (the success of this process depends on the names of the sprites).
3. For the “Custom Context” sprites, we have to apply the sprites manually as they cannot be assigned automatically.
4. Once we have our Icon Sets ready, we can drag them into the Manual Setup area of the Setup Window and then hit the “Pack sets to Sprite Assets” button. This will create the necessary sprite assets in the default folder for Sprite Assets (usually “Assets/TextMesh Pro/Resources/Sprite Assets”).
5. If the default folder for Sprite Assets has been moved, we need to let the tool know by updating the corresponding text field at the top of the Setup Window.
6. Sometimes, we may need to enter play mode or recompile the code for the new icons to be displayed.





Adding Or Updating Input Action Assets

We can use the Manual Setup section of the setup window to add and update entries to the TMP default style sheet whenever we make changes to our Input Action Assets.
To summarize the process of preparing the TMP default style sheet:



1. In the Manual Setup section of the Input Icons Setup window, we can either use the first button (Prepare Style Sheet manually) or the second button (Prepare Style Sheet automatically) to add empty values to the TMP default style sheet.
2. If we use the first button, we need to make a simple change in the inspector of the Style Sheet Asset, such as adding a character and then removing it, to save the changes.
3. If we use the second button, Unity will compile and save the changes made to the Style Sheet.
4. Now with the default style sheet prepared for the selected Input Action Asset, we can use the last button to write the action names into the style sheet.

5. It is recommended to use only one Input Action Asset and add control schemes for different controls, such as player controls and menu controls.

IMPORTANT: Add Input Icons Activator Prefab to Scene

It is important to know that Scriptable Objects only call their OnEnable methods when either being selected in the inspector or when being referenced by a MonoBehaviour or when being referenced by another Scriptable Object which is referenced.

Therefore the best way to ensure that the InputIconManagerSO (which is a Scriptable Object) works properly, is to reference it in the scene. We can easily do this by just dragging the **II_InputIconsActivator prefab** into the first scene.

How To Use Input Icons

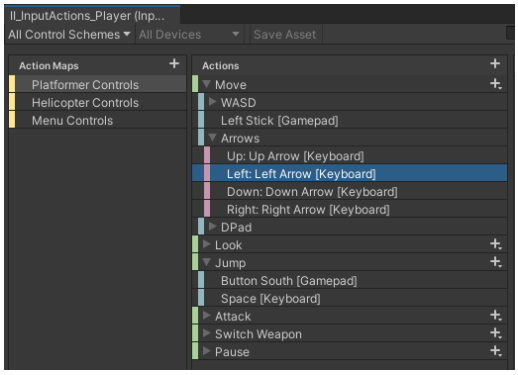
To ensure setup completion:

1. Ensure the names of the Input Action Assets match those in the Input Icons Manager (found in Assets/InputIcons/Resources/InputIcons).
2. Add the desired Input Action Assets to the list of used assets in the setup window or Input Icons Manager Scriptable Object.
3. Create Sprite Assets from Input Icon Sets and store them in "Assets/TextMesh Pro/Resources/Sprite Assets".
4. Add Input Action Asset actions to the TMPro Style Sheet in "Assets/TextMesh Pro/Resources/Style Sheets".

If you moved the TMPro folder to a different location, the Sprite Assets and Style Sheets should be in the corresponding folders.

Displaying Input Bindings in TMPro

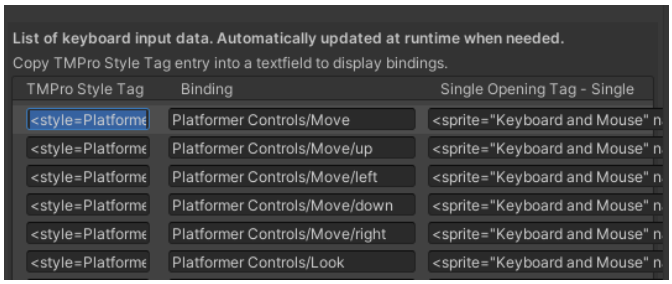
Once we have set up Input Icons, we can display sprite images related to our input actions by using the TMPro-style tag (as found on the website http://digitalnativestudios.com/textmeshpro/docs/rich-text/#style).



