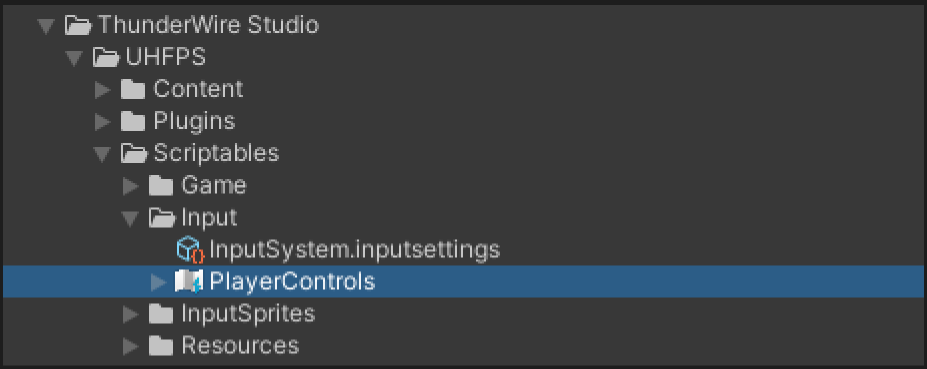
GUIDES

Managing Inputs

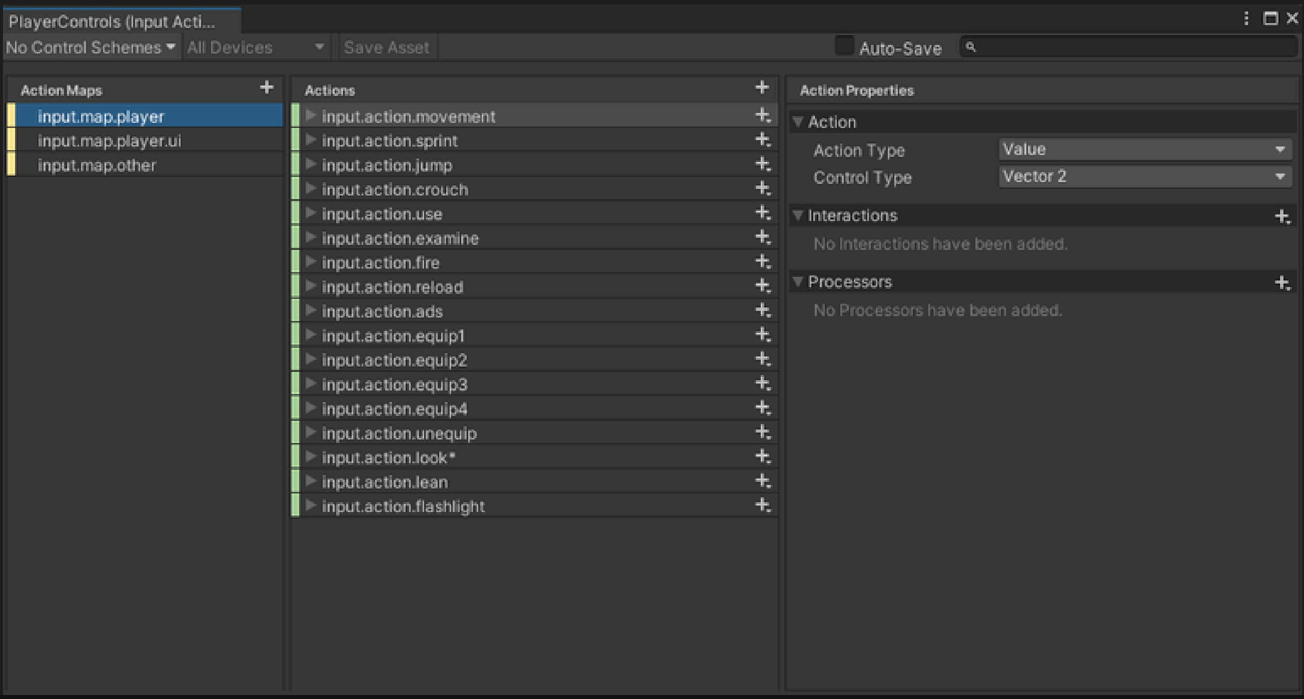
The asset utilizes the **Unity Input System** package, which means modifying inputs may differ from the old built-in Unity Input System. If you're unfamiliar with setting up the Input System package, please refer to the guide provided below.

