Table of Contents

1. Overview
2. Class Declaration
3. Core Purpose and Features
4. Important Fields and Configuration
   4.1 CameraTarget
   4.2 Camera Mode
   4.3 Camera Pivot and Actual Camera
   4.4 TPS (Third-Person) Settings
   4.5 Hood (FPS) and Wheel Camera Settings
   4.6 Fixed, Cinematic, and Top Camera Mode
   4.7 Orbit Controls
   4.8 Collision and Occlusion
5. Workflow / Lifecycle Methods
   5.1 Awake()
   5.2 OnEnable()
   5.3 Update()
   5.4 LateUpdate()
   5.5 FixedUpdate()
   5.6 OnDisable()
6. Camera Mode Implementations
   6.1 TPS Mode
   6.2 FPS (Hood) Mode
   6.3 Wheel Mode
   6.4 Fixed Mode
   6.5 Cinematic Mode
   6.6 Top Mode
7. Orbit Handling
8. Auto-Focus Logic
9. Collision Effects
10. Changing Camera Modes
11. Events and External Integration
12. Usage Notes and Best Practices
13. Summary

---

1. Overview

`RCC_Camera` is the **main camera controller** in **Realistic Car Controller (RCC)**. It supports multiple camera modes:

- Third-Person (TPS)

- Hood (FPS)
- Wheel Camera
- Fixed Camera
- Cinematic Camera
- Top (top-down)

It can optionally detect **occlusion**, handle **look-back** functionality, perform camera collisions, and smoothly switch between modes. Also includes orbit controls for the user to drag or rotate the camera.

---

2. Class Declaration

```
public class RCC_Camera : RCC_Core {
    // ...
}
```

- Extends `RCC_Core`, providing access to RCC utilities and settings.
- Typically placed in the scene as the main camera object or a prefab.

---

3. Core Purpose and Features

1. **Centrally Manages** the camera for a single RCC vehicle.
2. **Multiple Modes**:
    - **TPS** (behind the car, tilt, collision detection)
    - **FPS** (Hood camera)
    - **Wheel** (close-up on the wheels)
    - **Fixed** (stationary vantage points)
    - **Cinematic** (dynamic or staged shots)
    - **Top** (top-down or orthographic).
3. **Orbit Controls**: Rotates TPS or Hood camera by drag gestures or input.
4. **AutoFocus**: Adjusts TPS distance/height for big or small vehicles.
5. **Occlusion**: Raycasts to avoid camera clipping through walls.
6. **Collision Effects**: Shakes or offsets the camera if the vehicle experiences collisions.

---

4. Important Fields and Configuration

4.1 `CameraTarget`

```
[System.Serializable]
public class CameraTarget {
```

```
    public RCC_CarControllerV4 playerVehicle;
    public float Speed { ... }
    public Vector3 Velocity { ... }
    public RCC_HoodCamera HoodCamera { ... }
    public RCC_WheelCamera WheelCamera { ... }
}
```

- Stores references and quick info about the **vehicle** being followed.
- Exposes `.Speed`, `.Velocity`, `.HoodCamera`, and `.WheelCamera`.

4.2 Camera Mode

```
public enum CameraMode { TPS, FPS, WHEEL, FIXED, CINEMATIC, TOP }
public CameraMode cameraMode = CameraMode.TPS;
```

- The currently active mode.
- The camera can automatically cycle or be changed by the user.

4.3 Camera Pivot and Actual Camera

- **`public Camera actualCamera`**: The Unity `Camera` component used for rendering.
- **`public GameObject pivot`**: A pivot object for local offsets or collision offsets. The camera is a child of this pivot.

4.4 TPS (Third-Person) Settings

- **`public float TPSDistance = 6f, TPSHeight = 2f`**: Base offset behind and above vehicle.
- **`public float TPSMinimumFOV = 40f, TPSMaximumFOV = 60f`**: Auto-FOV range in TPS.
- **`public Vector3 TPSOffset`**: Additional local offset.
- **`public float TPSRotationDamping = 0.7f`**: Smoothness for camera rotation.
- **`public bool TPSAutoFocus = true, TPSAutoReverse = true, TPSCollision = true`**.
- **`public bool TPSLockX, TPSLockY, TPSLockZ`**: Locks camera rotation to that axis if true.
- **`public bool TPSFreeFall`**: If vehicle is airborne, can skip certain rotation constraints.
- **`public bool TPSDynamic`**: If true, camera position adjusts more with acceleration/velocity.

