Drive In VR

1.2.6

Generated by Doxygen 1.9.5

Chapter 1

Namespace Index

1.1 Package List

Here are the packages with brief descriptions (if available):

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

Namespace Documentation

4.1 VRDriving Namespace Reference

Classes

- class ControllerInfo

  A class that holds information about a controller.
- struct FloatMinMax
- class **InterfaceHelper**

  A static class that allows components with interfaces to be retrieved.

Enumerations

- enum ControllerSide

  An enumerate representing a controller by it's handedness. (ControllerSide.Left for left hand controllers, Controller↩ Side.Right for right hand controllers.)

4.1.1 Enumeration Type Documentation

4.1.1.1 ControllerSide

`enum VRDriving.ControllerSide`

An enumerate representing a controller by it's handedness. (ControllerSide.Left for left hand controllers, Controller↩ Side.Right for right hand controllers.)

Author: Mathew Aloisio

4.2 VRDriving.Events Namespace Reference

Classes

- class ControllerUnityEvent

  Arg0: ControllerInfo - The controller information about the controller associated with the event.
- class FloatUnityEvent
- class SteeringControllerUnityEvent

  An event that involves a VehicleGrabbableSteeringBase component and a controller.
- class SteeringUnityEvent

  An event that involves a VehicleGrabbableSteeringBase component.

4.3 VRDriving.Grabbing Namespace Reference

Classes

- interface IGrabbable

  The abstract base class for all grabbable types.

4.4 VRDriving.Math Namespace Reference

Classes

- class **FloatMath**
- class **VectorMath**

4.5 VRDriving.Steering Namespace Reference

Classes

- class VehicleGrabbableSteeringBase

  An abstract class that is the base of all grabbable steering mechanisms.
- class VehicleHandRelativeHandlebarSteering

  A component that should be attached to the vehicle's steering grabbable that sets a handlebar'd vehicle's steering angle based on hand/controller positions. One-handed steering uses the direction of the line perpendicular to the line between the driving hand and the handlebar's rotation pivot. Two-handed steering uses the direction of the line perpendicular to the line between each hands on the handlebar's rotation plane.
- class VehicleHandRelativeSteering

  A component that should be attached to the vehicle's steering grabbable that sets a vehicle's steering angle based on hand/controller rotations.
- class VehicleJointAngleSteering

  A component that sets a vehicle's steering angle based on a joint's angle.
- class VehicleSteeringBase

  An abstract class that is the base of all steering mechanisms.
- class VehicleWheelchairSteering

  A component that uses two VehicleOneHandWheelSteering components to make wheelchair style steering. Implements wheelchair steering and acceleration/deceleration input controls based on:

4.6 VRDriving.Wheels Namespace Reference

Classes

- class OneHandWheel

  A component that can be used to allow a user to accelerate, brake, or accelerate in reverse by spinning a wheel.

Chapter 5

Class Documentation

5.1 VRDriving.ControllerInfo Class Reference

A class that holds information about a controller.

Public Attributes

- Transform **transform**

  The Transform of the controller.
- ControllerSide **side**

  The side

5.1.1 Detailed Description

A class that holds information about a controller.

Author: Mathew Aloisio

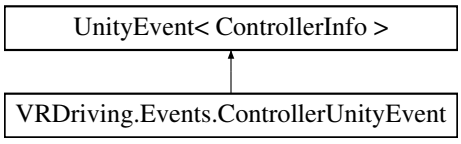
The documentation for this class was generated from the following file:

- ControllerInfo.cs

5.2 VRDriving.Events.ControllerUnityEvent Class Reference

Arg0: ControllerInfo - The controller information about the controller associated with the event.

Inheritance diagram for VRDriving.Events.ControllerUnityEvent:



5.2.1 Detailed Description

Arg0: ControllerInfo - The controller information about the controller associated with the event.

The documentation for this class was generated from the following file:

- ControllerUnityEvent.cs

5.3 VRDriving.FloatMinMax Struct Reference

Public Attributes

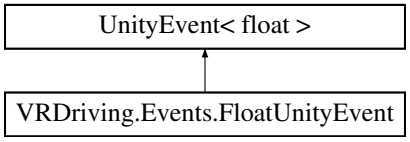
- float **minimum**
- float **maximum**

The documentation for this struct was generated from the following file:

- FloatMinMax.cs

5.4 VRDriving.Events.FloatUnityEvent Class Reference

Inheritance diagram for VRDriving.Events.FloatUnityEvent:



The documentation for this class was generated from the following file:

- FloatUnityEvent.cs

5.5 VRDriving.Steering.VehicleHandRelativeHandlebarSteering.↩ HandlebarController Class Reference

Stores reference to a controller that is grabbing the handlebars currently and some information about where it is grabbing the handlebars.

Public Attributes

- ControllerInfo **controller**

  The ControllerInfo associated with the handlebar controller.
- float **handlebarRotationMultiplier**

  A multiplier used to determine which direction the handlebars rotate based on controller position changes.

5.5.1 Detailed Description

Stores reference to a controller that is grabbing the handlebars currently and some information about where it is grabbing the handlebars.

