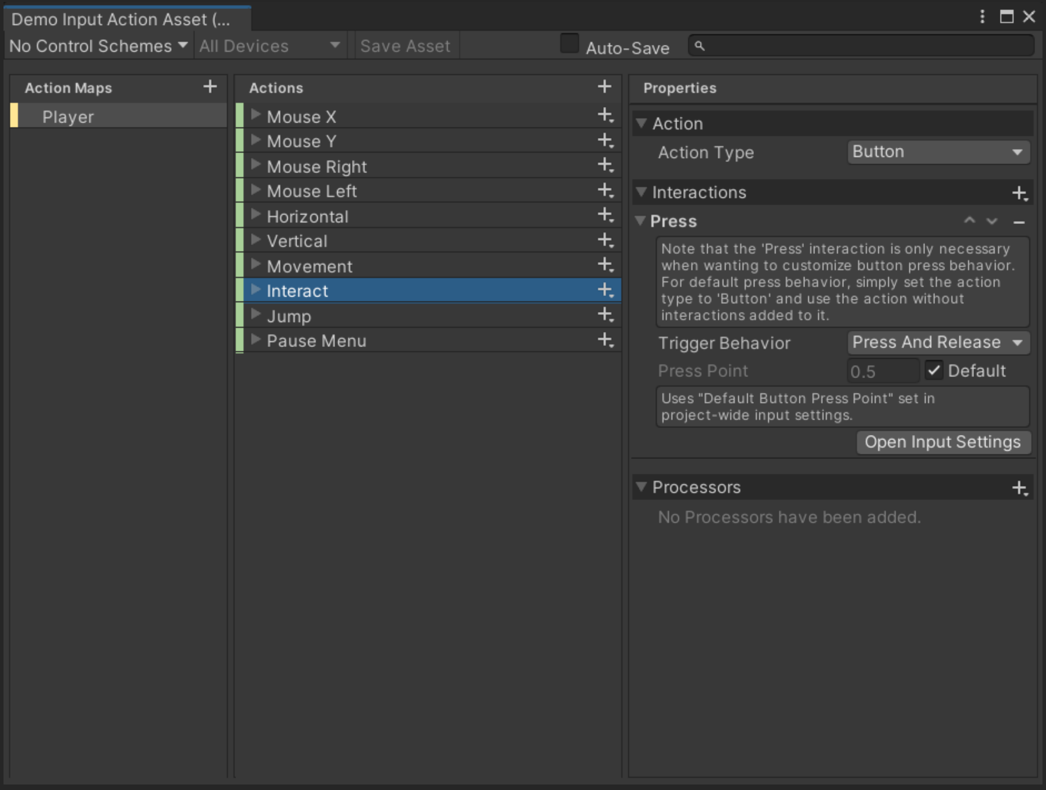
New Input By Reference Documentation

by The BlindEye

Contents

1. What is included with the Package?

- **Demo Input Action Asset** - it is an Input Action Asset that contains a collection of Input Actions that can be extended upon.



Action Type: Button

Note that the 'Press' interaction is only necessary when wanting to customize button press behavior. For default press behavior, simply set the action type to 'Button' and use the action without interactions added to it.

Trigger Behavior: Press And Release

Press Point: 0.5 ✓ Default

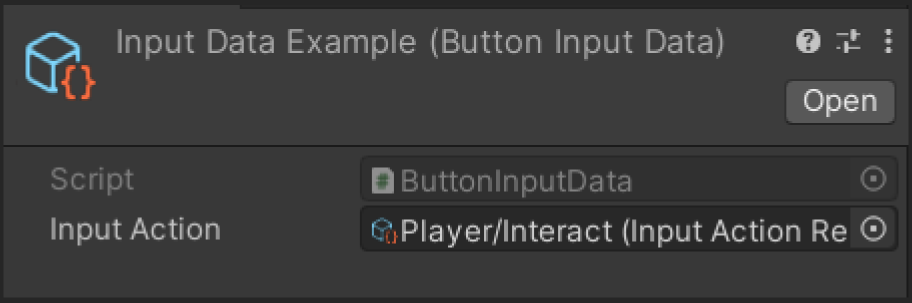
Uses "Default Button Press Point" set in project-wide input settings.

No Processors have been added.

Remark: All **Button Action Types** require a **Press Interaction** with the **Trigger Behavior** set to **Press and Release** in order for the NewInput.GetButtonDown function to behave properly (as shown in the image above).

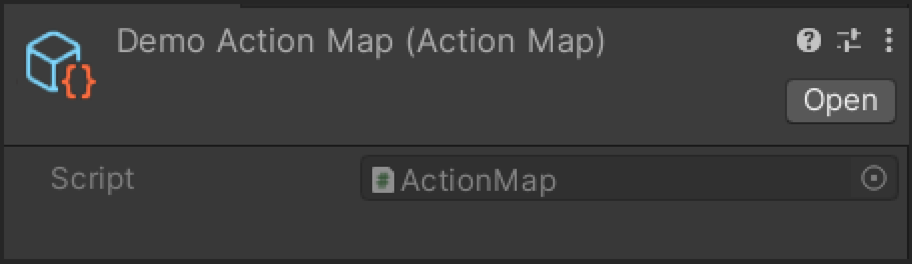
- **Demo Input Handler** - it is a Prefab that's already configured to function with the Demo Input Action Asset and can be used to skip the steps without * from point 3 of the Contents.

- **Input Data -** it is a Scriptable Object that stores a reference to an Input Action.



Script: ButtonInputData
Input Action: Player/Interact (Input Action Re

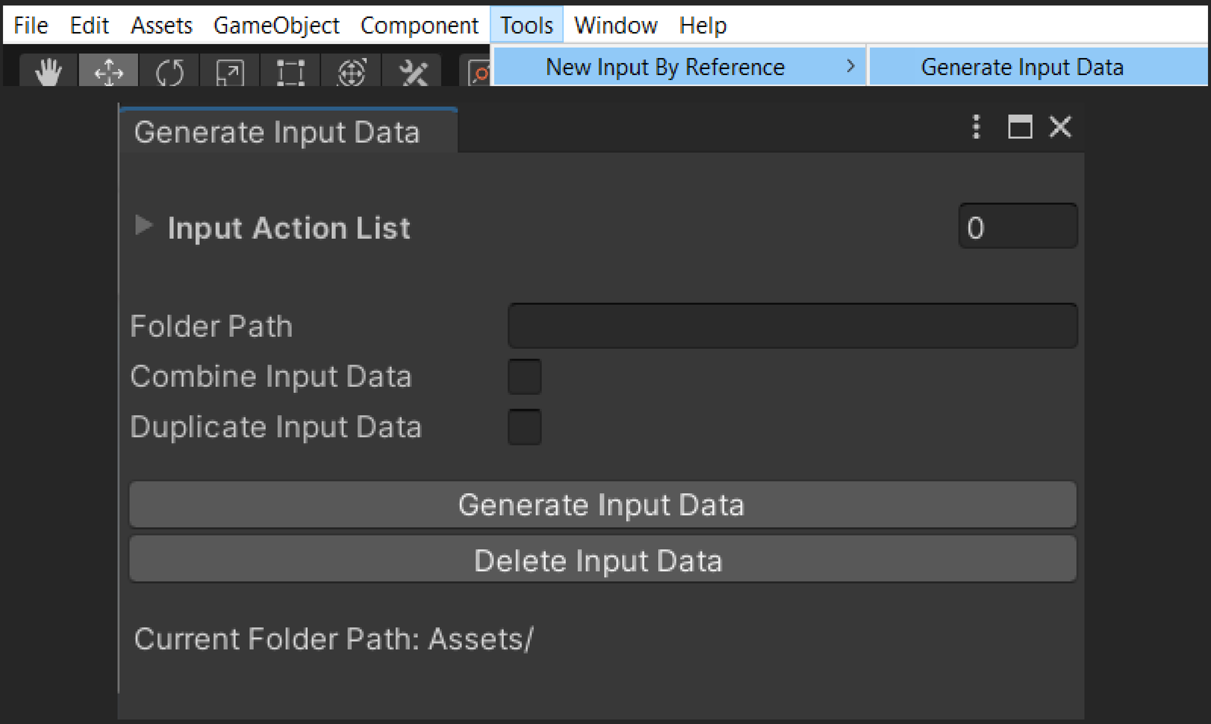
- **Action Map** - it is a Scriptable Object that stores a List of the Input Data inside it, that can be accessed by using **ActionMap.InputDataList.** It can be only generated using Generate Input Data Window.



Script: ActionMap

Remark: A **Demo Action Map** is included in the Package, which was generated from the **Demo Input Action Asset**.

- **Generate Input Data Window** - it is a Custom Editor Window that takes a list of Input Actions and generates Input Data from it.



