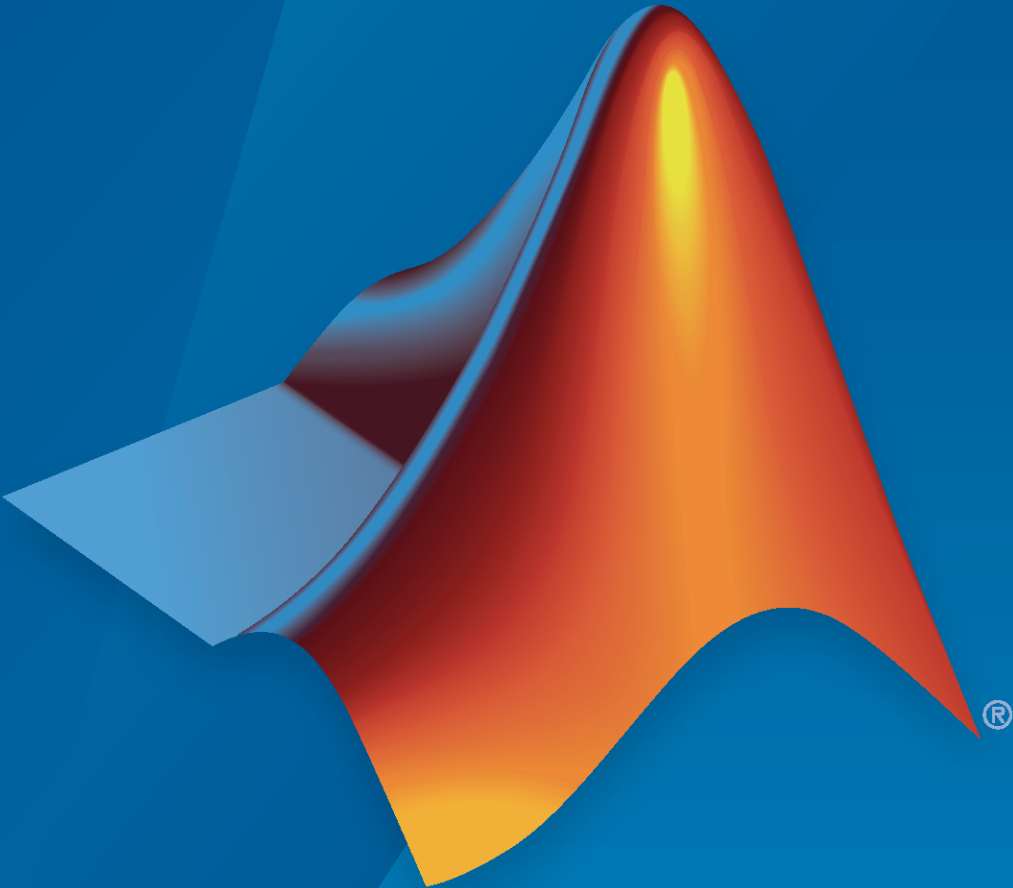


RoadRunner

Reference



How to Contact MathWorks



Latest news: www.mathworks.com

Sales and services: www.mathworks.com/sales_and_services

User community: www.mathworks.com/matlabcentral

Technical support: www.mathworks.com/support/contact_us



Phone: 508-647-7000



The MathWorks, Inc.
1 Apple Hill Drive
Natick, MA 01760-2098

RoadRunner Reference

© COPYRIGHT 2020–2023 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

MathWorks products are protected by one or more U.S. patents. Please see www.mathworks.com/patents for more information.

Revision History

April 2020	Online only	New for Version 1.0 (R2020a)
September 2020	Online only	Revised for Version 1.1 (R2020b)
March 2021	Online only	Revised for Version 1.2 (R2021a)
September 2021	Online only	Revised for Version 1.3 (R2021b)
March 2022	Online only	Revised for Version 1.4 (R2022a)
September 2022	Online only	Revised for Version 1.5 (R2022b)
March 2023	Online only	Revised for Version 1.6 (R2023a)

1	Tools
2	Assets
3	Functions
4	Objects
5	Protocol Buffers

Tools

Aerial Imagery Tool

Manage import and configuration of aerial imagery files

Description

The **Aerial Imagery Tool** manages the import and configuration of aerial imagery files. RoadRunner can import geolocated aerial imagery for use as a visual reference and for texture mapping onto surfaces. Geolocated imagery files can be imported through a variety of common formats, such as GeoTIFF (.tif, .tiff) and JPEG 2000 (.jp2), which contain the necessary map projection information to accurately position them on the surface of the Earth.

Refer to the **Aerial Image Assets** page for a list of the supported file types.

This tool can also be used to import and adjust nongeoreferenced imagery (for example, JPG screenshots).

Multiple images can be imported for an area to provide full coverage. This might cause some of the imported maps to overlap in certain regions. The priority of each image can be adjusted to determine which one takes priority in overlapping areas.

For links and examples about obtaining geographic information system (GIS) data compatible with RoadRunner, see “Download GIS Data for Use in RoadRunner”.



Open the Aerial Imagery Tool

On the RoadRunner toolbar, click the **Aerial Imagery Tool** button:



Examples

Import a Georeferenced Aerial Image

- 1 Click the **Aerial Imagery Tool** button.
- 2 In the **Library Browser**, navigate to the directory containing the aerial image on page 2-2 file you want to import. For more details on aerial images, see **Aerial Image Assets**.
- 3 Right-click the asset and make sure that the **Default Type** is set to **Aerial Image**.
- 4 Click and drag the asset from the **Library Browser** into the 3D scene.

Note If the geographic position has not yet been set for this scene, the scene center is set to the latitudinal and longitudinal center of the image. You can change the scene center using the **World Settings Tool**.

If the geographic position has already been set, but the imported image is outside of the maximum range of the scene, an error dialog box appears and cancels the import.

Import a Nongeoreferenced Aerial Image

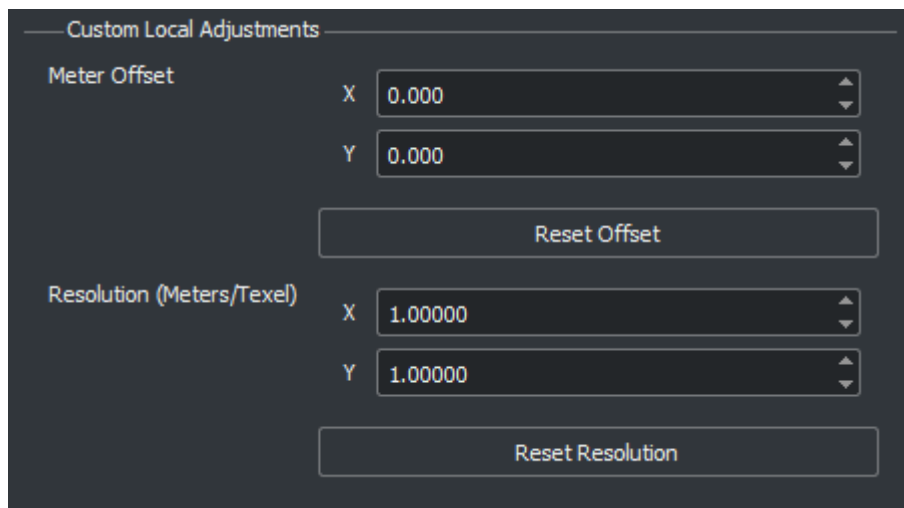
To correctly display satellite imagery in RoadRunner, the program must know how to position the image on the Earth. Obtaining satellite imagery that contains geolocation information (by using a format such as GeoTIFF or JPEG 2000) is strongly recommended. For links and examples about obtaining GIS data compatible with RoadRunner, see “Download GIS Data for Use in RoadRunner”.

If your imagery does not have geolocation information, it is possible to manually set geolocation information using the following steps.

If You Already Know the Projection

If you already know the specific projection to be used (that is, you have a 'proj' or 'wkt' projection string), you can set it on the file as follows:

- 1 Click the **Aerial Imagery Tool** button.
- 2 In the **Library Browser**, navigate to the directory containing the image file you want to import.
- 3 Right-click the file asset and make sure that the **Default Type** is set to **Aerial Image**.
- 4 Click the **Set Custom Projection** button in the **Attributes** pane.
- 5 Paste your 'proj' or 'wkt' string into the text field.
- 6 Click **OK**.
- 7 Scale the image by adjusting the **Resolution** to match the meters per pixel of the image.



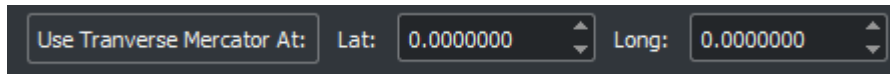
- 8 Click and drag the image asset from the **Library Browser** into the 3D scene.

If You Do Not Know the Projection

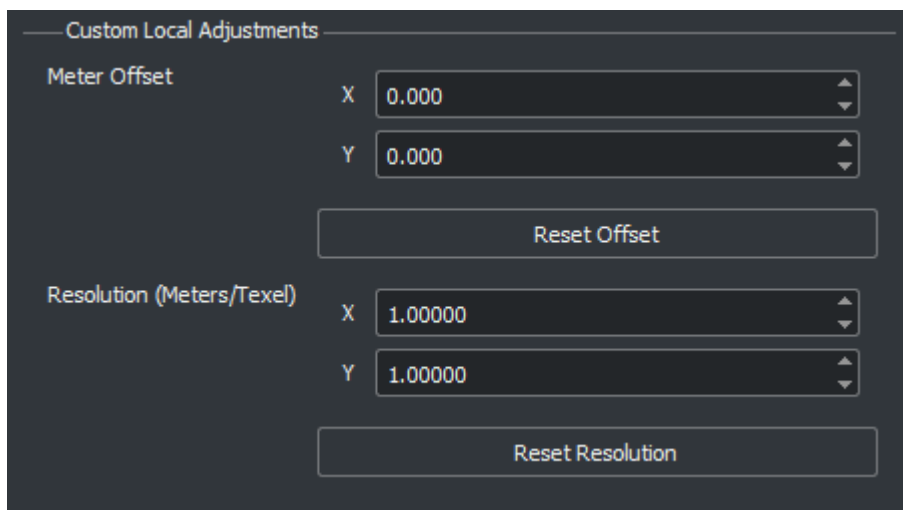
If you do not know the projection, you can experimentally try different projection values on the file. These instructions apply a Transverse Mercator projection to the file.

These steps enable you to use arbitrary images, such as a screenshot from a separate application. However, the result will not be highly accurate.

- 1 Click the **Aerial Imagery Tool** button.
- 2 In the **Library Browser**, navigate to the directory containing the image file you want to import.
- 3 Right-click the file asset and make sure that the **Default Type** is set to Aerial Image.
- 4 Press the **Set Custom Projection** button in the **Attributes** pane.
- 5 Determine the latitude and longitude of the center point of your image, then adjust the latitude and longitude values in the Custom Projection window, beside the **Use Transverse Mercator At** button, to match.



- 6 Click **Use Transverse Mercator At**. Then, click **OK**.
- 7 Scale the image by adjusting the **Resolution** to match the meters per pixel of the image.



- 8 Click and drag the image asset from the **Library Browser** into the 3D scene.

Remove an Aerial Image

- 1 Click the **Aerial Imagery Tool** button.
- 2 Click the aerial image you want to delete. The selected image is highlighted with a red bounding box.
- 3 Press the **Delete** key or select **Edit > Delete** from the menu bar.

Adjust the Resolution of the Loaded Aerial Imagery

- 1 Click the **Aerial Imagery Tool** button.
- 2 Under the Global Aerial Imagery Settings in the **Attributes** pane, adjust the **Meters Per Textel** value as desired.

Adjust the Properties of an Aerial Image

- 1 Click the **Aerial Imagery Tool** button.
- 2 Click the aerial image you want to edit.
- 3 Adjust the aerial image attributes as desired through the **Attributes** pane.

Note When more than one aerial image overlaps at a location, the system needs to decide which one to use. Selecting an aerial image and clicking the **Push to bottom** or **Bring to top** buttons in the **Attributes** pane adjusts a particular image's priority to resolve overlaps.

Assign an Aerial Imagery Material to a Surface

By default, aerial imagery is displayed only as a visual reference. You can optionally apply the aerial imagery to terrain surfaces by creating a new material and applying it to the surfaces.

Create a New Material

- 1 Click the **Aerial Imagery Tool** button. The global aerial imagery settings appear in the **Attributes** pane.
- 2 Press the **Generate Material** button in the **Attributes** pane.

This action generates a new image file called "Overlay.png" and a new material file called `Overlay.rmtl` inside the current directory within the **Library Browser**. It is necessary for the system to create a new image because the original aerial imagery might be in an incompatible projection or made up from multiple separate images. The **Generate Material** operation combines the multiple aerial images into one final image that can be mapped orthographically to the terrain surface.

Assign the Material to One or More Surfaces

- 1 Click the **Surface Tool** button.
- 2 Click and drag the material to assign it to a surface.

Toggle Display of Aerial Imagery

Select **View > Aerial Imagery** on the menu bar or press the **F4** key.

More About

Tips for Aerial Images in Large Areas

RoadRunner renders aerial images with a single texture image. In some cases, the size of the texture exceeds the maximum size supported by the graphics card and the imagery fail to render. This situation can be accompanied by an error message similar to this one:

```
ERROR: Unable to load overlay: Max byte count exceeded. Max: 16k x 16k texels with 4 channels.
```

Here are a few tips that can help with handling imagery of large areas.

Ensure Your Workspace Size Is No Larger Than Needed

The portion of the image that is loaded is determined by the **Workspace** and **World Origin** settings in the **World Settings Tool**. Ensure that your workspace covers only your area of interest.

In particular, if you georeferenced your scene by dragging **Aerial Image Assets** into the scene, the scene is centered on the middle of the image. If you care only about a portion of the scene in the corner of the image, do not just increase the workspace extents. Instead, try one of the following:

- Before dragging in the aerial image, use the **World Settings Tool** to set your latitudinal and longitudinal center of interest

- Use the **World Settings Tool** to move the workspace to the center of interest. Then, adjust the extents to cover the area of interest. Imagery is loaded only within the workspace extents.

Decrease the Imagery Sampling Rate

The **Meters Per Texel** option in the Aerial Imagery Tool controls the sampling rate of the imagery. Increase this value to enable viewing of larger areas at the expense of lower image quality.

Adjust the Workspace as You Work

If you have high-resolution imagery and want to create a large area, you can adjust the workspace based on the current area you are working on. For example, you can adjust the workspace to cover the northwest portion of your area when you need to edit the northwest area. Then, you can adjust the workspace again when you need to work on the northeast area.

If you use this approach, be sure to increase the workspace to cover your entire area when you export.

Version History

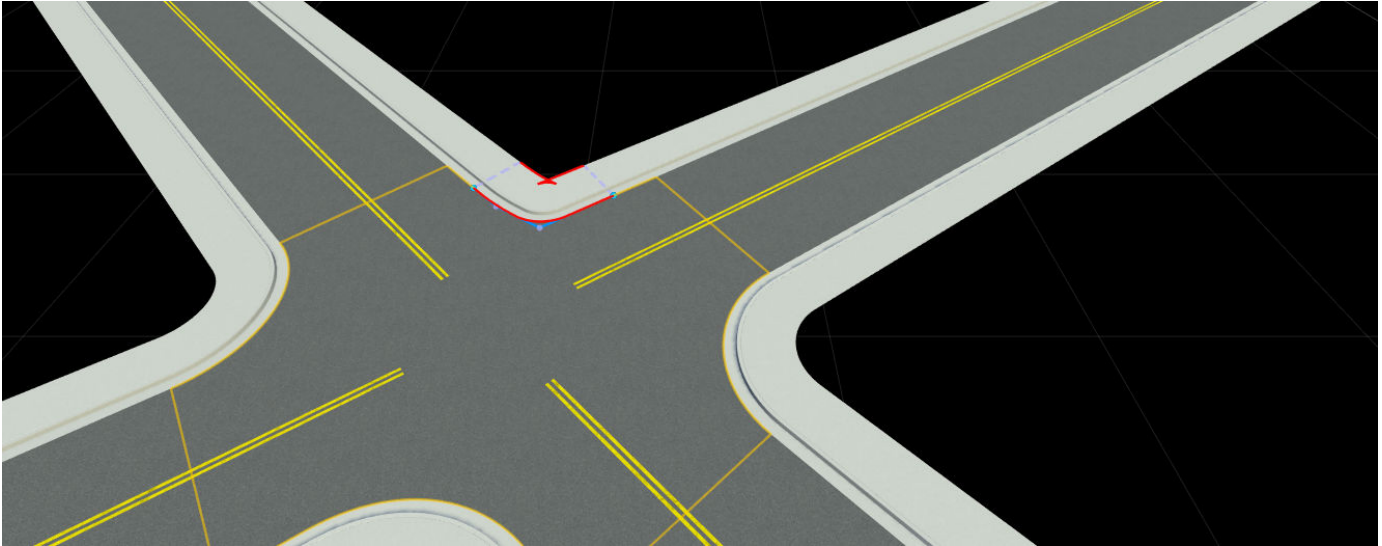
Introduced in R2020a

Corner Tool

Adjust shape and materials of junction corners

Description

The **Corner Tool** is used to adjust the shape and materials of junction corners.



Open the Corner Tool

On the RoadRunner toolbar, click the **Corner Tool** button:



Version History

Introduced in R2020a

Cross Section Tool

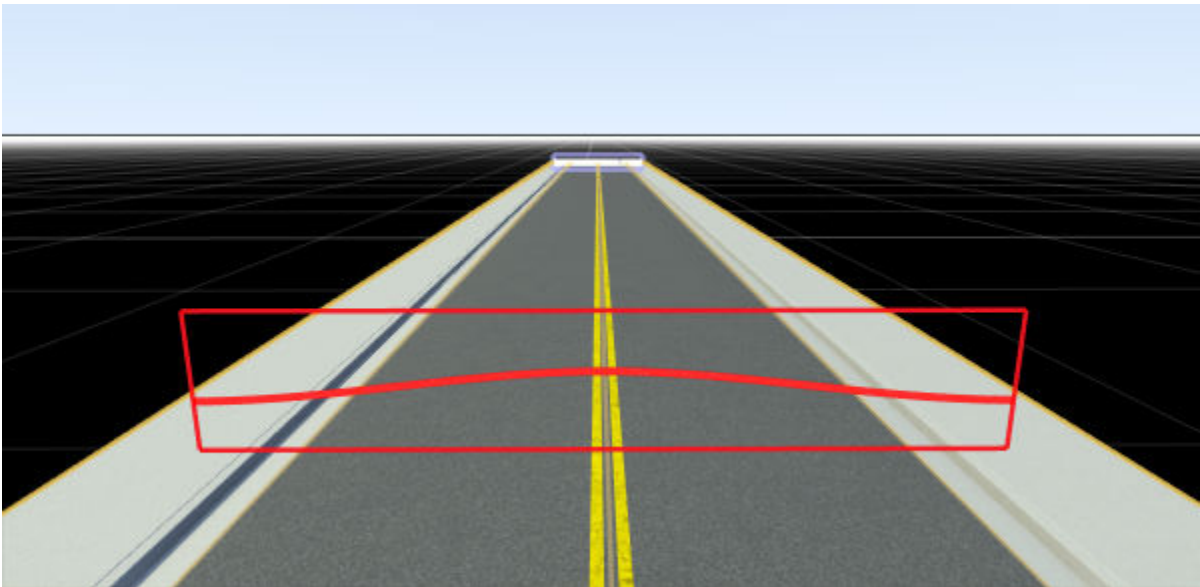
Manipulate banking, crowning, and curb shapes at road cross-sections

Description

The **Cross Section Tool** enables you to edit the shape of roads at specified cross-sections to manipulate banking, crowning, and curb shapes.

By default, all roads have a cross-section defined at the start and the end of the road. Use this tool to add additional cross-section nodes at arbitrary points along the road. When you modify these cross-section nodes, either in the scene editing canvas or the **2D Editor**, RoadRunner interpolates the shape between the nodes. You can also use this tool to modify cross-sections imported from ASAM OpenDRIVE® files.

To modify the banking angle along the full width of the road, use the **Road Superelevation Tool**. To modify the sidewalk heights and adjust the height of the curb, use the **Sidewalk Height Tool**.



Open the Cross Section Tool

On the RoadRunner toolbar, click the **Cross Section Tool** button:

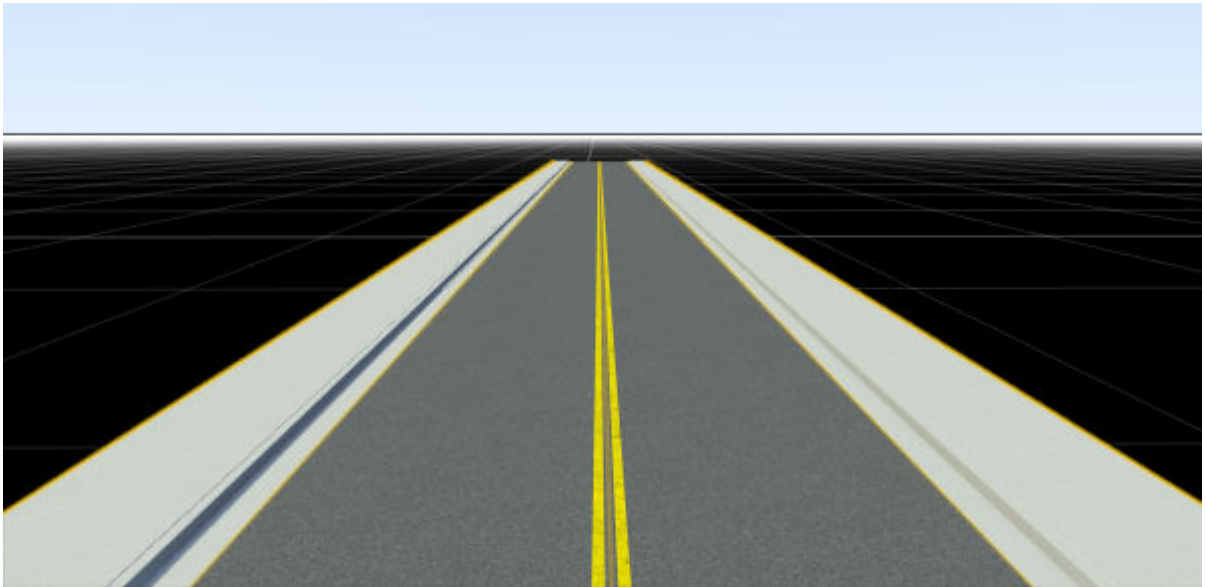


Examples

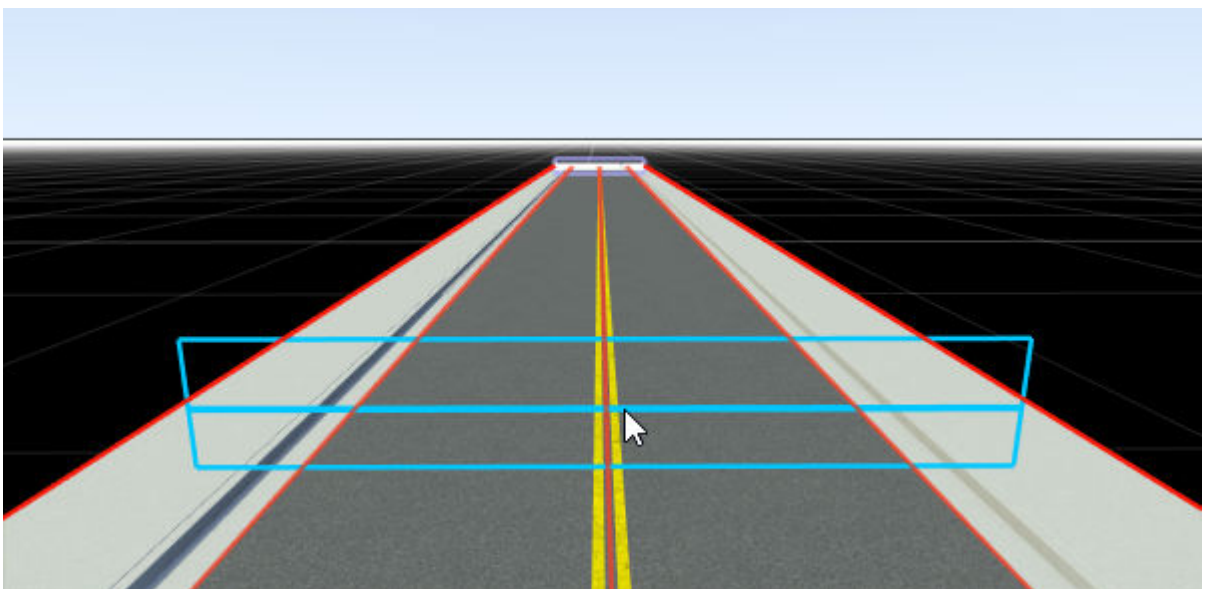
Modify Road at Cross-Section

Add crowning, or slight elevation, to a cross-section of a road.

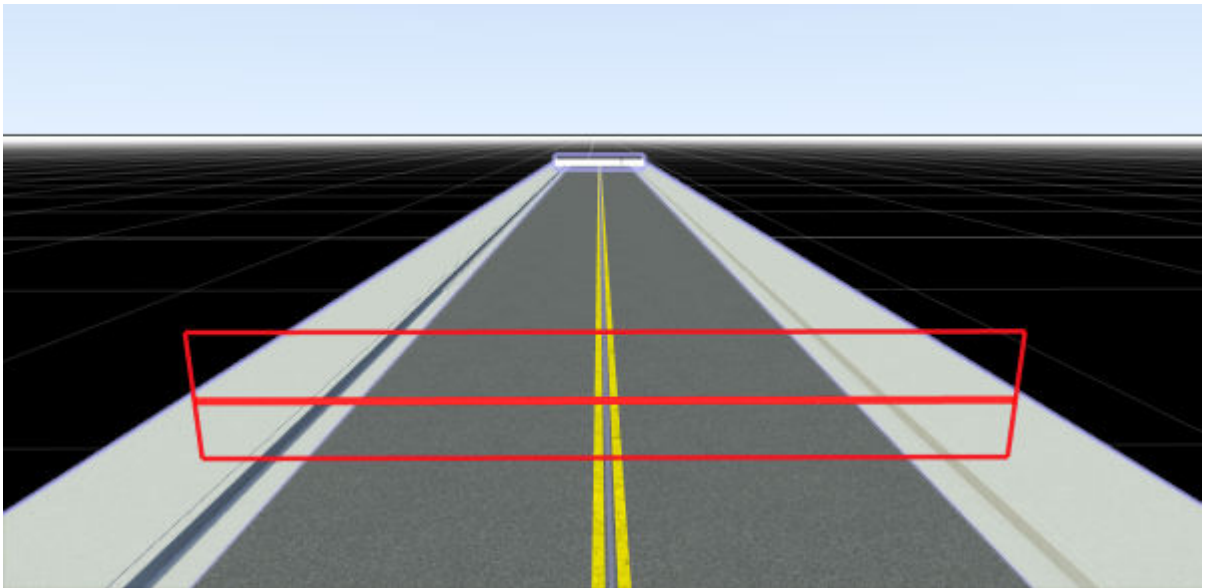
- 1 Create a straight road segment by using the **Road Plan Tool**. Zoom in on the center of the road, and rotate the camera to view the road from the center to one of the road edges.



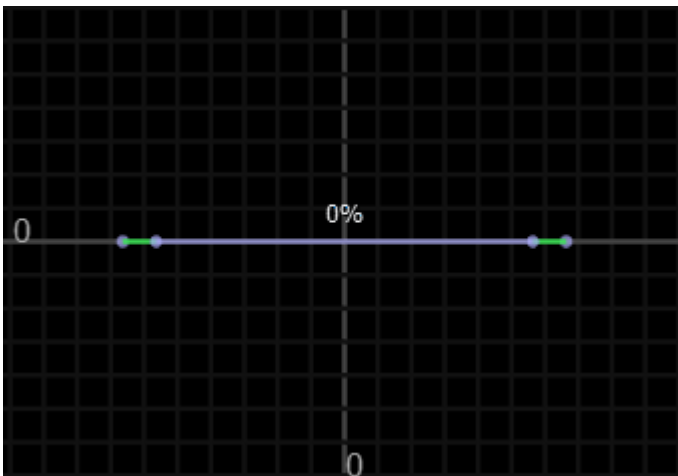
- 2 Click the road to select it. Then, click the **Road Cross Section Tool** button. As you move your pointer along the road, the road now displays a preview of a flat road cross-section in blue.



- 3 Create a road cross-section by right-clicking at the approximate center of the road. In the scene editing canvas, the cross-section is displayed in red.

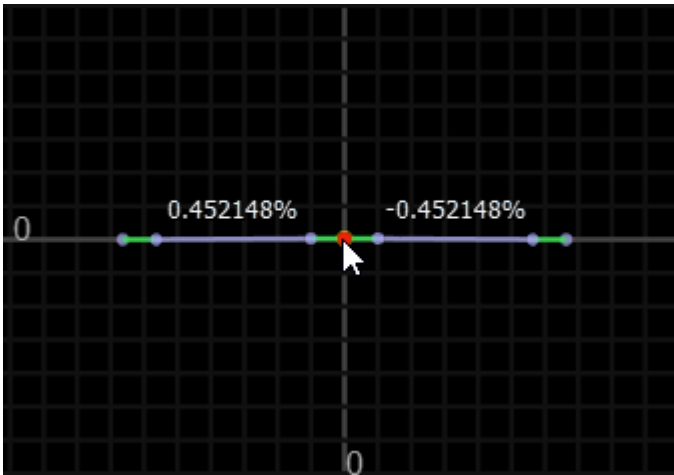


The cross-section is also displayed in the **2D Editor**.

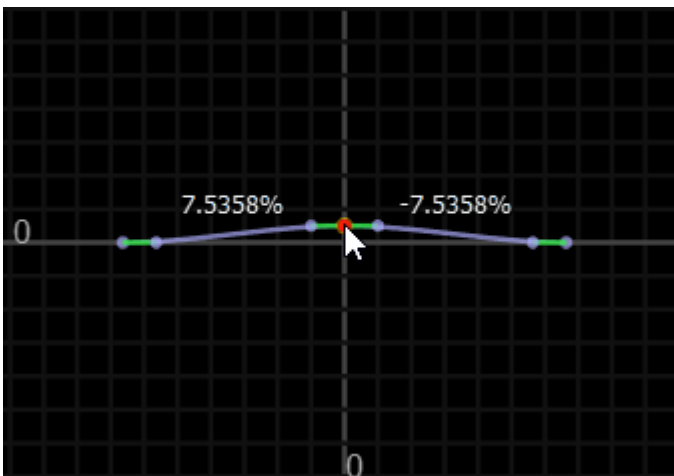


To modify the position of the cross-section further, either click and drag the cross-section or, in the **Attributes** pane, modify the **Distance** attribute of the cross-section. You cannot move a cross-section past another cross-section and you cannot move a cross-section past the start or end of a lane.

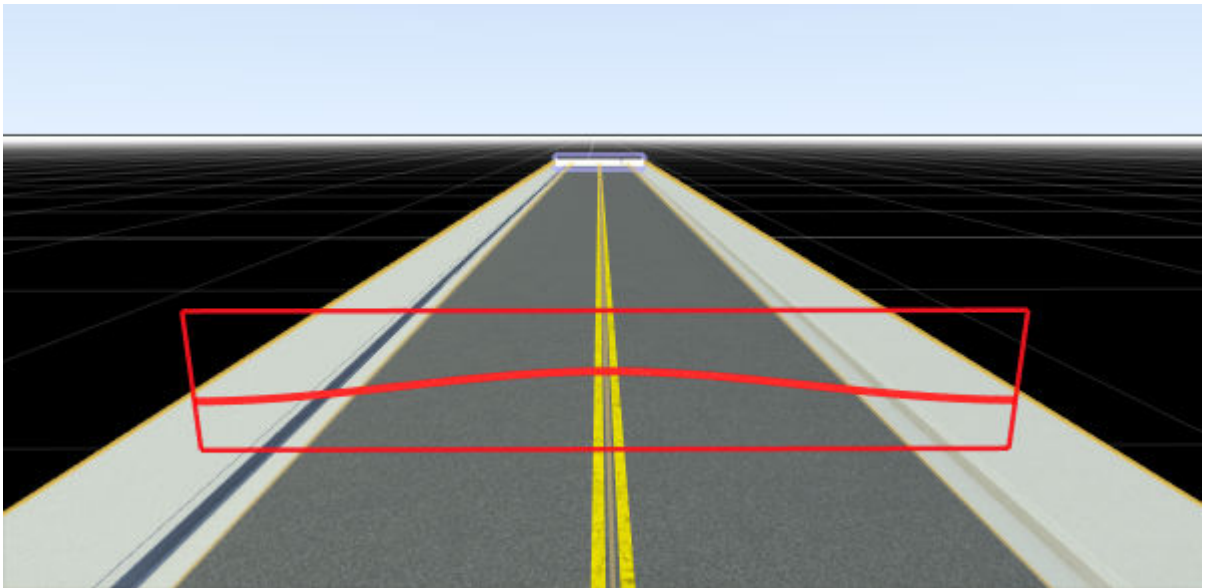
- 4 In the **2D Editor**, right-click the center of the cross-section to add an editable node to the cross-section.



- 5 Click and drag the center cross-section node up until it has an approximate height of 0.5 meters. For more precise control over the height of the node, select the node in the **2D Editor** and, in the **Attributes** pane, set the **Height** attribute to 0.5.



In the scene editing canvas, the road is now crowned at the inserted cross-section, sloping down to the left and right of its elevated center.



- 6 (Optional) Modify additional parts of the cross-section from the **2D Editor**. For example, drag the node tangents to change the slope of the road at the edges.

The cross-section of the road remains flat at the road edges, and RoadRunner interpolates the height and slope of the road between the inserted cross-section and the road edges.

Parameters

Road Cross-Section Attributes

Attribute	Description
Distance	Position of the cross-section along the road, in meters, specified as a nonnegative real scalar. Distance is relative to the edge of the road that was created first.

Cross-Section Node Attributes

Attribute	Description
Height	Height of node above the ground, in meters, specified as a real scalar.
Offset	Lateral offset of node from the center of the cross-section, in meters, specified as a real scalar.

Cross-Section Node Tangent Attributes

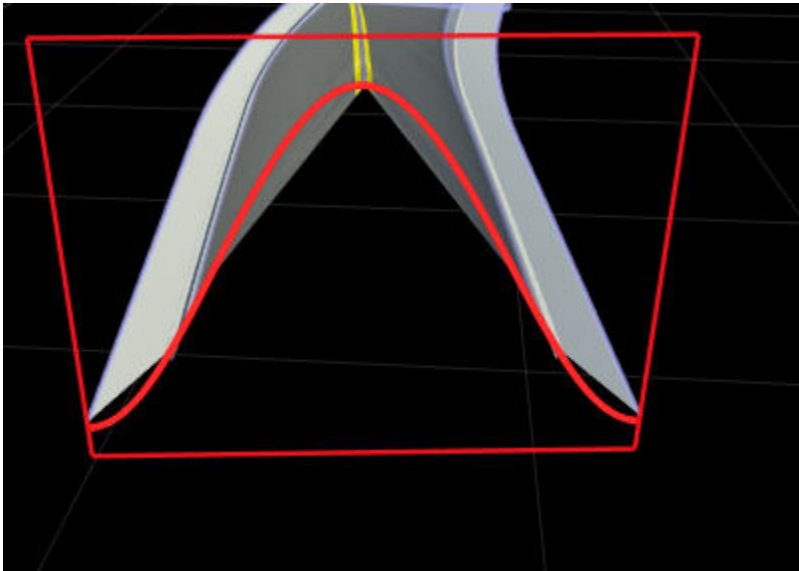
Attribute	Description
Slope	Slope of node tangent, in percent grade, specified as a real scalar.

Cross-Section Span Attributes

Attribute	Description
Height	Height of span above ground, in meters, specified as a real scalar. If the height varies along the span, then the Height attribute displays the height of the cross-section node connected to the left side of the span.
Slope	Slope of span, in percent grade, specified as a real scalar.

Limitations

- RoadRunner imports superelevation data from ASAM OpenDRIVE files but visualizes ASAM OpenDRIVE roads only by interpolating between the specified lane boundary positions. This visualization difference means that roads shown in the scene editing canvas do not match the superelevation data in the **Cross Section Tool**. For example, this imported road renders as having a sharp, triangular slope whereas the actual superelevation data shows that the road has a more bell-shaped slope.



When you export such roads back to ASAM OpenDRIVE, RoadRunner exports the superelevation data so that the roads maintain their correct analytical representation.

Tips

- You can save a cross-section as a road style template for use with creating future roads. Select a road cross-section and, in the **Attributes** pane, click **Create Road Style**. Save the new road style template to the **Library Browser**.

Version History

Introduced in R2020a

See Also

Road Superelevation Tool | Sidewalk Height Tool

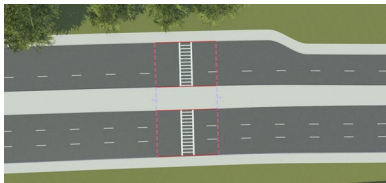
Crosswalk And Stop Line Tool

Add crosswalks and stop lines between corners at intersections

Description

The **Crosswalk and Stop Line Tool** can be used to add crosswalks and stop lines between corners at intersections.

Note Free-form crosswalks can be created using the **Marking Curve Tool**. Where possible, use the **Crosswalk And Stop Line Tool** instead. Crosswalks and stop lines created in this tool have more semantic linkage to the road topology.



Open the Crosswalk And Stop Line Tool

On the RoadRunner toolbar, click the **Crosswalk And Stop Line Tool** button:

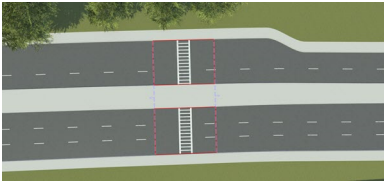


Examples

Add a Single Crosswalk to a Junction

- 1 Click the **Crosswalk and Stop Line Tool** button.
- 2 Optionally, select a desired crosswalk style in the **Library Browser**.
- 3 Click the corner of a junction that you want the crosswalk to start from.
- 4 Right-click the corner that you want the crosswalk to end at (in the same junction).

Tip If you need to create a standalone crosswalk (that is, a crosswalk along a road that does not involve an intersection), you can use the **Custom Junction Tool** to create a junction along a single road.



You can also create free-form crosswalks anywhere in your scene using the **Marking Curve Tool** (with **Crosswalk Marking Assets** selected). Note that free-form crosswalks might lack certain export-related functionality.

Quickly Add All Crosswalks to a Junction

- 1 Click the **Crosswalk and Stop Line Tool** button.
- 2 Check that no junctions are selected.
- 3 Optionally, select a desired crosswalk style in the **Library Browser**.
- 4 Right-click a junction to add crosswalks across each road.

Delete a Crosswalk

- 1 Click the **Crosswalk and Stop Line Tool** button.
- 2 Select a crosswalk.
- 3 Press the **Delete** key or select **Edit > Delete** from the menu bar.

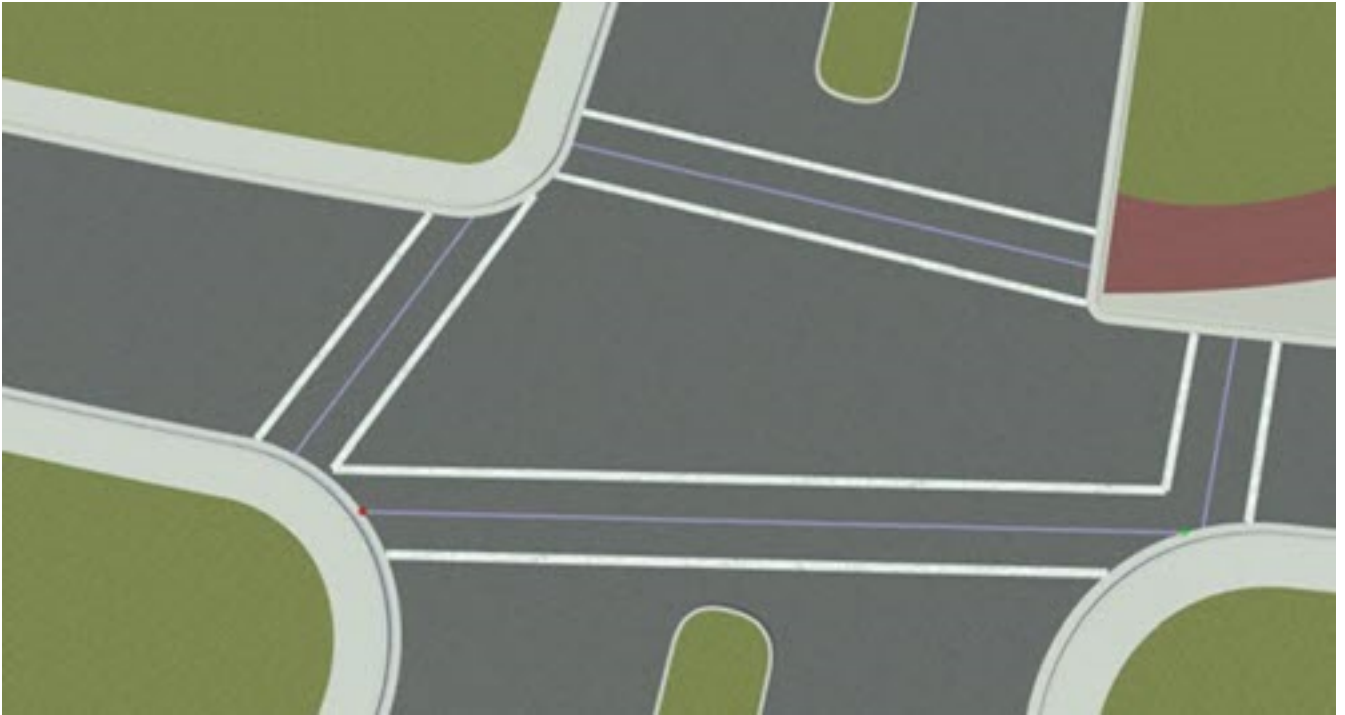
Assign a Style to a Crosswalk

- 1 Enter the **Crosswalk and Stop Line Tool**.
- 2 Select the crosswalk you want to edit.
- 3 Click and drag **Crosswalk Marking Assets** from the **Library Browser** onto the **Crosswalk Style** widget in the **Attributes** pane.

Alternatively, click and drag **Crosswalk Marking Assets** from the **Library Browser** onto a crosswalk in the scene.

Adjust the Location of a Crosswalk

After creating a crosswalk, you can move the locations of the crosswalk's end points (for example, to create crosswalks that cross a road at an angle).

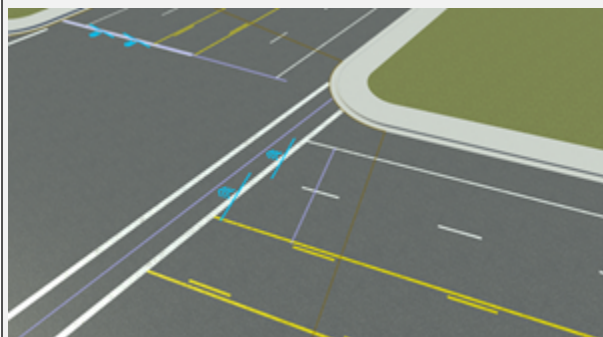


- 1 Click the **Crosswalk and Stop Line Tool** button.
- 2 Select the crosswalk you want to edit.
- 3 Click and drag a point at the end of the crosswalk. Alternatively, adjust the **Left Corner Offset** and **Right Corner Offset** values in the **Attributes** pane.

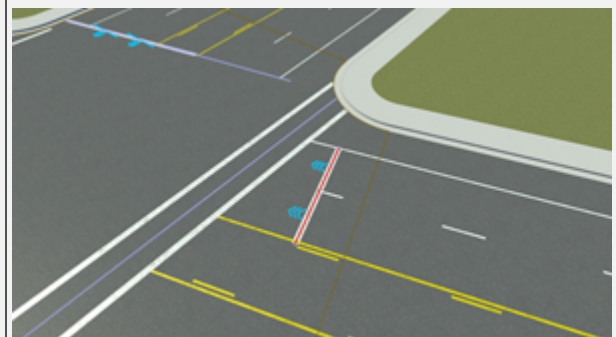
Note The points at the end of a crosswalk must lie within the extents of the junction. Note that you can expand the extents of a junction by extending the corners using the **Corner Tool**.

Add Stop Lines to a Junction

Preview shows where stop line will appear:



The new stop line updates the stopping locations:

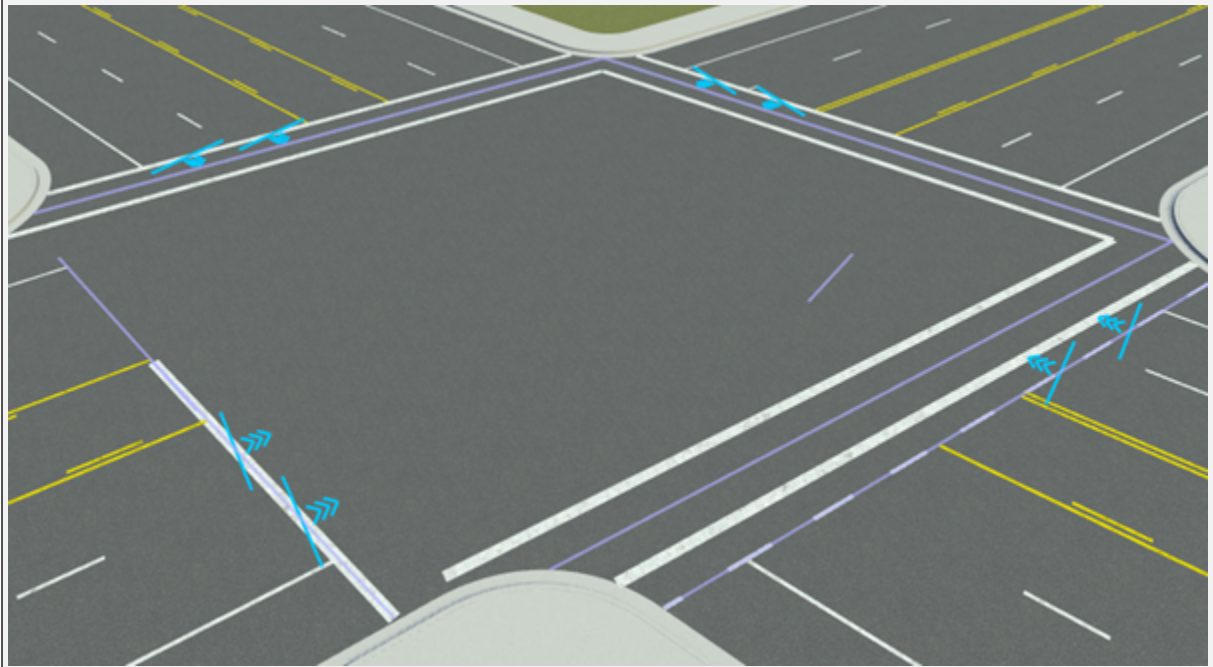


- 1 Click the **Crosswalk and Stop Line Tool** button.
- 2 Select the junction you want to edit.
- 3 Right-click the road at the location where you want to add a new stop line node.

View Stopping Locations for a Junction

Stopping locations are computed for all lanes approaching a junction. Stopping locations define the starting location of all maneuvers.

Stopping locations shown as blue chevrons:



To view stopping locations for a junction, follow these steps:

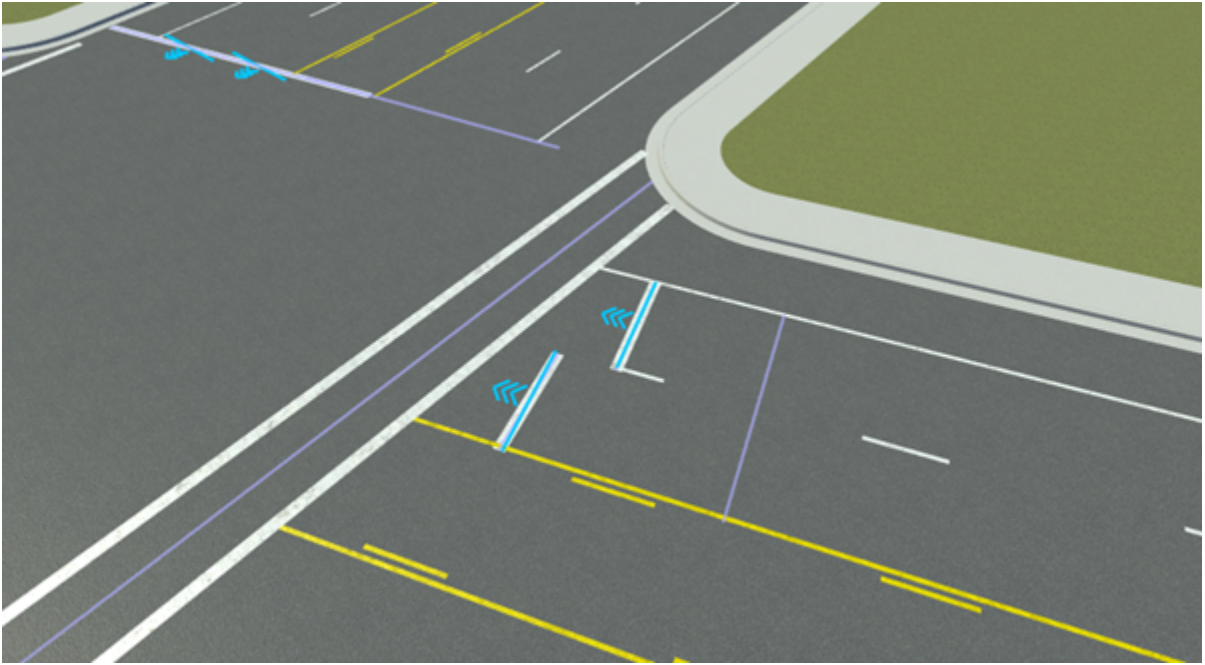
- 1 Click the **Crosswalk and Stop Line Tool** button.
- 2 Select the junction you want to edit.

Stop Line Editing

- 1 Click the **Crosswalk and Stop Line Tool** button.
- 2 Select the junction you want to edit.
- 3 Stop line editing is similar to marking curves. See **Marking Curve Tool** for documentation on how to edit stop lines.

Multiple Stop Lines

Multiple stop lines can be added to a single approach to add unique stopping locations for each lane.



Version History

Introduced in R2020a

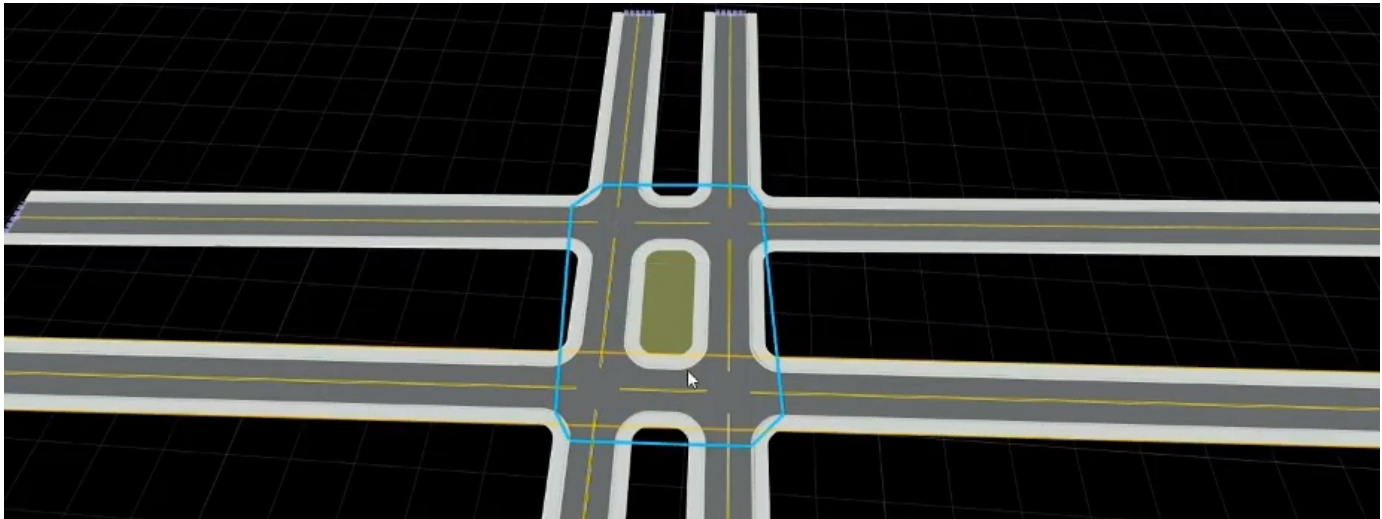
Custom Junction Tool

Override RoadRunner automatic junction functionality for advanced cases

Description

The **Custom Junction Tool** is used to override the RoadRunner automatic junction functionality for advanced cases. It enables the creation of junctions where no overlaps are present. Junctions that are automatically created by RoadRunner are referred to as automatic junctions. These junctions are automatically created, updated, and removed as necessary. Junctions that are manually created by this tool are referred to as locked junctions. These junctions are manually created and must be manually removed. It is sometimes desirable to convert an automatic junction to a locked junction, either to add more roads to the junction or to change the default stop locations.

The default stop location represents the start or end of a junction along a road. It is the location where newly created maneuver roads automatically stop and the default distance where corners begin. The default stop location has a direction that points out from the junction. This direction is used for determining which roads need to be connected by using corners.



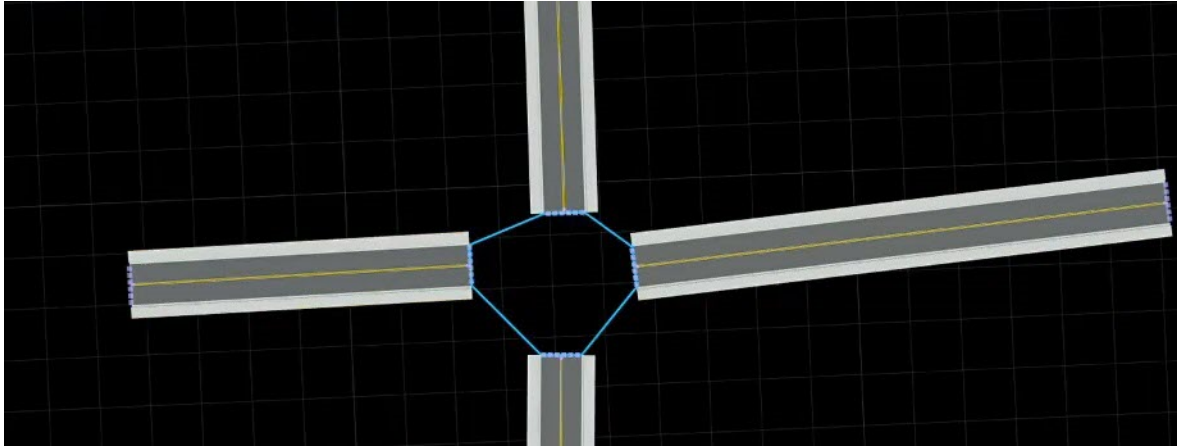
Open the Custom Junction Tool

On the RoadRunner toolbar, click the **Custom Junction Tool** button:



Examples

Create a Junction Between Roads That Do Not Overlap



- 1 Click the **Custom Junction Tool** button.
- 2 Right-click the end of each road that will be part of the junction.
- 3 Press **Spacebar** to create the junction.

Remove a Locked Junction

- 1 Click the **Custom Junction Tool** button.
- 2 Click the desired junction.
- 3 Press **Delete**.

Note Removing a locked junction can result in an automatic junction in its place if roads are overlapping.

Convert an Automatic Junction to a Locked Junction

- 1 Click the **Custom Junction Tool** button.
- 2 Click the desired junction.
- 3 In the **Attributes** pane, click **Convert to Locked Junction**.

Convert a Locked Junction to an Automatic Junction

- 1 Click the **Custom Junction Tool** button.
- 2 Click the desired junction.
- 3 In the **Attributes** pane, click **Convert to Automatic Junction**.

Note A locked junction cannot always be converted to an automatic junction. If no automatic junction is possible, the junction will be removed during conversion.

Add a Road to a Locked Junction

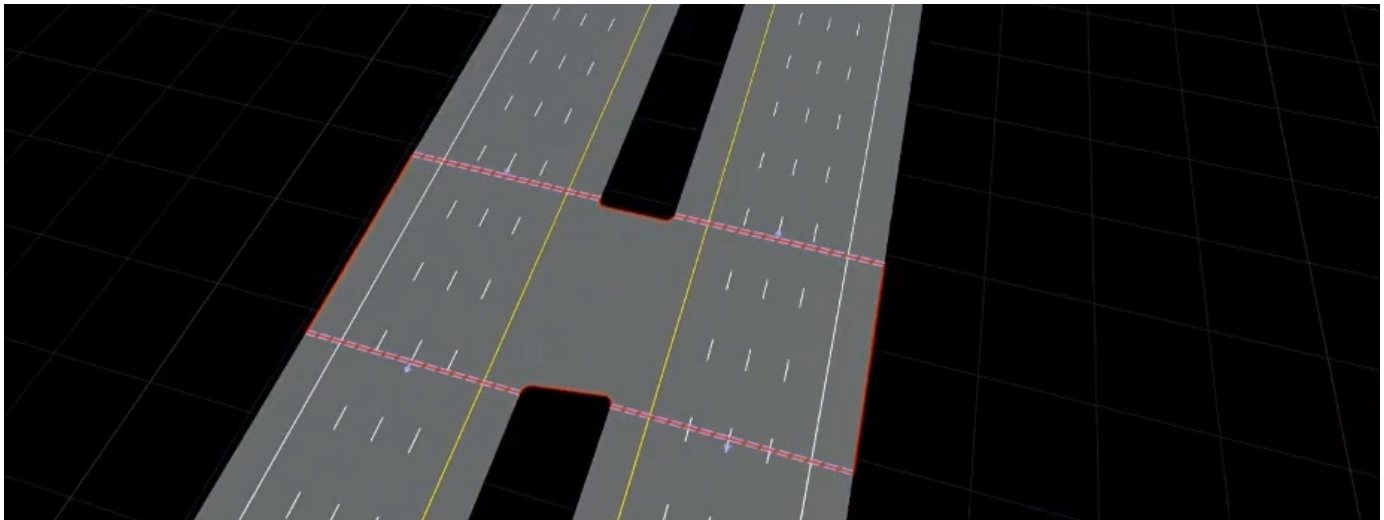
- 1 Click the **Custom Junction Tool** button.
- 2 Right-click the end of the road to add to the junction.
- 3 Right-click the junction.

- 4 Press **Spacebar** to add the road.

Remove a Road from a Locked Junction

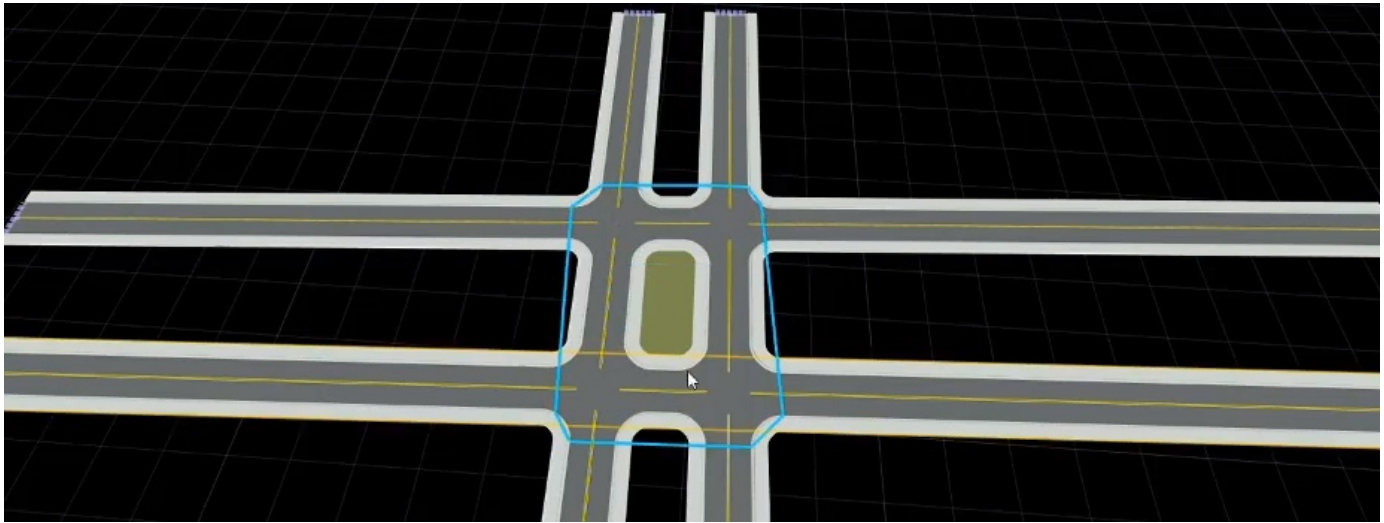
- 1 Click the **Custom Junction Tool** button.
- 2 Click the desired junction.
- 3 Click the **Default Stopline** of the road that you want to remove.
- 4 Press **Delete**.

Create a Junction Between Two Parallel Roads



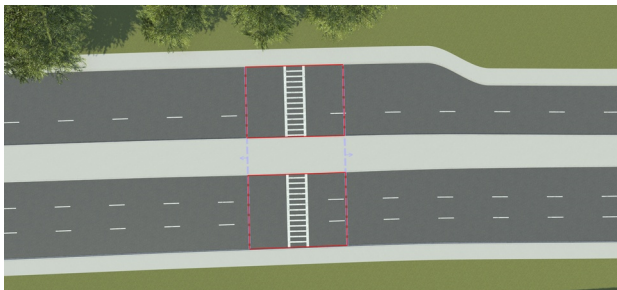
- 1 Click the **Custom Junction Tool** button.
- 2 Right-click at the start of the junction on one road and then the end of the junction on the same road.
- 3 Right-click at the start of the junction on the other road and then the end of the junction on the same road.
- 4 Press **Spacebar** to create the junction. If the corners do not appear correctly, see the “Troubleshooting Locked Junctions” on page 1-24 section.

Merge Two or More Junctions



- 1 Click the **Custom Junction Tool** button.
- 2 Right-click each junction to be merged.
- 3 Press **Spacebar** to merge the junctions together.
- 4 Click any extraneous default stop lines on the inside of the junction and press **Delete**.

Create a Junction Along a Single Road



Creating junctions along a single road is useful for creating standalone crosswalks. To create a junction along a single road:

- 1 Click the **Custom Junction Tool** button.
- 2 Right-click the road at the start of the junction and at the end of the junction. If you are making a crosswalk, the start and end of the junction roughly correspond to each side of the crosswalk.
- 3 Press **Spacebar** to create the junction. If the junction does not appear correctly, see the "Troubleshooting Locked Junctions" on page 1-24 section.
- 4 Optionally, if you are making a crosswalk:
 - 1 Click the **Crosswalk and Stop Line Tool** button.
 - 2 Click the outer side of the junction.
 - 3 Right-click the other side of the junction to create the crosswalk.

Adjust a Default Stop Location

- 1** Click the **Custom Junction Tool** button.
- 2** Click the desired junction.
- 3** If the junction is automatic, convert the junction to locked.
- 4** Click and drag the desired stop location.

Change the Direction of a Default Stop Location

- 1** Click the **Custom Junction Tool** button.
- 2** Click the desired junction.
- 3** Click the desired default stop location.
- 4** In the **Attributes** pane, click **Flip Direction**.

More About

Troubleshooting Locked Junctions

RoadRunner attempts to make a reasonable locked junction based on the specified default stop locations, but the software might be unable to determine the necessary corners to make a reasonable junction. Here are a few steps to take if a locked junction's corners are not computed correctly:

- Check that all default stop locations point outward from the junction. If a default stop location has been placed manually on a road, the initial direction might be flipped.
- Check that the default stop locations are not too close together. Try dragging the locations farther apart and clicking **Sort Rays** to redetermine the corners.
- Try using multiple smaller locked junctions instead of one large one. Remove roads from the junction as necessary.
- Try using overlaps instead of making the junction manually.

Version History

Introduced in R2020a

Elevation Map Tool

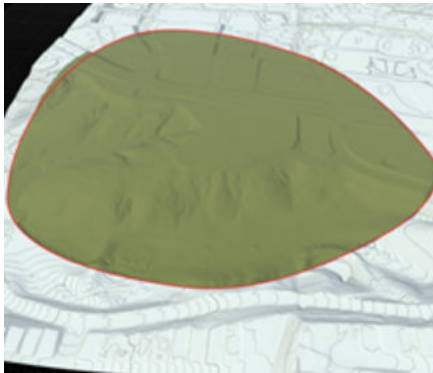
Manage import and configuration of digital elevation model (DEM) files

Description

The **Elevation Map Tool** manages the import and configuration of digital elevation model (DEM) files. RoadRunner can import elevation map data from a variety of file formats, such as DEM, IMG, JPEG 2000, and TIFF. Some of these formats support georeferencing and can be automatically positioned accordingly.

Refer to the **Elevation Map Assets** page for a list of the supported file types.

Multiple elevation maps can be imported for an area to provide full coverage. This can cause some of the imported maps to overlap in certain regions. You can adjust the priority of each map to determine which one takes priority in overlapping areas.



Open the Elevation Map Tool

On the RoadRunner toolbar, click the **Elevation Map Tool** button:



Examples

Import a Georeferenced Elevation Map

- 1 Click the **Elevation Map Tool** button.
- 2 In the **Library Browser**, navigate to the directory containing the **Elevation Map Assets** you want to import.
- 3 Right-click the assets and make sure that the **Default Type** is set to Elevation Map.
- 4 Click and drag the assets from the **Library Browser** into the scene editing canvas.

Note If the geographic position has not yet been set for this scene, the scene center is set to the latitudinal and longitudinal center of the image. You can change the scene center using the **World Settings Tool**.

If the geographic position has already been set, but the imported image is outside of the maximum range of the scene, an error dialog box appears and cancels the import.

Import a Nongeoreferenced Elevation Map

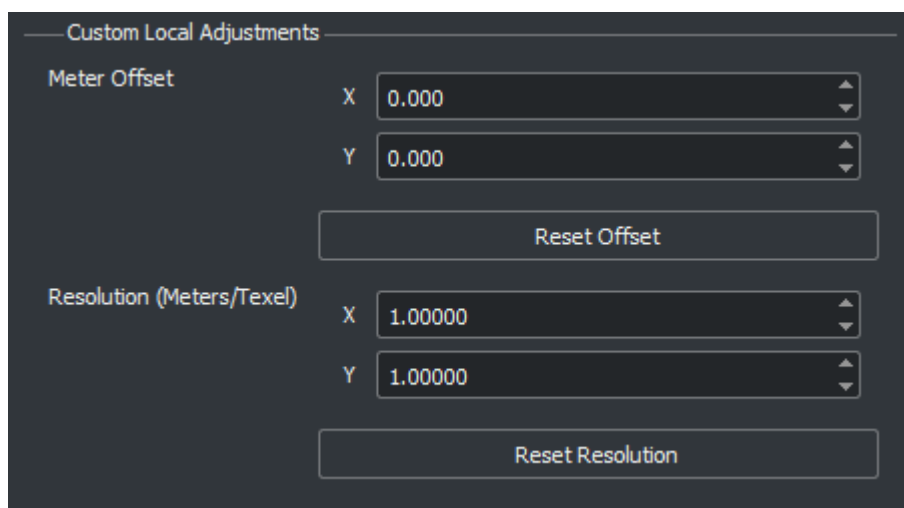
To correctly display an elevation map in RoadRunner, the program must know how to position the map on the Earth. For best accuracy, obtaining elevation data that contains geolocation information (by using a format such as GeoTIFF or JPEG 2000) is strongly recommended. For links and examples about obtaining GIS data compatible with RoadRunner, see “Download GIS Data for Use in RoadRunner”.

If your elevation map does not have geolocation information, it is possible to manually set geolocation information using the following steps.

