RCC_CarControllerV3.cs

Public Variables

public bool canControl = true; // Enables / Disables controlling the vehicle. If enabled, vehicle can receive all inputs from the InputManager.

public bool isGrounded = false; // Is vehicle grounded completely now?

public bool overrideBehavior = false; // Vehicle won't be affected by selected behavior in RCC Settings if override is selected.

// Wheel models of the vehicle.
public Transform FrontLeftWheelTransform;
public Transform FrontRightWheelTransform;
public Transform RearLeftWheelTransform;
public Transform RearRightWheelTransform;
public Transform[] ExtraRearWheelsTransform; // Extra wheels in case your vehicle has extra wheels.

// Wheel colliders of the vehicle. They will be generated automatically inside the editor when clicked “Generate WheelColliders” button.
public RCC_WheelCollider FrontLeftWheelCollider;
public RCC_WheelCollider FrontRightWheelCollider;
public RCC_WheelCollider RearLeftWheelCollider;
public RCC_WheelCollider RearRightWheelCollider;
public RCC_WheelCollider[] ExtraRearWheelsCollider; // Extra Wheels. In case of if your vehicle has extra wheels.

// All wheel colliders.
public RCC_WheelCollider[] AllWheelColliders

// All lights.
public RCC_Light[] AllLights

public bool hasExtraWheels = false; // Vehicle has extra wheels?

public bool overrideAllWheels = false; // Overriding individual wheel settings such as steer, power, brake, handbrake.

public int poweredWheels = 0; // Total count of powered wheels. Used for dividing total power per each wheel.

public Transform SteeringWheel; // Driver steering wheel model. In case of if your vehicle has individual steering wheel model in interior.