Folder Path: Provide a Folder Path. Default Folder Path is *Assets/*

Combine Input Data: Combine generated Input Data into a single Action Map

Duplicate Input Data: Generate Input Data even if in the Current Folder Path is already a duplicate of it

Current Folder Path: The location where the Input Data will be generated/deleted

Generate Input Data: Generate Input Data to the Current Folder Path

Delete Input Data: Delete Input Data from the Current Folder Path

2. What are the Functions?

- To access the functions, **using NewInputByReference;** is needed at the top of the script.

Action Map Functions

`void NewInput.SwitchActionMap(string actionMap)`

Summary	Change the current Action Map to the one indicated by "actionMap".
`actionMap`	Can be found in the Input Action Asset, under the "Action Maps" column.

`Void NewInput.EnableActionMap(string actionMap)`

Summary	Enable the Action Map indicated by "actionMap". Other Action Maps will not be disabled if they are currently active.
`actionMap`	Can be found in the Input Action Asset, under the "Action Maps" column.

`void NewInput.DisableActionMap(string actionMap)`

Summary	Disable the Action Map indicated by "actionMap". Other Action Maps will not be disabled if they are currently active.
`actionMap`	Can be found in the Input Action Asset, under the "Action Maps" column.

Rebind Functions

```
    void NewInput.StartRebinding(
InputData inputData,
int bindingIndex = 0,
bool allCompositeBindings = false,
Action onComplete = null,
InputData cancelInputData = null,
InputData[] excludedInputData = null)
```

Summary	Start the rebinding process for the input action indicated by "inputData", by waiting for a key input from the player.
`inputData`	Can be generated using Generate Input Data Window, or created from Create/New Input By Reference.
`bindingIndex`	Index of the input action, indicated by "inputData", binding that will be rebounded. If "isComposite" is true, "bindingIndex" becomes the starting index of the composite. Can be determined in the Input Action Asset, under the "Actions" column -> (any) Action -> Left DropDown Arrow. (check Documentation -> "6. What are some tips?", to understand how they are numbered)
`allCompositeBindings`	Can be true only if the input action indicated by "inputData" is of Action Type = Value and Control Type = Axis/Vector 2/Vector 3. If it's true, all the input actions from the same composite, as the input action indicated by "inputData", will start the rebinding process one after another. After all of them have been rebounded, the function will exit. (check Documentation -> "6. What are some tips?", to understand what is a composite)
`onComplete`	Delegate that invokes when the rebinding process is completed.
`cancelInputData`	Input action indicated by "cancelInputData" that upon pressing makes the function exit, and stops listening to the player's input. (Default Action Path = <Keyboard>/escape)
`excludedInputData`	Input actions indicated by "excludedInputData" that upon press will not rebind(nor exit the function) the input action indicated by "inputData". (excludedInputData.Length <= 3)

```
void NewInput.Rebind(InputData inputData, string bindingPath, int bindingIndex = 0)
```

Summary	Rebind the input action indicated by "inputData".
`inputData`	Can be generated using Generate Input Data Window, or created from Create/New Input By Reference.
`bindingPath`	Can be determined in the Input Action Asset, under the "Actions" column -> (any) Action -> Path. (e.g. <Keyboard>/e, <Gamepad>/leftStick) Remark: To avoid GC, the passed variable should be cached into the class.
`bindingIndex`	Index of the input action, indicated by "inputData", binding that will be rebounded. Can be determined in the Input Action Asset, under the "Actions" column -> (any) Action -> Left DropDown Arrow. (check Documentation -> "6.What are some tips?", to understand how they are numbered)

```
void NewInput.Rebind(string actionName, string bindingPath, int bindingIndex = 0)
```

Summary	Rebind the input action indicated by "actionName".
`actionName`	Can be found in the Input Action Asset, under the "Actions" column.
`bindingPath`	Can be determined in the Input Action Asset, under the "Actions" column -> (any) Action -> Path. (e.g. <Keyboard>/e, <Gamepad>/leftStick) Remark: To avoid GC, the passed variable should be cached into the class.
`bindingIndex`	Index of the input action, indicated by "actionName", binding that will be rebounded. Can be determined in the Input Action Asset, under the "Actions" column -> (any) Action -> Left DropDown Arrow. (check Documentation -> "6.What are some tips?", to understand how they are numbered)

```
string NewInput.GetBindingName(InputData inputData, int bindingIndex = 0)
```

Summary	Get the name of the current binding indicated by "inputData". (e.g. W, E, Up Arrow, Tab, etc.)
`inputData`	Can be generated using Generate Input Data Window, or created from Create/New Input By Reference.
`bindingIndex`	Index of the input action, indicated by "inputData", binding that will provide the name. If "bindingIndex" = 0 and the input action, indicated by "actionName", is a composite, the function will return all the bindings' names separated by "/". (e.g. W/S/A/D, S/W, A/D) Can be determined in the Input Action Asset, under the "Actions" column -> (any) Action -> Left DropDown Arrow. (check Documentation -> "6.What are some tips?", to understand how they are numbered)

```
string NewInput.GetBindingName(string actionName, int bindingIndex = 0)
```

Summary	Get the name of the current binding indicated by "inputData". (e.g. W, E, Up Arrow, Tab, etc.)
`actionName`	Can be found in the Input Action Asset, under the "Actions" column.
`bindingIndex`	Index of the input action, indicated by "actionName", binding that will provide the name. If "bindingIndex" = 0 and the input action, indicated by "actionName", is a composite, the function will return all the bindings' names separated by "/". (e.g. W/S/A/D, S/W, A/D) Can be determined in the Input Action Asset, under the "Actions" column -> (any) Action -> Left DropDown Arrow. (check Documentation -> "6.What are some tips?", to understand how they are numbered)

```
void NewInput.ResetRebinds()
```

Summary	Reset all rebinds of the current Input Action Asset, attached to the Player Input Component, to the default value.

```
void NewInput.ResetRebind(InputData inputData, int bindingIndex = 0)
```

Summary	Reset the rebinding of the input action indicated by "inputData".
`inputData`	Can be generated using Generate Input Data Window, or created from Create/New Input By Reference.
`bindingIndex`	Index of the input action, indicated by "inputData", binding that will be reset. If "bindingIndex" = 0, the function will reset all rebinds of the input action, indicated by "inputData". Can be determined in the Input Action Asset, under the "Actions" column -> (any) Action -> Left DropDown Arrow. (check Documentation -> "6.What are some tips?", to understand how they are numbered)

```
void NewInput.ResetRebind(string actionName, int bindingIndex = 0)
```

Summary	Reset the rebinding of the input action indicated by "actionName".
`actionName`	Can be found in the Input Action Asset, under the "Actions" column.
`bindingIndex`	Index of the input action, indicated by "actionName", binding that will be reset. If "bindingIndex" = 0, the function will reset all rebinds of the input action, indicated by "actionName". Can be determined in the Input Action Asset, under the "Actions" column -> (any) Action -> Left DropDown Arrow. (check Documentation -> "6.What are some tips?", to understand how they are numbered)

Button Action Type Functions

`bool NewInput.GetButtonDown(string actionName)`

Summary	Returns true if the user pressed the virtual Button indicated by "actionName" during the current frame.
`actionName`	Can be found in the Input Action Asset, under the "Actions" column. (Action Type = Button)

`bool NewInput.GetButtonDown(InputData inputData)`

Summary	Returns true if the user pressed the virtual Button indicated by "inputData" during the current frame.
`inputData`	Can be generated using Generate Input Data Window, or created from Create/New Input By Reference/Button Input Data.

`bool NewInput.GetButton(string actionName)`

Summary	Returns true if the user had pressed and did not release the virtual Button indicated by "actionName" during the current frame.
`actionName`	Can be found in the Input Action Asset, under the "Actions" column. (Action Type = Button)

`bool NewInput.GetButton(InputData inputData)`

Summary	Returns true if the user had pressed and did not release the virtual Button indicated by "inputData" during the current frame.
`inputData`	Can be generated using Generate Input Data Window, or created from Create/New Input By Reference/Button Input Data.

`bool NewInput.GetButtonUp(string actionName)`

Summary	Returns true if the user released the virtual Button indicated by "actionName" during the current frame.
`actionName`	Can be found in the Input Action Asset, under the "Actions" column. (Action Type = Button)

`bool NewInput.GetButtonUp(InputData inputData)`

Summary	Returns true if the user released the virtual Button indicated by "inputData" during the current frame.
`inputData`	Can be generated using Generate Input Data Window, or created from Create/New Input By Reference/Button Input Data.

Value Action Type Functions

`float NewInput.GetAxis(string actionName)`

Summary	Returns the value of the virtual Axis indicated by "actionName".
`actionName`	Can be found in the Input Action Asset, under the "Actions" column. (Action Type = Value, Control Type = Axis)

`float NewInput.GetAxis(InputData inputData)`

Summary	Returns the value of the virtual Axis indicated by "inputData".
`inputData`	Can be generated using Generate Input Data Window, or created from Create/New Input By Reference/Axis Input Data.

