Tiny Car Controller Advance

User manual v2.0

Table of Contents

Description

This controller allows you to create and control a basic vehicle with fully configurable, arcade-like physics.

Instead of using Unity's default wheel collider, which is often needlessly complicated and prone to glitches, the wheels on this controller are made of sphere colliders with several specifically configured joints to mimic a vehicle's motor, steering, and suspension.

Features

- Easy hassle-free setup
- Control acceleration, speed, friction, transmission, collisions, and more
- Uses Unity's physics engine for perfect compatibility with other assets
- Lightweight, perfect for mobile games
- Includes example scripts to take care of input and camera
- Compatible with any OS, render pipeline, or Unity version

Requirements

Unity version 2019.4 or later is recommended, but it should also work with any newer versions.

Intermediate C# programming knowledge is strongly advised.

Contact

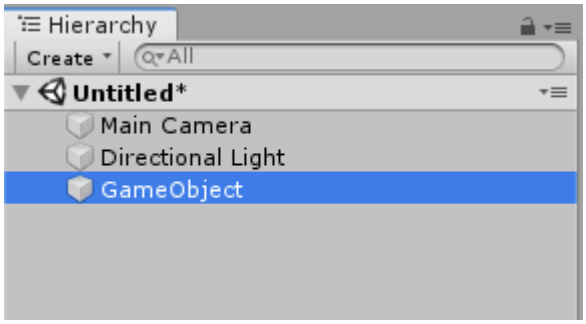
David Jalbert (programmer)
Email: jalbert.d@hotmail.com
Mastodon: https://mastodon.gamedev.place/@davidjayindie
Twitter: https://twitter.com/DavidJayIndie

Unity asset store package
https://assetstore.unity.com/packages/slug/198873

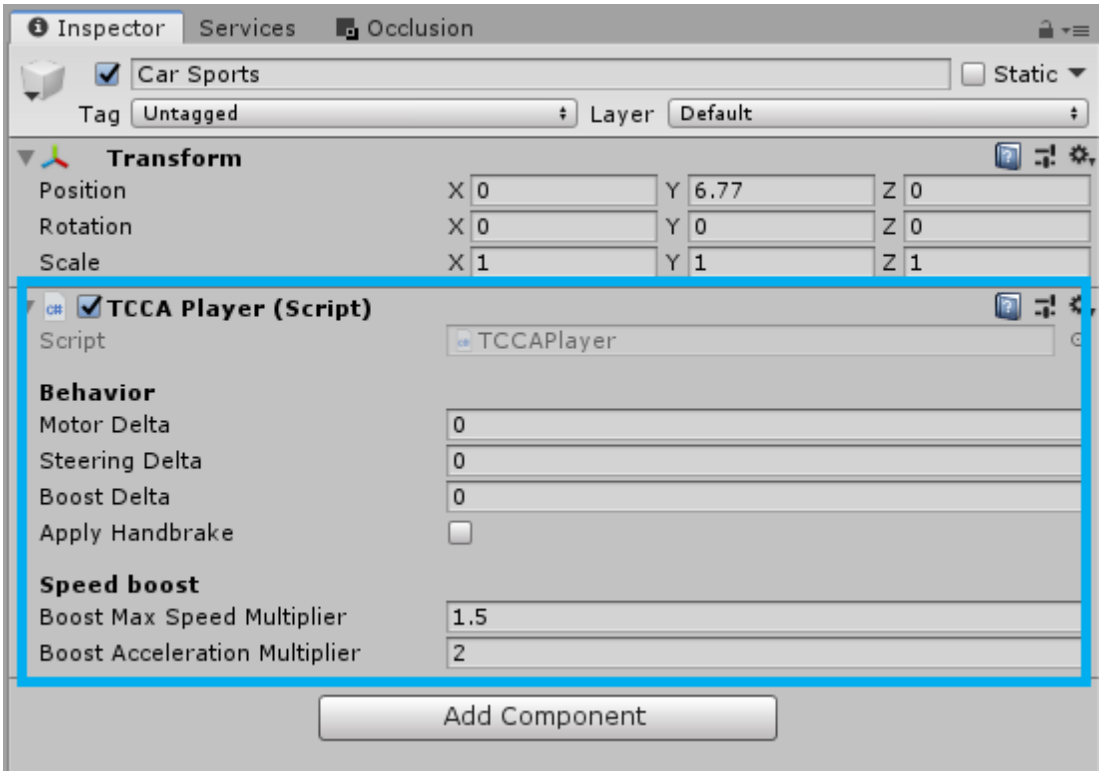
WebGL demo
https://davidjalbert.itch.io/tiny-car-controller-advance-webgl-demo