If You Already Know the Projection

If you already know the specific projection to be used (that is, you have a 'proj' or 'wkt' projection string), you can set it on the file as follows:

- 1 Click the **Elevation Map Tool** button.
- 2 In the **Library Browser**, navigate to the directory containing the elevation map file you want to import.
- 3 Right-click the file asset and make sure that the **Default Type** is set to Elevation Map.
- 4 Click the **Set Custom Projection** button in the **Attributes** pane.
- 5 Paste your 'proj' or 'wkt' string into the text field.
- 6 Click **OK**.
- 7 Scale the data by adjusting the **Resolution** to match the meters per pixel of the elevation map.



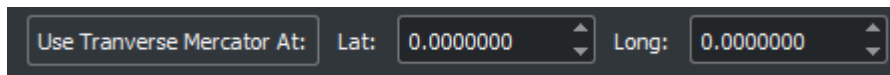
- 8 Click and drag the elevation map asset from the **Library Browser** into the 3D scene.

If You Do Not Know the Projection

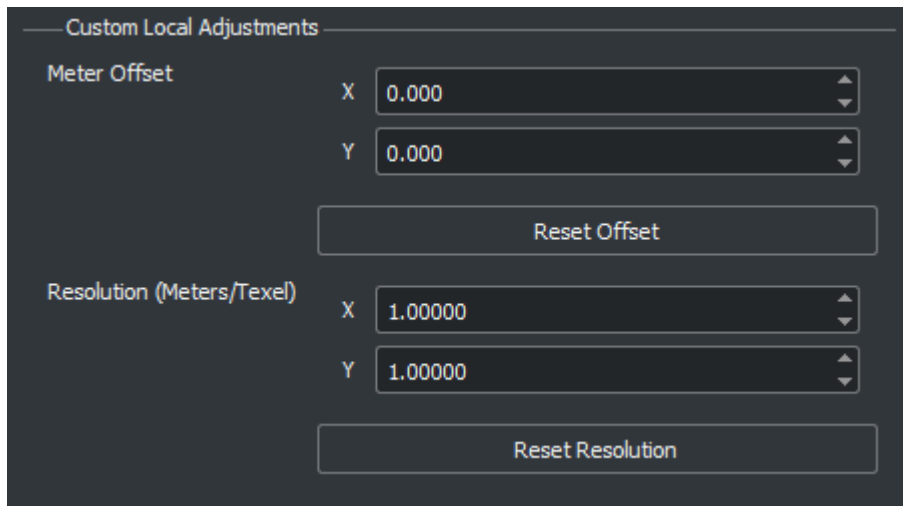
The following steps enable you to use arbitrary grayscale images for elevation, such as a screenshot from a separate application. However, the result will not be highly accurate.

If you do not know the projection, you can experimentally try different projection values on the file. These instructions apply a Transverse Mercator projection to the file.

- 1 Click the **Elevation Map Tool** button.
- 2 In the **Library Browser**, navigate to the directory containing the image file you want to import.
- 3 Right-click the file asset and make sure that the **Default Type** is set to Elevation Map.
- 4 Click the **Set Custom Projection** button in the **Attributes** pane.
- 5 Determine the latitude and longitude of the center point of your elevation data. Then, adjust the latitude and longitude values in the Custom Projection window to match.



- 6 Click **Use Transverse Mercator At**. Then, click **OK**.
- 7 Scale the data by adjusting the **Resolution** to match the meters per pixel of the image.



- 8 Click and drag the elevation map asset from the **Library Browser** into the 3D scene.

Remove an Elevation Map

- 1 Click the **Elevation Map Tool** button.
- 2 Click the elevation map you want to delete.
- 3 Press the **Delete** key or select **Edit > Delete** from the menu bar.

Adjust the Properties of an Elevation Map

- 1 Click the **Elevation Map Tool** button.
- 2 Click the elevation map you want to edit.
- 3 Adjust the elevation map attributes as desired through the **Attributes** pane.

Note When more than one elevation map overlaps at a location, the system needs to decide which one to use. Selecting an elevation map and clicking the **Push to bottom** or **Bring to top** buttons in the **Attributes** pane adjusts a map's priority to resolve overlaps.

Toggle the Display of Elevation Maps

Select **View > Elevation** from the menu bar, or press **F5**.

Project Roads and Other Objects to Elevation Maps

Many RoadRunner objects can be projected to an elevation map surface. The specific steps are documented in the appropriate tools. For example, the steps for projecting roads to elevation maps can be found here: [Project Roads to Elevation Maps](#) on page 1-131.

In most cases, the steps are the same:

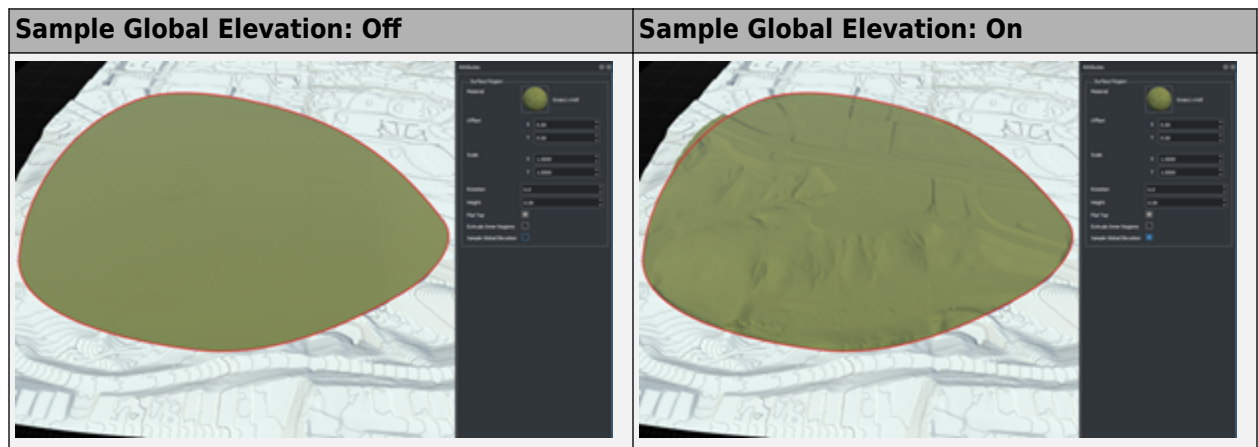
- 1 Select the appropriate tool.
- 2 Select one more objects in the scene editing canvas.
- 3 Click the appropriate projection button in the toolbar on the left.

Include Elevation Maps in Surface Triangulation

By default, the elevation maps are displayed as a white topographic surface. This surface is not included in the exported scene.

To include the visual influence of the elevation maps in the scene, you must create terrain surfaces covering the area, and then enable global elevation. For more details on terrain surfaces, see “[How Surfaces Work in RoadRunner](#)”.

For more information, refer to [Control Whether a Surface Uses Elevation Samples](#) on page 1-214 .



Version History

Introduced in R2020a

Topics

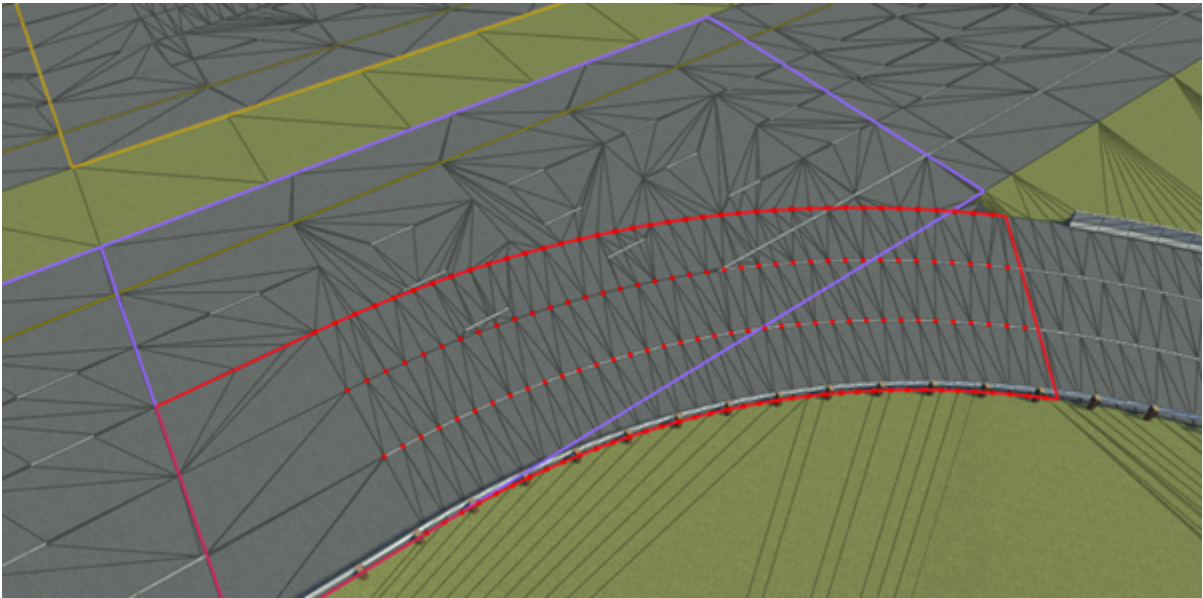
“[Create Roads Around Imported GIS Assets](#)”

Junction Surface Tool

Control how road elevations and cross-sections influence interior triangulation of intersections

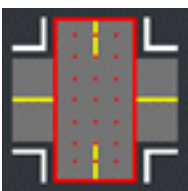
Description

The **Junction Surface Tool** controls how road elevations and cross-sections influence the interior triangulation of intersections.



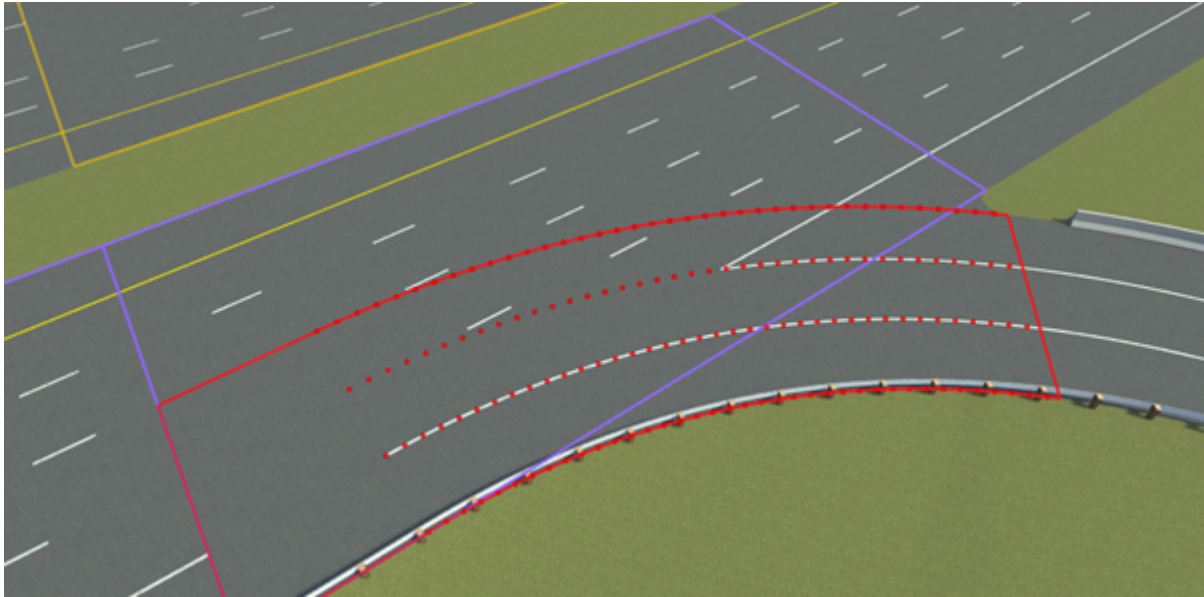
Open the Junction Surface Tool

On the RoadRunner toolbar, click the **Junction Surface Tool** button:



Examples

View Road Samples Within a Junction



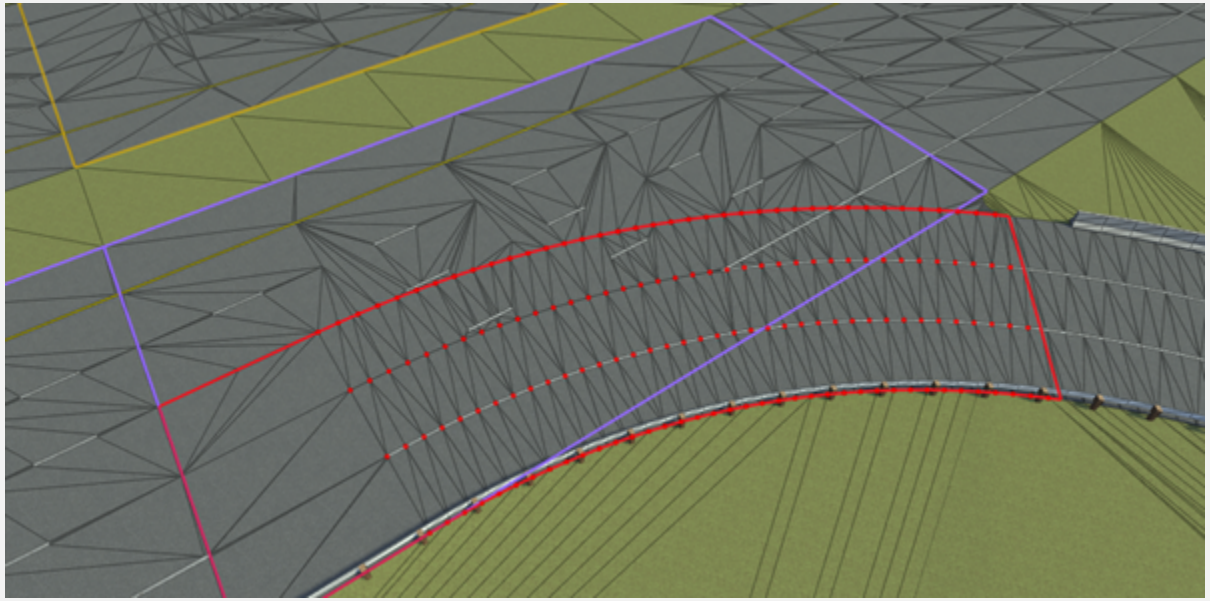
- 1 Click the **Junction Surface Tool** button.
- 2 Select a junction. This selection displays all the individual road surface spans that overlap the junction.
- 3 Click a road span. When selected, a road span draws the portion of the road that overlaps the junction and the samples that it includes.

Include or Exclude Samples from a Road Span

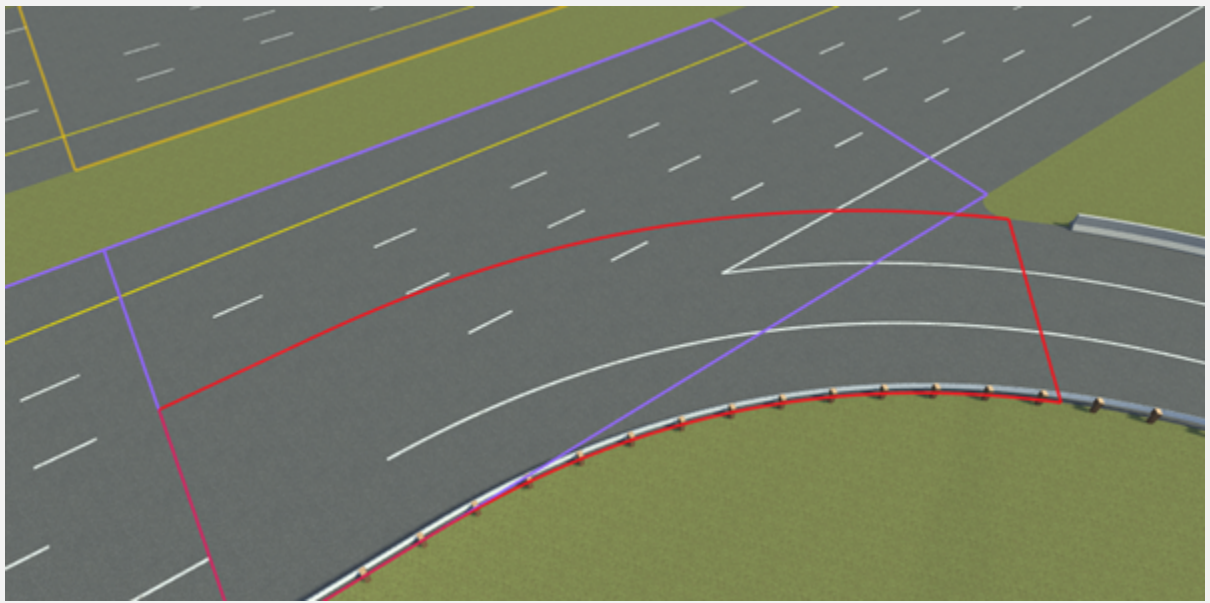
You can include or exclude samples from specific spans from the triangulation.

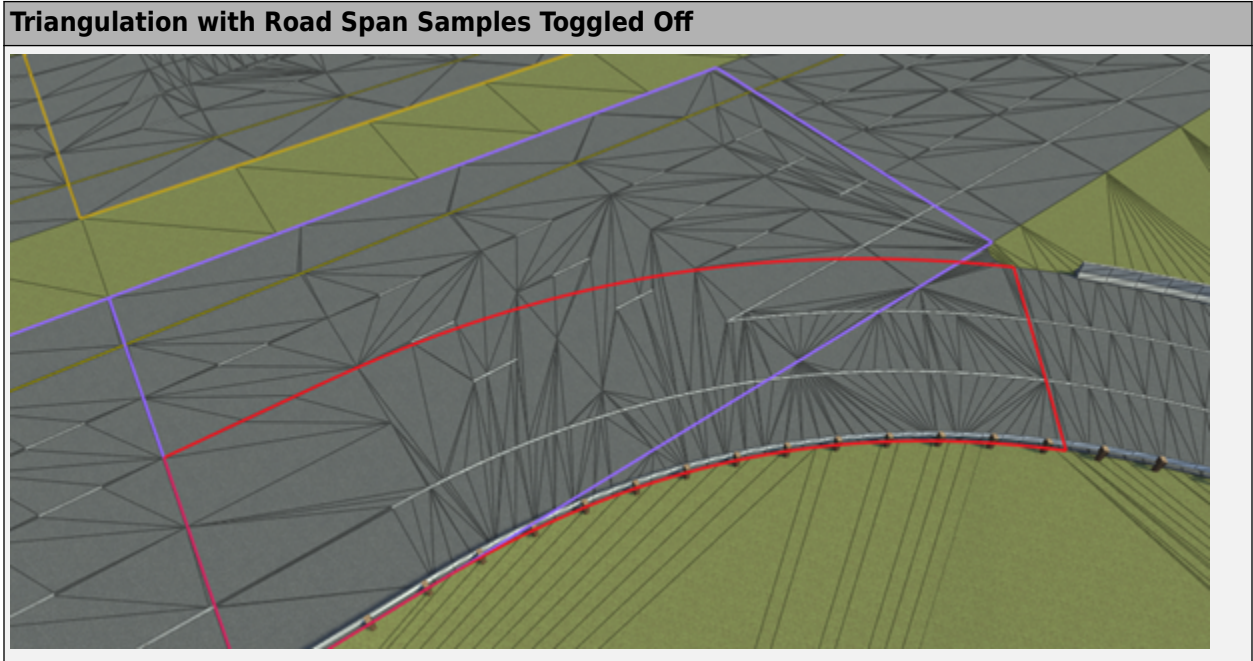
- 1 Click the **Junction Surface Tool** button.
- 2 Select a junction. This selection displays all the individual road surface spans that overlap the junction.
- 3 Click a road span. When selected, a road span draws the portion of the road that overlaps the junction and the samples that it includes.
- 4 Toggle the **Include Samples** check box in the **Attributes** pane to include or exclude the samples.

Triangulation with Road Span Samples Included



Road Span Samples Toggled Off



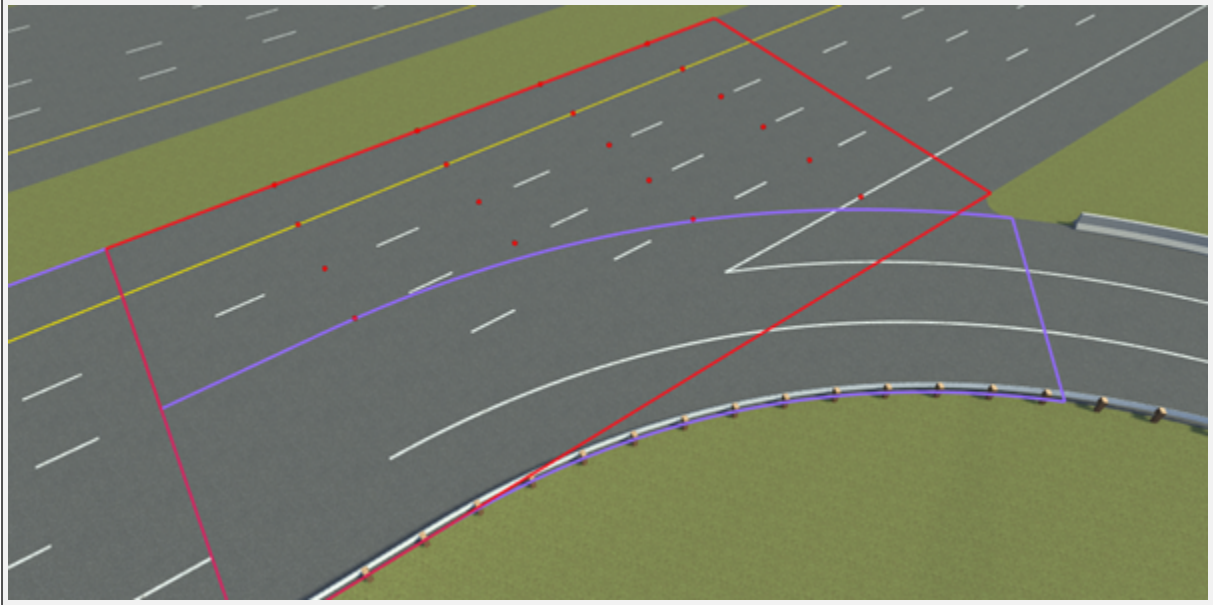


Change the Sorting of Road Span Samples

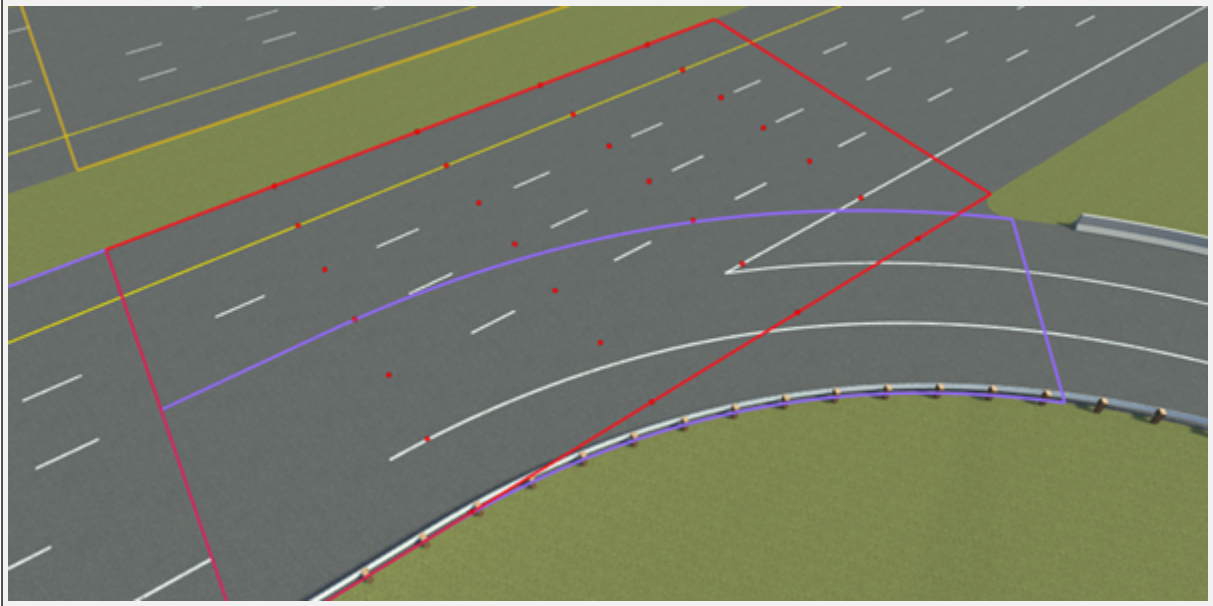
Sorting one span above another prevents the lower samples from being included.

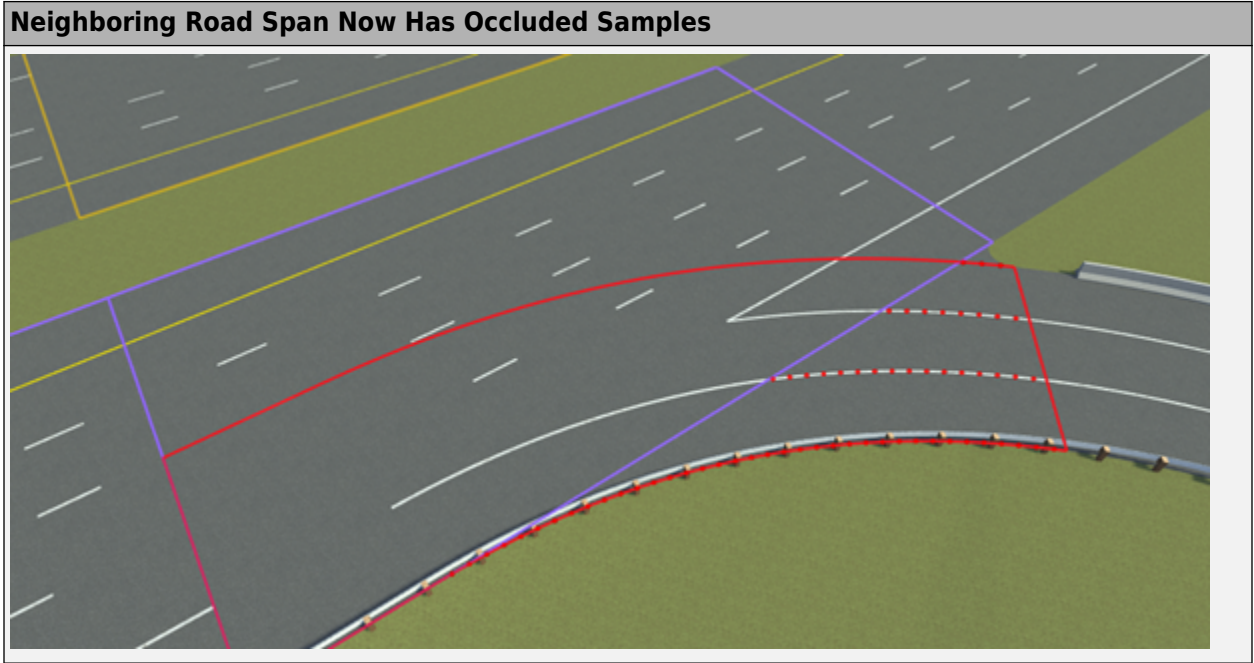
- 1 Click the **Junction Surface Tool** button.
- 2 Select a junction. This selection displays all the individual road surface spans that overlap the junction.
- 3 Click a road span. When selected, a road span draws the portion of the road that overlaps the junction and the samples that it includes.
- 4 Press the **Raise** or **Lower** button in the **Attributes** pane to raise or lower the **Sort Index** of the selected road span relative to the others.

Road Span with Occluded Samples



Road Span Samples Raised in Priority





Version History

Introduced in R2020a

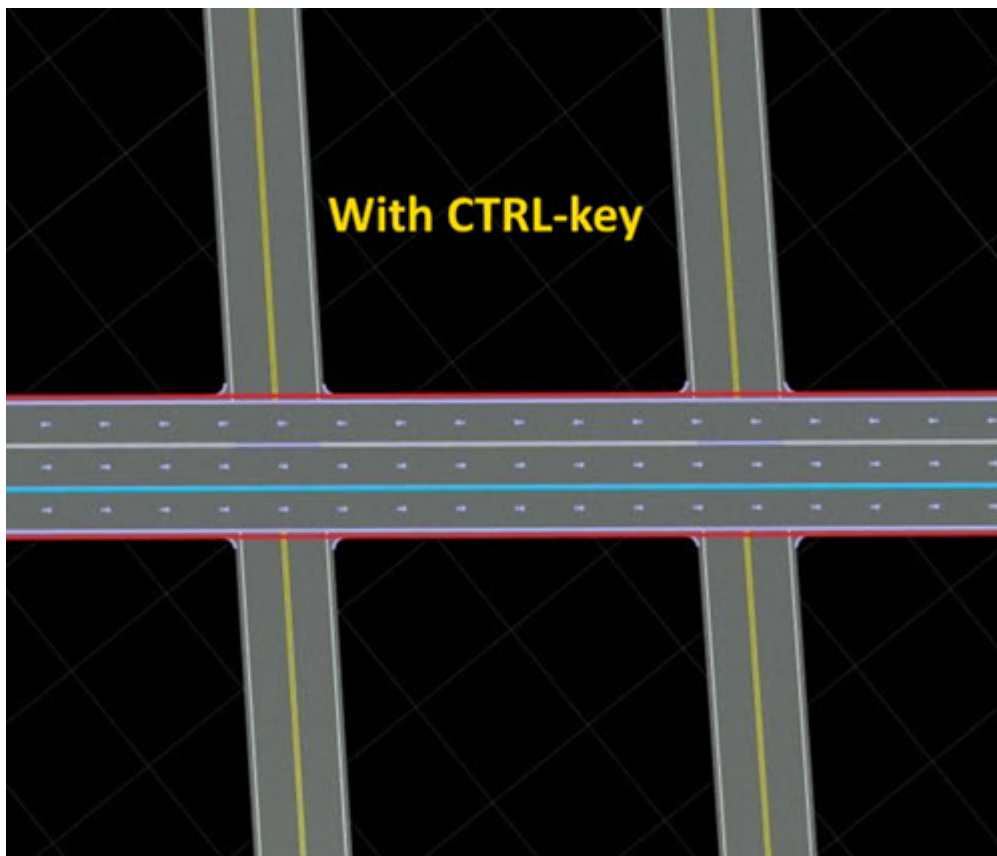
Lane Add Tool

Add fully formed lane along road

Description

The **Lane Add Tool** is used to add a fully formed lane along a road.

Note To create a new forming lane or an ending lane, use the **Lane Form Tool**.



Open the Lane Add Tool

On the RoadRunner toolbar, click the **Lane Add Tool** button:



Examples

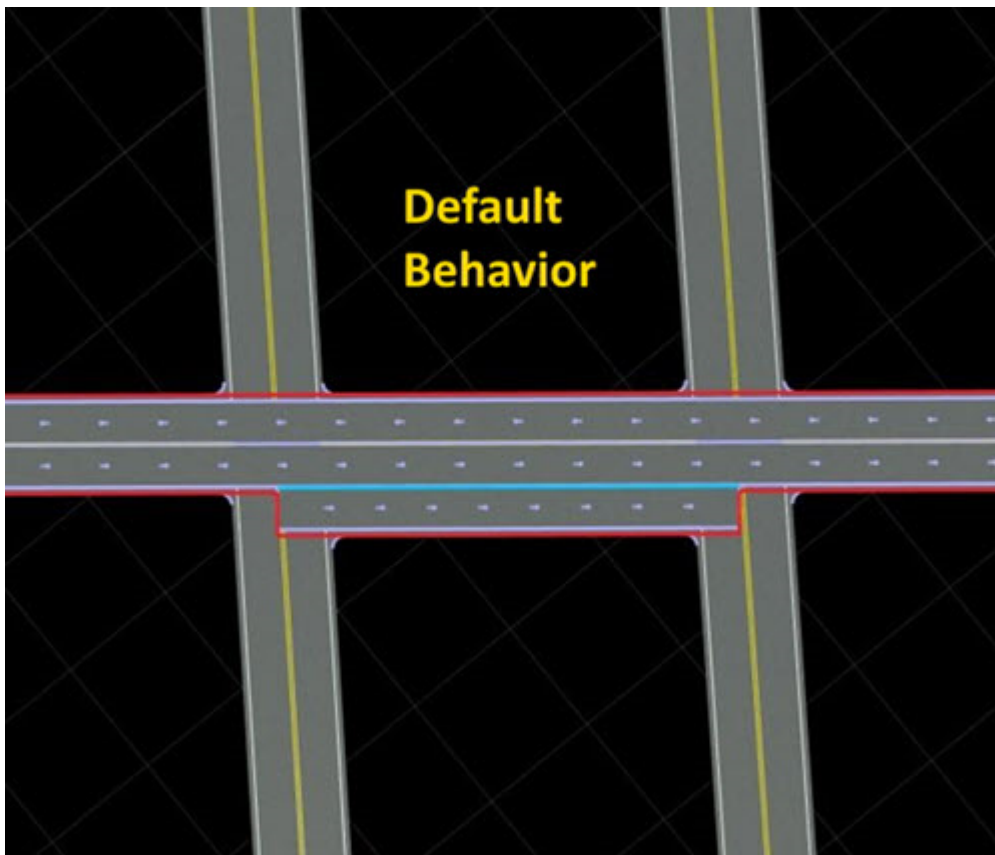
Add a New Lane to a Road

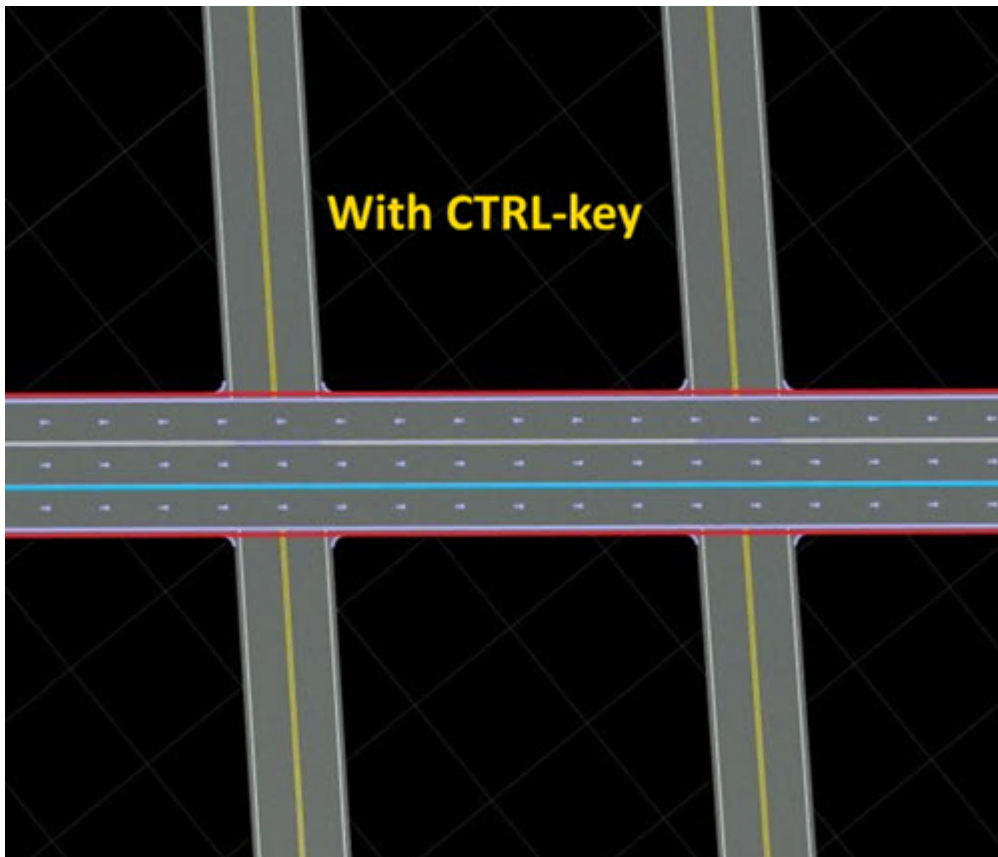
- 1 Click the **Lane Add Tool** button.
- 2 Click the road you want to edit.
- 3 Select the desired lane type in the **Options** pane.

Note If the lane type is set to **Automatic**, then the new lane copies the lane type of the neighboring lane.

- 4 Move the pointer near where you want to add a lane until you see a light blue line indicating where the new lane will be added. If you are pointing near the center reference curve of the road, you can choose which side of the road the new lane will go by moving the pointer to one side or the other of the center curve.

Note By default, the new lane will be added only between the two nearest intersections. To force the new lane to add along the entire road, hold the **Ctrl** key.





- 5 Right-click to add a new lane.
- 6 Optionally, you can hold the right-click button down and drag to adjust the width of the new lane.

Version History

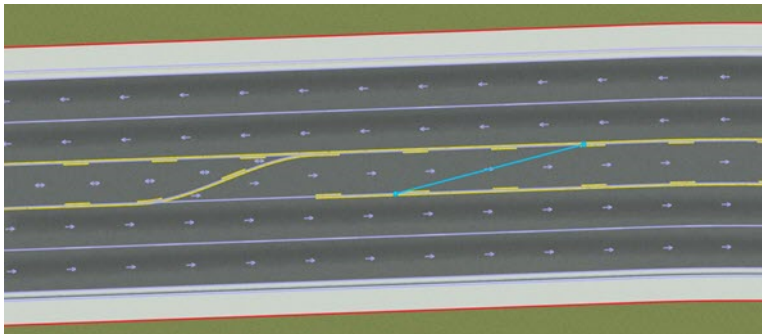
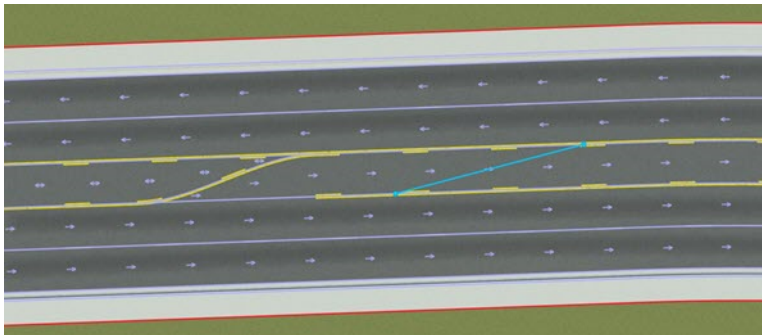
Introduced in R2020a

Lane Carve Tool

Create tapering cut in lane, such as dedicated turn lane in median

Description

The **Lane Carve Tool** is used to create a tapering cut in a lane, such as the dedicated turn lanes in a median.



Open the Lane Carve Tool

On the RoadRunner toolbar, click the **Lane Carve Tool** button:



Examples

Carve a Tapering Cut Into a Lane

- 1 Click the **Lane Carve Tool** button.
- 2 Click the road you want to edit.

- 3 Move the pointer to where you want to start the carve. The lane highlights and a light blue point indicates where the carving will start.
- 4 Right-click and drag the pointer to carve the lane. You can move up or down the road and left or right along the lane to determine where the carving ends. A light blue curve indicates where the carve will take place.

Note A lane cut always starts at the boundary of a lane. It can end either at the opposite side of the lane or in the middle of the lane.

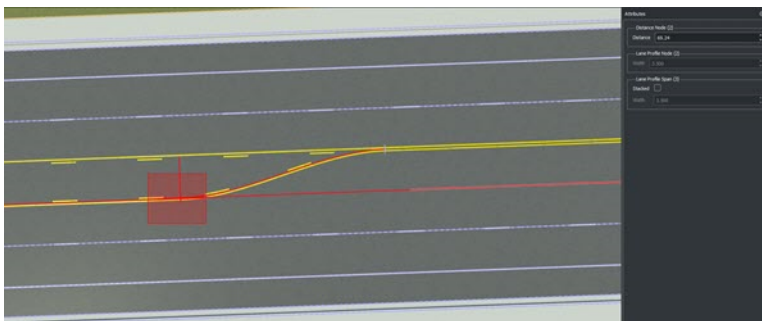
- 5 When you have moved to the desired end location of the carve, release the right-click button.

Modify the Carve Location After Performing a Carve

If you have performed a lane carve and later want to change where the carve starts or ends, follow the steps in *Move the End of a Lane* on page 1-56 and adjust the ends of a lane by using the **Lane Width Tool**.

Depending on which end of the carve you are adjusting, you might need to move both the tapered end of one lane and the width marker on the adjacent lane. These moves can be tricky, but the simplest approach is to box-select both elements together before dragging. For more details on box-selection, see “Manipulate Scene Objects”.

This image shows a box selection of two UI elements together. After selecting, you can click and drag them or adjust the **Distance** value in the **Attributes** pane.



Modify the Lane Widths After Performing a Carve

See the **Lane Width Tool**.

Version History

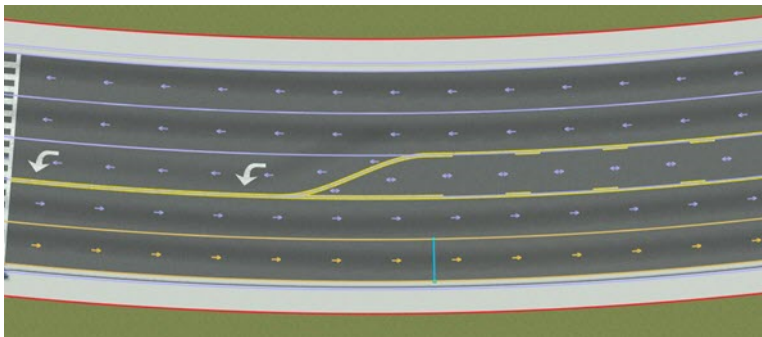
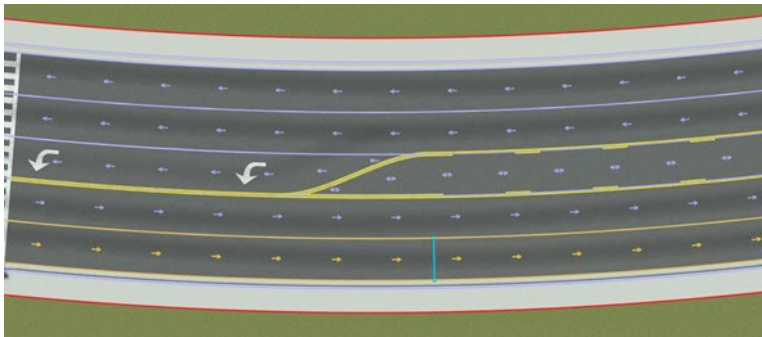
Introduced in R2020a

Lane Chop Tool

Cut single lane into two lanes

Description

The **Lane Chop Tool** can be used to cut a single lane into two lanes at a desired location. Chopping does not have an immediately visible effect, but it enables you to make instantaneous changes in lane properties or have lanes start or end abruptly.



Open the Lane Chop Tool

On the RoadRunner toolbar, click the **Lane Chop Tool** button:



Examples

Chop a Lane

- 1 Click the **Lane Chop Tool** button.

- 2** Click the road you want to edit.
- 3** Move the mouse cursor to the location you want to chop. You will see a light blue line indicating where the chop operation will take place.
- 4** Right-click to chop the lane at the desired location.

Version History

Introduced in R2020a

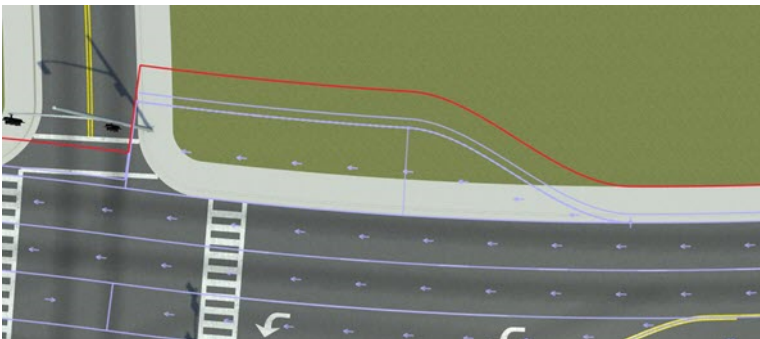
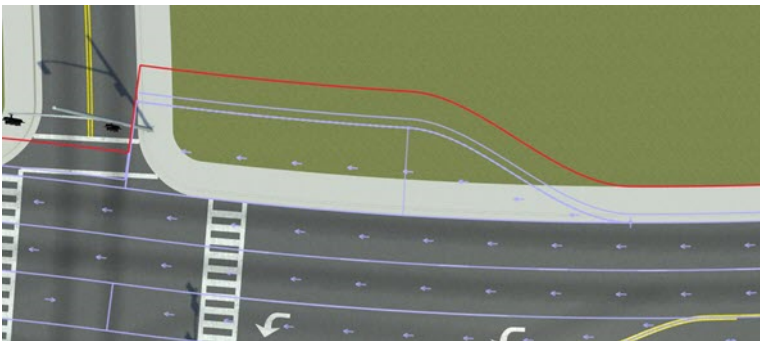
Lane Form Tool

Add forming or ending lane along road

Description

The **Lane Form Tool** is used to add a forming or ending lane along a road.

Note To create a fully formed lane, use the **Lane Add Tool**.



Open the Lane Form Tool

On the RoadRunner toolbar, click the **Lane Form Tool** button:



Examples

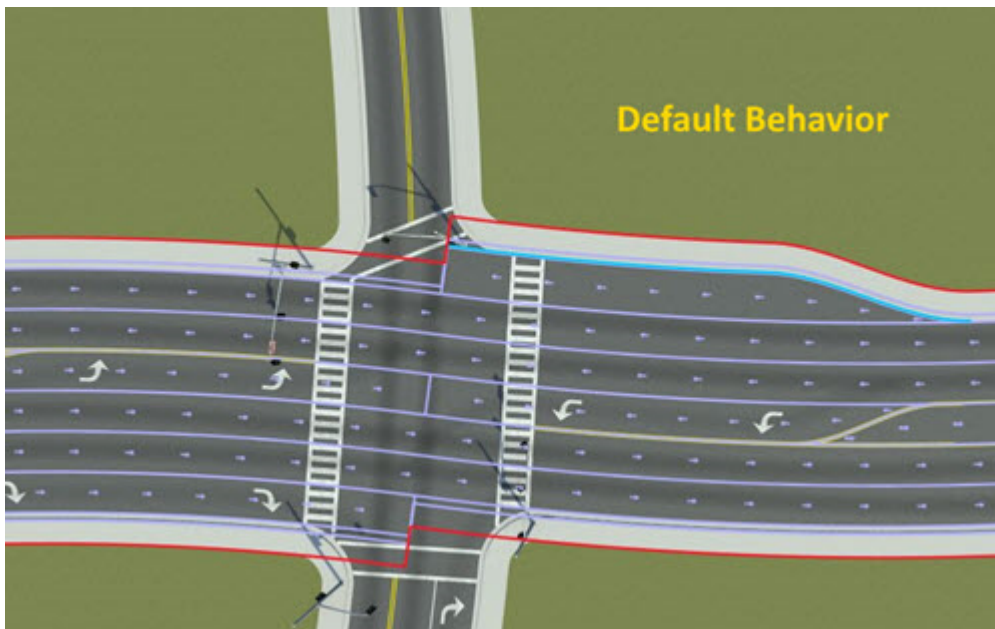
Add a New Lane to a Road

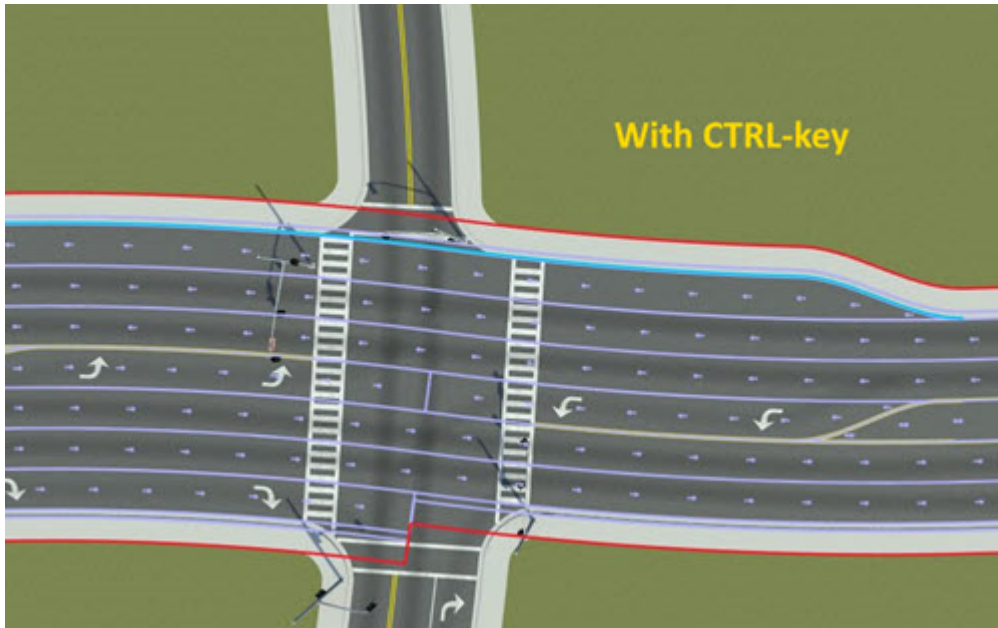
- 1 Click the **Lane Form Tool** button.
- 2 Click the road you want to edit.
- 3 Select the desired lane type in the **Options** pane.

Note If the lane type is set to **Automatic**, then the new lane copies the lane type of the neighboring lane.

- 4 Move the pointer near where you want to add a lane until you see a light blue line indicating where the new lane will be added. If you are pointing near the center reference curve of the road, you can choose which side of the road the new lane will go by moving the pointer to one side or the other of the center curve.

By default, the new lane will be added only between the two nearest intersections. To force the new lane to add along the entire road, hold the **Ctrl** key.





- 5** Right-click and drag outward from the center of the road to create the new lane and adjust the end of the tapering section.
- 6** Optionally, you can drag up or down the length of the road to switch between a forming lane and an ending lane.

Version History

Introduced in R2020a

Lane Marking Tool

Add linear markings to lane boundaries

Description

The **Lane Marking Tool** adds linear markings to lane boundaries. To assign marking styles to a lane marking, you must first create some marking styles in the Asset Browser. The RoadRunner sample project has several common marking styles pre-defined in the Assets/Markings directory. You can create and modify your own marking styles and add them to the project as well.



Open the Lane Marking Tool

On the RoadRunner toolbar, click the **Lane Marking Tool** button:



Examples

Create and Modify Lane Markings Along a Lane

See “Span Editing”.

Note Lane marking spans store **Lane Marking Assets** data, which can be directly dragged onto a lane span from the **Library Browser**. This operation works in any tool. The Lane Marking Tool is activated when the mouse is released.

Parameters

See **Marking Curve Tool** attributes.

Version History

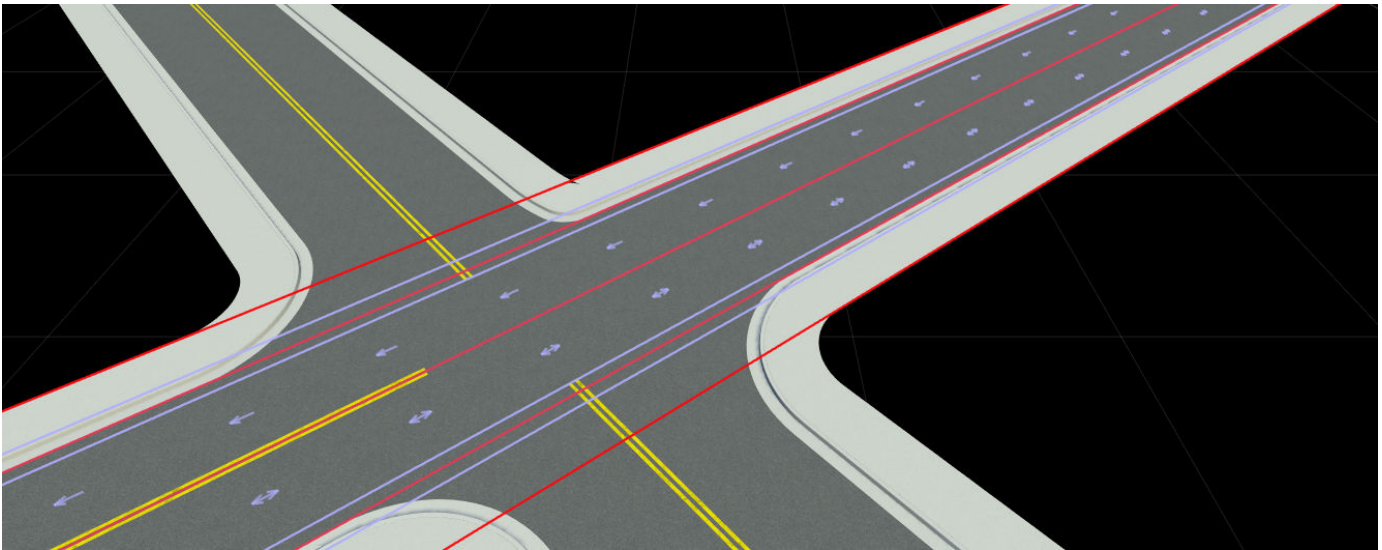
Introduced in R2020a

Lane Offset Tool

Adjust location of center lane of road

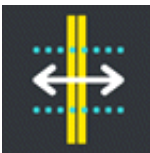
Description

The **Lane Offset Tool** is used to adjust the location of the center lane of a road. This tool is nearly identical to the **Lane Width Tool**, except that it operates only on the center lane. For more information on usage, see the **Lane Width Tool**.



Open the Lane Offset Tool

On the RoadRunner toolbar, click the **Lane Offset Tool** button:



Version History

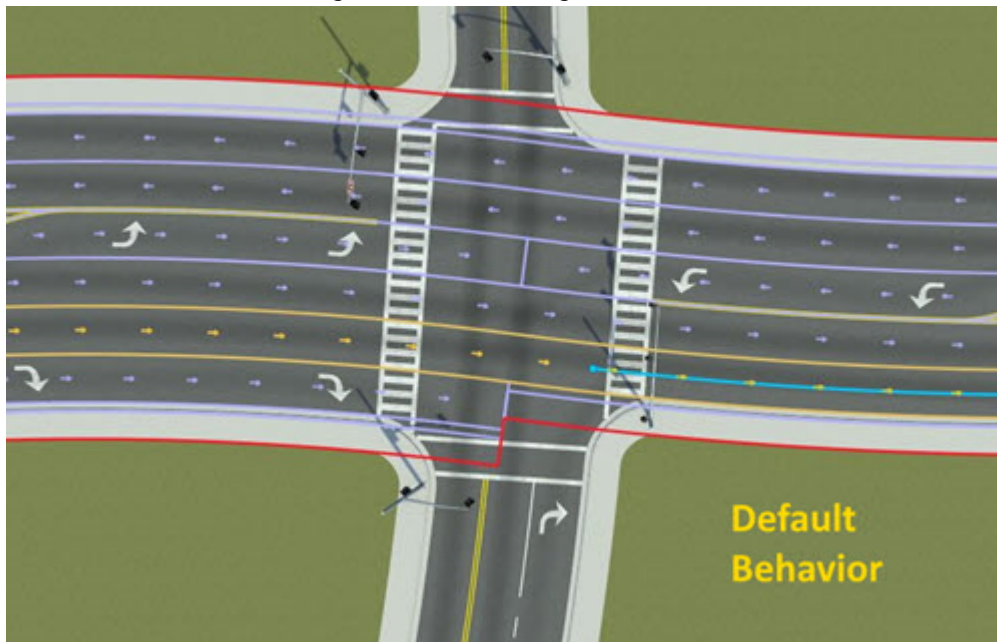
Introduced in R2020a

Lane Split Tool

Split lane lengthwise into two lanes

Description

The **Lane Split Tool** is used to split a lane lengthwise into two lanes. Splitting a lane automatically adds a default lane marking that can be changed or removed with the **Lane Marking Tool**.



Open the Lane Split Tool

On the RoadRunner toolbar, click the **Lane Split Tool** button:

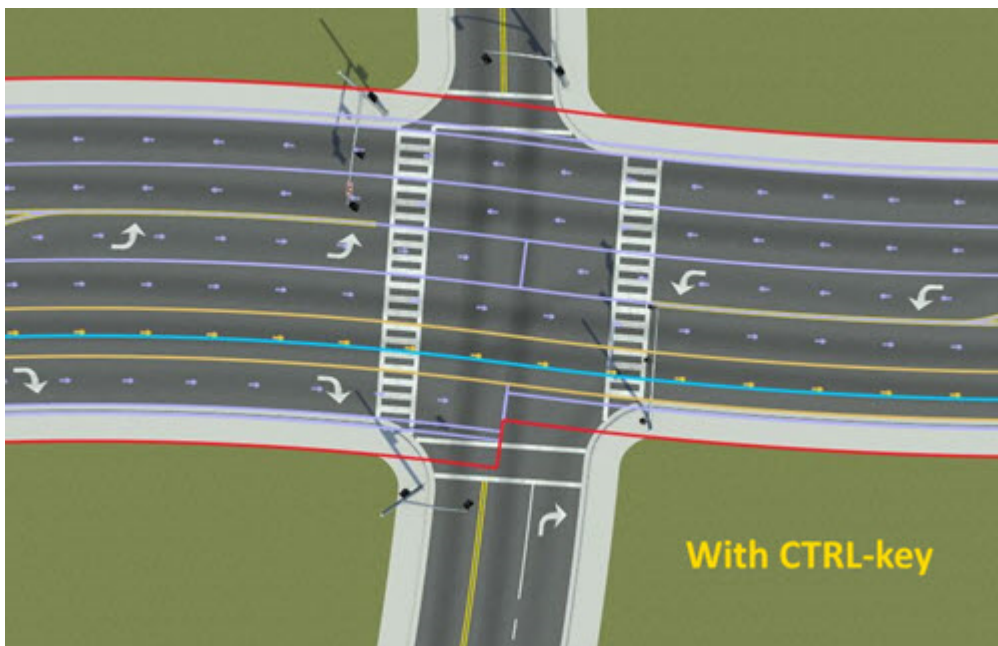
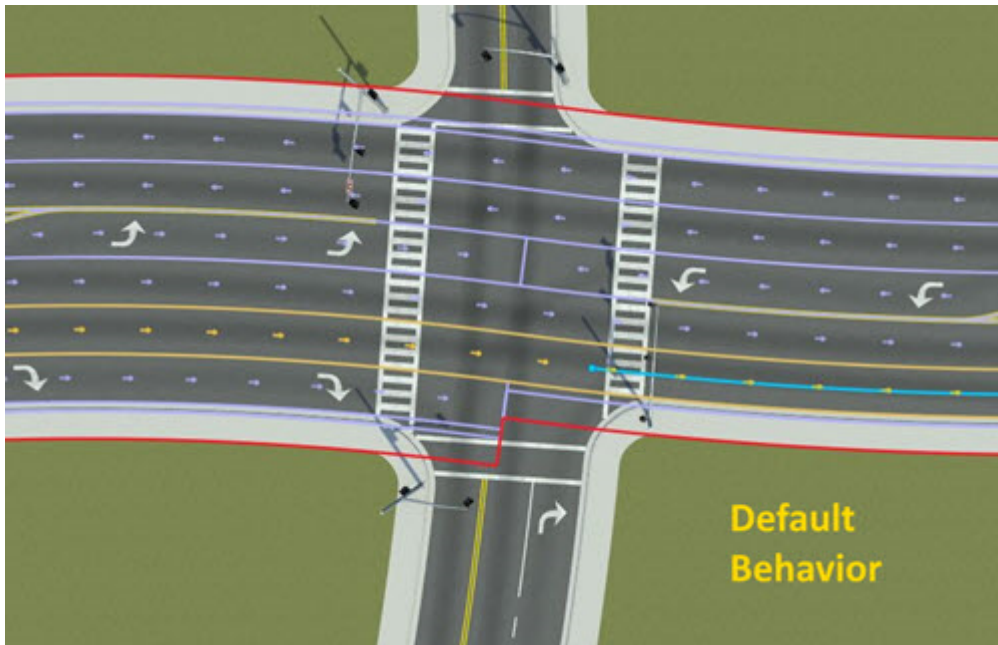


Examples

Split a Lane Lengthwise

- 1 Click the **Lane Split Tool** button.
- 2 Click the road you want to edit.
- 3 Move the pointer to the location you want to split. A light blue line indicates where the split operation will take place.

By default, the lane split affects only the lane between the two nearest intersections. To force the lane split to occur along the entire road, hold the **Ctrl** key.



- 4 Right-click to split the lane.

Version History

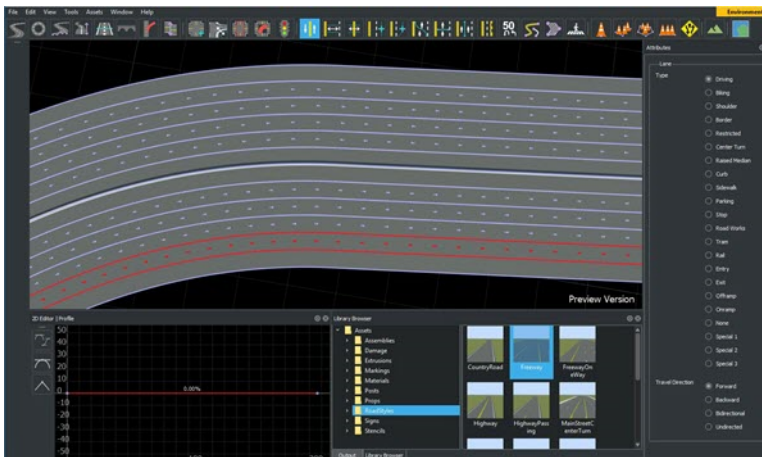
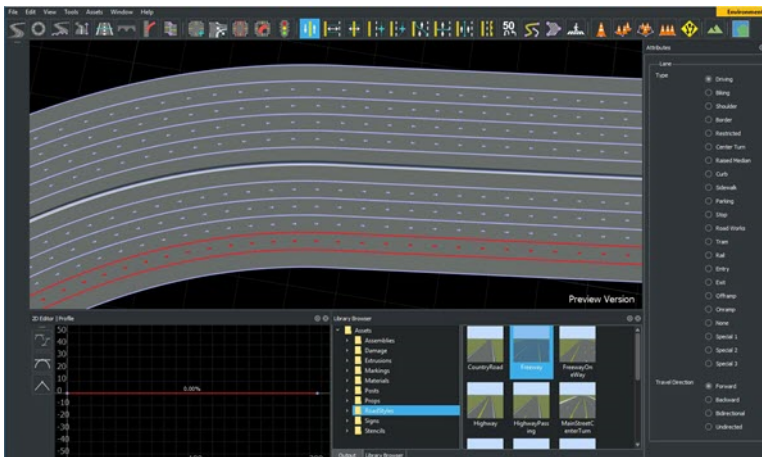
Introduced in R2020a

Lane Tool

Delete lanes, update lane type, and reverse lane travel direction

Description

The **Lane Tool** is used to delete lanes, make changes to lane attributes, such as the lane type, and reverse lane travel directions.



Open the Lane Tool

On the RoadRunner toolbar, click the **Lane Tool** button:



Examples

Change Lane Type

- 1 Click the **Lane Tool** button.
- 2 Click the road containing the target lane.
- 3 Click a lane to display its attributes. The selected lane is highlighted in red.
- 4 Select a lane type in the **Attributes** pane.

Change Lane Travel Direction

- 1 Click the **Lane Tool** button.
- 2 Click the road containing the target lane.
- 3 Click a lane to display its attributes. The selected lane is highlighted in red.
- 4 Select a travel direction in the **Attributes** pane.

Delete a Lane

- 1 Click the **Lane Tool** button.
- 2 Click the road containing the target lane.
- 3 Click a lane.
- 4 Press the **Delete** key or select **Edit > Delete** from the menu bar.

Reverse Lane Travel Direction

- 1 Click the **Lane Tool** button.
- 2 Click the road containing the target lane.
- 3 Click a lane whose travel direction you want to reverse. The selected lane is highlighted in red. You can also select multiple roads and lanes to reverse their directions simultaneously.
- 4 On the toolbar to the left of the scene editing canvas, click the **Reverse Travel Direction** button.



This reverses the travel direction of the selected lanes.

- Forward travel direction lanes change to backward travel direction lanes.
- Backward travel direction lanes change to forward travel direction lanes.
- Bidirectional and undirected lanes are unaffected.

Version History

Introduced in R2020a

Lane Width Tool

Adjust lane widths along road

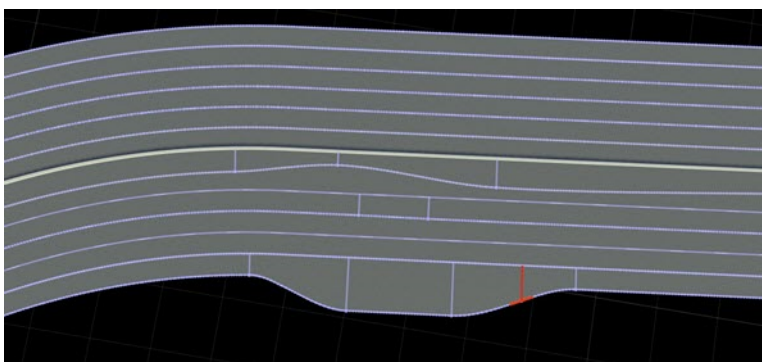
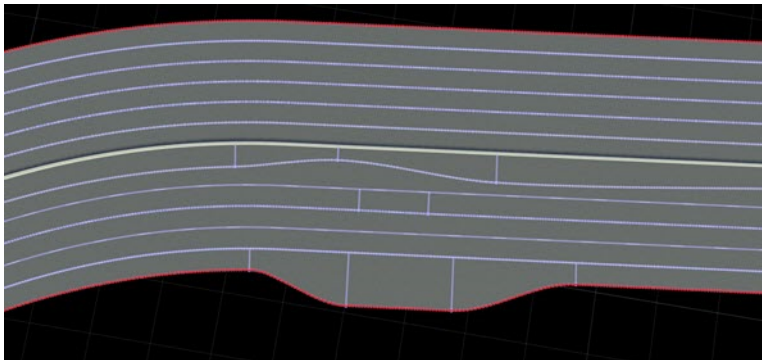
Description

The **Lane Width Tool** is used to adjust lane widths at any location along a road. The width of individual lanes can be varied across the entire lane or more locally at specified locations. Width values are stored on lane width markers, which can be positioned independently along lanes. The width of the lane on the sections between markers is interpolated from the marker widths.

All lanes will automatically have lane width markers at the beginning and end of the lane.

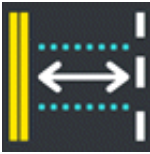
When the **Lane Width Tool** is selected and a road is highlighted, all the lane width markers for the road will be displayed.

Note The **Lane Width Tool** does not allow adjustments to the center lane. To adjust the center lane, use the **Lane Offset Tool**.



