Table of Contents

1. Overview

`RCC_Settings` is a `ScriptableObject` that holds all general, shared configuration values for **Realistic Car Controller** (RCC). It centralizes parameters for:

- Vehicle behavior (steering, traction, ABS, TCS, etc.)
- Fixed updates and frame rate overrides
- Layers and physics material references
- UI and mobile control setups
- Audio mixing and sound effects
- Particle and skidmark toggles

Because it inherits from `ScriptableObject`, this configuration asset can be edited in the Unity Editor and is typically stored in a **Resources** folder for easy access at runtime.

2. Class Declaration

```
[System.Serializable]
public class RCC_Settings : ScriptableObject {
   // Implementation details...
}
```

The class is marked `[System.Serializable]` for proper serialization in Unity’s editor and extends `ScriptableObject`, meaning instances of `RCC_Settings` can be created and customized via the Project window.

---

3. Singleton Implementation

```
private static RCC_Settings instance;
public static RCC_Settings Instance {
    get {
        if (instance == null)
            instance = Resources.Load("RCC Assets/RCC_Settings") as RCC_Settings;
        return instance;
    }
}
```

- Ensures there is always a single accessible instance of `RCC_Settings`.
- Looks for a `ScriptableObject` named `RCC_Settings` inside the **Resources/RCC Assets/** directory.
- Use `RCC_Settings.Instance` anywhere in your code to access this global configuration.

> **Note**: The string path `"RCC Assets/RCC_Settings"` must match the actual file location under `Assets/Resources`.

---

4. Fields and Properties

Below is a breakdown of the primary fields available in `RCC_Settings`. These fields control overall RCC behavior, physics, UI, and audio settings.

4.1 Behavior Selection and Overrides

- **`public int behaviorSelectedIndex = 0;`**
  Index of the currently selected driving behavior profile.

- **`public BehaviorType selectedBehaviorType { get; }`**
  Returns the `BehaviorType` at `behaviorSelectedIndex`. If `overrideBehavior` is false, this property returns `null` instead of a valid behavior.

- **`public bool overrideBehavior = true;`**
  When true, forces RCC to use the `behaviorSelectedIndex` for setting behaviors.

If false, default or external methods might be used to control the behavior.

- `public BehaviorType[] behaviorTypes;`
  An array of possible behavior definitions. Each element describes a separate tuning profile for steering, traction, etc.

4.2 `BehaviorType` Nested Class

```
[System.Serializable]
public class BehaviorType {
    public string behaviorName = "New Behavior";
    // Steering helpers
    public bool steeringHelper = true;
    public bool tractionHelper = true;
    public bool angularDragHelper = false;
    public bool counterSteering = true;
    public bool limitSteering = true;
    public bool steeringSensitivity = true;
    public bool ABS = false;
    public bool ESP = false;
    public bool TCS = false;
    public bool applyExternalWheelFrictions = false;
    public bool applyRelativeTorque = false;

    public RCC_CarControllerV4.SteeringType steeringType =
RCC_CarControllerV4.SteeringType.Curve;
    public AnimationCurve steeringCurve = new AnimationCurve(...);

    public RCC_CarControllerV4.COMAssisterTypes comAssister =
RCC_CarControllerV4.COMAssisterTypes.Off;

    // Steering angle limits
    public float highSpeedSteerAngleMinimum = 20f;
    public float highSpeedSteerAngleMaximum = 40f;
    public float highSpeedSteerAngleAtspeedMinimum = 100f;
    public float highSpeedSteerAngleAtspeedMaximum = 200f;

    // Counter steering
    public float counterSteeringMinimum = .1f;
    public float counterSteeringMaximum = 1f;

    // Steering sensitivity
    public float steeringSensitivityMinimum = .5f;
    public float steeringSensitivityMaximum = 1f;

    // Steering helper strengths
    [Range(0f, 1f)] public float steerHelperAngularVelStrengthMinimum = .1f;
    [Range(0f, 1f)] public float steerHelperAngularVelStrengthMaximum = 1;
    [Range(0f, 1f)] public float steerHelperLinearVelStrengthMinimum = .1f;
    [Range(0f, 1f)] public float steerHelperLinearVelStrengthMaximum = 1f;

    // Traction helper strength
    [Range(0f, 1f)] public float tractionHelperStrengthMinimum = .1f;
    [Range(0f, 1f)] public float tractionHelperStrengthMaximum = 1f;

    // Anti-roll
    public float antiRollFrontHorizontalMinimum = 1000f;
    public float antiRollRearHorizontalMinimum = 1000f;

    // Gear shifting delay
    [Range(0f, 1f)] public float gearShiftingDelayMaximum = .15f;

    // Angular drag
    [Range(0f, 10f)] public float angularDrag = .1f;
    [Range(0f, 1f)] public float angularDragHelperMinimum = .1f;
    [Range(0f, 1f)] public float angularDragHelperMaximum = 1f;

    // Wheel frictions
    public float forwardExtremumSlip = .4f;
    public float forwardExtremumValue = 1f;
    public float forwardAsymptoteSlip = .8f;
    public float forwardAsymptoteValue = .5f;
    public float sidewaysExtremumSlip = .2f;
    public float sidewaysExtremumValue = 1f;
    public float sidewaysAsymptoteSlip = .5f;
    public float sidewaysAsymptoteValue = .75f;
}
```