When we enter "play mode", the opening tags in the style sheet will automatically be filled with the appropriate tags, displaying the icons for the device currently in use through the Input Icons Manager.

To display the bindings for a specific action, we can insert the following into a TMPro text field: **<style=NameOfActionMap/NameOfAction>.** For example, to display the bindings for a "jump" action in the "Platform Controls" action map, we would insert **<style=Platform Controls/Jump>** into the text field. Similarly, to display the bindings for a "move" action, we would insert **<style=Platform Controls/Move>** which would likely display either the "W A S D" keyboard keys or arrow key sprites, or the left stick sprite if a gamepad is being used. To display only one part of the move action we can write insert **<style=Platform Controls/Move/Down>.**

If we don't want to have to memorize all our action maps and bindings, we can open the Input Icons Manager and scroll down to find a list that contains all the information needed to display bindings. To display a binding, simply copy and paste the entry in the first column (labeled "TMPro Style Tag") into a text field.



Keyboard And Gamepad Support

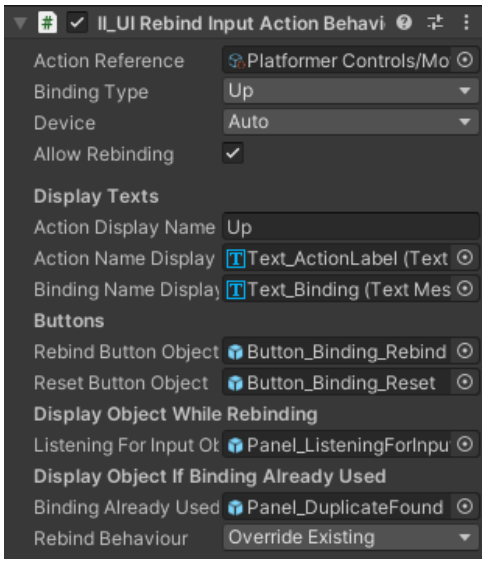
To support keyboards and gamepads, our Input Action Asset(s) need to be correctly setup (see section "**Setting Up The Input Action Asset**")

In earlier versions, the Input Icons Manager would listen for the "Controls Changed" event on a Player Input component. Since Input Icons 1.3.0 this is no longer the case and changing between keyboard and gamepad icons now also works without a Player Input component in the scene.

Rebind Buttons

In the prefab folder we can find the II_UI_RebindActionObject which is a prefab we can use to rebind our input. We might want to create our own prefab with custom visuals but this prefab is a good starting point. (**I recommend to create another prefab out of this prefab so your buttons don't get messed up when you update this asset**)

The script takes an Input Action as a reference which we can then rebind by using the button and then pressing the desired new key.



For composite bindings, we need to select a Binding Type (Up, Down, Left, Right, Forward, Backward) in order to rebind the correct part of the action.

Device: We can select Auto, Keyboard and Mouse or Gamepad. When using Auto, the button will display the icons of the currently used device. The other options will always display either keyboard or gamepad icons and can only be rebound to the device specific bindings.

The toggle "Allow Rebinding" can be turned off for input actions we don't want to be rebound, but should still be visible to the player.

The text field "Action Display Name" provides a quick way to change the text of the action name label without having to select the text object in the hierarchy.

The last field "Rebind Behavior" can be used to choose the wanted behavior in case the player wants to bind a key to an already existing key. This option is global can also be found on the Input Icons Manager. The options are

1. "Override Existing" in that case, the new binding gets accepted and the other action will be unbound.
2. "Cancel Override If Binding Already Exists" will display a message like "Binding already in use" to the player and cancel the rebind function. The player will have to first find and rebind the other action.

Rebinding By Using Your Own Methods Or By Using A Generated C# Class

If you decide to not use the rebind buttons that come with this asset, you can use the "InputIconsManagerSO.ApplyBindingOverridesToInputActionAssets" method to apply binding overrides to the Input Action Assets managed by the InputIconsManager.