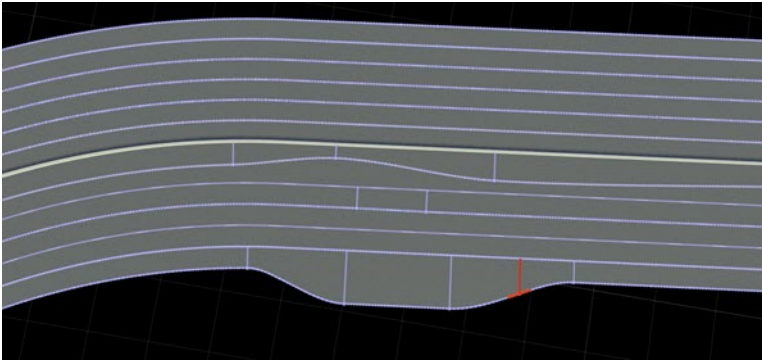
Open the Lane Width Tool

On the RoadRunner toolbar, click the **Lane Width Tool** button:



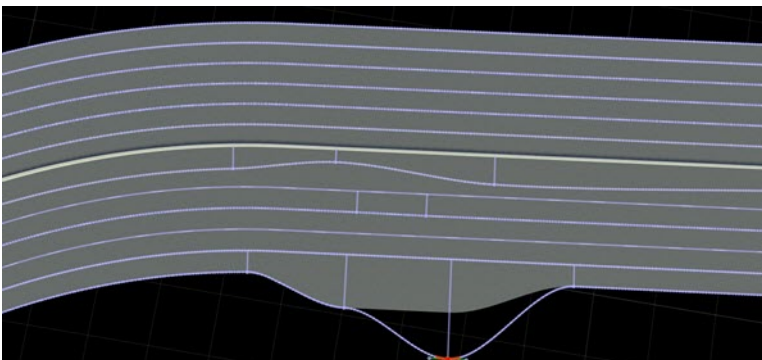
Examples

Create a New Lane Width Marker



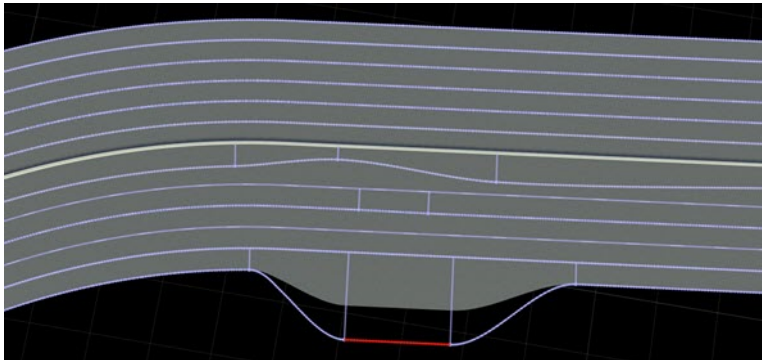
- 1 Click the **Lane Width Tool** button.
- 2 Click the road you want to edit.
- 3 Move the pointer over the lane where you want to insert the marker. A light blue line appears, indicating where the marker will be inserted.
- 4 Right-click to add a new width marker.
- 5 Optionally, press the right-click button and drag to adjust the marker. If you initially right-click near the lane boundary, you can drag the lane width at the marker. If you initially right-click near the inside of the lane, you can drag the position of the width marker.

Adjust the Width at a Marker



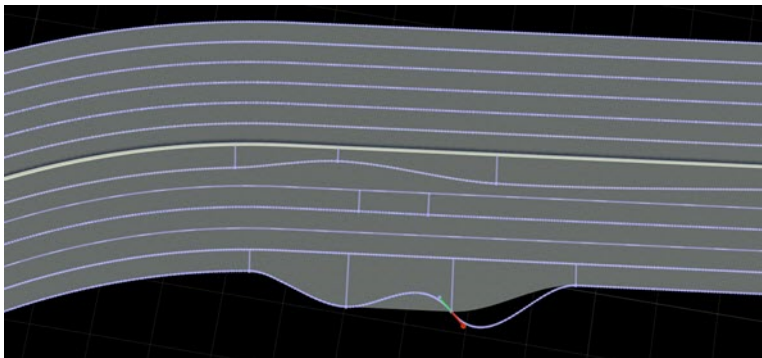
- 1 Click the **Lane Width Tool** button.
- 2 Click the road you want to edit. The lane width markers will be displayed on the picked road.
- 3 Click and drag the point on the outer boundary of the width marker you want to edit.
- 4 Optionally, once the width marker is selected, you can type the desired width directly into the **Width** slider in the **Attributes** pane.

Adjust the Width of a Lane Section Between Two Markers



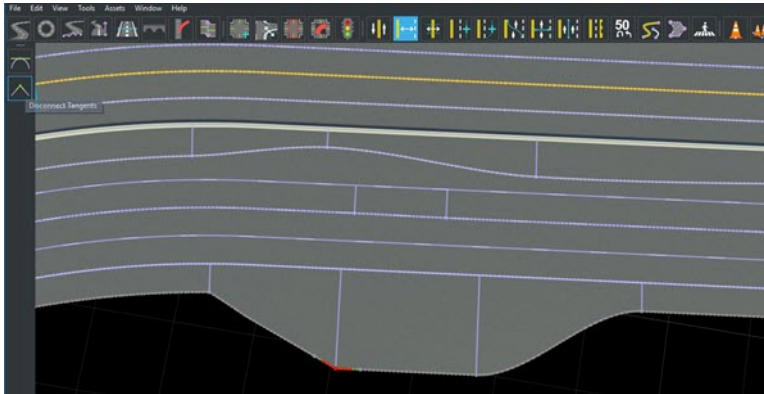
- 1 Click the **Lane Width Tool** button.
- 2 Click the road you want to edit. The lane width markers and lane boundary lines are displayed for the selected road.
- 3 Click and drag the lane border curve to move it in or out. The action automatically adjusts the markers at the start and end of the section.
- 4 Optionally, once the lane section is picked, you can type the desired width directly into the **Width** slider in the **Attributes** pane to set the width at the start and end of the lane section.

Adjust the Angle at a Marker



- 1 Click the **Lane Width Tool** button.
- 2 Click the road you want to edit. The lane width markers are displayed on the selected road.
- 3 Click a lane width marker. Two new angle points are displayed.
- 4 Click and drag an angle point to adjust the slope.
- 5 Optionally, once the angle point is picked, you can type the desired slope value directly into the **Slope** slider in the **Attributes** pane.

Create Sharp Angles

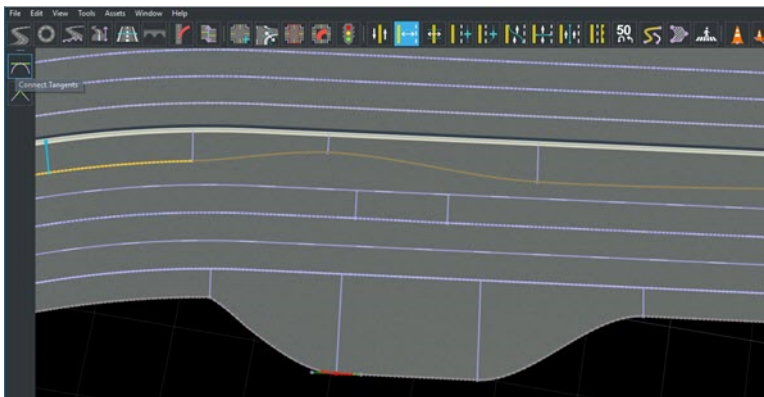


By default, the slope on either side of a lane width marker is kept continuous. You can create sharp angles by disconnecting the slopes as follows:

- 1 Click the **Lane Width Tool** button.
- 2 Click the road you want to edit. The lane width markers are displayed on the selected road.
- 3 Click a lane width marker. Two new angle points are displayed.
- 4 Click the **Disconnect Tangents** button. The angle points are no longer kept continuous, and you can control the slopes on either side independently.

Tip When you click **Disconnect Tangents**, the slopes are always set to point at the next or previous width marker, even if the slopes are already disconnected.

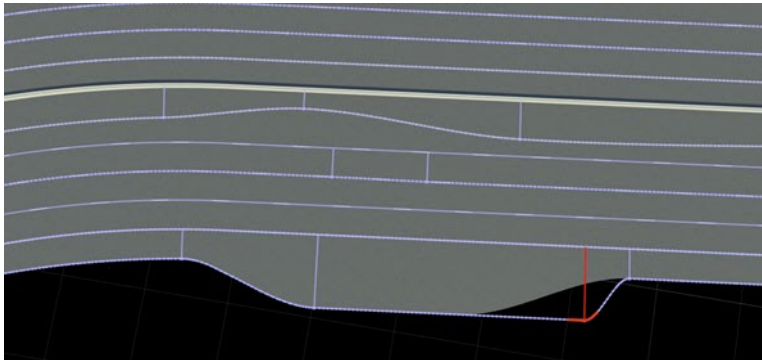
Remove Sharp Angles



To convert a sharp angle into a smooth angle at a lane width marker, connect the slopes as follows:

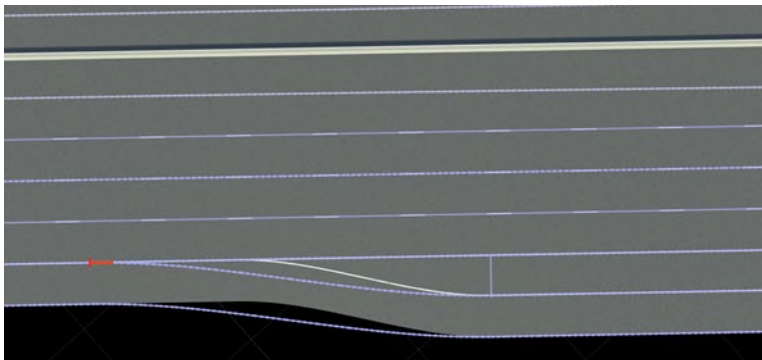
- 1 Click the **Lane Width Tool** button.
- 2 Click the road you want to edit. The lane width markers are displayed on the selected road.
- 3 Click a lane width marker. Two new angle points are displayed.
- 4 Click the **Connect Tangents** button. The angle points are now kept continuous.

Move a Width Marker



- 1 Click the **Lane Width Tool** button.
- 2 Click the road you want to edit. The lane width markers are displayed on the selected road.
- 3 Click and drag near the middle of the line of the width marker that you want to move.

Move the End of a Lane



- 1 Click the **Lane Width Tool** button.
- 2 Click the road you want to edit. The lane width markers will be displayed on the selected road.
- 3 Click and drag the horizontal tick at the end of the lane.

Note It can be difficult to select the horizontal tick because the UI favors selecting the width point at the same location. Rather than clicking directly on the tick, move the pointer to the side of the tick until the tick is highlighted in yellow.

Delete a Width Marker

- 1 Click the **Lane Width Tool** button.
- 2 Click the road you want to edit. The lane width markers and lane boundary lines are displayed on the selected road.
- 3 Press the **Delete** key or select **Edit > Delete** from the menu bar.

Note You cannot delete the markers at the start or end of the entire lane.

More About

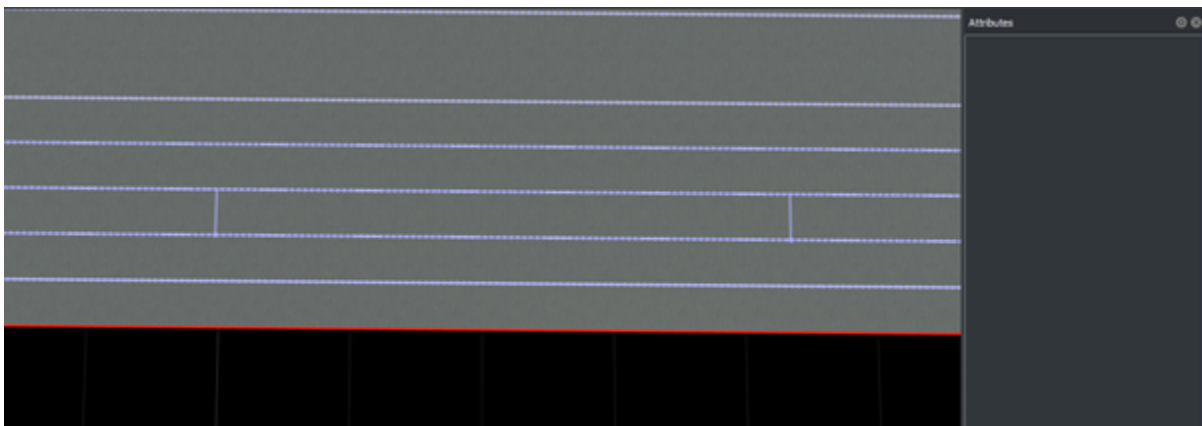
Stacked and Unstacked Boundaries

Lane boundaries behave differently depending on whether they are stacked or not.

Stacked boundaries are offset from their neighboring lanes. In this way, adjusting a lane boundary affects all lanes facing outward from the center lane.

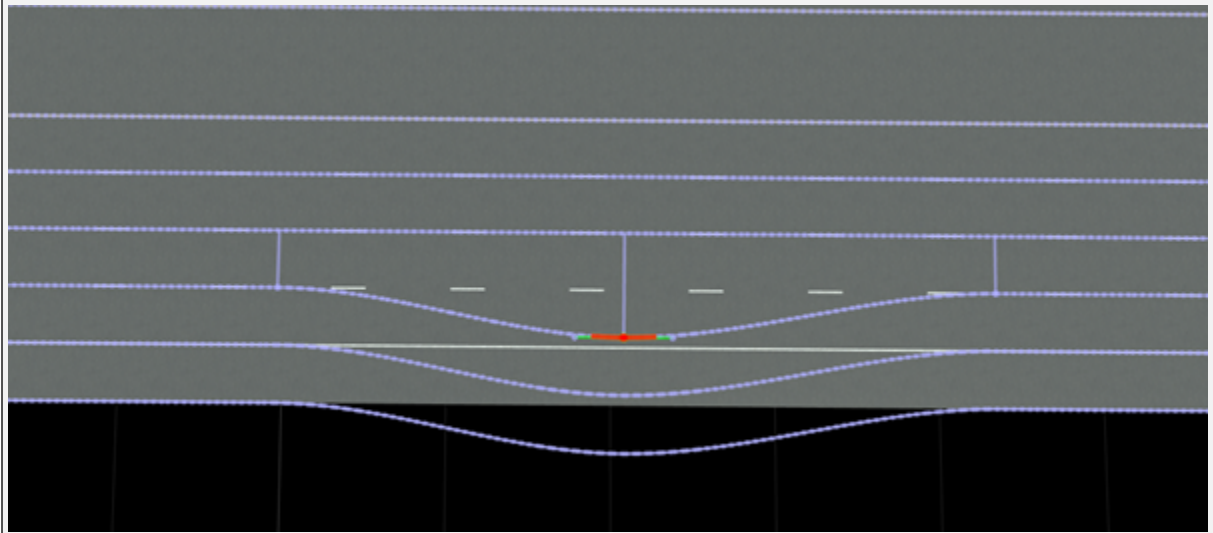
When a boundary is not stacked, its width is determined as an offset from the road's center lane, not its neighboring lanes.

Set Stacked and Unstacked Behavior

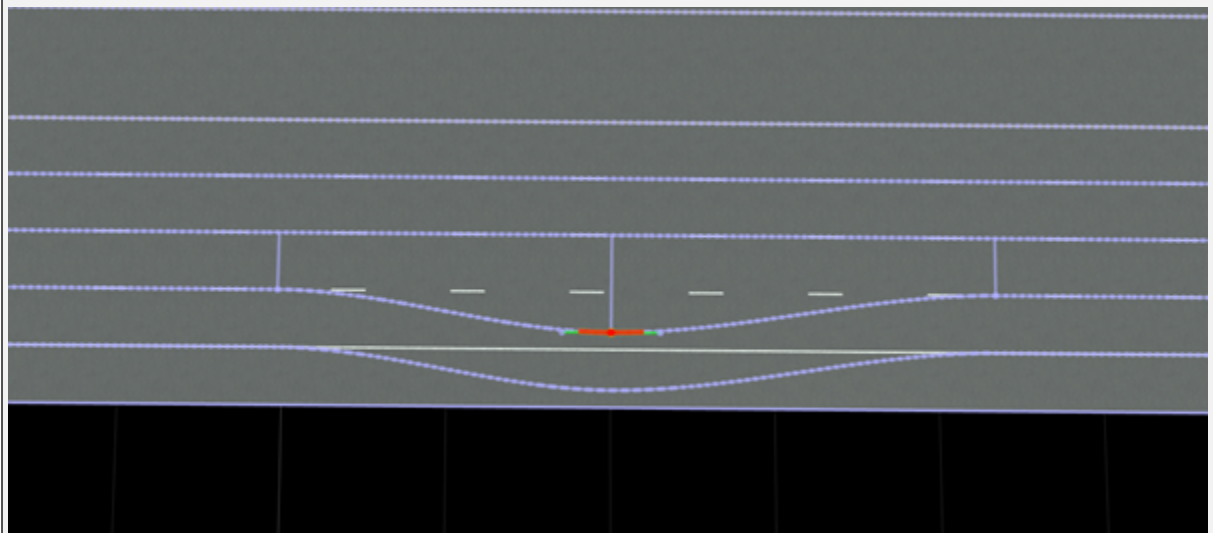


- 1 Click the **Lane Width Tool** button.
- 2 Click the road containing the target lane.
- 3 Click a lane boundary to select it and display its attributes. The selected lane boundary is highlighted in red.
- 4 Toggle the Stacked attribute in the **Attributes** pane.

Moving a lane where the outermost boundary is stacked



Moving a lane where the outermost boundary is not stacked



Version History

Introduced in R2020a

Maneuver Tool

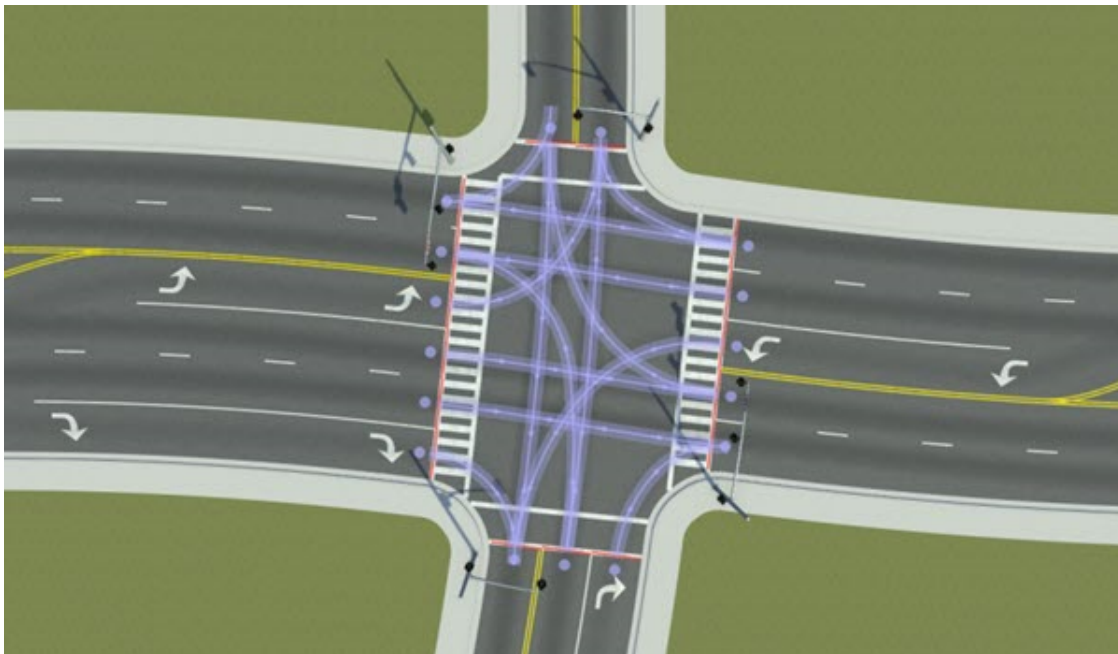
Manipulate individual maneuver roads (paths) within junction

Description

Maneuver roads represent the movement of a car through an intersection. The **Maneuver Tool** enables the manipulation of the various individual maneuver roads (paths) within a junction. These maneuver roads do not affect the geometry of the road model, but they do affect traffic behavior. Maneuver roads export to formats such as ASAM OpenDRIVE.

By default, whenever roads cross in a junction, the RoadRunner software creates maneuver roads automatically and makes reasonable assumptions about which roads to connect by maneuvers. However, it is occasionally necessary to add or remove maneuvers manually. The **Maneuver Tool** enables you to do so.

Maneuver roads are similar to normal roads, but they have certain restrictions. Maneuver roads are slip roads at both ends, which means that their start and end locations and directions are constrained to align with the anchor roads that they are attached to. For more details on slip roads, see the **Slip Road Tool**.



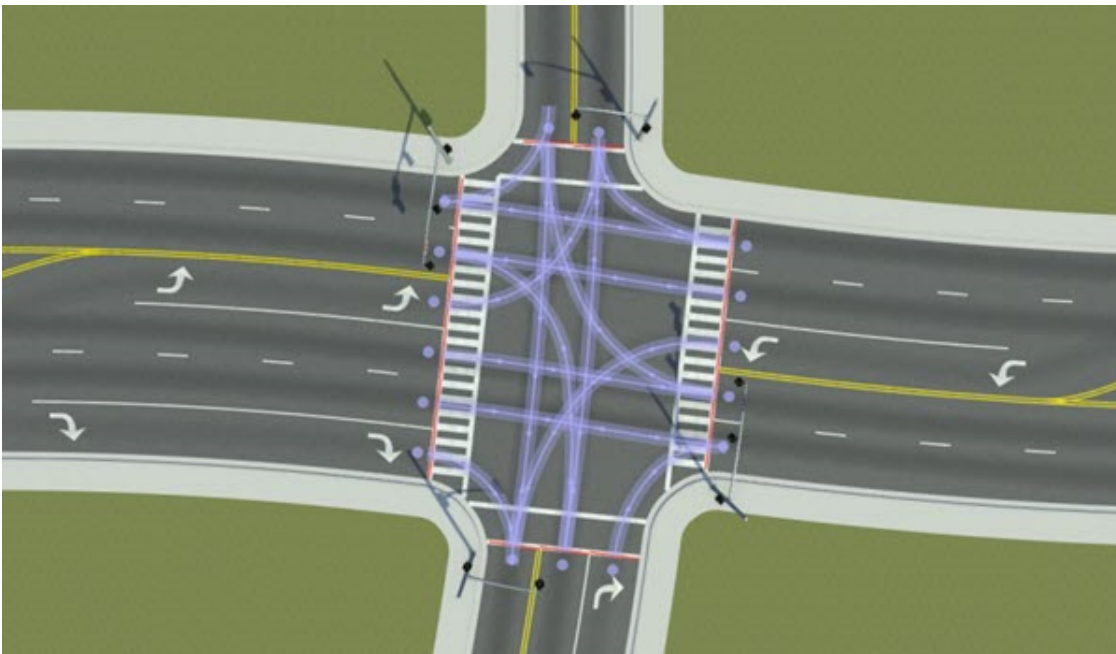
Open the Maneuver Tool

On the RoadRunner toolbar, click the **Maneuver Tool** button:



Examples

View Maneuver Roads Within Junction



- 1 Click the **Maneuver Tool** button.
- 2 Select a junction. This selection displays all the individual maneuvers within the junction.

Enable or Disable Automatic Maneuver Creation

By default, maneuver roads within a junction are automatically created or removed. Various operations cause maneuver roads to be recomputed, such as moving road geometry, adding or removing lanes, or changing lane travel directions.

Turn Automatic Maneuver Creation Off

Add a new maneuver road on page 1-61 or delete a maneuver road. These actions disable automatic maneuver creation for the junction.

Turn Automatic Maneuver Creation On

- 1 Click the **Maneuver Tool** button.
- 2 Click the junction you want to examine. When a junction is selected, it displays all of the individual maneuvers allowed within the junction.
- 3 Click **Rebuild Maneuver Roads** to recreate all maneuver roads and re-enable automatic maneuver creation.

Enable or Disable Automatic Maneuver Geometry

By default, maneuver road geometry is automatically updated. Various operations cause maneuver road geometry to be updated, such as moving road geometry or changing lane widths.

Turn Automatic Maneuver Geometry On

Perform an action that modifies the geometry of the maneuver road, such as adjusting the start or end location of a maneuver road on page 1-62 or moving a maneuver road control point on page 1-62. The geometry of that maneuver road locks and no longer updates automatically.

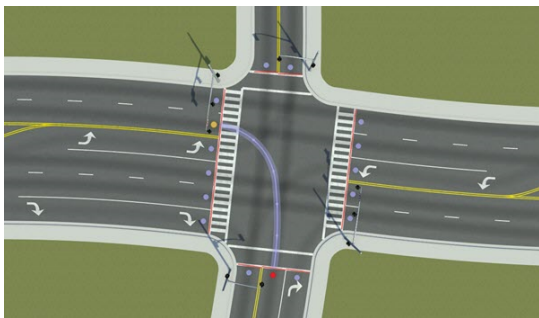
Alternatively, follow these steps:

- 1 Click the **Maneuver Tool** button.
- 2 Click the junction you want to examine. When a junction is selected, it displays all the individual maneuvers allowed within the junction.
- 3 Click the maneuver road you want to edit.
- 4 Select the **Lock Geometry** check box.

Turn Automatic Maneuver Geometry Off

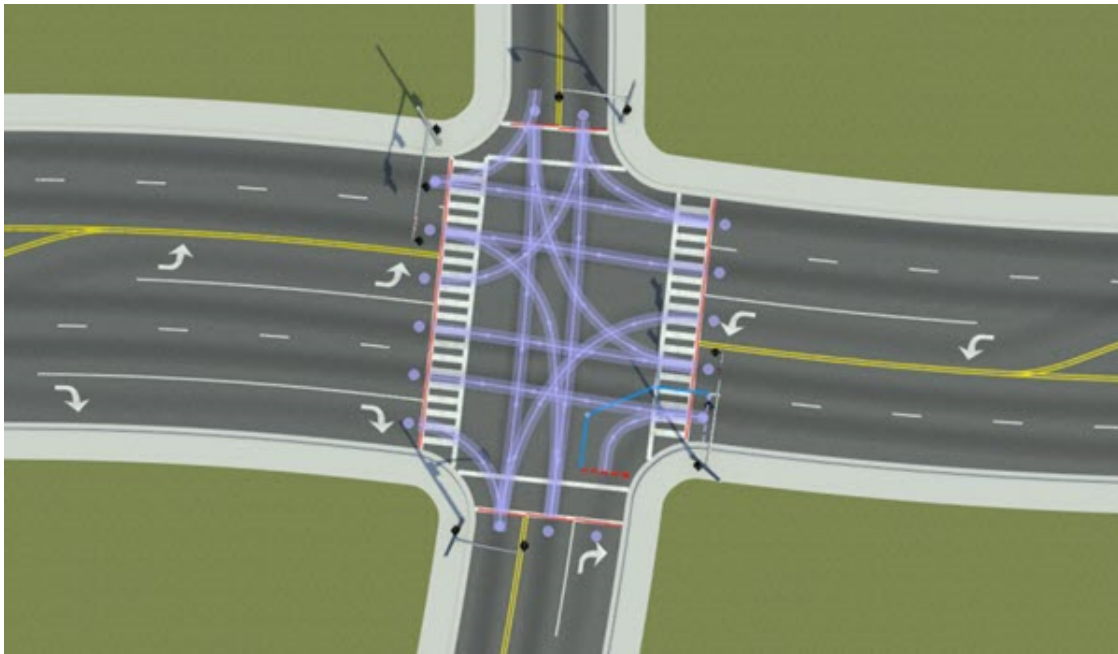
- 1 Click the **Maneuver Tool** button.
- 2 Click the junction you want to examine. When a junction is selected, it displays all the individual maneuvers allowed within the junction.
- 3 Click the maneuver road you want to edit.
- 4 Clear the **Lock Geometry** check box.

Add New Maneuver Road



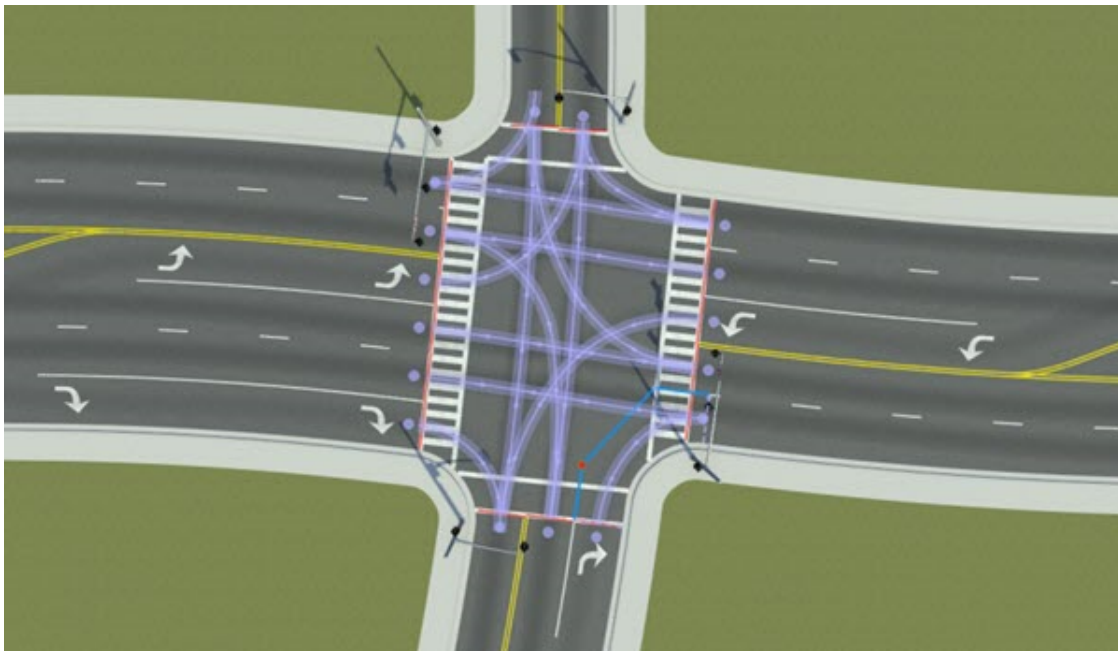
- 1 Click the **Maneuver Tool** button.
- 2 Click the junction you want to edit. All the individual maneuvers within the junction are displayed.
- 3 Click the node point where you want the maneuver to begin.
- 4 Right-click the node point where you want the maneuver to end. The new maneuver road is created and visible.

Adjust Start or End Location of Maneuver Road



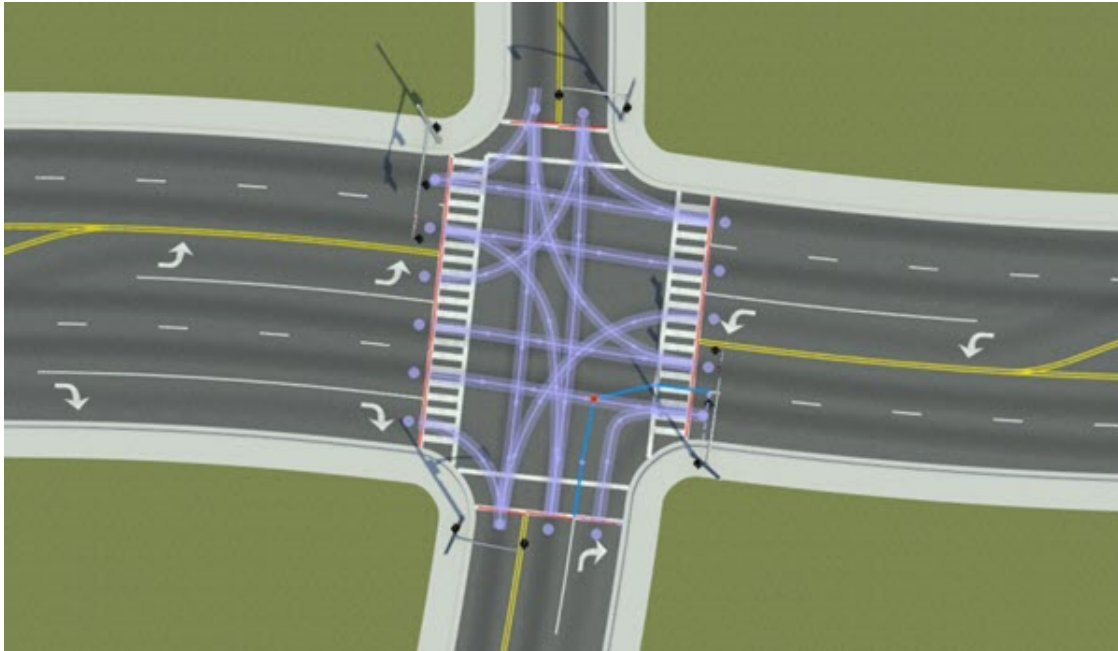
- 1 Click the **Maneuver Tool** button.
- 2 Click the junction you want to edit. All of the individual maneuvers within the junction are displayed.
- 3 Click the maneuver road you want to edit.
- 4 Click and drag the start or end line to adjust the shape of the maneuver road. Each line is constrained to lie along the anchor road the line is attached to.

Move Maneuver Road Control Point



- 1 Click the **Maneuver Tool** button.
- 2 Click the junction you want to edit. All the individual maneuvers within the junction are displayed.
- 3 Click the maneuver road you want to edit.
- 4 Click and drag the desired control point to move it. Because maneuver roads are slip roads, the first and last control points are constrained to lie along a fixed direction. For more details on slip roads, see the **Slip Road Tool**.

Insert a New Control Point Within an Existing Maneuver Road

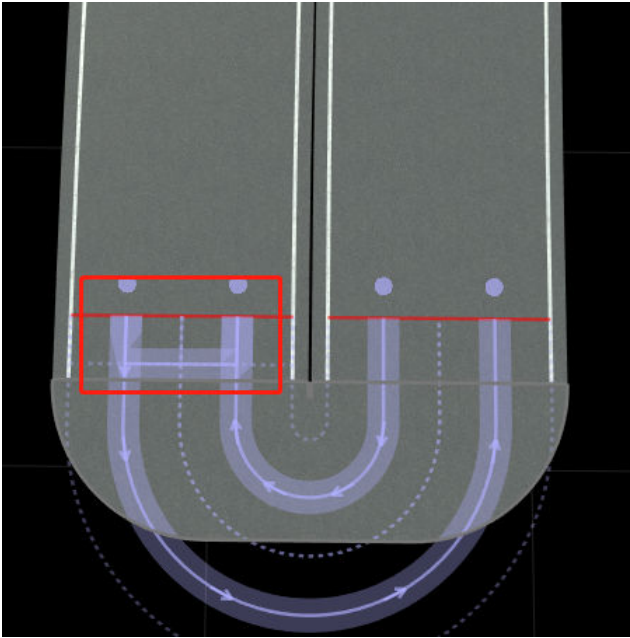


If you need to define more complex trajectories through a junction, you can insert additional control points as follows:

- 1 Click the **Maneuver Tool** button.
- 2 Click the junction you want to edit. All of the individual maneuvers within the junction are displayed.
- 3 Click the maneuver road you want to edit.
- 4 Point to the blue control line at the location you want to insert a node.
- 5 Right-click to insert a new node within the control line of the maneuver road.

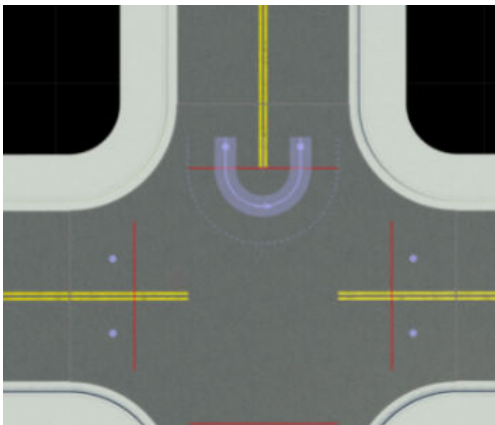
Create U-turns Within Two Lanes on the Same Road

In some cases, the visual representation of the U-turn maneuver from one lane to another on the same road may not be a visually smooth U-turn.



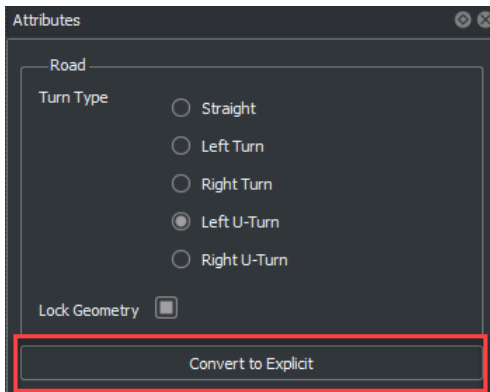
To refine the visual representation of the U-turn, follow these steps:

- 1 Click the **Lane Add Tool** to insert a new lane between the two target lanes.
- 2 Set the width of the newly added lane to 0. This results in an automatically generated maneuver.

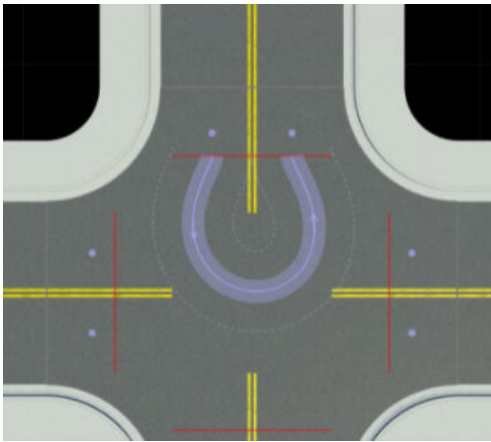


If you cannot add a new lane to your road, or doing so causes issues with the scene, use this alternative method.

- 1 Click the **Maneuver Tool** button.
- 2 Select the U-turn maneuver road you want to edit.
- 3 In the **Attributes** pane, click **Convert to Explicit**. This option enables you to edit the control points of any maneuver road that is either automatically generated or manually created.

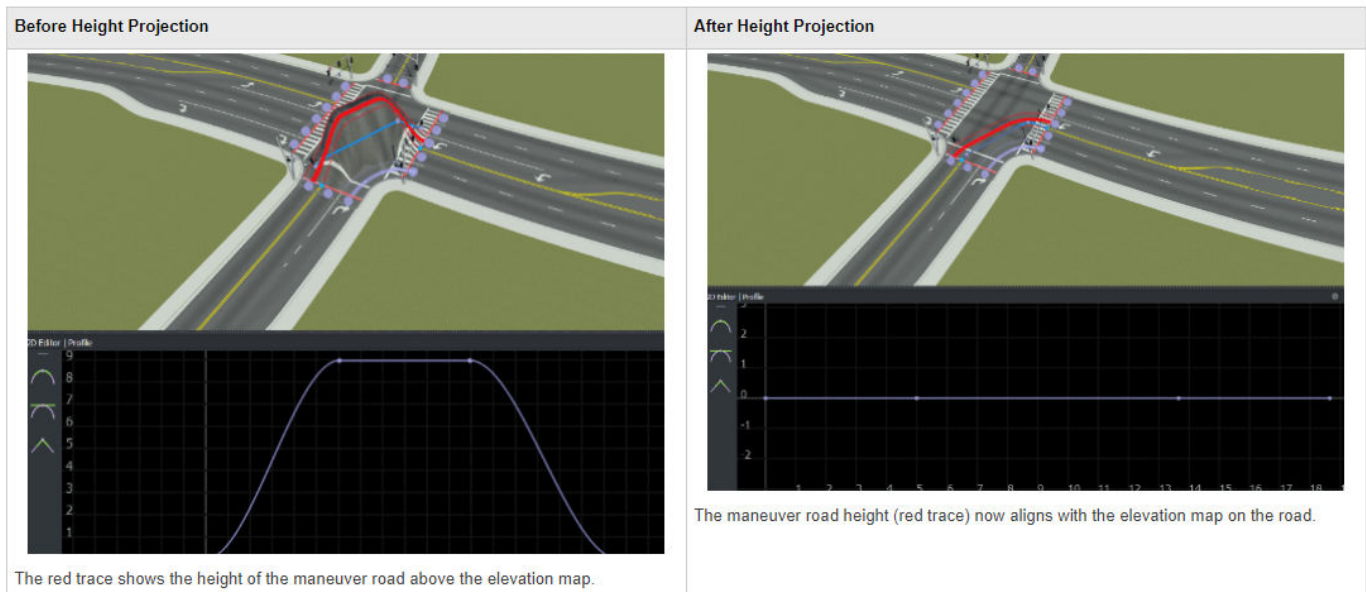


- 4 Drag the desired control points to manually modify the geometry of the U-turn. Manually editing the control points can result in a U-turn with smooth and refined edges.



Project Maneuver Roads to Elevation Maps

RoadRunner can project one or more maneuver roads down to the height of an elevation map (see **Elevation Map Assets**). This process samples the elevation data under the maneuver road and performs a constrained fit of the maneuver road elevation to the trace of the elevation data beneath the maneuver road.



- 1 Click the **Maneuver Tool** button.
- 2 Select the maneuver roads you want to project. (You can perform a **Select All** operation to select all the maneuver roads in the scene).
- 3



Click the **Project Roads** button on the toolbar on the left. The selected maneuver roads are elevated to match the elevation data present in the scene.

There are two types of height projection that can be performed: a relaxed fit and a tight fit. The type of fit used depends on whether the road uses quadratic or cubic interpolations.

Relaxed Fit

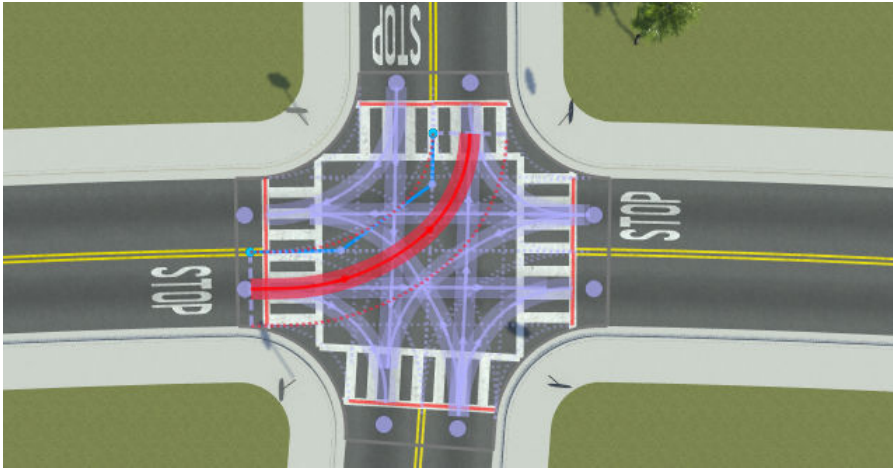
If a road is using quadratic interpolation on page 1-130, a relaxed, approximate fit is used. This interpolation is best when the terrain data is either noisy or low resolution.

Tight Fit

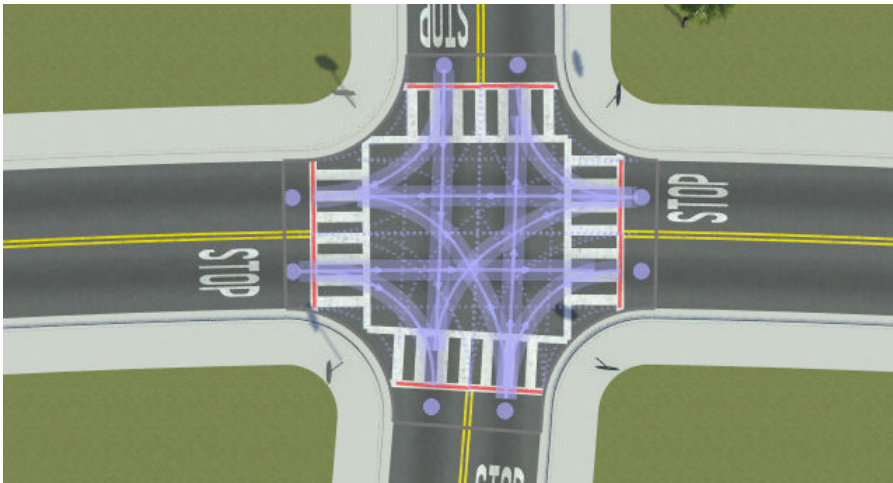
If a road is using cubic interpolation on page 1-130, a much tighter fitting method is used. This interpolation is best when you want the road to closely match the heights of the elevation map.

Delete Maneuver Road

- 1 Click the **Maneuver Tool** button.
- 2 Select the junction you want to edit. The canvas displays all maneuvers within the junction.
- 3 Click the maneuver road you want to delete. The selected maneuver road is highlighted in red.



- 4 Press **Delete**. Alternatively, select **Edit** from the menu bar, and then click **Delete**. This deletes the selected maneuver road.



Parameters

Attribute	Description
Turn Type	<p>Identifies the semantic turn type for the maneuver road (left turn, U-turn, and so on).</p> <p>This type is computed automatically based on the geometry of the junction, but there might be some complex junctions where the type is computed incorrectly (for example, a sharp left turn is perceived as a U-turn).</p> <p>This turn type affects the role of the maneuver in junction signalization. See the Signal Tool.</p>
Lock Geometry	See Enable or Disable Automatic Maneuver Geometry on page 1-61.

Version History

Introduced in R2020a

R2022b: Project Roads: Project the height of roads using Maneuver Tool

You can use the **Maneuver Tool** in RoadRunner to project one or more maneuver roads down to the height of an elevation map. This process samples the elevation data under the maneuver road and performs a constrained fit of the maneuver road elevation to the trace of the elevation data beneath the maneuver road.

See Also

Custom Junction Tool

Marking Curve Tool

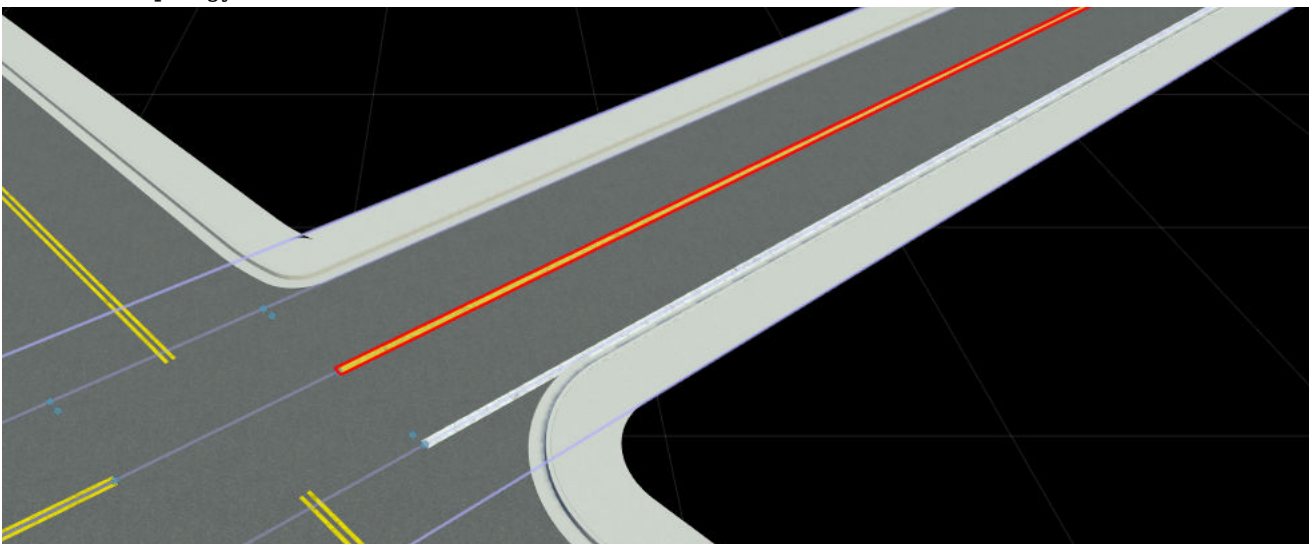
Place straight or curved markings at arbitrary locations

Description

The **Marking Curve Tool** can be used to place straight or curved markings at arbitrary locations.

Marking curves show up on both road surfaces and terrain surfaces. They use the same linear marking style assets as the **Lane Marking Tool**.

Although you can create free-form crosswalks with this tool, use the **Crosswalk And Stop Line Tool** where possible, because crosswalks created in that tool have more semantic linkage to the road topology.



Open the Marking Curve Tool

On the RoadRunner toolbar, click the **Marking Curve Tool** button:



Examples

Edit Marking Curves

See “Curve Editing”.

Note When creating a new marking curve, you must have **Lane Marking Assets** or **Crosswalk Marking Assets** selected in the **Library Browser**.

Change the Marking on a Curve

- 1 Click the **Marking Curve Tool** button.
- 2 Select the marking curve you want to change.
- 3 Click and drag **Lane Marking Assets** or **Crosswalk Marking Assets** from the **Library Browser** onto the **Marking** widget in the **Attributes** pane.

Alternatively, click and drag **Lane Marking Assets** or a **Crosswalk Marking Assets** from the **Library Browser** directly onto the desired marking curve. You do not need to click the **Marking Curve Tool** button first to perform this operation, because it works from any tool. Once done, RoadRunner automatically enters the **Marking Curve Tool** mode and selects the changed marking curve for further editing.

Parameters

Attribute	Description
Marking Style	The Lane Marking Assets or Crosswalk Marking Assets assigned.
Flip Side	If true, the order of the marking stripes is reversed (for marking types with more than one stripe). Note that this value is initially computed based on the neighboring lane travel directions when Lane Marking Assets are assigned with the Lane Marking Tool .
Start Blend Distance	If nonzero, the start of the marking is gradually faded to transparent over this distance. Note that the start is dependent on the digitization direction of the marking curve or road. This option has an effect only if the material's diffuse texture contains transparent (nonopaque) content.
End Blend Distance	If nonzero, the end of the marking will be gradually faded to transparent over this distance. See the additional comments in Start Blend Distance .
Phase Shift	This is used to offset dashed marking types along the curve. This is useful for synchronizing the spacing of dashes where two roads meet.
Color	Color multiplied by the material color.

Attribute	Description
Material	Material Assets to use for this marking instance. If set, this will override the material in the Lane Marking Assets . If not set, the marking asset's material is used instead.
Texture Scale	Scale to apply to each dimension of the marking's texture coordinates. Note Note that the texture coordinate space depends on the value of the Curve Space Texture option in the Lane Marking Assets .
Texture Rotation	If nonzero, specifies an additional rotation to apply to the marking's texture coordinates.
Texture Offset	If nonzero, specifies an additional offset to apply to each dimension of the marking's texture coordinates.
Sort Index	When markings of any type overlap, this value is used to determine which markings appear on top. Given two overlapping markings, the marking with the higher sort index is drawn on top.

Version History

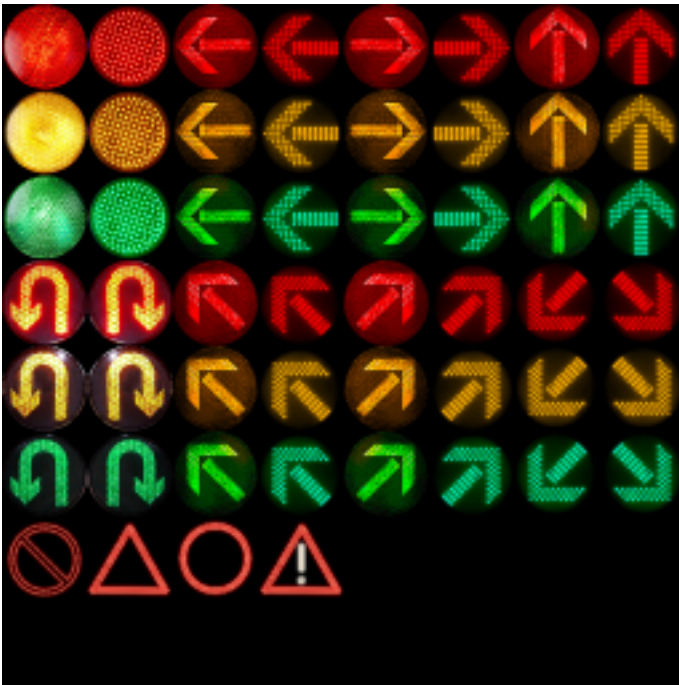
Introduced in R2020a

Marking Point Tool

Place point markings (stencils), such as arrows and words, on road surfaces

Description

The **Marking Point Tool** enables you to place point markings (or stencils), such as arrows and words, on road surfaces. Point markings can be added as either free-form markings, or anchored to the center of a lane. Both **Stencil Marking Assets** and **Texture Assets** can be used as point markings.



Open the Marking Point Tool

On the RoadRunner toolbar, click the **Marking Point Tool** button:



Examples

Create a Road Point Marking Anchored to a Lane

- 1 Click the **Marking Point Tool** button.
- 2 Select one of the **Stencil Marking Assets** or **Texture Assets** in the **Library Browser**.
- 3 Move the pointer over the center of a lane. A lane center curve is displayed.
- 4 Right-click to add a new road point marking and anchor it to the lane.

Alternatively, click and drag one of the **Stencil Marking Assets** from the **Library Browser** onto the center curve of the desired lane.

Note This operation works from any tool. Once done, it automatically selects the **Marking Point Tool** and selects the new point marking for further editing.

Create a Free-Form Road Point Marking

- 1 Click the **Marking Point Tool** button.
- 2 Select one of the **Stencil Marking Assets** or **Texture Assets** in the **Library Browser**.
- 3 Check that the point is not over the center of a lane. The lane center curve must not display.
- 4 Right-click to add a new free-form road point marking.

Move a Road Point Marking Anchored to a Lane

- 1 Click the **Marking Point Tool** button.
- 2 Click and drag the marking you want to move to the desired location. The marking slides along the center of the lane.

Move a Road Point Marking

- 1 Click the **Marking Point Tool** button.
- 2 Click and drag the marking you want to move to the desired location.

Note The marking point applies only to a road surface if it has a similar height as the road surface.

You can use the **Project Control Point** button in the toolbar on the left to automatically set the height of a point marking. You can also adjust the height manually by using the **Z** value in the **Attributes** pane.

More About

Stencil Markings and Texture Assets

Both **Stencil Marking Assets** and **Texture Assets** can be used as point markings. The distinctions are as follows.

Stencil Marking Assets

The marking outline is a group of polygons defined by the polygons in the SVG file. This results in more geometry, but less overdraw when rendering. No alpha channel is required. **Stencil Marking Assets** also support optional materials to fill the interior of the polygonal region.

Texture Assets

The marking outline is a rectangle. This allows any image file for **Texture Assets** to be used as a point marking, but requires an alpha channel and more rendering overdraw to control the transparent portions of the marking.

Version History

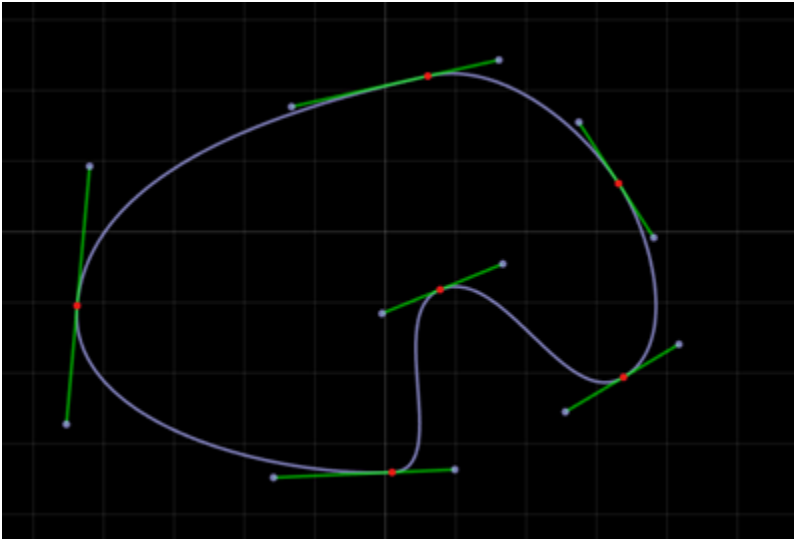
Introduced in R2020a

Marking Polygon Tool

Define areas of asphalt patches or repeated marking stripes on roads and terrain surfaces

Description

The **Marking Polygon Tool** can be used to define areas of asphalt patches or repeated marking stripes on roads and terrain surfaces. Marking polygons support the assignment of polygon marking styles, which define the marking appearance.



Open the Marking Polygon Tool

On the RoadRunner toolbar, click the **Marking Polygon Tool** button:



Examples

Edit Marking Polygons

See “Polygon Editing”.

Note To create a new marking polygon, you must have one or more **Polygon Marking Assets** selected in the **Library Browser**.

Change the Marking on a Polygon

- 1** Click the **Marking Polygon Tool** button.
- 2** Click the marking polygon you want to change.
- 3** Click and drag a polygon marking asset from the **Library Browser** onto the **Marking** widget in the **Attributes** pane.

Alternatively, click and drag a polygon marking asset from the **Library Browser** directly onto the desired marking polygon.

Note This operation works from any tool. Once done, RoadRunner automatically enters the appropriate tool and selects the object for further editing.

Version History

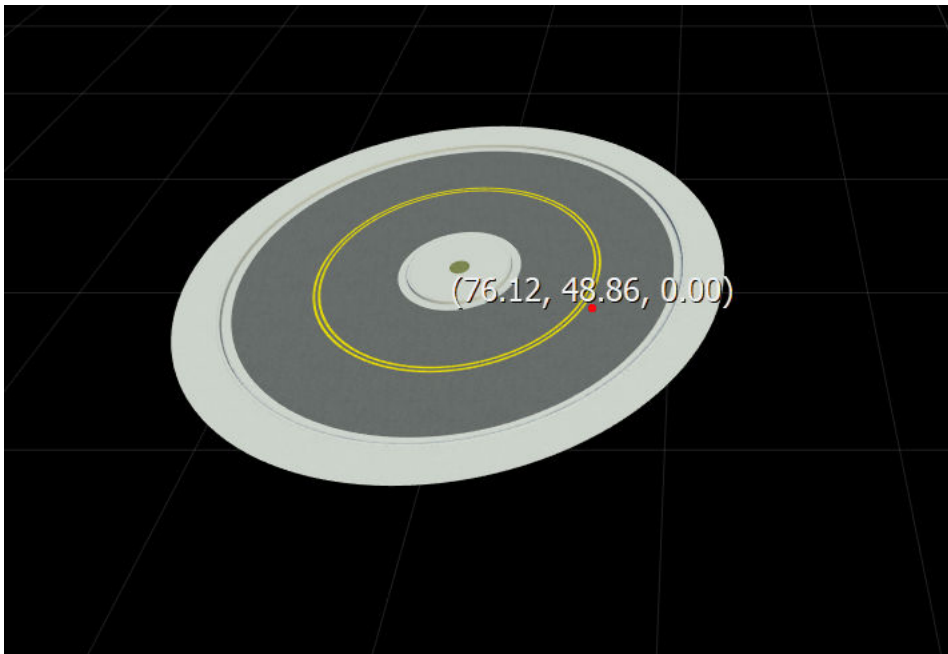
Introduced in R2020a

Measurement Tool

Measure positions, distances, and angles in scene

Description

The **Measurement Tool** enables you to measure positions, distances, and angles. You can make multiple individual measurements and place them in the scene. Measurements remain visible in other tools, although they can be selected and modified only within the **Measurement Tool**. Measurements are not permanent and are not exported with the scene.



Open the Measurement Tool

On the RoadRunner toolbar, click the **Measurement Tool** button:



Examples

Measure the XYZ Position at a Single Point

- 1 Click the **Measurement Tool** button.

- 2 If another measurement is already picked, click away from any measurements to unpick it.
- 3 Right-click the point you want to query. The XYZ location of the point is displayed next to it. The location is relative to the scene origin (0,0,0). The X value is positive in the north direction. Y is positive in the east direction. Z is positive in the up direction.

Measure the Distance Between Two Points

- 1 Click the **Measurement Tool** button.
- 2 If another measurement is already picked, click away from any measurements to unpick it.
- 3 Right-click the first point. The XYZ location of the point is displayed next to it.
- 4 Right-click the second point. The distance between the two points is displayed. If the two points are at roughly the same altitude, then only the 3D distance is displayed. If the two points are at different altitudes, then the horizontal and vertical distances are also displayed.

Measure an Angle Between Three Points

- 1 Click the **Measurement Tool** button.
- 2 If another measurement is already picked, click away from any measurements to unpick it.
- 3 Right-click the first point. The XYZ location of the point is displayed next to it.
- 4 Right-click the second point. The distance between the two points is displayed.
- 5 Right-click the third point. The angle between the three points is displayed.

Measure the Distance Along Multiple Points

- 1 Click the **Measurement Tool** button.
- 2 If another measurement is already picked, click away from any measurements to unpick it.
- 3 Right-click to create points along the distance you want to measure.
- 4 Look at the number floating above the middle point of the measurement to see the distance the set of points covers.

Delete a Measurement

- 1 Click the **Measurement Tool** button.
- 2 Click the measurement you want to delete.
- 3 Press the **Delete** key or select **Edit > Delete** from the menu bar.

Delete All Measurements

- 1 Click the **Measurement Tool** button.
- 2 If any measurements are already selected, press **Ctrl+D** or select **Edit > Deselect All** to clear the selection set.
- 3 Press **Ctrl+A** or choose **Edit > Select All** to select all measurements.
- 4 Press the **Delete** key or select **Edit > Delete** from the menu bar.

Version History

Introduced in R2020a

OpenDRIVE Export Preview Tool

Visualize and validate ASAM OpenDRIVE export of scene and load external ASAM OpenDRIVE files



Description

The **OpenDRIVE Export Preview Tool** is used to visualize and validate an ASAM OpenDRIVE export of the current scene and to load external ASAM OpenDRIVE files. Upon entering the tool, the ASAM OpenDRIVE export dialog box is displayed. After clicking **Export**, the current scene is exported to a temporary ASAM OpenDRIVE file, which is then loaded, validated, and displayed.

Validations are run both before export and after loading the exported data. This validation output is printed to the **Output** pane. Most validation errors or warnings include hyperlinks in the message. Click a link to focus on the object or issue.

To avoid visual clutter, click the **Show Background Scene** button to toggle display of the scene meshes. The **Show OpenDRIVE® Lane Markings** button toggles display of the ASAM OpenDRIVE lane marking attributes, which are displayed only when a road curve is selected.

You can also use the **OpenDRIVE Export Preview Tool** to validate and display existing ASAM OpenDRIVE files. This can be useful for validating files from other sources or previewing an ASAM OpenDRIVE file prior to import.

OpenDRIVE Export Preview Tool with Visual Scene On	OpenDRIVE Export Preview Tool with Visual Scene Off
	



Open the OpenDRIVE Export Preview Tool

On the RoadRunner toolbar, click the **OpenDRIVE Export Preview Tool** button:



Examples

Preview ASAM OpenDRIVE Data for the Current Scene

- 1 Click the **OpenDRIVE Export Preview Tool** button to open a dialog box with some ASAM OpenDRIVE options.
- 2 Choose the options you want and click **Export**. RoadRunner exports the current scene to ASAM OpenDRIVE and displays the resulting ASAM OpenDRIVE roads and lanes.

Load an External ASAM OpenDRIVE File for Display

See **OpenDRIVE Viewer Tool**.

Note This feature does not interpret any georeferencing data in the loaded file.

This feature is purely for viewing and validating an ASAM OpenDRIVE file. To import an ASAM OpenDRIVE file into RoadRunner for editing, see “Importing ASAM OpenDRIVE Files”.

View the Attributes of ASAM OpenDRIVE Features

From within the **OpenDRIVE Export Preview Tool**, select lanes and other objects in the 3D edit window to view their attributes in the **Attribute** pane.

Toggle the Display of 3D Scene Geometry

From within the **OpenDRIVE Export Preview Tool** (or any tool), select the **View > Scene** option in the menu bar or press the **F8** key.

Selectively Display ASAM OpenDRIVE Lanes

By default, the **OpenDRIVE Export Preview Tool** displays all ASAM OpenDRIVE lanes and objects in the scene. The tool optionally provides a mode to display lanes and objects for only the selected ASAM OpenDRIVE roads.

From within the **OpenDRIVE Export Preview Tool**:

- 1 Click the **Show All Lanes** button on the left toolbar.
- 2 The tool now displays on the ASAM OpenDRIVE road plan curves by default. When you select a road curve, the lanes and objects for that road are displayed.

View Validation Results

From within the **OpenDRIVE Export Preview Tool**, examine the validation report output in the **Output** pane. Most errors or warnings include a hyperlink. Click a link to focus on the object or issue.

Search for ASAM OpenDRIVE Features

From within the **OpenDRIVE Export Preview Tool**:

- 1 Click the **Search** button in the sub-tool bar on the left.
- 2 Enter the desired search properties into the search dialog box.
- 3 Press the **Find Next** button on the search dialog box.

This feature is useful for debugging problems if a separate application reports an issue with a specific road ID.

Version History

Introduced in R2020a

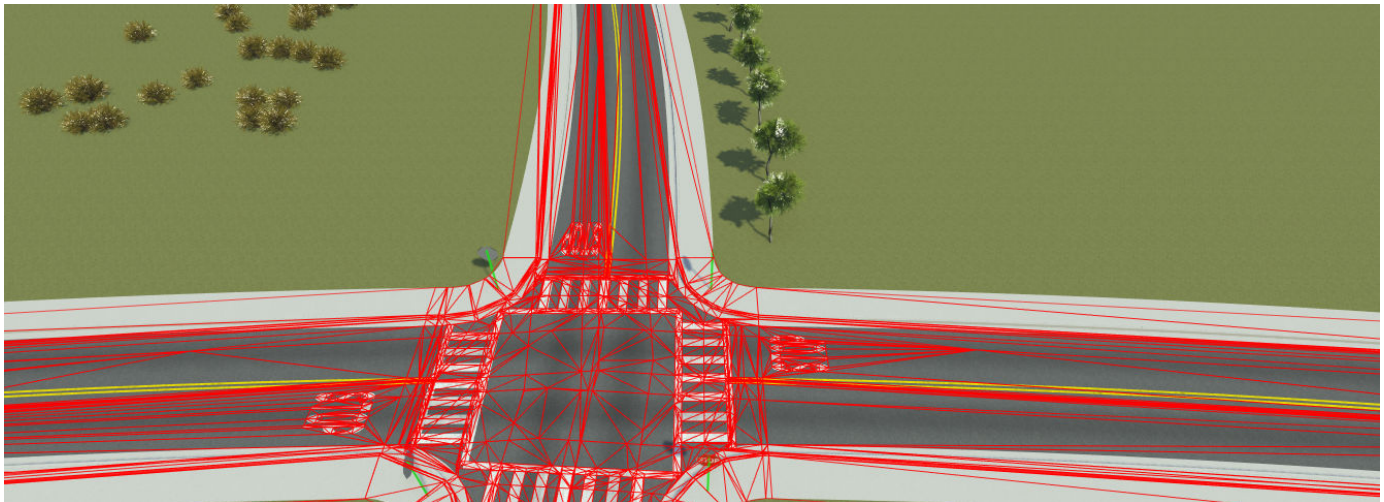
OpenDRIVE Viewer Tool

Visualize ASAM OpenDRIVE data for import

Description

The **OpenDRIVE Viewer Tool** is used to visualize ASAM OpenDRIVE data for import.

The **OpenDRIVE Viewer Tool** has a similar interface as the **OpenDRIVE Export Preview Tool**. ASAM OpenDRIVE data brought in to the tool is visible in the scene, along with any other vector data.



Open the OpenDRIVE Viewer Tool

On the RoadRunner toolbar, click the **OpenDRIVE Viewer Tool** button:



Examples

Load an External ASAM OpenDRIVE File for Display

- 1 Click the **OpenDRIVE Viewer Tool** button.
- 2 Click and drag the desired ASAM OpenDRIVE file from the **Library Browser**.

View the Attributes of ASAM OpenDRIVE Features

From within the **OpenDRIVE Viewer Tool**, select lanes and other objects in the 3D edit window to view their attributes in the **Attributes** pane.