Key points:

- **Steering and Traction Helpers**: Toggles for advanced driving aids like traction control (TCS), stability control (ESP), and advanced friction control.
- **Steering Curves**: Defines how steering angles scale with speed or input, enabling more realistic turning behavior at high speed.
- **Wheel Friction**: Specifies slip and friction values for forward/backward and lateral movement.

Each `BehaviorType` can be seen as a *profile* or *preset* for different vehicle handling experiences (e.g., arcade vs. realistic).

4.3 Time and Performance Settings

- **`public bool overrideFPS = true;`**
  If true, RCC will set the target frame rate to `maxFPS`.

- **`public bool overrideFixedTimeStep = true;`**
  If true, RCC will override Unity's default `Time.fixedDeltaTime` with `fixedTimeStep`.

- **`[Range(.005f, .06f)] public float fixedTimeStep = .02f;`**
  The fixed timestep used if `overrideFixedTimeStep` is enabled.

- **`public int maxFPS = 60;`**
  The target frames-per-second when `overrideFPS` is enabled.

- **`[Range(.5f, 20f)] public float maxAngularVelocity = 6;`**
  Maximum angular velocity for the physics engine—commonly set for stable wheel rotations.

- **`public bool useFixedWheelColliders = true;`**
  Option to use a heavier mass for wheel colliders, potentially improving stability at high speeds.

4.4 Layer and Physics Settings

- **`public bool setLayers = true;`**
  If true, RCC automatically organizes the vehicle's game objects under specific layers for collision filtering.

- **`public string RCCLayer = "RCC_Vehicle";`**
  Layer name used for RCC vehicles.

- **`public string WheelColliderLayer = "RCC_WheelCollider";`**
  Layer name for wheel collider objects.

- **`public string DetachablePartLayer = "RCC_DetachablePart";`**
  Layer name for any detachable/physically breakable parts.

- **`public string PropLayer = "RCC_Prop";`**
  Layer name used for environment props or objects that may interact with RCC vehicles.

- `public PhysicMaterial colliderMaterial;`
  Reference to the default physics material for vehicle bodies.

- `public bool autoReset = true;`
  Allows automatic resetting of the vehicle if it flips upside down (internal logic handled by RCC scripts).

4.5 UI and Mobile Settings

- `public UIType uiType = UIType.UI;`
  Specifies the UI system to use (Unity's default UI, NGUI, or None).

- `public enum UIType { UI, NGUI, None }`

- `public bool useShortcuts = false;`
  Toggles in-editor shortcuts (Shift+E, Shift+R, Shift+S, etc.).

- `public bool lockAndUnlockCursor = true;`
  Locks the mouse cursor when the application is focused, unlocking it otherwise.

- `public Units units = Units.KMH;`
  Determines the speed unit (KMH vs. MPH).

- `public enum Units { KMH, MPH }`

- `public bool mobileControllerEnabled = false;`
  Whether mobile controls are active. If true, RCC uses UI input buttons or gyro-based steering for vehicles.

- `public enum MobileController { TouchScreen, Gyro, SteeringWheel, Joystick }`
  Defines the type of mobile input method: on-screen buttons, gyroscope, virtual steering wheel, or joystick.

- `public float UIButtonSensitivity = 10f;`

- `public float UIButtonGravity = 10f;`

- `public float gyroSensitivity = 2f;`
  Various scaling factors for input responsiveness on mobile devices.

4.6 Lighting and Visuals

- `public bool useHeadLightsAsVertexLights = false;`
  When true, headlights are handled as vertex lights for performance reasons.
- `public bool useBrakeLightsAsVertexLights = true;`
- `public bool useReverseLightsAsVertexLights = true;`
- `public bool useIndicatorLightsAsVertexLights = true;`
- `public bool useOtherLightsAsVertexLights = true;`
  Similar toggles for brake, reverse, indicator, and additional lights.
- `public Object lensflareURP;`
  Potential URP (Universal Render Pipeline) lens flare effect reference.
- `public GameObject headLights;`
- `public GameObject brakeLights;`
- `public GameObject reverseLights;`
- `public GameObject indicatorLights;`
- `public GameObject interiorLights;`
- `public GameObject lightTrailers;`
- `public GameObject mirrors;`
  Prefab references for various light or mirror game objects that can be attached to vehicles.