Minimal setup

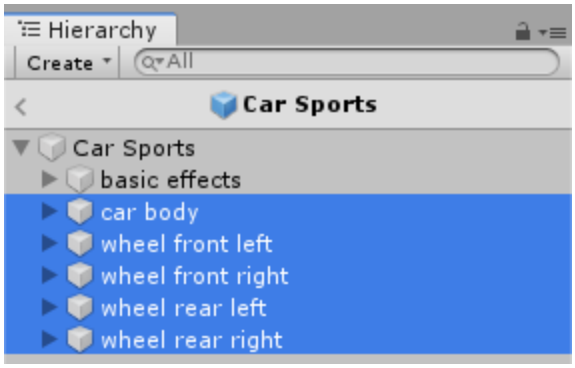
1) Create an empty GameObject in your scene.



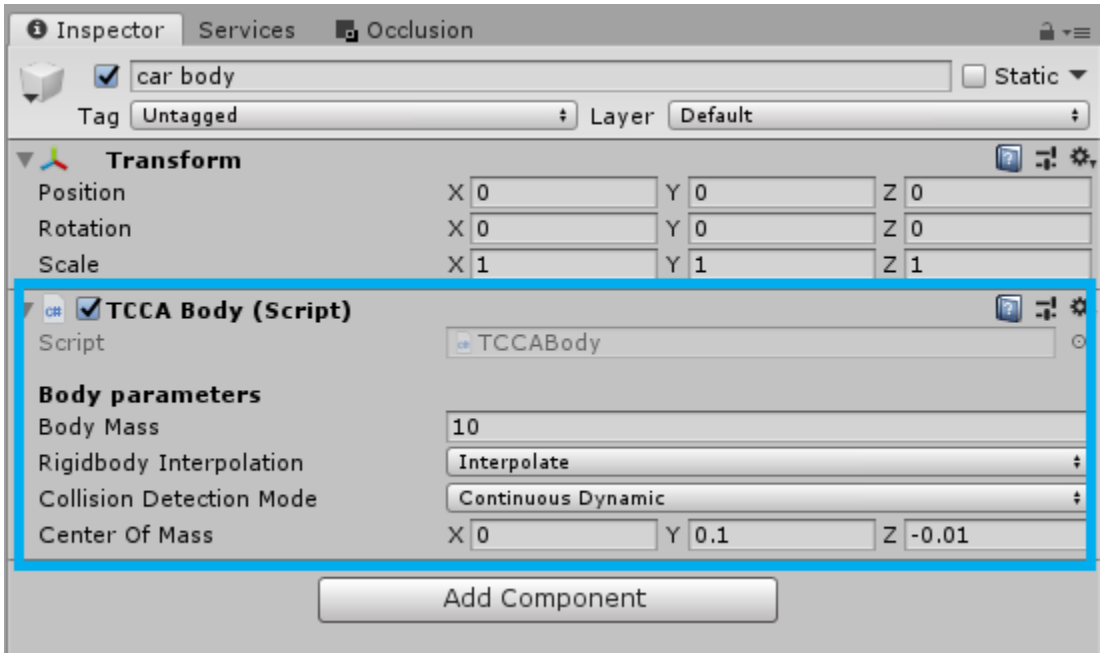
2) Add the script at “Assets/DavidJalbert/TinyCarControllerAdvance/Scripts/Core/TCCAPlayer.cs” to the empty GameObject