Toggle the Display of 3D Scene Geometry

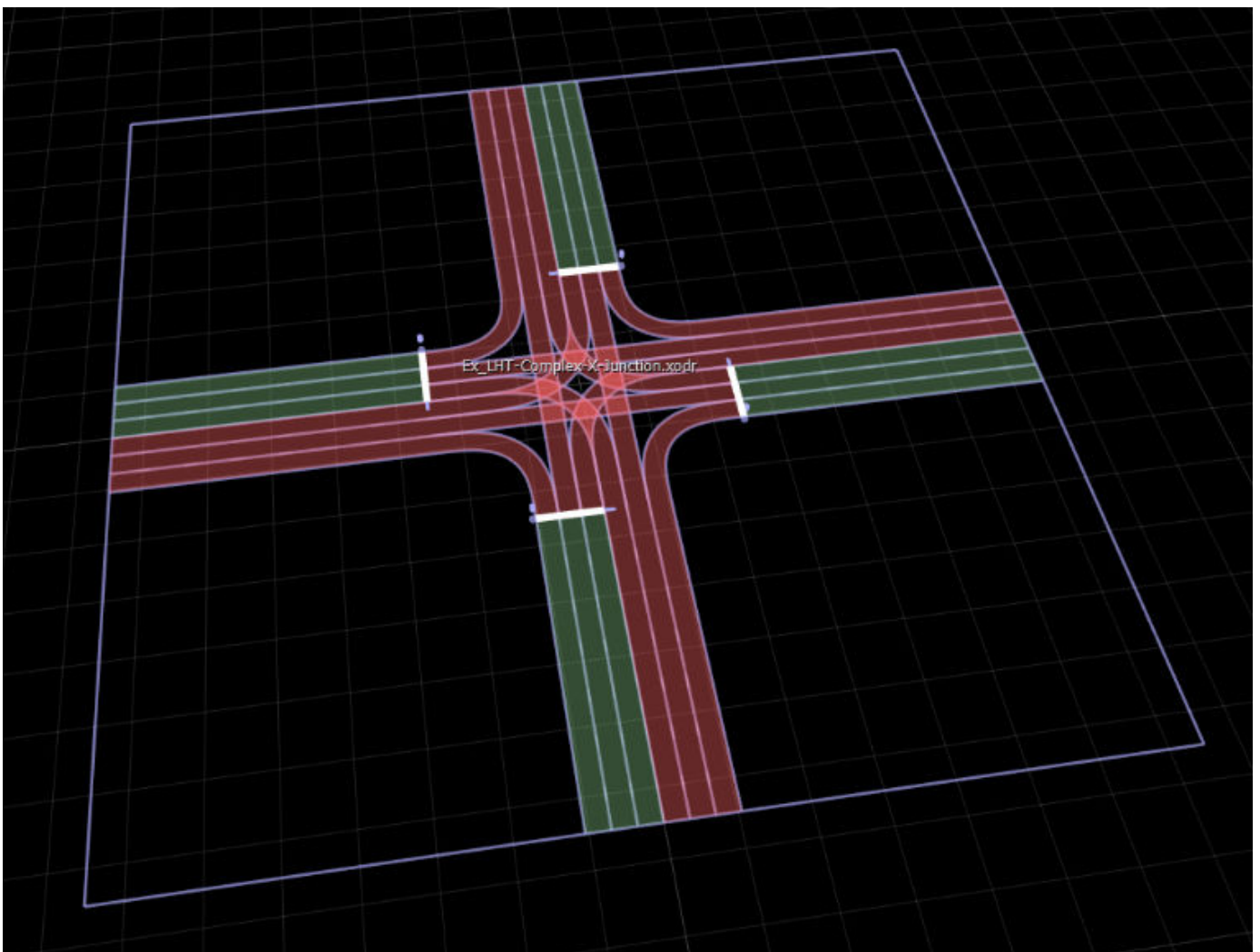
From within the **OpenDRIVE Viewer Tool** (or any tool), select the **View > Scene** option in the menu bar or press the **F8** key.

Toggle the Display of Loaded ASAM OpenDRIVE Data (Outside of OpenDRIVE Viewer Tool)

From within any tool other than the **OpenDRIVE Viewer Tool**, select the **View > Vector** option in the menu bar or press the **F7** key.

Import an ASAM OpenDRIVE File

- 1 Click the **OpenDRIVE Viewer Tool** button.
- 2 Click the visualized file's bounding rectangle. In this figure, the visualized file's bounding box is represented by the lavender colour rectangle.



Alternatively:

- Hold the **Ctrl** key and click to select multiple files.
 - Deselect all files to import all ASAM OpenDRIVE files.
- 3** Click the **Build Scene** button.



- 4** Select desired options.

View Validation Results

From within the **OpenDRIVE Viewer Tool**, examine the validation report output in the **Output** pane. Most errors or warnings include a hyperlink. Click a link to focus on the object or issue.

Search for ASAM OpenDRIVE Features

From within the **OpenDRIVE Viewer Tool**:

- 1** Click the **Search** button in the left toolbar.
- 2** Enter the desired search properties into the search dialog box.
- 3** Press the **Find Next** button on the search dialog box.

Version History

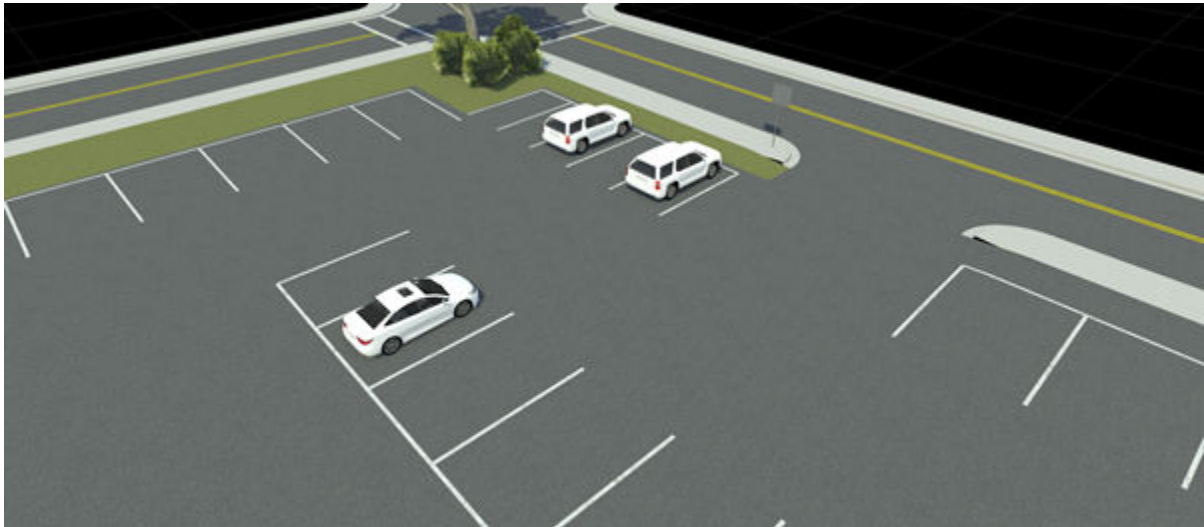
Introduced in R2020a

Parking Tool

Define parking spaces and other parking-related markings

Description

The **Parking Tool** can be used to define parking spaces or other parking-related markings. You can assign marking styles to the boundaries of spaces and parking nodes of parking spaces.

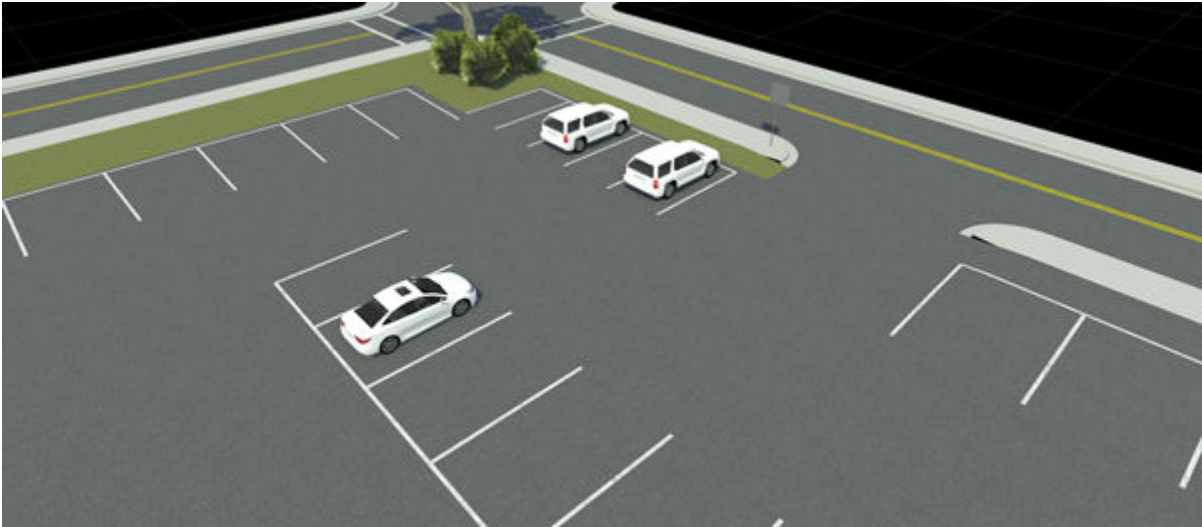


Parking spaces are created as a curve, where spaces are automatically created along the curve. Parking curves are automatically created for lanes of type parking.

Parking spaces are modeled as a region graph, where each closed region represents a parking space.

Markings can be placed on the graph edge curves to represent painted boundaries.

A graph edge curve can be optionally marked as an **Entry Edge**. This indicates that a vehicle can enter the parking space from that edge.



Open the Parking Tool

On the RoadRunner toolbar, click the **Parking Tool** button:



Examples

Create a New Parking Curve

- 1 Click the **Parking Tool** button.
- 2 Check that no objects are selected. For example, you can use the **Edit > Deselect All** option in the menu bar.
- 3 Right-click (and optionally drag) to create a curve with a single starting point. The new curve is automatically assigned to the selected asset.
- 4 Right-click (and optionally drag) to extend the curve by adding additional control points.

For lanes with type parking, parking curves are automatically created and updated. For more details, see **Lane Tool**.

Create a Single Parking Space

- 1 Click the **Parking Tool** button.
- 2 Create a parking curve with two points.
- 3 Check that the second point is close enough to create only one parking space.

Change the Marking on a Parking Curve

- 1 Click the **Parking Tool** button.
- 2 Select the parking curve you want to change.

- 3 Click and drag **Lane Marking Assets** from the **Library Browser** onto the **Marking** widget in the **Attributes** pane.

Alternatively, click and drag **Lane Marking Assets** from the **Library Browser** directly onto the desired parking space edge.

Assign a Stencil Marking Style to a Parking Space

- 1 Click the **Parking Tool** button.
- 2 Click a parking curve.
- 3 Click and drag a stencil marking style onto the parking curve or assign it in the **Attributes** pane.



Note The asset pickers grouped under **Node Markings** in the **Attributes** pane accept only the point-style markings present in the **Stencils** folder in the **Library Browser**. To set a line-style marking for the parking space, drag an asset from the **Markings** folder to the **Inner Edge Marking** asset picker in the **Attributes** pane under **Edge Markings**.

Create a Parking Lot

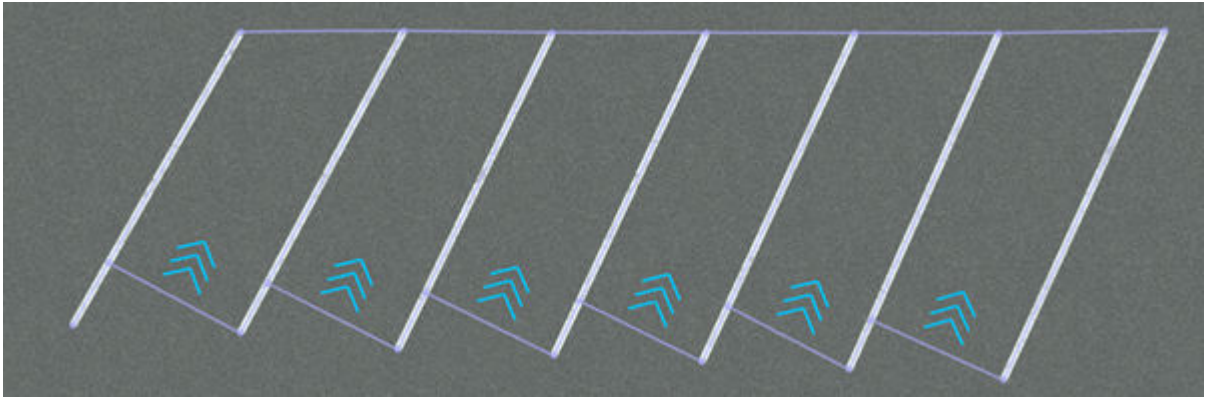
- 1 Click the **Road Plan Tool** button.
- 2 Create roads with driving lanes to define the drivable areas of the parking lot.
- 3 Click the **Parking Tool** button.
- 4 Create parking curves for each set of spaces.

Change the Width of Parking Spaces

- 1 Click the **Parking Tool** button.
- 2 Select the desired parking curve
- 3 In the **Attributes** pane, set the desired **Width**.

Create Angled Parking Spaces

- 1 Click the **Parking Tool** button.
- 2 Create a new parking curve or select an existing parking curve.
- 3 In the **Attributes** pane, set the desired **Angle**.
- 4 Assign marking styles to the desired edges.



Version History

Introduced in R2020a

Point Cloud Tool

Manage import and configuration of lidar point cloud files

Description

The **Point Cloud Tool** manages the import and configuration of lidar point cloud files. RoadRunner can import a variety of point cloud file formats, such as LAS, LAZ, and PCD. Some of these formats support georeferencing and can be automatically positioned accordingly.

Refer to the **Point Cloud Assets** page for a list of supported formats.

Point clouds often come from lidar scans, typically from an aerial flyover, a static terrestrial scan, or from a moving ground vehicle.

RoadRunner can import multiple point clouds and display them all in the same 3D space, for use as visual reference. The positioning and properties of each point cloud can be adjusted individually using this tool.

PCD Loading

- For PCD files to load points properly, the file format must have **SIZE 4** and **TYPE F** (float) for coordinate **FIELDS** of **x**, **y**, and **z**. The data can be **ASCII**, **binary**, or **compressed**.
- For **intensity**, the file format must have **FIELDS intensity**, **SIZE 4**, and **TYPE F**.
- For **color**, the file format must have **FIELDS rgba**, **SIZE 4**, and **TYPE U**.



Open the Point Cloud Tool

On the RoadRunner toolbar, click the **Point Cloud Tool** button:



Examples

Import a Georeferenced Point Cloud

- 1 Click the **Point Cloud Tool** button.

- 2 In the **Library Browser**, navigate to the directory containing the point cloud file you want to import. For more details on supported point cloud files, see **Point Cloud Assets**.
- 3 Click and drag the asset from the **Library Browser** into the 3D scene.

Note If the geographic position has not yet been set for this scene, the scene center is set to the latitudinal and longitudinal center of the image. You can change the scene center using the **World Settings Tool**.

If the geographic position has already been set, but the imported image is outside of the maximum range of the scene, an error dialog box appears and cancels the import.

Note Certain newer LAZ files are not supported in RoadRunner. As a workaround, decompress the LAZ files into the LAS format. For instructions, see “Decompress LAZ Files”.

Remove a Point Cloud from a Scene

- 1 Click the **Point Cloud Tool** button.
- 2 Click within the bounding box of the point cloud you want to delete.
- 3 Press the **Delete** key or select **Edit > Delete** from the menu bar.

Adjust the Properties of a Point Cloud

- 1 Click the **Point Cloud Tool** button.
- 2 Click the point cloud you want to edit. The attributes of the point cloud appear in the **Attributes** pane.
- 3 Adjust the point cloud attributes as desired through the **Attributes** pane.

Note The attributes of a point cloud are associated with the current scene, not to the point cloud file itself. This means that any modifications to the attributes affects only the point cloud as it appears in the current scene. These modifications do not affect if or how it appears in other scenes.

Toggle the Display of Point Clouds

Select **View > Point Clouds** on the menu bar or press the **F6** key.

Version History

Introduced in R2020a

Topics

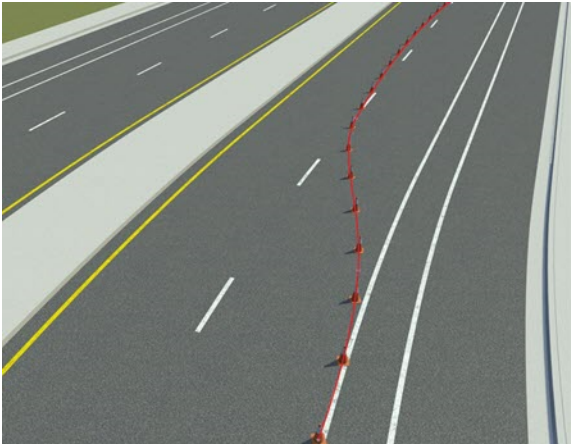
“Create Roads Around Imported GIS Assets”

Prop Curve Tool

Place props and extrusions along free-form curves.

Description

The **Prop Curve Tool** is used to place props and extrusions along free-form curves. By default, props and extrusions are aligned with the surface terrain.



Open the Prop Curve Tool

On the RoadRunner toolbar, click the **Prop Curve Tool** button:



Examples

Edit Prop Curves

See “Curve Editing”.

Note When creating a new prop curve, you must have a compatible asset (**Prop Model Assets**, **Prop Set Assets**, **Extrusion Assets**, and so on) selected in the **Library Browser**.

Change the Prop Asset on a Curve

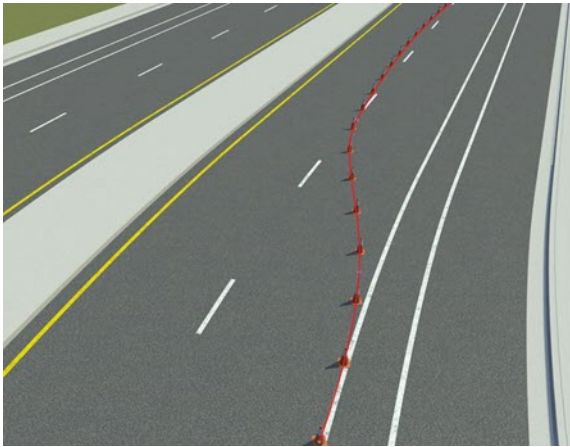
- 1 Click the **Prop Curve Tool** button.
- 2 Click the prop curve you want to change.

- 3** Click and drag a prop asset from the **Library Browser** onto the **Prop** widget in the **Attributes** pane.

Alternatively, click and drag a prop asset from the **Library Browser** directly onto the desired prop curve. (Note: You do not need to select the **Prop Curve Tool** first to perform this operation, because it works from any tool.) Once done, RoadRunner automatically enters the **Prop Curve Tool** and selects the changed prop curve for further editing.

Convert Props Along a Prop Curve Into Individual Instances

- 1** Click the **Prop Curve Tool** button.
- 2** Click the prop curve you want to change.



- 3** In the **Attributes** pane, click **Bake** to turn all instances of a prop along the prop curve into individual instances. This operation also deletes the existing prop curve and automatically enters the **Prop Point Tool**.



Version History

Introduced in R2020a

Prop Point Tool

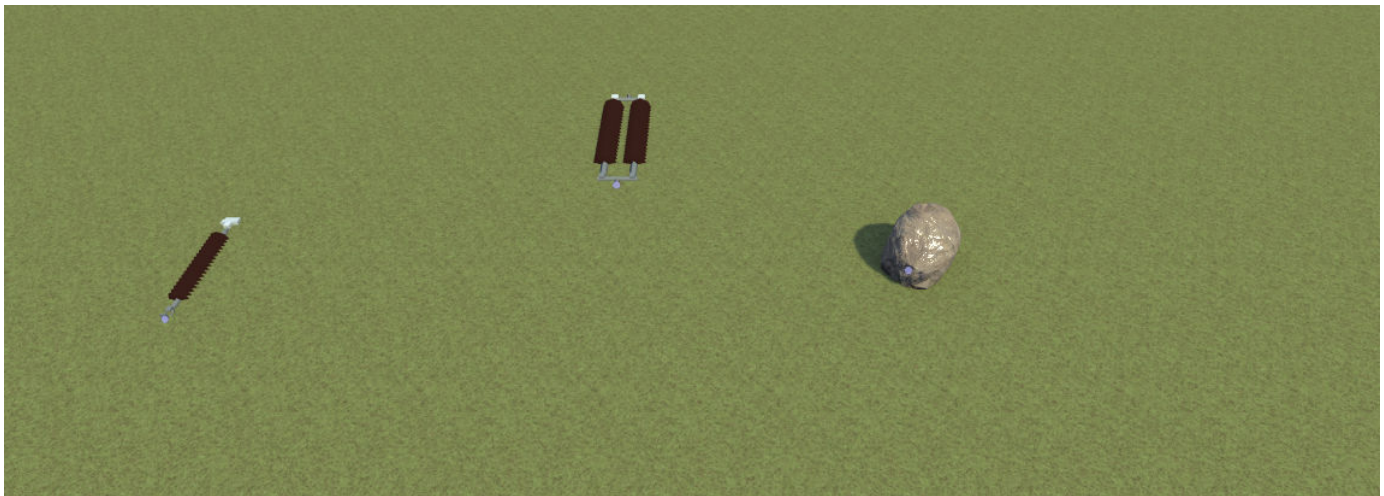
Place individual props in scene and connect them to other props

Description

The **Prop Point Tool** is used for placing individual props in the scene or connecting them to other props.

To place a prop, select the prop in the **Asset Browser** and right-click in the 3D scene. The prop should appear in the scene and be positioned on whatever surface the mouse cursor is over. In this way, props are automatically positioned on the terrain surface. If the **Align Normal** option is set for the prop, it will also rotate itself to align with the surface it is placed on.

If any of the **Rotation Variance** or **Scale Variance** attributes are set for the prop, then these properties will be randomized when the prop is placed. For example, for trees and plants, it is very useful to set the Z-value of the **Rotation Variance** to 360 degrees, causing the prop to be randomly rotated around the vertical axis and making it easier to reuse the same plant models in multiple places without the repetition being obvious. In addition, setting the **Uniform Scale Variance** to 0.1 will cause a 10% randomization in the overall size of the plant.



Open the Prop Point Tool

On the RoadRunner toolbar, click the **Prop Point Tool** button:



Examples

Edit Prop Points

See “Point Editing”.

To create a new prop point, you must have a compatible asset (**Prop Model Assets**, **Prop Set Assets**, and so on) selected in the **Library Browser**.

Add a Prop Point to the Scene

To add a prop point, follow the steps in the “Point Editing” page. Alternatively, click and drag a compatible asset directly from the **Library Browser** onto the 3D scene.

Note This operation works from any tool. Once done, RoadRunner automatically enters the appropriate tool and selects the object for further editing.

Attach a Prop to Another Prop

Simple props can be combined and attached to one another to form more complex groups of attached props. For example, a custom traffic signal can be constructed by combining a post, a mast, and multiple signal heads.

- 1 Click the **Prop Point Tool** button.
- 2 Click and drag a compatible asset directly from the **Library Browser** onto the green prop attachment curve of another prop in the scene.

For more information about prop attachment curves, see Prop Attachment Curves on page 2-20.

More About

Prop Assemblies

A group of connected props can be saved as an asset to form a prop assembly. Prop assemblies enable you to re-instantiate a complex group of attached objects in a scene. For more details, see **Prop Assembly Assets**.

Version History

Introduced in R2020a

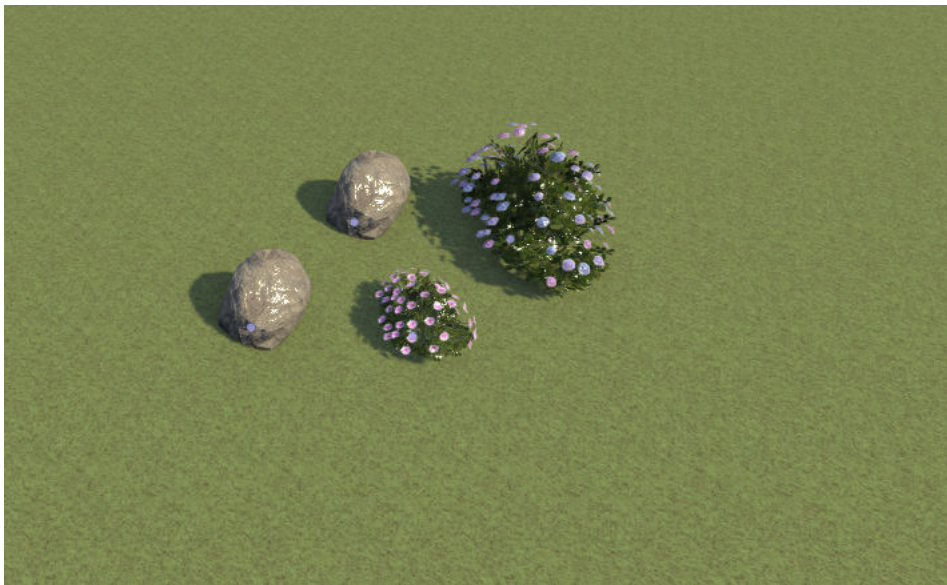
Prop Polygon Tool

Place props within arbitrarily shaped regions

Description

The **Prop Polygon Tool** allows props to be placed within arbitrarily shaped regions.

Note Prop polygons are limited to producing 10,000 individual prop instances each.



Open the Prop Polygon Tool

On the RoadRunner toolbar, click the **Prop Polygon Tool** button:



Examples

Edit Prop Polygons

See “Polygon Editing”.

To create a new prop polygon, you must have a compatible asset (**Prop Model Assets**, **Prop Set Assets**, **Extrusion Assets**, and so on) selected in the **Library Browser**.

Change the Prop Asset on a Polygon

- 1 Click the **Prop Polygon Tool** button.
- 2 Click the prop polygon you want to change.
- 3 Click and drag a prop asset from the **Library Browser** onto the prop widget in the **Attributes** pane.

Alternatively, click and drag a prop asset from the **Library Browser** directly onto the desired prop polygon. (You do not need to select the **Prop Polygon Tool** first to perform this operation, because it works from any tool.) Once done, RoadRunner automatically enters the **Prop Polygon Tool** and selects the changed prop polygon for further editing.

Convert Props Along a Prop Polygon Into Individual Instances

- 1 Click the **Prop Polygon Tool** button.
- 2 Click the prop polygon you want to change.
- 3 Click the **Bake** button in the **Attributes** pane. This operation turns all instances of a prop within the prop polygon into individual instances. This operation also deletes the existing prop polygon and automatically enters the **Prop Point Tool**.

Version History

Introduced in R2020a

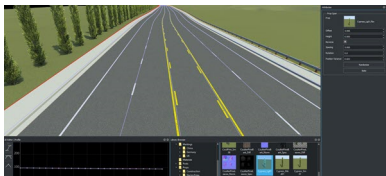
Prop Span Tool

Place props and extrusions along road

Description

The **Prop Span Tool** is used to place props and extrusions along a road.

Unlike the **Prop Curve Tool**, props placed along spans remain anchored to the road and update automatically when the road is moved.



Open the Prop Span Tool

On the RoadRunner toolbar, click the **Prop Span Tool** button:



Examples

Create and Modify Prop Spans Along a Lane

See “Span Editing”.

Prop spans store prop assets (**Prop Model Assets**, **Prop Set Assets**, **Extrusion Assets**, and so on), which can be directly dragged on to a lane span from the **Library Browser** in this tool.

Note Only a single prop asset can be assigned to a given span (although you can split a span into two spans). To assign two prop assets (for example, a guardrail on the edge of the road and a row of trees behind it), you can work around this limitation by assigning props to a span curve on a different lane, and then adjusting the **Offset** value to shift the props.

For example, you can assign a tree asset to the inner sidewalk curve, and then increase the **Offset** value to push the trees outward past the outer edge of the sidewalk.

Version History

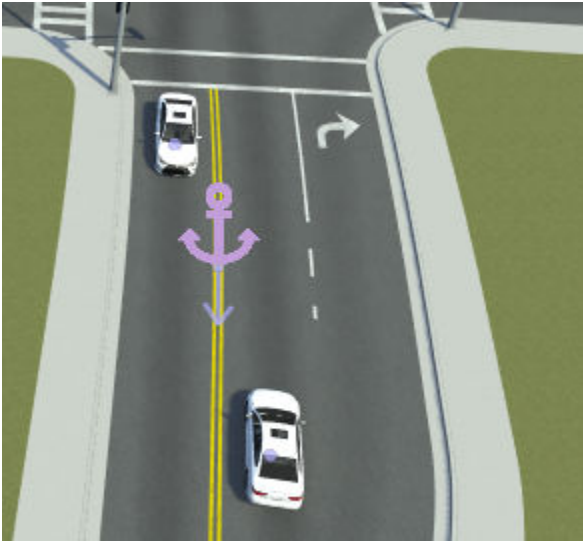
Introduced in R2020a

Road Anchor Tool

Define road anchors to place scenarios within scenes (requires RoadRunner Scenario)

Description

The **Road Anchor Tool** enables you to define road anchors within a scene. You can use these anchors as relative points by which to define the positions of actors within a scenario. For more details, see “Scenario Anchoring System” (RoadRunner Scenario).



Open the Road Anchor Tool

On the RoadRunner toolbar, click the **Road Anchor Tool** button:

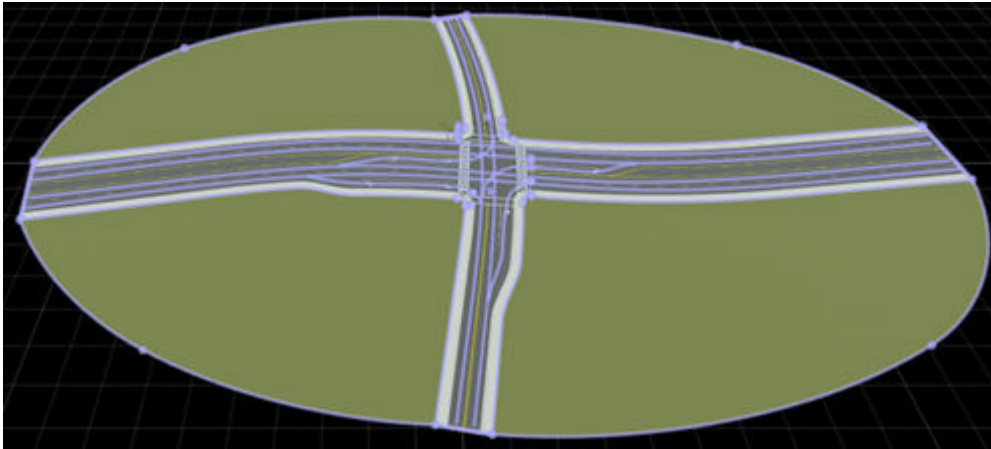



Examples

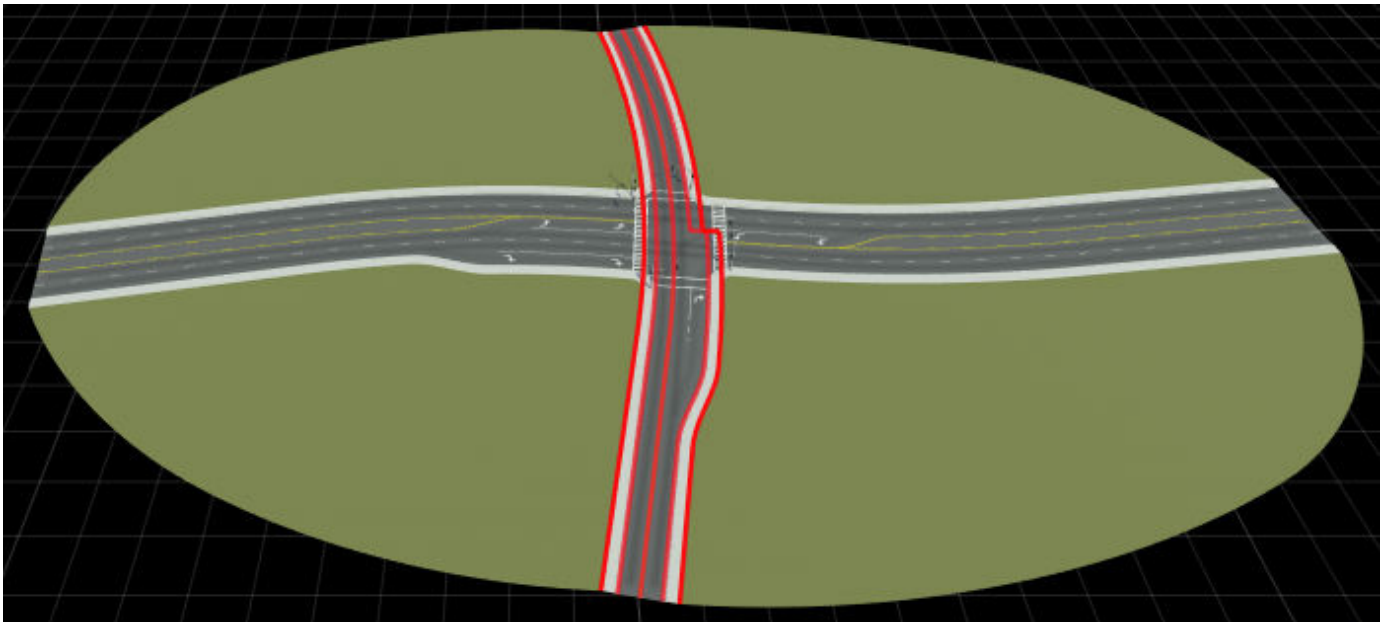
Add Road Anchor to Scene

Add a road anchor to a scene and define a simple scenario in which scenario actors attach to this road anchor. This example requires a RoadRunner Scenario license.

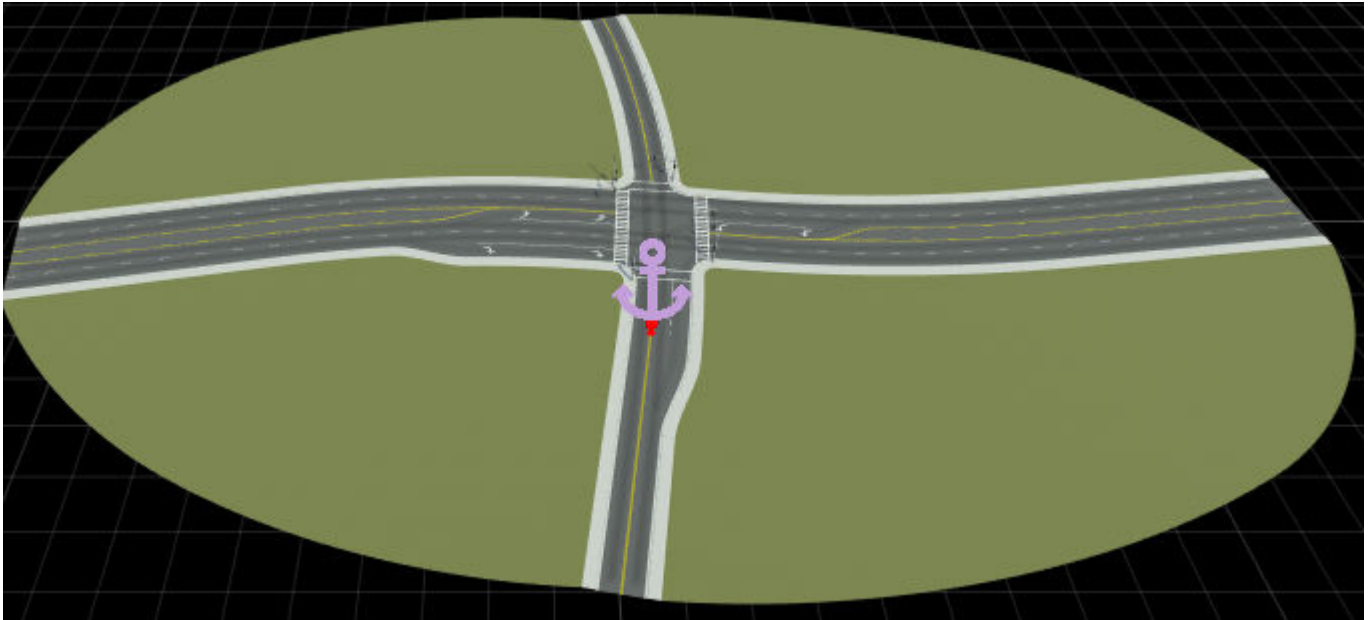
- 1 Open a sample scene. From the **File** menu, select **Open Scene**, and then select the `FourWaySignal` scene. This scene is included by default in the `Scenes` folder of the current project.



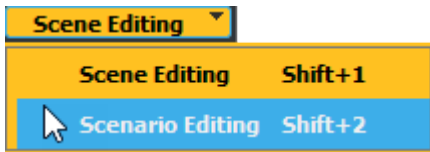
- 2 From the RoadRunner toolbar, click the **Road Anchor Tool**  button.
- 3 Click to select one of the roads in the scene.



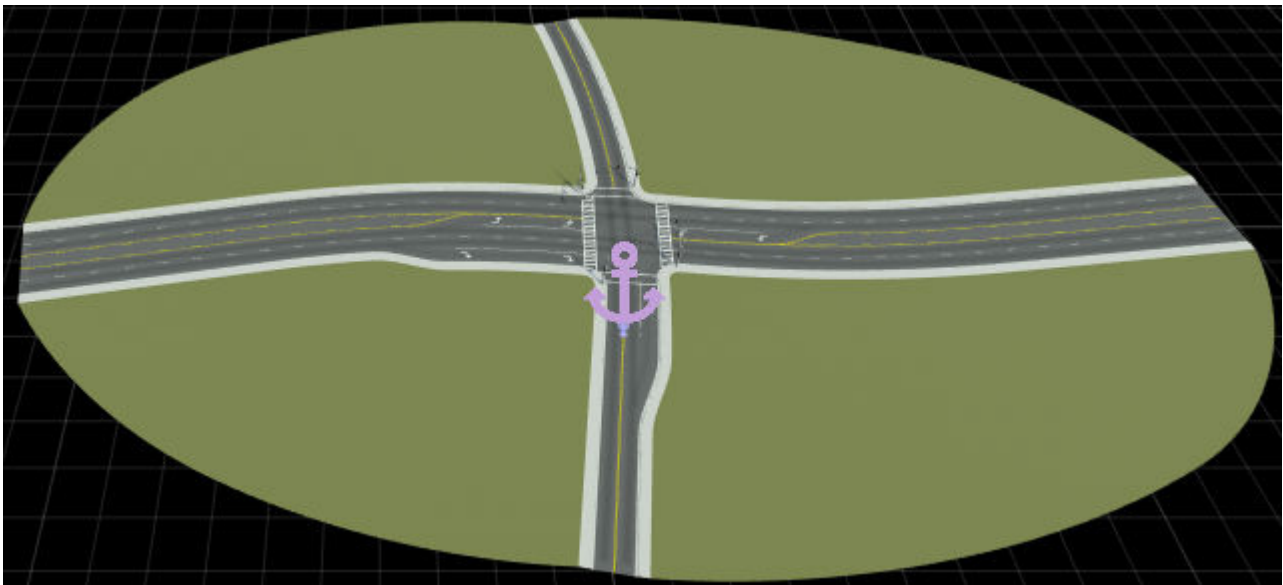
- 4 Right-click the road to add a road anchor at any point along the road. The road anchor attaches to the center of the road.



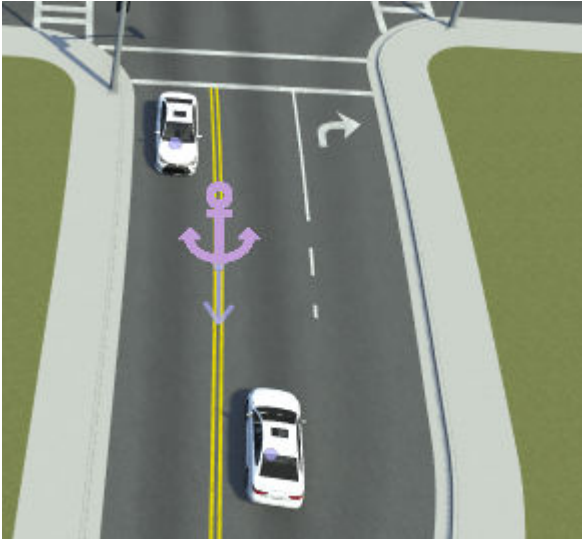
- 5 By default, the road anchor is given a node ID as its name. Give the anchor a recognizable name. In the **Attributes** pane, set **Name** to `ScenarioStart`, to indicate that this anchor is the starting point for the scenario.
- 6 Switch to scenario editing mode. From the top-right corner of the RoadRunner application, select **Scene Editing**, then **Scenario Editing**.



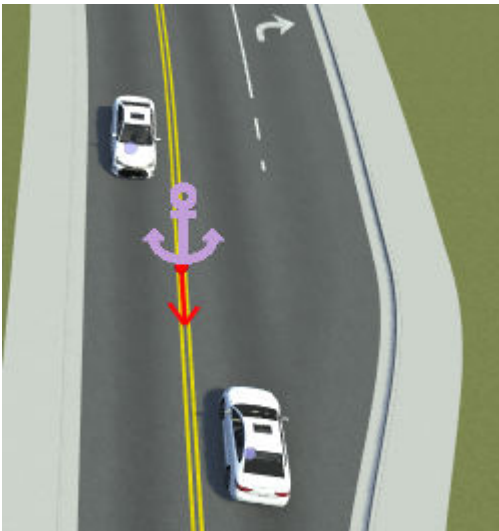
All scene objects are locked for editing except for the road anchor.



- 7 Add vehicles near the road anchor. For example, from the **Library Browser**, drag Sedan assets to the lanes on either side of the road anchor.



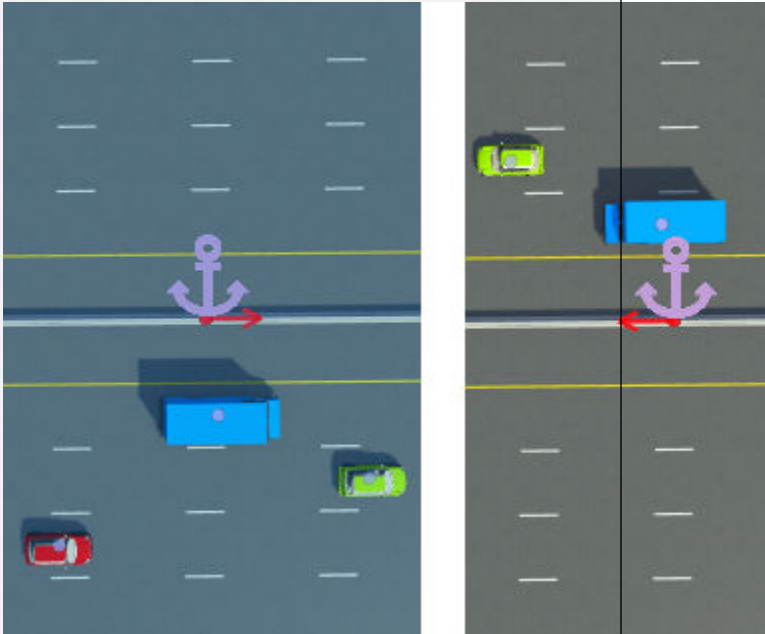
- 8 Drag the road anchor along the road. The vehicles maintain their positions relative to the road anchor.



Parameters

Road anchors added to the scene have these attributes.

Attribute	Description
Name	<p>Name of the road anchor. By default, road anchors are given a node ID name, but you can rename anchors to give them more recognizable names.</p> <hr/> <p>Note If you change the name of a road anchor, any actors previously attached to the anchor become unanchored and attach to the scenario center. To re-attach actors to the renamed anchor, follow these steps:</p> <ol style="list-style-type: none">1 Select the anchor from the scenario.2 In the Attributes pane, click the old Anchor name.3 From the scenario editing canvas, select the renamed road anchor.

Attribute	Description
<p>Travel Direction</p>	<p>Travel direction of road anchor, specified as either Forward or Backward. Travel directions are with respect to the direction in which the road is created. For example, if you create a road from left to right, then Forward points right and Backward points left.</p> <p>Changing the travel directions of an anchor changes the travel direction of all actors attached to that anchor. For example, this figure shows how the positions of vehicles change when you change the travel direction from Forward to Backward.</p>  <p>For more details on working with travel directions, see “Change Travel Direction of Actors” (RoadRunner Scenario).</p>
<p>Distance</p>	<p>Longitudinal distance, in meters, of the road anchor from the start of the road. The start of the road is based on the direction in which the road is created. For example, if you create a road from left to right, then the start of the road is on the left end.</p>

Version History

Introduced in R2022a

See Also

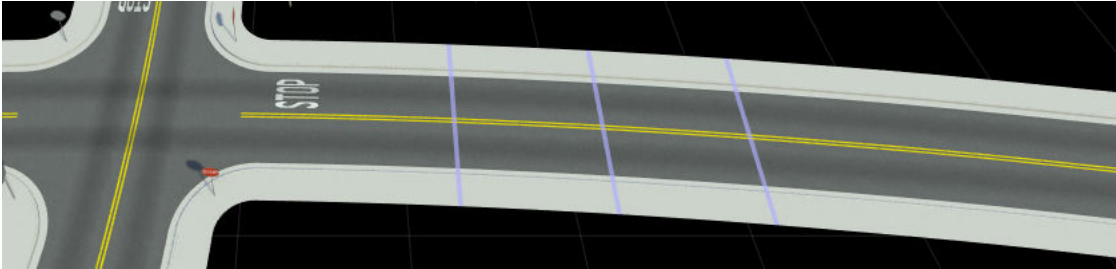
“Explore and Simulate a Simple Scenario” (RoadRunner Scenario) | “Scenario Anchoring System” (RoadRunner Scenario) | “Relocate Scenarios” (RoadRunner Scenario)

Road Chop Tool

Chop single road into two connected roads

Description

The **Road Chop Tool** chops a single road into two connected roads



Open the Road Chop Tool

On the RoadRunner toolbar, click the **Road Chop Tool** button:



Examples

Chop a Road

- 1 Click the **Road Chop Tool** button.
- 2 Click the road you want to chop. Once selected, a light blue line following the pointer appears, indicating where the chop will occur.
- 3 Move the pointer so that the blue line is at the location you want to chop. Then, right-click to chop the road.

Parameters

Use the **Automatic Road Arc/Spiral Chop Method** parameters of the **Road Chop Tool** for road chop for automatic sections:

Attribute	Description
Chop Around Linear Segments (Road May Be Chopped >1 Time)	If chopping on an arc or spiral segment of a road, the road will first be chopped at a prior and subsequent linear segment (if present). The chopped arc or spiral segment will be converted to an Explicit road. Use this method to maintain geometric accuracy while keeping the rest of the road in Automatic mode.
Convert to Explicit	If chopping an arc or spiral segment of a road, prior to performing the chop, the entire road will first be converted to Explicit. Use this method to maintain geometric accuracy and to ensure only one chop is inserted. For more details on explicit curves, see “Explicit Road Curves” on page 1-150.
Insert Control Point(s)(Road Shape May Be Affected)	Does not convert any portion of the road to Explicit. If chopping on an arc or spiral segment of a road, multiple control points may be inserted into the resulting roads near the chop to approximate the original curve of the road. Note that due to the insertion of control points, the resulting shape of the chopped roads may not exactly match the original road. Use this method to ensure only one chop is inserted with the resulting roads in Automatic mode.

Version History

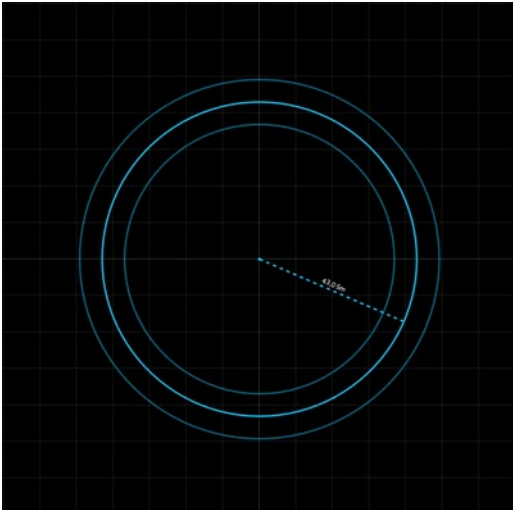
Introduced in R2020a

Road Circle Tool

Build closed circular loop road, such as for creating roundabouts

Description

The **Road Circle Tool** builds a closed, circular loop based on the currently selected road style. It can be used to build roundabouts and other circular road features. The tool builds the circle from four connected roads, each making a single 90-degree turn.



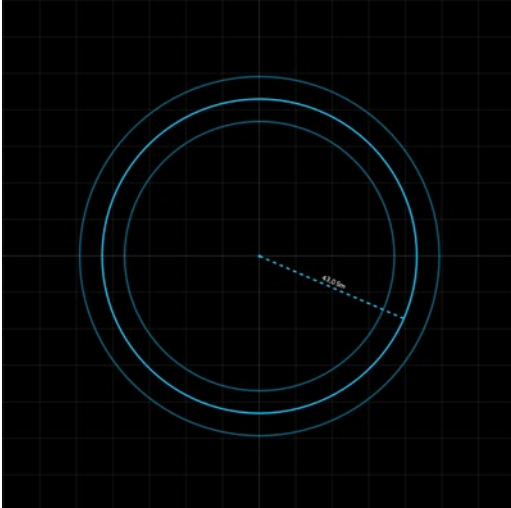
Open the Road Circle Tool

On the RoadRunner toolbar, click the **Road Circle Tool** button:



Examples

Build a Circular Road



- 1 Click the **Road Circle Tool** button.
- 2 Optionally, click the desired road style in the **Library Browser** if you want to build a road of a particular style. If no road style is picked, a basic default style is used.
- 3 Right-click and drag from the location of the center of the circle to the desired radius.

Version History

Introduced in R2020a

Road Construction Tool

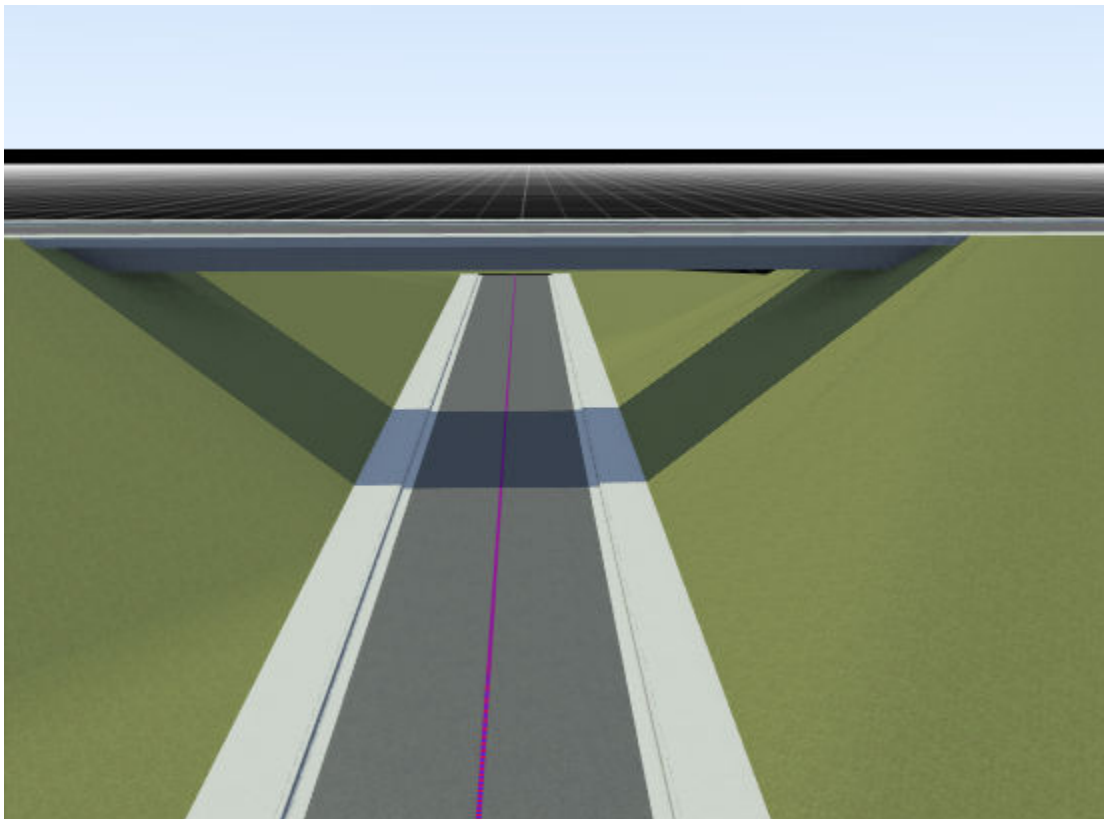
Specify physical construction of road sections

Description

The **Road Construction Tool** enables you to specify how a section of road is physically constructed. Currently, only standard (on terrain) and bridge types are supported. Tunnels, channels, and abutments are not supported.

You can set the construction type for individual spans along the road. The spans are bounded by construction nodes, which you can insert and move to arbitrary locations along the road reference curve.

Note You cannot create intersections for bridges. Raised intersections, merges, and splits are not supported.



Open the Road Construction Tool

On the RoadRunner toolbar, click the **Road Construction Tool** button:



Examples

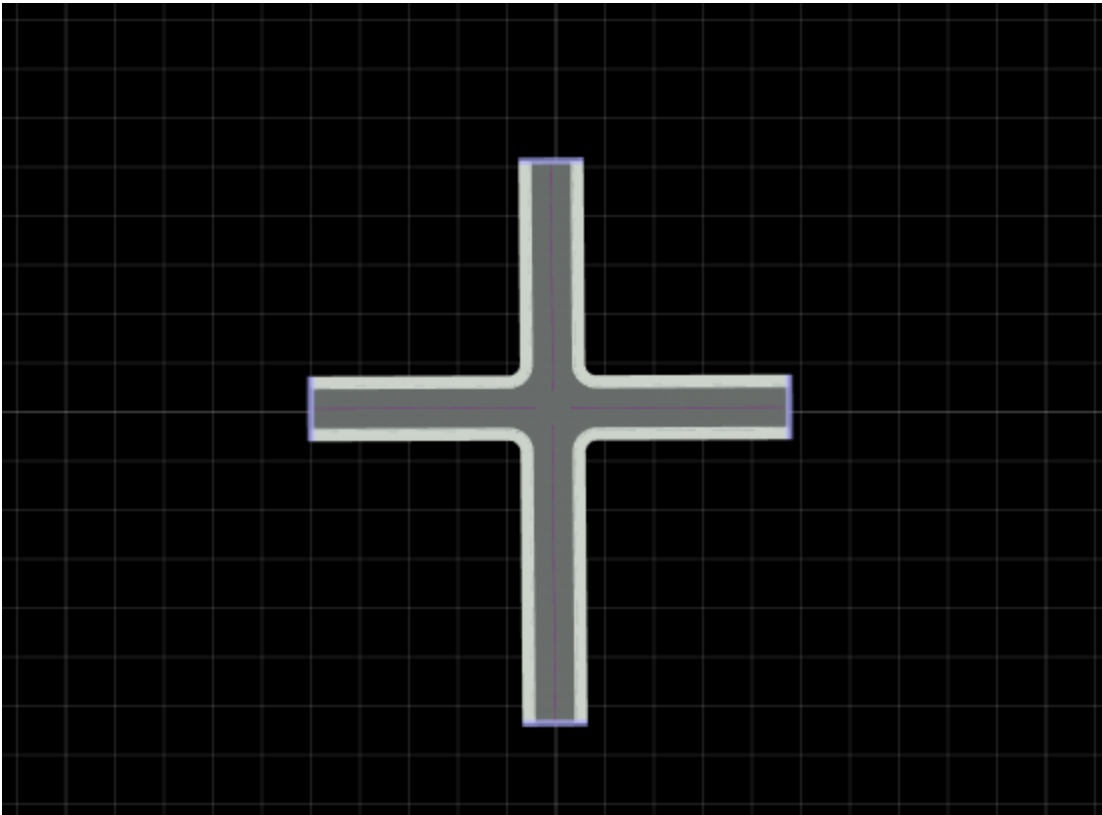
Create and Modify Construction Types Along a Road

See “Span Editing”.

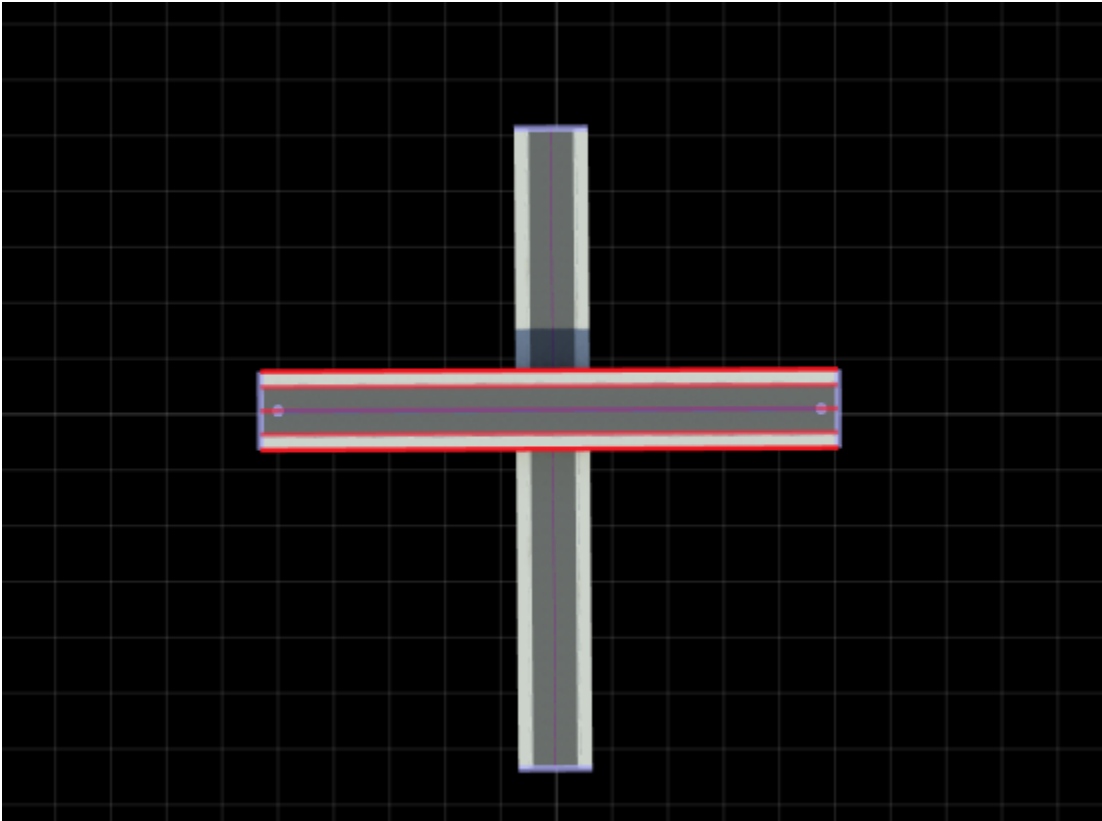
Fix Bridges Along a Road

When you create scenes that have bridges, or build scenes that have bridged roads such as by using the **Scene Builder Tool**, the initially created bridge spans might not form correctly. This example shows how to use the **Road Construction Tool** to fix such a bridge span in a scene.

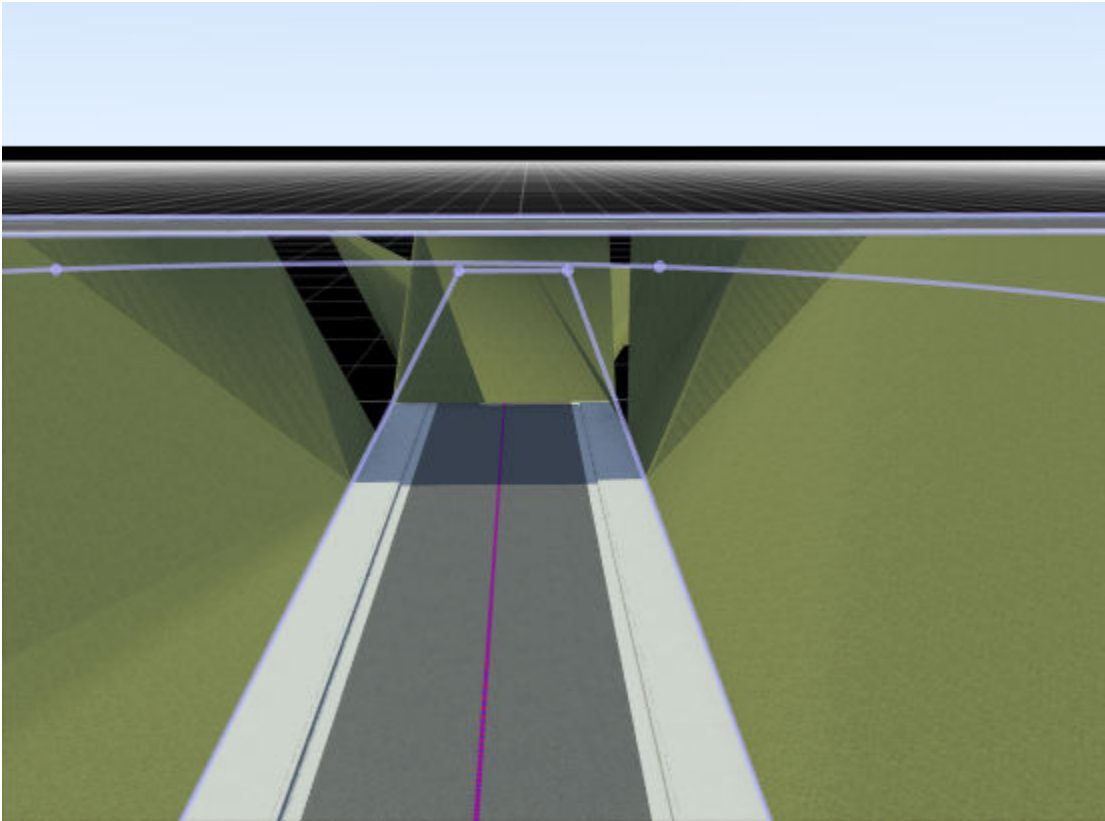
Create two intersecting roads by using the **Road Plan Tool**.



Click one of the roads to select it. In the **2D Editor** pane, drag the selected road until it is 10 meters above the other road.



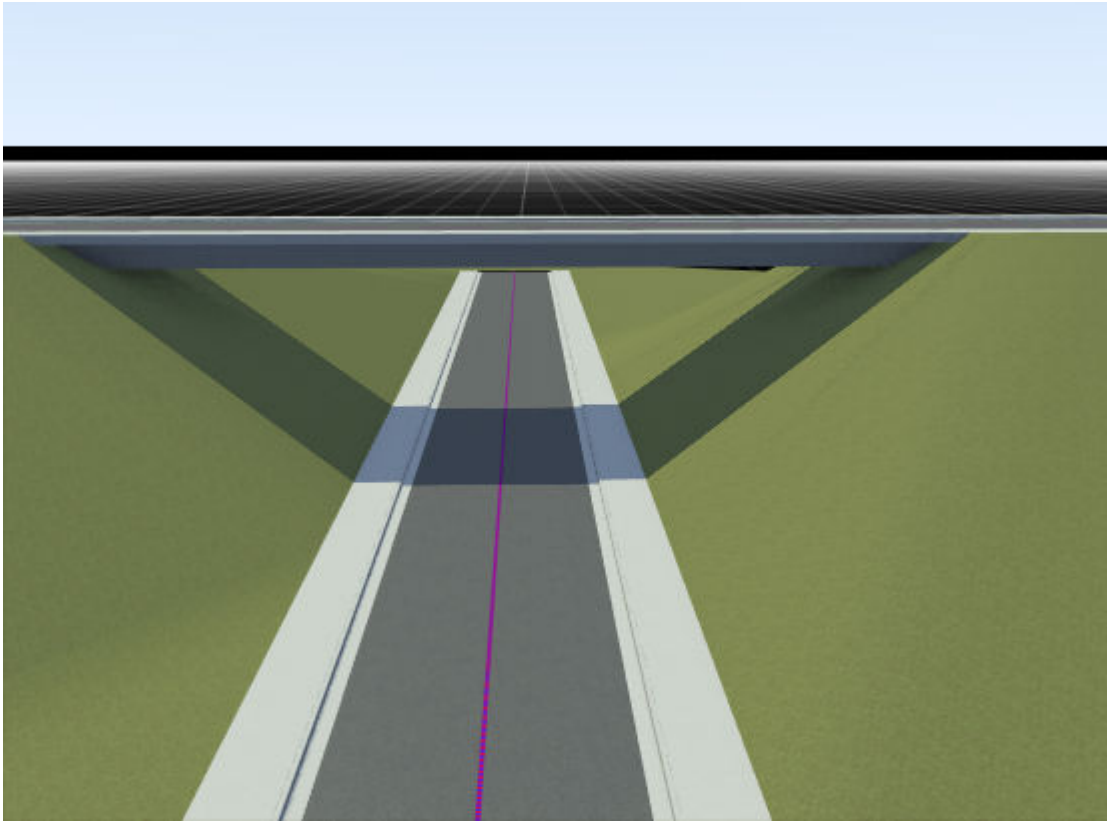
Create ground terrain around the roads by using the **Surface Tool**. Move the camera to view the road intersection. The ground attaches to the elevated road, which produces visual artifacts. These visual artifacts occur because the section of road above the other road is not designated as a bridge span.



To fix the visual artifacts, set the elevated road to be a bridge span. The **Surface Tool** ignores bridge road spans and does not attach to these spans.

- 1** Click the **Road Construction Tool** button.
- 2** Click the elevated road to select it.
- 3** In the toolbar on the left, click the **Auto Assign Bridges** button.
- 4** In the **Auto Bridge Span** window, use the **Bridge Span Inflation** option to optionally set the length of the bridge span. By default, the tool extends the bridge span by 20 meters on either side of the bottom road. If the remaining road length to the left or right of the road is less the **Bridge Span Inflation** value, then the bridge span extends to the end of the road. Click **OK**.
- 5** In the **Auto Bridge Span Results** window, confirm that the road was updated. Remember:
 - If you selected the bottom road, then the tool processes the road but does not create a bridge span.
 - If you previously created a bridge span for the elevated road, then the tool overrides that original bridge span.

Close the window and view the road that has the created bridge span.



With the **Road Construction Tool** still selected, if you select the middle portion of the road, the **Attributes** pane shows that this portion has its **Construction Span** value set to **Bridge**. The other two portions are set to **Standard**.

If you have multiple bridges to fix, then you can select multiple roads and use **Auto Assign Bridges** to fix all of them at once. If you are building scenes by using RoadRunner Scene Builder, then you can use the **Auto Detect Bridges** option to fix bridges. Using this option is similar to running the **Auto Assign Bridges** operation on all bridges in a scene. For more details, see **Scene Builder Tool**.

Parameters

Attribute	Description
Construction Span	<ul style="list-style-type: none"> • Standard — This portion of the road is at ground level. The surrounding terrain attaches to the sides of the span. • Bridge — This portion of the road is elevated. The surrounding terrain is not connected to the sides of the span. A bridge underside mesh is generated along the span. <p>When you select this option, the resulting bridge span might be too long. To address this issue, choose one of these options:</p> <ul style="list-style-type: none"> • Use the Road Chop Tool to divide the road into sections. Then, select the section that you want to convert into a bridge and set Construction Span to Bridge. • Use the Auto Assign Bridges button to create a bridge span.
Material	Material asset to be used for the sides and underside of the bridge. See Material Assets .

Version History

Introduced in R2020a

See Also

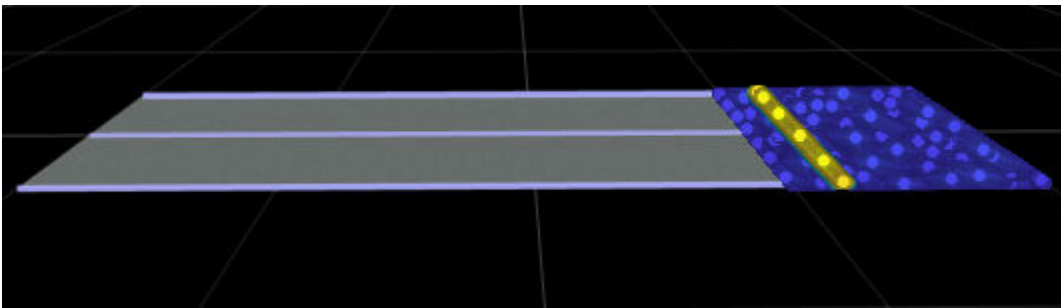
Scene Builder Tool

Road CRG Tool

Apply and visualize road surface data

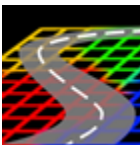
Description

The **Road CRG Tool** enables you to visualize road surface data specified using the curved regular grid (CRG) format. You can generate synthetic CRG data in RoadRunner by using **Synthetic CRG Assets**. Alternatively, you can import ASAM OpenCRG® files that store road surface data in the CRG format. You can import one or more ASAM OpenCRG files with or without an ASAM OpenDRIVE file. You can control the length and location of the road segment to which you apply the CRG data.



Open the Road CRG Tool

On the RoadRunner toolbar, click the **Road CRG Tool** button:



Examples

Create CRG Asset by Importing File

Follow these steps to create a CRG asset by importing an ASAM OpenCRG file. You can also create a new CRG asset within RoadRunner and modify the created asset. For more information on creating and modifying CRG assets within RoadRunner, see **Synthetic CRG Assets**.