For example: You have generated a C# class out of an Input Action Asset and set binding overrides to that class using your own rebinding tools. In this case the InputIconsManager would not know about the changes made. You have to use the above mentioned method to sync up the changes every time you make changes, so the InputIconsManager can display the correct sprites.

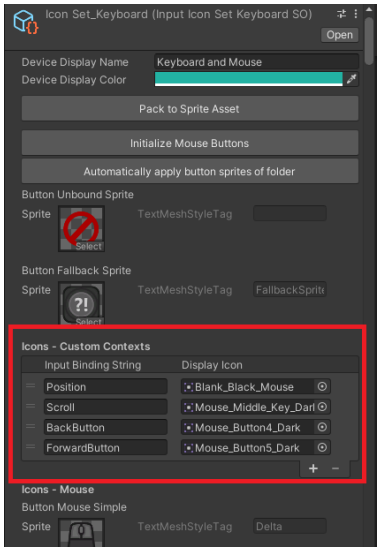
Limitations

- Supported keyboard layouts (with the exception of special characters) are **QWERTY**, **QWERTZ** and **AZERTY.** For all other keyboard layouts, the QWERTY layout (which is the most commonly used layout) will be used to display action bindings.

- If we use more than one Input Action Asset, we should not use the same names for the action maps across these Input Action Assets. Doing so might cause the manager to override style strings in the TMPro Default Style Sheet.

Customization

Adding Custom Context Sprites



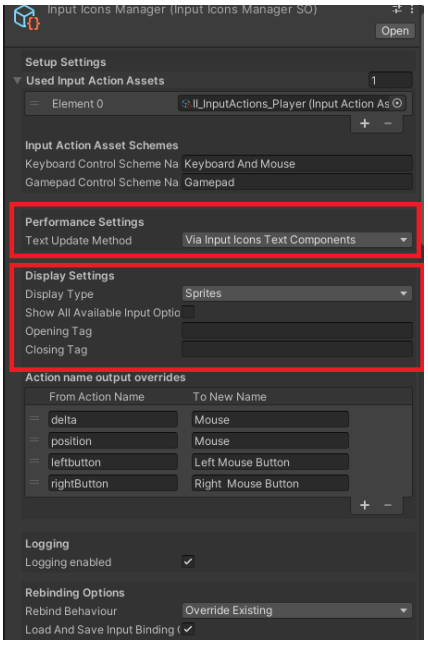
The preset Input Icon Sets that come with this package (in the folder **Assets/InputIcons/**) already have sprites defined for all common input keys, gamepad buttons and joysticks. In case we need more input controls, we can add them in the Custom Contexts list in the Input Icon Sets. We need the input binding string and a display icon, to add a new binding.

An easy way to find the input binding string for an input is to use the script “II_UITextDisplayAllActions” which is in the example folder of this package. Add it to a TextMeshProUGUI object to display current input bindings. Then we can use a rebind button to override the current binding with the new binding we want to add. The input binding string for that binding will be displayed in the brackets. For the mousewheel, the input binding string would be “Scroll” for example.

Important: Whenever we make changes to an Input Icon Set, we will have to create a new Sprite Asset out of the Input Icon Set. Otherwise, our changes only exists in the Input Icon Set, but the desired sprite will not be available in the Sprite Asset used to display sprites within TMPro texts.

Performance Options

We can improve the performance when the user switches devices by selecting the Text Update Method “Via Input Icons Text Components” in the Input Icons Manager.



The default setting is “Search and Update”, which will search for all text objects in the scene and update them when the user switches devices.

By using the “Via Input Icons Text Components” method, the manager won’t have to search through the whole scene every time the user switches devices. But we have to ensure that the required text objects have a InputIconsText component attached to them.

Display Settings

The InputIconsManager provides options for how we want to display the input action bindings.

Show All Available Input Option: We can switch between showing only the first available binding, e.g. “WASD” or all available bindings, e.g. “WASD or Up Left Down Right” for a move action.

Multiple Input Delimiter: If we show all available options, we can define a delimiter between these actions. This delimiter can have TMPro tags for customization. The standard delimiter is “ <size=80%>or</size> “ and will therefore display a slightly smaller “or” between different bindings.