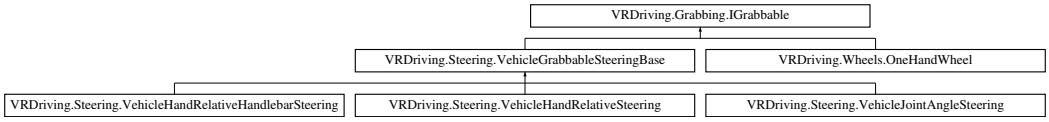
The documentation for this class was generated from the following file:

- VehicleHandRelativeHandlebarSteering.cs

5.6 VRDriving.Grabbing.IGrabbable Interface Reference

The abstract base class for all grabbable types.

Inheritance diagram for VRDriving.Grabbing.IGrabbable:



Public Member Functions

- abstract void OnGrabbed (Transform pControllerTransform, ControllerSide pControllerSide)

  A callback intended to be invoked when a controller grabs the wheel.
- abstract void OnReleased (Transform pControllerTransform, ControllerSide pControllerSide)

  A callback intended to be invoked when a controller releases the wheel.

5.6.1 Detailed Description

The abstract base class for all grabbable types.

Author: Mathew Aloisio

5.6.2 Member Function Documentation

5.6.2.1 OnGrabbed()

```
abstract void VRDriving.Grabbing.IGrabbable.OnGrabbed (
            Transform pControllerTransform,
            ControllerSide pControllerSide )  [pure virtual]
```

A callback intended to be invoked when a controller grabs the wheel.

Parameters

pControllerTransform	The Transform of the controller that grabbed the wheel.
pControllerSide	The ControllerSide for the controller that grabbed the wheel.

Implemented in VRDriving.Steering.VehicleGrabbableSteeringBase, and VRDriving.Wheels.OneHandWheel.

5.6.2.2 OnReleased()

```
abstract void VRDriving.Grabbing.IGrabbable.OnReleased (
            Transform pControllerTransform,
            ControllerSide pControllerSide )  [pure virtual]
```

A callback intended to be invoked when a controller releases the wheel.

Parameters

pControllerTransform	The Transform of the controller that grabbed the wheel.
pControllerSide	The ControllerSide for the controller that grabbed the wheel.

Implemented in VRDriving.Steering.VehicleGrabbableSteeringBase, and VRDriving.Wheels.OneHandWheel.

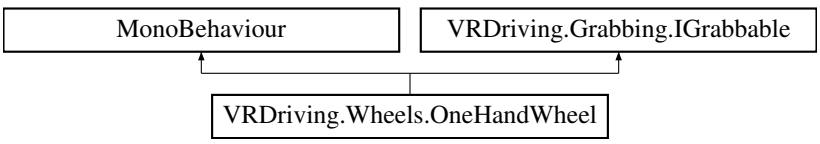
The documentation for this interface was generated from the following file:

- IGrabbable.cs

5.7 VRDriving.Wheels.OneHandWheel Class Reference

A component that can be used to allow a user to accelerate, brake, or accelerate in reverse by spinning a wheel.

Inheritance diagram for VRDriving.Wheels.OneHandWheel:



Classes

- struct VelocityEntry
- class WheelController

  Stores reference to a controller that is grabbing a wheel and what wheel side was grabbed.

Public Member Functions

- void SimulateMomentum (Vector3 pVelocity, float pRPM)

  May be invoked (every frame) to simulate the effect(s) of momentum on the 'one hand wheel' controller.
- bool CheckIfBrakingConditionsMet ()

  Checks if the conditions for braking of this wheel are met, returns true if met, otherwise false. Also updates 'Last↩ BrakingConditionsMetTime' and 'LastBrakingConditionsNotMetTime'.
- void OnGrabbed (Transform pControllerTransform, ControllerSide pControllerSide)

  A callback intended to be invoked when a controller grabs the wheel.
- void OnReleased (Transform pControllerTransform, ControllerSide pControllerSide)

  A callback intended to be invoked when a controller releases the wheel.

Public Attributes

- float **acceleration**

  The current acceleration input value for this component.
- float **accelerationScalar** = 8f

  The acceleration scalar that is used to multiply the AverageControllerLocalForwardSpeed when accelerating.
- float **decelerationRate** = 0.025f

  The rate at which the
- float **brakeDelay** = 0.4f

  The number of seconds you have to hold a wheel with the controller
- float **brakeRate** = 1f

  The rate at which the
- float **controllerSpeedThreshold** = 0.006f

  The minimum controller absolute forward speed threshold before acceleration input will be applied.
- float **controllerVelocityHistoryTime** = 0.4f

  The
- Transform **wheelTransform**

  The Transform that defines the orientation of the wheel. Movement in the wheelTransforms forward direction is used to determine acceleration.
- Vector3 **wheelForward** = Vector3.forward

  The direction of the positive forward axis of the wheel in wheelTransform
- ControllerUnityEvent **ControllerGrabbed**

  An event that is invoked whenever a controller grabs this wheel.
  \nArg
- ControllerUnityEvent **ControllerReleased**

  An event that is invoked whenever a controller releases this wheel.
  \nArg

Properties

- bool **IsBraking** `[get]`

  Returns true while braking the wheel, otherwise false.
- float **Braking** `[get]`

  Starting at 0 approaches 1 at 'brakeRate' units per second while braking.
- WheelController **GrabbedBy** `[get]`

  A reference to the WheelController currently grabbing this component, otherwise null.
- bool **GrabbedThisFrame** `[get]`

  Returns true if this component was grabbed this frame, otherwise false.