- 1 Navigate to the folder in the **Library Browser** to which you want to add one or more ASAM OpenCRG files.
- 2 Using the file explorer for your operating system, select the ASAM OpenCRG files, and any associated ASAM OpenDRIVE file, in their source folder.
- 3 Drag the selected files into the **Library Browser**.

View and Modify Attributes of Imported CRG Asset

- 1 Select an imported CRG asset in the **Library Browser**.

2 View these attribute in the **Attributes** pane.

- **Samples Per Square Meter** — Specifies the density of the grid points showing road surface data. The default value of this parameter is 2500. You can modify this value.

When the total number of samples for a CRG asset exceeds the value of the **Max Samples per CRG Asset** parameter, RoadRunner changes the value of the **Samples Per Square Meter** parameter to a value that reduces the number of samples in the CRG asset to less than the **Max Samples per CRG Asset** parameter value. For more information on the **Max Samples per CRG Asset** parameter, see “Modify Max Samples per CRG Asset Parameter” on page 1-118.

- **Z Minimum** — Specifies the minimum height value of the road surface data. Units are in meters.

This attribute is read-only.

- **Z Maximum** — Specifies the maximum height value of the road surface data. Units are in meters.

This attribute is read-only.

- **Grid Length** — Specifies the length of the grid. Units are in meters.

This attribute is read-only.

- **Grid Width** — Specifies the width of the grid. Units are in meters.

This attribute is read-only.

Modify Max Samples per CRG Asset Parameter

The **Max Samples per CRG Asset** parameter specifies the maximum number of samples that RoadRunner can process for a CRG asset. The default value of this parameter is 5000000.

This parameter enables you to reduce the time required to load large CRG assets by adjusting the value of the **Samples Per Square Meter** parameter. For example, given the default value for **Samples Per Square Meter** and **Max Samples per CRG Asset**, if you import a CRG asset with a size of 3,000 square meters, RoadRunner automatically reduces the value of Samples Per Square Meter to reduce the number of samples in the asset from 7500000 to less than 5000000.

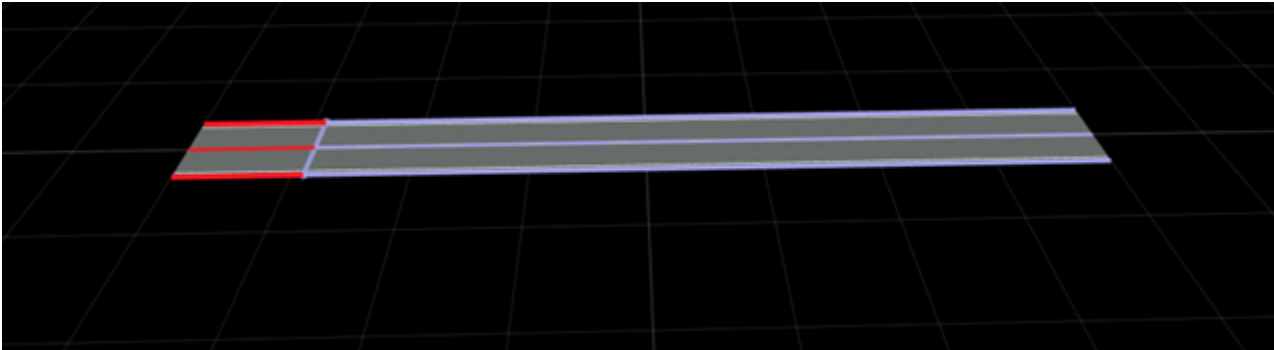
You can modify the value of the **Max Samples per CRG Asset** parameter depending on your computer configuration. If you are using a high-end computer configuration, consider increasing the value of the **Max Samples per CRG Asset** parameter. If you encounter performance issues, consider reducing the value of this parameter.

To modify the value of the **Max Samples per CRG Asset** parameter, follow these steps:

- 1** Select **Edit > Preferences**.
- 2** In the Application Preferences window, modify the value of the **Max Samples per CRG Asset** parameter.

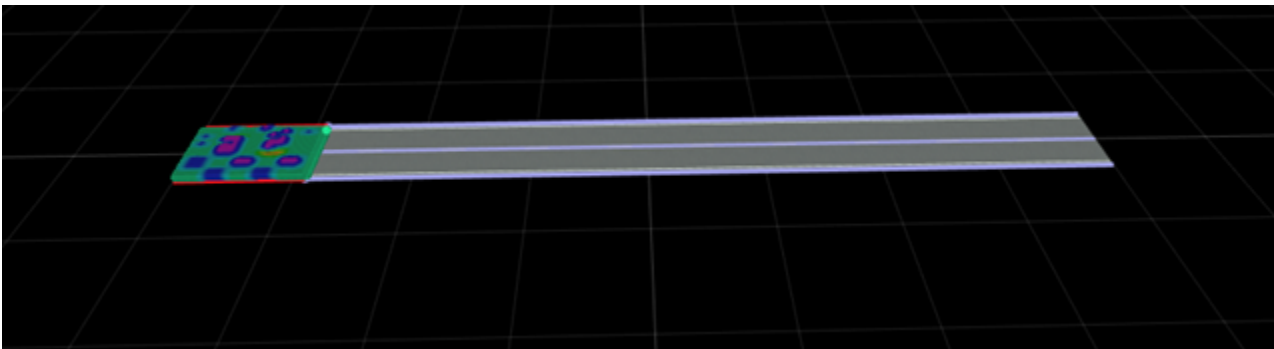
Note You must restart RoadRunner to apply your changes to the **Max Samples per CRG Asset** value.

Apply CRG Data to Road Segment



- 1 Click the **Road CRG Tool** button on the toolbar.
- 2 Select the road to which you want to apply road surface data.
- 3 Right-click a location on the road to insert a span node that divides the road into two segments. Insert more span nodes if you want to divide the road into more segments.
- 4 Select one of the road segments.
- 5 Drag the ASAM OpenCRG file (.crg) or synthetic CRG asset file (.rrcrg) from the **Library Browser** to the **CRG file** asset holder in the **Attributes** pane.

Visualize CRG Data



- 1 Click the **Road CRG Tool** button on the toolbar.
- 2 Select the road for which you want to visualize CRG data. This highlights the road and displays the span nodes using light blue lines.
- 3 Select a road segment bounded by span nodes. If CRG data has been applied to the selected road segment, then RoadRunner visualizes the road surface variations using a colormap.

Change CRG Color Gradient for Visualizing CRG Data

RoadRunner supports these color gradients for visualizing CRG data.

Parula (default) | Bone | HSV | Autumn | Cool | Hot | Copper | Gray | Jet | Pink | Spring | Summer | Turbo | White | Winter

Follow these steps to change a color gradient.

- 1 Select **Edit > Preferences**.
- 2 In the Application Preferences window, select the desired **CRG Color Gradient** from the drop-down list.

Note You must restart RoadRunner to apply your changes to the **CRG Color Gradient** value.

Edit CRG Span

When the length of the selected road span does not match the length of the CRG reference line, the **Road CRG Tool** uses the lesser of these lengths to display surface data along the road. You can edit the length of the selected span to match it to the length of the CRG reference line. To edit the road span for the **Road CRG Tool**, move one of the span nodes by following these steps:

- 1 Select the road containing the span.
- 2 Select the desired node and drag it along the road.

For more details on operating span instances, see “Span Editing”.

Replace CRG Data from Road Segment

- 1 Click the **Road CRG Tool** on the toolbar.
- 2 Select the road and road segment for which you want to replace CRG data.
- 3 Select the desired CRG file from a folder within the **Library Browser**.
- 4 Drag the CRG file into the **CRG file** asset holder in the **Attributes** pane.

Remove CRG Data from Road Segment

- 1 Click the **Road CRG Tool** on the toolbar.
- 2 Select the road and road segment from which you want to remove CRG data.
- 3 Right-click the **CRG file** asset holder in the **Attributes** pane, and select **Clear**. Alternatively, select the **CRG file** asset holder and press **Delete**.

Export CRG Data

You can export CRG data to an ASAM OpenCRG file when you export a scene to an ASAM OpenDRIVE file. For more information, see “Export to ASAM OpenCRG”.

Parameters

Attribute	Description
CRG file	Filename of the assigned CRG asset.
CRG Orientation	<p>Orientation of the CRG data, specified as one of these values:</p> <ul style="list-style-type: none"> • Same — CRG data is laid out in the same direction as the road digitization direction. • Opposite — CRG data is laid out in the direction opposite to the road digitization direction. Do not specify this value when CRG Mode is set to OpenCRG - Absolute Height (genuine). <p>Default: Same</p>

Attribute	Description
CRG Mode	<p>Mode of combining the CRG data with the road data, specified as one of these values:</p> <ul style="list-style-type: none"> • Road Curve - Relative Height (attached) — This mode uses the reference line of the road and adds the height specified in the CRG data to the height of the road in the scene. • Road Curve - Absolute Height (attached0) — This mode disregards the height of the road in the scene. Instead, RoadRunner exclusively uses the height specified in the CRG data to represent the road surface. • OpenCRG - Absolute Height (genuine) — This mode uses the reference line of the CRG data by positioning it relative to the reference line of the road. The geometry of the CRG data and of road data must match. Similarly to the attached0 mode, RoadRunner exclusively uses the height specified in the CRG data to represent the road surface. • OpenCRG - Absolute Height (global) — This mode references CRG data in its native coordinate system without applying translation and rotation. This mode disregards the height of the road in the scene. Instead, RoadRunner exclusively uses the height specified in the CRG data to represent the road surface. <p>To visualize the CRG data, the road surface defined by the CRG data must overlap with a selected road segment.</p> <hr/> <p>Note This mode is not supported when you attach synthetic CRG data to a road segment.</p> <p>Default: Road Curve - Relative Height (attached)</p>

Attribute	Description
CRG Purpose	Physical purpose of the CRG data, specified as one of these values: <ul style="list-style-type: none"> • Elevation — CRG data specifies height values for the road surface. • Friction — CRG data specifies friction values for the road surface. Default: Elevation
Z Offset	z-offset between the reference lines of the road and the CRG data, specified as a real scalar. Do not specify this parameter when the CRG Purpose is set to Friction . Units are in meters. Default: 0
Z Scale	Factor for scaling CRG data along the z-direction, specified as a real scalar. Do not specify this parameter when the CRG Purpose is set to Friction . Default: 1
S Offset	s-offset between the reference lines of the road and the CRG data, specified as a real scalar. Units are in meters. Default: 0
T Offset	t-offset between the reference lines of the road and the CRG data, specified as a real scalar. Units are in meters. Default: 0
Heading Offset	Heading offset between reference lines of the road and the CRG data, specified as a real scalar. Units are in degrees. Only specify this parameter when CRG Mode is set to OpenCRG - Absolute Height (genuine) . Default: 0

To create a new synthetic CRG asset for a road segment, in the **Attributes** pane, select **Create Synthetic CRG Asset**. For more information, see “Create and Apply Synthetic CRG Data from Attributes Pane” on page 2-47.

Note Specify **S Offset** and **T Offset** values such that the road surface defined by the CRG data overlaps with a selected road segment. You cannot visualize CRG data outside the extent of the selected road segment.

Version History

Introduced in R2021b

See Also

Synthetic CRG Assets | OpenDRIVE Viewer Tool

Topics

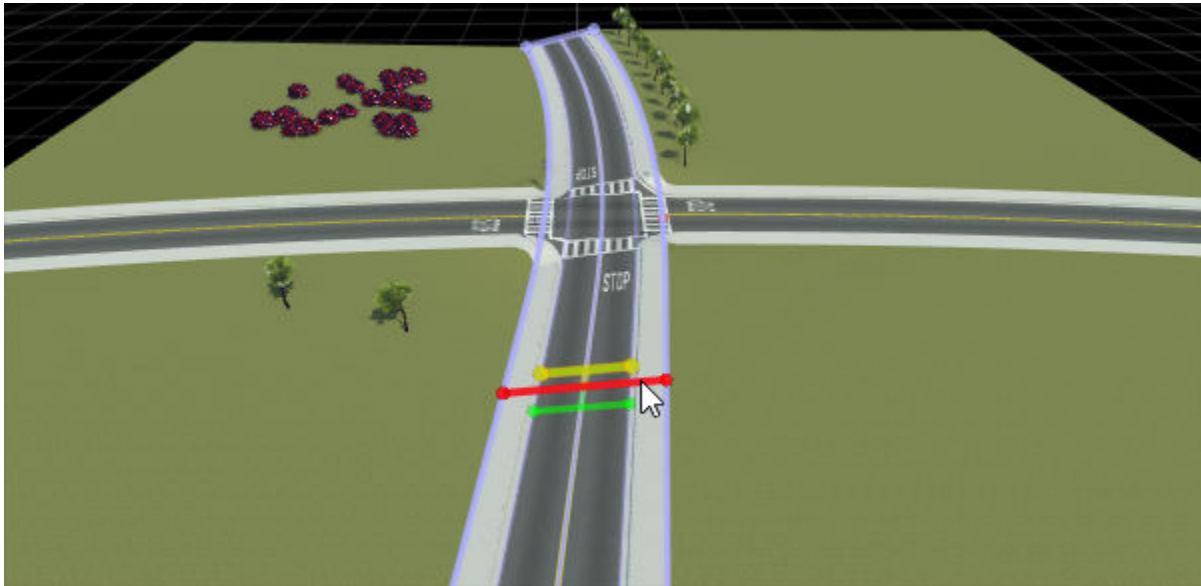
"Importing ASAM OpenCRG Files"
"Importing ASAM OpenDRIVE Files"
"Export to ASAM OpenCRG"
"Export to ASAM OpenDRIVE"
"Span Editing"

Road Superelevation Tool

Adjust superelevation (slope and banking angle) for full width of road

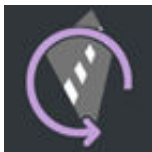
Description

The **Road Superelevation Tool** enables you to adjust the superelevation (slope and banking angle) for the full width of a road at specified distances. To modify the slope or banking angle at specified cross-sections of a road, use the **Cross Section Tool** instead.



Open the Road Superelevation Tool

On the RoadRunner toolbar, click the **Road Superelevation Tool** button:



Examples

Adjust Banking Angle Along Road

- 1 Open the `FourWayStop.rrscene` scene, which is one of the prebuilt scenes that is included with newly created RoadRunner projects. Zoom in on the scene and rotate the camera to view the scene at an angle. All roads in this scene are flat.



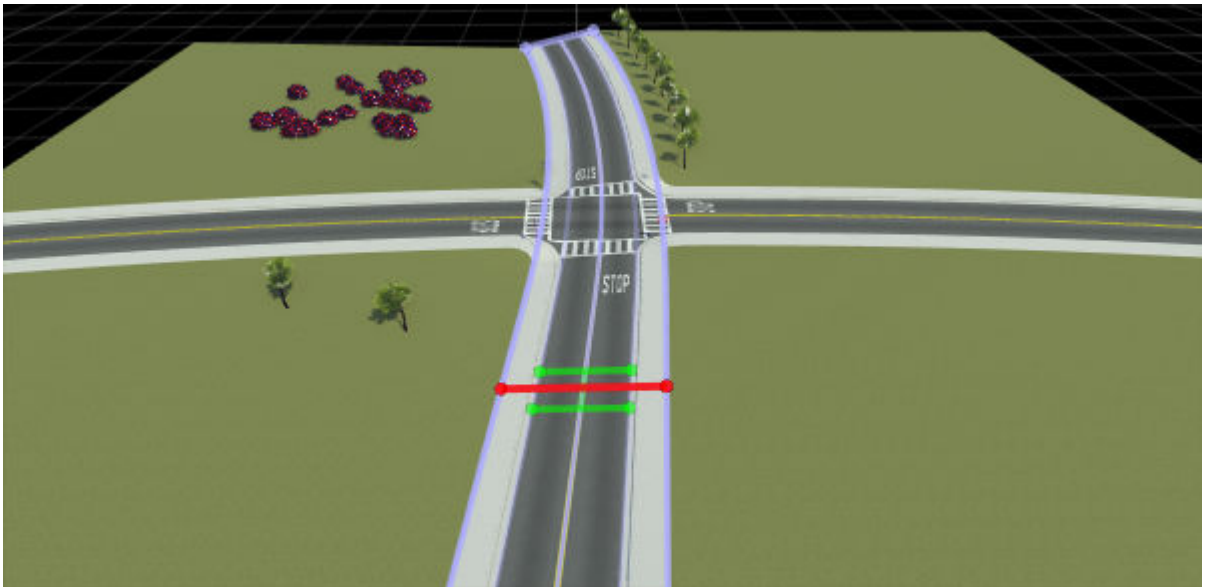
- 2 Click the **Road Superelevation Tool** button.
- 3 Click to select one of the roads. Then, in the **Attributes** pane, set the **Angle** attribute to 10 degrees. The banking angle of the road increases by 10 degrees along the length of the road.



- 4 Right-click at a point along the road to add a superelevation node. The node appears as a red bar across the width of the road, with smaller green tangent nodes for adjusting the slope at this node. To adjust the position of the superelevation node along the road, click and drag the red bar.



- 5 In the **Attributes** pane, decrease the **Angle** value of the selected node from 10 to 5. The road now has a banking angle of 5 degrees at this node. RoadRunner interpolates the banking angle between nodes. Because the entire road previously had a banking angle of 10 degrees, the banking angle now gradually increases from 5 degrees at this node to 10 degrees at the road edges.



Parameters

Attribute	Description
Angle	Superelevation angle, in degrees, specified as a real scalar. Angle is clockwise-positive relative to the direction in which the road was created. You can specify the superelevation angle of the entire road or at superelevation nodes along the road.
Distance	Position of superelevation node along the road, in meters, specified as a nonnegative real scalar. Distance is relative to the edge of the road that was created first.
Slope	Slope of superelevation node tangent, in degrees per meter, specified as a real scalar.

Version History

Introduced in R2021a

See Also

Cross Section Tool

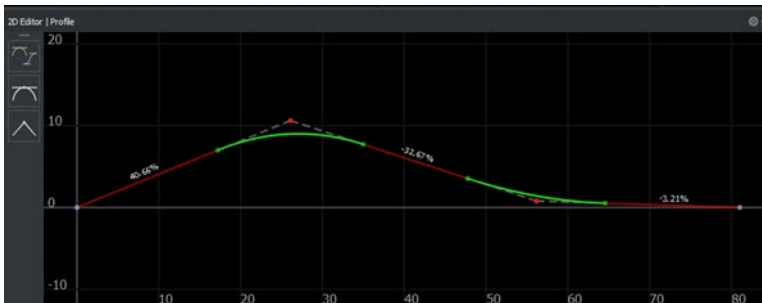
Road Height Tool

Manipulate vertical profile of roads

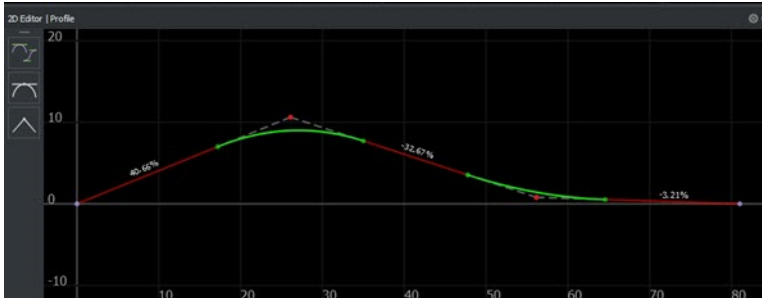
Description

The **Road Height Tool** allows the manipulation of the vertical profile of roads.

The height profile is defined relative to the distance along the road and is constructed of straight line sections and quadratic (parabolic) blend sections. The height profile can also be converted to a sequence of cubics with tangents on nodes.



For tips on dealing with height alignment in intersections, see “Resolve Triangulation Issues in Junctions”.



Open the Road Height Tool

On the RoadRunner toolbar, click the **Road Height Tool** button:



Examples

Insert a New Height Node

- 1 Click the **Road Height Tool** button.
- 2 Click to select a road.
- 3 Right-click the road curve at which you want to insert a height node.
- 4 Optionally, you can hold the right-click button and drag the height node to the desired height.

Alternatively, insert a height node from the **2D Editor** pane.

- 1 Click the **Road Height Tool** or **Road Plan Tool** button.
- 2 Click to select a road.
- 3 In the **2D Editor** pane, right-click the road curve where you want to insert a height node.

Adjust the Height of a Node

- 1 Click the **Road Height Tool** button.
- 2 Click to select a road.
- 3 Click and drag the node you want to edit to the desired height.
- 4 Optionally, click the node and type the desired height into the **Attributes** pane.

Alternatively:

- 1 Click the **Road Height Tool** or **Road Plan Tool** button.
- 2 Click to select a road.
- 3 In the **2D Editor** pane, click and drag the node along the vertical axis.

Move a Height Node Along the Road

- 1 Click the **Road Height Tool** button.
- 2 Click to select a road.
- 3 Click to select a node.
- 4 Adjust the **Distance** value in the **Attributes** pane.

Alternatively:

- 1 Click the **Road Height Tool** or **Road Plan Tool** button.
- 2 Click to select a road.
- 3 In the **2D Editor** pane, click and drag the node along the horizontal axis.

Delete a Road Height Node

- 1 Click the **Road Height Tool** button.
- 2 Click to select a road.
- 3 Click to select a node.
- 4 Press the **Delete** key or select **Edit > Delete** from the menu bar.

Adjust the Height of a Section Between Nodes

- 1 Click the **Road Height Tool** button.
- 2 Click to select a road.
- 3 Click and drag the curve section you want to edit to the desired height.

- 4 Optionally, click the curve section and type the desired height into the **Attributes** pane. This value sets the height of the nodes at the start and end of the curve section.

Alternatively:

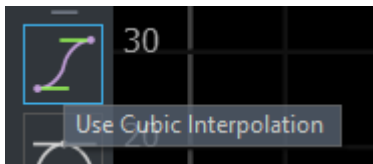
- 1 Click the **Road Height Tool** or **Road Plan Tool** button.
- 2 Click to select a road.
- 3 In the **2D Editor** pane, click and drag the curve section along the vertical axis.

Adjust the Size of a Quadratic Blend Section

- 1 Click the **Road Height Tool** button.
- 2 Click to select a road.
- 3 Click the height node in the middle of the blend section you want to edit. Two green dots appear at the limits of the blend region.
- 4 Click and drag either of the green dots to adjust the range of the blend region.
- 5 Optionally, you can type the desired blend range into the **Attributes** pane. This value sets the height of the nodes at the start and end of the curve section.

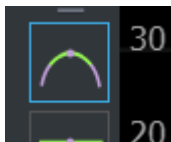
Convert the Profile to Cubic Interpolation

- 1 Click the **Road Height Tool** button.
- 2 Click to select a road.
- 3 Click **Use Cubic Interpolation**.



Convert the Profile to Quadratic Interpolation

- 1 Click the **Road Height Tool** button.
- 2 Click to select a road.
- 3 Click **Use Quadratic Interpolation**.



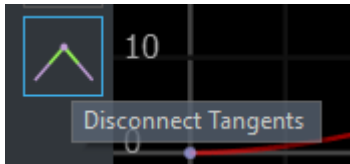
Adjust the Tangent of a Height Node

- 1 Click the **Road Height Tool** button.
- 2 Click to select a road.
- 3 Click the height node.
- 4 Click and drag either of the green dots to adjust the slope.

Disconnect the Tangents of a Height Node

- 1 Click the **Road Height Tool** button.
- 2 Click to select a road.

- 3 Click the height node.
- 4 Click **Disconnect Tangents**.



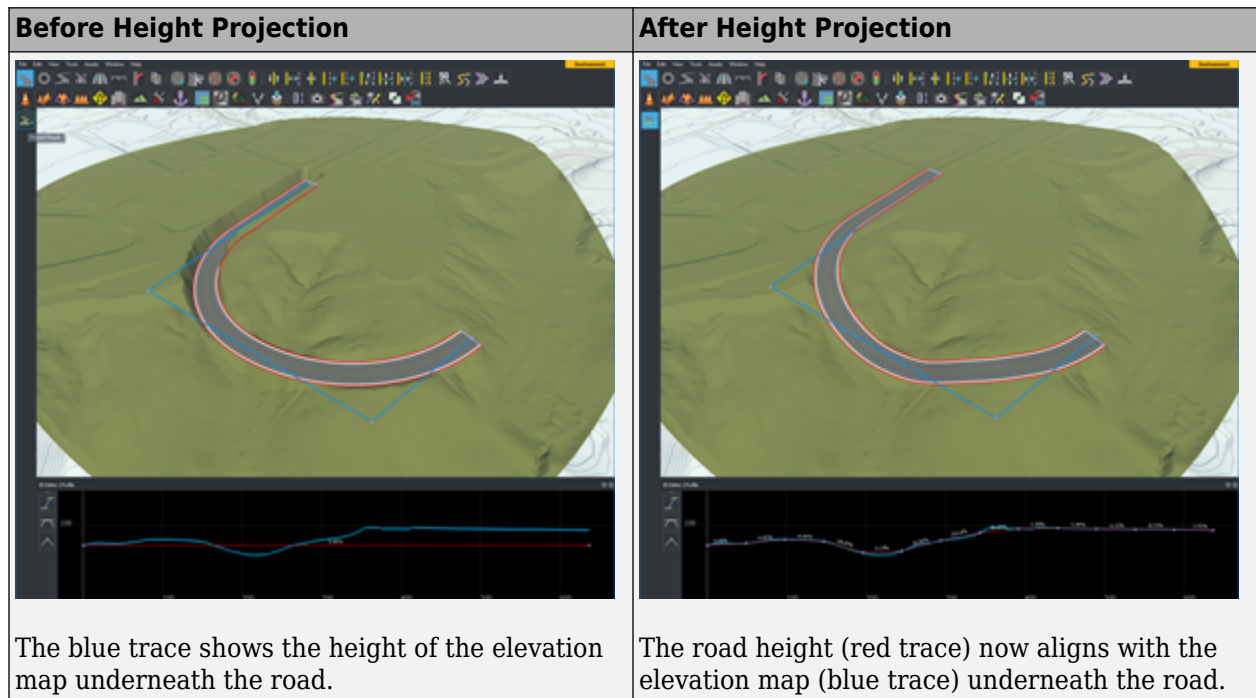
Connect the Tangents of a Height Node

- 1 Click the **Road Height Tool** button.
- 2 Click to select a road.
- 3 Click the height node.
- 4 Click **Connect Tangents**.



Project Roads to Elevation Maps

RoadRunner can project one or more roads down to the height of an elevation map (see **Elevation Map Assets**). This process samples the elevation data under the road and performs a constrained fit of the road elevation to the trace of the elevation data beneath the road.



- 1 Click the **Road Plan Tool** button.
- 2 Select the roads you want to project. (You can perform a **Select All** operation to select all roads in the scene).

3 Click the **Project Roads** button on the toolbar on the left.

There are two types of height projection that can be performed: a relaxed fit and a tight fit. The type of fit used depends on whether the road uses quadratic or cubic interpolations.

Relaxed Fit

If a road is using quadratic interpolation on page 1-130, a relaxed, approximate fit is used. This interpolation is best when the terrain data is either noisy or low resolution.

Tight Fit

If a road is using cubic interpolation on page 1-130, a much tighter fitting method is used. This interpolation is best when you want the road to closely match the heights of the elevation map.

Version History

Introduced in R2020a

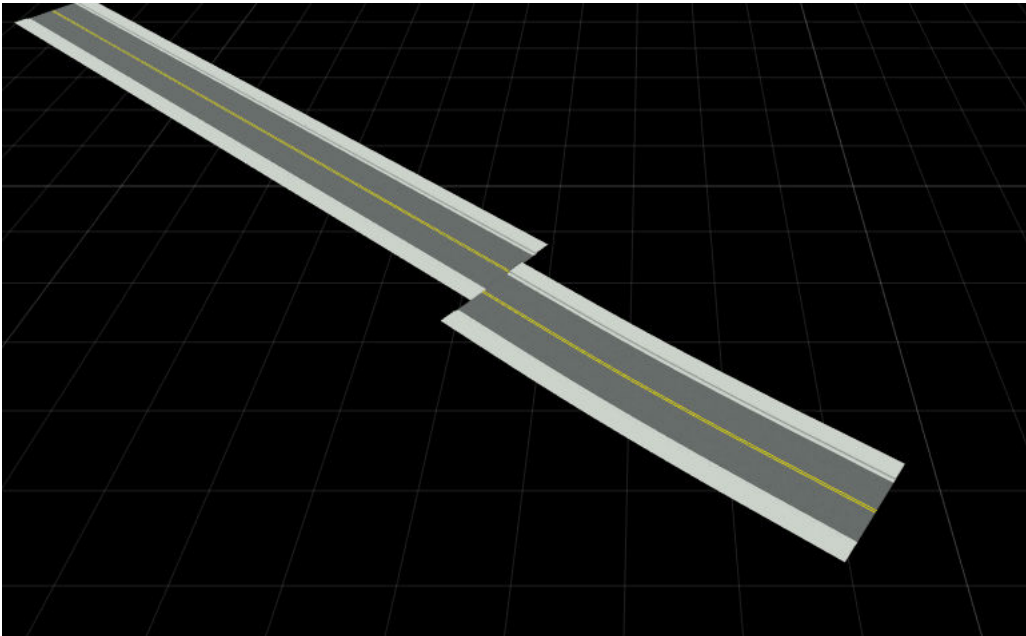
Road Offset Tool

Adjust connection between two end-to-end roads

Description

The **Road Offset Tool** is used to adjust the connection between two end-to-end roads. Roads can be shifted laterally to align the lanes of each road.

Note Slip roads cannot be offset using this tool. Slip roads will automatically align to the side of the road from which they originate.



Open the Road Offset Tool

On the RoadRunner toolbar, click the **Road Offset Tool** button:



Examples

Offset a Road

- 1** Click the **Road Offset Tool** button.
- 2** Click the desired road. The road must be connected to another road at its other end.
- 3** Click and drag the arrows at the desired end of the road.

Note Holding the **Ctrl** key disables lane snapping.

Version History

Introduced in R2020a

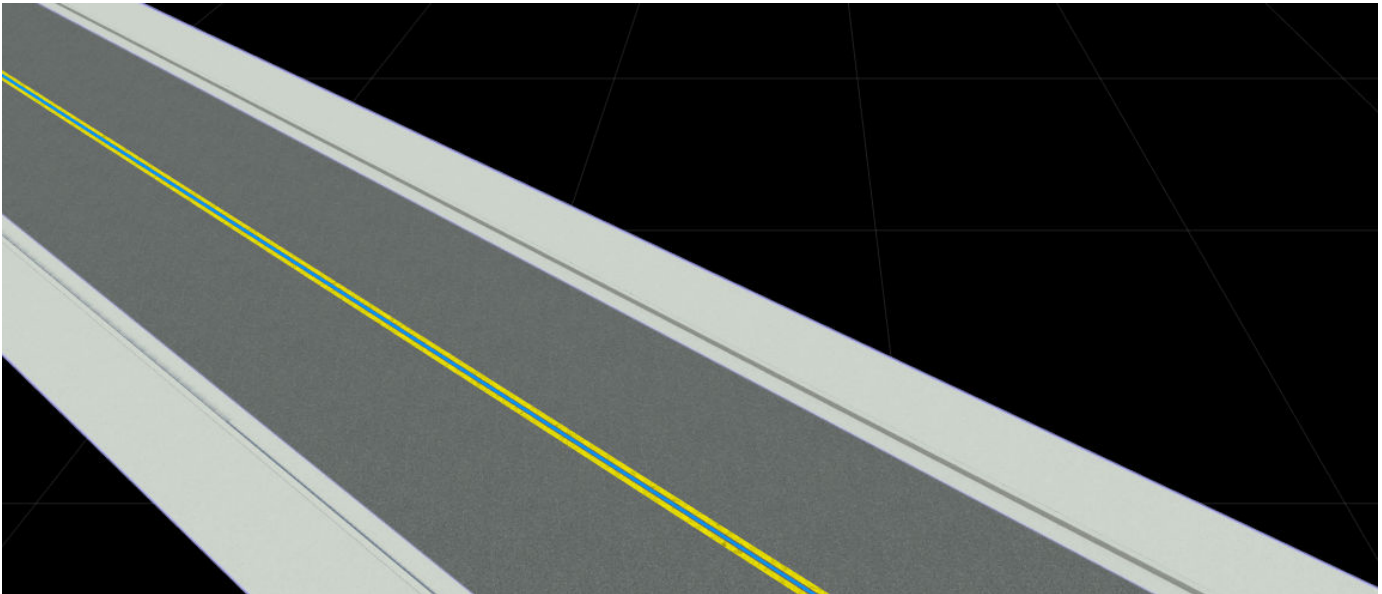
Road Plan Tool

Create and lay out roads

Description

The **Road Plan Tool** is the primary tool for creating and laying out roads. It allows for creation and manipulation of the 2D reference curve that the road layout is based on. The height of the road can be manipulated independently using the **Road Height Tool**. By default, the **Road Plan Tool** automatically creates intersections at locations where roads overlap. For information on how to avoid creating automatic junctions, see “Prevent Creation of Automatic Junctions Between Roads” on page 1-148.

Roads automatically participate in the terrain surface graph. For more information about this interaction, refer to the “How Surfaces Work in RoadRunner”.



Open the Road Plan Tool

On the RoadRunner toolbar, click the **Road Plan Tool** button:



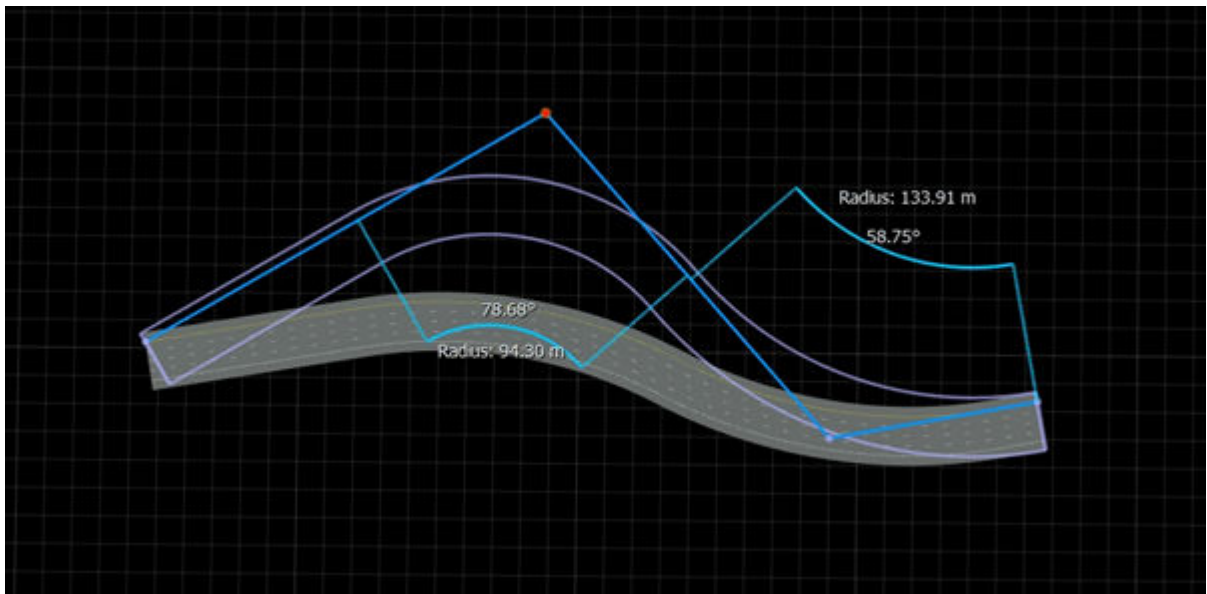
Examples

Create a New Road

- 1 Click the **Road Plan Tool** button.
- 2 If another road is already selected, click away from any road to unselect it.
- 3 Optionally, click the desired road style in the **Assets Browser** to build a road of a particular style. If no road style is picked, a basic default style will be used. For more information about road styles, see **Road Style Assets**.
- 4 Right-click at the location you want to start a new road.
- 5 Right-click additional times to create additional road control points to extend and shape the road.

Move Road Control Point

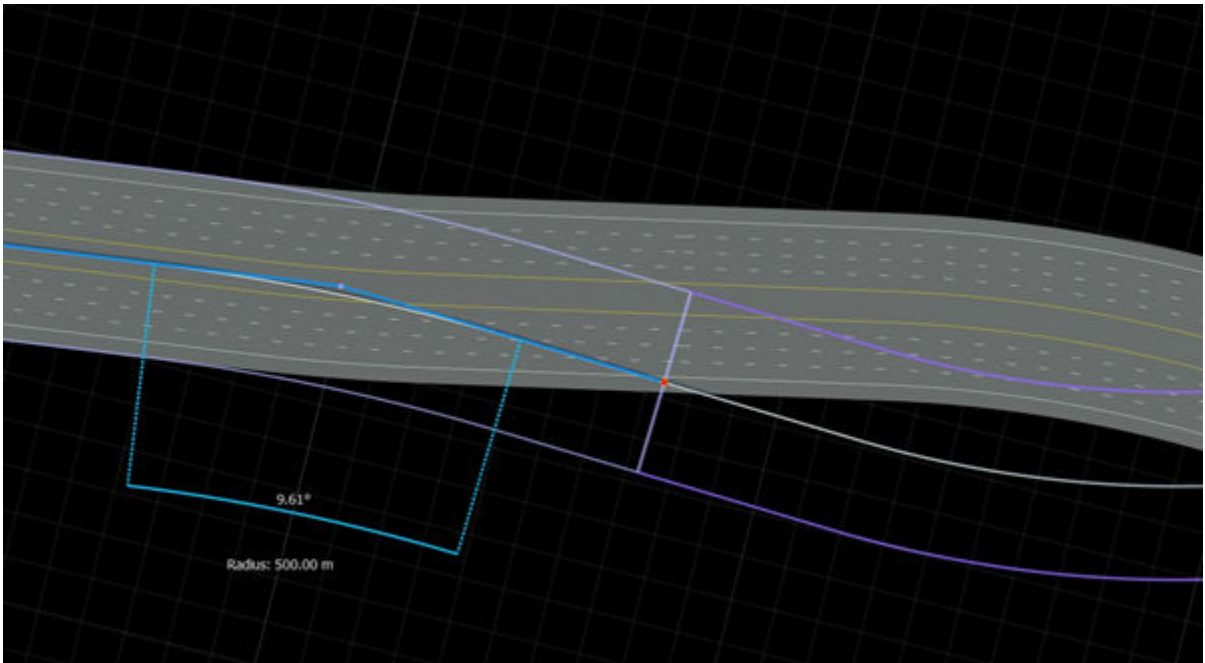
- 1 Click the **Road Plan Tool** button.
- 2 Click the road you want to edit. The road is highlighted, and the control points are displayed and connected by light blue lines.



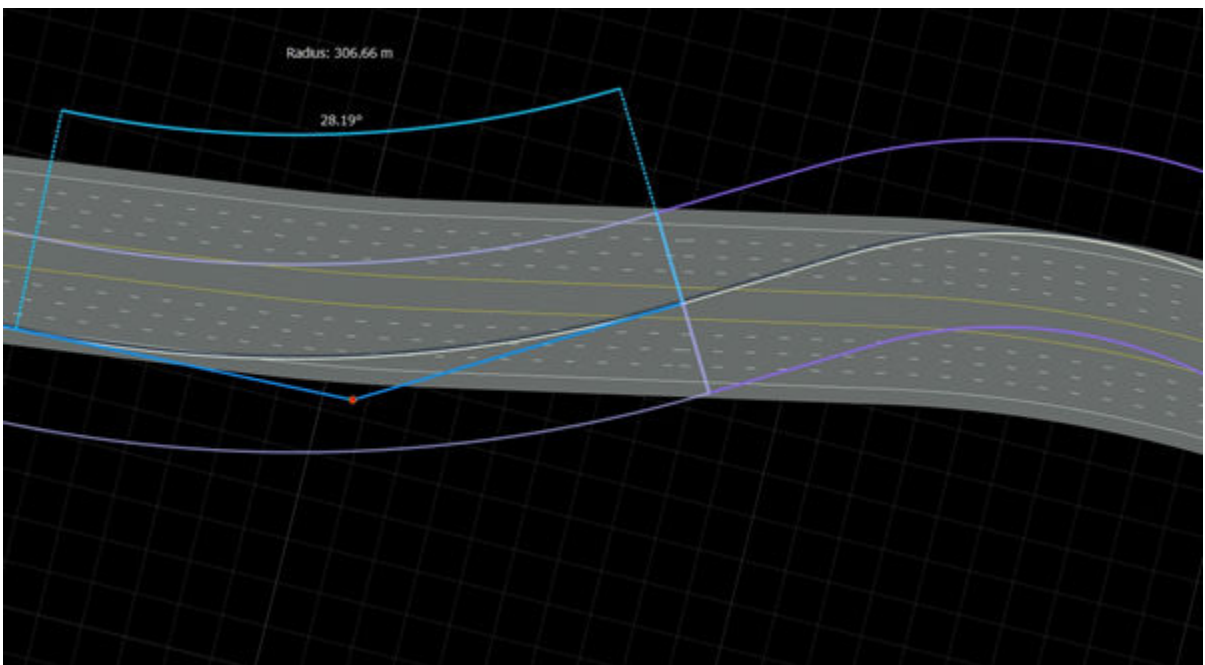
- 3 Click and drag the desired control point to move it.
- 4 Optionally, you can click to select the point, and then type a precise position in the **Attributes** pane.

Note The end control points and the first interior control points have some special properties when roads are connected end-to-end:

- Moving the end control point of one of the roads will move the end of the connected road and update the first interior control point of the other road to ensure that the road directions remain aligned at this end, as shown here:



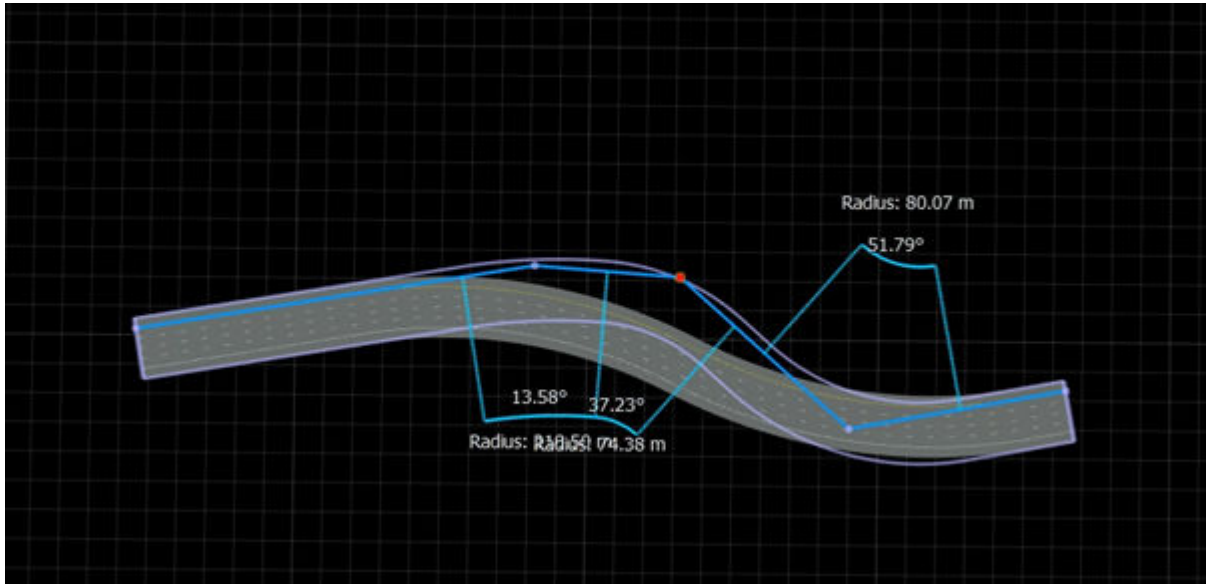
- Moving the first interior control point of one of the roads will move the first interior control point of the other road (by rotating it about the end point) to ensure that the road directions remain aligned at this end, as shown here:



Insert New Control Point Within Existing Road

- 1 Click the **Road Plan Tool** button.

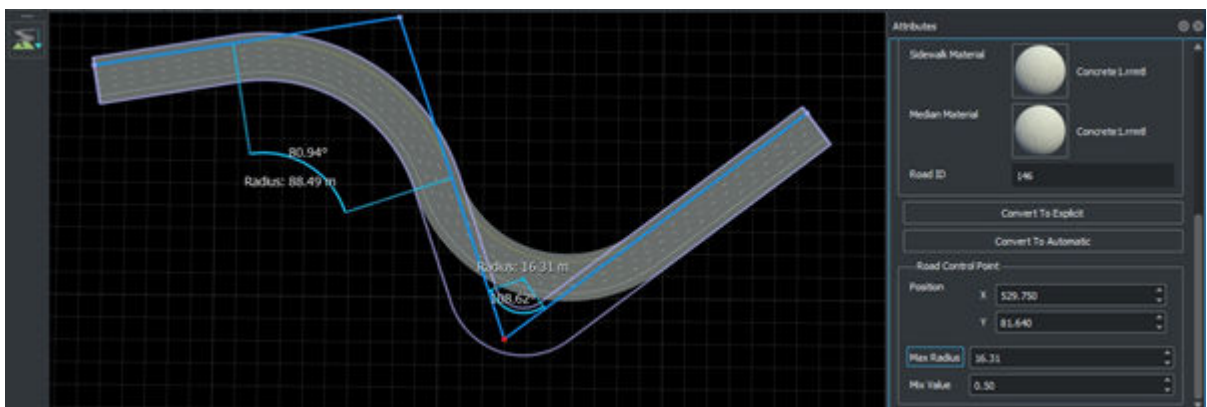
- 2 Click the road you want to edit.
- 3 Move the mouse cursor over the blue control line at the location you want to insert a node.
- 4 Right-click to insert a new node within the control line of the road.



Adjust Radius of Road Curve

By default, the circular arcs in the road curves will fit in the space available. If a smaller curve radius is desired:

- 1 Click the **Road Plan Tool** button.
- 2 Click the road you want to edit. The road is highlighted, and the control points are displayed and connected by light blue lines.
- 3 Click the control point closest to the circular arc you want to modify. The attributes of the selected control point will appear in the **Attributes** pane.
- 4 Adjust the **Max Radius** value to the desired radius. If you do not see anything change, then try a lower value, because **Max Radius** will limit the maximum radius of the arc.



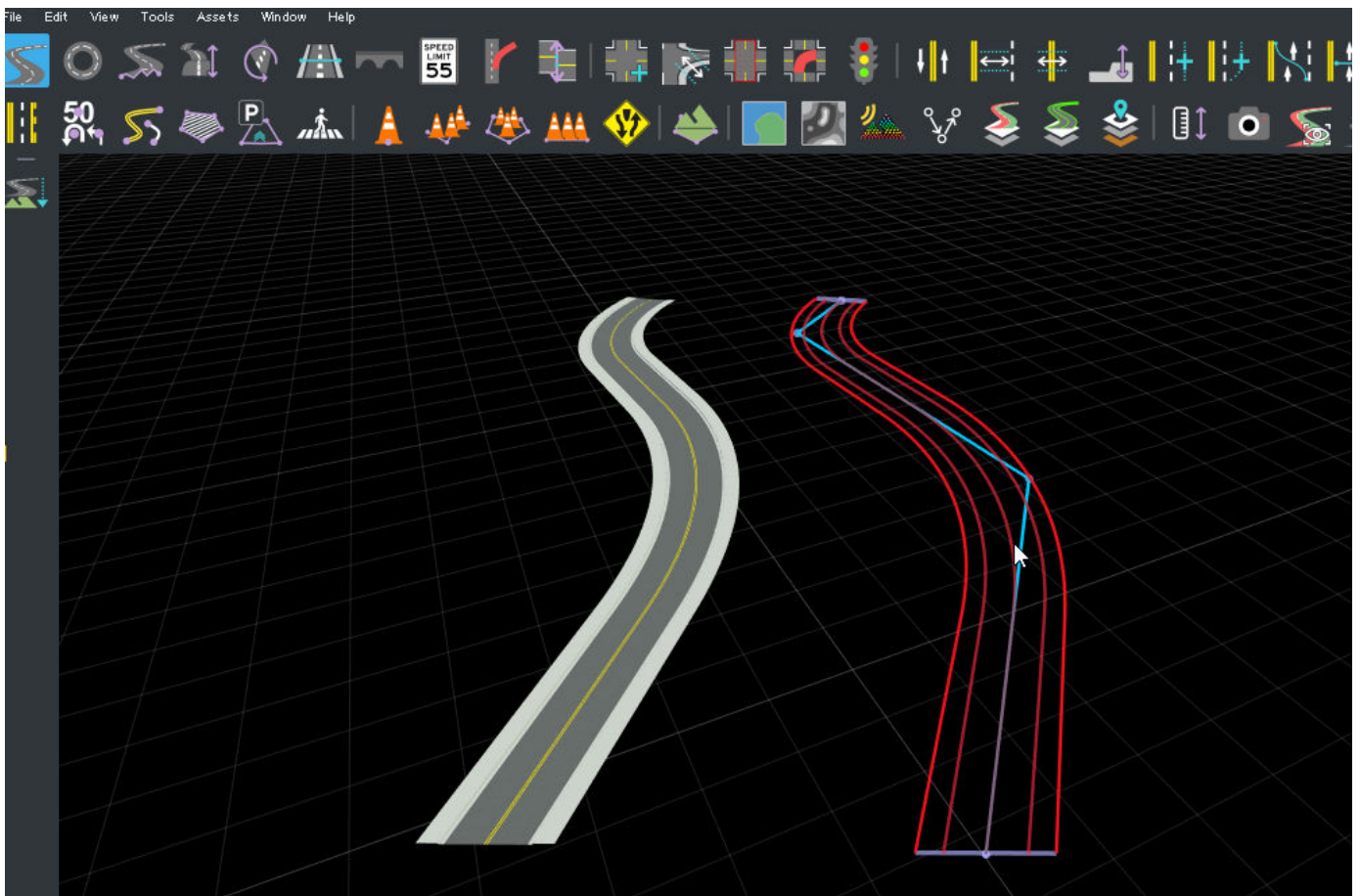
Adjust Curvature of Road Curve

See “Explicit Road Curves” on page 1-150.

Shift Single Road Using Control Line

You can shift a single road on the editing canvas using a control line.

- 1 Click the **Road Plan Tool** button.
- 2 Click the road you want to shift. The road is highlighted, and the control points are displayed and connected by light blue control lines.
- 3 To shift the entire road, drag one of the control lines. While dragging the light blue line, the canvas displays a preview of the shifted road as a red outline. Note that the road mesh does not move until you release the control line.



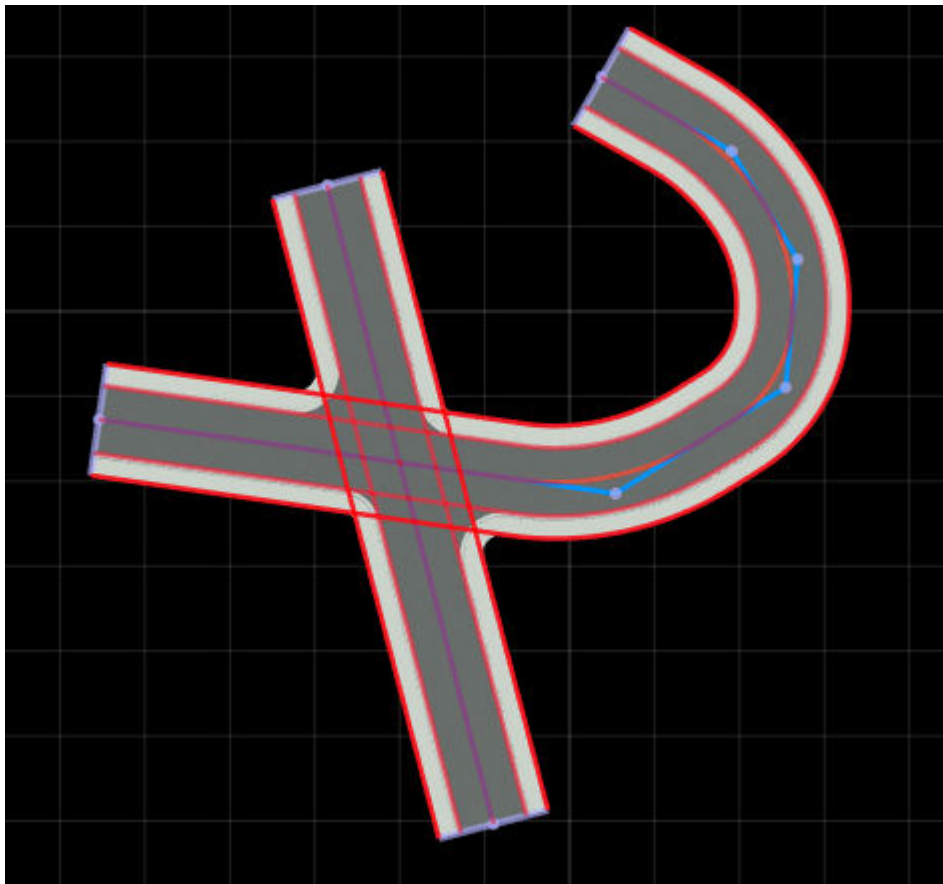
- 4 When you are satisfied with the road location indicated by the preview, release the control line. The road mesh shifts from the source location to the destination location.

Shift Multiple Roads Using Control Point

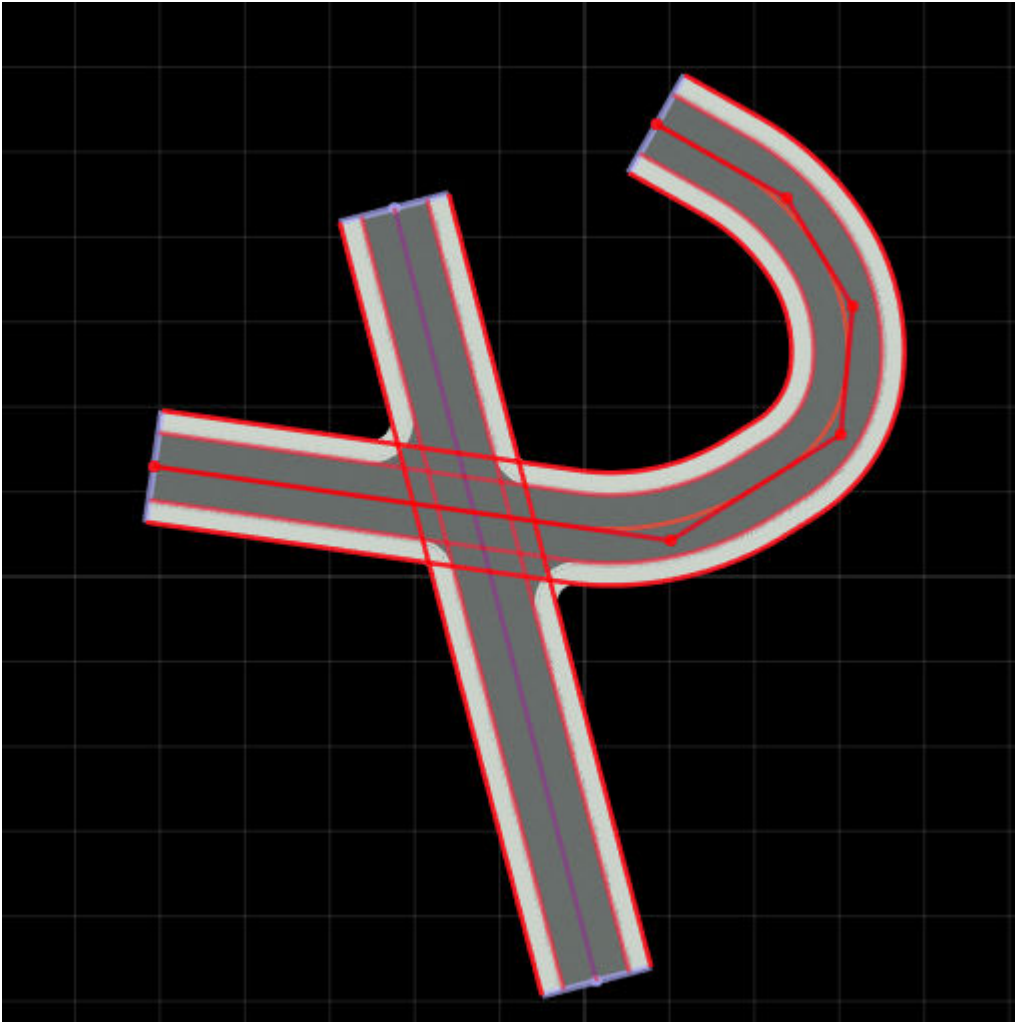
You can shift multiple roads on the editing canvas at the same time using control points.

- 1 Click the **Road Plan Tool** button.

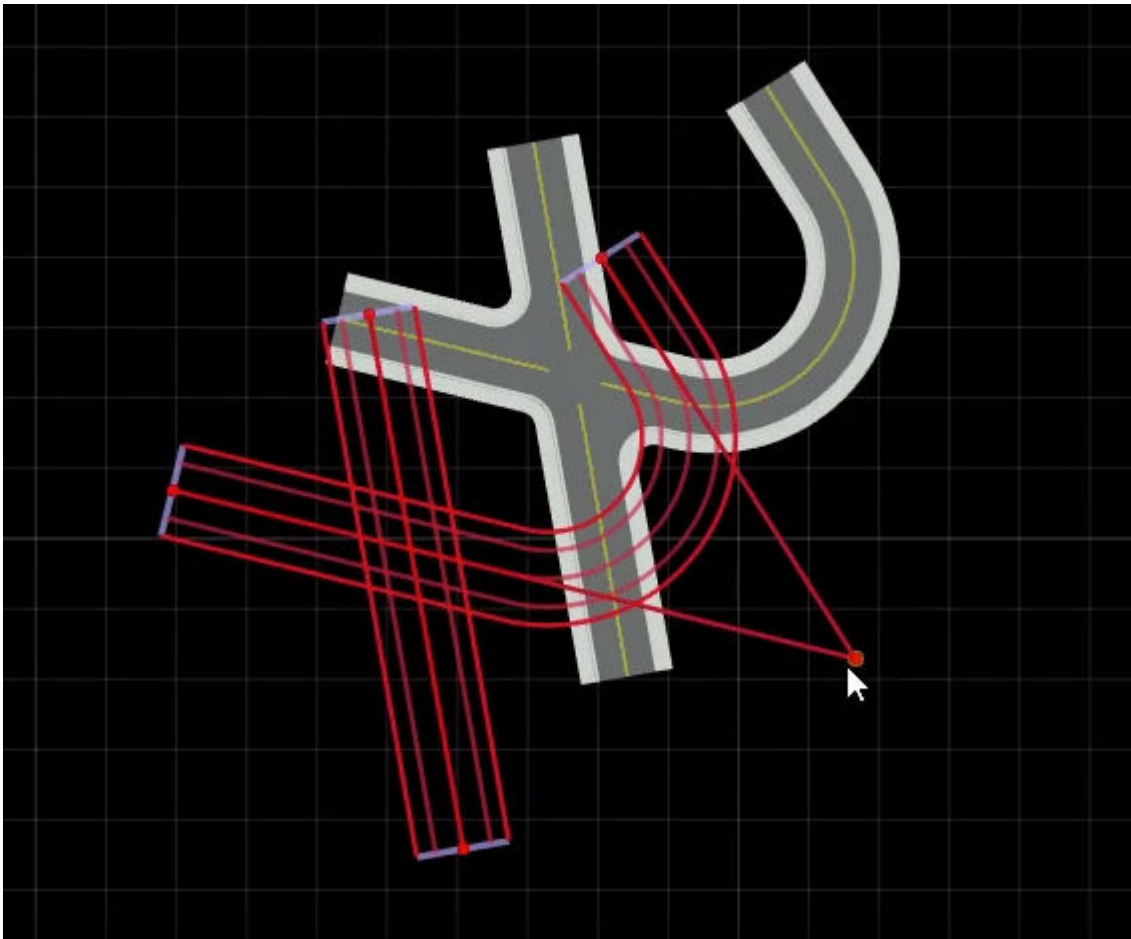
- 2** Click and drag from the left side of the canvas to select multiple roads you want to move. The roads are highlighted, and the control points are displayed and connected by light blue control lines.



- 3** Click and drag from the right side of the canvas to select the same roads again. The control lines and control points turn red.



- 4 Click on a control point and drag it to a new location. While dragging the control point, the canvas displays a preview of the shifted roads as a red outline. Note that the road meshes do not move until you release the control point.

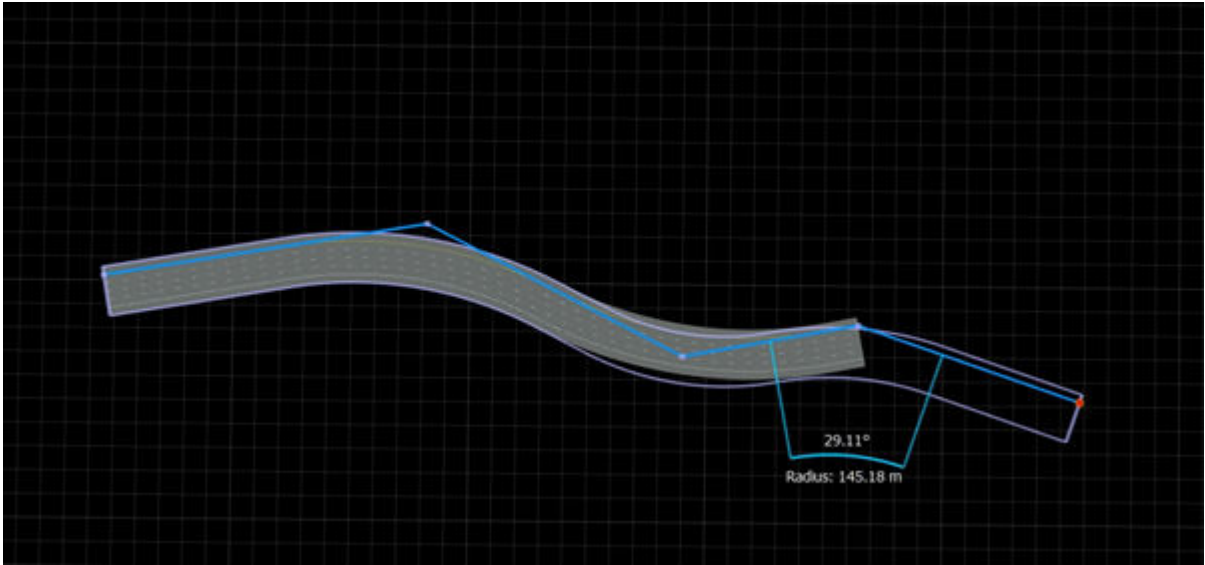


- 5 When you are satisfied with the new location of the roads indicated by the preview, release the control point. The road meshes shift from the source location to the destination location.

Extend Existing Road

You can extend an existing road in either direction by adding more control points, as follows:

- 1 Click the **Road Plan Tool** button.
- 2 Click the road you want to extend. The road is highlighted, and the control points are displayed and connected by light blue lines.
- 3 Click the control point on the end of the road you want to extend.
- 4 Right-click to create a new control point and extend the road.



Note For optimal performance, avoid very long individual roads. Keeping individual roads under 500 m is recommended. To create stretches of road longer than 500 m, use multiple roads connected end-to-end. Refer to “Create New Road Connected End-to-End with Another Road” on page 1-143 and “Connect Two Roads End-to-End” on page 1-144.

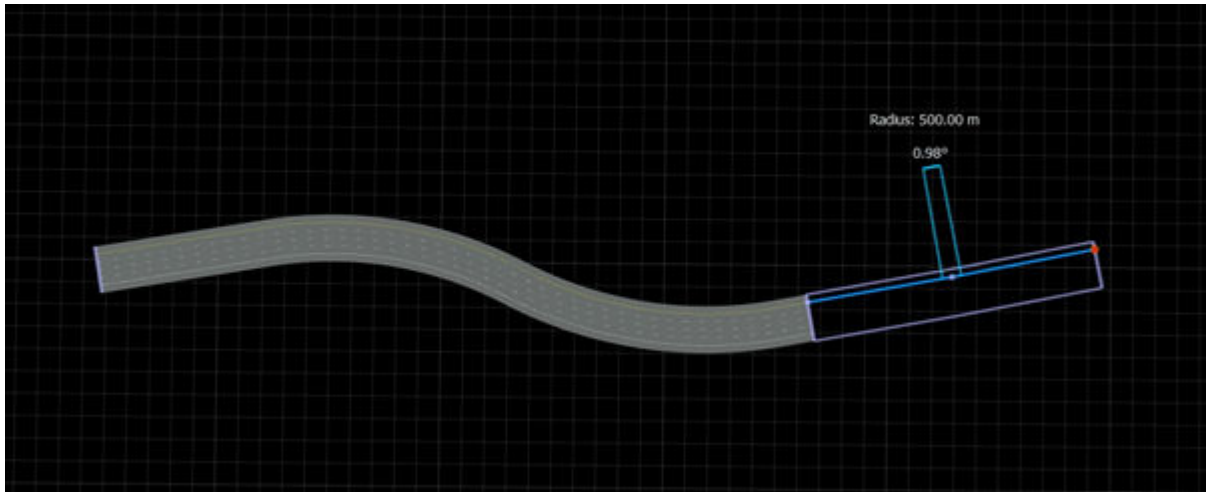
Create New Road Connected End-to-End with Another Road

In a similar fashion to extending an existing road, you can also create a new road that connects end-to-end with an existing road. The visual result is similar to extending the existing road, but there are some important situations where end-to-end roads are needed:

- To avoid extremely long roads for performance reasons
- To create a road loop or self-intersecting road

You can create an end-to-end road that connects with an existing road as follows:

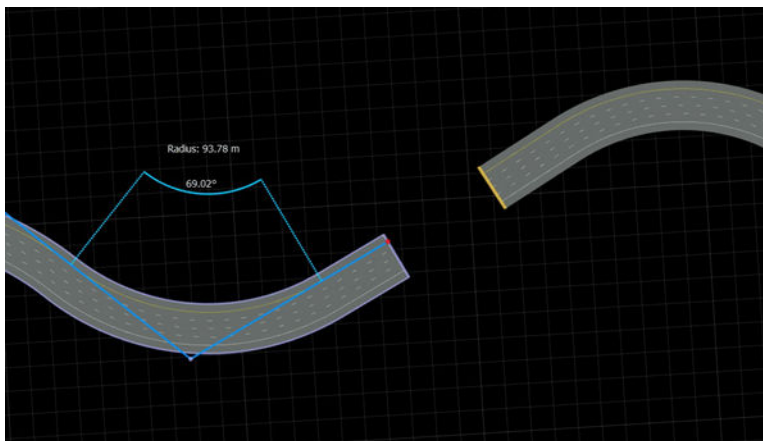
- 1** Click the **Road Plan Tool** button.
- 2** If another road is already selected, click away from any road to unselect it.
- 3** Click the lavender road node line at the end of a road.
- 4** Right-click to create a new control point, which creates a new road that extends off the existing one.



Connect Two Roads End-to-End

Similar to the steps above, you can extend a road and simultaneously connect it to the end of an existing road as follows:

- 1 Click the **Road Plan Tool**.
- 2 Click the road you want to extend. The road is highlighted, and the control points are displayed and connected by light blue lines.



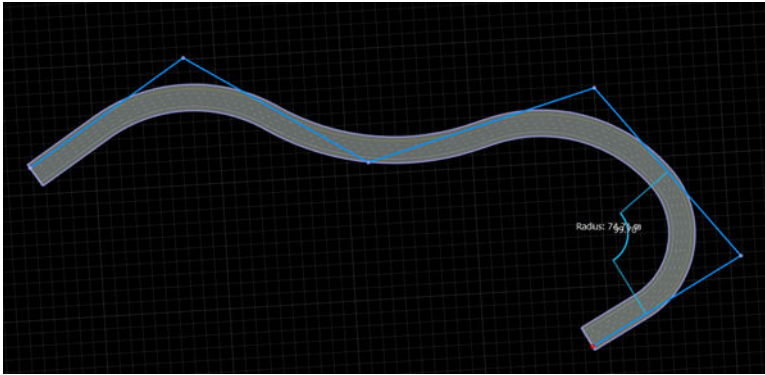
- 3 Click the control point on the end of the road you want to extend.
- 4 Right-click the lavender line at the end of another road.

