Table of Contents

1. Overview

`RCC_WheelCollider` is a wrapper around Unity’s `WheelCollider` in **Realistic Car Controller (RCC)**. It handles:

- **Wheel alignment** (spinning the visual mesh).
- **Friction** (forward/sideways slip, traction, drift).
- **Skid logic** (particles, audio, marks).
- **Brake, handbrake, and engine torque** applications.
- **Damage, deflation** (flat tires).
- **Ackerman steering** angles for front wheels.

Used by `RCC_CarControllerV4` to integrate each physical wheel with advanced RCC features.

---

2. Class Declaration

```
[RequireComponent(typeof(WheelCollider))]
public class RCC_WheelCollider : RCC_Core {
   // Implementation details...
}
```

- Inherits from `RCC_Core`, giving it access to shared resources (e.g., ground materials, settings).
- Requires a `WheelCollider` component on the same GameObject.

---

3. Primary Responsibilities

1. **Aligning** the wheel mesh to the **WheelCollider**’s position and rotation each frame.
2. **Applying** brake torque, motor torque, or steering angles from `RCC_CarControllerV4`.
3. **Detecting slip** to play skid sounds, spawn particles, and draw skidmarks.
4. **Friction logic** for drifting, traction helper, and dynamic friction curves.
5. **Deflating/Inflating** tires if needed.
6. **Ackerman** steering for realistic front wheel geometry.

---

4. Core Fields and Properties

4.1 References and Geometry

- **`public WheelCollider WheelCollider`**: The underlying Unity WheelCollider.
- **`public Rigidbody Rigid`**: The parent vehicle’s rigidbody (from `CarController.Rigid`).
- **`public Transform wheelModel`**: Mesh transform for visual alignment.

4.2 Wheel State and Alignment

- **`public bool isGrounded`**: Whether the wheel is on the ground (from `GetGroundHit`).

- `public bool alignWheel = true`: If true, automatically aligns `wheelModel` each frame.
- `public float wheelWidth, wheelOffset`: Basic geometry parameters.
- `[Range(-5f, 5f)] public float camber, caster, toe`: Visual angles for wheel tilt.

4.3 Friction and Slip Tracking

- `wheelHit`: A `WheelHit` struct from `WheelCollider.GetGroundHit`.
- `public float wheelSlipAmountForward, wheelSlipAmountSideways`: Slip values.
- `public float totalSlip`: Combined slip.
- `forwardFrictionCurve`, `sidewaysFrictionCurve`: Active friction curves.
- `forwardFrictionCurve_Org`, `sidewaysFrictionCurve_Org`: Original friction curves for reference.

4.4 Audio and Particles

- `private AudioSource audioSource`: For skid sounds.
- `public List<ParticleSystem> allWheelParticles`: One particle system per ground material.
- Optionally spawns **deflation** particles if tire goes flat.

4.5 Traction Helpers and Deflation

- `public float tractionHelpedSidewaysStiffness = 1f`: Extra factor for traction helper or ESP.
- `public bool deflated`: Tire is currently deflated.
- `public float deflateRadiusMultiplier = 0.8f`: Wheel radius shrinks on deflation.
- `public float deflatedStiffnessMultiplier = 0.5f`: Reduces friction curves if deflated.

4.6 Ackerman Steering

- `public float ackermanWheelBase = 2.55f, ackermanTrackWidth = 1.5f, ackermanSteerReference = 6f`
- Used to compute left/right steering angles for better cornering geometry.

---

5. Initialization and Setup

5.1 Awake()

- Resizes wheel mass if `Settings.useFixedWheelColliders` is true (scales with the parent rigidbody mass).
- Creates a pivot transform around `wheelModel` for correct spinning.
- Applies friction curves from the RCC settings or the vehicle's override.
- Creates an **AudioSource** for skid SFX and spawns **ParticleSystems** for each ground material.

5.2 OnEnable()

- Subscribes to `RCC_SceneManager.OnBehaviorChanged` to refresh friction on global changes.
- Resets slip, bump, and audio volumes.
- Resets motor/brake torque to zero.

---

6. Update and FixedUpdate Logic

6.1 Wheel Alignment in `Update()`

- If `alignWheel = true`, calls **`WheelAlign()`** to set wheel mesh position/rotation from the physics wheel.