- Vector3 **LastControllerLocalPosition** [get]

  The local space position of the controller last frame (or the first frame of a grab).
- float **LastGrabTime** [get]

  Returns the last Time.time this component was grabbed.
- float **LastReleaseTime** [get]

  Returns the last Time.time this component was released.
- Vector3 **AverageControllerLocalVelocity** [get]

  Returns the average velocity of the controller currently grabbing the wheel in local space, otherwise Vector3.zero.
- float **AverageControllerLocalForwardSpeed** [get]

  Returns the average speed of the controller in the WheelForwardDirection in local space.
- float **LastBrakingConditionsMetTime** [get]

  Returns the last Time.time that the braking conditions were met for this wheel.
- float **LastBrakingConditionsNotMetTime** [get]

  Returns the last Time.time that the braking conditions were not met for this wheel.
- Vector3 **WheelForwardDirection** [get]

  Returns the wheel's forward direction in world space.

5.7.1 Detailed Description

A component that can be used to allow a user to accelerate, brake, or accelerate in reverse by spinning a wheel.

- The 'acceleration' value is from -1 to 1, it always moves towards 0 at a constant rate.
- The 'acceleration' value goes up when 'spinning wheel' forward, either wheel.
- The 'acceleration' value goes down when 'spinning wheel' backward, either wheel.
- Holding the wheel causes the 'acceleration' input to be set to 0 instantly.
- Holding the wheel instantly causing braking to start and the 'brake' input will approach 1 at brakeRate units per second.
- After releasing the wheel the 'brake' input will be immediately zero'd.
- To add acceleration the controller moves into the wheel's trigger, moves, then leaves the wheel's trigger and the acceleration (in either direction) is determined by the controller's average velocity in the wheel's forward direction.
- Acceleration in either direction will only be applied if the controller's absolute average forward speed is above the controllerSpeedThreshold on release.
- Momentum is considered when determing deceleration rate, this makes the users wheel inputs less effective when on hills, etc.
- To brake the user grabs the wheel.

AVERAGE MEASURED RELEASE FORWARD SPEED BY ONTROLLER

5.7.1.1 CONTROLLER | SPEED RANGE MEASURED (units/sec)

Valve Index | 0.0069 to 0.0115

NOTE: Only ONE controller can grab this type of component at a time. Author: Mathew Aloisio

5.7.2 Member Function Documentation

5.7.2.1 CheckIfBrakingConditionsMet()

```
bool VRDriving.Wheels.OneHandWheel.CheckIfBrakingConditionsMet ( )
```

Checks if the conditions for braking of this wheel are met, returns true if met, otherwise false. Also updates 'Last↩ BrakingConditionsMetTime' and 'LastBrakingConditionsNotMetTime'.

Returns

true if braking conditions for this wheel are met, otherwise false.

5.7.2.2 OnGrabbed()

```
void VRDriving.Wheels.OneHandWheel.OnGrabbed (
            Transform pControllerTransform,
            ControllerSide pControllerSide )  [virtual]
```

A callback intended to be invoked when a controller grabs the wheel.

Parameters

pControllerTransform	The Transform of the controller that grabbed the wheel.
pControllerSide	The ControllerSide for the controller that grabbed the wheel.

Implements VRDriving.Grabbing.IGrabbable.

5.7.2.3 OnReleased()

```
void VRDriving.Wheels.OneHandWheel.OnReleased (
            Transform pControllerTransform,
            ControllerSide pControllerSide )  [virtual]
```

A callback intended to be invoked when a controller releases the wheel.

Parameters

pControllerTransform	The Transform of the controller that grabbed the wheel.
pControllerSide	The ControllerSide for the controller that grabbed the wheel.

Implements VRDriving.Grabbing.IGrabbable.

5.7.2.4 SimulateMomentum()

```
void VRDriving.Wheels.OneHandWheel.SimulateMomentum (
            Vector3 pVelocity,
            float pRPM )
```

May be invoked (every frame) to simulate the effect(s) of momentum on the 'one hand wheel' controller.

Parameters

pVelocity	The velocity of the wheel.
pRPM	The RPM the wheel is rotating at.

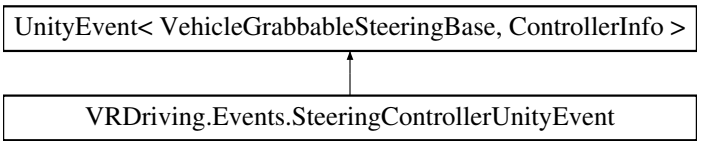
The documentation for this class was generated from the following file:

- OneHandWheel.cs

5.8 VRDriving.Events.SteeringControllerUnityEvent Class Reference

An event that involves a VehicleGrabbableSteeringBase component and a controller.

Inheritance diagram for VRDriving.Events.SteeringControllerUnityEvent:



5.8.1 Detailed Description

An event that involves a VehicleGrabbableSteeringBase component and a controller.

Arg0: VehicleGrabbableSteeringBase - The component involved in the event. Arg1: ControllerInfo - The info about the controller involved in the event.

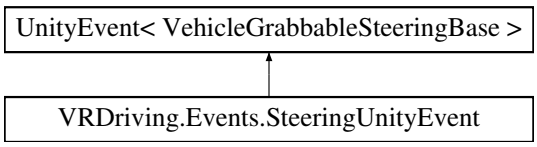
The documentation for this class was generated from the following file:

- SteeringUnityEvents.cs

5.9 VRDriving.Events.SteeringUnityEvent Class Reference

An event that involves a VehicleGrabbableSteeringBase component.