`Vector2 NewInput.GetVector2(string actionName)`

Summary	Returns the value of the virtual Vector2 indicated by "actionName".
`actionName`	Can be found in the Input Action Asset, under the "Actions" column. (Action Type = Value, Control Type = Vector 2)

`Vector2 NewInput.GetVector2(InputData inputData)`

Summary	Returns the value of the virtual Vector2 indicated by "inputData".
`inputData`	Can be generated using Generate Input Data Window, or created from Create/New Input By Reference/Vector2 Input Data.

`Vector3 NewInput.GetVector3(string actionName)`

Summary	Returns the value of the virtual Vector3 indicated by "actionName".
`actionName`	Can be found in the Input Action Asset, under the "Actions" column. (Action Type = Value, Control Type = Vector 3)

```
Vector3 NewInput.GetVector3(InputData inputData)
```

Summary	Returns the value of the virtual Vector3 indicated by "inputData".
`inputData`	Can be generated using Generate Input Data Window, or created from Create/New Input By Reference/Vector3 Input Data.

- There are also shortcuts for the functions above when using Input Data references.

  Button Action Type

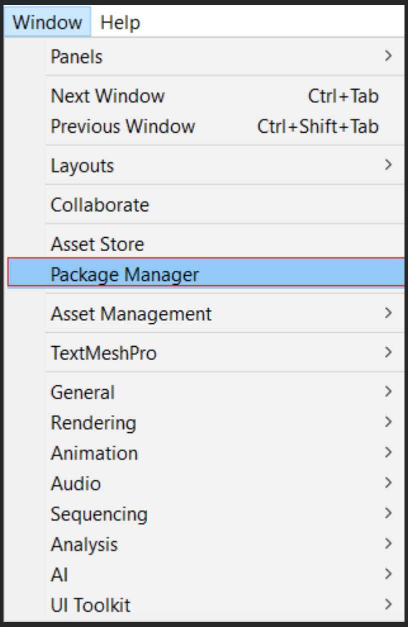
  ```
  bool ButtonInputData.ButtonPressed
  bool ButtonInputData.ButtonHolding
  bool ButtonInputData.ButtonReleased
  ```

  Value Action Type

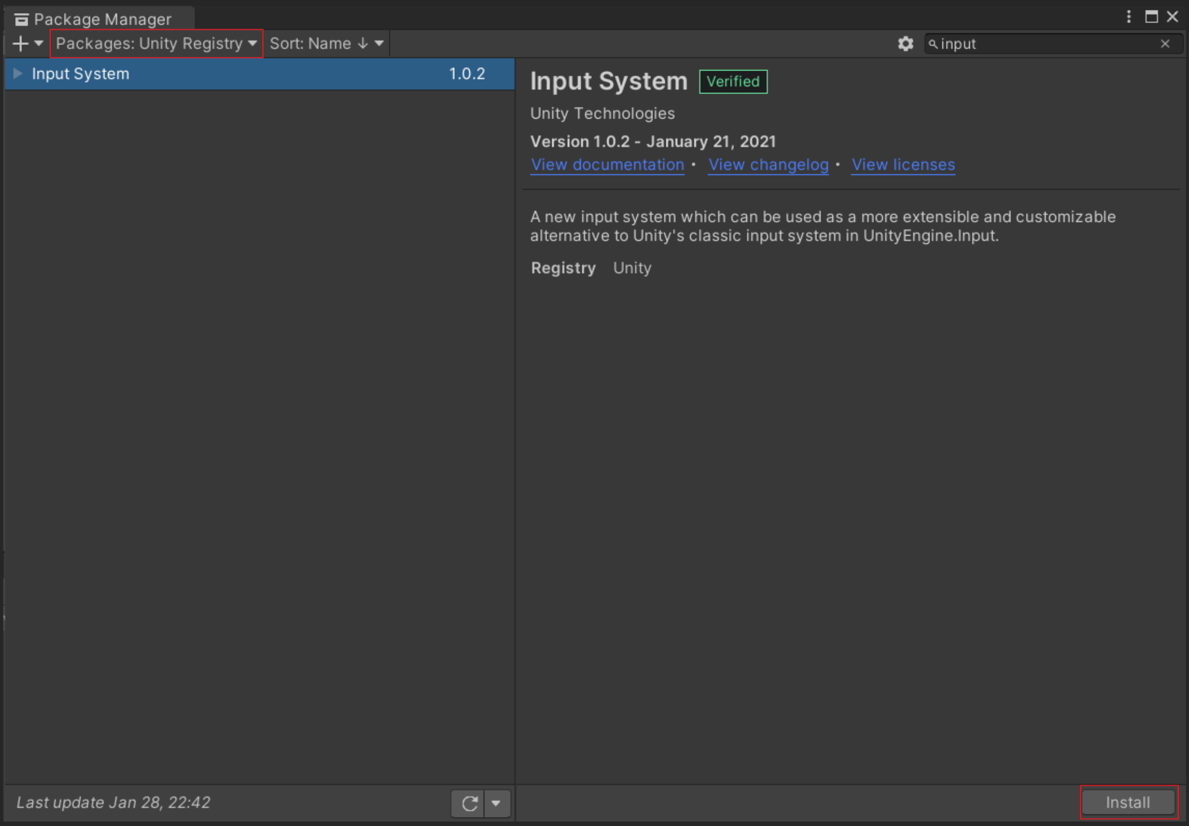
  ```
  float AxisInputData.Axis
  Vector2 Vector2InputData.Vector2
  Vector3 Vector3InputData.Vector3
  ```

3. How to set up a Project to use the Package?

- *Open the **Package Manager Window** from Window/Package Manager



- *Install the **Input System Package** from the **Unity Registry**



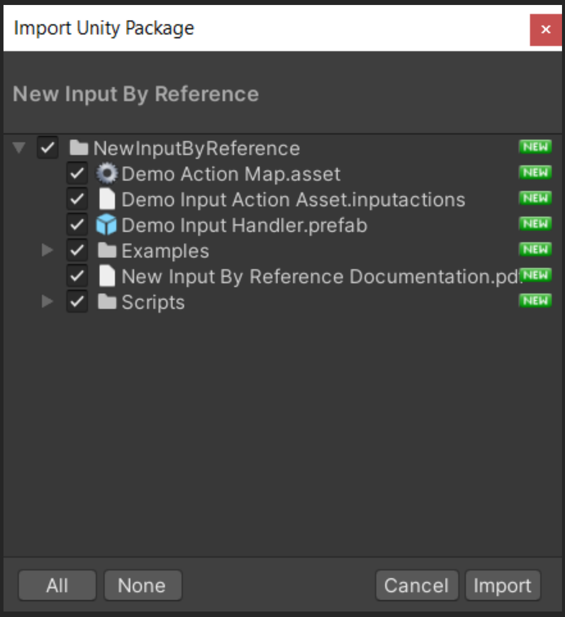
Input System
Unity Technologies
Version 1.0.2 - January 21, 2021

A new input system which can be used as a more extensible and customizable alternative to Unity's classic input system in UnityEngine.Input.

Registry Unity

Last update Jan 28, 22:42

- *Import the **New Input By Reference Package** into the project.

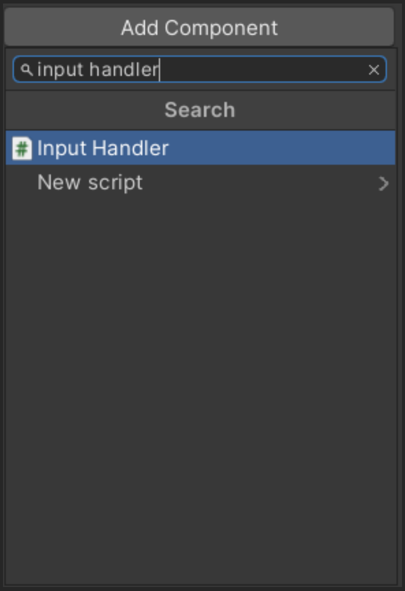


Import Unity Package

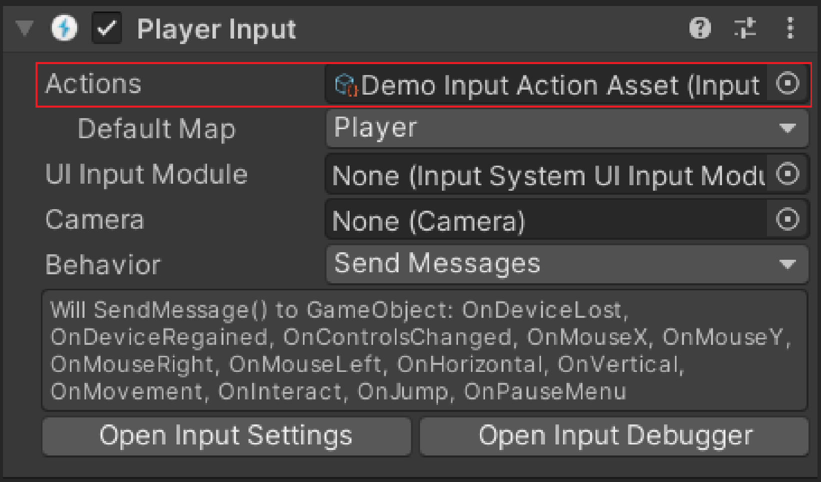
New Input By Reference

- NewInputByReference
  - Demo Action Map.asset
  - Demo Input Action Asset.inputactions
  - Demo Input Handler.prefab
  - Examples
  - New Input By Reference Documentation.pd
  - Scripts

- There must be an **Input Handler** script attached to a game object in the scene (frequently found on the player).