Opening and Closing Tag: Here we can add general styling options for the displayed icons. For example we can write <size=120%> in the opening tag and </size> into the closing tag to make all displayed icons a little bit bigger.

Display Type: input actions as sprites, text, or text in brackets.

Text Display For Unbound Actions: If we display bindings as Text or TextInBrackets, we can choose which text should be displayed for an unbound action.

Text Display Language: If we display bindings as Text or TextInBrackets, we can decide if we want to display this text in English or in System Language (the language currently used by the device, might produce very different results in text length and is generally not recommended).

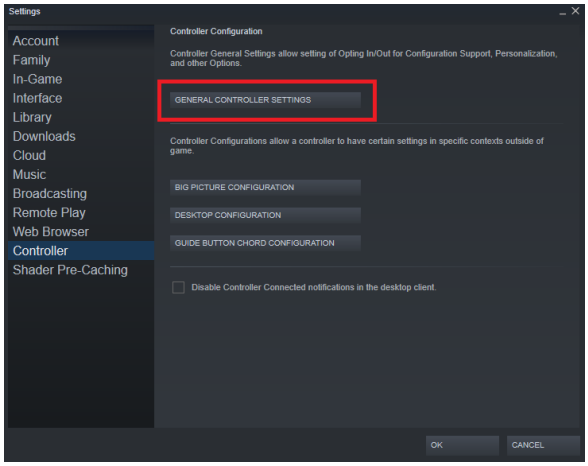
Action name output overrides: If we use Text or TextInBrackets as the option to display input actions, we can override the text which would be displayed. For example it makes more sense to display [MOUSE] instead of [POSITION] for pointer controls.

Steam Integration

Integrating the Steamworks.NET (https://steamworks.github.io/) into the project can cause some problems for the new Input System as Steam seems to literally hijack XInput. Therefore whenever we use steamworks we have to use some helper methods provided by steamworks in order to detect the currently used device. This is done in the script “InputIconsSteamworksExtensionSO” which will be active when we are using steamworks.

Fortunately, we don’t have to do a lot to support Input Icons in Steam, as the above mentioned script already handles the most important things for us. However, there are some limitations when playing via Steam. Since we use the Steam helper methods, it is more difficult to detect the currently used gamepad. For this reason Input Icons will display the used gamepad on index 0, which means that if we have a connected PS controller and a connected XBox controller, Input Icons will always display the first connected controller and the only way to display the other one is to disconnect one controller and restart the game.

Another issue that I noticed when developing for Steam is that the gamepads might sometimes not respond altogether. This can happen in the Unity editor or in a build when the editor or the build is run via Steam. A possible solution is to go into the Steam launcher and go to Steam -> Settings -> Controller -> General Controller Settings and disabling the Configuration Support for PlayStation, Xbox and Switch Pro. Then restart Unity or the game. By doing this, Unity’s Input System will regain control and the checks for the current device will work the as if we were not using Steam. Of course we can not expect players to go into these settings and disable them, but even if they are enabled, our games using Input Icons will still be able to display the correct keyboard and gamepad icons as long as users do not switch between different gamepad types (PS, XBox, Switch).