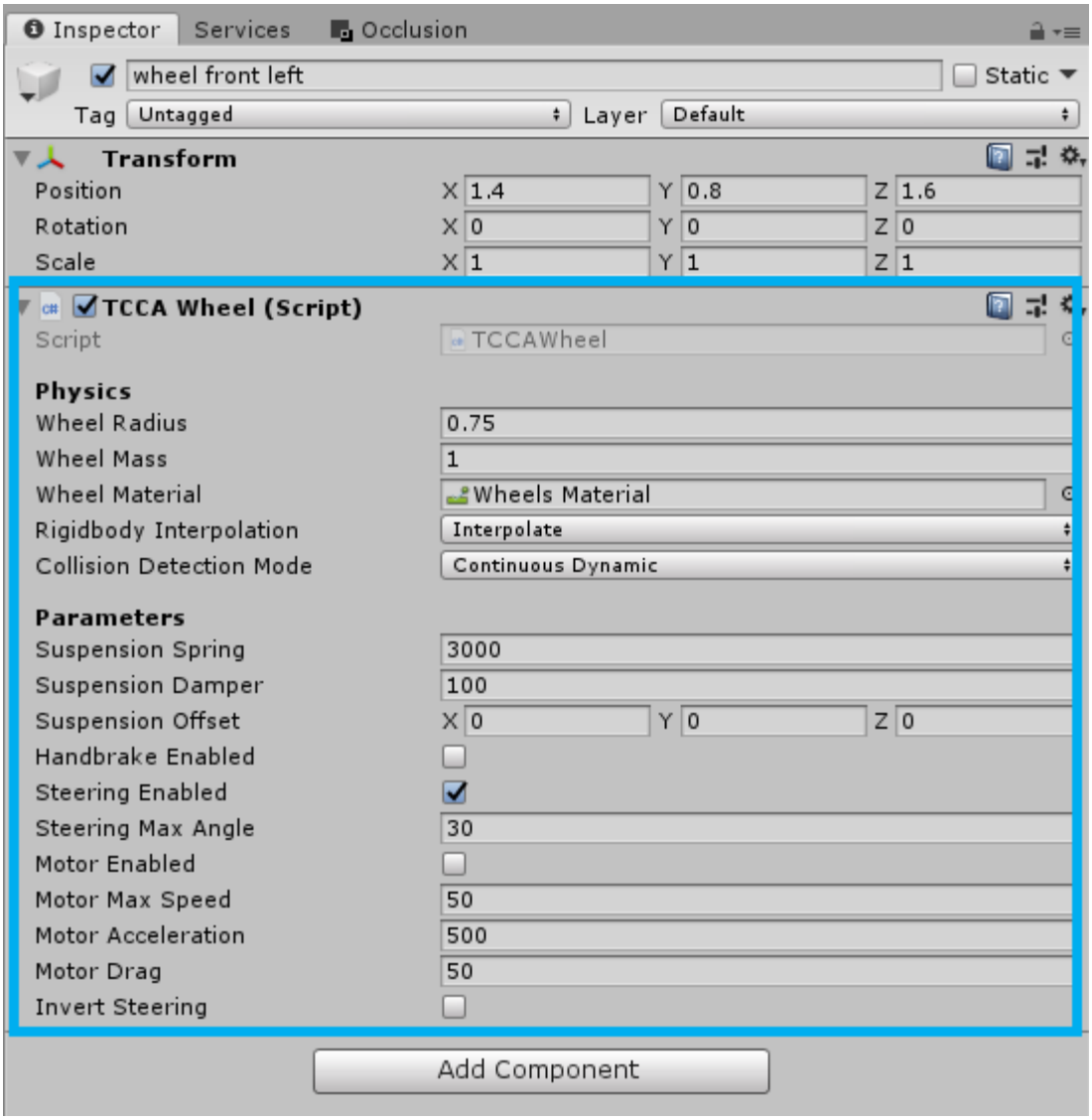
3) Create children for the car body and its wheels. Put your 3d models inside these objects.