- On the **Player Input Component** which was added by the Input Handler script, an **Input Action Asset** needs to be assigned to the **Actions field** (the package comes with a demo version of an Input Action Asset that can be used out of the box).



Player Input

Actions	Demo Input Action Asset (Input
Default Map	Player
UI Input Module	None (Input System UI Input Modu
Camera	None (Camera)
Behavior	Send Messages

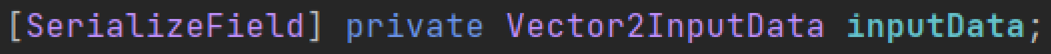
Will SendMessage() to GameObject: OnDeviceLost, OnDeviceRegained, OnControlsChanged, OnMouseX, OnMouseY, OnMouseRight, OnMouseLeft, OnHorizontal, OnVertical, OnMovement, OnInteract, OnJump, OnPauseMenu

4. What do custom errors indicate?

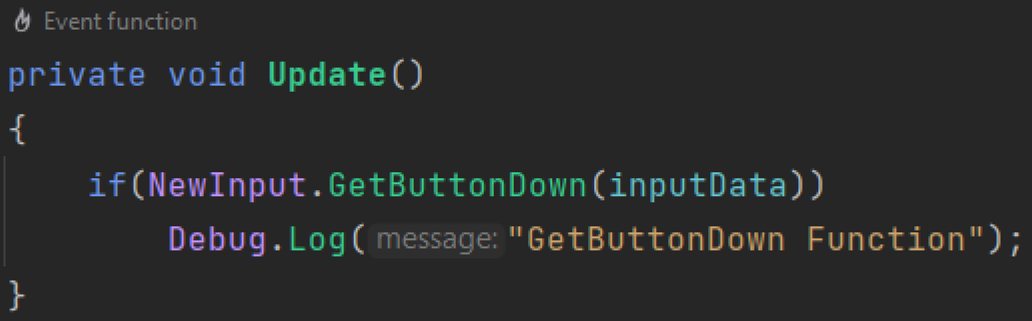
- Invalid Control Type for Action Name. Got Type, expected Type. Action Map: Input Action Asset: Action Map .

The error indicates that a **wrong parameter** was passed to a *NewInput.* function.

In this example, we are passing a **Vector2InputData** as a parameter for the **NewInput.GetButtonDown** (a function that can only take **ButtonInputData** or **Button Type Action Name**).



```
[SerializeField] private Vector2InputData inputData;
```



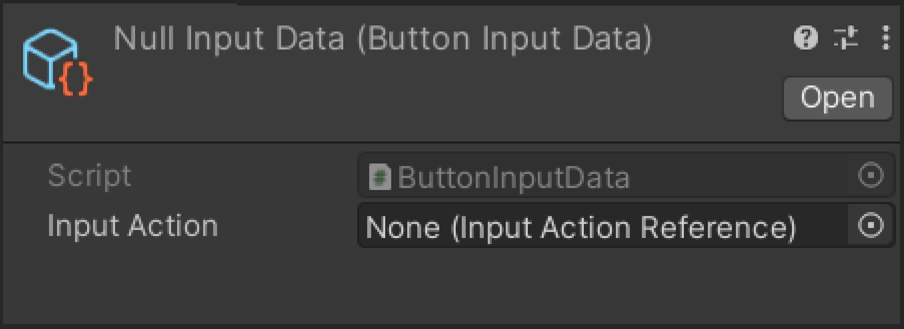
```
Event function
private void Update()
{
    if(NewInput.GetButtonDown(inputData))
        Debug.Log( message: "GetButtonDown Function");
}
```

We're getting the following error: Invalid Control Type for Action Movement. Got Vector2, expected Single. Action Map: Demo Input Action Asset (UnityEngine.InputSystem.InputActionAsset):Player.

To solve the error, we need to modify the **inputData** field's type to **ButtonInputData**.

- Input Action is Null. Ensure that an Input Action is assigned to all Input Data.

The error indicates that an **Input Action field** from an **Input Data** is Null.


Null Input Data (Button Input Data)

Open

Script ButtonInputData

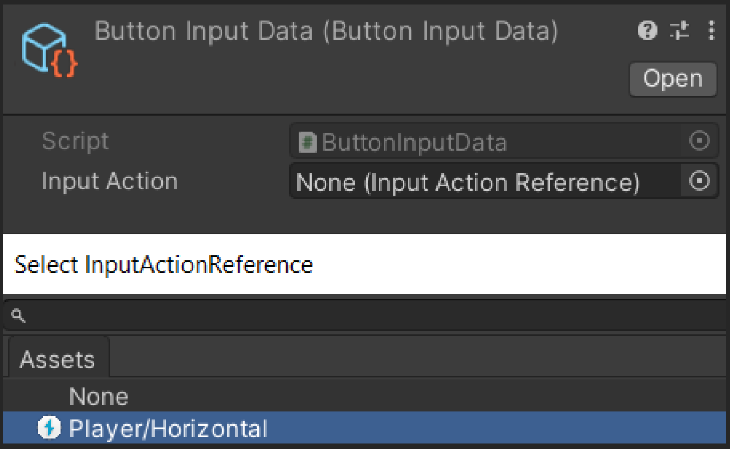
Input Action None (Input Action Reference)

To solve this error, we need to assign an **Input Action** to the **Input Data**.

- Invalid Control Type for Action Name.

The error indicates a **wrong Input Action Control Type** assigned to the **Input Data**.

In this example, we are trying to assign **Horizontal Action** to a **Button Input Data**. As a result, we are getting the following error: Invalid Control Type for Action Horizontal.


Button Input Data (Button Input Data)

Open

Script ButtonInputData

Input Action None (Input Action Reference)