Setting up Input System

1. Locate the **PlayerControls** asset within the **UHFPS → Scriptables → Input** folder. To open the **Input Actions** window, double-click on the asset.



ThunderWire Studio
UHFPS
Content
Plugins
Scriptables
Game
Input
InputSystem.inputsettings
PlayerControls
InputSprites
Resources



PlayerControls (Input Acti...
No Control Schemes
All Devices
Save Asset
Auto-Save

Action Maps
input.map.player
input.map.player.ui
input.map.other

Actions
input.action.movement
input.action.sprint
input.action.jump
input.action.crouch
input.action.use
input.action.examine
input.action.fire
input.action.reload
input.action.ads
input.action.equip1
input.action.equip2
input.action.equip3
input.action.equip4
input.action.unequip
input.action.look*
input.action.lean
input.action.flashlight

Action Properties
Action
Action Type: Value
Control Type: Vector 2
Interactions
No Interactions have been added.
Processors
No Processors have been added.

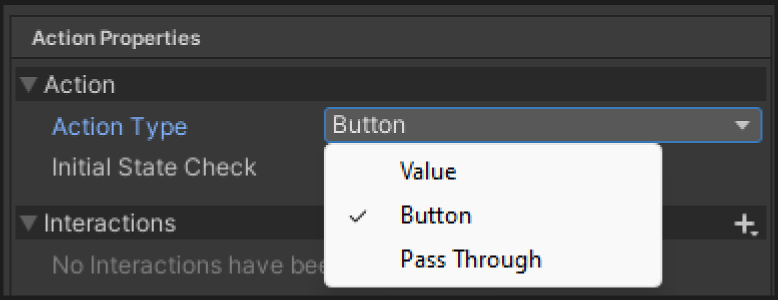
Upon opening the Input Actions window, you may notice that the action names follow the pattern **input.action.something**. This is because action names serve as localization keys, allowing for easy translation of action names.

2. To create a new input action, simply click on the **(+)** icon found to the right of the **Actions** tab name. Enter the name of the action in the localization key pattern.

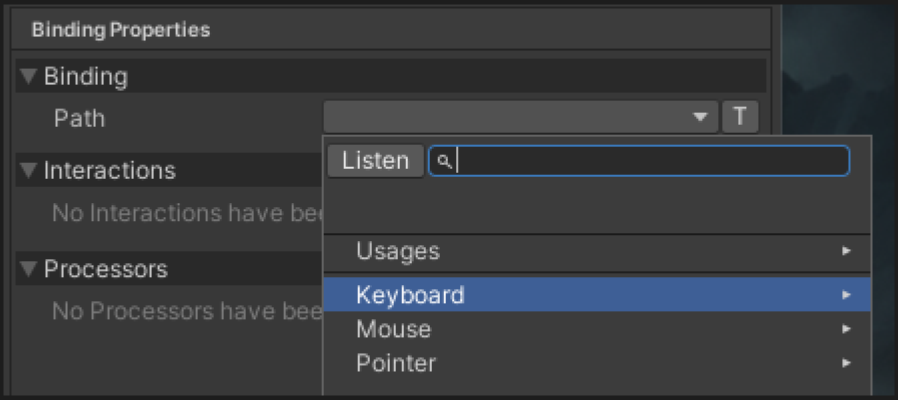
Keep in mind that actions added to the first Action Map (**input.map.player**) will be only displayed in the **Main/Pause** menu. This means that these actions can only be rebound and serialized.

Input names containing an **asterisk (*)** will be automatically excluded from menu display and serialization. For instance, **input.action.look*** will be disregarded.

