Realistic Car Controller Pro

Thank you for purchasing and using Realistic Car Controller Pro. This documentation will guide you on definition of the input manager and how the inputs work.

Content

Main Input Receiver (RCCP_InputManager)

RCCP_InputManager is the responsible for receiving player inputs from the current device. You won't be able to see it in your scene hierarchy, it's a hidden gameobject. All systems related to the inputs will be using this manager script.

RCCP_Input (attached as component to the vehicle) will read all inputs from the **RCCP_InputManager** and feed the vehicle components with these inputs. **RCCP_Input** doesn't read the player inputs, **RCCP_InputManager** does. **RCCP_Input** only receives inputs from the **RCCP_InputManager** and feeds the vehicle components.

Note: If the vehicle doesn't have the **RCCP_Input** component, it won't receive player inputs from the **RCCP_InputManager**.

Input Types

Realistic Car Controller Pro supports both input types. Old legacy and new input system. **RCCP_InputManager** will read the player inputs with selected input type. You can select the old or new input system in the **RCCP_Settings**. Once you change the option there, scripting symbols will be added / removed to your project to compile scripts.

Note: Default input system is new input system. You can change it from the **RCCP_Settings**.

RCCP_InputManager includes both input systems as well. It will be instantiated on your scene at runtime and all systems related to the input will take this instance as reference. **RCCP_Canvas** includes two event systems for old and new input system. It will enable / disable right one depending on your selection. Old and new input systems are using different event systems for the UI. If you are not planning to use the old input system, you can delete old event system in your UI canvas.

On old input system, only using get key method would work. But on new input system, you'll need to work with events, which is when pressed button, when canceled button, when holding button, etc. You can inspect the **RCCP_InputManager** script to understand how it works. On each player action, corresponding event will be fired and all scripts listening to this event will be informed and noticed.

Note: I would recommend you to use the new input system. It's more reliable, customizable. But not easy to use as old legacy input system. It's based on events. If you can inspect the **RCCP_InputManager** script, you can see all the events based on new input system and old legacy input system.

Old Input System

All key strings can be found in the **RCCP_InputManager** script. Simply open the script, and you'll be able to edit all **GetButton**, **GetKey** buttons here. No need to work with events.

Note: You'll need to import project settings to work with old input manager. Old input manager is using specific inputs. To import the project settings, go to **Tools → BCG → RCCP → Welcome Window** and click "**Import Project Settings**" button.

New Input System

New Input System requires "**Input System**" package installed through the **Package Manager**. Open the **Package Manager**, and search for "**Input System**". Download and import the latest stable version. After that, Unity will ask you to switch to new input system. Choose "**Yes**". This process will override the "**Active Input Handler**" option in your **Player Settings**. Go to **Edit → Project Settings → Player** and check your current active input handler there. If you want to use both systems at the same time, you can choose "**Both**".

RCCP_InputManager is responsible for receiving player inputs via Unity's New Input System. Inputs in this class has been used for controlling the vehicles and the cameras. Inputs in **RCCP_InputManager** and input types (axes/buttons/vectors) have been explained in the table below;

Input Name	Input Type	Button/Axis	Additional Info
Throttle	Axis 0f, 1f	W, Right Trigger	
Brake	Axis 0f, 1f	S, Left Trigger	
Steering	Axis -1f, 1f	A/D, Left Stick Left, Left Stick Right	
Handbrake	Axis 0f, 1f	Space, South Button	
NOS/Boost	Axis 0f, 1f	F, East Button	
Gear Shift Up	Button	Left Shift, Right Trigger	
Gear Shift Down	Button	Left CTRL, Left Trigger	
Low Beam Lights	Button	L, D-Pad Up	
High Beam Lights	Button	K, Left Stick Press	
Indicator Hazard	Button	Z, D-Pad Down	
Indicator Left	Button	Q, D-Pad Left	
Indicator Right	Button	E, D-Pad Right	
Start / Stop Engine	Button	I, North Button	
Trailer Detach	Button	T, Right Stick Press	