6.2 Friction, Slip, and Collisions in `FixedUpdate()`

1. Calculates approximate wheel speed from `WheelCollider.rpm`.
2. Re-checks if it can power or steer depending on the vehicle's drivetrain or overrides.
3. **`GroundMaterial()`**: identifies surface friction index.
4. **`Frictions()`**: updates friction curves based on slip, drift logic, or traction helper.
5. **`TotalSlip()`**: merges sideways + forward slip.
6. **`SkidMarks()`**: draws skid lines if `totalSlip` exceeds threshold.
7. **`Particles()`**: toggles dust/smoke for slip.
8. **`Audio()`**: plays skid audio, checks bumps.
9. **`CheckDeflate()`**: sees if tire is deflated or not.
10. **`ESP()`**: extra brake if under/over-steering with `CarController.ESP`.

---

7. Friction and Slip Methods

7.1 Ground Material Detection

- **`GroundMaterial()`**:

    - Uses `WheelCollider.GetGroundHit()` and checks `sharedMaterial` vs. `GroundMaterials.frictions`.
    - For **terrain**, tries to detect which splat index is under the wheel.
    - Sets `groundIndex` used to pick friction slip thresholds, audio, and particles.

7.2 Applying Engine Torque, Steering, and Braking

- **`ApplyMotorTorque(float torque)`**: Applies motor torque with TCS logic. If slip is too high, reduce torque.
- **`ApplySteering(float steerInput, float angle)`**: Optionally uses an Ackerman formula for left/right wheels.
- **`ApplyBrakeTorque(float torque)`**: If ABS is on, can cut brake torque if slip exceeds threshold.

7.3 Skid Marks, Particles, and Audio

- **`SkidMarks()`**: If slip > ground friction threshold, logs positions for the global `RCC_SkidmarksManager`.
- **`Particles()`**: Activates the matching ground material's ParticleSystem if slipping.
- **`Audio()`**:
    - Fades in skid sound based on slip magnitude.
    - Plays a **bump** sound if wheelHit force changes sharply.

7.4 Drift Logic

- If the selected RCC behavior uses drifting logic, **`Drift()`** function modifies friction curves to reduce lateral and forward friction. It can also add small sideways forces for a drifting feel (particularly for RWD wheels).

---

8. Deflation Handling

- **`deflated`** toggles if the tire is flat.
- **`Deflate()`**: sets `WheelCollider.radius` to `defRadius * deflateRadiusMultiplier`, triggers deflate audio and possibly a deflate particle effect.
- **`Inflate()`**: restores original radius, friction, and plays inflate audio.
- If deflated, friction is multiplied by `deflatedStiffnessMultiplier`.

---

9. Ackerman Steering (Optional)

- `ApplySteering()` can do basic ackerman if the wheel is on the left or right side.
- Uses `ackermanWheelBase`, `ackermanTrackWidth`, `ackermanSteerReference` to compute a correct turn angle difference between left and right wheels.

---

10. Events and Deactivation

- On behavior changes (`RCC_SceneManager.OnBehaviorChanged`), re-applies friction settings.
- **`OnDisable()`**: stops audio, resets slip, motor torque, brake torque, etc.

---

11. Usage and Best Practices

1. **Link** each `RCC_WheelCollider` with a **visual wheel** (`wheelModel`) in the inspector.
2. **Pivot** the wheel model in the center for correct spinning (the script calls `CreatePivotOfTheWheel()` automatically).
3. **Ground Materials** must be set up in `GroundMaterials.frictions` for different surfaces (asphalt, grass, etc.).
4. **Terrain** detection is only used if the terrain is registered in `RCC_SceneManager` and `RCC_SceneManager.terrainsInitialized = true`.
5. **Deflation** is triggered if the ground material marks it as deflate or if manually calling `Deflate()`.
6. **Steering**
   - If you want custom angles (not Ackerman), you can just set `WheelCollider.steerAngle = angle`.
7. **Audio**
   - The skid audio is looped, volume changes with slip. Bump sounds are triggered on collisions in `audioSource`.
8. **Performance**
   - Each wheel in `FixedUpdate()` does a linecast for ground materials, friction updates, etc. For many wheels or complex terrain, consider optimization.
9. **Drift**
   - The `Drift()` method is quite simplified. You can further customize friction logic for different drift styles.

---

12. Summary

`RCC_WheelCollider` is a **comprehensive wheel script** in Realistic Car Controller. It handles:

- **Alignment** of the visual wheel mesh.
- **Ground detection** to pick friction, slip thresholds, and skid logic.
- **Applying torque, brake, steering** including advanced TCS/ABS/ESP.
- **Skidmarks, particles, audio** for slipping or drift scenarios.
- **Deflation** logic (flat tires) with radius changes and friction penalties.
- **Ackerman** steering for realistic front steering geometry.

It integrates seamlessly with `RCC_CarControllerV4`, providing each wheel the **simulation detail** needed for realistic driving experiences.