

RCC_Damage

Public Variables

```
internal RCC_CarControllerV3 carController; // Car controller.

public bool automaticInstallation = true; // If set to enabled, all parts of the vehicle will be
processed. If disabled, each part can be selected individually.

// Mesh deformation
public bool meshDeformation = true;
public DeformationMode deformationMode = DeformationMode.Fast;

public int damageResolution = 100; // Resolution of the deformation.

public LayerMask damageFilter = -1; // LayerMask filter. Damage will be taken from the objects
with these layers.

public float damageRadius = .5f; // Vertices in this radius will be effected on collisions.

public float damageMultiplier = 1f; // Damage multiplier.

public float maximumDamage = .5f; // Maximum Vert Distance For Limiting Damage. 0 Value Will
Disable The Limit.

public struct originalMeshVerts { public Vector3[] meshVerts; } // Struct for Original Mesh Vertices
positions.

public struct originalWheelPos { public Vector3 wheelPosition; public Quaternion wheelRotation; }
public struct meshCol { public Collider col; public bool created; }

public originalMeshVerts[] originalMeshData; // Array for struct above.

public originalMeshVerts[] damagedMeshData; // Array for struct above.

public originalWheelPos[] originalWheelData; // Array for struct above.

public originalWheelPos[] damagedWheelData; // Array for struct above.

public bool repairNow = false; // Repairing now.

public bool repaired = true; // Returns true if vehicle is completely repaired.

public bool recalculateNormals = true; // Recalculate normals while deforming / restoring the
mesh.

public bool recalculateBounds = true; // Recalculate bounds while deforming / restoring the
mesh.

public bool wheelDamage = true; // Use wheel damage.
```

```

public float wheelDamageRadius = .5f;    // Wheel damage radius.

public float wheelDamageMultiplier = 1f;    // Wheel damage multiplier.

public bool wheelDetachment = true;    // Use wheel detachment.

public bool lightDamage = true;    // Use light damage.

public float lightDamageRadius = .5f;    //Light damage radius.

public float lightDamageMultiplier = 1f;    //Light damage multiplier.

public bool partDamage = true;    // Use part damage.

public float partDamageRadius = .5f;    //Light damage radius.

public float partDamageMultiplier = 1f;    //Light damage multiplier.

public MeshFilter[] meshFilters;    // Collected mesh filters.

public RCC_DetachablePart[] detachableParts;    // Collected detachable parts.

public RCC_Light[] lights;    // Collected lights.

public RCC_WheelCollider[] wheels;    // Collected wheels.

```

Public Methods

```

public void Initialize(RCC_CarControllerV3 _carController) {}    // Collecting all
meshes and detachable parts of the vehicle.

/// Gets all meshes.
public void GetMeshes(MeshFilter[] allMeshFilters) {}

/// Gets all lights.
public void GetLights(RCC_Light[] allLights) {}

/// Gets all detachable parts.
public void GetParts(RCC_DetachablePart[] allParts) {}

/// Gets all wheels
public void GetWheels(RCC_WheelCollider[] allWheels) {}

/// Moving deformed vertices to their original positions while repairing.
public void UpdateRepair() {}

/// Moving vertices of the collided meshes to the damaged positions while deforming.
public void UpdateDamage() {}

```

```
/// Detaches the target wheel.  
public void DetachWheel(RCC_WheelCollider wheelCollider) {}
```

```
/// Raises the collision enter event.  
public void OnCollision(Collision collision) {}
```