InputManager_AshVP

Input Manager Documentation for `InputManager_AshVP` Script

This documentation provides a step-by-step guide to using and customizing the `InputManager_AshVP` script for vehicle control in Unity. The script integrates Unity's new Input System to handle player input for acceleration, steering, and braking actions.

Script Overview

The `InputManager_AshVP` script allows for input from both a keyboard and gamepad. It assigns input actions for acceleration, steering, and braking, and passes these inputs to a car controller (referenced as `carController`).

The script provides two methods for adding default bindings for both the keyboard and gamepad. The script also handles enabling and disabling of input actions when the game starts or stops.

Key Script Components

Public Variables:

1. `carController`: This references the car controller script, which have a method `ProvideInputs(float acceleration, float steering, float brake)` that accepts these inputs.
2. `accelerateAction`: The input action that handles acceleration input.
3. `steerAction`: The input action that handles steering input.
4. `brakeAction`: The input action that handles braking input.

Private Variables:

1. `accelerationInput`: Stores the current value of the acceleration input (ranging from -1 to 1).
2. `steerInput`: Stores the current value of the steering input (ranging from -1 to 1).
3. `brakeInput`: Stores the current value of the brake input (typically ranges from 0 to 1).

How to Use the Script

1. Attaching the Script:

- Attach the `InputManager_AshVP` script to any GameObject in your scene.
- In the Inspector, assign your car controller script to the **Car Controller** field.

2. Adding Input Bindings:

The script provides two methods for adding default keyboard and gamepad bindings:

- **Add Default Keyboard Bindings**
- **Add Default Gamepad Bindings**

These methods can be triggered using the context menu in the Unity Inspector, or programmatically via code.

To add default bindings:

1. In the Inspector, right-click on the component and select "Add Default Keyboard Bindings" or "Add Default Gamepad Bindings".
2. The script will automatically set up the following controls:

Keyboard Bindings:

- Acceleration: W (forward), S (reverse)
- Steering: D (right), A (left)
- Braking: Space (brake)

Gamepad Bindings:

- Acceleration: Right Trigger (accelerate), Left Trigger (reverse)

- Steering: Left Stick (left/right)
- Braking: South Button (typically A on Xbox, X on PlayStation)

3. Assigning Custom Bindings:

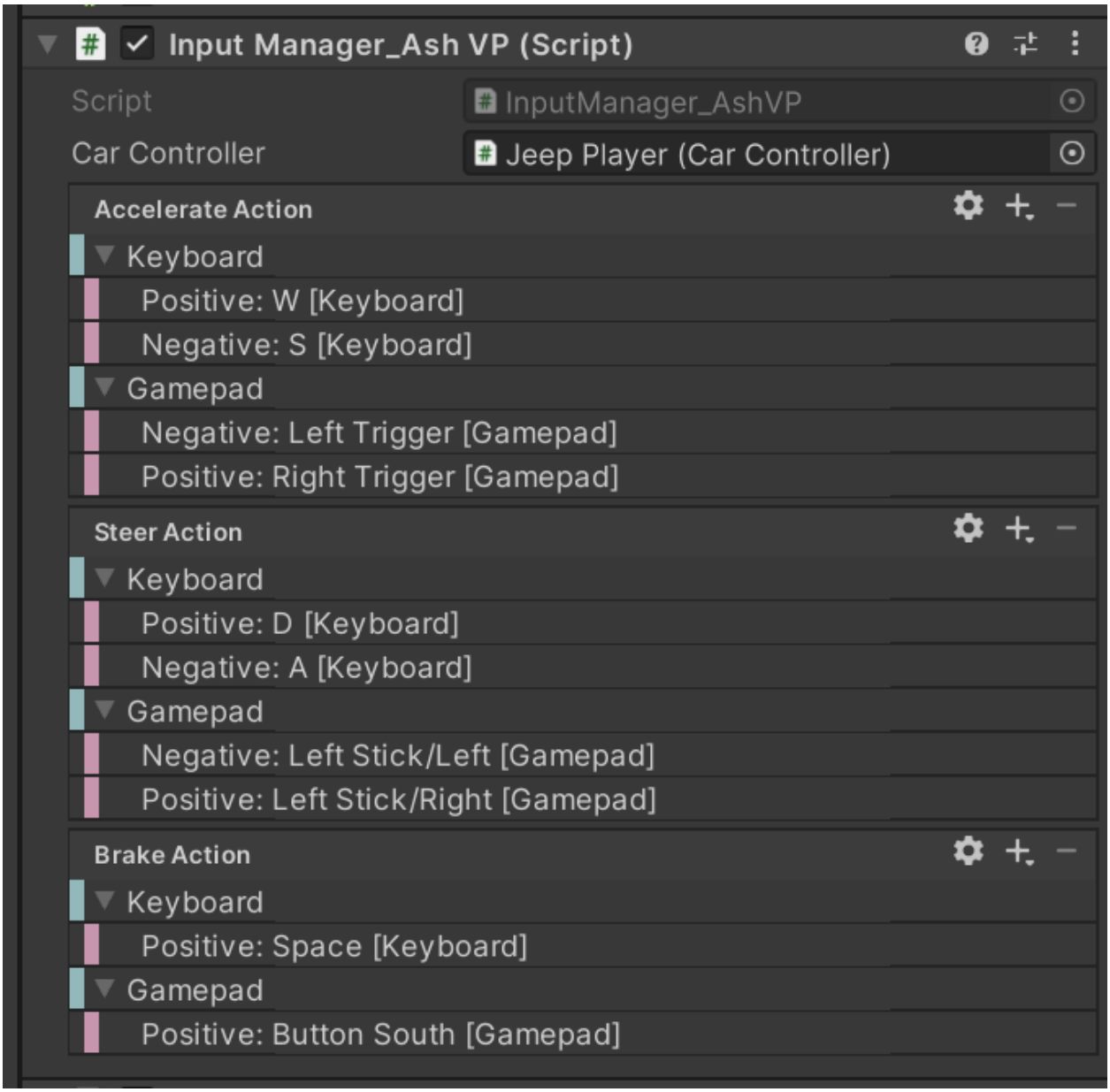
If you wish to assign custom bindings, you can do so in the Unity Input System's Input Actions window, or programmatically via the script using the `AddCompositeBinding()` method, as shown in the example in the script.

Example Workflow

1. **Enable/Disable Input Actions:**
   The script automatically enables the input actions when the GameObject is active and disables them when it's inactive, thanks to the `OnEnable()` and `OnDisable()` methods.
2. **Input Reading:**
   Each frame, the `Update()` method reads the current values of the acceleration, steer, and brake inputs using `ReadValue<float>()`. These values are then passed to the car controller's `ProvideInputs()` method, where the vehicle is controlled based on the input received.

Inspector Setup

The following image shows how the bindings and inputs are set up in the Unity Editor Inspector:



How to Add custom bindings

- to add custom binding instead of default, you can double click on any of the binding(like W[keyboard] as shown in image) for an action (like Accelerate Action as shown in image) and choose any other binding you want.
- to add mobile inputs - you can add “OnScreenButton” script to the ui button and choose which keyboard key it should mimic. for example if you want accelerate forward with a button you can choose W key.