Inheritance diagram for VRDriving.Events.SteeringUnityEvent:



5.9.1 Detailed Description

An event that involves a VehicleGrabbableSteeringBase component.

Arg0: VehicleGrabbableSteeringBase - The component involved in the event.

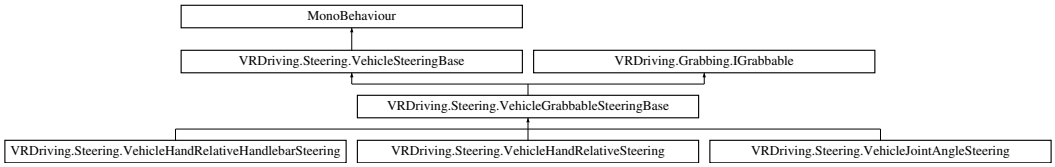
The documentation for this class was generated from the following file:

- SteeringUnityEvents.cs

5.10 VRDriving.Steering.VehicleGrabbableSteeringBase Class Reference

An abstract class that is the base of all grabbable steering mechanisms.

Inheritance diagram for VRDriving.Steering.VehicleGrabbableSteeringBase:



Public Member Functions

- ControllerInfo GetControllerInfo (int pIndex)

  Returns the ControllerInfo at the specified index in the 'grabbing controllers' array.
- void OnGrabbed (Transform pControllerTransform, ControllerSide pControllerSide)

  A callback intended to be invoked when a controller grabs the steering mechanism.
- void OnReleased (Transform pControllerTransform, ControllerSide pControllerSide)

  A callback intended to be invoked when a controller releases the steering mechanism.

Public Attributes

- SteeringControllerUnityEvent **ControllerGrabbed** = new SteeringControllerUnityEvent()

  An event that is invoked when a controller grabs the steering mechanism.
  \nArg
- SteeringControllerUnityEvent **ControllerReleased** = new SteeringControllerUnityEvent()

  An event that is invoked when a controller releases the steering mechanism.
  \nArg
- SteeringControllerUnityEvent **LastControllerReleased** = new SteeringControllerUnityEvent()

  An event that is invoked when the last controller releases the steering mechanism.
  \nArg

Protected Member Functions

- virtual void **Awake** ()
- void OnLastControllerReleased (ControllerInfo pController)

  Invoked after the last controller releases the steering mechanism.

Properties

- int **GrabbingControllersCount** `[get]`

  Returns an integer representing the number of 'ControllerInfo' entries in the 'grabbing controllers' array. Use Vehicle↩ KinematicSteeringBase.GetControllerInfo(int pIndex) to return the ControllerInfo for a specified index.
- float **LastControllerReleasedTime** `[get, set]`

  The Time.time of the last invokation to 'OnLastControllerReleased'.

5.10.1 Detailed Description

An abstract class that is the base of all grabbable steering mechanisms.

Author: Mathew Aloisio

5.10.2 Member Function Documentation

5.10.2.1 GetControllerInfo()

```
ControllerInfo VRDriving.Steering.VehicleGrabbableSteeringBase.GetControllerInfo (
            int pIndex )
```

Returns the ControllerInfo at the specified index in the 'grabbing controllers' array.

Parameters

pIndex	

Returns

the ControllerInfo at the specified index in the 'grabbing controllers' array.

5.10.2.2 OnGrabbed()

```
void VRDriving.Steering.VehicleGrabbableSteeringBase.OnGrabbed (
            Transform pControllerTransform,
            ControllerSide pControllerSide )  [virtual]
```

A callback intended to be invoked when a controller grabs the steering mechanism.

Parameters

pControllerTransform	The Transform of the controller that grabbed the steering mechanism.
pControllerSide	The ControllerSide for the controller that grabbed the steering mechanism.

Implements VRDriving.Grabbing.IGrabbable.

5.10.2.3 OnLastControllerReleased()

```
void VRDriving.Steering.VehicleGrabbableSteeringBase.OnLastControllerReleased (
            ControllerInfo pController )  [protected]
```

Invoked after the last controller releases the steering mechanism.

Parameters

pController	

5.10.2.4 OnReleased()

```
void VRDriving.Steering.VehicleGrabbableSteeringBase.OnReleased (
            Transform pControllerTransform,
            ControllerSide pControllerSide )  [virtual]
```

A callback intended to be invoked when a controller releases the steering mechanism.

Parameters

pControllerTransform	The Transform of the controller that grabbed the steering mechanism.
pControllerSide	The ControllerSide for the controller that grabbed the steering mechanism.

Implements VRDriving.Grabbing.IGrabbable.