4.5 Hood (FPS) and Wheel Camera Settings

- **`public float hoodCameraFOV = 60f`**: FOV for hood mode.
- **`public float wheelCameraFOV = 60f`**: FOV for wheel camera mode.
- If a hood camera or wheel camera is present on the vehicle, `RCC_Camera` can adopt that vantage point.

4.6 Fixed, Cinematic, and Top Camera Mode

- **`public bool useFixedCameraMode`**, **`public bool useCinematicCameraMode`**, **`public bool useTopCameraMode`**: Toggles whether these modes are included in the cycle.
- **`public bool useOrthoForTopCamera`**: If true, top camera uses orthographic projection.
- **`public Vector3 topCameraAngle = new Vector3(45f, 45f, 0f)`**: A typical vantage angle for top-down.
- **`public float topCameraDistance = 100f`**: Zoom distance in top-down.
- **`public float minimumOrtSize = 10f, maximumOrtSize = 20f`**: If orthographic, sets the min/max size.

4.7 Orbit Controls

- **`public float orbitX, orbitY`**: Current orbit angles.
- **`public float orbitXSpeed, orbitYSpeed`**: Speed of orbit rotation.
- **`public float minOrbitY, maxOrbitY`**: Vertical clamp.
- **`public bool orbitReset`**: If true, orbit angles reset after a timer if the car is fast enough.

4.8 Collision and Occlusion

- **`public bool useOcclusion`**: If true, `OccludeRay()` checks line of sight to the vehicle and repositions camera to avoid clipping.
- **`public LayerMask occlusionLayerMask`**: Layers to check for collision.
- **`public bool TPSCollision`**: If true, the camera shakes or offsets on collisions.
- The class tracks collisions in `Collision()`.

---

5. Workflow / Lifecycle Methods

5.1 Awake()

- Ensures a pivot object exists.
- Finds or creates `actualCamera`.
- Checks for URP pipeline if `RCC_URP` is defined (optional).

5.2 OnEnable()

- Fires **`OnBCGCameraSpawned`** event, notifying that this camera is active.
- Subscribes to:
    - **`RCC_CarControllerV4.OnRCCPlayerCollision`** for collision shake.
    - **`RCC_InputManager.OnChangeCamera`** for switching camera mode.
    - **`RCC_InputManager.OnLookBack`** for toggling rear view.

5.3 Update()

- Manages **field-of-view** smoothing.
- If no player vehicle is set, does nothing.

5.4 LateUpdate()

- For each frame:
    1. If no vehicle, return.
    2. Executes logic for the current `cameraMode` (TPS, FPS, Wheel, etc.).
    3. If `useAutoChangeCamera` is true, increments a timer and can auto-switch after some seconds.

5.5 FixedUpdate()

- Checks for occlusion linecast.
- Calculates local acceleration from the vehicle's velocity changes for possible camera offsets.

5.6 OnDisable()

- Unsubscribes from collision and input events.

---

6. Camera Mode Implementations

6.1 TPS Mode

- **`TPS()`**:
    1. Rotates camera behind the car with optional rotation damping.
    2. Offsets position with `TPSDistance` / `TPSHeight` and optional `TPSOffset`.
    3. If `useOcclusion`, does a linecast.
    4. Applies collision offset.
    5. Tilts camera based on sideways velocity.
    6. Adjusts FOV from `TPSMinimumFOV` to `TPSMaximumFOV` based on speed.

6.2 FPS (Hood) Mode

- **`FPS()`**:
    1. Camera is parented under the hood camera transform.
    2. Possibly uses orbit rotation if `useOrbitInHoodCameraMode`.
    3. Field of view = `hoodCameraFOV`.