4) Add the script at “Assets/DavidJalbert/TinyCarControllerAdvance/Scripts/Core/TCCABody.cs” to the car body object.



5) Add the script at “Assets/DavidJalbert/TinyCarControllerAdvance/Scripts/Core/TCCAWheel.cs” to each of the car's wheel objects. Note that if there are colliders to the wheel models, they will be disabled when the game starts.



6) Set the parameters of the three scripts to your liking (see below for an explanation).

At this point, your controller is ready to use, but it will not have any input or camera control. To do that, you need additional scripts, which are used in the example scene. Check out the scene at “DavidJalbert\TinyCarControllerAdvance” to see how to use them.

Parameters

TCCA Player

Motor Delta

How much torque to apply to the wheels. 1 is full speed forward, -1 is full speed backward, 0 is rest.

Steering Delta

How much steering to apply to the wheels. 1 is right, -1 is left, 0 is straight.

Boost Delta

How much boost to apply to the wheels. 1 is full boost, 0 is no boost.

Apply Handbrake

Whether to apply the handbrake to the wheels.

Boost Max Speed Multiplier

Speed multiplier to apply when using the boost.

Boost Acceleration Multiplier

Acceleration multiplier to apply when using the boost.

Freeze Position X/Y/Z

Prevents the vehicle from moving along an axis.

TCCA Body

Body Mass

The mass that will be applied to the body.

Rigidbody Interpolation

Whether to apply interpolation to the body.

Collision Detection Mode

Which collision detection mode to use on the body.

Center Of Mass

The center of mass of the body in local space. Ideally this should be the center of the car at ground level. Change the Z value to make the car lean backward or forward when in the air.

Roll Counter Mode

When to apply roll countering force.

Roll Counter Target Angle

The angle in degrees to which to rotate the vehicle. Set to 0 to roll perfectly upright.

Roll Counter Force

How much force to apply to rotate the vehicle upright if it rolls over.

Roll Counter Smoothing

How fast to rotate the vehicle upright if it rolls over. Set to zero to make this instantaneous.

Roll Counter Over Speed

How much force, between 0 (none) and 1 (max), to apply relative to the vehicle's speed, between 0 (stationary) and 1 (max speed).

Pitch Counter Mode

When to apply pitch countering force.

Pitch Counter Target Angle

The angle in degrees to which to rotate the vehicle. Set to 0 to level perfectly straight.

Pitch Counter Force

How much force to apply to level the vehicle.

Pitch Counter Smoothing

How fast to level the vehicle. Set to zero to make this instantaneous.

Pitch Counter Over Speed

How much force, between 0 (none) and 1 (max), to apply relative to the vehicle's speed, between 0 (stationary) and 1 (max speed).

Steering Stabilizer Mode

When to apply force to stabilize the steering direction. This adds an additional angular force to the body corresponding to the current steering value.

Steering Stabilizer Force

How much force to apply to stabilize the steering.

Steering Stabilizer Smoothing

How fast to stabilize the steering. Set to zero to make this instantaneous.

Steering Stabilizer Over Speed

How much stabilization to apply to the vehicle depending on its speed. Normally you want it to be zero when not moving (at zero speed) and increase it to one (full stabilization) when the speed starts raising.

TCCA Wheel

Wheel Radius

Radius of the wheel collider. This should be equal to the size of the wheel model.

Wheel Mass

Mass of the wheel rigidbody.

Wheel Material

Material of the wheel rigidbody.

Rigidbody Interpolation

Whether to use interpolation for the wheel rigidbody.

Collision Detection Mode

Which collision detection mode to use for the wheel collider.

Suspension Spring

Force applied to the suspension. Higher values make the suspension stiffer.