Select InputActionReference

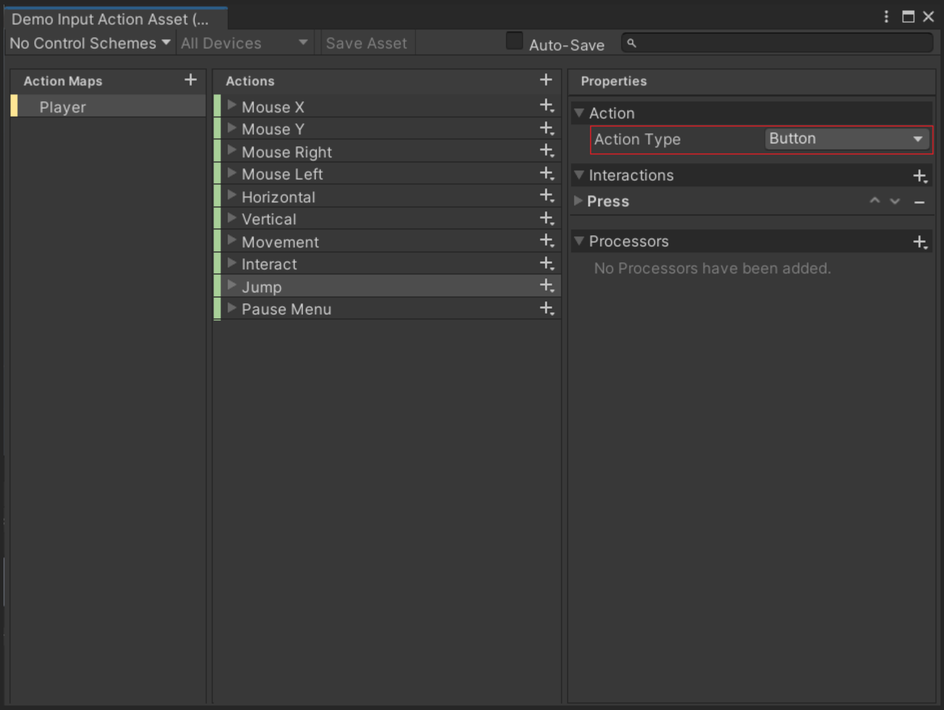
Assets

None

Player/Horizontal

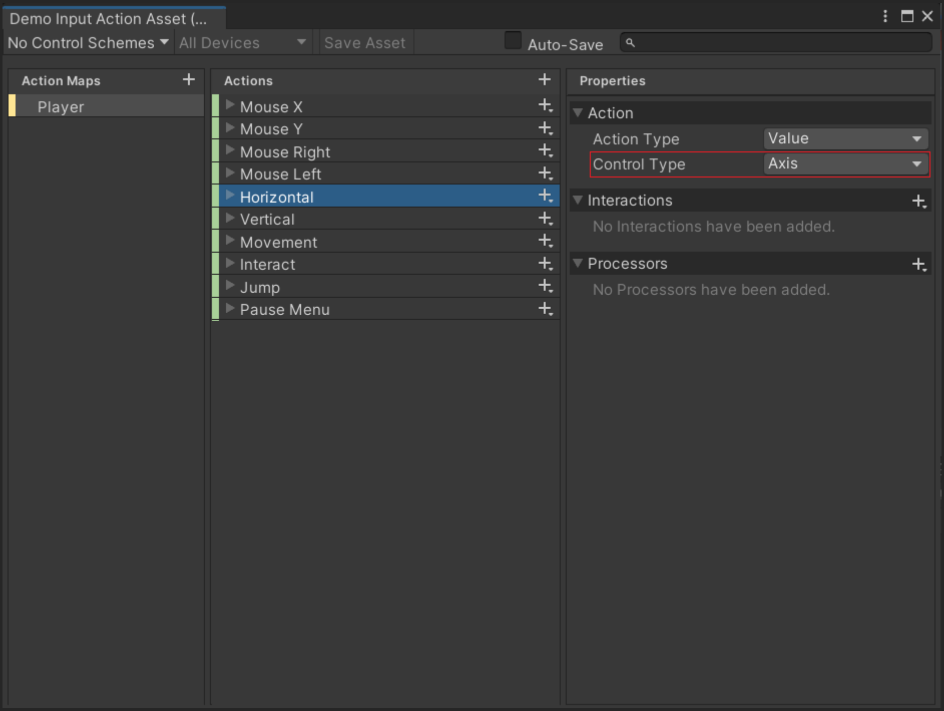
To solve the error, we need to assign a **Button Control Type Action** to the field.

Remark: The **Control Type** field can be found in the **Properties Column** of the **Input Action Asset**. Keep in mind that while Unity does not display a **Control Type** field for **Button Action Types**, it assigns it to **Button** behind the scenes.



Action Type Button

No Processors have been added.



Action Type Value

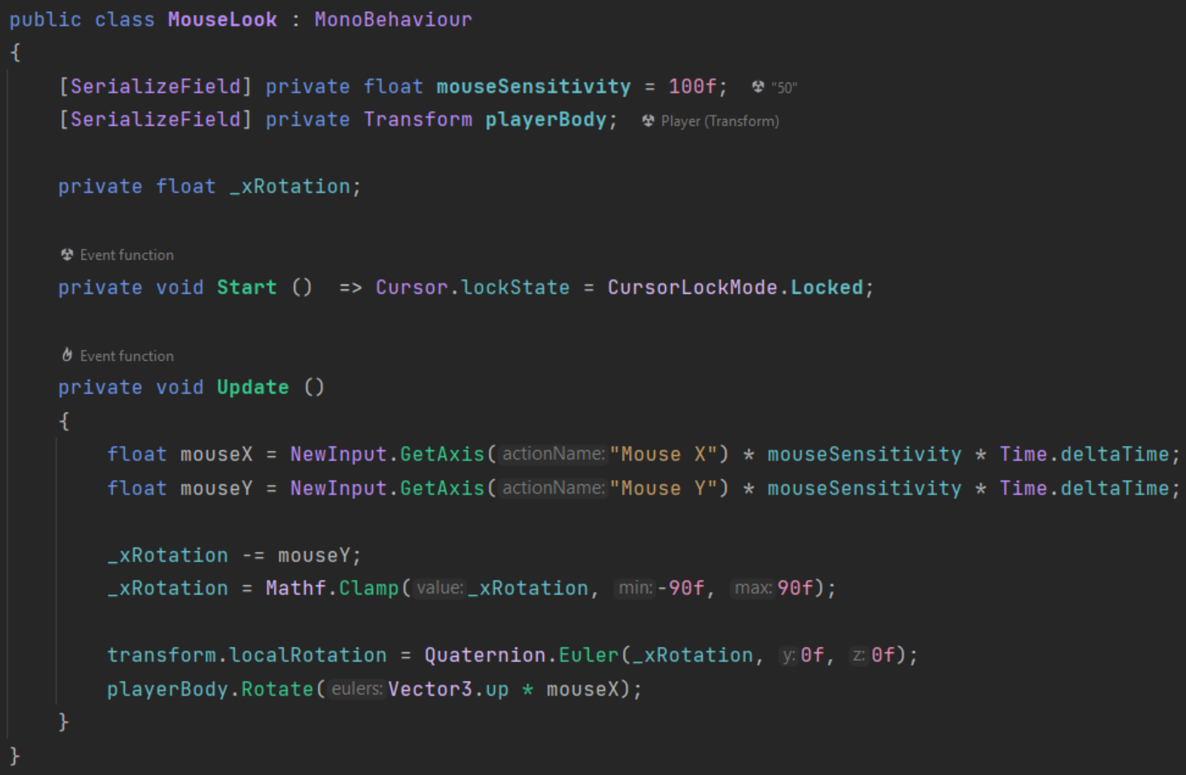
Control Type Axis

No Interactions have been added.

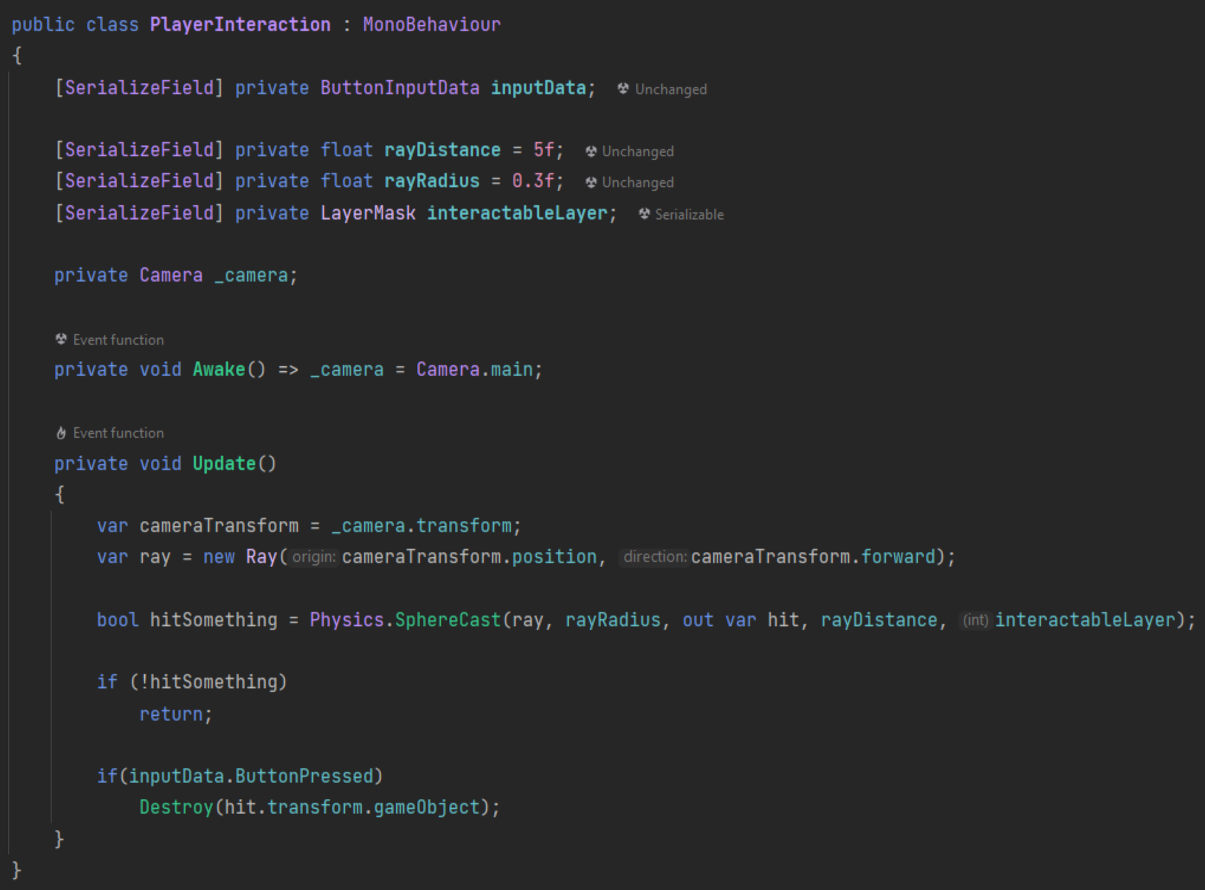
No Processors have been added.