4.7 Audio and Sound FX

- `public AudioMixerGroup audioMixer;`
  Unity AudioMixer for grouping and mixing RCC-related sounds.
- `public AudioClip[] gearShiftingClips;`
  Array of audio clips played on gear shifts.

- `public AudioClip[] crashClips;`
  Collision or crash impact sounds.
- `public AudioClip reversingClip;`
- `public AudioClip windClip;`
- `public AudioClip brakeClip;`
- `public AudioClip wheelDeflateClip;`
- `public AudioClip wheelInflateClip;`
- `public AudioClip wheelFlatClip;`
- `public AudioClip indicatorClip;`
- `public AudioClip bumpClip;`
- `public AudioClip NOSClip;`
- `public AudioClip turboClip;`
- `public AudioClip[] blowoutClip;`
- `public AudioClip[] exhaustFlameClips;`
  Various single or array-based sound effects for reversing, tire blowing out, engine idle, etc.
- **Volume Limitations**
  - `[Range(0f, 1f)] public float maxGearShiftingSoundVolume = .25f;`
  - `[Range(0f, 1f)] public float maxCrashSoundVolume = 1f;`
  - `[Range(0f, 1f)] public float maxWindSoundVolume = .1f;`
  - `[Range(0f, 1f)] public float maxBrakeSoundVolume = .1f;`
    Clamps the volume of specific effects to avoid overpowering the mix.

4.8 Particle and Skidmarks Control

- `public GameObject contactParticles;`
  Instantiated upon wheel or vehicle contact with surfaces.

- `public GameObject scratchParticles;`
  For scratching or collision visuals.

- `public GameObject wheelDeflateParticles;`
  Used when a tire is deflated or destroyed.

- `public RCC_SkidmarksManager skidmarksManager;`
  Reference to a skidmark manager object controlling skidmark generation in the scene.

- `public bool dontUseAnyParticleEffects = false;`
  If true, disables all RCC-related particle instantiations.

- `public bool dontUseSkidmarks = false;`
  If true, no skidmarks will be generated even if `skidmarksManager` is assigned.

4.9 Folding Sections (Editor Only)

- `public bool foldGeneralSettings = false;`
- `public bool foldBehaviorSettings = false;`
- `public bool foldControllerSettings = false;`
- `public bool foldUISettings = false;`
- `public bool foldWheelPhysics = false;`
- `public bool foldSFX = false;`
- `public bool foldOptimization = false;`
- `public bool foldTagsAndLayers = false;`

Flags used in custom editor scripts to hide or show different sections of RCC Settings. These do not affect runtime behavior but improve organization in the Inspector.

---

5. Usage Notes and Best Practices

1. **Creating the ScriptableObject**

   - Right-click in Unity's Project window, select *Create → RCC → RCC_Settings* (if the menu is provided by RCC).
   - Alternatively, manually create a scriptable object file and place it in `Assets/Resources/RCC Assets/` with the filename `RCC_Settings.asset`.

2. **Referencing via `RCC_Settings.Instance`**
    - Make sure the asset is named exactly `RCC_Settings` and resides under a folder `Resources/RCC Assets/`.
    - Access configuration anywhere with `RCC_Settings.Instance.<fieldName>`.
3. **Behavior Profiles**
    - Add multiple `BehaviorType` entries to `behaviorTypes`. Each can be tuned differently (arcade vs. simulation).
    - Toggle `overrideBehavior` to decide whether these profiles are forced or not.
4. **Performance**
    - Adjusting `fixedTimeStep` and `maxAngularVelocity` can drastically affect physics performance and realism.
    - If performance is poor on lower-end devices, consider increasing `fixedTimeStep` (e.g., from 0.02 to 0.03) and capping FPS to reduce load.
5. **Layer Setup**
    - The layers (`RCCLayer`, `WheelColliderLayer`, `DetachablePartLayer`, `PropLayer`) must match entries in Unity's **Project Settings → Tags and Layers**.
    - If `setLayers` is true, RCC tries to manage these automatically. Conflicts can occur if other systems reuse these layer names.
6. **Audio**
    - Keep an AudioMixer assigned to `audioMixer` if you want finer control over volumes.
    - Ensure correct volume ranges to avoid overwhelming the player with SFX.
7. **Particles and Skidmarks**
    - Disabling `dontUseAnyParticleEffects` or `dontUseSkidmarks` can yield performance gains, especially on mobile.
    - If you do want skidmarks, ensure `skidmarksManager` is assigned in the scene.

---

6. Summary

`RCC_Settings` is the **central configuration asset** that governs how vehicles, physics, UI, audio, and special effects are handled within **Realistic Car Controller**. By organizing these settings in a single `ScriptableObject`, developers can easily create multiple presets, target different platforms, and customize the entire RCC experience from one location.

Use this asset carefully—changes to fixed timestep or behavior profiles can dramatically affect vehicle feel and performance. Test thoroughly across your target devices or build settings to ensure you achieve the desired driving experience with optimal performance.