Suspension Damper

Damper applied to the suspension. Higher values make the suspension settle faster.

Suspension Offset

Shifts the position of the wheel relative to its initial position. Useful to mimic hydraulics.

Steering Enabled

Whether to allow the wheel to turn left or right.

Steering Max Angle

Maximum angle at which the wheel can turn.

Steering Over Speed

How much steering, between 0 (none) and 1 (max), to apply relative to the vehicle's speed, between 0 (stationary) and 1 (max speed).

Steering Spring

The amount of force to apply to the steering axle.

Steering Damper

The amount of friction to apply to the steering axle.

Invert Steering

Whether to invert steering. Useful for the rear wheels if you want to have a four wheel steering.

Motor Enabled

Whether to allow the wheel to accelerate.

Motor Max Speed

Maximum speed to which the wheel can keep accelerating when using the motor.

Motor Acceleration

Maximum acceleration to apply to the wheel when using the motor.

Motor Acceleration Over Speed

How much acceleration, between 0 (none) and 1 (max), to apply relative to the vehicle's speed, between 0 (stationary) and 1 (max speed).

Motor Drag

Speed at which the wheel will decelerate when not accelerating.

Handbrake Enabled

Whether to allow the wheel to use the handbrake.

Show Colliders

By default the collider objects and joints are hidden in the inspector. You can make them visible with this option.

Scripts Reference

TCCAPlayer

Initializes and controls the behavior of the wheels and body, and includes functions to set and return the position and rotation of the vehicle, check the grounding state, velocities, etc.

`public TCCABody getCarBody()`

Returns the associated TCCABody component.

`public Rigidbody getRigidbody()`

Returns the Rigidbody component of the TCCABody object.

`public void setMotor(float d)`

Sets the torque applied to each child wheel that has its motor enabled. Takes any value between -1 and 1.

`public float getMotor()`

Returns the torque currently applied to the wheels.

`public void setSteering(float d)`

Sets the steering direction to each child wheel that has its steering enabled. -1 to turn left, 1 to turn right.

`public float getSteering()`

Returns the steering direction currently applied to the wheels.

`public void setHandbrake(bool e)`

Sets whether the wheels torque should be locked.

`public bool getHandbrake()`

Returns whether the wheels are currently locked.

`public void setBoost(float d)`

How much boost to apply to the wheels. 1 is full boost, 0 is no boost.

`public float getBoost()`

Returns the current amount of boosting applied to the vehicle.

`public float getWheelsMaxSpin(int direction = 0)`

Returns the maximum torque of the wheels on this vehicle. "direction" sets whether to only count the torque of a wheel if it's going in a certain direction. 1 is forward, -1 is back, 0 to count both forward and back rotations.

public float getWheelsMaxSpeed()

Returns the top max speed of the wheels on this vehicle.

public float getPitchAngle()

Returns the current pitch (x axis in degrees) of the vehicle's body.

public float getRollAngle()

Returns the current roll (z axis in degrees) of the vehicle's body.

public float getForwardVelocity()

Returns the current velocity of the vehicle relative to its forward direction.

public float getForwardVelocityDelta()

Returns the current velocity of the vehicle relative to its forward direction, divided by the max speed of the wheels.

public float getLateralVelocity()

Returns the current velocity of the vehicle relative to its right direction.

public float getVerticalVelocity()

Returns the current velocity of the vehicle relative to its up direction.

public bool isPartiallyGrounded()

Returns whether at least one wheel is grounded but not all of them.

public bool isGrounded()

Returns whether at least one wheel is grounded.

public bool isFullyGrounded()

Returns whether all wheels are grounded.

public void immobilize()

Sets all velocities and torques to zero.

public void translate(Vector3 position)

Adds the value of "position" to the current position of the vehicle.

public void rotate(Quaternion rotation)

Adds the value of "rotation" to the current rotation of the vehicle.

public void recenter()

If the root GameObject is not at position zero, resets it.