public SteeringWheelRotateAround steeringWheelRotateAround; // Current rotation of steering wheel.

```
public float steeringWheelAngleMultiplier = 11f;            // Angle multiplier of steering wheel.

// Drivetrain type of the vehicle.
public WheelType wheelTypeChoise = WheelType.RWD;

public bool externalController = false;     // AI Controller. External inputs can feed the vehicle.

public SteeringType steeringType;           // Steering type.

public AnimationCurve steerAngleCurve = new AnimationCurve();           // Steering angle limiter curve based on speed.

public float steerAngle = 40f;      // Maximum Steer Angle Of Your Vehicle.

public float highspeedsteerAngle = 5f;    // Maximum Steer Angle At Highest Speed.

public float highspeedsteerAngleAtspeed = 120f;            // Highest Speed For Maximum Steer Angle.

public float antiRollFrontHorizontal = 1000f;       // Anti Roll Horizontal Force For Preventing Flip Overs And Stability.

public float antiRollRearHorizontal = 1000f;        // Anti Roll Horizontal Force For Preventing Flip Overs And Stability.

public float antiRollVertical = 0f;             // Anti Roll Vertical Force For Preventing Flip Overs And Stability. I know it doesn't exist, but it can improve gameplay if you have high COM vehicles like monster trucks.



//  Rigidbody.
public Rigidbody Rigid

public Transform COM;                                   // Center of mass.

public float brakeTorque = 2000f;                       // Maximum brake torque.,

public float downForce = 25f;                           // Applies downforce related with vehicle speed.

public float speed = 0f;                                // Vehicle speed in km/h or mp/h.

public float maxspeed = 240f;                           // Top speed.

public AnimationCurve engineTorqueCurve = new AnimationCurve();     //          Engine torque curve based on RPM.

public bool autoGenerateEngineRPMCurve = true;      // Auto create engine torque curve. If min/max engine rpm, engine torque, max engine torque at rpm, or top speed has been changed at runtime, it will generate new curve with them.

public float maxEngineTorque = 300f;                    // Maximum engine torque at target RPM.
```

```
public float maxEngineTorqueAtRPM = 5500f;        //      Maximum peek of the engine at this RPM.

public float minEngineRPM = 800f;                    // Minimum engine RPM.

public float maxEngineRPM = 7000f;                    // Maximum engine RPM.

public float engineRPM = 0f;                          // Current engine RPM.

public float engineRPMRaw = 0f;                      // Current raw engine RPM.

public float engineInertia = .15f;      // Engine inertia. Engine reacts faster on lower values.

public bool useRevLimiter = true;                     // Rev limiter above maximum engine RPM. Cuts
gas when RPM exceeds maximum engine RPM.

public bool useExhaustFlame = true;                   // Exhaust blows flame when driver cuts gas at
certain RPMs.

public bool engineRunning = false;                                                  // Engine running now?

public bool useSteeringLimiter = true;                // Limits maximum steering angle when
vehicle is sliding. It helps to keep the vehicle in control.

public bool useCounterSteering = true;                // Applies counter steering when vehicle is
drifting. It helps to keep the vehicle in control.

public bool useSteeringSensitivity = true;            //          Steering sensitivity.

public float counterSteeringFactor = .5f;             // Counter steering multiplier.
public float steeringSensitivityFactor = 1f;     // Steering sensitivity multiplier.

public float oldSteeringInput = 0f;    //    Old steering input.

public float steeringDifference = 0f;   //   Steering input difference.

public bool useFuelConsumption = false;    // Enable / Disable Fuel Consumption.

public float fuelTankCapacity = 62f;             // Fuel Tank Capacity.

public float fuelTank = 62f;                     // Fuel Amount.

public float fuelConsumptionRate = .1f;        // Fuel Consumption Rate.

public bool useEngineHeat = false;                    // Enable / Disable engine heat.

public float engineHeat = 15f;                        // Engine heat.

public float engineCoolingWaterThreshold = 90f;    // Engine cooling water engage point.

public float engineHeatRate = 1f;                     // Engine heat multiplier.
```

```
public float engineCoolRate = 1f;                    // Engine cool multiplier.

public Gear[] gears = null;              // Gear class.

public int totalGears = 6;        //Total count of gears.

public int currentGear = 0;    // Current gear of the vehicle.

public bool NGear = false;        // N gear.

public float finalRatio = 3.23f;                              //        Final drive gear ratio.

public float gearShiftingDelay = .35f;          //    Gear shifting delay with time.

public float gearShiftingThreshold = .75f;    //    Shifting gears at lower RPMs at higher values.

public float clutchInertia = .25f;              //    Adjusting clutch faster at lower values. Higher
values for smooth clutch.

public float gearShiftUpRPM = 6500f;           // Shifting up when engine RPM is high enough.

public float gearShiftDownRPM = 3500f;    //    Shifting down when engine RPM is low enough.

public bool changingGear = false;              // Changing gear currently?

public int direction = 1;                    // Reverse gear currently?


// How many audio sources we will use for simulating engine sounds?. Usually, all modern driving
games have around six audio sources per vehicle.
// Low RPM, Medium RPM, and High RPM. And their off versions.
public AudioType audioType;

// If you don't have their off versions, generate them.
public bool autoCreateEngineOffSounds = true;

// AudioSources and AudioClips.
public AudioClip engineStartClip;
public AudioClip engineClipHigh;
public AudioClip engineClipMed;
public AudioClip engineClipLow;
public AudioClip engineClipIdle;

public AudioClip engineClipHighOff;
public AudioClip engineClipMedOff;
public AudioClip engineClipLowOff;

// Min / Max sound pitches and volumes.
public float minEngineSoundPitch = .75f;
public float maxEngineSoundPitch = 1.75f;
```

```
public float minEngineSoundVolume = .05f;
public float maxEngineSoundVolume = .85f;
public float idleEngineSoundVolume = .85f;

// Positions of the created audio sources.
public Vector3 engineSoundPosition = new Vector3(0f, 0f, 1.5f);
public Vector3 gearSoundPosition = new Vector3(0f, -.5f, .5f);
public Vector3 turboSoundPosition = new Vector3(0f, 0f, 1.5f);
public Vector3 exhaustSoundPosition = new Vector3(0f, -.5f, -2f);
public Vector3 windSoundPosition = new Vector3(0f, 0f, 2f);

// Inputs. All values are clamped 0f - 1f. They will receive proper input values from
RCC_InputManager class.
public RCC_Inputs inputs;
public float throttleInput = 0f;
public float brakeInput = 0f;
public float steerInput = 0f;
public float counterSteerInput = 0f;
public float clutchInput = 0f;
public float handbrakeInput = 0f;
public float boostInput = 0f;
public float fuelInput = 0f;
public bool cutGas = false;
public bool permanentGas = false;

public bool lowBeamHeadLightsOn = false;    // Low beam head lights.

public bool highBeamHeadLightsOn = false;   // High beam head lights.

public IndicatorsOn indicatorsOn;                   // Indicator system.

public float indicatorTimer = 0f;                   // Used timer for indicator on / off sequence.

public RCC_Damage damage;     //          Damage class.

public bool useDamage = true;      // Use deformation on collisions.

public bool useCollisionParticles = true;       //     Use particles on coliisions.

public bool useCollisionAudio = true;       //          Play crash audio clips on collisions

public int maximumContactSparkle = 5;               //          Contact Particles will be ready to use for
collisions in pool.




// Driving Assistances.
public bool ABS = true;
public bool TCS = true;
```

```
public bool ESP = true;
public bool steeringHelper = true;
public bool tractionHelper = true;
public bool angularDragHelper = false;

// Driving Assistance thresholds.
public float ABSThreshold = .35f;       //  ABS will be engaged at this threshold.
public float TCSStrength = .5f;
public float ESPThreshold = .5f;        //  ESP will be engaged at this threshold.
public float ESPStrength = .25f;
public float steerHelperLinearVelStrength = .1f;
public float steerHelperAngularVelStrength = .1f;
public float tractionHelperStrength = .1f;
public float angularDragHelperStrength = .1f;

// Is Driving Assistance is in action now?
public bool ABSAct = false;
public bool TCSAct = false;
public bool ESPAct = false;

// ESP malfunction.
public bool ESPBroken = false;

// Used For ESP.
public float frontSlip = 0f;
public float rearSlip = 0f;

// ESP Bools.
public bool underSteering = false;
public bool overSteering = false;
#endregion

// Drift Variables.
internal bool driftingNow = false;      // Currently drifting?
internal float driftAngle = 0f;            // If we do, what's the drift angle?

// Turbo and NOS.
public float turboBoost = 0f;
public float NoS = 100f;
private float NoSConsumption = 25f;
private float NoSRegenerateTime = 10f;

public bool useNOS = false;
public bool useTurbo = false;

public RCC_TruckTrailer attachedTrailer;
```