Create Road Loop

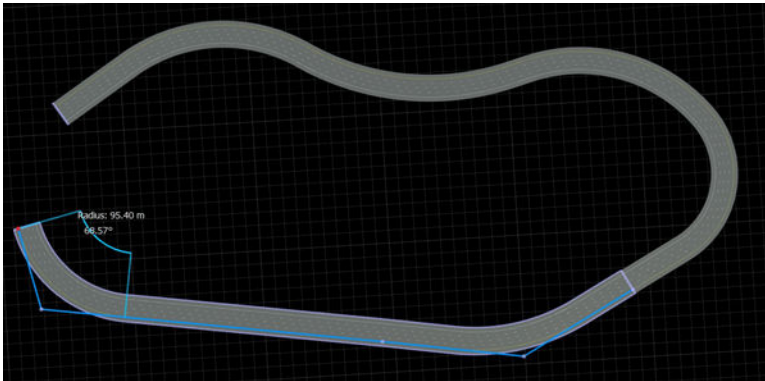
You can create a closed loop road by combining the steps above or by using the **Road Circle Tool**.

Note Closed loops require at least three separate roads. You cannot form a loop from a single road.

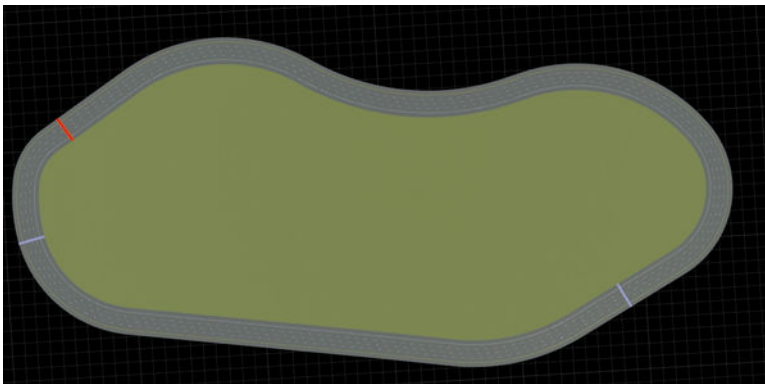
- 1 Create the first road by using the “Create a New Road” on page 1-136 steps.



- 2 Create the second road by using the “Create New Road Connected End-to-End with Another Road” on page 1-143 steps.



- 3 Create the final road by using the “Create New Road Connected End-to-End with Another Road” on page 1-143 steps, followed by the “Connect Two Roads End-to-End” on page 1-144 steps to end the road.



Delete Road Control Point

- 1 Click the **Road Plan Tool** button.
- 2 Click the road you want to delete the point from. The road is highlighted, and the control points are displayed and connected by light blue lines.
- 3 Click the control point you want to delete.
- 4 Press the **Delete** key, or select **Edit > Delete** from the menu bar.

Delete Road

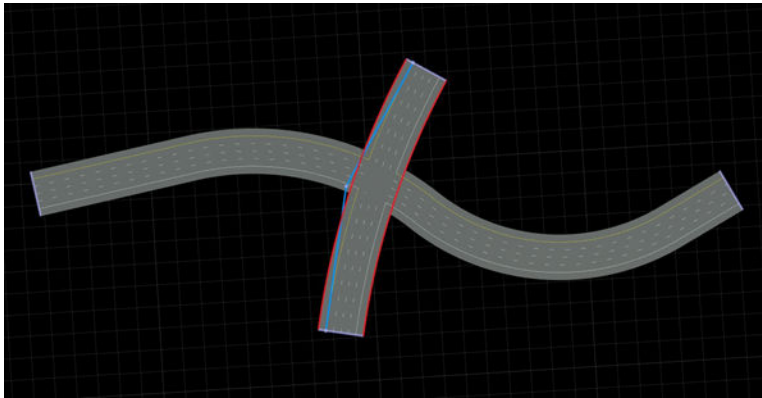
- 1 Click the **Road Plan Tool** button.
- 2 Click the road you want to delete.
- 3 Press the **Delete** key, or select **Edit > Delete** from the menu bar.

Create Intersection

At-grade intersections are created automatically in RoadRunner wherever two or more roads cross.

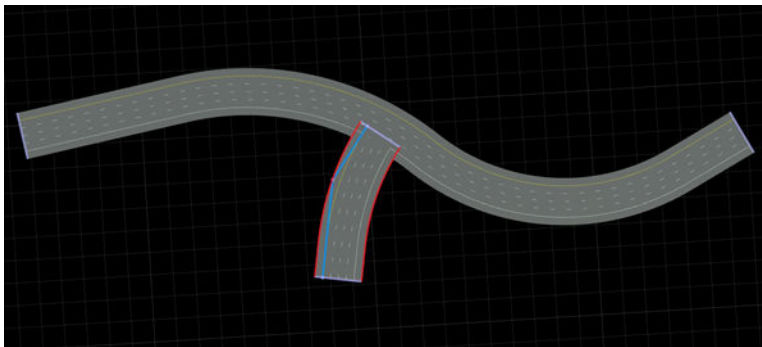
Four-Way Intersections

To create a four-way intersection, create two roads that fully overlap:

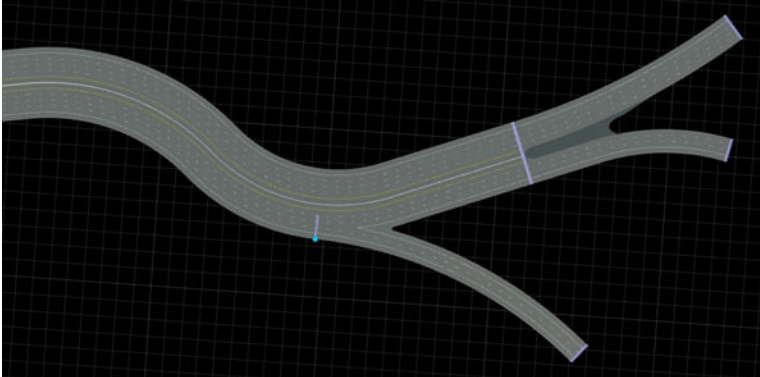


T-Junctions

To create a T-junction, create two roads where one ends within the extents of the other:



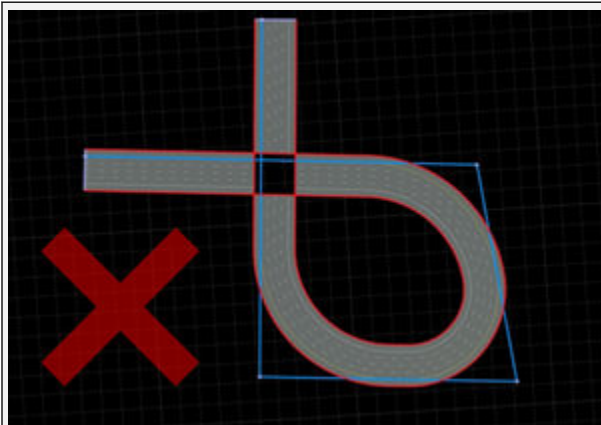
Ramps and Splits



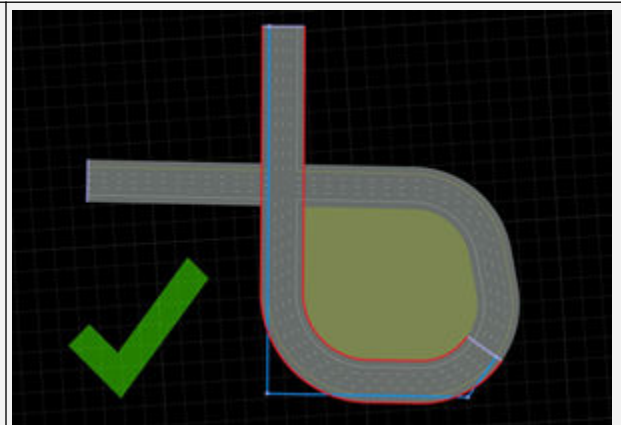
To create onramps, offramps, and road splits, refer to the **Slip Road Tool** documentation.

Self-Intersections

A single road should not overlap itself. If you need to create a road that loops back on itself, either chop the road with the **Road Chop Tool**, or create roads connected end-to-end:



Roads may not cross themselves

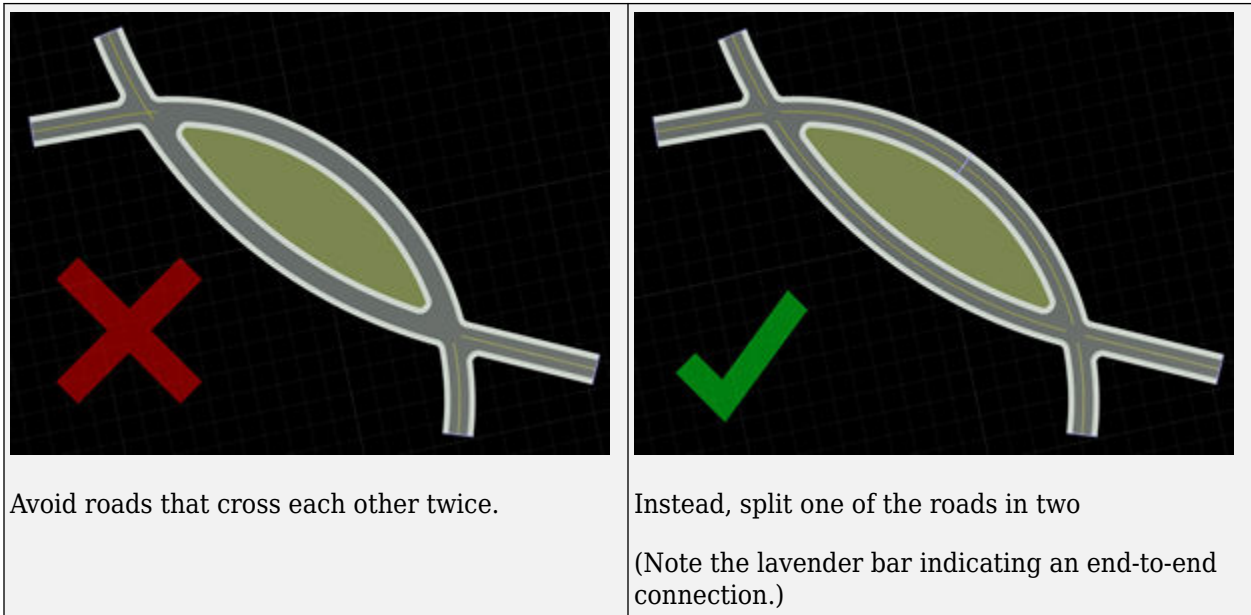


Instead, create two or more roads connected end-to-end

Double-Crossings

You might observe issues with lane markings when two roads cross each other twice (that is, two at-grade intersections are formed between the same two roads).

Avoid double-crossing roads. If you need to create a double-crossing situation, either chop one of the roads with the **Road Chop Tool**, or initially create one of the roads using two end-to-end roads.



Note Intersections are only created when the roads have similar heights at the crossing locations. To adjust road heights, use the **Road Height Tool**.

Prevent Creation of Automatic Junctions Between Roads

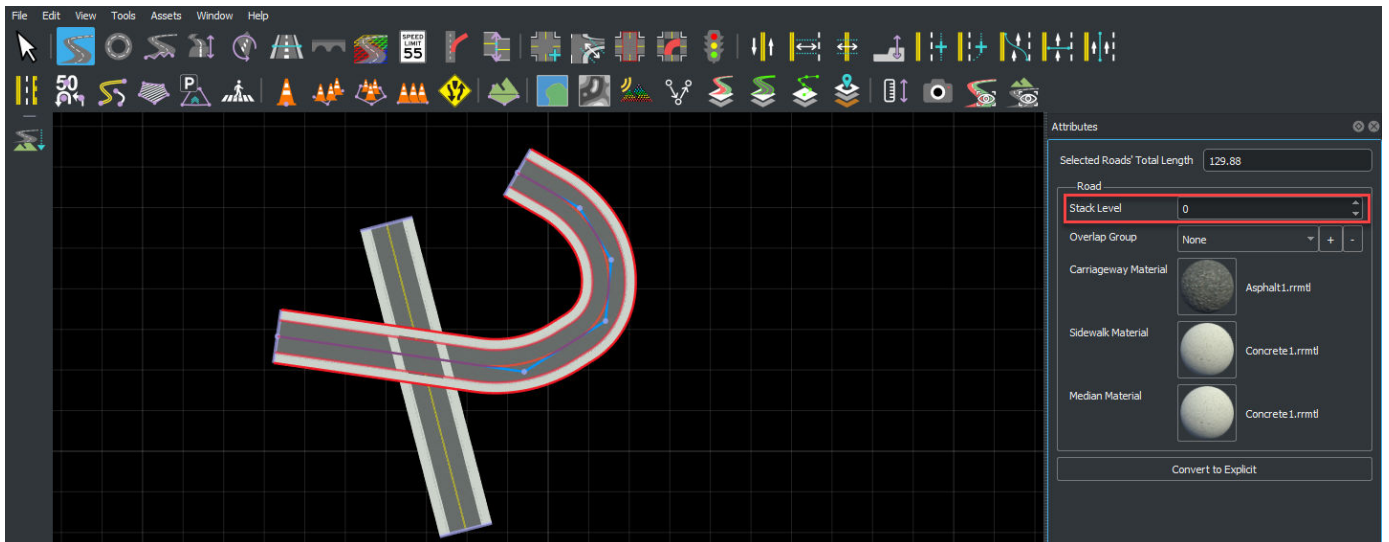
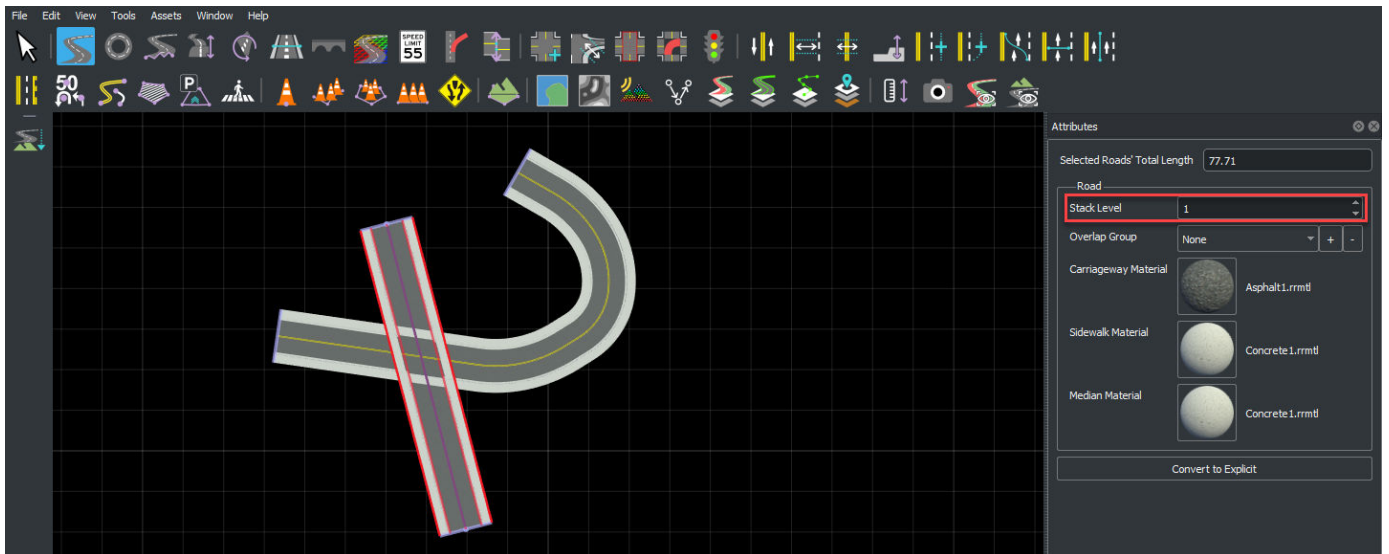
RoadRunner automatically creates intersections wherever two or more roads cross. To prevent RoadRunner from creating automatic junctions between roads, use the **Stack Level** or **Overlap Group** attribute.

Stack Level

Two roads only produce an automatic junction if their **Stack Level** values are the same. For example, when creating an overpass in a highway interchange, you can define the levels of the overpass by assigning each level a different **Stack Level** value. If you also assign a road an **Overlap Group** value, then the road only creates automatic junctions with overlapping roads with the same **Stack Level** value and different **Overlap Group** values.

To modify the **Stack Level** value for a road, follow these steps:

- 1 Click the **Road Plan Tool** button.
- 2 Click the road for which you want to modify the **Stack Level** value.
- 3 In the **Attributes** pane, under **Road**, increase or decrease the **Stack Level** value as desired to suit your scene. For example, in the figure, a straight road and a curved road overlap. However, because the **Stack Level** value for the straight road is set to 1 and the **Stack Level** value for the curved road is set to 0, these two roads do not automatically produce a junction.



Overlap Group

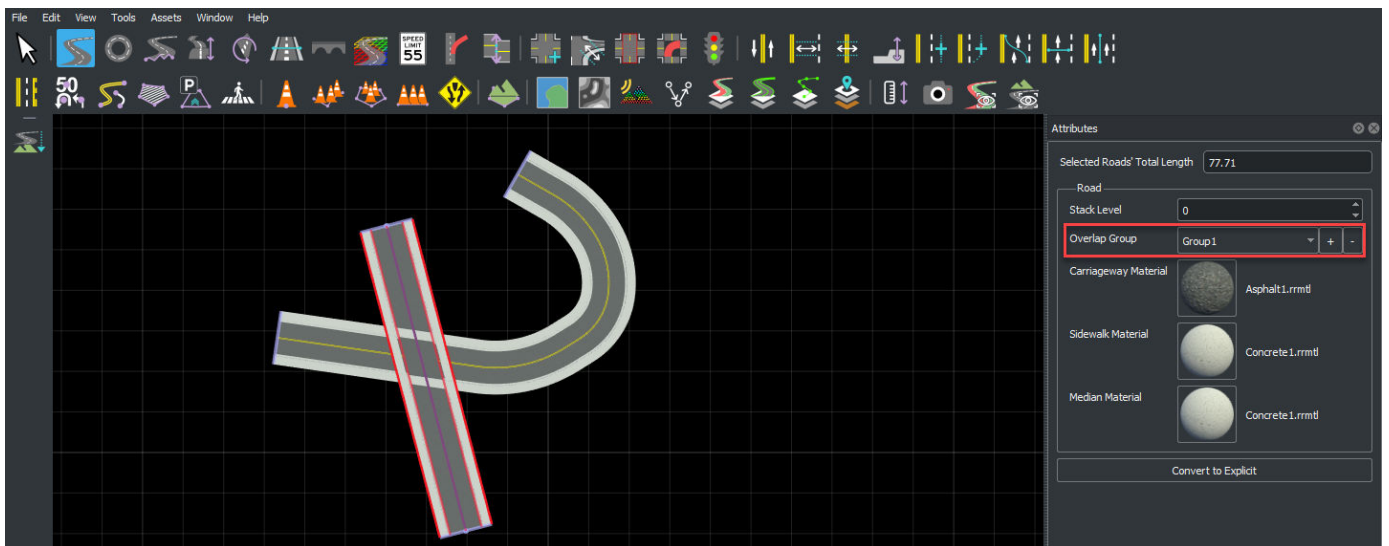
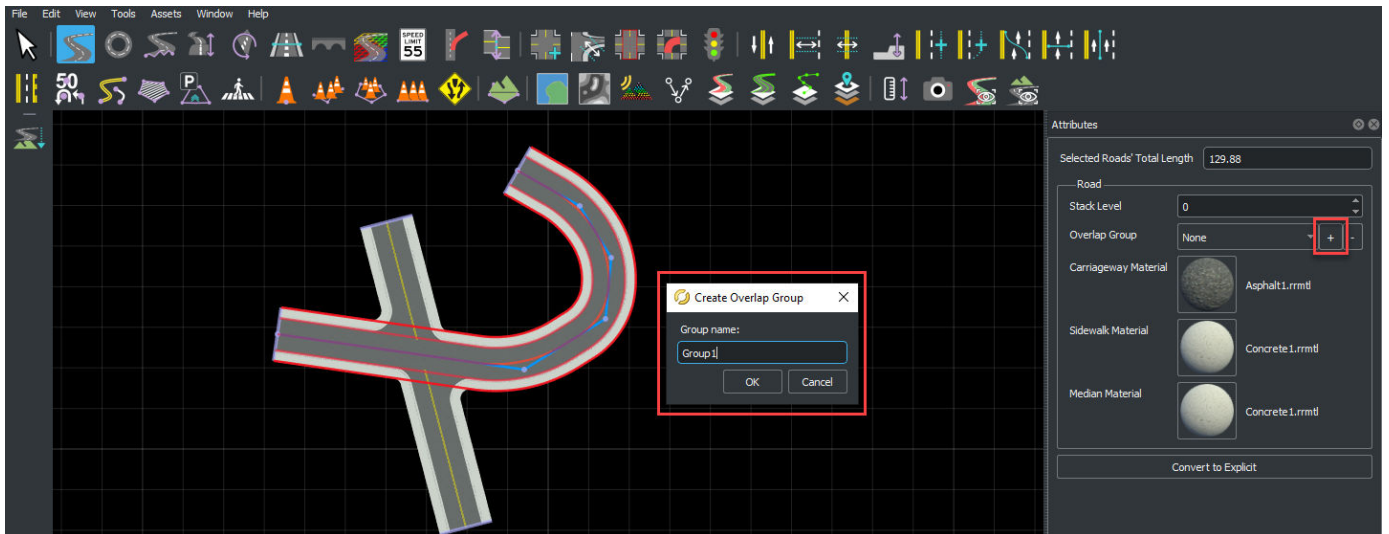
Roads with the same **Overlap Group** value do not produce automatic junctions, even if they have the same **Stack Level** value. You can use the **Overlap Group** attribute when importing data where the junctions have already been defined (such as from ASAM OpenDRIVE) to ensure that RoadRunner does not produce additional automatic junctions.

To modify the **Overlap Group** values for roads, follow these steps:

- 1 Click the **Road Plan Tool** button.
- 2 Click the road for which you want to assign an **Overlap Group** value.
- 3 In the **Attributes** pane, under **Road**, click the **+** button next to **Overlap Group**. This opens the Create Overlap Group dialog box. Enter the name of a group to assign the road to that overlap group.

Repeat this step to assign the remaining roads to overlap groups. The roads assigned to the same overlap group do not produce an automatic junction. For example, in this figure, a curved road

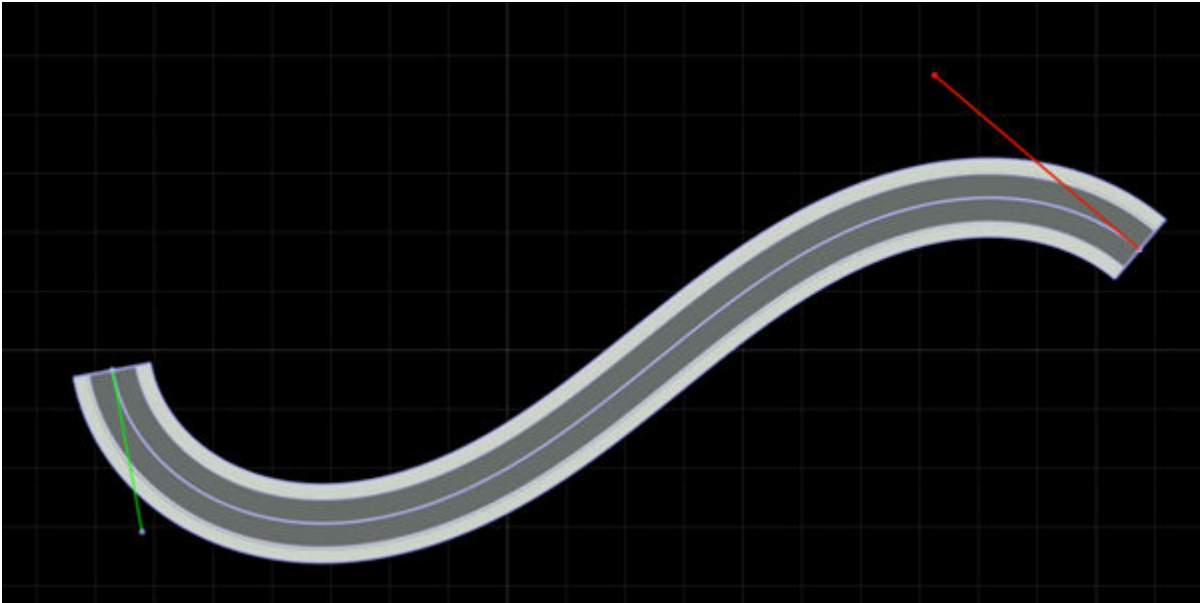
and a straight road overlap. Because the curved road and the straight road both have an **Overlap Group Group1**, these two roads do not produce a junction.



Explicit Road Curves

By default, new roads will be created out of straight lines and circular arcs. Roads created in this method are called "Automatic". It is sometimes desirable to instead define a road curve as an explicit set of straight lines, circular arcs, clothoids (spirals), and parametric cubics (Hermite curves).

Roads created in this method are called "Explicit." Each straight line, circular arc, clothoid (spiral), or parametric cubic (Hermite curve) is called a "Segment." The explicit road curve also allows you to set the tangents of the road at each control point. Editing the control points of an explicit curve is done the same as with an automatic curve.



Building Roads With Explicit Curves

Explicit curves can be used to create a road with a very specific profile (for example, a 50 m linear section, followed by a 20 m spiral with specific starting and ending curvatures, followed by a 30 m arc with specific curvature, and so on).

To build such a road, follow these steps:

- 1 Click the **Road Plan Tool** button.
- 2 Create a new road using these steps: “Create a New Road” on page 1-136.
- 3 Convert the new road to explicit form using these steps: “Make Road Curve Explicit” on page 1-151.
- 4 Adjust the type and properties of the first road section.
- 5 For each new section you want to add to the road:
 - 1 Add a new section using these steps: “Extend Existing Road” on page 1-142.
 - 2 Adjust the type and properties of the new road section.

Make Road Curve Explicit

- 1 Click the **Road Plan Tool** button.
- 2 Click the road you want to change.
- 3 In the **Attributes** pane, press **Convert to Explicit**.

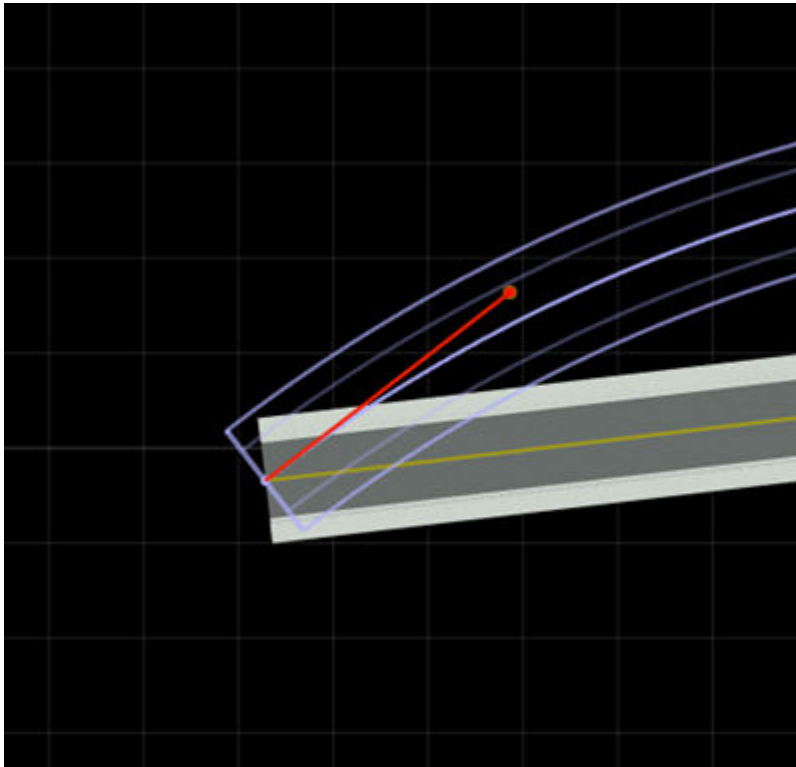
Make Road Curve Automatic

- 1 Click the **Road Plan Tool** button.
- 2 Click the road you want to change.
- 3 In the **Attributes** pane, press **Convert to Automatic**.

Note Converting an explicit curve to an automatic curve can slightly change the curve and insert additional points.

Change Tangent of Explicit Curve

- 1 Click the **Road Plan Tool** button.
- 2 Click the road you want to change.
- 3 Click and drag one of the tangent control points and move it to set the desired tangent.



Note Setting tangents on a road may change the type of the segments connected to the affected control point.

Change Type of Segment

- 1 Click the **Road Plan Tool** button.
- 2 Click the road you want to change.
- 3 Click the segment you want to change.
- 4 In the **Attributes** pane, select the **Type** of the segment. This will automatically constrain the segment's points and tangents to match the type.

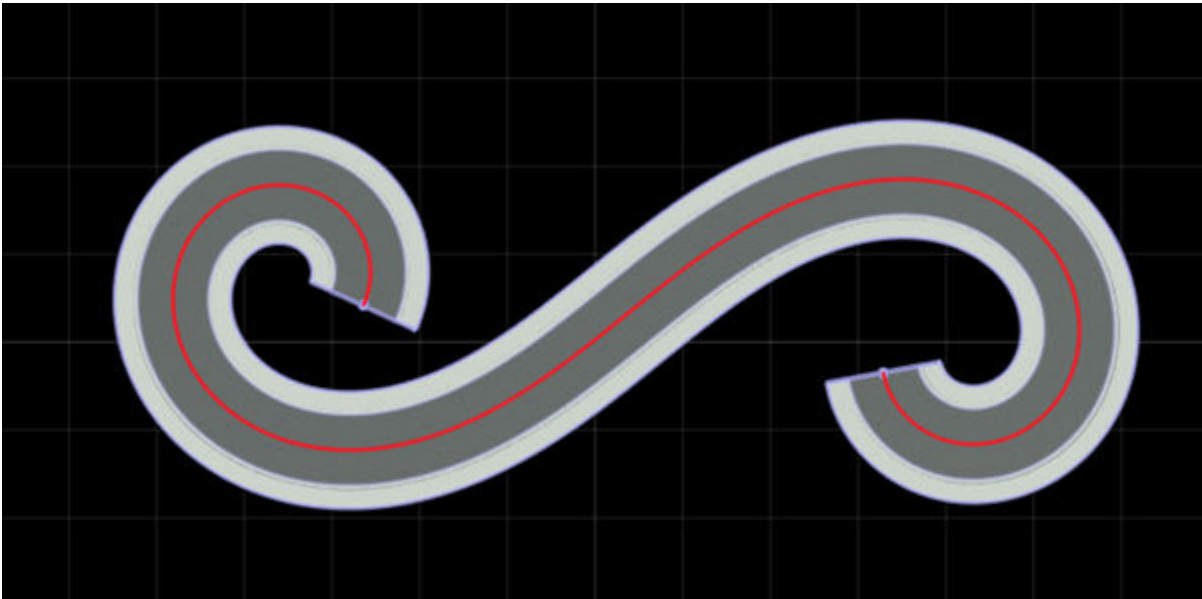
Change Length of Segment

- 1 Click the **Road Plan Tool** button.
- 2 Click the road you want to change.
- 3 Click the segment you want to change.
- 4 In the **Attributes** pane, adjust the **Length** of the segment to the desired length.

Note You can set the length of a segment only if the segment is a line, arc, or spiral.

Change Curvature of Segment

- 1 Click the **Road Plan Tool** button.
- 2 Click the road you want to change.
- 3 Click the segment you want to change.
- 4 In the **Attributes** pane, adjust the **Curvature** (for circular arcs) or **Start Curvature/End Curvature** (for spirals).



Note You can set the curvature of a segment only if the segment is an arc or spiral.

Version History

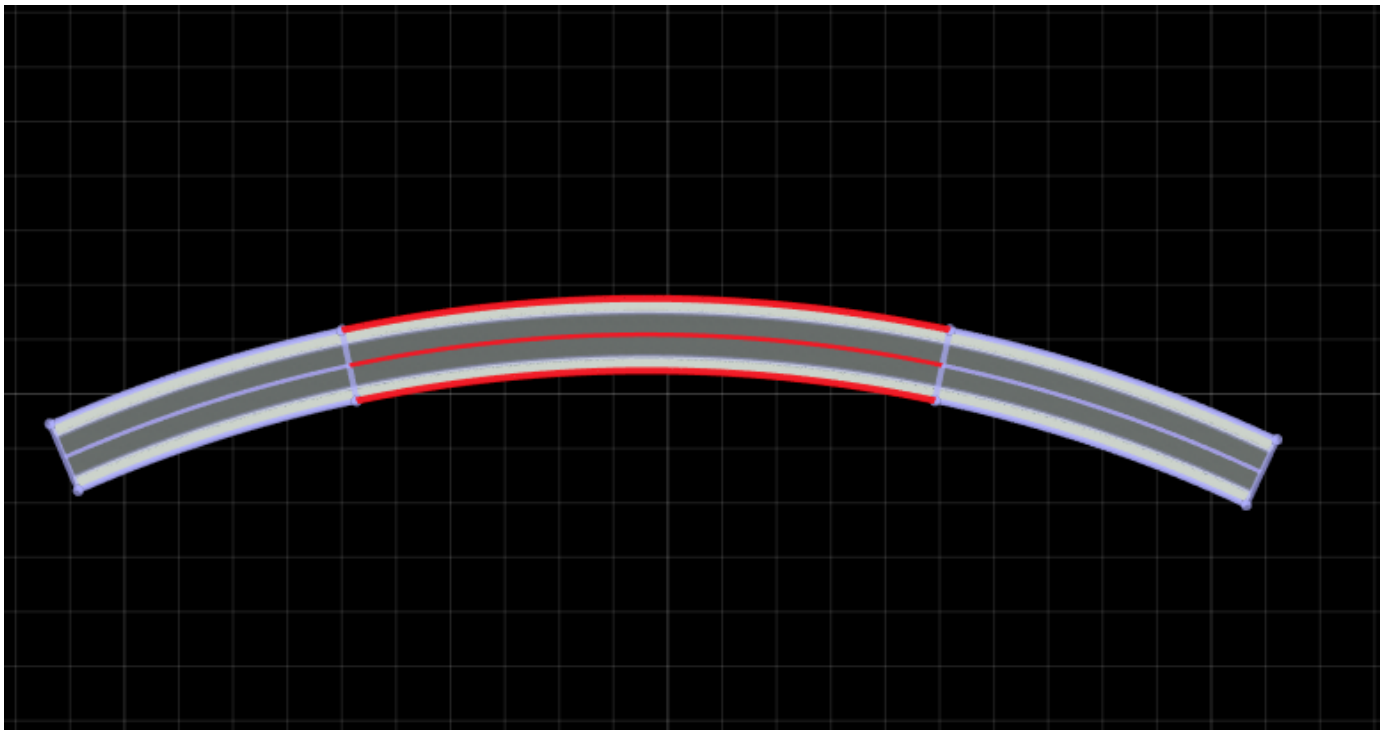
Introduced in R2020a

Road Speed Limits Tool

Set speed limits along road spans

Description

The **Road Speed Limits Tool** enables you to set varying speed limits along sections of a road. When you export your scene to a simulator, you can use these values to test whether vehicles drive the set speed limits.



Open the Road Speed Limits Tool

On the RoadRunner toolbar, click the **Road Speed Limits Tool** button:

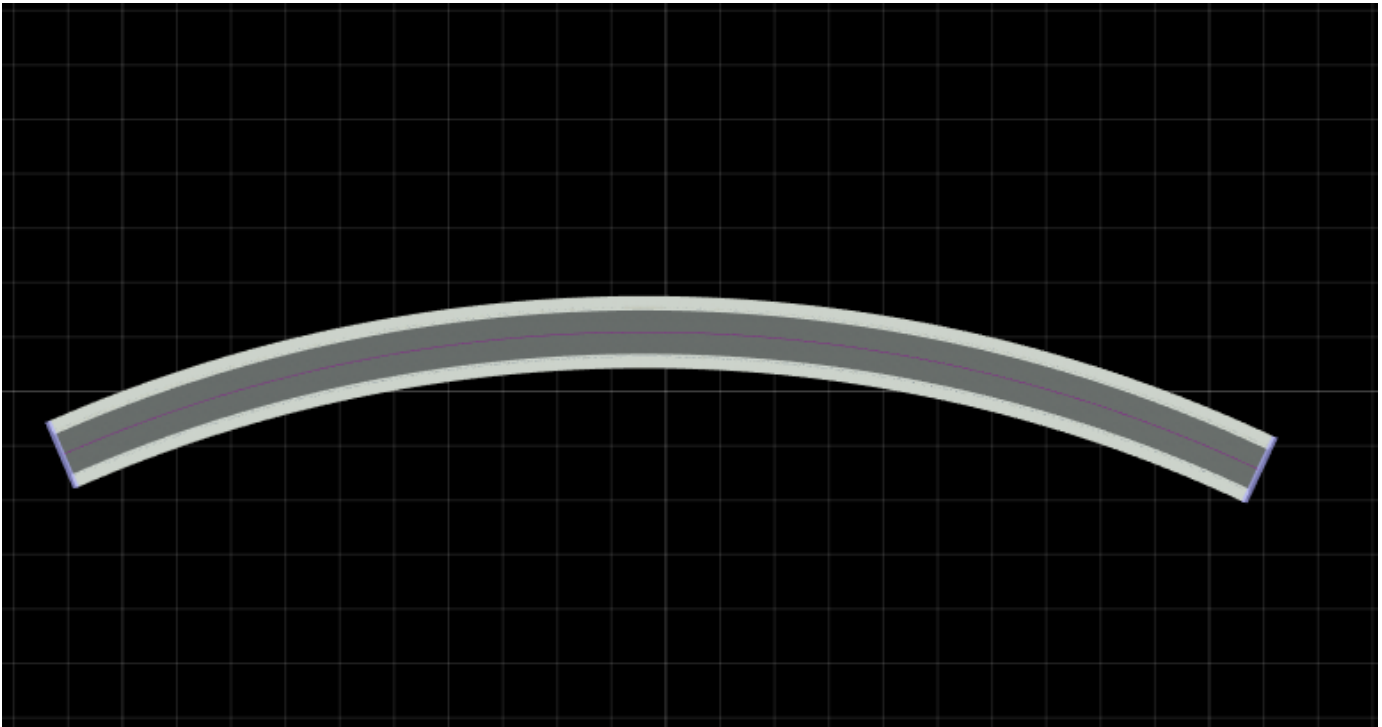


Examples

Set Speed Limits Along Road

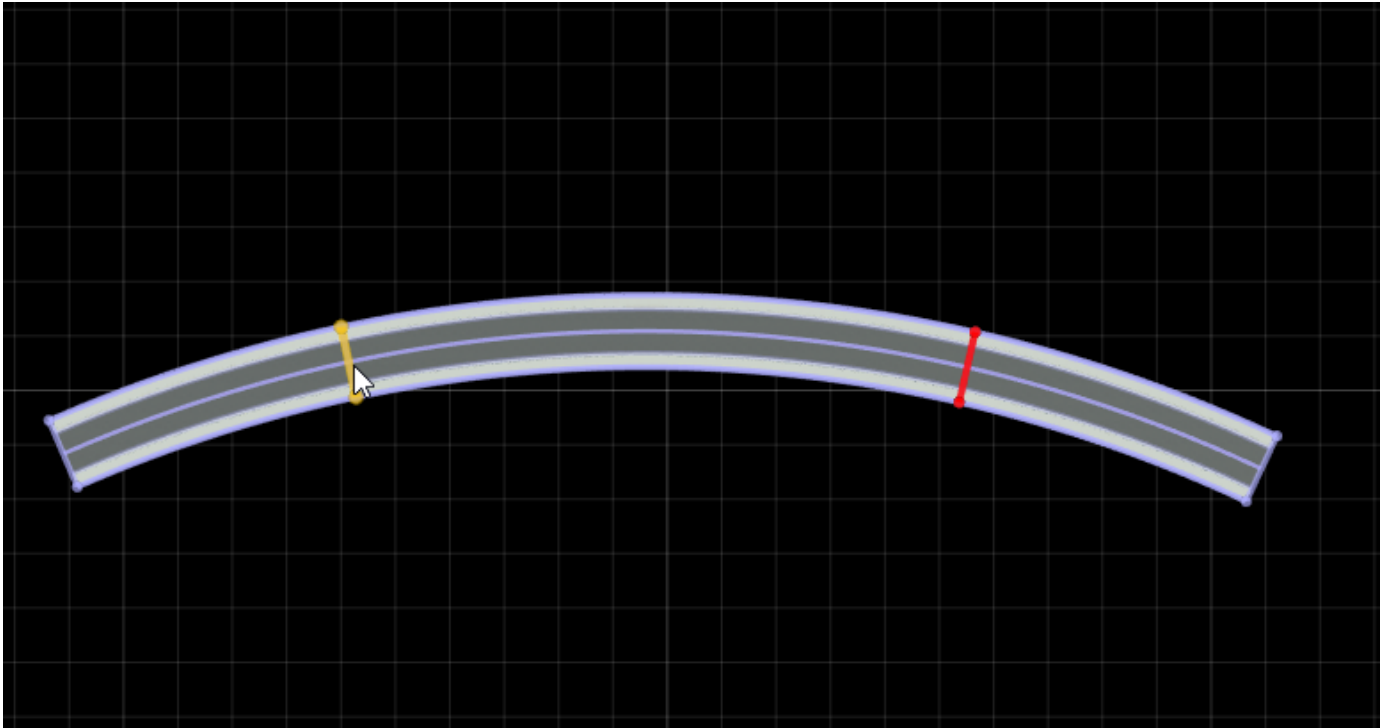
This example shows how to set varying speed limits along a simple curved road.

Create a curved road segment by using the **Road Plan Tool**.



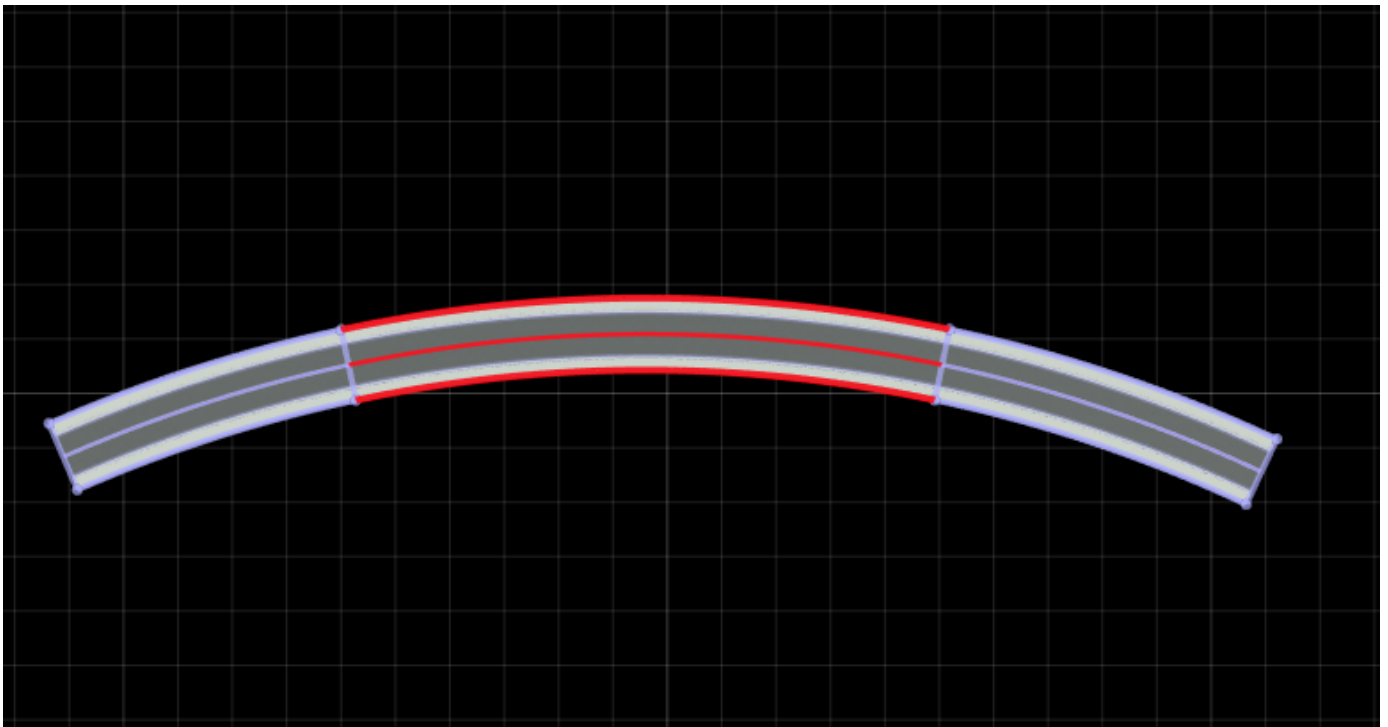
Divide the road into sections along which to set different speed limits.

- 1** Select the **Road Speed Limits Tool** button.
- 2** Select the road by clicking it.
- 3** Right-click the road at two locations to divide it into three sections. To adjust the sections, click and drag the dividing nodes along the curve of the road.



By default, each section of the road has a speed limit of 40 miles per hour. Set the middle section of the road to a lower speed limit.

- 1 Click the middle section of the road to select it.



- 2 In the **Attributes** pane on the right, set the **Speed Limit** to 20.

You can then adjust the sections further or add or delete sections. To delete a section, click a dividing node to select it, and then click **Delete**. When you delete a dividing node, the merged section inherits the speed limit of the road section that had the higher speed limit.

Parameters

Attribute	Description
Speed Limit	Speed limit of road section, specified as a nonnegative integer. By default, units are in miles per hour. To change units to kilometers per hour, in the menu bar, select Tools > Scene Settings . Then, set Speed Units to KPH. Default: 40
Road Type	Type of road, specified as Unknown, Rural, Motorway, Town, Low Speed, Pedestrian, or Bicycle. Default: Town

Version History

Introduced in R2020b

See Also

Road Plan Tool

Scene Builder Tool

Generate 3D scenes from HD Map data

Description

To import, inspect, and build scenes from HD Map data, use the **Scene Builder Tool**. After building the scene, you can edit and export the scene by using other RoadRunner capabilities.

Currently, the RoadRunner supports a pair of HD Map services:

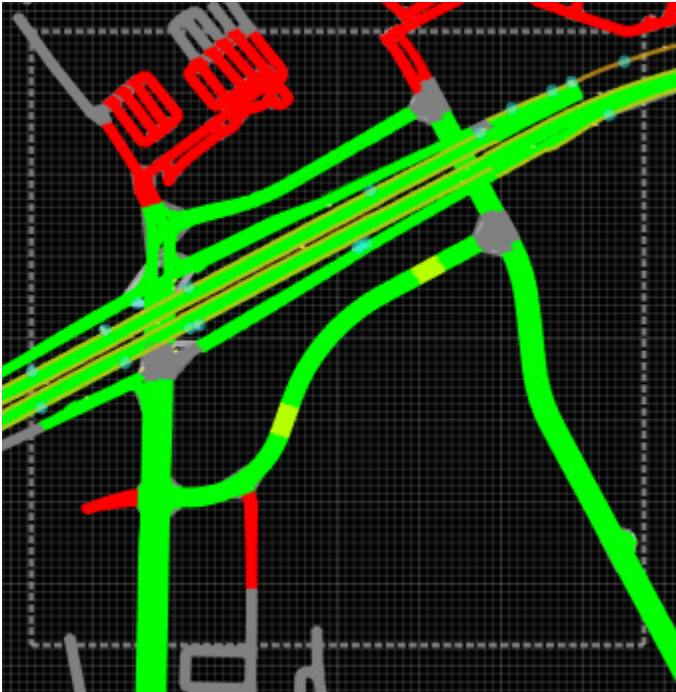
- HERE HD Live Map ¹ — Developed by HERE Technologies, this cloud-based web service that enables you to access highly accurate, continuously updated map data. The data is composed of tiled map layers containing information such as the topology and geometry of roads and lanes and scene objects such as signs, poles, and barriers.
- TomTom HD Map — Developed by TomTom Technologies, this delivers a highly accurate, up-to-date, and realistic representations of roads to suit the purpose of automated driving. RoadRunner Scene Builder supports the TomTom HD Map data formats Avro and GeoPackage. Each format contains information such as lane geometry, lane types, road borders, traffic signs, curbs, and barriers. The supported versions for TomTom HD Map Avro data formats are specified in the table.

Data Format Type	Version
HD Map Avro Format Specification	1.2.0.0
HD Map Avro Schema	1.1

- Apollo — Developed by Baidu Apollo[®], this is an open platform that enables you to access highly accurate HD maps for automated driving applications. RoadRunner Scene Builder supports the Apollo 3.0 and 5.0 XML formats. Each format contains information such as lane geometry, lane types, road boundary and road intersection geometry.

The **Scene Builder Tool** is part of RoadRunner Scene Builder, an add-on product that requires an addition to your RoadRunner license. For more details, see “Get RoadRunner Updates and Upgrades”. You also need to enter into a separate agreement with HERE Technologies to gain access to HERE HD Live Map data and to get the required Marketplace credentials, **Access Key ID** and **Access Key Secret**.

¹ You need to enter into a separate agreement with HERE in order to gain access to the HDLM services and to get the required credentials (access_key_id and access_key_secret) for using the HERE Service.



Open the Scene Builder Tool

On the RoadRunner toolbar, click the **Scene Builder Tool** button:



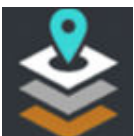
Examples

View HERE HD Live Map Scene Data for Visual Reference

View HERE HD Live Map data in RoadRunner for an area in Santa Clara, California. You can use this data as a visual reference to manually create roads on or around it.

Choose Area of Interest

Open the **World Settings Tool** from the toolbar by clicking the **World Settings Tool** button. Specify the area of interest by using the **World Settings Tool**.



In the **Attributes** pane, in the **World Origin** section, specify the **Latitude** as 37.4156 degrees and the **Longitude** as -121.9749 degrees. In the **Workspace** section, under **Extents**, specify both **X** and **Y** as 750 meters. Apply your changes by selecting **Apply World Changes**.

Import and Explore Data

Open the **Scene Builder Tool** from the toolbar by clicking the **Scene Builder Tool** button.



Then, import the HERE HD Live Map data by clicking the **Import Data For Area** button on the toolbar to the left of the scene editing canvas.



To import data from a directory containing HERE protobuf files, select **Open HD Map Root Folder** from the toolbar to the left of the scene editing canvas.



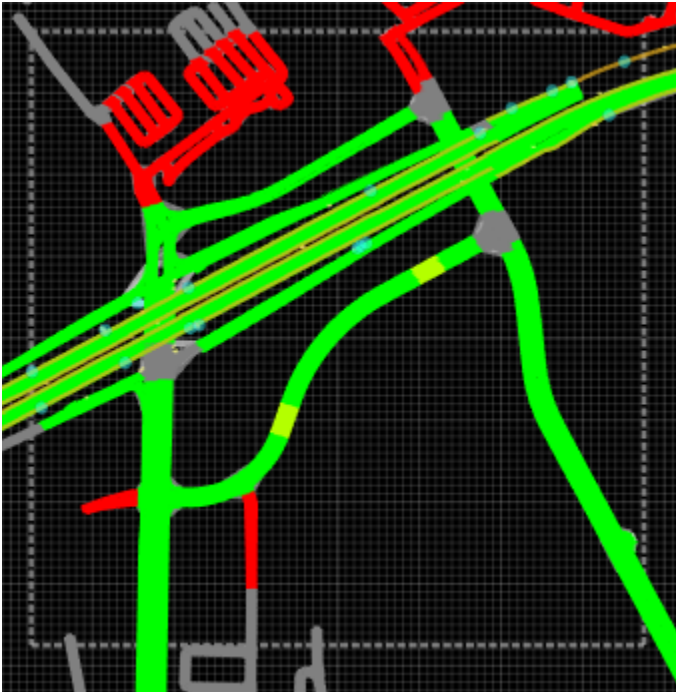
This opens an Import HD Data folder dialog box to specify the type and source of the HD Map data. Set the **Source Type** parameter to **HERE**, and set the **Protobuf Folder's Root Folder** by navigating to the directory that contains your HERE protobuf files. By default, RoadRunner Scene Builder unpacks the **.gz** file containing HERE protobuf files and imports the data. To skip this step, clear the **Uncompress using gzip** parameter.

RoadRunner saves the specified Import HD Data folder dialog box parameters for future RoadRunner sessions on your machine.

Before you import the Live Map data, you must enter a valid HERE HD Live Map access credentials, **Access Key ID** and **Access Key Secret**, then click **OK**. RoadRunner saves these credentials for the rest of this RoadRunner session on your machine. To save these credentials for future RoadRunner sessions on your machine, select **Save my credentials**. The credentials remain saved until you delete them.

You can delete the existing credentials by erasing them from the HERE HD Live Map Marketplace Credentials dialog box and clicking **OK**. RoadRunner displays an error message indicating that your credentials are missing, or that your connection to the download server is broken. Ignore the error message, and select **Cancel** to close the dialog box.

The **Scene Builder Tool** imports HERE HD Live Map data from HERE tiles that intersect your workspace, converts the data into a preview called a **RoadRunner HD Map**, and displays the **RoadRunner HD Map** in the scene editing canvas. For more information on **RoadRunner HD Map** data see "RoadRunner HD Map Data" on page 1-166.



Explore the imported data by selecting control points, lane boundaries, lanes, lane groups, and scene objects. You can view their properties on the **Attributes** pane. The type of road element selected in the HD Map scene editing canvas determines the available properties.

Scene Builder

- **Color by Confidence Classification** — By default, the **RoadRunner HD Map** displays the confidence classification of the imported data using various colors. Refer to the **Classification Legend** in the **Attributes** pane for specific information on these colors. For more information about classification, contact HERE Technologies.

To disable both the **Classification Legend** in the **Attributes** pane and the color classification in the **RoadRunner HD Map** preview, clear this option.

- **Selected Lane Length** — Length of the selected lanes.
- **Scene Lane Length** — Length of the total lanes in the scene.

Lane

- **Id** — Unique identification number for the selected lane.
- **Speed Limit** — Maximum allowed driving speed of the lane.
- **Lane Type** — Type of the lane, specified as driving or shoulder.
- **Travel Direction** — Direction of travel for the road segment, specified as forward, backward, or bidirectional.
- **Start Lanes** — Displays the list of lanes, and their IDs, connected to the start of the selected lane.
- **End Lanes** — Displays the list of lanes, and their IDs, connected to the end of the selected lane.

You can view unique IDs for the **Lane Boundary** and **Lane Group** on the **Attributes** pane.

Barrier

- **Id** — Unique identification number for the selected barrier.
- **Side** — Lane side on which the barrier is placed, specified as left or right.
- **Extrusion Asset** — Displays the extrusion asset image available in the asset library which is relevant to the imported scene object.

You can also view the unique IDs and **Prop Asset** images available in the asset library which are relevant to the imported **Pole** and **Sign** scene objects.

You can manually create on or around this data. Alternatively, if you have a RoadRunner Scene Builder license, click **Build Scene** to build roads and scene objects directly from the data.

View TomTom HD Map Scene Data for Visual Reference

View TomTom HD Map data in RoadRunner for an area in Michigan, USA. You can use this data as a visual reference to manually create roads on or around it.

Choose Area of Interest

You can specify an area of interest by using the **World Settings Tool** and specifying its world origin latitude, world origin longitude, and the X and Y workspace extents. For more information see “Build Scenes Using TomTom HD Map Data”.



For this example, import the whole data with out specifying the area of interest.

Import and Explore Data

Open the **Scene Builder Tool** from the toolbar by clicking the **Scene Builder Tool** button.



Then, import the TomTom HD Map data by clicking the **Open HD Map Root Folder** button on the toolbar to the left of the scene editing canvas.



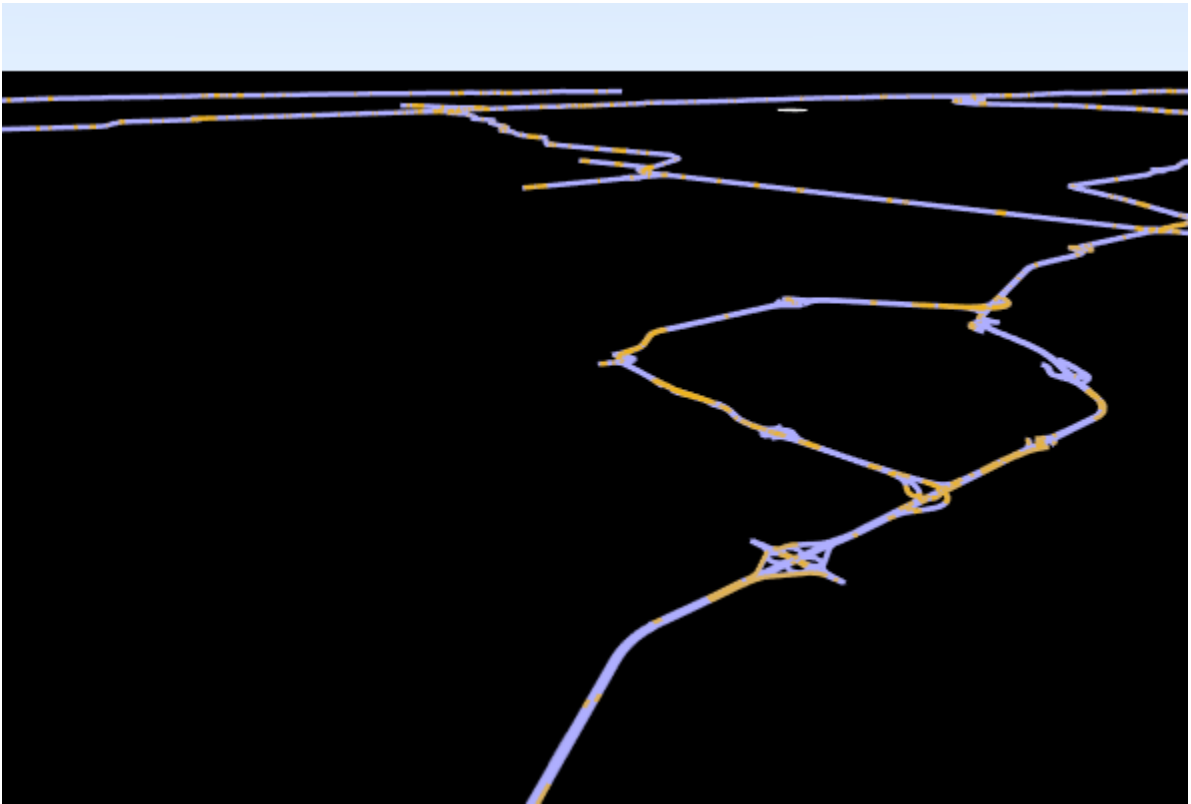
This opens an Import HD Data folder dialog box to specify the type and source of the HD Map data. Set the **Source Type** parameter to **TomTom**. RoadRunner Scene Builder supports the TomTom HD Map data formats **Avro** and **GeoPackage**.

Set the **Choose data format** parameter to **Avro** or **GeoPackage**, and specify the path of the **TomTom Root Folder** by navigating to the directory containing your TomTom HD Map data. Each format has several files that contain data such as the lane border, lane center line, lane group, lane trajectory, junction area, speed restrictions, and traffic signs. For access to TomTom HD Map data, contact TomTom Technologies.

RoadRunner saves the specified Import HD Data folder dialog box parameters for future RoadRunner sessions on your machine.

For this example, set **Choose data format** to **Avro**, and specify the path of the **TomTom Root Folder** by navigating to a folder containing Avro HD Map Data, and import the data by selecting **Import**.

The **Scene Builder Tool** imports the Avro HD Map data from the specified folder, converts the data into a preview called a **RoadRunner HD Map**, and displays the **RoadRunner HD Map** in the scene editing canvas. For more information on **RoadRunner HD Map** data, see “RoadRunner HD Map Data” on page 1-166.



Explore the imported data by selecting lanes, lane boundaries, barriers, and traffic signs. You can view their properties on the **Attributes** pane. The type of road element selected in the **RoadRunner HD Map** scene editing canvas determines the available properties.

Scene Builder

- **Color by Confidence Classification** — Disabled for imported TomTom HD data.
- **Selected Lane Length** — Length of the selected lanes.
- **Scene Lane Length** — Length of the total lanes in the scene.

Lane

- **Id** — Unique identification number for the selected lane.
- **Speed Limit** — Maximum allowed driving speed of the lane.

- **Lane Type** — Type of the lane.
- **Travel Direction** — Direction of travel for the road segment, specified as forward, backward, or bidirectional.
- **Start Lanes** — Displays the list of lanes, and their IDs, connected to the start of the selected lane.
- **End Lanes** — Displays the list of lanes, and their IDs, connected to the end of the selected lane.

You can view the unique IDs for the **Lane Boundary** on the **Attributes** pane.

Barrier

- **Id** — Unique identification number for the selected barrier.
- **Side** — Lane side on which the barrier is placed, specified as left or right.
- **Extrusion Asset** — Displays the extrusion asset image available in the asset library which is relevant to the imported scene object.

Sign

- **Id** — Unique identification number for the selected scene object.
- **Prop Asset** — Displays the asset image available in the asset library which is relevant to the imported scene object.

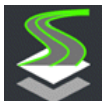
You can manually create on or around this data. Alternatively, if you have a RoadRunner Scene Builder license, click **Build Scene** to build roads and scene objects directly from the data.

View Apollo HD Map Scene Data for Visual Reference

View Apollo HD Map data in RoadRunner. You can use this data as a visual reference to manually create roads on or around it.

Import and Explore Data

Open the **Scene Builder Tool** from the toolbar by clicking the **Scene Builder Tool** button.



Then, import Apollo HD map data by clicking the **Open HD Map Root Folder** button on the toolbar to the left of the scene editing canvas.



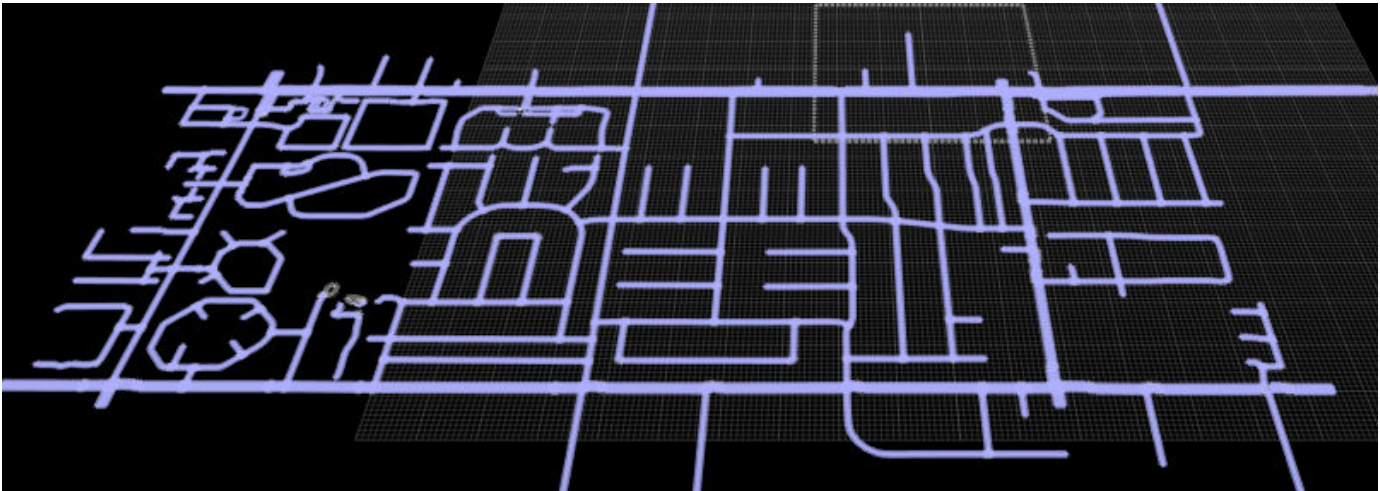
This opens an Import HD Data folder dialog box to specify the type and source of the HD Map data. Set the **Source Type** parameter to **Apollo**. RoadRunner Scene Builder supports the Apollo 3.0 and 5.0 XML formats.

Specify the **File Path** by navigating to the directory containing your Apollo HD Map data. Each file contains data such as lane borders, lane center lines, lane groups, lane trajectories, and junction areas.

RoadRunner saves the specified Import HD Data folder dialog box parameters for future RoadRunner sessions on your machine.

For this example, specify the **File Path** by navigating to a folder containing Apollo HD Map data, and import the data by selecting **Import**.

The **Scene Builder Tool** imports the Apollo HD Map data from the specified **File Path**, converts the data into a preview called a **RoadRunner HD Map**, and displays the **RoadRunner HD Map** in the scene editing canvas. For more information on **RoadRunner HD Map** data, see “RoadRunner HD Map Data” on page 1-166.



Explore the imported data by selecting lanes and lane boundaries. You can view their properties on the **Attributes** pane. The type of road element selected in the **RoadRunner HD Map** scene editing canvas determines the available properties.

Scene Builder

- **Color by Confidence Classification** — Disabled for imported Apollo HD data.
- **Selected Lane Length** — Length of the selected lanes.
- **Scene Lane Length** — Length of the total lanes in the scene.

Lane

- **Id** — Unique identification number for the selected lane.
- **Speed Limit** — Maximum allowed driving speed of the lane.
- **Lane Type** — Type of the lane.
- **Travel Direction** — Direction of travel for the road segment, specified as forward, backward, or bidirectional.
- **Start Lanes** — Displays the list of lanes, and their IDs, connected to the start of the selected lane.
- **End Lanes** — Displays the list of lanes, and their IDs, connected to the end of the selected lane.

You can view the unique IDs for the **Lane Boundary** on the **Attributes** pane.

You can manually create on or around this data. Alternatively, if you have a RoadRunner Scene Builder license, click **Build Scene** to build roads and scene objects directly from the data.

More About

RoadRunner HD Map Data

The **RoadRunner HD Map** is a preview that displays the imported HD Map data in the scene editing canvas. The **RoadRunner HD Map** data contains confidence color classification, lanes, lane boundaries, lane groups, lane length, barriers, poles, and traffic signs. You can also view their properties on the **Attributes** pane.

The **RoadRunner HD Map** displays trajectory lanes as gray, barriers as orange, poles as blue dots, and traffic signs as yellow. Selecting a lane border gives information of the extrusion types, such as guard rail and fence.

For imported TomTom HD Map data, the **Color by Confidence Classification** option is disabled, by default, and color classification is excluded in the **RoadRunner HD Map** preview.

Tips

You can view the **RoadRunner HD Map** while using a different tool by selecting **View > RoadRunner HD Map** from the menu bar.

Version History

Introduced in R2020b

See Also

World Settings Tool

Topics

“Build Scenes Using HERE HD Live Map Data”

“Build Scenes Using TomTom HD Map Data”

“Build Scenes Using Apollo HD Map Data”

“Configure Assets for Imported HERE HD Live Map Data”

“Configure Assets for Imported TomTom HD Map Data”

“Create Route and Build Scene Using HD Map Data”

Scene Export Preview Tool

Preview scene geometry to be exported

Description

The **Scene Export Tool** can be used to visualize the exact scene geometry that will be exported. You can also use it to query properties, such as triangle and material counts. Tile counts are determined by fitting the **Tile Size** and **Center** parameters to the **Scene Extents**.



Open the Scene Export Preview Tool

On the RoadRunner toolbar, click the **Scene Export Preview Tool** button:



Examples

Adjust the Export Tile Size

- 1 Select the **Export to Tiles** toggle.
- 2 Adjust the **Tile Size** parameters to modify the size of individual tiles.
- 3 Adjust the **Center** parameters to modify the overall tiling center.
- 4 Click **Preview Export**.