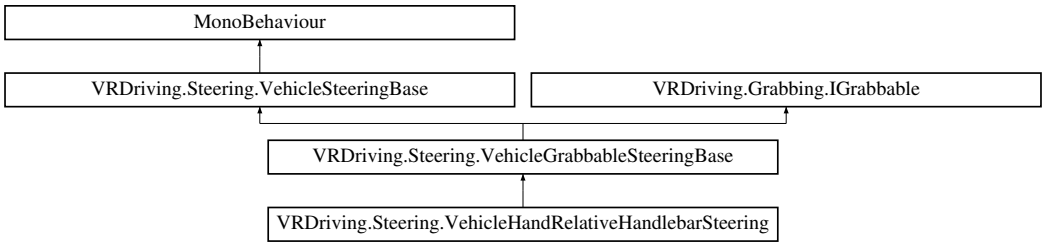
The documentation for this class was generated from the following file:

- VehicleGrabbableSteeringBase.cs

5.11 VRDriving.Steering.VehicleHandRelativeHandlebarSteering Class Reference

A component that should be attached to the vehicle's steering grabbable that sets a handlebar'd vehicle's steering angle based on hand/controller positions. One-handed steering uses the direction of the line perpendicular to the line between the driving hand and the handlebar's rotation pivot. Two-handed steering uses the direction of the line perpendicular to the line between each hands on the handlebar's rotation plane.

Inheritance diagram for VRDriving.Steering.VehicleHandRelativeHandlebarSteering:



Classes

- class HandlebarController

  Stores reference to a controller that is grabbing the handlebars currently and some information about where it is grabbing the handlebars.

Public Types

- enum **HandlebarLimit**

Public Attributes

- bool **debug**

  Enable debugging?
- float **sensitivity** = 1f

  Controls the sensitivity of the steering. [
- bool **invertSteeringAngle**

  Should the steering angle value be inverted?
- FloatMinMax **handlebarAngleRange** = new() { minimum = -90f, maximum = 90f }

  The angle range the handlebars are allowed to remain within.
- Vector3 **handlebarRelativeHandMoveDirection** = new(0f, 0f, 1f)

  The direction to look for changes in hand positions in relative to the handlebar.
- Vector3 **handlebarHorizontalDirection** = new(1f, 0f, 0f)

The direction of the positive horizontal axis of the handlebars.

- Vector3 **handlebarRotationAxis** = new(0f, 1f, 0f)

  The rotation axis for the handlebars.
- float **handlebarMaxReturnSpeed** = 60f

  The maximum speed with which the steering wheel can rotate per second to return to the center position.
- float **handlebarMaxReturnSpeedVelocity** = 7f

  The forward or backward velocity of the vehicle at which the maximum steering wheel return speed will be reached.
- float **handlebarReturnVehicleVelocityThreshold** = 1f

  The minimum forward or backward velocity of the vehicle required to start re
- AnimationCurve **handlebarTimeReturnCurve** = AnimationCurve.Linear(0, 1f, 1f, 1f)

  A simple curve that plots the relationship of time since last grabber release on the x axis
- AnimationCurve **handlebarAngleReturnCurve** = AnimationCurve.Linear(0, 1f, 1f, 1f)

  A simple curve that plots the relationship of
- Transform **handlebarTransform**

  The transform to use when modifying the rotation of the steering wheel.

Protected Attributes

- Vector3 **m_DefaultHandlebarLocalEulerAngles**

  The default local euler angles of the handlebar transform.
- Vector3 **m_DefaultHandlebarFixedForwardDirection**

  The default fixed forward direction of the handlebar transform.

Properties

- float **HandlebarAngle** `[get, protected set]`

  A running total for the handlebar's angle.

Additional Inherited Members

5.11.1 Detailed Description

A component that should be attached to the vehicle's steering grabbable that sets a handlebar'd vehicle's steering angle based on hand/controller positions. One-handed steering uses the direction of the line perpendicular to the line between the driving hand and the handlebar's rotation pivot. Two-handed steering uses the direction of the line perpendicular to the line between each hands on the handlebar's rotation plane.

Author: Mathew Aloisio

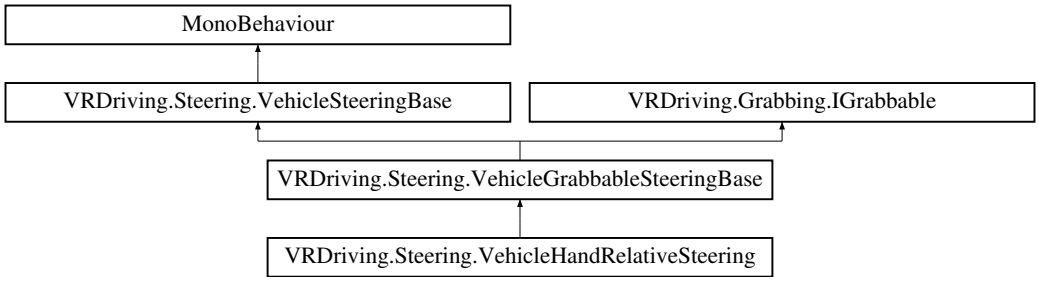
The documentation for this class was generated from the following file:

- VehicleHandRelativeHandlebarSteering.cs

5.12 VRDriving.Steering.VehicleHandRelativeSteering Class Reference

A component that should be attached to the vehicle's steering grabbable that sets a vehicle's steering angle based on hand/controller rotations.

Inheritance diagram for VRDriving.Steering.VehicleHandRelativeSteering:



Public Types

- enum **SteeringMode**
- enum **SteeringWheelLimit**

Public Member Functions

- float GetControllerRotation (Transform pControllerTransform)

  Returns the rotation around the controllerRotationAxis for the controller with the given Transform.
- float GetControllerCenterDistance (Transform pControllerTransform)

  Returns the projected center distance of the controller from the steering mechanism center.

