Input System for Realistic Car Controller

RCC_InputManager is responsible for receiving player inputs via Unity's Input System. Inputs in this class have been used for controlling the vehicles and the cameras.

Inputs in the **RCC_InputManager** and input types (axes/buttons/vectors) have been explained in the table below;

Input Name	Input Type	Button / Axis	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, Mouse X	
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	Not used for gamepads, and mobile
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

Currently added controller types are

- **Keyboard & Mouse**
- **Gamepads**
- **Mobile (Not using this input system)**
- **Logitech Steering Wheel (Requires SDK and integration package)**
- **Oculus Quest 1 / 2**

RCC_InputActions as Input Actions

Input System is using the 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 RCC from the **Resources → RCC_InputActions**.

RCC_InputActions have three action maps for **vehicles**, **cameras**, and **optional**. Each action map has proper inputs for keyboard & mouse, and gamepads.

Mobile controller is using my own input system instead of the new input manager. Each UI controller button has "**RCC_UIController**" script for inputs. These buttons feeds **RCC_InputManager** with normalized float values. You can adjust UI buttons sensitivity and gravity from the **RCC Settings**. Switching the mobile controller to the new input manager is easy, however I don't recommend doing this. Because UI buttons will simulate gamepad buttons in this case.

If you want to switch mobile controller to the new input system, UI buttons must be simulating the gamepad inputs. Each UI button should have a script named "**OnScreenButton**". Simulated button of the gamepad can be changed from this component. Joystick is using "**OnScreenStick**" script.

How to Add New Inputs, Change Inputs, Remove Inputs

Adding, changing on removing inputs directly from the **RCC_InputActions**, which can be found in the **Resources** folder of the RCC. Double click the **RCC_InputActions** to open the 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 use. 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**" as well.

How the RCC_InputManager Works?

RCC_InputManager is receiving all player inputs with the **Unity's Input System**. In the old system, inputs were using **Input.GetKey, Input.GetAxis, Input.GetButton** methods. They were many lines for each controller type 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 Input System.

RCC_InputManager is listening to all events on the **RCC_InputActions**. For example, if player pushes start/stop engine, **"StartStopEngine_performed()"** event will be fired. And whatever listens to this event, gets notified. **RCC_CarControllerV4** is listening to this event too. When player pushes that button, **"RCC_InputManager_OnStartStopEngine()"** in **RCC_CarControllerV4** will be fired and corresponding function will be played. In this case, the 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, the minimum range of the axis will be reached. When player doesn't push any buttons, 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.

RCC_CarControllerV4 and **RCC_Camera** scripts are listening to these events and receiving axis inputs from the **RCC_InputManager**.