5. What are some script examples?

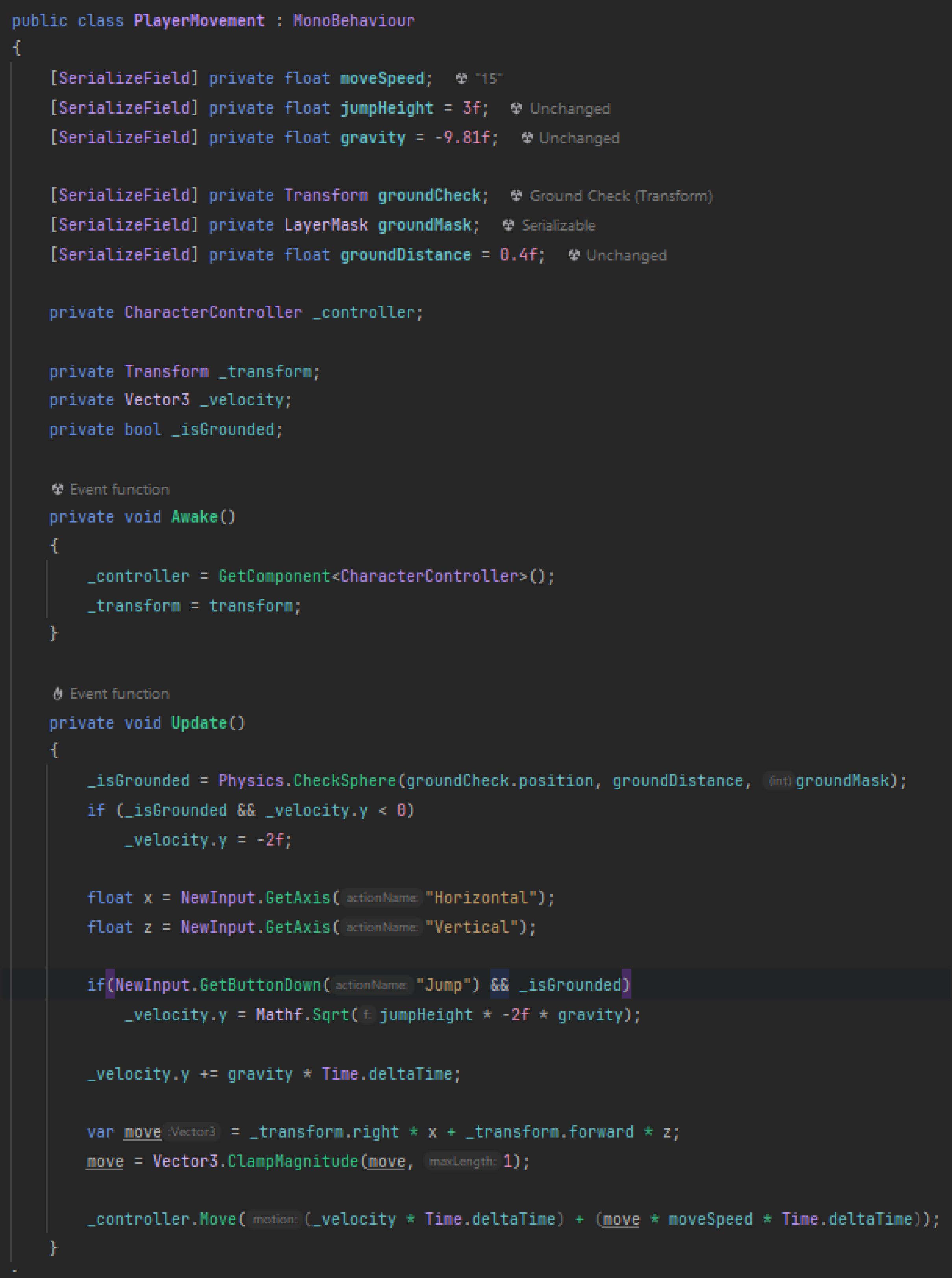
- All the scripts below can be found in the **NewInputByReference/Examples** folder.



```
public class MouseLook : MonoBehaviour
{
    [SerializeField] private float mouseSensitivity = 100f;  "50"
    [SerializeField] private Transform playerBody;  Player (Transform)

    private float _xRotation;

    Event function
    private void Start ()  => Cursor.lockState = CursorLockMode.Locked;

    Event function
    private void Update ()
    {
        float mouseX = NewInput.GetAxis( actionName: "Mouse X") * mouseSensitivity * Time.deltaTime;
        float mouseY = NewInput.GetAxis( actionName: "Mouse Y") * mouseSensitivity * Time.deltaTime;

        _xRotation -= mouseY;
        _xRotation = Mathf.Clamp( value: _xRotation,  min: -90f,  max: 90f);

        transform.localRotation = Quaternion.Euler(_xRotation,  y: 0f,  z: 0f);
        playerBody.Rotate( eulers: Vector3.up * mouseX);
    }
}
```



```
public class PlayerInteraction : MonoBehaviour
{
    [SerializeField] private ButtonInputData inputData;  Unchanged

    [SerializeField] private float rayDistance = 5f;  Unchanged
    [SerializeField] private float rayRadius = 0.3f;  Unchanged
    [SerializeField] private LayerMask interactableLayer;  Serializable

    private Camera _camera;

    Event function
    private void Awake() => _camera = Camera.main;

    Event function
    private void Update()
    {
        var cameraTransform = _camera.transform;
        var ray = new Ray(origin: cameraTransform.position, direction: cameraTransform.forward);

        bool hitSomething = Physics.SphereCast(ray, rayRadius, out var hit, rayDistance, (int) interactableLayer);

        if (!hitSomething)
            return;

        if(inputData.ButtonPressed)
            Destroy(hit.transform.gameObject);
    }
}
```

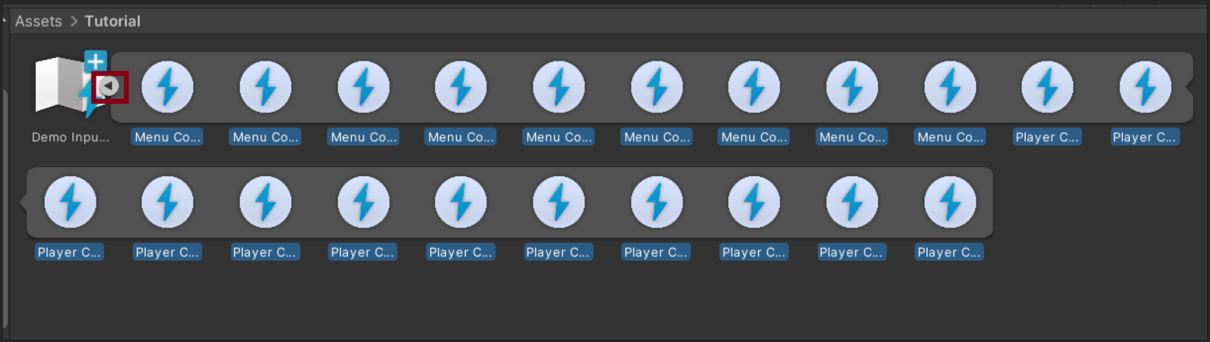


```
public class PlayerMovement : MonoBehaviour
{
    [SerializeField] private float moveSpeed;  "15"
    [SerializeField] private float jumpHeight = 3f;  Unchanged
    [SerializeField] private float gravity = -9.81f;  Unchanged

    [SerializeField] private Transform groundCheck;  Ground Check (Transform)
    [SerializeField] private LayerMask groundMask;  Serializable
    [SerializeField] private float groundDistance = 0.4f;  Unchanged

    private CharacterController _controller;

    private Transform _transform;
    private Vector3 _velocity;
    private bool _isGrounded;

    Event function
    private void Awake()
    {
        _controller = GetComponent<CharacterController>();
        _transform = transform;
    }

    Event function
    private void Update()
    {
        _isGrounded = Physics.CheckSphere(groundCheck.position, groundDistance, (int) groundMask);
        if (_isGrounded && _velocity.y < 0)
            _velocity.y = -2f;

        float x = NewInput.GetAxis( actionName: "Horizontal");
        float z = NewInput.GetAxis( actionName: "Vertical");

        if(NewInput.GetButtonDown( actionName: "Jump") && _isGrounded)
            _velocity.y = Mathf.Sqrt( f: jumpHeight * -2f * gravity);

        _velocity.y += gravity * Time.deltaTime;

        var move :Vector3 = _transform.right * x + _transform.forward * z;
        move = Vector3.ClampMagnitude(move,  maxLength: 1);

        _controller.Move( motion: (_velocity * Time.deltaTime) + (move * moveSpeed * Time.deltaTime));
    }
```

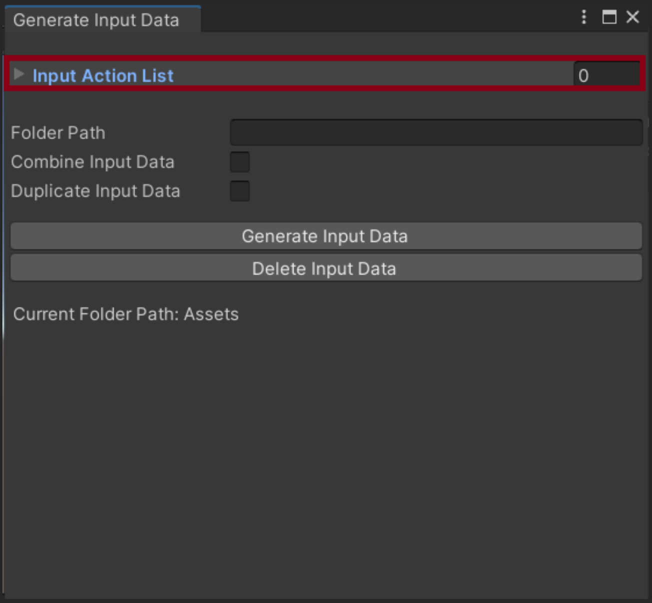
6. What are some tips?