Public Attributes

- bool **invertSteeringAngle**

  Should the steering angle value be inverted?
- SteeringMode **steeringMode** = SteeringMode.Delta

  The steering mode to use.
  \nOffset
- float **centerDistanceThreshold**

  The center distance threshold
- FloatMinMax **steeringWheelAngleRange** = new FloatMinMax() { minimum = -90f, maximum = 90f }

  The angle range the steering wheel is allowed to remain within.
- Vector3 **steeringWheelUpAxis** = Vector3.up

  The up axis for the steering wheel. Must be the same for
- Vector3 **steeringWheelRotationAxis** = Vector3.forward

  The rotation axis for the steering wheel. Must be the same for
- float **steeringWheelMaxReturnSpeed** = 60f

  The maximum speed with which the steering wheel can rotate per second to return to the center position.
- float **steeringWheelMaxReturnSpeedVelocity** = 7f

  The forward or backward velocity of the vehicle at which the maximum steering wheel return speed will be reached.
- float **steeringWheelReturnVehicleVelocityThreshold** = 1f

  The minimum forward or backward velocity of the vehicle required to start re

- AnimationCurve **steeringTimeReturnCurve** = AnimationCurve.Linear(0, 1f, 1f, 1f)

  A simple curve that plots the relationship of time since last grabber release on the x axis
- AnimationCurve **steeringAngleReturnCurve** = AnimationCurve.Linear(0, 1f, 1f, 1f)

  A simple curve that plots the relationship of
- Transform **steeringWheelTransform**

  The transform to use when modifying the rotation of the steering wheel.

Properties

- float **SteeringWheelAngle** `[get, protected set]`

  A running total for the steering wheel's angle.
- Transform **LeftController** `[get, protected set]`

  The left hand currently holding the steering wheel or null.
- float **LeftControllerCumulativeRotation** `[get, protected set]`

  The cumulative rotation of the left controller since grab.
- float **LeftControllerGrabSteeringAngle** `[get, protected set]`

  The value of SteeringWheelAngle at the time the left controller grabbed.
- float **LeftControllerLastAngle** `[get, protected set]`

  The angle around the controller rotation axis of the left controller the last frame the steering wheel was grabbed with the left hand.
- float **LeftControllerDeltaDebt** `[get, protected set]`

  Tracks the accumulated 'debt' of the left controller past the steering limit when in delta steering mode.
- float **LastLeftControllerDeltaAngle** `[get, protected set]`

  The last delta angle for the left controller currently grabbing the steering mechanism.
- Transform **RightController** `[get, protected set]`

  The right hand currently holding the steering wheel or null.
- float **RightControllerCumulativeRotation** `[get, protected set]`

  The cumulative rotation of the right controller since grab.
- float **RightControllerGrabSteeringAngle** `[get, protected set]`

  The value of SteeringWheelAngle at the time the right controller grabbed.
- float **RightControllerLastAngle** `[get, protected set]`

  The angle around the controller rotation axis of the right controller the last frame the steering wheel was grabbed with the right hand.
- float **RightControllerDeltaDebt** `[get, protected set]`

  Tracks the accumulated 'debt' of the right controller past the steering limit when in delta steering mode.
- float **LastRightControllerDeltaAngle** `[get, protected set]`

  The last delta angle for the left controller currently grabbing the steering mechanism.

Additional Inherited Members

5.12.1 Detailed Description

A component that should be attached to the vehicle's steering grabbable that sets a vehicle's steering angle based on hand/controller rotations.

Author: Mathew Aloisio

5.12.2 Member Function Documentation

5.12.2.1 GetControllerCenterDistance()

```
float VRDriving.Steering.VehicleHandRelativeSteering.GetControllerCenterDistance (
            Transform pControllerTransform )
```

Returns the projected center distance of the controller from the steering mechanism center.

Parameters

pControllerTransform	

Returns

a float representing the distance between the projected controller transform from the projected steering mechanism center.

5.12.2.2 GetControllerRotation()

```
float VRDriving.Steering.VehicleHandRelativeSteering.GetControllerRotation (
            Transform pControllerTransform )
```

Returns the rotation around the controllerRotationAxis for the controller with the given Transform.

Parameters

pControllerTransform	

Returns

the rotation around the controllerRotationAxis for the controller with the given Transform.

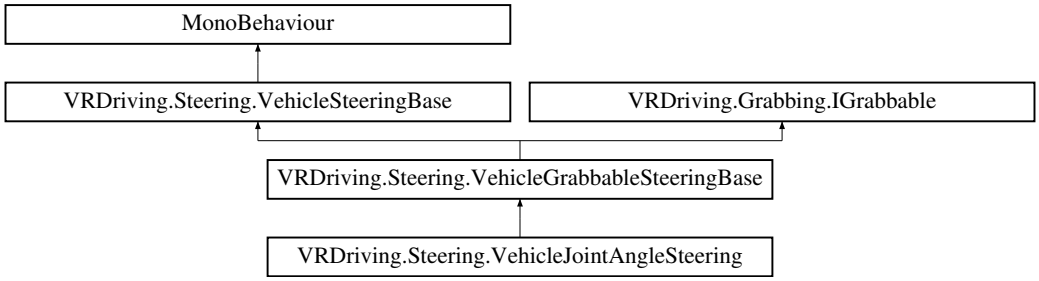
The documentation for this class was generated from the following file:

- VehicleHandRelativeSteering.cs

5.13 VRDriving.Steering.VehicleJointAngleSteering Class Reference

A component that sets a vehicle's steering angle based on a joint's angle.