public void setPosition(Vector3 position)

Sets the position of the vehicle to "position".

public void setRotation(Quaternion rotation)

Sets the rotation of the vehicle to "rotation".

public Vector3 getPosition()

Returns the current position of the vehicle in world space.

public Quaternion getRotation()

Returns the current rotation of the vehicle in world space.

public Vector3 getInitialPosition()

Returns the position of the vehicle where it was first instantiated.

public Quaternion getInitialRotation()

Returns the rotation of the vehicle when it was first instantiated.

public void setParent(Transform parent)

Sets the Transform in which the vehicle will be contained.

TCCABody

Controls the main body and immplements roll-countering forces. Adds a "TCCABodyCollider" component to this GameObject to control collision detection.

public virtual void onCollisionStay(Collision collision)

public virtual void onCollisionEnter(Collision collision)

public virtual void onCollisionExit(Collision collision)

public virtual void onTriggerStay(Collider other)

public virtual void onTriggerEnter(Collider other)

public virtual void onTriggerExit(Collider other)

Called from the functions of the same names in TCCABodyCollider. You can create a custom class that extends TCCABody and override the above functions to detect collisions and triggers.

public void initialize(TCCAPlayer parent)

Initializes the vehicle body with the specified TCCAPlayer component.

public void refresh(float deltaTime)

Updates the parameters and physics of the vehicle body. Called once every frame in Update().

public float getPitchAngle()

Returns the current pitch (x axis in degrees) of the vehicle's body.

public float getRollAngle()

Returns the current roll (z axis in degrees) of the vehicle's body.

public bool canCounterRotation(CounterMode m)

Returns whether the vehicle will currently try to balance itself with the specified rotation countering mode.

public Vector3 getForwardAngularVelocity()

Returns the rotational velocity of the vehicle represented in radians per second.

public void setForwardAngularVelocity(Vector3 v)

Sets the rotational velocity of the vehicle represented in radians per second.

public float getForwardVelocity()

Returns the current velocity of the body relative to its forward direction.

public float getLateralVelocity()

Returns the current velocity of the body relative to its right direction.

public float getVerticalVelocity()

Returns the current velocity of the vehicle relative to its up direction.

public TCCAPlayer getParentPlayer()

Returns the TCCAPlayer component connected to this body.

public Vector3 getPosition()

Returns the current position of the body in world space.

public Quaternion getRotation()

Returns the current rotation of the body in world space.

public void setPosition(Vector3 position)

Sets the current position of the body in world space.

public void setRotation(Quaternion rotation)

Sets the current rotation of the body in world space.

public void translate(Vector3 offset)

Adds the value of "position" to the current position of the body.

public void rotate(Quaternion rotation)

Adds the value of "rotation" to the current rotation of the body.

public void immobilize()

Sets all velocities and torques to zero.

public void setParent(Transform parent)

Sets the Transform in which this body should be contained. This should not be called by the user in normal circumstances.

TCCAWheel

Creates the necessary objects and components to generate a wheel with steering and suspension. The new objects are instantiated in the object containing the TCCAPlayer script.

public virtual void onCollisionStay(Collision collision)

public virtual void onCollisionEnter(Collision collision)

public virtual void onCollisionExit(Collision collision)

public virtual void onTriggerStay(Collider other)

public virtual void onTriggerEnter(Collider other)

public virtual void onTriggerExit(Collider other)

Called from the functions of the same names in TCCAWheelCollider. You can create a custom class that extends TCCAWheel and override the above functions to detect collisions and triggers.

public void initialize(TCCAPlayer parent)

Initializes the vehicle body with the specified TCCAPlayer component.

public void refresh(float deltaTime)

Updates the parameters and physics of the vehicle wheel. Called once every frame in Update().

public TCCAPlayer getParentPlayer()

Returns the TCCAPlayer component connected to this wheel.

public void setSpeedMultiplier(float m)

The value by which to multiply the wheel's max speed. Default is 1.