- How can the Input Action List of the Generate Input Data Window be assigned more quickly?

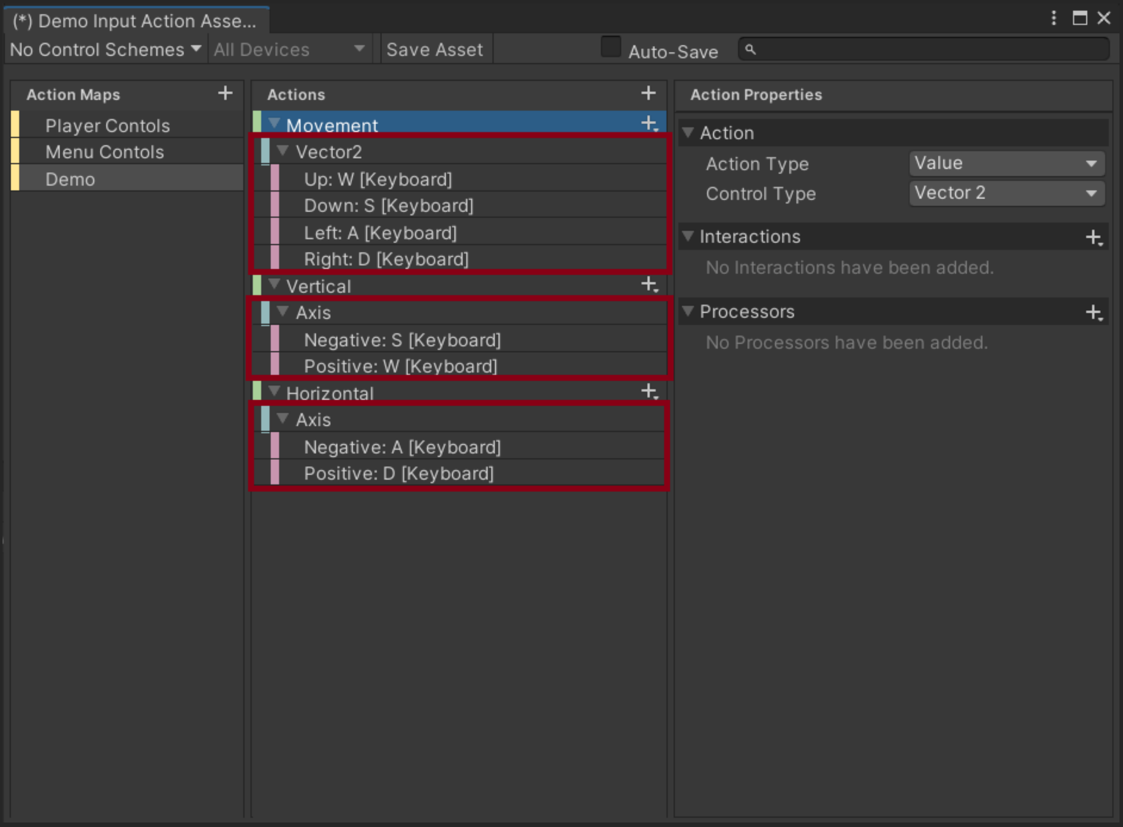
Expand the Input Action Asset and select all the Input Actions (Shift + Click).



Drag them onto the Input Action List from the Generate Input Data Window.



- What is a composite?



(*) Demo Input Action Asse...

No Control Schemes | All Devices | Save Asset | Auto-Save

Action Maps
- Player Contols
- Menu Contols
- Demo

Actions
- Movement
  - Vector2
    - Up: W [Keyboard]
    - Down: S [Keyboard]
    - Left: A [Keyboard]
    - Right: D [Keyboard]
- Vertical
  - Axis
    - Negative: S [Keyboard]
    - Positive: W [Keyboard]
- Horizontal
  - Axis
    - Negative: A [Keyboard]
    - Positive: D [Keyboard]

Action Properties

Action
- Action Type: Value
- Control Type: Vector 2

Interactions

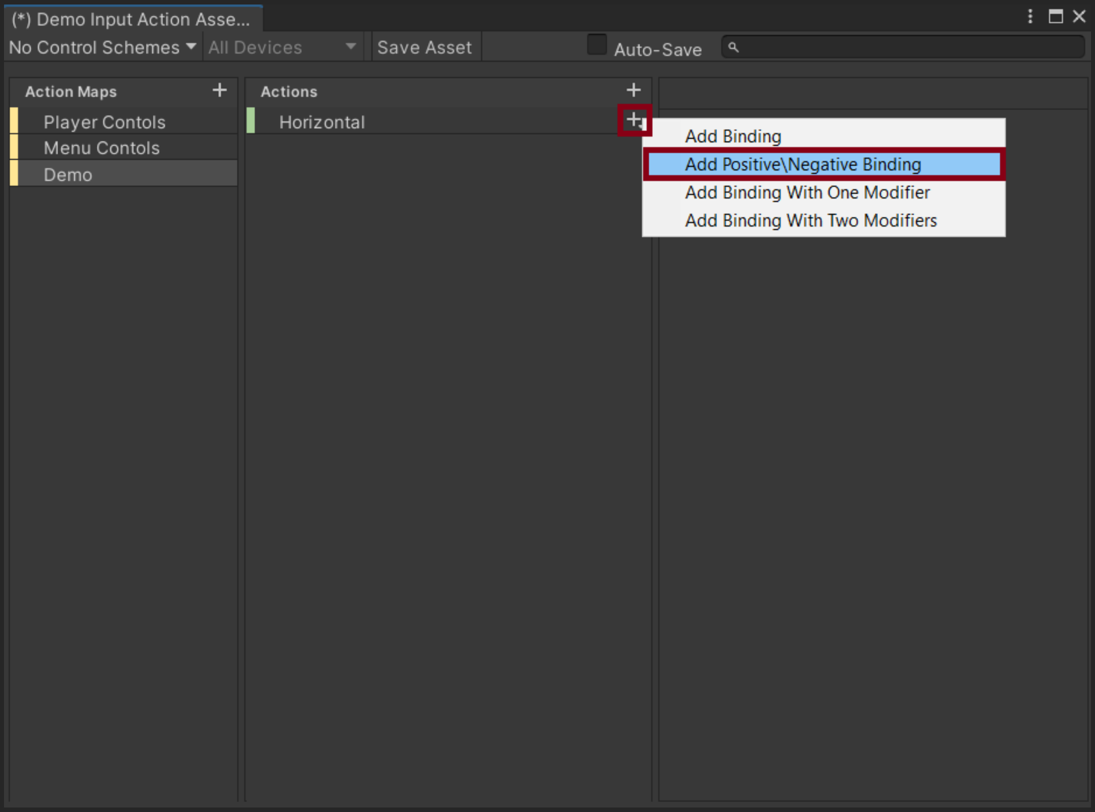
No Interactions have been added.