Preview Segmentation Output

- 1 Modify the **Split by Segmentation** toggle.
- 2 Press **Preview Export**.

- 3** Enable segmentation display to preview segmentation categories in this tool.

Version History

Introduced in R2020a

Topics

“Customize Levels of Detail in Exported Scenes”

Screenshot Tool

Generate and save image of current camera view

Description

The **Screenshot Tool** generates and saves an image of the current camera view. The dimensions of the screenshot can be set independently of the dimensions of the actual viewport. This allows for full control over the resolution and aspect ratio of the image without being limited to the resolution of the monitor. The **Screenshot Tool** also allows control over the camera field of view.



Open the Screenshot Tool

On the RoadRunner toolbar, click the **Screenshot Tool** button:



Examples

Generate a Screenshot

- 1 Click the **Screenshot Tool** button.
- 2 Adjust the resolution and field of view as desired through the **Attributes** pane.
- 3 Click the **Take Screenshot** button on the **Attributes** pane.
- 4 Specify the file name and location in the **File Save** dialog box and click **Save**.

Generate a Quick Screenshot

- 1 Open any tool with the desired camera position. Quick screenshots use the current viewport settings and resolution.
- 2 Either select **Tools > Quick Screenshot** from the menu bar or press **Ctrl+P**.
- 3 Specify the file name and location in the file save dialog box and click **Save**.

Version History

Introduced in R2020a

SD Map Viewer Tool

Generate 3D scenes using SD Map services

Description

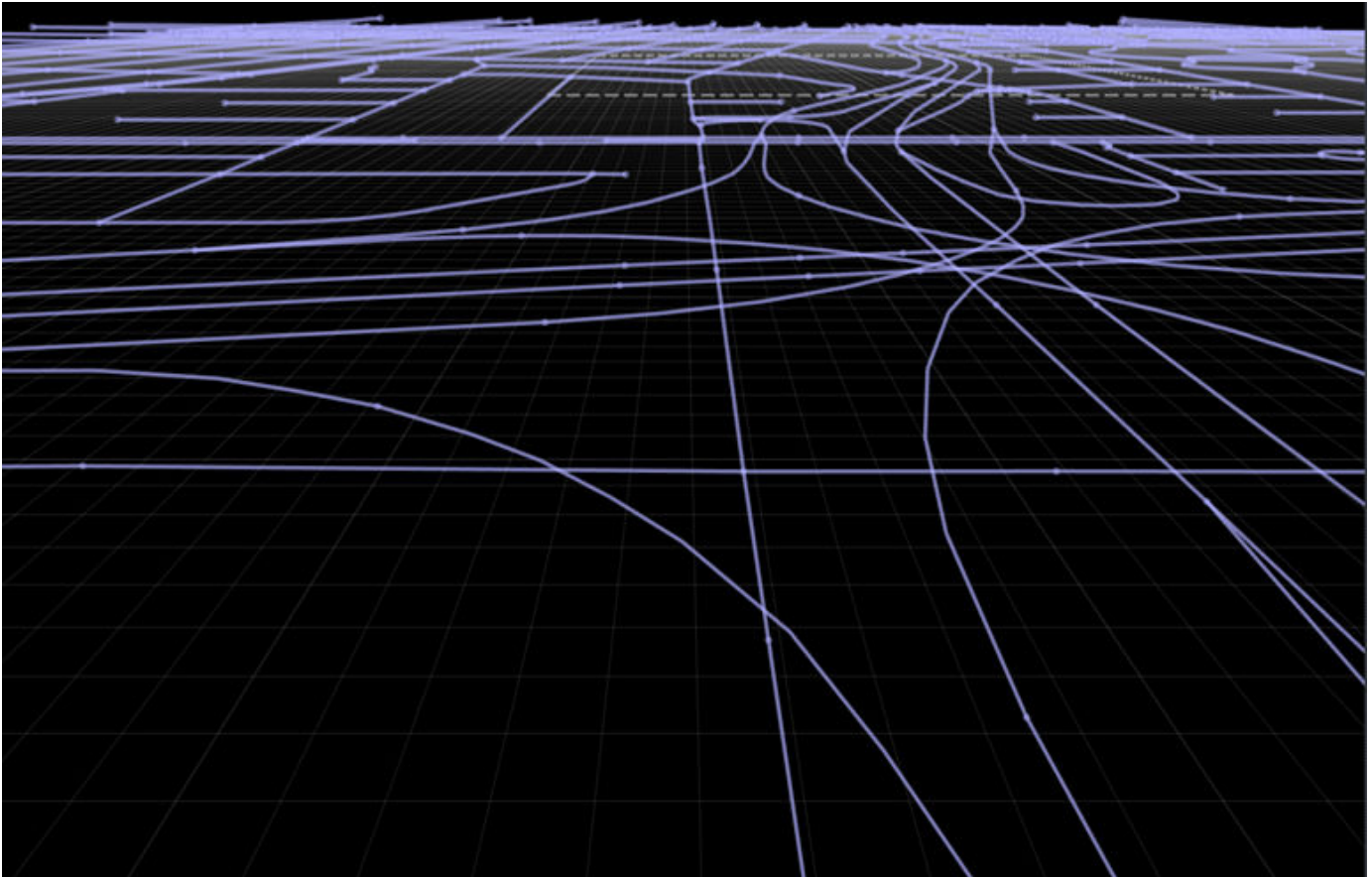
Use the **SD Map Viewer Tool** to import, inspect, and build scenes from SD Map data. After building the roads, you can edit and export the scene by using other RoadRunner capabilities.

RoadRunner supports importing map data from these SD Map services:

- Zenrin Japan Map API 3.0 (Itsumo NAVI API 3.0) ² — This cloud-based web service, developed by ZENRIN DataCom CO., LTD., enables you to access accurate, continuously updated map data. To import Zenrin Japan Map API 3.0 (Itsumo NAVI API 3.0) data, you must enter a valid **Access Client ID** and **Access Key Secret**.
- OpenStreetMap — This is a free, open-source, web map service that enables you to access crowd-sourced map data all over the world. To import OpenStreetMap data, you must have a valid OpenStreetMap file with the `.osm` file extension.

When you import map data from these web services, RoadRunner converts it to SD Map data containing nodes and links. Each link is a road segment with a node at each end, and each node connects to other road segments. When more than one link joins at a point, the SD Map allocates a single node to all of the links that connect at that location.

² To gain access to the Zenrin Japan Map API 3.0 (Itsumo NAVI API 3.0) service and get the required credentials (a client ID and secret key), you must enter into a separate agreement with ZENRIN DataCom CO., LTD.



Open the SD Map Viewer Tool

On the RoadRunner toolbar, click the **SD Map Viewer Tool** button:



Examples

View Zenrin Japan Map API 3.0 (Itsumo NAVI API 3.0) Scene Data for Visual Reference

View Zenrin Japan Map API 3.0 (Itsumo NAVI API 3.0) data in RoadRunner for an area in Koto City, Tokyo, Japan. You can use this data as a visual reference to manually create roads on or around it.

Choose Area of Interest

Specify the area of interest by using the **World Settings Tool**.



In the **Attributes** pane, in the **World Origin** section, specify the **Latitude** as 35.6380286 degrees and the **Longitude** as 139.7958767 degrees. In the **Workspace** section, under **Extents**, specify both **X** and **Y** as 500 meters. Apply your changes by selecting **Apply World Changes**.

Import and Explore Data

Open the **SD Map Viewer Tool** from the toolbar. Then, import the Zenrin Japan Map API 3.0 (Itsumo NAVI API 3.0) data by clicking the **Import Data For Area** button on the toolbar to the left of the scene editing canvas.

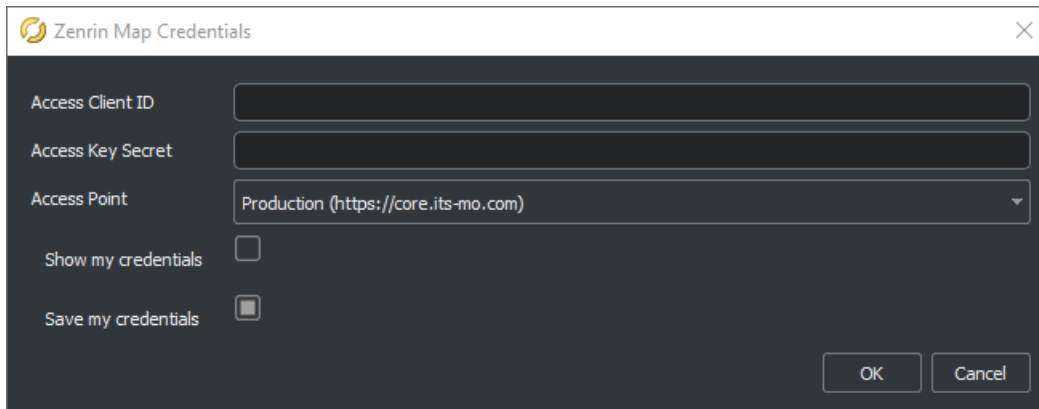


Before you import the data, you must enter valid Zenrin Japan Map API 3.0 (Itsumo NAVI API 3.0) access credentials in the Zenrin Map Credentials dialog box.

- **Access Client ID** — Specify your Zenrin client ID.
- **Access Key Secret** — Specify your Zenrin secret key.
- **Access Point** — Select the appropriate option for your Zenrin license:
 - **Production** (<https://core.its-mo.com>) — Use this option when you have a permanent license agreement to access Zenrin Japan Map API 3.0 (Itsumo NAVI API 3.0).
 - **Verification** (<https://test.core.its-mo.com>) — Use this option when you have an evaluation license agreement to access Zenrin Japan Map API 3.0 (Itsumo NAVI API 3.0).
- **Show my credentials** — Select this parameter to see the explicit values you enter in the **Access Client ID** and **Access Key Secret** fields. By default, this parameter is not selected and your input is hidden.
- **Save my credentials** — Select this parameter to save the specified credentials for future RoadRunner sessions. The credentials remain saved until you delete them. If you do not want to save the credentials for future sessions, clear this parameter. In this case, RoadRunner saves the credentials for only the rest of this RoadRunner session on your machine. By default, this parameter is not selected.

You can delete your saved credentials by clearing the **Access Client ID** and **Access Key Secret** boxes and clicking **OK**. RoadRunner displays an error message indicating an issue with either your credentials or your connection to the download server. Ignore the error message, and click **Cancel** to close the dialog box.

Once you have specified your access credentials, click **OK**.

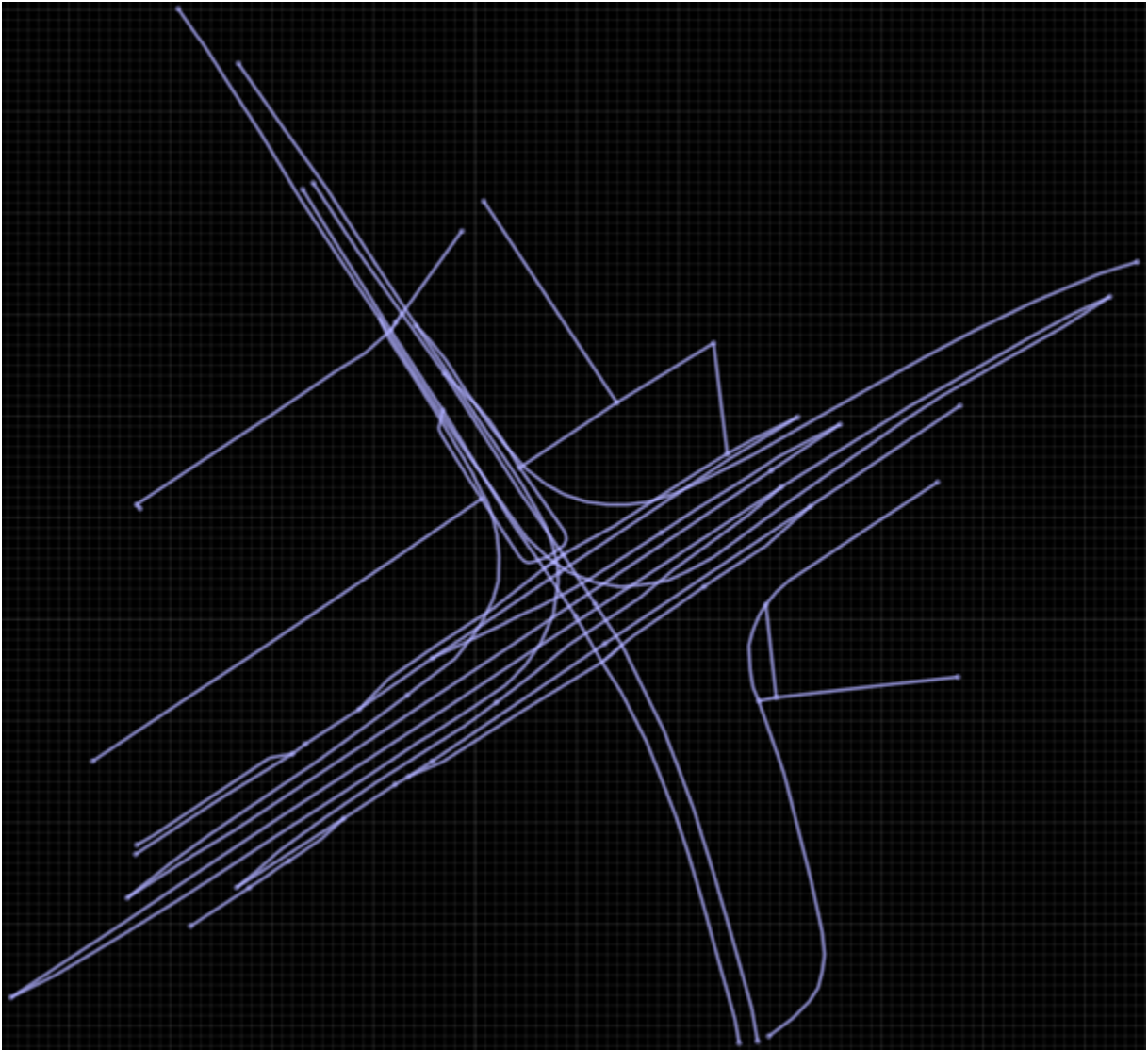


The image shows a dialog box titled "Zenrin Map Credentials" with a close button (X) in the top right corner. The dialog contains the following fields and options:

- Access Client ID:** A text input field.
- Access Key Secret:** A text input field.
- Access Point:** A dropdown menu currently showing "Production (https://core.its-mo.com)".
- Show my credentials:** An unchecked checkbox.
- Save my credentials:** A checked checkbox.

At the bottom right of the dialog are two buttons: "OK" and "Cancel".

The **SD Map Viewer Tool** imports SD Map data that intersects your workspace, converts the data into a preview called an **SD Map**, and displays the **SD Map** in the scene editing canvas. The **SD Map** displays the nodes and links of the road data.



Explore the imported data by selecting links and nodes. You can view their attributes in the **Attributes** pane. The type of road element selected in the SD Map scene editing canvas determines the available attributes.

Simple Link

- **Id** — Unique identification number for the selected link.
- **Skip During Build** — Specifies whether to add or skip this link during the build process. If you select this attribute, the **SD Map** represents this link as a dashed line, and the link is ignored in the build process. To include the link in the build process, which displays it as a solid line, clear this attribute.

If you clear the **Skip During Build** attribute, the **SD Map Viewer Tool** imports the actual links and displays them as solid lines.

SD Map data can contain unsupported links for the same area. By default, the **SD Map Viewer Tool** imports these unsupported links with the **Skip During Build** attribute enabled, displaying

them in the **SD Map** as dashed lines.. You can see a list of detected unsupported link IDs in the **Output** window. Select these IDs to navigate to and view the links in the **SD Map**.

Note You can click and drag to select multiple links within a rectangular region of interest. You can also hold **Shift** and click additional links to add them to the selection. You can control the **Skip During Build** attribute collectively, for all selected links, in the **Attributes** pane.

- **Road Width (in meters)** — **Min** and **Max** road width.
- **Number of Lanes** — Number of forward and backward lanes. The **Forward** and **Backward** attributes show the **Min** and **Max** values if the map data specifies it.
- **Travel Direction** — Direction of travel for the road segment, specified as **Forward**, **Backward**, or **Bidirectional**.

Each link has several control points, and each **Control Point** contains a **Position** attribute specifying its (X,Y,Z) location.

Simple Node

- **Id** — Unique identification number for the selected node.
- **Connecting Links** — Displays all the links connected to the selected node. Each connected link is labelled with its associated ID and orientation.

For details on the programmatic use of these parameters while building roads, see “Road Width and Number of Lane Calculations” on page 1-177.

View OpenStreetMap Scene Data for Visual Reference

View OpenStreetMap data in RoadRunner for the MathWorks® Apple Hill campus. You can use this data as a visual reference to manually create roads on or around it.

Import OpenStreetMap File

To import OpenStreetMap data, you must first select an OpenStreetMap file containing road geometry. To get these files, visit openstreetmap.org, specify a map location, manually adjust the region around this location, and export the road geometry for that region to an OpenStreetMap file with extension `.osm`. OpenStreetMap exports only the roads whose whole lengths are within the specified region. In this example, you use an OpenStreetMap file previously exported from the website.

- 1 Open the **SD Map Viewer Tool** from the toolbar by clicking the **SD Map Viewer Tool** button.



- 2 Click the **Open OpenStreetMap File** button on the toolbar to the left of scene editing canvas.



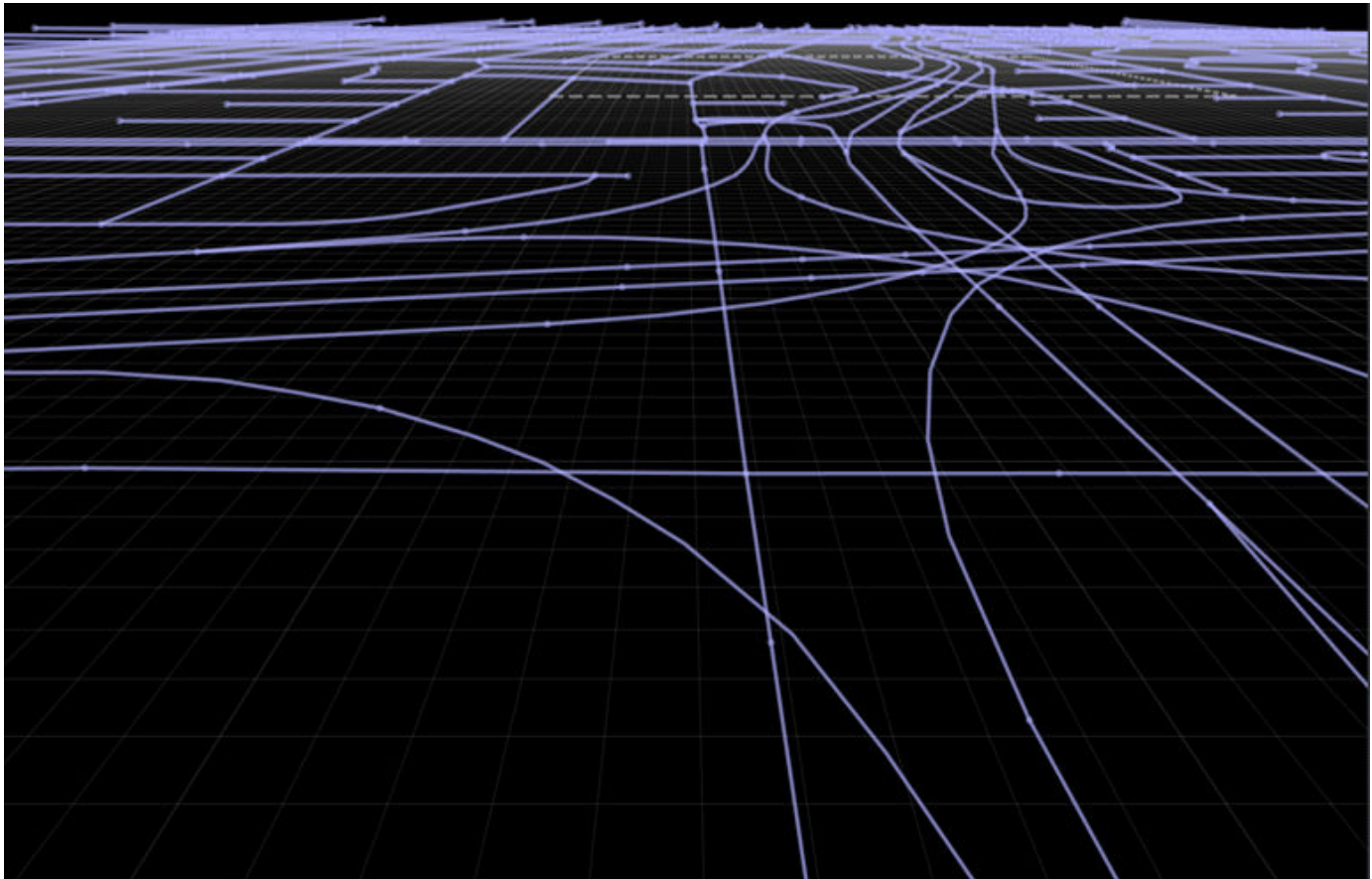
- 3 In the Open OpenStreetMap File dialog box, browse to this file, select it, and click **Open**.

`RRInstallFolder/bin/platform/AssetsInstall/SampleFiles/applehill.osm`

- *RRInstallFolder* is your local RoadRunner installation folder.
- *platform* is the folder name for your OS platform.

The file was downloaded from <https://www.openstreetmap.org>, which provides access to crowd-sourced map data all over the world. The data is licensed under the Open Data Commons Open Database License (ODbL), <https://opendatacommons.org/licenses/odbl/>.

The **SD Map Viewer Tool** imports SD Map data that intersects your workspace, converts the data into a preview called an **SD Map**, and displays the **SD Map** in the scene editing canvas. The **SD Map** displays the nodes and links of the road data.



Note

- When you import an OpenStreetMap file into a new scene, the **SD Map Viewer Tool** automatically sets the world origin using the geographic bounds specified in the file. However, to successfully import an OpenStreetMap file into an existing scene that has an already specified **World Origin**, the geographic bounds specified in the file must approximately match the **World Origin** value of the existing scene.
 - OpenStreetMap does not specify road junction information. When a road intersects another road, the **SD Map Viewer Tool** generates a separate link for each segment of the intersecting roads before and after the intersection. As a result, the number of generated links does not match the number of roads specified in the OpenStreetMap file.
-

Explore Imported Data

Explore the imported data by selecting links and nodes. You can view their attributes in the **Attributes** pane. The type of road element selected in the SD Map scene editing canvas determines the available attributes.

Simple Link

- **Id** — Unique identification number for the selected link.
- **Skip During Build** — Specifies whether to add or skip this link during the build process. If you select this attribute, the **SD Map** represents this link as a dashed line, and the link is ignored in the build process. To include the link in the build process, which displays it as a solid line, clear this attribute.

If you clear the **Skip During Build** attribute, the **SD Map Viewer Tool** imports the actual links and displays them as solid lines.

Note You can click and drag to select multiple links within a rectangular region of interest. You can also hold **Shift** and click additional links to add them to the selection. You can control the **Skip During Build** attribute collectively, for all selected links, in the **Attributes** pane.

- **Road Width (in meters)** — Width of the road corresponding to the selected link.
- **Number of Lanes** — Number of **Forward** and **Backward** lanes for the road corresponding to the selected link.
- **Travel Direction** — Direction of travel for the road corresponding to the selected link, specified as **Forward**, **Backward**, or **Bidirectional**. If the input file does not specify the oneway tag for a road, the **SD Map Viewer Tool** assumes the road is bidirectional.

Each link has several control points and each **Control Point** contains a **Position** attribute specifying its (X,Y,Z) location.

Simple Node

- **Id**— Unique identification number for the selected node.
- **Connecting Links**— Displays all the links connected to the selected node. Each connected link is labelled with its associated ID and orientation.

For details on the programmatic use of these parameters while building roads, see “Road Width and Number of Lane Calculations” on page 1-177.

Tips

You can view the **SD Map** while using a different tool by selecting **View > SD Map** from the menu bar. Alternatively, you can press **F11**.

Algorithms

Road Width and Number of Lane Calculations

Zenrin Japan Map API 3.0 (Itsumo NAVI API 3.0) Data

The Zenrin Japan Map API 3.0 (Itsumo NAVI API 3.0) data format supports specifying minimum and maximum values for the road width, number of forward lanes, and the number of backward lanes of

each road. This list shows how the **SD Map Viewer Tool** estimates these parameters when you import SD Map data and build roads:

- Road width — The **SD Map Viewer Tool** programmatically estimates the width of each road based on the **Min** and **Max** road width parameters of that road. You can find the **Min** and **Max** values for the **Road Width (in meters)** attribute for each link in the **Attributes** pane. The calculation changes depending on whether **Min** and **Max** are undefined.

Min Road Width	Max Road Width	Road Width (estimated)
X	Y	$(X + Y)/2$
X	-1 (undefined)	X
-1 (undefined)	Y	Y
-1 (undefined)	-1 (undefined)	3.5 m

- Number of lanes (forward or backward) — The **SD Map Viewer Tool** builds each road using the specified minimum value for the number of lanes in each direction. If both the minimum and maximum values are not specified, the **SD Map Viewer Tool** assumes the number of lanes per direction of travel as 1. If the input data specifies the minimum and maximum values for the number of forward or backward lanes, you can find that information for each link in the **Attributes** pane.

Note The **Attributes** pane represents an unspecified value for the number of forward or backward lanes as -1.

OpenStreetMap Data

When you import OpenStreetMap data and build roads, RoadRunner internally calculates these parameters:

- Road width — The **SD Map Viewer Tool** sets the lane width to 3.5 meters, because OpenStreetMap does not provide lane width information. The estimated road width is the product of the lane width and the number of lanes within a road.
- Number of lanes — The **SD Map Viewer Tool** sets the number of lanes as specified in the input file. If the input file does not specify the number of lanes for a road, the **SD Map Viewer Tool** estimates one lane for each travel direction. By default, one-way roads have one lane and bidirectional roads have two total lanes, one for each travel direction.

Version History

Introduced in R2021b

See Also

World Settings Tool

Topics

“Build Roads by Using Zenrin Japan Map API 3.0 (Itsumo NAVI API 3.0) Data”

“Build Roads Using OpenStreetMap Data”

“Export to ASAM OpenDRIVE”

“Importing ASAM OpenCRG Files”

“Importing ASAM OpenDRIVE Files”

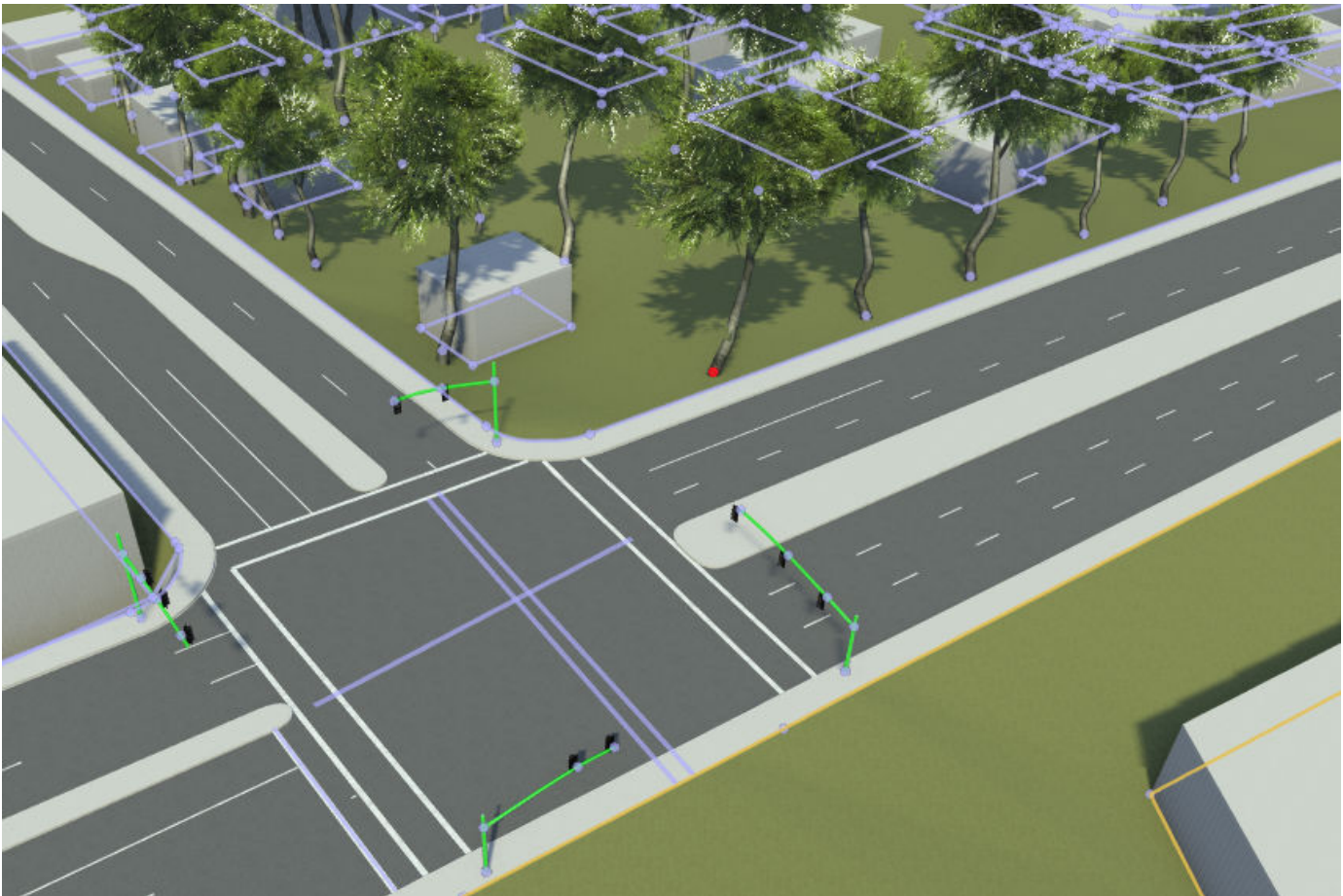
Selection Tool

Select one or more roads and props in a scene

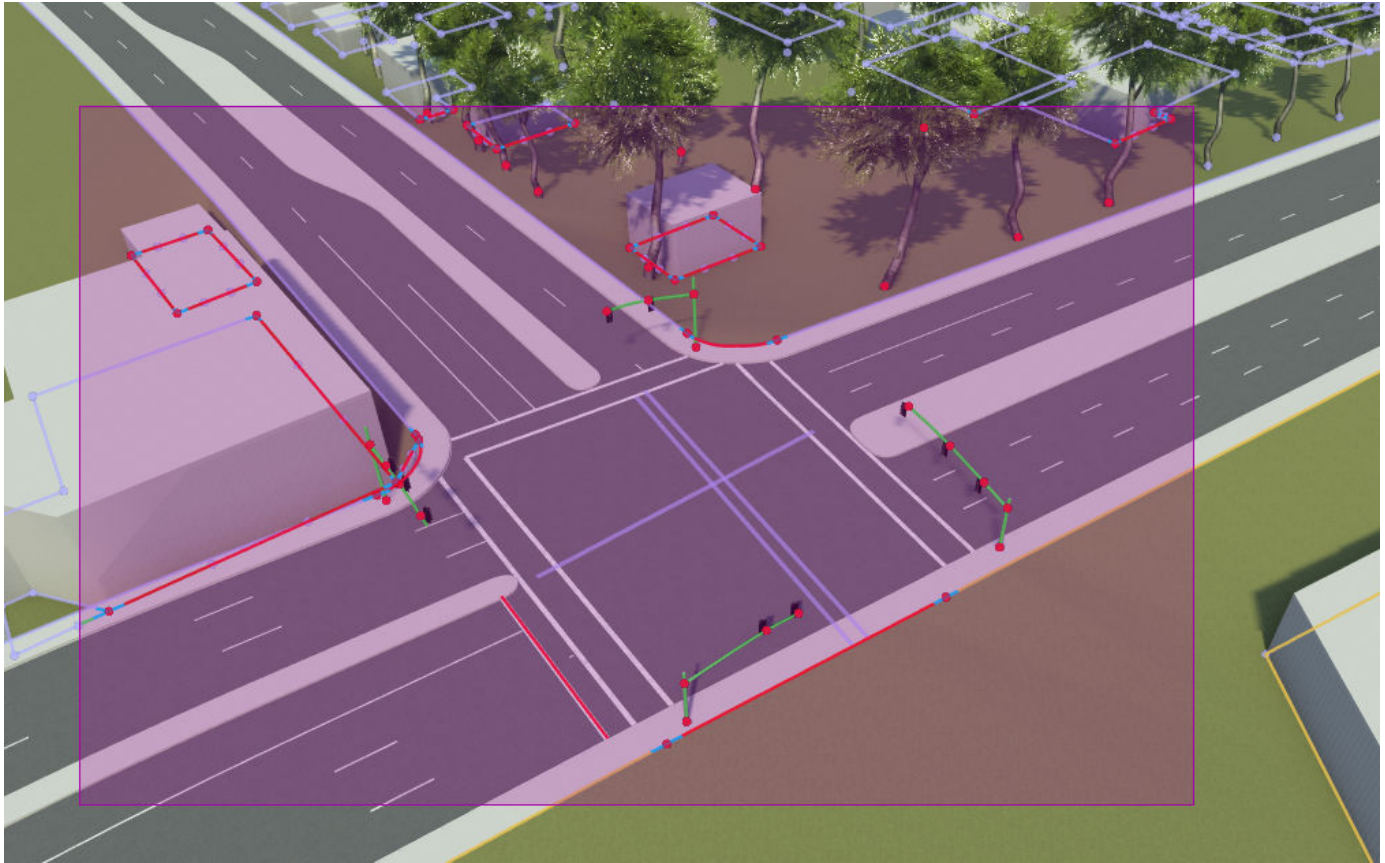
Description

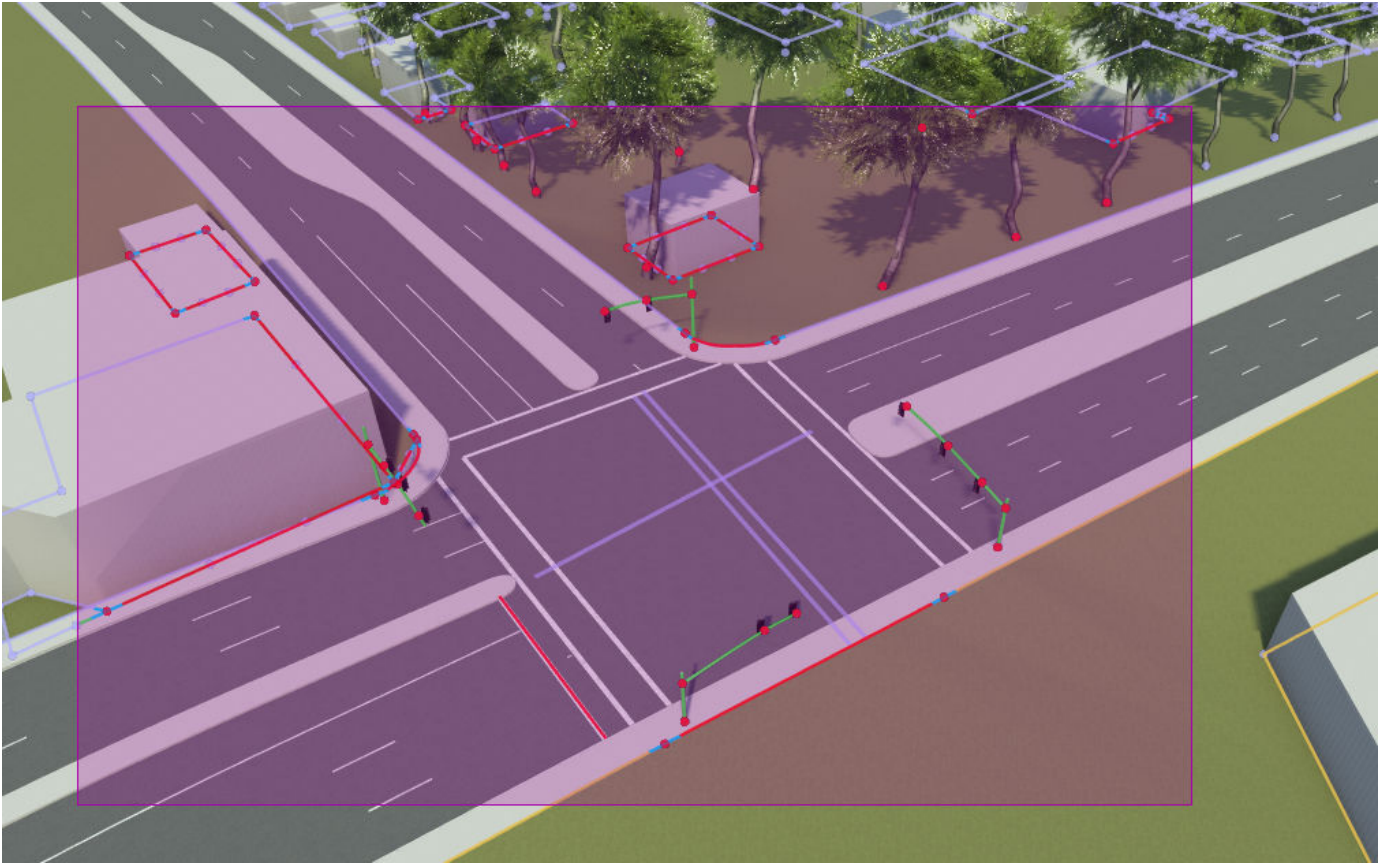
The **Selection Tool** enables you to select one or more of roads and props in the scene.

To select a single road or prop object in the scene, select the **Selection Tool**. When the **Selection Tool** is active, the active scene road and prop points are highlighted. Left-click an object in the 3D scene to select it. The root point of the selected object highlights in red. This image shows a selected tree prop.



To select a group of objects in the scene, select the **Selection Tool**. When the **Selection Tool** is active, left-click in the 3D scene and drag to select the objects in a rectangular region. The root points of the selected objects highlight in red. This image shows the selection of a group of objects.





Open the Selection Tool

On the RoadRunner toolbar, click the **Selection Tool** button:



More About

Prop Assemblies

A group of connected objects can be saved as an asset to form a scene assembly. Scene assemblies enable you to re-instantiate a complex group of attached objects in a scene. For more details, see “Create, Import, and Modify Scene Assets”.

Version History

Introduced in R2021b

See Also

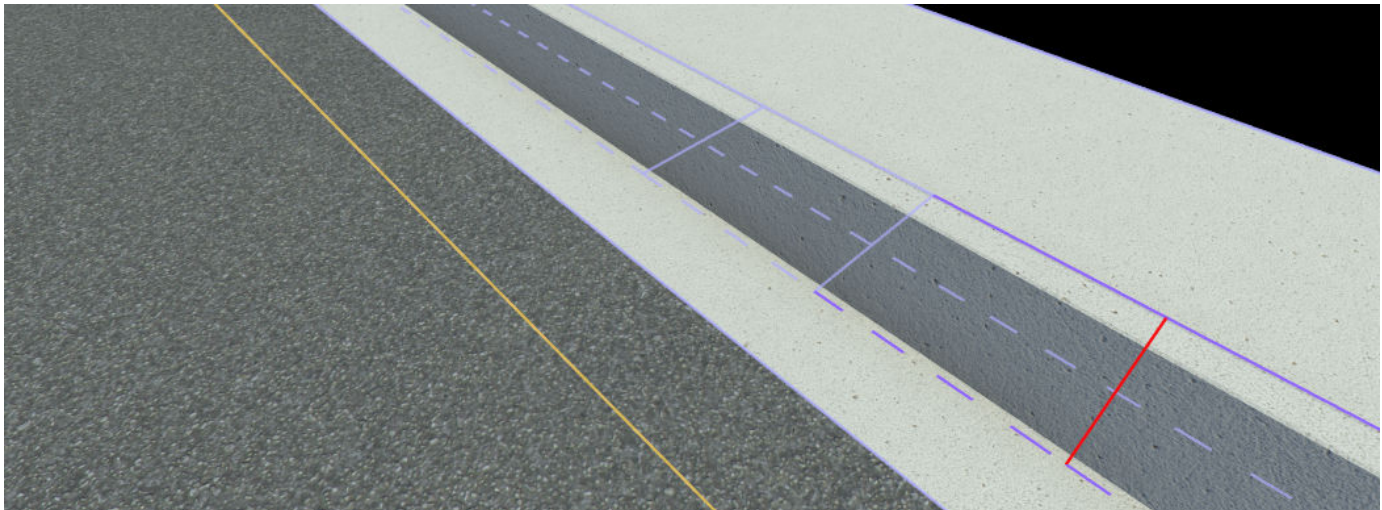
“Create, Import, and Modify Scene Assets”

Sidewalk Height Tool

Modify sidewalk and curb heights

Description

The **Sidewalk Height Tool** enables you to modify the heights of sidewalks and curb heights along a road. You can modify these heights along the entire length of a road or at arbitrary points along a road.



Open the Sidewalk Height Tool

On the RoadRunner toolbar, click the **Sidewalk Height Tool** button:



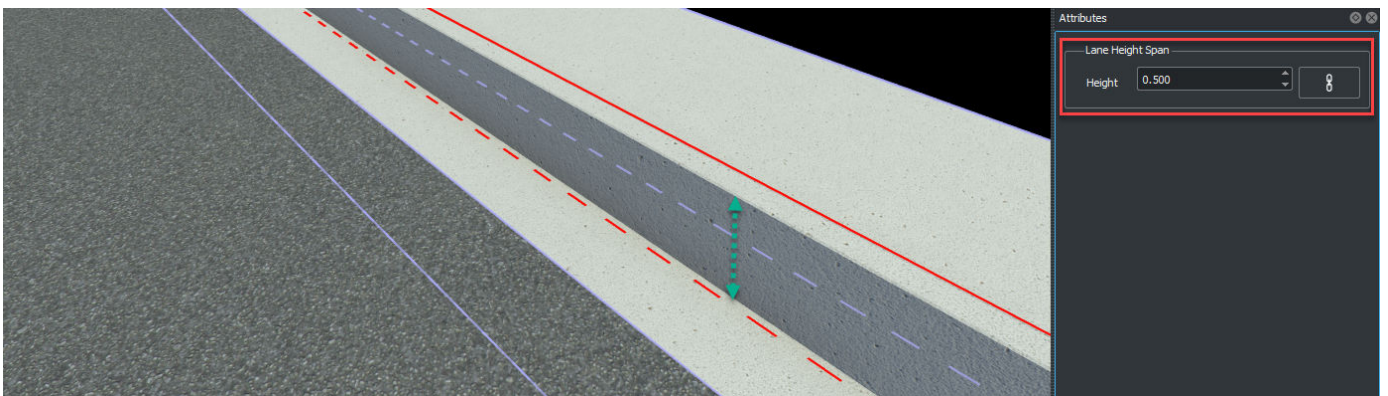
Examples

Modify Height of Sidewalk

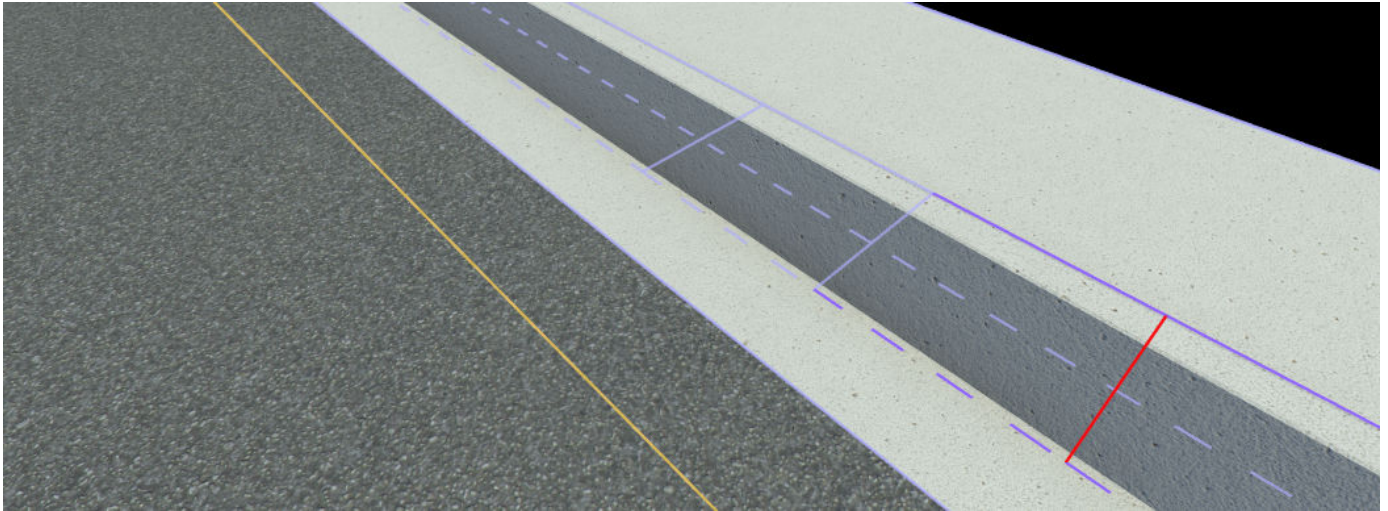
- 1 Create a straight road segment by using the **Road Plan Tool**. Zoom in on one side of the road and rotate the camera so that part of the sidewalk is clearly in view.



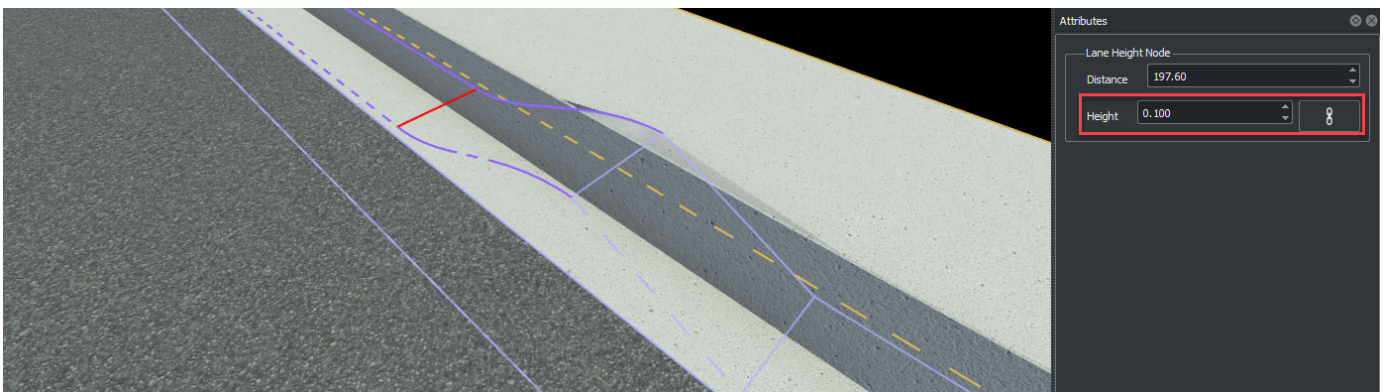
- 2 Click the **Sidewalk Height Tool** button.
- 3 While pressing Shift, click on the sidewalk lane and curb lane to select them. Then, in the **Attributes** pane, under **Lane Height Span** set the **Height** attribute to 0.5 meters. This increases the height of the sidewalk along the entire road edge.



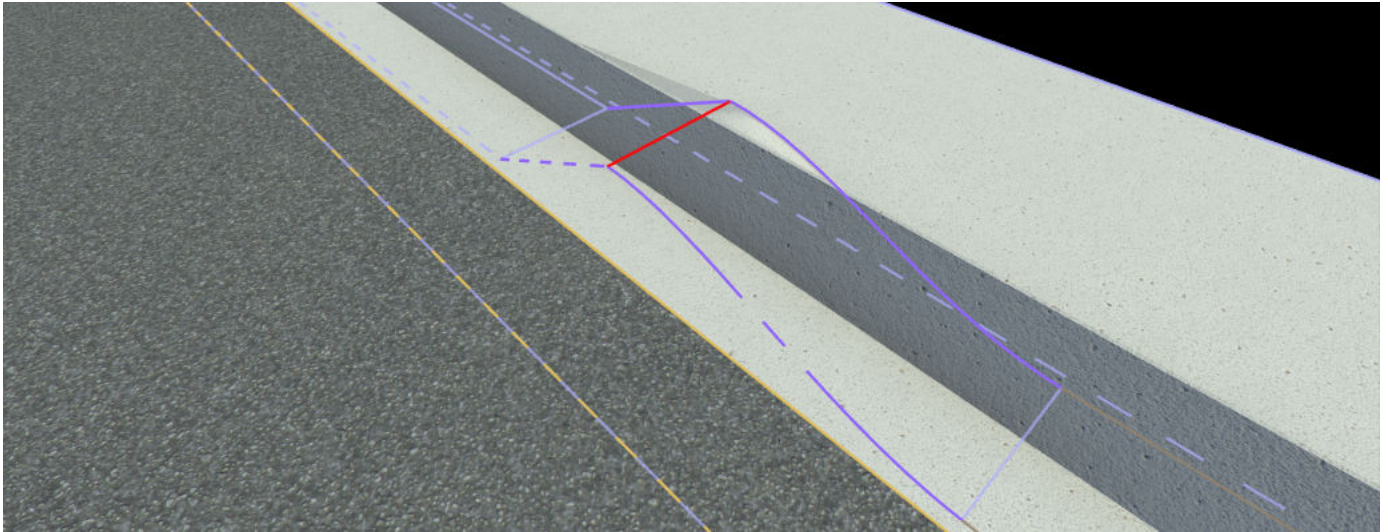
- 4 Right-click along the curb lane and along the sidewalk lane to add three evenly spaced height nodes. The unselected height nodes are represented by a horizontal lavender line and the selected height node is represented by a red horizontal line.



- 5 Select the outer nodes for the curb and sidewalk lane. In the **Attributes** pane, under **Lane Height Node**, set the **Height** to **0.1** meters. The sidewalk now has a slope at the center node, which still has a height of 0.5 meters.



- 6 Drag the center node towards one of the outer nodes. The sidewalk height changes as you drag the center node. You cannot drag a sidewalk height node past other height nodes or past the start or end of a lane.

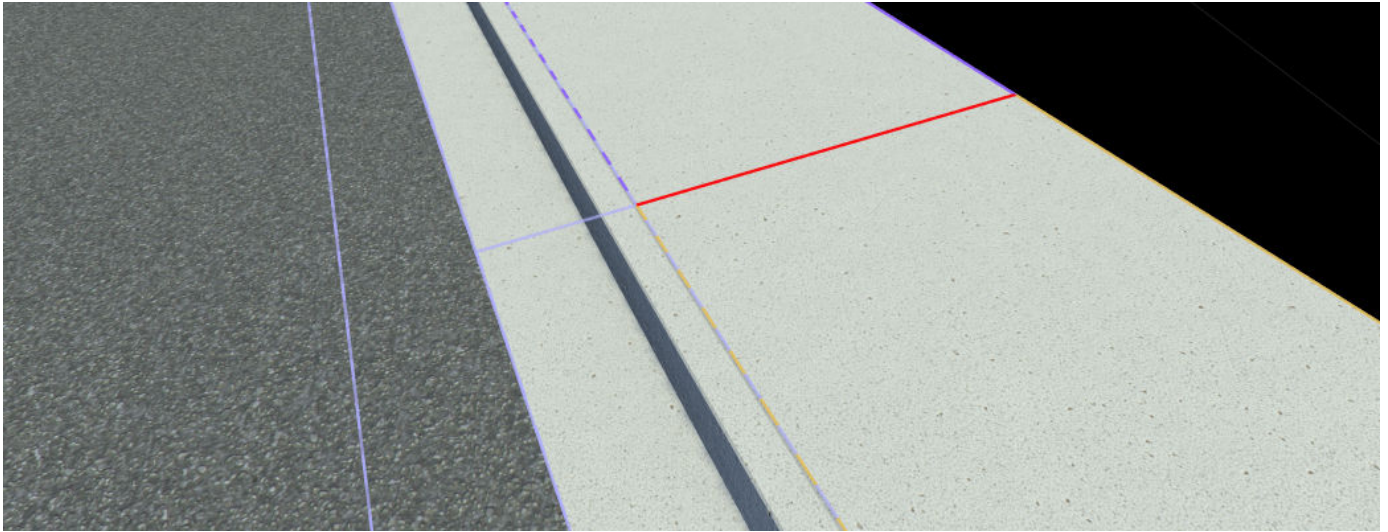


Modify Inner and Outer Heights of Sidewalk

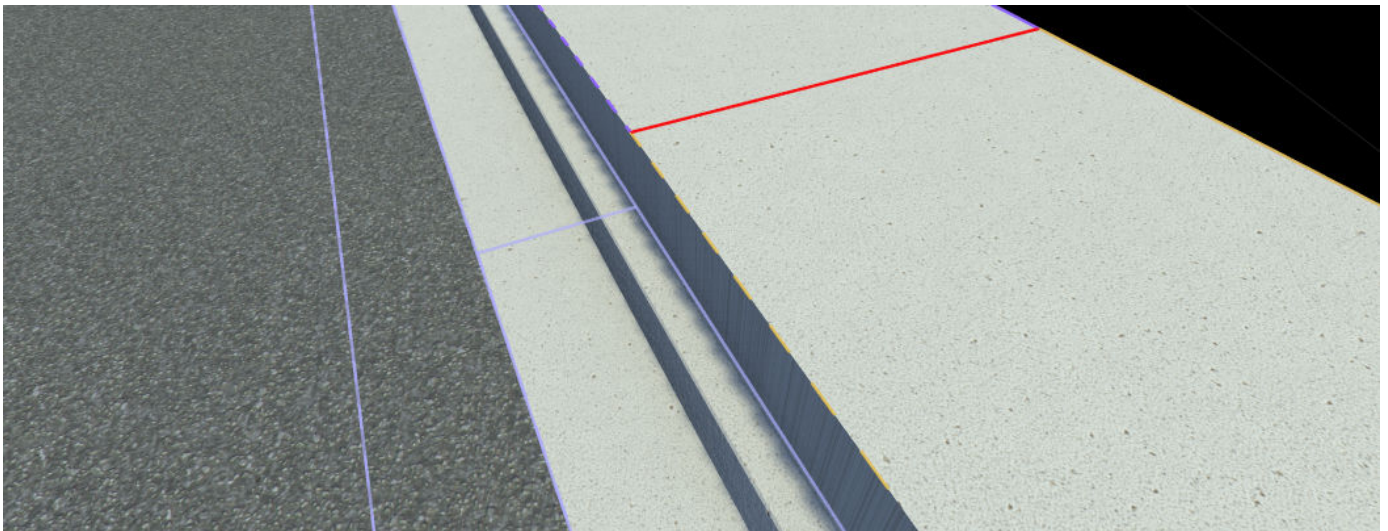
- 1** Create a straight road segment by using the **Road Plan Tool**. Zoom in on one side of the road and rotate the camera so that part of the sidewalk is clearly in view.



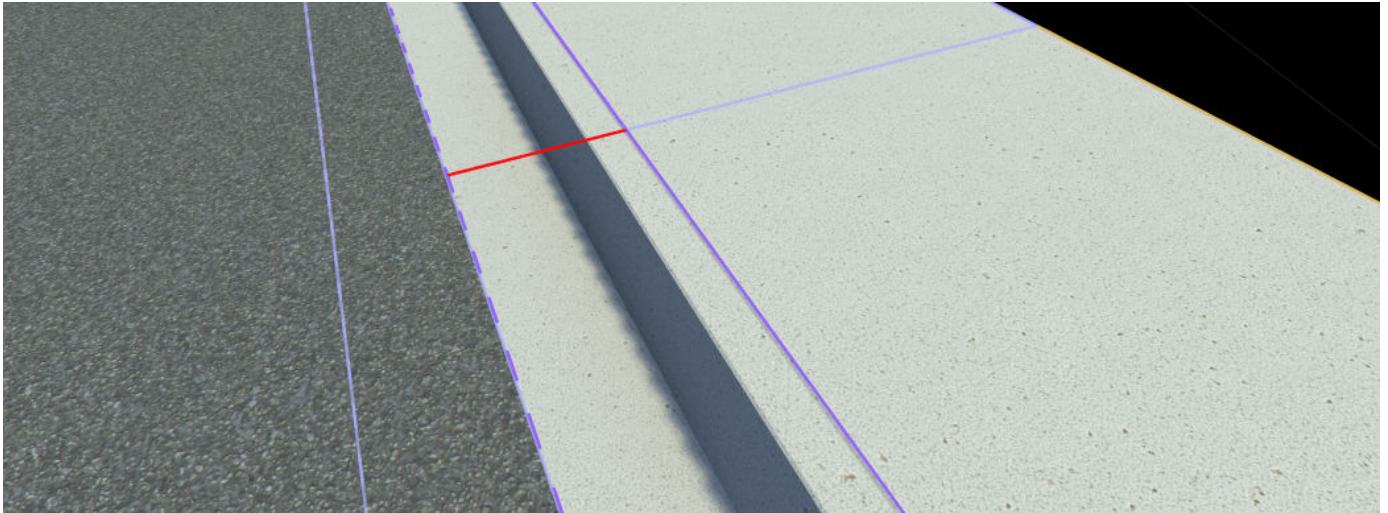
- 2** Click the **Sidewalk Height Tool** button.
- 3** Right-click along the curb lane and along the sidewalk lane to add height nodes along the curb and the sidewalk. The height nodes are represented by horizontal lines. Click the node on the sidewalk. It turns red.




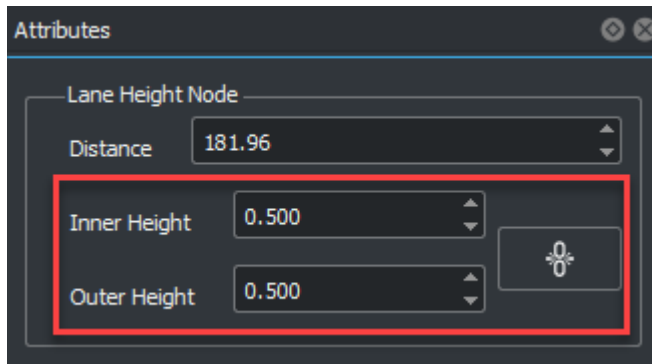
- 4** In the **Attributes** pane, under **Lane Height Node**, set the **Height** to 0.5 meters. This sets both the inner and outer heights of the sidewalk to 0.5 meters. By default, the inner and outer height nodes are linked together and assigning one value to the **Height** attribute sets both the inner and outer heights of the sidewalk to the same value.




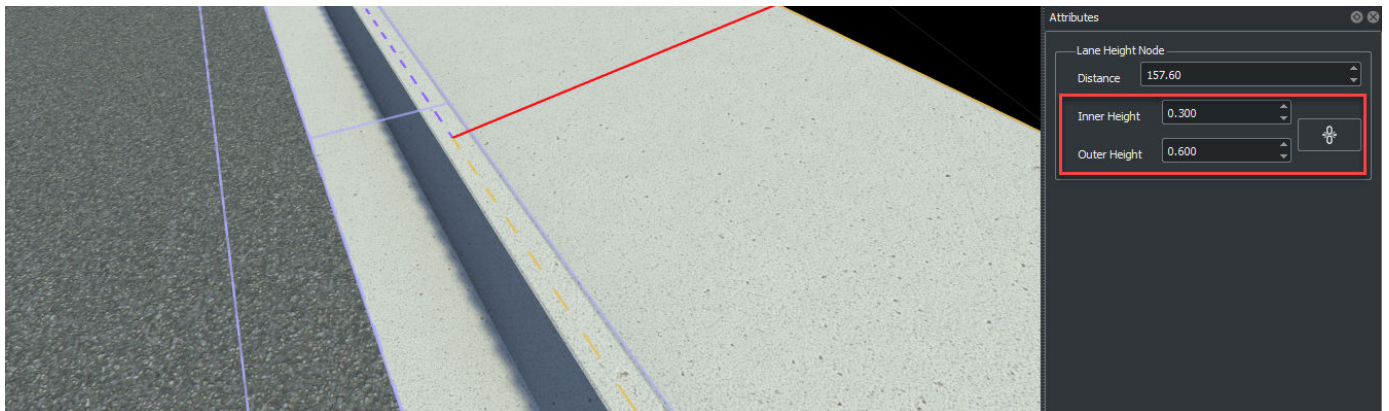
- 5** Click the height node on the curb to select it. In the **Attributes** pane, under **Lane Height Node**, set the **Height** to 0.5 meters. This sets both the inner and outer heights of the curb to 0.5 meters.




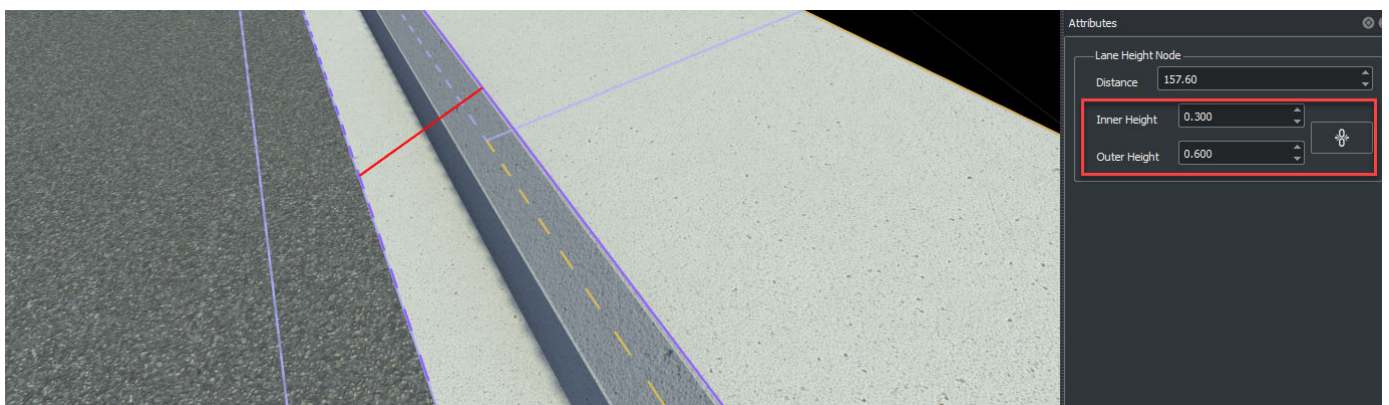
- 6 To assign different height values to the inner and outer parts of the sidewalk, select the height node on the sidewalk. Then, click the link button  in the **Attributes** pane under **Lane Height Node**. This unlinks the inner and outer heights, creating two separate input fields where you can enter the inner and outer height values individually.



- 7 In the **Attributes** pane, under **Lane Height Node**, set the **Inner Height** to 0.3 meters and the **Outer Height** to 0.6 meters. This sets the inner and outer heights of the sidewalk to two different values. To use the same values for inner and outer heights, click the unlink button  in the **Attributes** pane.



- 8 Click the height node on the curb to select it. In the **Attributes** pane, under **Lane Height Node**, set the **Inner Height** to 0.3 meters and the **Outer Height** to 0.6 meters. This sets the inner and outer heights of the curb to two different values. To use the same values for inner and outer heights, click the unlink button  in the **Attributes** pane.



Version History

Introduced in R2021a

See Also

Cross Section Tool

Signal Tool

Configure junction signalization and signal traffic phases

Description

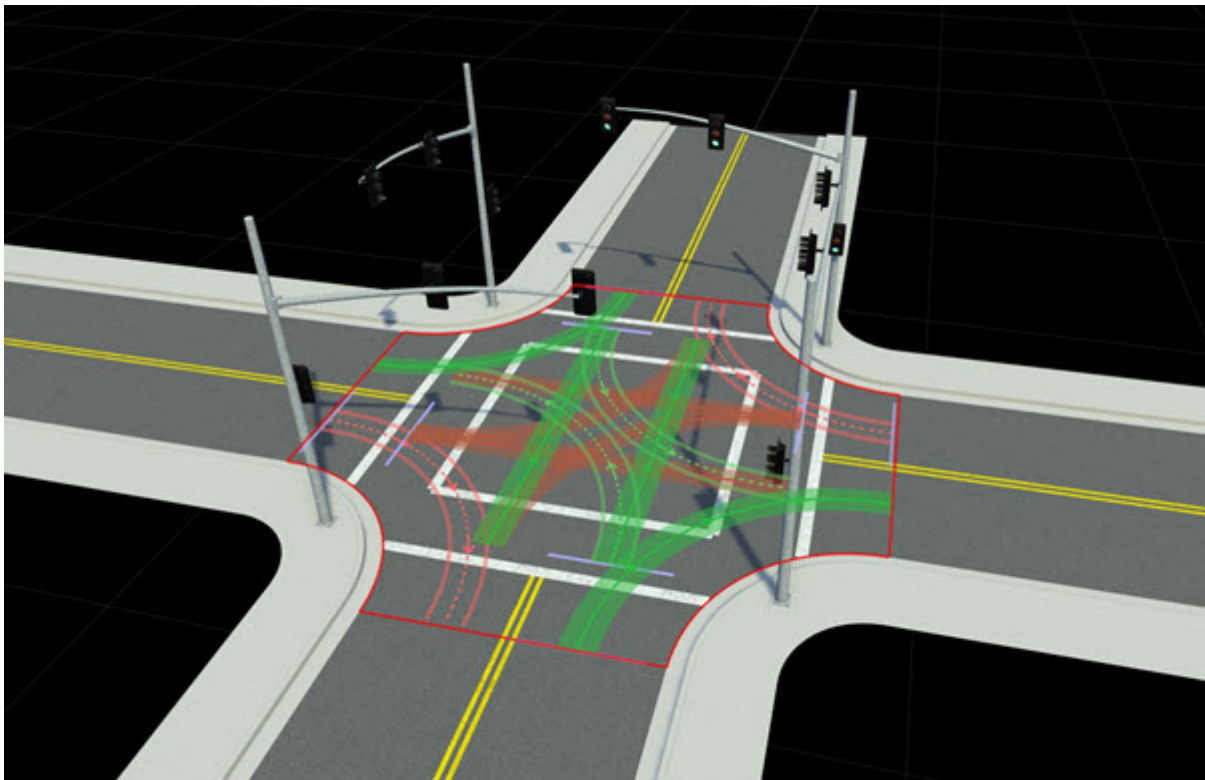
The **Signal Tool** is used to configure junction signalization and signal traffic phases.

A junction's signalization can be static (not changing, for example, controlled by stop signs) or dynamic (controlled by dynamic traffic signals). The signalization of a junction is defined using phases. A phase indicates which signals are active and the state of the maneuver roads (for example, whether traffic may enter the junction along a given maneuver road — for more details, see **Maneuver Tool**).

Each phase is composed of several intervals. An interval is a period in a junction that corresponds to allowed movements. Typically, there are three intervals in a phase: green, yellow, and red.

The **Signal Tool** provides several autosignalization operations for automatically applying signalization templates to a junction. These operations can also automatically place **Prop Assembly Assets** and **Signal Assets**.

Signal assets are linked to junctions by associating them to maneuver roads. This association can be performed manually, but an automatic detection operation can attempt to identify nearby signals and compute associations. A single signal asset can be associated with maneuver roads in multiple separate junctions.