Orbit	2D Vector	Mouse Delta X/Mouse Delta Y, Right Stick	
Change Camera	Button	C, Left Stick Press	
Look Back	Button	B, West Button	
Slow Motion	Button	G, null	Not used for gamepads, and mobile
Record	Button	P, null	Not used for gamepads, and mobile
Replay	Button	R, null	Not used for gamepads, and mobile

Supported controller types are;

- **Keyboard & Mouse**
- **Gamepads**
- **Mobile**

RCCP_InputActions as Input Actions for the New Input System

New Input System is using **Input Actions**, which can be customized without any code. Each input can be customized with the scheme. You can access default **Input Actions** of the RCCP from **Resources → RCCP_InputActions**.

RCCP_InputActions have three action maps for vehicles, cameras, and optional. Each action has proper inputs for keyboard & mouse, gamepads. Mobile UI controllers are using independent input system. **RCCP_MobileInputs** is receiving inputs from the UI controller buttons directly, and **RCCP_InputManager** is reading all these inputs through the **RCCP_MobileButtons** attached to the RCCP UI canvas.

How to Add New Inputs, Change Inputs, Remove Inputs for the New Input System

Adding, changing on removing inputs directly from **RCCP_InputActions**, which can be found in **Resources** folder of the RCCP. Double click the **RCCP_InputActions** to open input actions window. There are two controller schemes (keyboard/mouse, and gamepads). You may want to select "all controller schemes" to see all inputs. *Do not change the name of the any action map, or action*. Otherwise, it will generate new C# script with different fields. Reference scripts will not compile, and editor will throw many errors.

Each action has child groups for wide range usage. For example, throttle has three child groups for wasd keys, arrow keys, and gamepad keys. Keys can be changed or can be added here with the new group. To create a new group, click the plus sign near the action name. Select your positive and negative buttons, and you are done! To remove a group, right click it and click delete. To save changes, click "**Save Asset**" button at top of the window. Also, you may want to enable "**Auto Save**".

How the RCCP_InputManager Works?

RCCP_InputManager is receiving player inputs with the **Unity's New Input System**. In old system, inputs were using **Input.GetKey, Input.GetAxis, Input.GetButton** methods. They were many lines for each controller types and hardcoded as well. PS4 controller has different inputs, Xbox controller has different inputs, keyboard has different inputs. Instead of using many hardcoded lines, only one line will do the whole job with the new Input System.

RCCP_InputManager is listening all events on **RCCP_InputActions**. For example, if player pushes start/stop engine, **"StartStopEngine_performed()"** event will be fired. And whatever listens this event, gets notified. **RCCP_CarController** is listening to this event too. When player pushes that button, **"RCC_InputManager_OnStartStopEngine()"** in **RCCP_CarController** will be fired and corresponding function will be played. In this case, engine will stop, or start.

Same things go for axis too. There are positive and negative buttons. When player pushes the positive button, maximum range of the axis will be reached. When player pushes the negative button, minimum range of the axis will be reached. When player doesn't push any button, it will be at center. For example, when player pushes right steering button, axis will be 1f, and -1 for the left steering. 0 will be center.

RCCP_Input and **RCCP_Camera** scripts are listening to events and receiving axis inputs from the **RCCP_InputManager**.

Active Input Handler in Player Settings

There is an option in your **Player Settings** named "**Active Input Handler**". You can set your input type there. Go to **Edit → Project Settings → Player** and check your current active input. If you want to use both systems at the same time, you can choose "**Both**".

Mobile Inputs with RCCP_MobileController

All UI mobile controller buttons have **RCCP_UIController** script attached to them. This script has an axis value of **0 – 1**. When player pushes the button, it will be 1, otherwise 0. All these buttons are managed and observed by **RCCP_MobileController** script attached to the **RCCP_Canvas**. **RCCP_MobileController** is receiving all inputs from the buttons and let the **RCCP_InputManager** reads them. And that's how the mobile controller works with simplest way.

Rebind Inputs at Runtime

There is a new panel in the options menu for rebinding the inputs which you can use it to rebind the inputs at realtime. Each button has RCCP_UI_RebindInput script. When player pushes the button, it will be using this script attached to the gameobject.