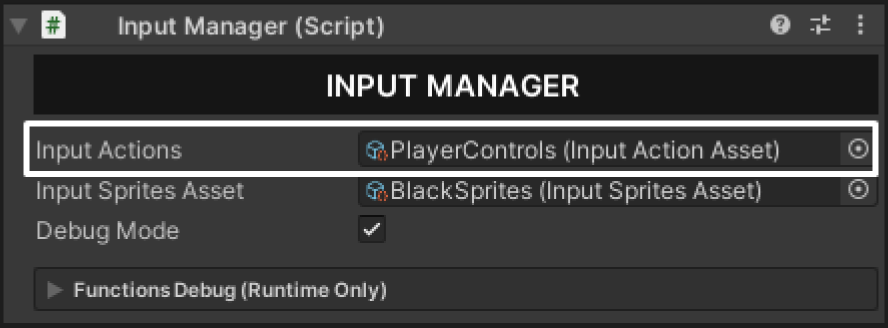
3. Select the newly created action and modify the **Action Type** to the desired input type.



4. Next, click on the **<No Binding>** and choose the key you want your action to utilize.



5. If you are using a custom controls asset, ensure that you assign the reference to this asset within the **Input Manager** component.



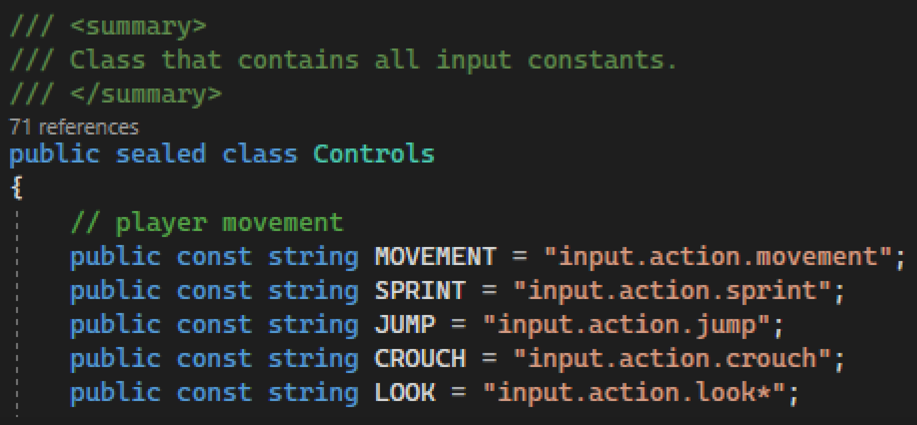
Input Actions: PlayerControls (Input Action Asset)
Input Sprites Asset: BlackSprites (Input Sprites Asset)
Debug Mode: ✓

6. In order to access input actions within a script, simply use the **Input Manager** functions. There are multiple ways to utilize the input.

In Update | Performed | Observable

```
using UnityEngine;
using UHFPS.Input;

private void Update()
{
    if (InputManager.ReadButtonToggle(this, Controls.CROUCH))
    {
        // do something when toggled on
        // even if you don't hold any button
    }

    if (InputManager.ReadButtonOnce(this, Controls.FIRE))
    {
        // do something once as button
    }

    if (InputManager.ReadButton(Controls.FIRE))
    {
        // do something every update as button
    }

    if(InputManager.ReadInput(Controls.LEAN, out float direction))
    {
        // do something every update with value
    }
}
```

You can find a script called **Controls**, which stores all the input names so that you don't have to keep typing out the whole input name over and over again.



```
/// <summary>
/// Class that contains all input constants.
/// </summary>
71 references
public sealed class Controls
{
    // player movement
    public const string MOVEMENT = "input.action.movement";
    public const string SPRINT = "input.action.sprint";
    public const string JUMP = "input.action.jump";
    public const string CROUCH = "input.action.crouch";
    public const string LOOK = "input.action.look*";
```

Previous
Support

Next
Setting up Input System

Last updated 10 months ago