Inheritance diagram for VRDriving.Steering.VehicleJointAngleSteering:



Public Types

- enum **SteeringWheelLimit**

Public Attributes

- bool **invertSteeringAngle**

  Should the steering angle value be inverted?
- FloatMinMax **steeringWheelAngleRange** = new FloatMinMax() { minimum = -180f, maximum = 180f }

  The angle range the steering wheel is allowed to remain within.
- float **steeringWheelReturnDeadzone** = 0.5f

  The number of degrees of
- float **steeringWheelReturnVelocity** = 30f

  The target velocity for the steering wheel returning to the center position.
- float **steeringWheelReturnForce** = 100f

  The maximum force the joint motor can apply to return the steering wheel to the center position.
- float **steeringWheelMaxReturnSpeedVelocity** = 7f

  The forward velocity of the vehicle at which the maximum steering wheel return speed will be reached.
- float **steeringWheelReturnVehicleVelocityThreshold** = 1f

  The minimum forward or backward velocity of the vehicle required to start re
- HingeJoint **joint**

  The hinge joint to read the reference angle from.

Properties

- float **LastSteeringWheelAngleDifference** `[get, protected set]`

  Returns the difference between the steering wheel angle from this frame to last frame.
- float **SteeringWheelAngle** `[get, protected set]`

  A running total for the steering wheel's angle.

Additional Inherited Members

5.13.1 Detailed Description

A component that sets a vehicle's steering angle based on a joint's angle.

Author: Mathew Aloisio

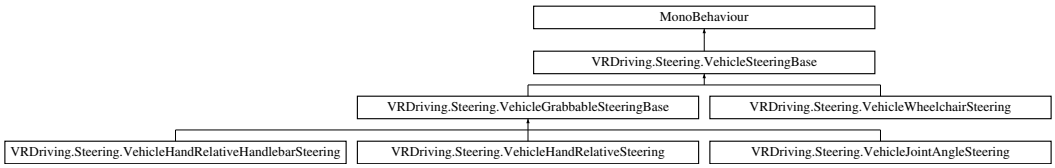
The documentation for this class was generated from the following file:

- VehicleJointAngleSteering.cs

5.14 VRDriving.Steering.VehicleSteeringBase Class Reference

An abstract class that is the base of all steering mechanisms.

Inheritance diagram for VRDriving.Steering.VehicleSteeringBase:



Public Member Functions

- void SetVehicle (Transform pVehicleTransform, Rigidbody pVehicleRigidbody)

  Sets the 'vehicle' associated with this steering component. The Transform refers to the Transform of the vehicle being controlled. The Rigidbody refers to a Rigidbody in the vehicle being controlled that may be used to get local velocity and mass.

Public Attributes

- FloatUnityEvent **SteeringAngleUpdated**

  An event that is invoked each frame that the steering angle is updated.
- Transform **vehicleTransform**

  The Transform of the vehicle that steering is being controlled for.
- Rigidbody **vehicleRigidbody**

Properties

- float **SteeringAngleMultiplier** `[get, protected set]`

  A multiplier to use when setting the steering angle of a vehicle. (Generally steering angle is set by doing max↩ VehicleSteeringAngle ∗ SteeringAngleMultiplier)
- Vector3 **VehicleLocalVelocity** `[get]`

  Returns vehicleTransform.InverseTransformDirection(vehicleRigidbody.velocity)

5.14.1 Detailed Description

An abstract class that is the base of all steering mechanisms.

Author: Mathew Aloisio

5.14.2 Member Function Documentation

5.14.2.1 SetVehicle()

```
void VRDriving.Steering.VehicleSteeringBase.SetVehicle (
            Transform pVehicleTransform,
            Rigidbody pVehicleRigidbody )
```

Sets the 'vehicle' associated with this steering component. The Transform refers to the Transform of the vehicle being controlled. The Rigidbody refers to a Rigidbody in the vehicle being controlled that may be used to get local velocity and mass.

Parameters

pVehicleTransform	
pVehicleRigidbody	

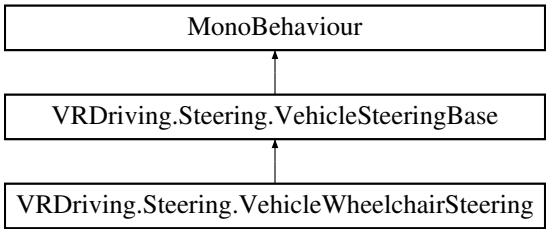
The documentation for this class was generated from the following file:

- VehicleSteeringBase.cs

5.15 VRDriving.Steering.VehicleWheelchairSteering Class Reference

A component that uses two VehicleOneHandWheelSteering components to make wheelchair style steering. Implements wheelchair steering and acceleration/deceleration input controls based on:

Inheritance diagram for VRDriving.Steering.VehicleWheelchairSteering:



Public Member Functions

- virtual float CalculateAcceleration ()

  Calculates the acceleration input for the wheelchair and returns it. (Return value must be between -1f and 1f.)
- void SetAcceleration (float pAcceleration)

  Sets the 'Acceleration' property of this component. Useful for use with Unity editor events. (Range: -1f to 1f) If 'calculateAcceleration' is true the component will automatically recalculate the 'Acceleration' property every frame.
- void SetCalculateAcceleration (bool pCalculate)

  Sets the 'calculateAcceleration' field for this component. Useful for use with Unity editor events.
- void SetCalculateBraking (bool pCalculate)

  Sets the 'calculateBraking' field for this component. Useful for use with Unity editor events.
- void SetCalculateSteering (bool pCalculate)

  Sets the 'calculateSteering' field for this component. Useful for use with Unity editor events.