6.3 Wheel Mode

- **`WHEEL()`**:
    - Camera is parented under a `RCC_WheelCamera` on the car.
    - If occluded, might revert to TPS.

6.4 Fixed Mode

- **`FIXED()`**:
    - Switches to a static vantage from `RCC_FixedCamera`.
    - If occluded, tries to relocate the fixed camera or revert to TPS.

6.5 Cinematic Mode

- **`CINEMATIC()`**:
    - The camera is parented under `RCC_CinematicCamera` pivot.
    - If occluded, reverts to TPS or tries another angle.

6.6 Top Mode

- **`TOP()`**:
    - Optionally sets `actualCamera.orthographic = useOrthoForTopCamera`.
    - Positions camera above the car at `topCameraAngle` and `topCameraDistance`.
    - Adjusts `topCameraDistanceOffset` with speed for slight trailing.
    - Clamps orthographic size or perspective FOV accordingly.

---

7. Orbit Handling

- **`orbitX, orbitY`** store user rotation.
- If user drags with pointer or joystick, these angles shift.
- Angles clamp vertical orbit with `minOrbitY, maxOrbitY`.
- After a period (`orbitResetTimer`), it can reset to 0 if the vehicle is moving quickly.

---

8. Auto-Focus Logic

- **`AutoFocus()`** coroutines measure **vehicle bounds** via `RCC_GetBounds` and smoothly adjust `TPSDistance` and `TPSHeight`.
- Helps large or small vehicles fit well in the camera frame from the get-go.

---

9. Collision Effects

- Subscribes to **`RCC_CarControllerV4.OnRCCPlayerCollision`**.
- If in TPS mode (`TPSCollision = true`), shifts pivot position / rotation briefly and modifies field-of-view in response to collision impulse.

---

10. Changing Camera Modes

- **`ChangeCamera()`** cycles through modes in an order: TPS → FPS (hood) → Wheel → Fixed → Cinematic → Top.
- Skips modes that are disabled or not available.
- Or **`ChangeCamera(CameraMode mode)`** directly sets it.

---

11. Events and External Integration

- **`OnBCGCameraSpawned`** fired on `OnEnable()`, letting other scripts know this camera is active.
- Hooks into `RCC_InputManager` for `OnChangeCamera()` and `OnLookBack()`.
- Hooks into `RCC_CarControllerV4.OnRCCPlayerCollision` for collisions.

---

12. Usage Notes and Best Practices

1. **Assigning the Vehicle**
    - Typically, call `SetTarget(RCC_CarControllerV4 playerVehicle)` after spawning or switching cars.
2. **Pivot**
    - The `pivot` child organizes local offsets and collision offsets. The `actualCamera` is another child.
3. **Modes**
    - Each mode has distinct behavior. If you don't want a certain mode, set the corresponding boolean (e.g. `useTopCameraMode = false`).
4. **Performance**
    - Occlusion Raycasts happen in `FixedUpdate()`. For large scenes or many cameras, consider reducing frequency or disabling occlusion.
5. **LookBack**
    - `lookBackNow` in TPS mode flips the direction factor.
6. **AutoFocus**
    - Useful for quickly matching camera distance to any newly spawned vehicle.
    - If you want manual control, set `TPSAutoFocus = false`.
7. **Orbit**
    - If using orbit on a platform with pointer or joystick, ensure you call `OnDrag()` or handle orbitX/orbitY yourself.

---

13. Summary

`RCC_Camera` orchestrates a versatile camera system for **Realistic Car Controller**:

- **Multiple dynamic camera modes**: TPS, FPS (hood), wheel, fixed, cinematic, top-down.
- **Orbit** and **occlusion** support.
- **Collision** camera shake and auto-FOV.
- **AutoFocus** to accommodate vehicle size.
- Events for integration (spawn, collision, change camera).

By toggling its features and calling the relevant functions (e.g., `SetTarget`, `ChangeCamera`), developers can easily tailor a driving camera that suits a variety of gameplay styles, from realistic chase cam to top-down arcade or cinematic follow shots.