public void setAccelerationMultiplier(float m)

The value by which to multiply the wheel's acceleration. Default is 1.

public SphereCollider getCollider()

Returns the wheel's SphereCollider component.

public bool isTouchingGround()

Returns whether the wheel is currently touching the ground.

public void setSteering(float value)

Sets the steering direction of the wheel if it has its steering enabled. -1 to turn left, 1 to turn right.

public void setMotor(float value)

Sets the torque applied to the wheel if it has its motor enabled. Takes any value between -1 and 1.

public void setHandbrake(bool e)

Sets whether the wheel's torque should be locked.

public float getSteering()

Returns the steering direction currently applied to the wheel.

public float getMotor()

Returns the torque currently applied to the wheel.

public Vector3 getForwardAngularVelocity()

Returns the rotational velocity of the wheel represented in radians per second.

public void setForwardAngularVelocity(Vector3 v)

Sets the rotational velocity of the wheel represented in radians per second.

public float getForwardSpinVelocity()

Returns the calculated speed of the wheel in meters per second according to its torque.

public void setForwardSpinVelocity(float v)

Sets the torque of the wheel according to the desired velocity in meters per second.

public float getForwardVelocity()

Returns the current velocity of the wheel relative to its forward direction.

public float getLateralVelocity()

Returns the current velocity of the wheel relative to its right direction.

`public float getVerticalVelocity()`

Returns the current velocity of the wheel relative to its up direction.

`public Vector3 getRelativeVelocity()`

Returns the current velocity of the wheel in local coordinates.

`public void immobilize()`

Sets all velocities and torques to zero.

`public Vector3 getPosition()`

Returns the current position of the wheel in world space.

`public Quaternion getRotation()`

Returns the current rotation of the wheel in world space.

`public void setParent(Transform parent)`

Sets the Transform in which the wheel will be contained. This should not be called by the user in normal circumstances.

Troubleshooting

Can I use different colliders for the wheels than the default sphere?

Unfortunately, this controller has been designed to use a specific combination of rigidbodies, colliders, and joints for the wheels, so you can't use custom colliders. However, you can use any type of collider for the car body, and the wheels won't collide with it, so if you wanted for instance to have the wheels only touch the ground and not the sides of another object, you could create a collider on the car body that intersect the wheels.

How can I detect when a wheel/body collides with a custom object?

You can create a script that extends the TCCAWheel or TCCABody script and use that instead. Then you can override the following collision functions, which are called at the same time as their Rigidbody counterparts;

- public virtual void onCollisionStay(Collision collision) { }
- public virtual void onCollisionEnter(Collision collision) { }
- public virtual void onCollisionExit(Collision collision) { }
- public virtual void onTriggerStay(Collider other) { }
- public virtual void onTriggerEnter(Collider other) { }
- public virtual void onTriggerExit(Collider other) { }

The car jumps a bit when running over seams in the road. How can I fix that?

Try changing the values of the "Default Contact Offset" parameter in the Physics tab of the Project Settings to minimize the effects of "ghost collisions". Ideally, this controller works best on connected meshes.

How do I change the position and rotation of the vehicle?

You can set the position and rotation of the vehicle with the functions "TCCAPlayer.setPosition(Vector3)" and "TCCAPlayer.setRotation(Quaternion)". These will reset the container object to the absolute position/rotation of the vehicle body and move the container to the values passed as parameters.

If you need to reset the vehicle completely, you might want to also call the function "TCCAPlayer.immobilize()", which will set the vehicle's velocities to zero.

You can also use the functions "TCCAPlayer.translate(Vector3)" and "TCCAPlayer.rotate(Quaternion)" if you just want to add values to the current position and rotation.

When steering the vehicle, the car itself turns slightly left and right.

Try raising the mass of the body and wheels until the turning is hardly noticeable.

The wheels are not staying perfectly in place when going too fast.

Try raising the stiffness (spring) of the suspension until the jitter is hardly noticeable.