Public Attributes

- bool **invertSteeringAngle**

  Should the steering angle value be inverted?
- bool **calculateAcceleration** = true

  Enables or disables automatic
- bool **calculateBraking** = true

  Enables or disables automatic
- bool **calculateSteering** = true

Enables or disables automatic

- OneHandWheel **leftSteering**

  A reference to the VehicleOneHandWheelSteering component for the wheelchair
- OneHandWheel **rightSteering**

  A reference to the VehicleOneHandWheelSteering component for the wheelchair
- FloatUnityEvent **AccelerationChanged**

  An event that is invoked everytime the
- FloatUnityEvent **LeftBrakingChanged**

  An event that is invoked everytime the
- FloatUnityEvent **RightBrakingChanged**

  An event that is invoked everytime the

Properties

- float **Acceleration** `[get, set]`

  The acceleration input value. (Range: -1f to 1f) If 'enableAcceleration' is true the component will automatically recalculate the 'Acceleration' property every frame.
- float **SteeringOffset** `[get]`

  Returns a value between -1 and 1 (-1 being left, 1 being right, 0 being neutral) that is calculated using the 'acceleration' input of each wheelchair wheel.
- float **LeftBraking** `[get, set]`

  The 'Braking' input value for the left wheel. (Range -1f to 1f)
- float **RightBraking** `[get, set]`

  The 'Braking' input value for the right wheel. (Range -1f to 1f)

5.15.1 Detailed Description

A component that uses two VehicleOneHandWheelSteering components to make wheelchair style steering. Implements wheelchair steering and acceleration/deceleration input controls based on:

- The 'Acceleration' input is the average of the two VehicleOneHandWheelSteering components associated with each wheel.
- The 'Steering Offset' is a value between -1 and 1 (-1 being left, 1 being right, 0 being neutral), it is calculated using the 'acceleration' input of each VehicleWheelSteering component.
- When the right wheel is spun forward the 'steering offset' moves towards 1, when spun backward the 'steering offset' moves towards -1.
- When the left wheel is spun forward the 'steering offset' moves towards -1, when spun backward the 'steering offset' moves towards 1.
- Holding either wheel under the maxControllerBrakeVelocity for an amount of time over than or equal to the specified 'brakeDelay' causes the 'acceleration' value to approach 0 and causes a turn similar to spinning the wheel backwards. Author: Mathew Aloisio

5.15.2 Member Function Documentation

5.15.2.1 CalculateAcceleration()

```
virtual float VRDriving.Steering.VehicleWheelchairSteering.CalculateAcceleration ( )  [virtual]
```

Calculates the acceleration input for the wheelchair and returns it. (Return value must be between -1f and 1f.)

Returns

a float representing the acceleration input for the wheelchair.

5.15.2.2 SetAcceleration()

```
void VRDriving.Steering.VehicleWheelchairSteering.SetAcceleration (
            float pAcceleration )
```

Sets the 'Acceleration' property of this component. Useful for use with Unity editor events. (Range: -1f to 1f) If 'calculateAcceleration' is true the component will automatically recalculate the 'Acceleration' property every frame.

Parameters

pAcceleration	

5.15.2.3 SetCalculateAcceleration()

```
void VRDriving.Steering.VehicleWheelchairSteering.SetCalculateAcceleration (
            bool pCalculate )
```

Sets the 'calculateAcceleration' field for this component. Useful for use with Unity editor events.

Parameters

pCalculate	

5.15.2.4 SetCalculateBraking()

```
void VRDriving.Steering.VehicleWheelchairSteering.SetCalculateBraking (
            bool pCalculate )
```

Sets the 'calculateBraking' field for this component. Useful for use with Unity editor events.

Parameters

pCalculate	

5.15.2.5 SetCalculateSteering()

```
void VRDriving.Steering.VehicleWheelchairSteering.SetCalculateSteering (
            bool pCalculate )
```

Sets the 'calculateSteering' field for this component. Useful for use with Unity editor events.

Parameters

pCalculate	

The documentation for this class was generated from the following file:

- VehicleWheelchairSteering.cs

5.16 VRDriving.Wheels.OneHandWheel.VelocityEntry Struct Reference

Public Attributes

- Vector3 **velocity**
  The velocity for the VelocityEntry instance.
- float **time**
  The Time.time the VelocityEntry was recorded.

The documentation for this struct was generated from the following file:

- OneHandWheel.cs

5.17 VRDriving.Wheels.OneHandWheel.WheelController Class Reference

Stores reference to a controller that is grabbing a wheel and what wheel side was grabbed.

Public Attributes

- ControllerInfo **controller**
  The ControllerInfo associated with the wheel.

5.17.1 Detailed Description

Stores reference to a controller that is grabbing a wheel and what wheel side was grabbed.

The documentation for this class was generated from the following file:

- OneHandWheel.cs

Index