Public Methods

public void Repair() {} //Repairs the vehicle.

public void SetEngine(bool state) {} //Sets the engine state.

public void SetExternalControl(bool state) {} //Sets “externalController” bool of the vehicle.

public void SetCanControl(bool state) {} //Sets controllable state of the vehicle. Vehicle will receive inputs if it’s enabled.

public void DetachTrailer() {} //Detaches attached trailer.

public void GearShiftDown() {} //Shifts gear down.

public void GearShiftTo(int gear) {} //Shifts gear to target gear.

public void GearShiftUp() {} //Shifts gear up.

public void KillEngine() {} //Kills the engine.

public void StartEngine(bool instantStart) {} //Starts the engine.

public void StartEngine() {} //Starts the engine.

public void KillOrStartEngine() {} //Kills or starts the engine.

public void InitGears() {} //Initializes gears. Useful after changing total gear count.

public void ReCreateEngineTorqueCurve() {} //Recreates the engine torque curve.

Events

```
/// On RCC player vehicle spawned.
public delegate void onRCCPlayerSpawned(RCC_CarControllerV3 RCC);
public static event onRCCPlayerSpawned OnRCCPlayerSpawned;

/// On RCC player vehicle destroyed.
public delegate void onRCCPlayerDestroyed(RCC_CarControllerV3 RCC);
public static event onRCCPlayerDestroyed OnRCCPlayerDestroyed;

/// On RCC player vehicle collision.
public delegate void onRCCPlayerCollision(RCC_CarControllerV3 RCC, Collision collision);
public static event onRCCPlayerCollision OnRCCPlayerCollision;
```