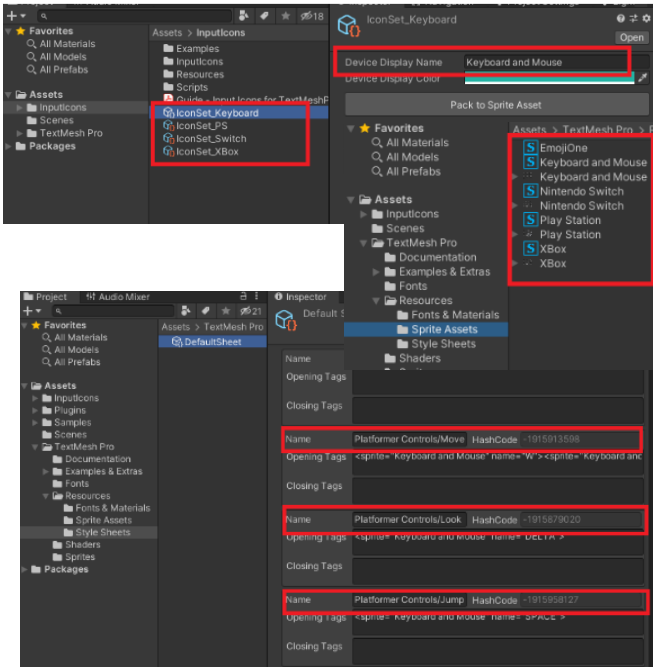


Troubleshooting

Most problems appear from an incorrect setup. Remember, whenever you make changes to an Input Action Asset, to also update the style sheet by using the setup tool (Tools → Input Icons Setup).

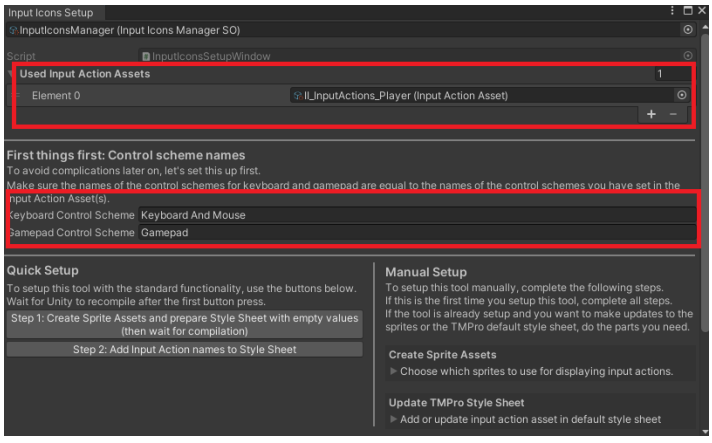
If you have updated to a new version of Input Icons, the values in the InputIconsManager might very likely have changed. To solve this, do the Quick Setup again and everything should work again.

Have our Sprite Assets been created correctly and does the TMPro Default Style Sheet contain our actions?

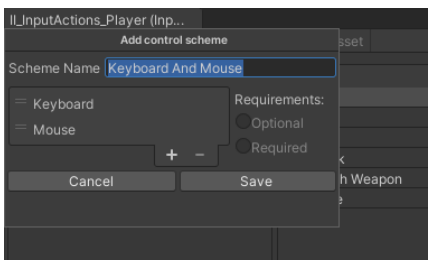


- **Sprite Assets**: In "Assets/TextMesh Pro/Resources/Sprite Assets/" we should find assets containing the sprites of the keys we want to display. It is important that these assets have the same names as the Icon Sets in "Assets/InputIcons/".
- **TMPro Default Style Sheet**: The Default Style Sheet should somewhere contain the actions defined in our Input Action Assets. Check the style sheet in "Assets/TextMesh Pro/Resources/Style Sheets/"
  Special case: The styles were added correctly, but **the styles disappear once we restart Unity** … the Default Style Sheet sometimes won't save the changes. A workaround is to make a small change manually in any of the fields of the style sheet and then undo the change. This seems to work and the changes will be saved.

If we don't see any input icons



- The "**Used Input Action Assets**" list contains all Input Actions we use and we have updated the Default Style Sheet with all changes we made to the Input Action Asset(s).
- All our Input Action Assets have **control schemes for keyboard/gamepad** and their names are **equal to** the ones set in the Input Icons Setup or in the **Input Icons Manager** respectively.



- The **control schemes** of our Input Action Assets have **devices** added to them (see chapter "Adding Devices To Control Schemes")

Icons are shown, but don't get updated when a different device is used:

- Check section "**Keyboard And Gamepad Support**"

Controls are weird or do not working anymore

- The reason might be that you have several active Player Input components in the scene. Search the hierarchy for "Player Input".
- Another reason could be that you run the Unity Player via Steam. Steam can mess up your controls if run simultaneously.

If you have gone through this guide and still experience problems, feel free to contact me at support@octacube-studios.com