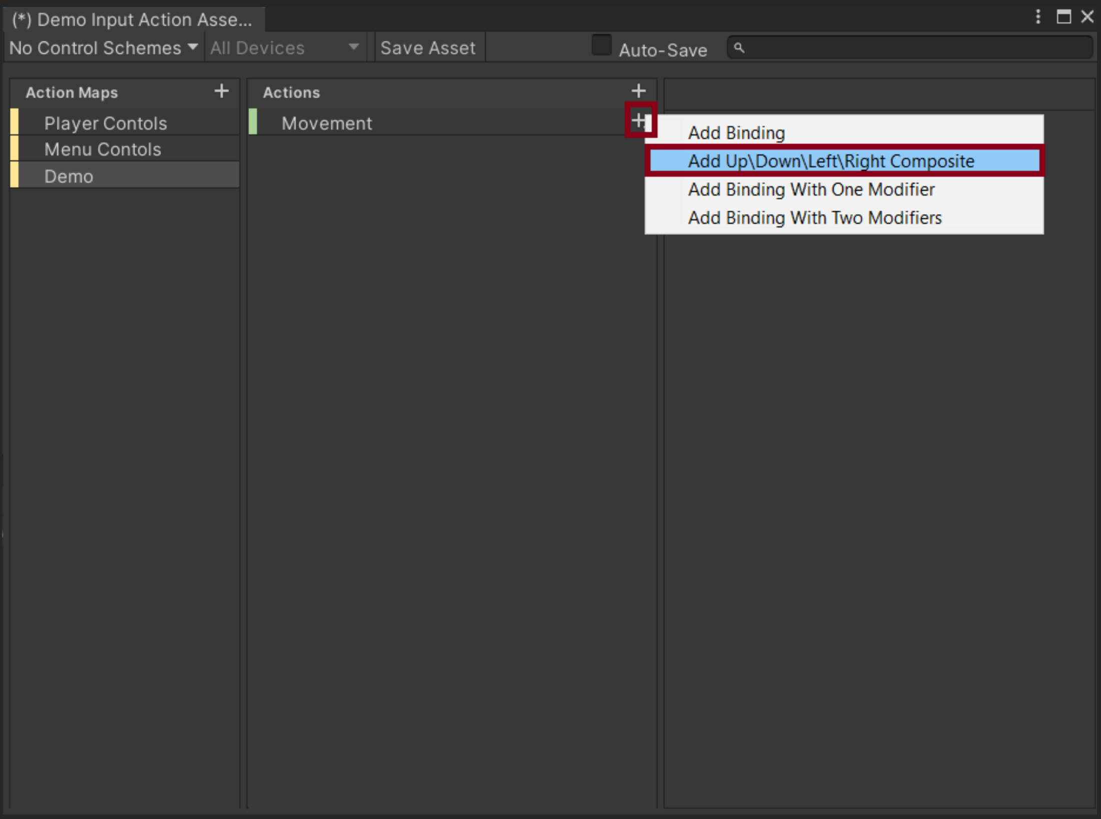
Processors

No Processors have been added.

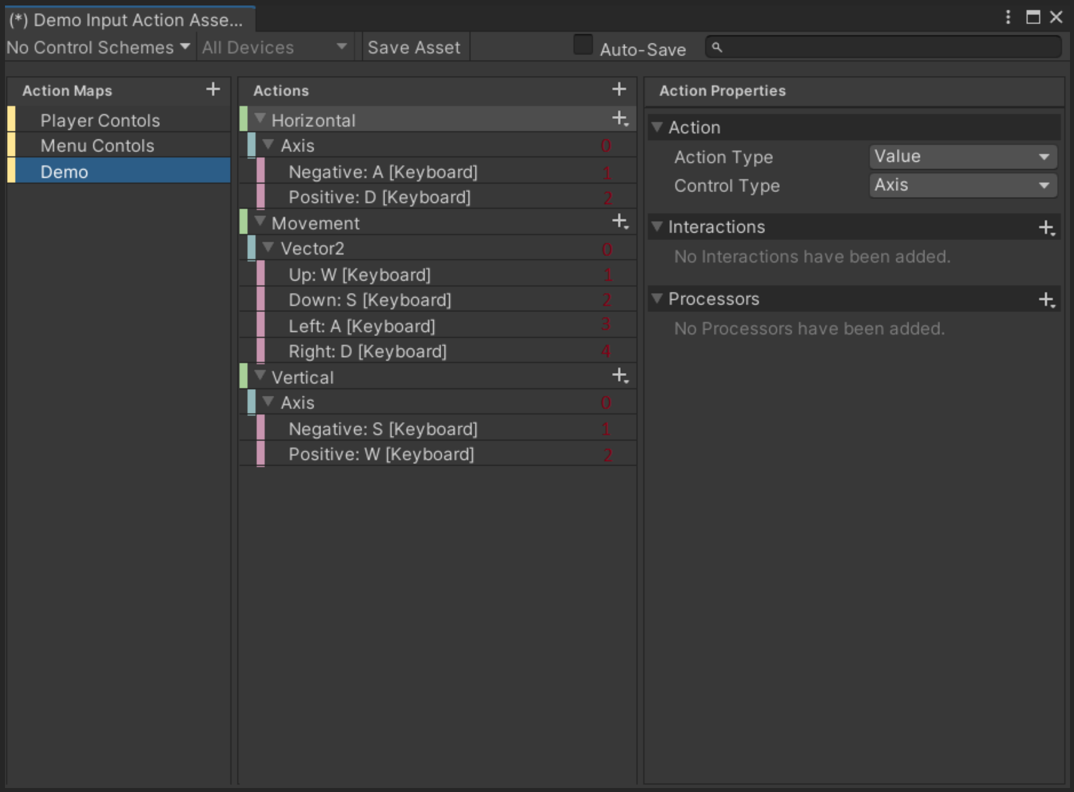
- How to create a composite?

Only available for Action Type = Value and Control Type = Axis/Vector 2/Vector 3.


(*) Demo Input Action Asse...
No Control Schemes
All Devices
Save Asset
Auto-Save
Action Maps
Player Contols
Menu Contols
Demo
Actions
Horizontal
Add Binding
Add Positive\Negative Binding
Add Binding With One Modifier
Add Binding With Two Modifiers


(*) Demo Input Action Asse...
No Control Schemes
All Devices
Save Asset
Auto-Save
Action Maps
Player Contols
Menu Contols
Demo
Actions
Movement
Add Binding
Add Up\Down\Left\Right Composite
Add Binding With One Modifier
Add Binding With Two Modifiers

- How are composites numbered?

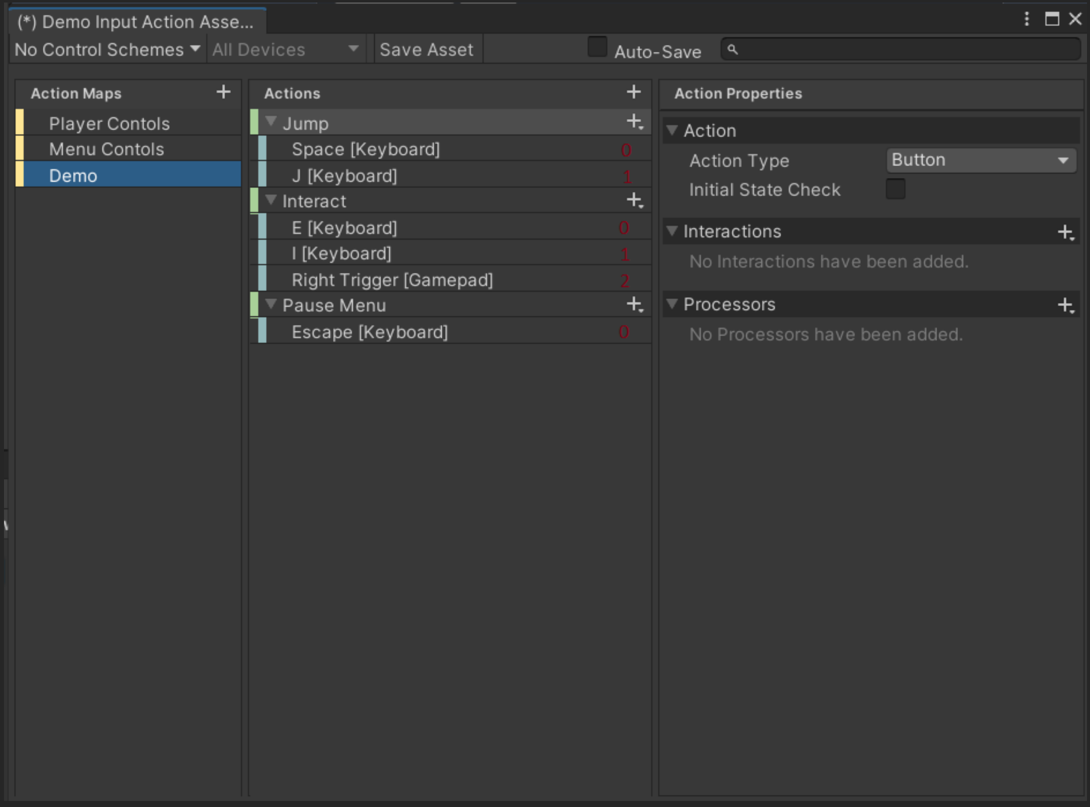

(*) Demo Input Action Asse...
No Control Schemes | All Devices | Save Asset | Auto-Save

Action Maps
Player Contols
Menu Contols
Demo

Actions
Horizontal
Axis 0
Negative: A [Keyboard] 1
Positive: D [Keyboard] 2
Movement
Vector2 0
Up: W [Keyboard] 1
Down: S [Keyboard] 2
Left: A [Keyboard] 3
Right: D [Keyboard] 4
Vertical
Axis 0
Negative: S [Keyboard] 1
Positive: W [Keyboard] 2

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

- How are non-composites numbered?


(*) Demo Input Action Asse...
No Control Schemes | All Devices | Save Asset | Auto-Save

Action Maps
Player Contols
Menu Contols
Demo

Actions
Jump
Space [Keyboard] 0
J [Keyboard] 1
Interact
E [Keyboard] 0
I [Keyboard] 1
Right Trigger [Gamepad] 2
Pause Menu
Escape [Keyboard] 0

Action Properties
Action
Action Type: Button
Initial State Check
Interactions
No Interactions have been added.
Processors
No Processors have been added.