Realistic Car Controller V3.3 by

BoneCracker Games

# RCC.cs

Explained in seperate document.

# RCC\_AIBrakeZone.cs

Brake Zones are meant to be used for slowing AI vehicles. If you have a sharp turn on your scene, you can simply use one of these Brake Zones. It has a target speed. AI will adapt it’s speed to this target speed while in this Brake Zone. It's simple.

# RCC\_AIBrakeZonesContainer.cs

Used for holding a list for brake zones, and drawing gizmos for all of them on Editor.

# RCC\_AICarController.cs

AI Controller of RCC. It's not professional, but it does the job. Follows all waypoints, or chases the player. Must be attached to root of the vehicle. RCC\_CarControllerV3.cs will not receive any inputs from player when this script attached to the vehicle.

# RCC\_AIWaypointsContainer.cs

Used for holding a list for waypoints, and drawing gizmos for all of them.

**RCC\_APIExample.cs**

An example script to shows how the RCC API works. Uses the RCC.cs.

# RCC\_Caliper.cs

Rotates the caliper.

# RCC\_Camera.cs

Main RCC Camera controller. Includes 7 different camera modes with many customizable settings. It doesn't use different cameras on your scene like \*other\* assets. Simply it parents the camera to their positions that's all. Also supports collision detection.

# RCC\_CarControllerV3.cs

Main vehicle controller that includes Wheels, Steering, Suspensions, Mechanic Configuration, Stability, Lights, Sounds, and Damage. All In One script.

# RCC\_ChangableWheels.cs

Changes wheels (visual only) at runtime. It holds changable wheels as prefab in an array.

# RCC\_CharacterController.cs

Animates Driver Sofie (Credits to 3DMaesen). Simply feeds floats and bools of Sofie's animator component.

# RCC\_Chassis.cs

Simulates chassis movement based on vehicle rigidbody velocity.

# RCC\_CinematicCamera.cs

Tracks the car and keeps orientation nicely for cinematic angles. It has a pivot gameobject named "Animation Pivot". This gameobject has 3 animations itself currently.

# RCC\_CreateAudioSource.cs

Creates new audiosources at runtime with specified settings.

# RCC\_Customization.cs

Main Customization Class For RCC.

# RCC\_CustomizerExample.cs

A simple customizer example script used for receiving methods from UI elements and send them to RCC\_Customization script. Also updates all UI elements for new spawned vehicles too.

# RCC\_DashboardColors.cs

Changes HUD image colors by UI Sliders.

# RCC\_DashboardInputs.cs

Receiving inputs from active vehicle on your scene, and feeds dashboard needles, texts, images.

# RCC\_DashboardObjects.cs

Receiving inputs from active vehicle on your scene, and feeds visual dashboard needles.

# RCC\_Demo.cs

A simple manager script for all demo scenes. It has an array of spawnable player vehicles, public methods, setting new behavior modes, restart, and quit application.

# RCC\_Exhaust.cs

Exhaust based on Particle System. Based on vehicle engine RPM.

# RCC\_FixedCamera.cs

Fixed camera system for RCC Camera. It simply parents the RCC Camera, and calculates target position, rotation, FOV, etc...

# RCC\_FOVForCinematicCamera.cs

Animation attached to "Animation Pivot" of the Cinematic Camera is feeding FOV float value.

# RCC\_GetBounds.cs

Gets total bound size of a gameobject.

# RCC\_GroundMaterials.cs

Configurable Ground Materials are collected in an array of class.

# RCC\_HoodCamera.cs

RCC Camera will be parented to this gameobject when current camera mode is Hood Camera.

# RCC\_Light.cs

General lighting system for vehicles. It has all kind of lights such as Headlight, Brake Light, Indicator Light, Reverse Light.

# RCC\_LightEmission.cs

Feeding material's emission channel for self illumin effect.

# RCC\_Mirror.cs

It must be attached to external camera. This external camera will be used as mirror.

# RCC\_MobileButtons.cs

Receiving inputs from UI buttons, and feeds active vehicles on your scene.

# RCC\_PoliceSiren.cs

Flashes red and blue lights with proper timing. If vehicle is an AI vehicle and chaser vehicle, toggles lights on / off automatically.

# RCC\_Recorder.cs

Record / Replay system. Saves player's input on record, and replays it when on playback.

# RCC\_SceneManager.cs

Scene manager that contains current player vehicle, current player camera, current player UI, current player character, recording/playing mechanim, and other vehicles as well.

# RCC\_Settings.cs

Stored all general shared RCC settings here.

# RCC\_ShadowRotConst.cs

Locks rotation of the shadow projector to avoid stretching.

# RCC\_Skidmarks.cs

Skidmarks Manager for RCC.

# RCC\_SuspensionArm.cs

Rotates and moves suspension arms based on wheelcollider suspension distance.

# RCC\_TruckTrailer.cs

Truck trailer has additional wheelcolliders. This script handles center of mass of the trailer, wheelcolliders, and antiroll.

# RCC\_UIController.cs

UI input (float) receiver from UI Button.

# RCC\_UIDashboardButton.cs

UI buttons used in options panel. It has an enum for all kind of buttons.

# RCC\_UIDashboardDisplay.cs

Handles dashboard elements.

# RCC\_UIDrag.cs

Mobile UI Drag used for orbiting RCC Camera.

# RCC\_UISliderTextReader.cs

Receives float from UI Slider, and displays the value as a text.

# RCC\_UISteeringWheelController.cs

UI Steering Wheel controller.

**RCC\_UnetNetwork.cs**

Streaming player input, or receiving data from server. And then feeds the RCC.

# RCC\_WheelCamera.cs

RCC Camera will be parented to this gameobject when current camera mode is Wheel Camera.

# RCC\_WheelCollider.cs

Based on Unity's WheelCollider. Modifies few curves, settings in order to get stable and realistic physics depends on selected behavior in RCC Settings.

**Photon Scripts**

# RCC\_PhotonDemo.cs

A simple manager script for photon demo scene. It has an array of networked spawnable player vehicles, public methods, restart, and quit application.

# RCC\_PhotonManager.cs

Connects to Photon Server, registers the player, and activates player UI panel when connected.

# RCC\_PhotonNetwork.cs

Syncs the player. Streams player input, or receiving data from server. And then feeds the RCC.