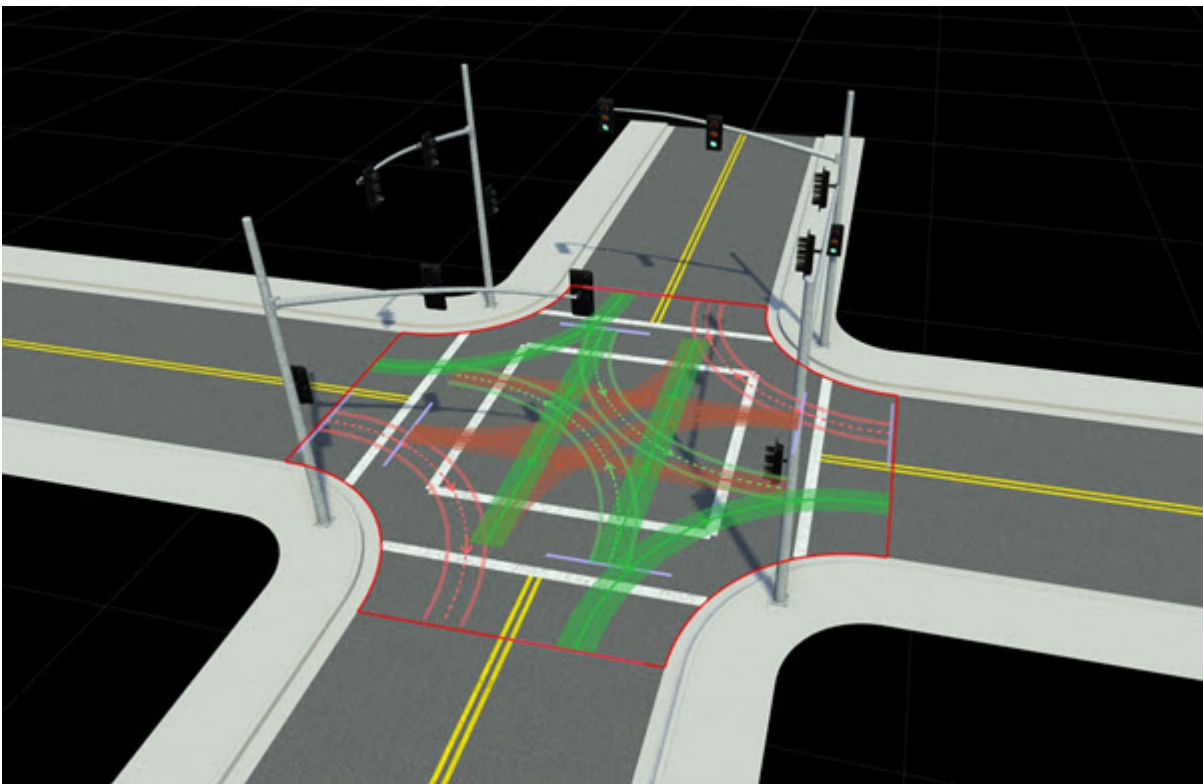
Open the Signal Tool

On the RoadRunner toolbar, click the **Signal Tool** button:



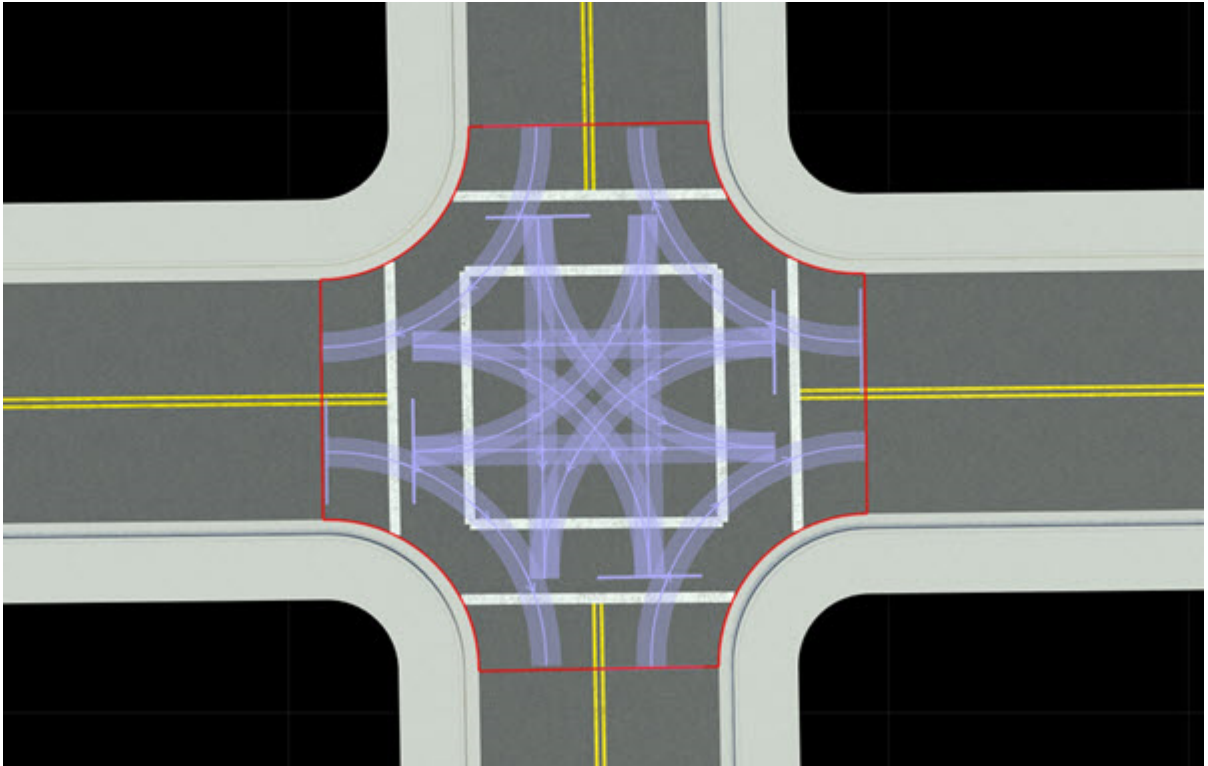
Examples

Autosignalize a Junction

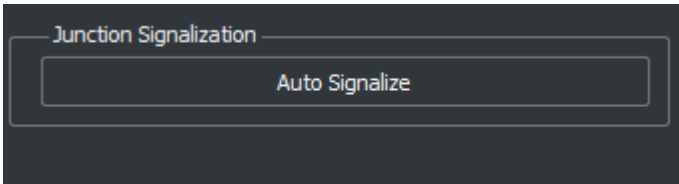


RoadRunner provides an autosignalization feature that can be used to apply common signalization templates to an intersection. For example, use these steps to quickly configure a four-way stop, a signalized intersection with protected left turns, and so on:

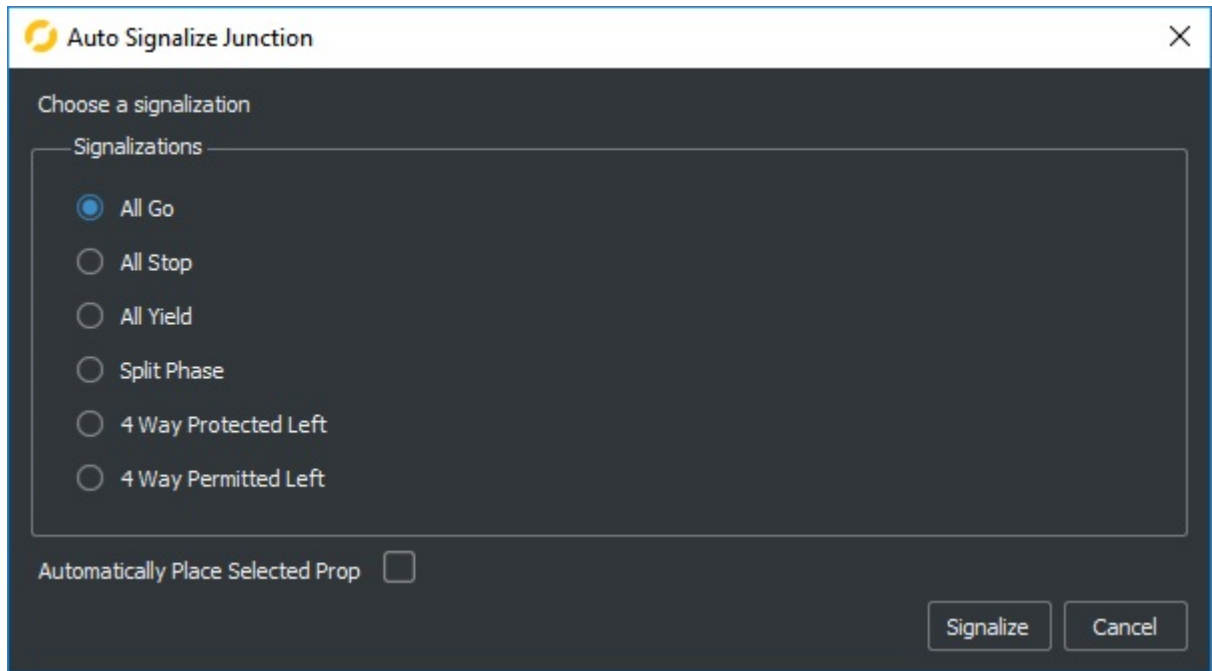
- 1 Click the **Signal Tool** button.
- 2 Select a junction.



3 In the **Attributes** pane, click **Auto Signalize**.



4 From the Auto Signalize Junction window, select a signalization template. Then, click **Signalize**.

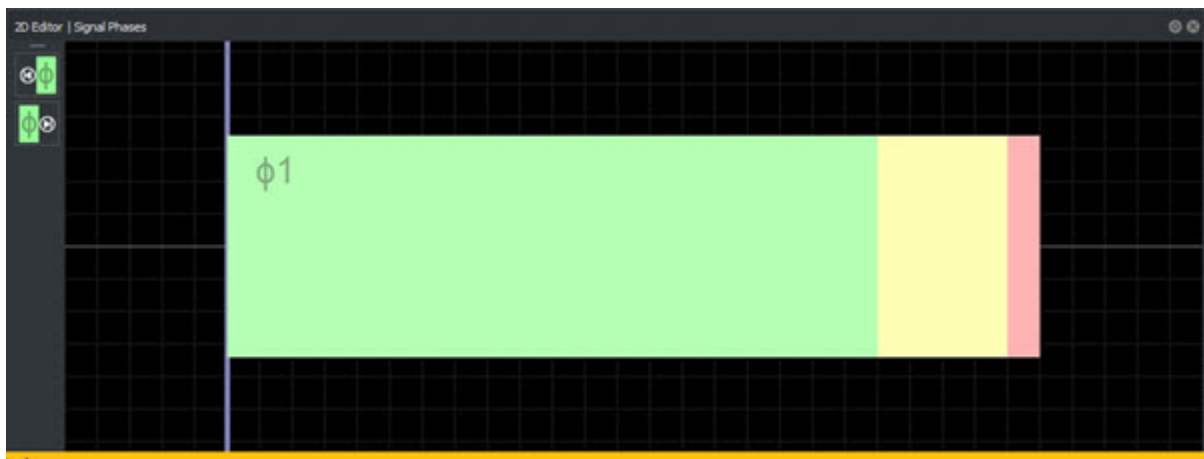


Tip If a prop or assembly is selected in the **Library Browser** first, select **Automatically Place Selected Prop** to automatically place the prop and connect any signals to the junction.

Add an Empty Phase

- 1 Click the **Signal Tool** button.
- 2 Select a junction.
- 3 Right-click beyond the end of the phases in the **2D Editor** pane.

Alternatively, right-click an existing phase to duplicate it.



Clear All Phases

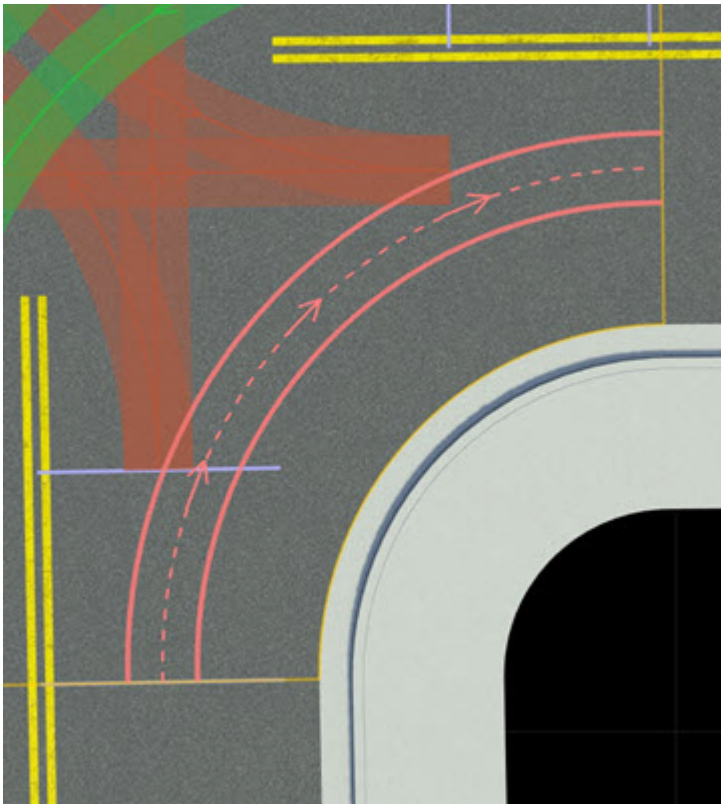
- 1 Click the **Signal Tool** button.

- 2 Select a junction.
- 3 Press the **Delete** key or select **Edit > Delete** from the menu bar.

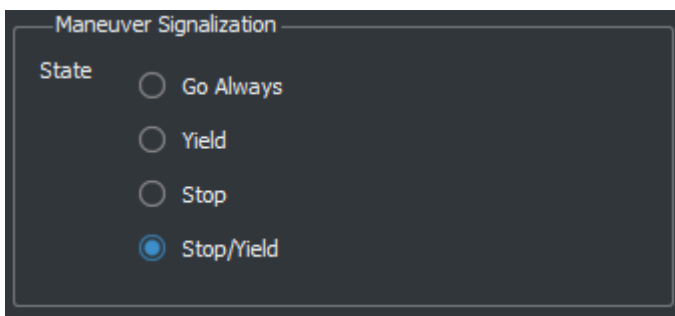
Specify a Maneuver Road's State in a Phase

To control the traffic state of a maneuver road in a given signal phase, follow these steps:

- 1 Click the **Signal Tool** button.
- 2 Select a junction.
- 3 Select the desired phase in the **2D Editor** pane.
- 4 Select the maneuver road whose state you want to change.



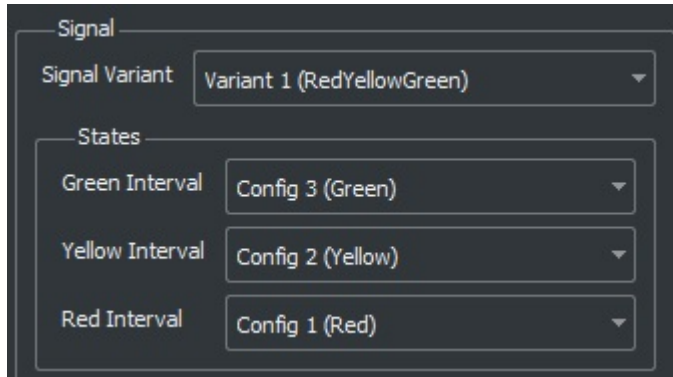
- 5 Choose a **State** in the **Attributes** pane.



Specify a Signal State in a Phase

To control a traffic signal's bulb states in each interval of a given signal phase, follow these steps:

- 1 Click the **Signal Tool** button.
- 2 Select a junction.
- 3 Select a gate. Gates appear as lavender bars on maneuver roads.
- 4 Select a signal you want to disassociate.
- 5 Choose a signal state for each interval in the **States** group of the **Attributes** pane.

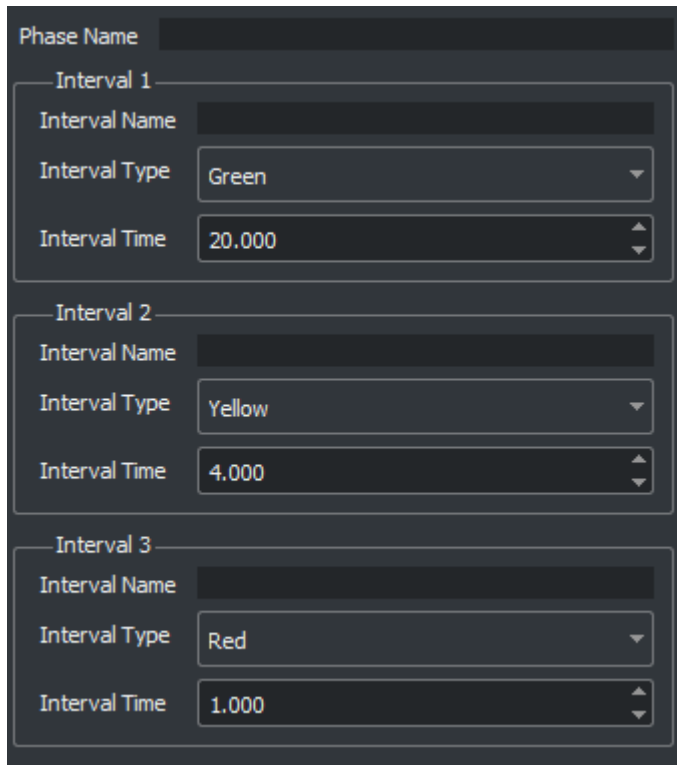


Note If no **States** group appears, then the selected signal is not associated with any maneuver road gates in this junction. Refer to the Associate a Signal with a Maneuver Road Gate on page 1-197 section.

Edit a Phase Duration

To change the duration of each interval in a signal phase, follow these steps:

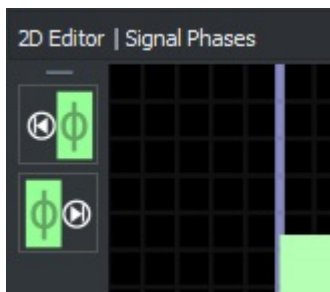
- 1 Click the **Signal Tool** button.
- 2 Select a junction.
- 3 Select the desired phase in the **2D Editor** pane.
- 4 Change the **Interval Time** in the **Attributes** pane.



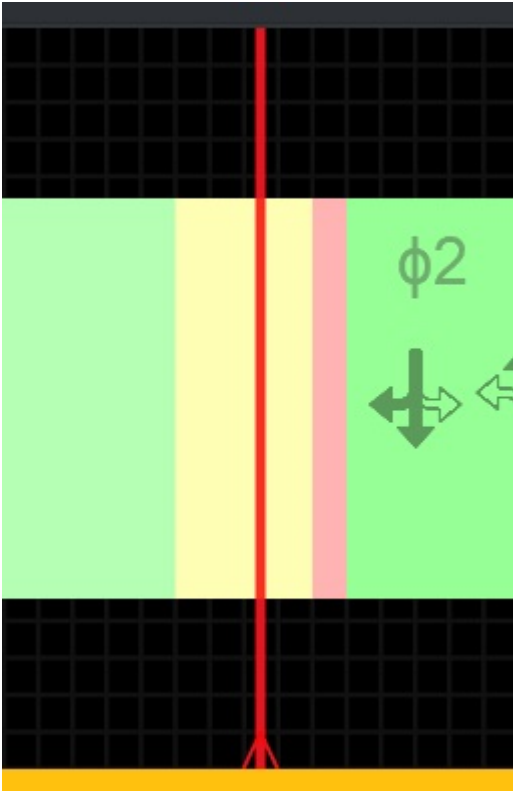
Change the Current Phase

- 1 Click the **Signal Tool** button.
- 2 Select a junction.
- 3 Select the desired phase in the **2D Editor** pane.

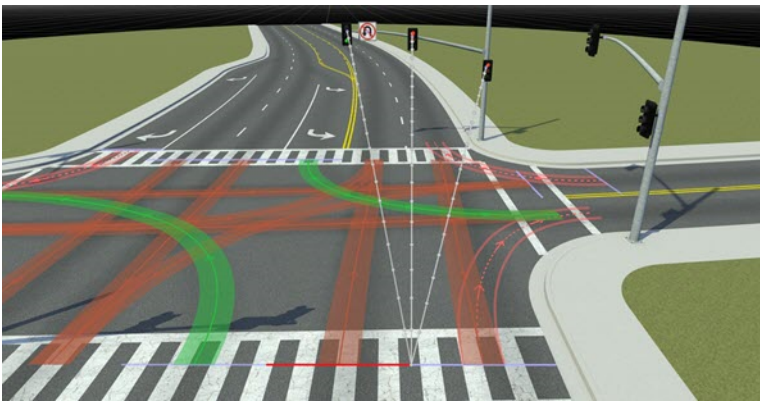
Alternatively, click the phase arrow buttons in the **2D Editor** pane.



Tip You can scrub through the signal phases by dragging the timeline bar in the **2D Editor** pane:



Associate a Signal with a Maneuver Road Gate

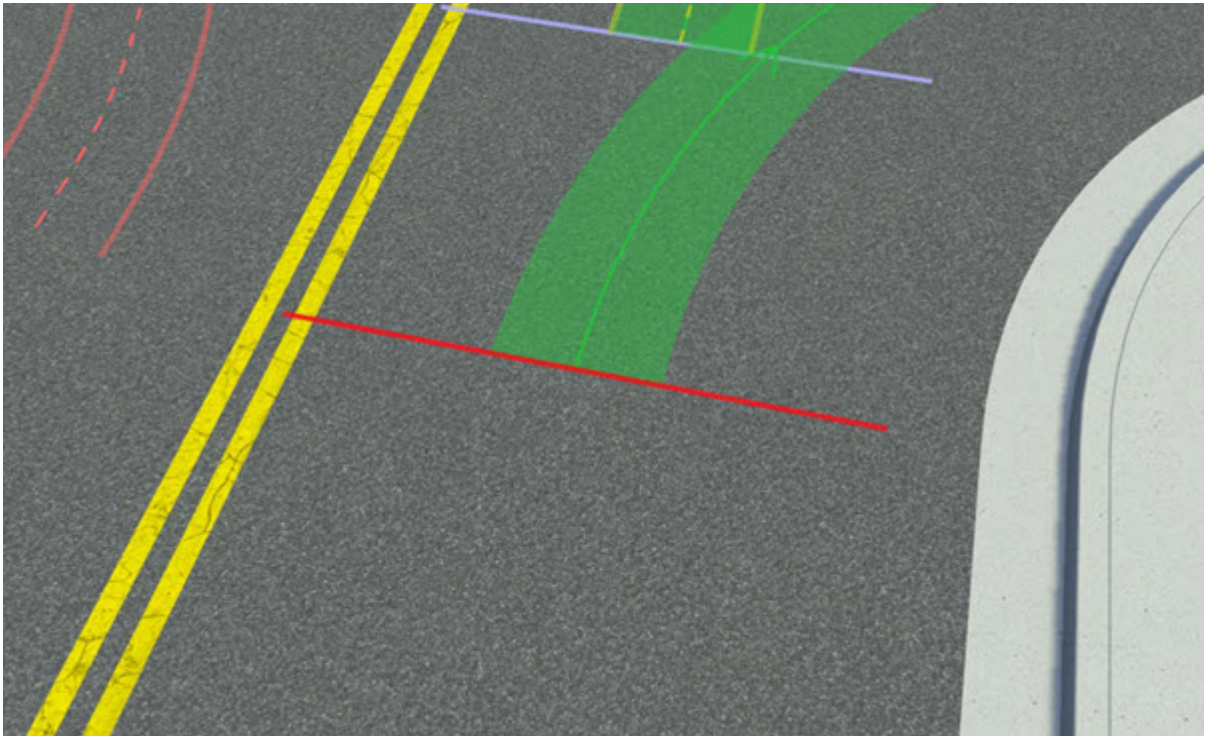


To manually associate a maneuver road with one or more **Signal Assets**, follow these steps.

Note In many cases, the **Auto Detect Signals** operation (see next section) works sufficiently. Try that operation first.

- 1 Click the **Signal Tool** button.
- 2 Select a junction.

- 3 Select a gate. Gates appear as lavender bars on a maneuver road.



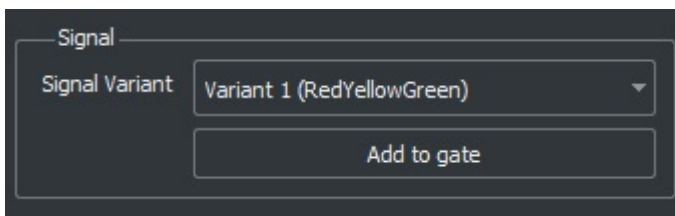
Note Multiple gates might overlap. Cycle-select to choose which gate to edit. For more details, see “Cycle-Select Overlapping Objects”.

- 4 Right-click the signal you want to associate the maneuver with.

Note You cannot associate to signals in **Prop Assembly Assets**. You must first expand the assembly. See Expand a Prop Assembly on page 2-17.

Alternatively:

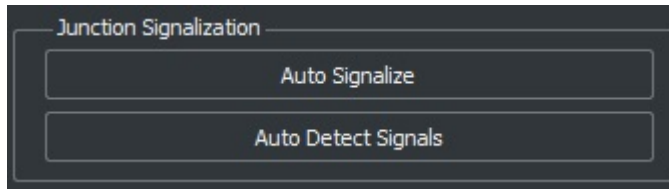
- 1 Click the **Signal Tool** button.
- 2 Select a junction.
- 3 Select a gate. Gates appear as lavender bars on a maneuver road.
- 4 Select a signal.
- 5 Click **Add to gate** in the **Attributes** pane.



Automatically Associate Signals with Maneuver Road Gates

This operation locates signals surrounding an intersection and attempts to automatically associate signals with maneuver road gates.

- 1 Click the **Signal Tool** button.
- 2 Select a junction.
- 3 Click the **Auto Detect Signals** button in the **Attributes** pane. At least one phase must be present in the junction.



This operation uses the maneuver road turn types when determining which signals to associate with. If a maneuver road is being associated to an inappropriate signal, verify that the maneuver road's turn type is correct using the **Maneuver Tool**.

Note Automatic detection does not work for signals in **Prop Assembly Assets**. You must first expand the assembly. See Expand a Prop Assembly on page 2-17.

Remove a Signal Association from a Maneuver Road Gate

- 1 Click the **Signal Tool** button.
- 2 Select a junction.
- 3 Select a gate. Gates appear as lavender bars on a maneuver road.
- 4 Select a signal you want to disassociate.
- 5 Click **Remove from gate** in the **Attributes** pane.



Tips

- If the signals do not automatically detect the correct states, choose the phase, click the signal, and set the desired states. Properly setting the **Supported Turn Types** for the signal can result in better automatic choices.
- Before autosignalizing to automatically place signals, create a traffic light assembly and select it in the **Library Browser**.

Version History

Introduced in R2020a

Topics

“Create Traffic Signals at Junctions”

Sign Tool

Modify custom signs, such as street name signs and freeway billboards

Description

Use the **Sign Tool** to modify custom signs, such as street name signs and freeway billboards. You can partition the sign into rectangular regions and then place text, graphics, and colored rectangles within the sign.





Open the Sign Tool

On the RoadRunner toolbar, click the **Sign Tool** button:



Examples

Place Instance of Sign in Scene

To place a sign in a scene, click and drag the sign asset from the **Library Browser** into the scene. Alternatively, follow these steps:

- 1 Click the **Sign Tool** button.
- 2 Click the desired sign asset.
- 3 Right-click in the scene to place the sign.

Modify Sign Regions

You can create separate regions within a sign and modify each region separately.

Split a Region into Two Regions

- 1 Click the **Sign Tool** button.
- 2 Click the sign you want to edit: either a sign asset in the Asset Browser or an instance of a sign in the 3D scene.

- 3 Click the region you want to split.
- 4 Click the **Add Horizontal Split** or **Add Vertical Split** button.



When you click the button, the region splits at the middle and draws a separator line. To adjust the position of the split, you can click and drag the separator itself to the appropriate position.

Delete a Region

- 1 Click the **Sign Tool** button.
- 2 Click the sign you want to edit.
- 3 Click the region to delete.
- 4 Press **Delete** to remove the region.

Edit the Color and Boundary Properties of a Region

- 1 Click the **Sign Tool** button.
- 2 Click the sign you want to edit.
- 3 Click the region you want to edit.
- 4 Set the **Color** and **Boundary** properties in the **Attributes** pane.

Modify Sign Text

You can add text boxes to signs and edit the text within them.

You can also modify the visual properties of the text and modify the fonts. RoadRunner uses the standard system fonts installed through the operating system. For details on adding new fonts, see “Text Fonts” on page 1-206.

Add New Text Box

- 1 Click the **Sign Tool** button.
- 2 Click the sign you want to edit: either a sign asset in the Asset Browser or an instance of a sign in the 3D scene.
- 3 Click the **Add Text** button to create a new editable text box in the sign.



The text box inherits the properties of the last picked text box and appears centered in the last picked region.

Delete Text Box

- 1 Click the **Sign Tool** button.
- 2 Click the sign you want to edit: either a sign asset in the Asset Browser or an instance of a sign in the 3D scene.
- 3 Click the text box you want to delete.
- 4 Press the **Delete** key, or select **Edit > Delete** from the menu bar.

Edit Text in Text Box

- 1 Click the **Sign Tool** button.
- 2 Click the sign you want to edit: either a sign asset in the Asset Browser or an instance of a sign in the 3D scene.
- 3 Click the text box you want to edit.
- 4 Select the **Text** attribute on the **Attributes** pane, and type in the desired string.

Change Font or Other Visual Properties of Text Box

- 1 Click the **Sign Tool** button.
- 2 Click the sign you want to edit: either a sign asset in the Asset Browser or an instance of a sign in the 3D scene.
- 3 Click the text box you want to edit. The attributes of the text box appear in the **Attributes** pane.
- 4 Edit the desired attributes on the **Attributes** pane. The **Small Caps** option can be used to format lowercase letters as smaller capital letters

Modify Sign Graphics

You can add bitmap or vector graphic to signs and modify the shape, color, or visual properties of these graphics.

Add New Sign Graphic

- 1 Click the **Sign Tool** button.
- 2 Click the sign you want to edit: either a sign asset in the Asset Browser or an instance of a sign in the 3D scene.
- 3 Click and drag a graphic onto the sign.

Delete Sign Graphic

- 1 Click the **Sign Tool** button.
- 2 Click the sign you want to edit: either a sign asset in the Asset Browser or an instance of a sign in the 3D scene.
- 3 Click the graphic you want to delete.

- 4 Press the **Delete** key or select **Edit > Delete** from the menu bar.

Edit Shape of Graphic

- 1 Click the **Sign Tool** button.
- 2 Click the sign you want to edit: either a sign asset in the Asset Browser or an instance of a sign in the 3D scene.
- 3 Click the graphic you want to edit.
- 4 Click and drag the edges of the graphic. The pointer changes to the resize symbol.

Change Color or Other Visual Properties of Graphic

- 1 Click the **Sign Tool** button.
- 2 Click the sign you want to edit: either a sign asset in the Asset Browser or an instance of a sign in the 3D scene.
- 3 Click the graphic you want to edit. The attributes of the graphic appear in the **Attributes** pane.
- 4 Edit the desired attributes in the **Attributes** pane.

Modify Sign Rectangles

You can add rectangles to within signs and modify the shape, color, and other visual properties of the rectangle.

Add New Sign Rectangle

- 1 Click the **Sign Tool** button.
- 2 Click the sign you want to edit: either a sign asset in the Asset Browser or an instance of a sign in the 3D scene.
- 3 Click the **Add Rectangle** button on the toolbar on the left.



The new rectangle inherits the attributes of the last rectangle selected or has default values if no rectangle has been selected.

Delete Sign Rectangle

- 1 Click the **Sign Tool** button.
- 2 Click the sign you want to edit: either a sign asset in the Asset Browser or an instance of a sign in the 3D scene.
- 3 Click the rectangle you want to delete.
- 4 Press the **Delete** key or select **Edit > Delete** from the menu bar.

Edit Shape of Rectangle

- 1 Click the **Sign Tool** button.
- 2 Click the sign you want to edit: either a sign asset in the Asset Browser or an instance of a sign in the 3D scene.

- 3 Click the rectangle you want to edit.
- 4 Click and drag the edges of the rectangle. The pointer changes to the resize symbol.

Change Color or Other Visual Properties of Rectangle

- 1 Click the **Sign Tool** button.
- 2 Click the sign you want to edit: either a sign asset in the Asset Browser or an instance of a sign in the 3D scene.
- 3 Click the rectangle you want to edit. The attributes of the rectangle appear in the **Attributes** pane.
- 4 Edit the desired attributes in the **Attributes** pane.

More About

Text Fonts

RoadRunner uses the standard system fonts installed through the operating system. You can specify the font for each text box individually. Fonts for street signs in different countries are available from various sources online. For example, Roadgeek 2005 is a good resource of sign fonts.

In the United States, most freeway and street signs are based on a set of fonts known as Highway Gothic, formerly known as the FHWA Series fonts. These fonts were originally published in 1948 as the FHWA's Standard Alphabets for Traffic-Control Devices and have been updated several times. The original series had variants ranging from Series A (narrow) to Series F (wide). Series A is no longer used in the U.S. because it was too narrow. Series E has a wider variant known as Series E-Modified or sometimes Series E(M), which is used on many freeway guide signs. Highway Gothic is used in several other countries as well.

Several U.S. states have adopted a newer font called Clearview, which was developed to improve readability over the Highway Gothic font. There are different sets of Clearview fonts for light letters on dark backgrounds, and dark letters on light backgrounds. There are six sizes in each set that vary from narrow to wide. This leads to at least 12 variations on the font, usually labeled 1B-6B, and 1W-6W, plus a 13th revised version of 5W called 5WR.

Version History

Introduced in R2020a

Slip Road Tool

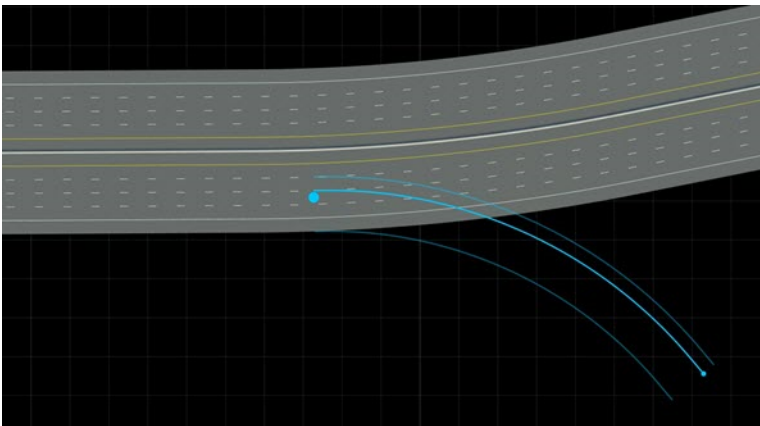
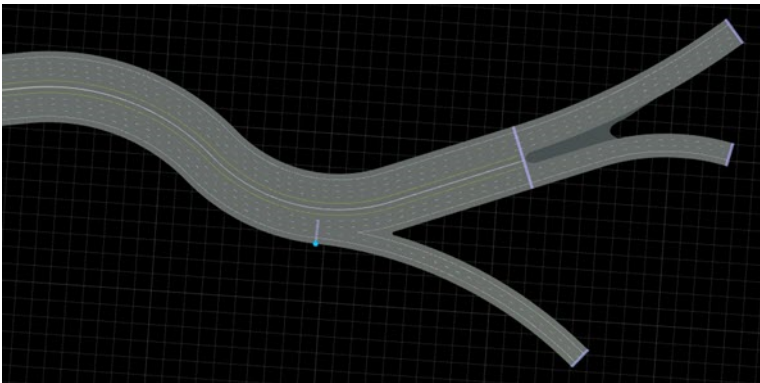
Create onramps, offramps, and road splits

Description

The **Slip Road Tool** is used to create onramps, offramps, and road splits.

When using the **Slip Road Tool**, a light blue circle follows the pointer. This circle snaps to lanes and the lavender road node lines at the ends of roads.

Different actions occur depending on which objects you select, the locations where you right-click, and whether you are pressing the **Shift** key.



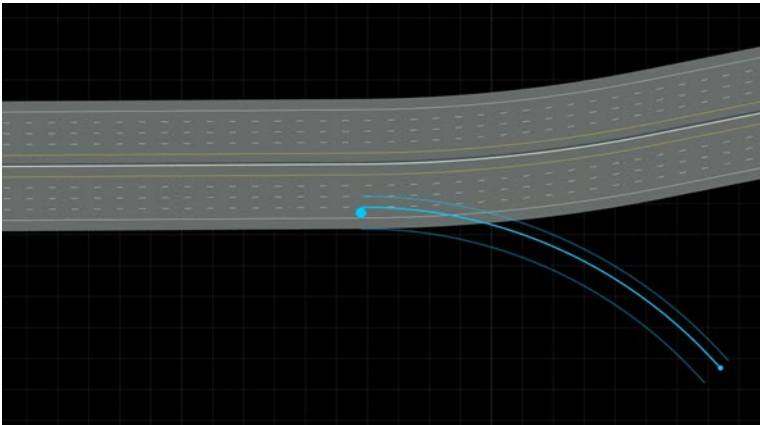
Open the Slip Road Tool

On the RoadRunner toolbar, click the **Slip Road Tool** button:



Examples

Create a Single-Lane Onramp or Offramp

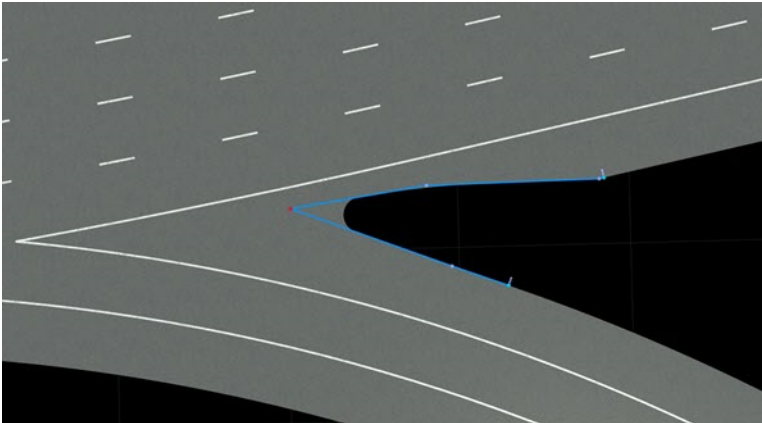


- 1 Click the **Slip Road Tool** button.
- 2 Move the pointer over the outermost driving lane of a road. The light blue circle gets slightly larger.
- 3 Right-click and drag to show a preview of the slip road to create. This action forms either an onramp or offramp, depending on whether you are dragging ahead of the starting point or behind the starting point.
- 4 Release the right-click button to create the road.

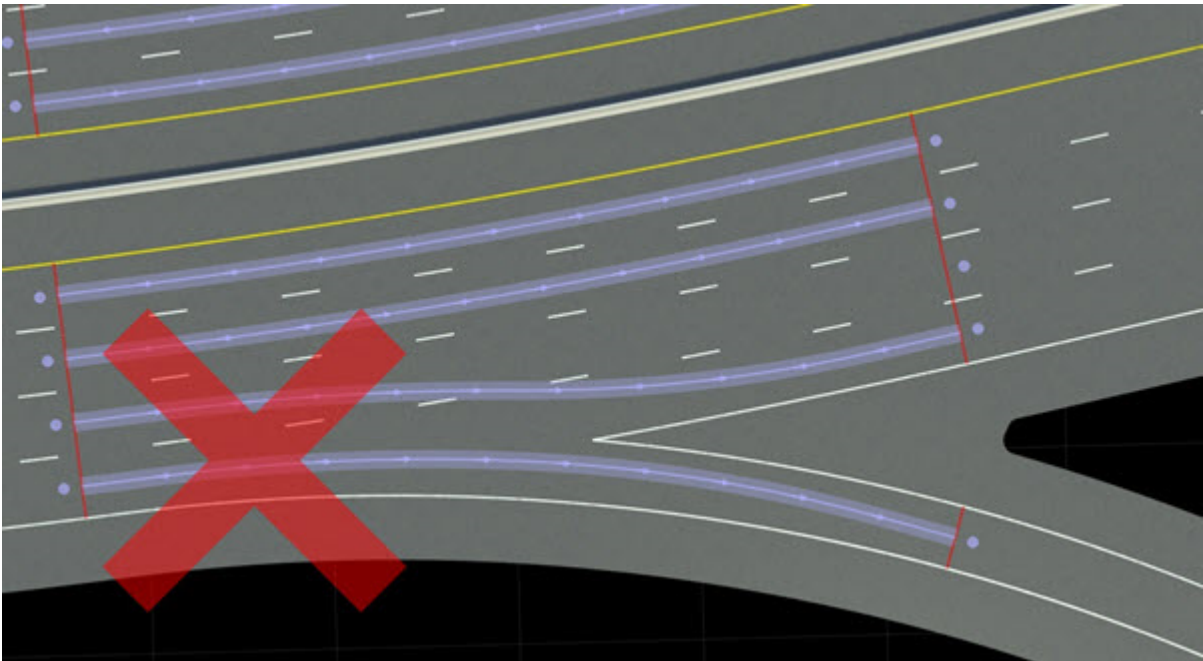
Tip If you release the right-click button on another road, the new road is a slip road at both ends. For example, it could be an offramp for the first road and an onramp for the second.

Junctions are automatically created whenever multiple roads overlap, including in slip road cases. Refer to “Junctions and Traffic Signals” for more information on working with junctions.

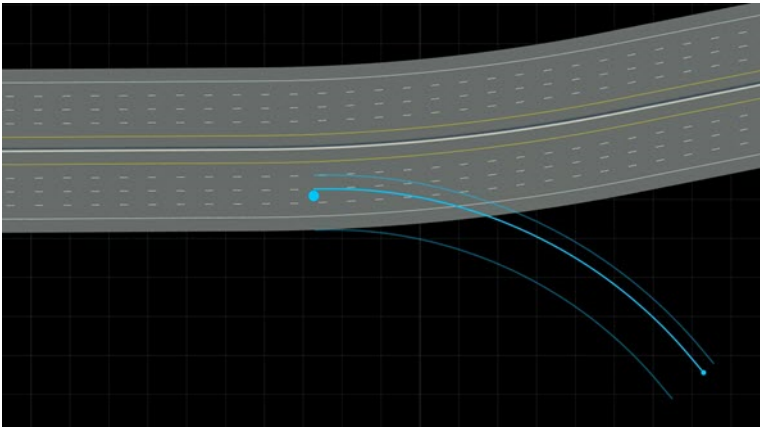
For example, you can use the **Corner Tool** to adjust the geometry of the curve between the ramp and the main road:



Junctions involving slip roads still contain maneuver roads (see **Maneuver Tool**). The automatically-created maneuver roads are not guaranteed to match the expected connectivity. It is recommended to double-check maneuver road topology in slip road junctions as you would do with an at-grade intersection. This is important to ensure that the lane connectivity is correct when exporting to a format like ASAM OpenDRIVE.



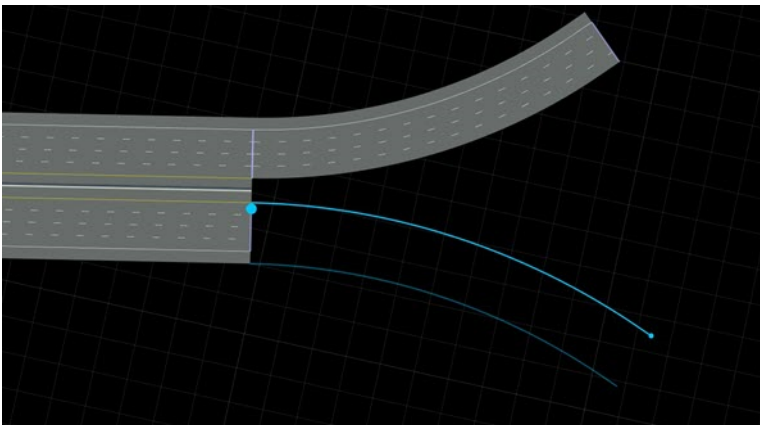
Create a Multilane Onramp / Offramp



- 1 Click the **Slip Road Tool** button.
- 2 Move the pointer over the innermost driving lane that you want to connect to the ramp. The light blue circle gets slightly larger. This lane and all lanes up to the edge of the road will connect to the ramp.
- 3 Right-click and drag to show a preview of the slip road to create. This action forms either an onramp or offramp, depending on whether you drag ahead of the starting point or behind the starting point.
- 4 Release the right-click button to create the road.

Note By default, the driving lanes between the selected lane and the edge of the road end at the new junction. That is, in an offramp case, these lanes are dedicated offramp lanes that do not continue along the main road. If you hold the **Shift** key when you release the right-click button, this behavior is disabled and all driving lanes continue along the main road.

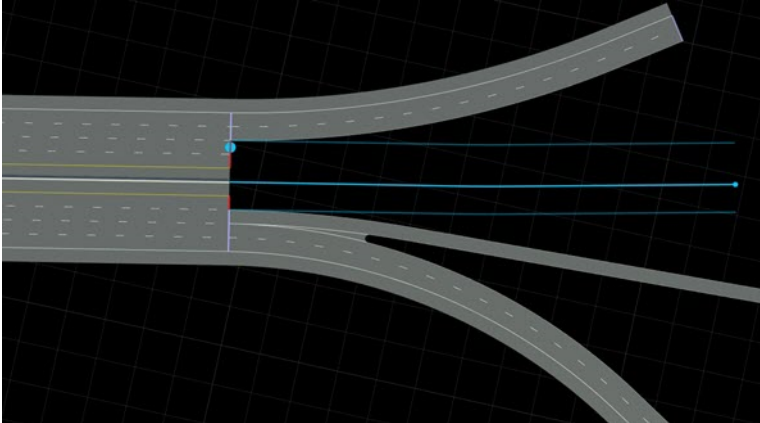
Create a Road Split



You can split a road into two roads by creating slip roads at the end of a road.

- 1 Click the **Slip Road Tool** button.

- 2 On the lavender road node line at the end of a road, move the pointer over the innermost driving lane that you want to connect to the new slip road. The new slip road will include this lane and all lanes up to the edge of the road.
- 3 Right-click and drag to show a preview of the slip road.
- 4 Release the right-click button to create the road.



You can optionally provide more fine-grained control over the lanes to be included in the slip road. This is useful for creating more advanced types of road splits.

Before following the steps above, select a range of lanes on the lavender road node line. The created slip road will connect to the range of lanes defined by the innermost and outermost selected lanes.

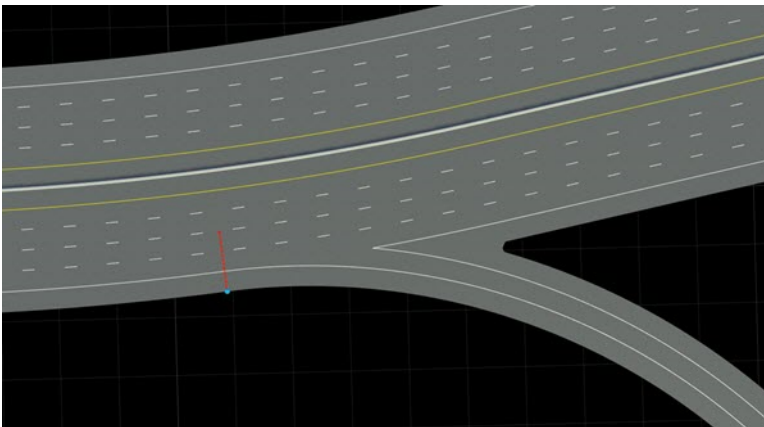
Create a Channelized Turn Lane

You can also use the **Slip Road Tool** to create physically separated, branching lanes in any situation.

For example, you can create a channelized right turn lane on an at-grade intersection. Follow the steps to create a single lane on page 1-208 or multilane on page 1-210 ramp on a road leading up to an intersection. If you end your drag on the crossing road, it will create a channelized turn lane.



Adjust Where a Slip Road Is Connected



You can adjust where a slip road is attached to the road at its ends:

- 1** Select the **Road Plan Tool**.
- 2** Click and drag the dashed lavender road node at the end of a slip road.

Version History

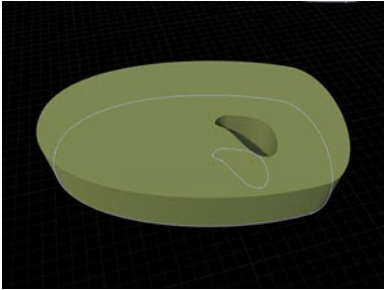
Introduced in R2020a

Surface Tool

Model surfaces around roads, such as walkways, driveways, parking lots, and natural terrain

Description

The **Surface Tool** models surfaces around roads, such as walkways, driveways, parking lots, and natural terrain. For more details on surfaces, see “How Surfaces Work in RoadRunner”.



Open the Surface Tool

On the RoadRunner toolbar, click the **Surface Tool** button:



Examples

Edit Terrain Surface Curves and Regions

See “Region Graph Editing”.

Insert a Terrain Surface Node Along a Road Boundary

- 1 Click the **Surface Tool** button.
- 2 Right-click a terrain surface curve along a road boundary.

Note You can slide these types of nodes along the road using click and drag.

Change the Material Assigned to a Terrain Surface

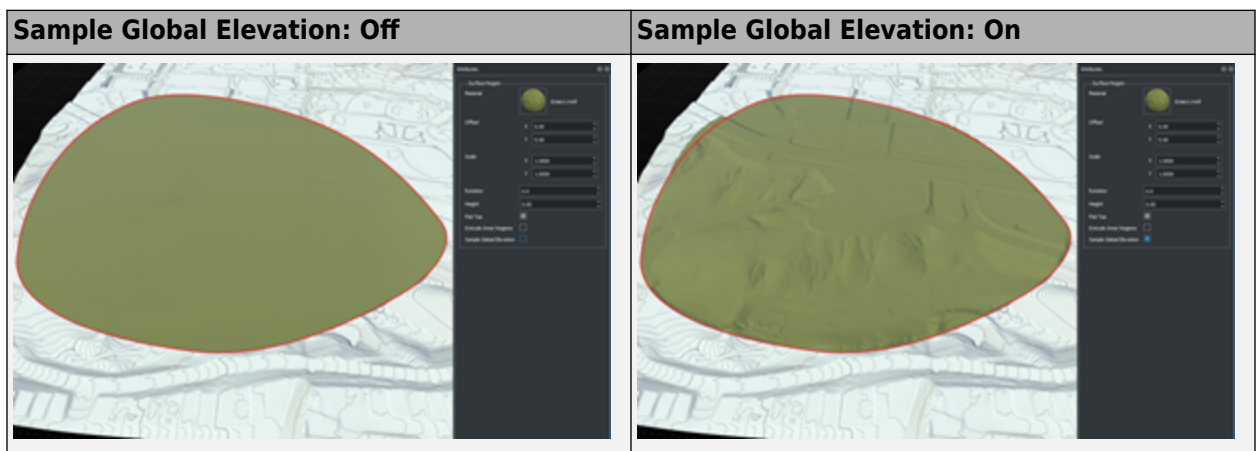
- 1 Click the **Surface Tool** button.
- 2 Select a terrain surface.
- 3 Assign one of the **Material Assets** to the **Material** widget in the **Attributes** pane.

Alternatively, click and drag one of the **Material Assets** from the **Library Browser** onto a terrain surface. This operation can be performed in any tool.

Control Whether a Surface Uses Elevation Samples

By default, the interior of a surface is smoothly interpolated from its boundaries. Surfaces can optionally use the elevation maps in the scene to determine the heights of interior points as follows:

- 1 Configure the elevation maps through the **Elevation Map Tool** as desired, using the previous instructions.
- 2 Click the **Surface Tool** button.
- 3 Select a surface.
- 4 Enable the **Sample Global Elevation** option in the **Attributes** pane.

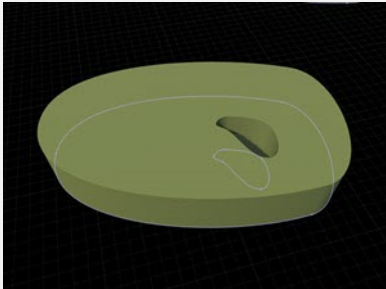


Note This setting affects only the interior of the surface. The heights along the perimeter of the surface are still defined by the surface curves, which are independent from the elevation map.

You can better align the perimeter of a surface to the elevation map by first inserting additional nodes along the perimeter of the surface (where needed). Then, project the nodes to the surface using the **Project Nodes** button in the toolbar on the left.

Parameters

Attribute	Description
Material	The Material Assets to apply to the surface.
Offset	An XY offset to apply to the texture coordinates of the surface.
Scale	An XY scale to apply to the texture coordinates of the surface.
Rotation	An additional rotation to apply to the texture coordinates of the surface.
Height	If nonzero, the surface is extruded upward by this amount.

Attribute	Description
Flat Top	<p>If a terrain surface's curves are not flat, then this option forces the extruded top of the surface to be flat.</p> <p>This attribute affects only surfaces with nonzero Height values.</p>
Extrude Inner Regions	<p>If true, any enclosed surfaces form either holes or raised areas (depending on the elevation of their surface curves). If false, the outer surface slopes downwards (or upwards) to meet the height of the enclosed surface.</p> <p>Only impacts surfaces with nonzero Height values.</p> 

Version History

Introduced in R2020a

See Also

"How Surfaces Work in RoadRunner"

Traffic Island Tool

Create freeform traffic islands in RoadRunner scene

Description

The **Traffic Island Tool** enables you to create traffic islands in RoadRunner scenes without using road lanes and junction corners. Using the **Traffic Island Tool**, you can:

- Create traffic islands in a RoadRunner scene. Drag control points to modify the shape of the traffic island. Traffic islands in RoadRunner support driveways and sloped curbs.
- Modify the border of the traffic island by specifying attributes such as border style, border width, and the material of the border.
- Adjust the height of each node on the traffic island.
- Add an inner shape to the traffic island. This figure shows a traffic island that has an inner rounded oblong shape covered by grass.

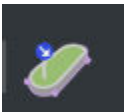


The **Traffic Island Tool** supports accurate importing and exporting of traffic islands to and from the ASAM OpenDRIVE format, offering extended compatibility with ASAM OpenDRIVE versions 1.5 and 1.6.



Open the Traffic Island Tool

On the RoadRunner toolbar, click the **Traffic Island Tool** button:

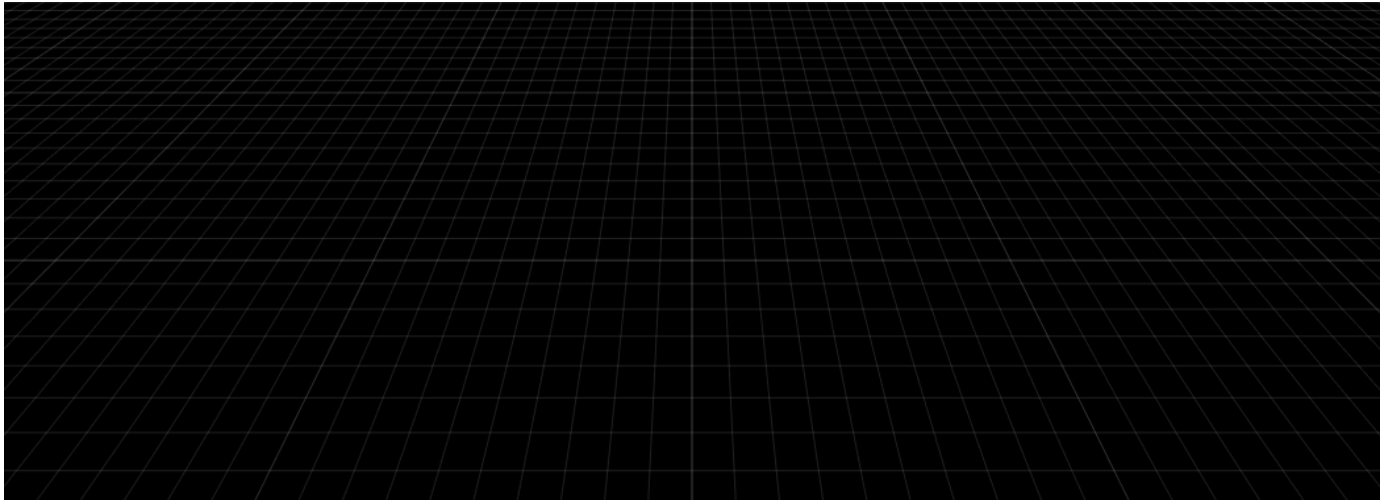


Examples

Create Custom Traffic Island

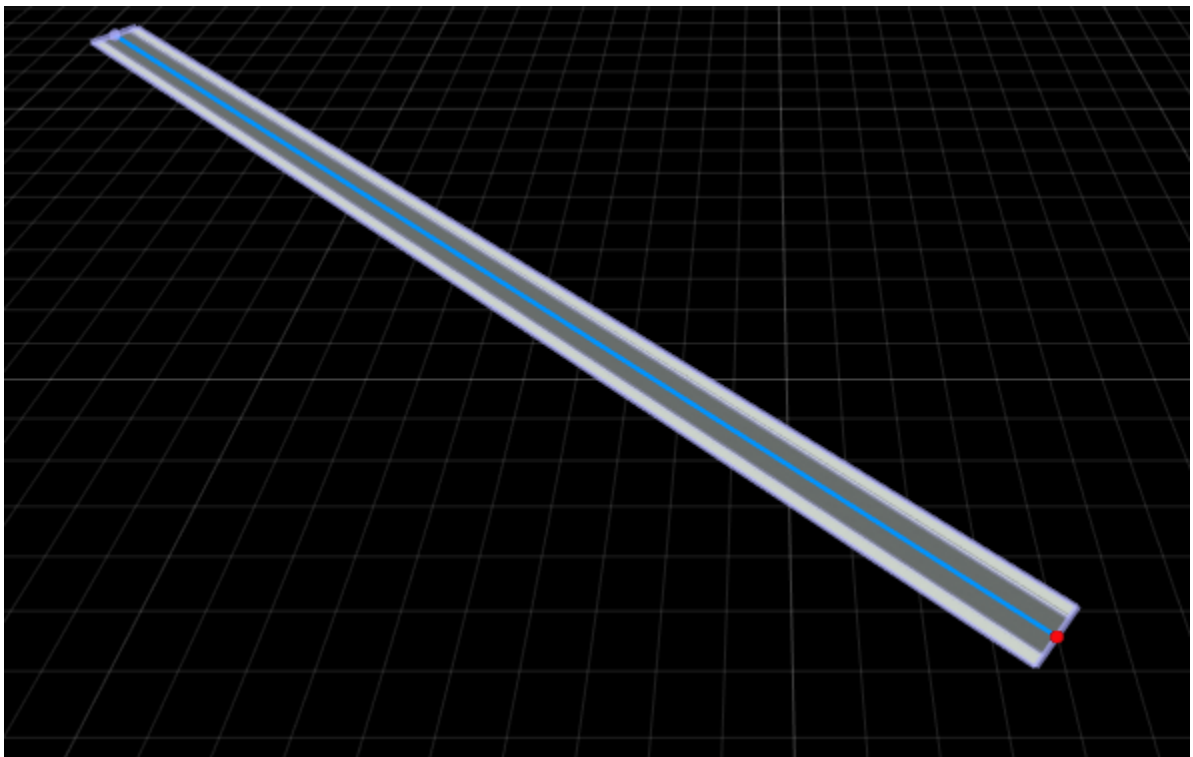
Create a traffic island in an existing RoadRunner scene and add an interior shape to it.

Open a new scene. From the **File** menu, select **New Scene**. This opens a blank scene in the RoadRunner editor.

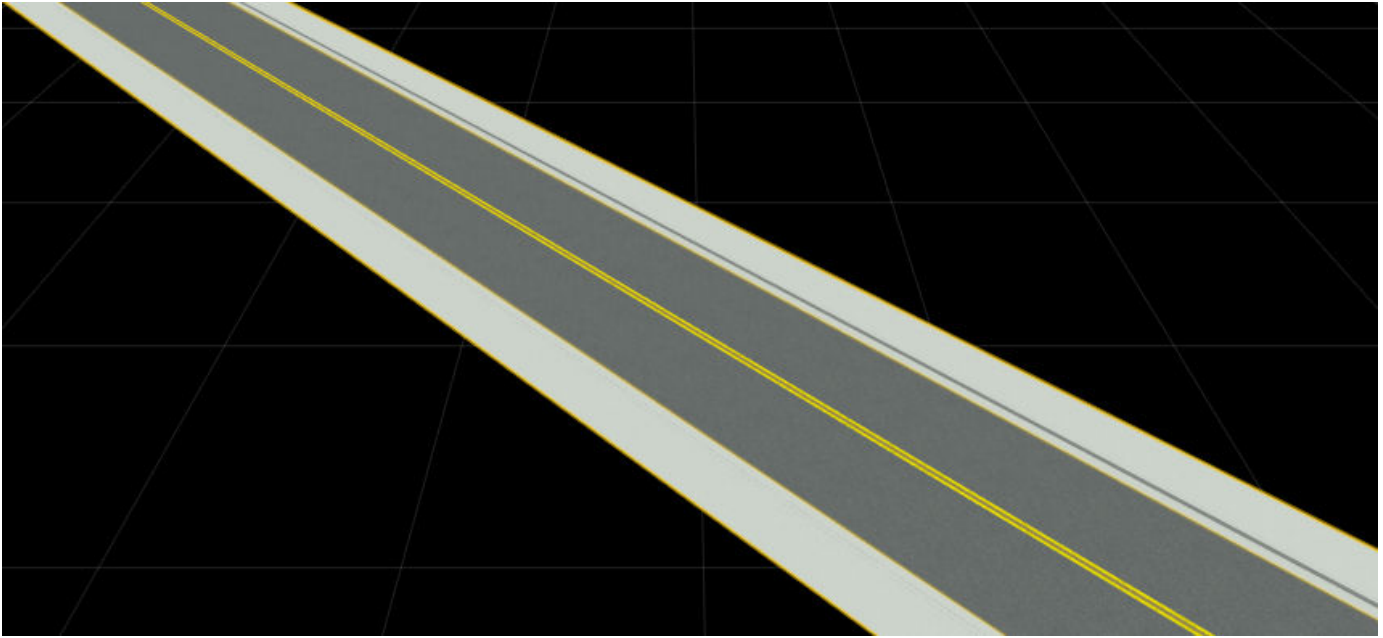


From the RoadRunner toolbar, click the **Road Plan Tool** button.

Right-click the location at which you want to start a new road. Right-click additional locations to create additional road control points that extend and shape the road.



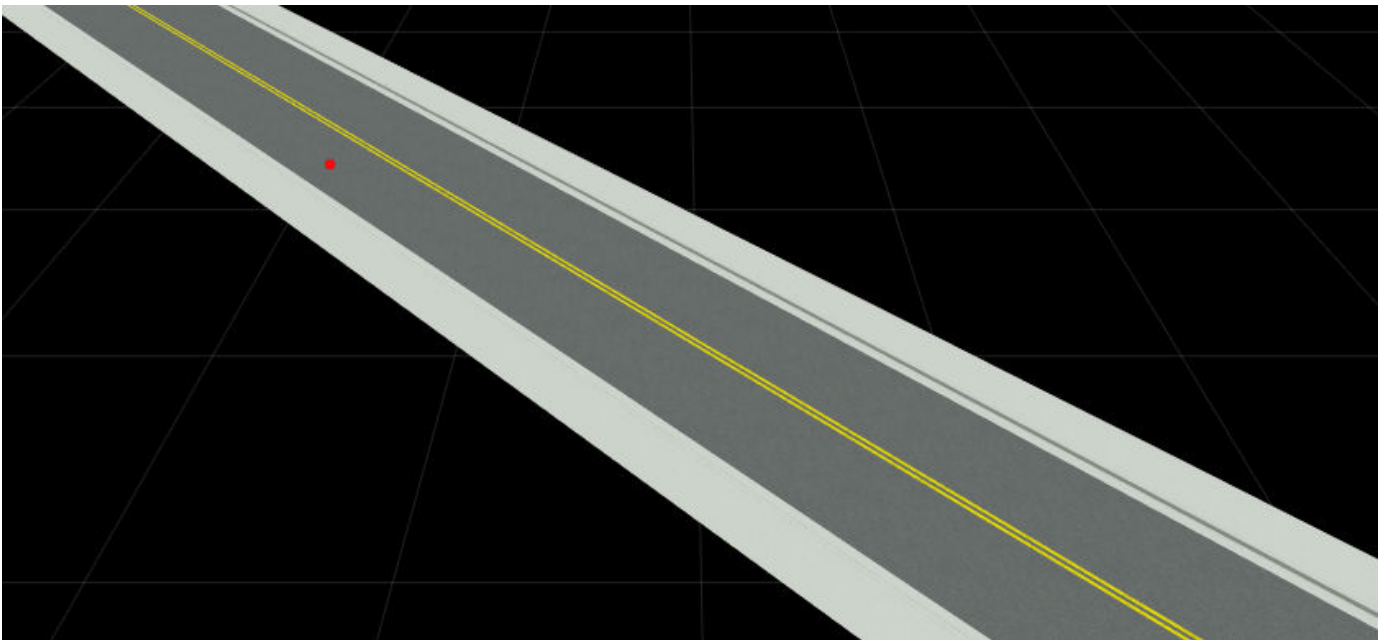
Zoom in on the location where you want to insert a traffic island on the road.



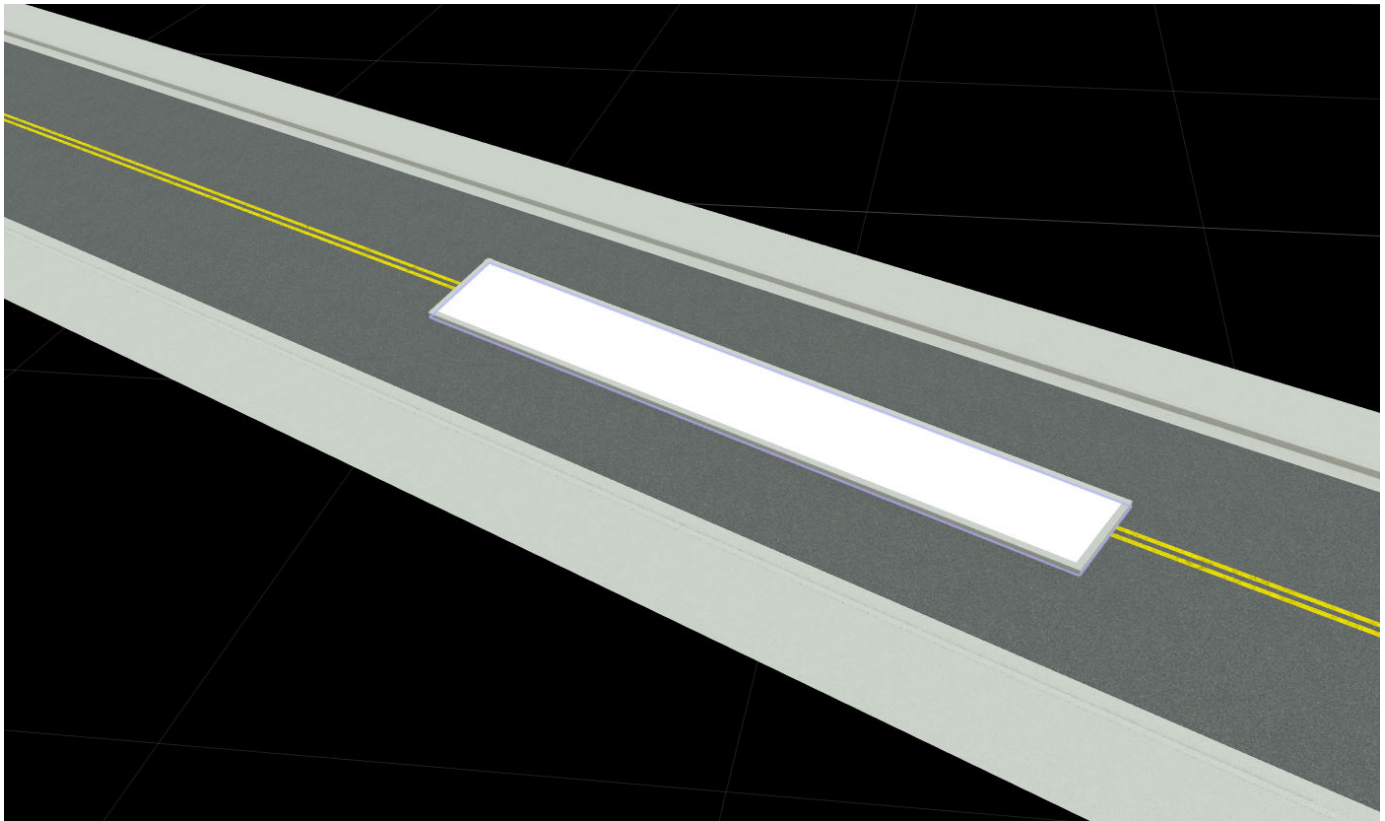
On the RoadRunner toolbar, click the **Traffic Island Tool** button:



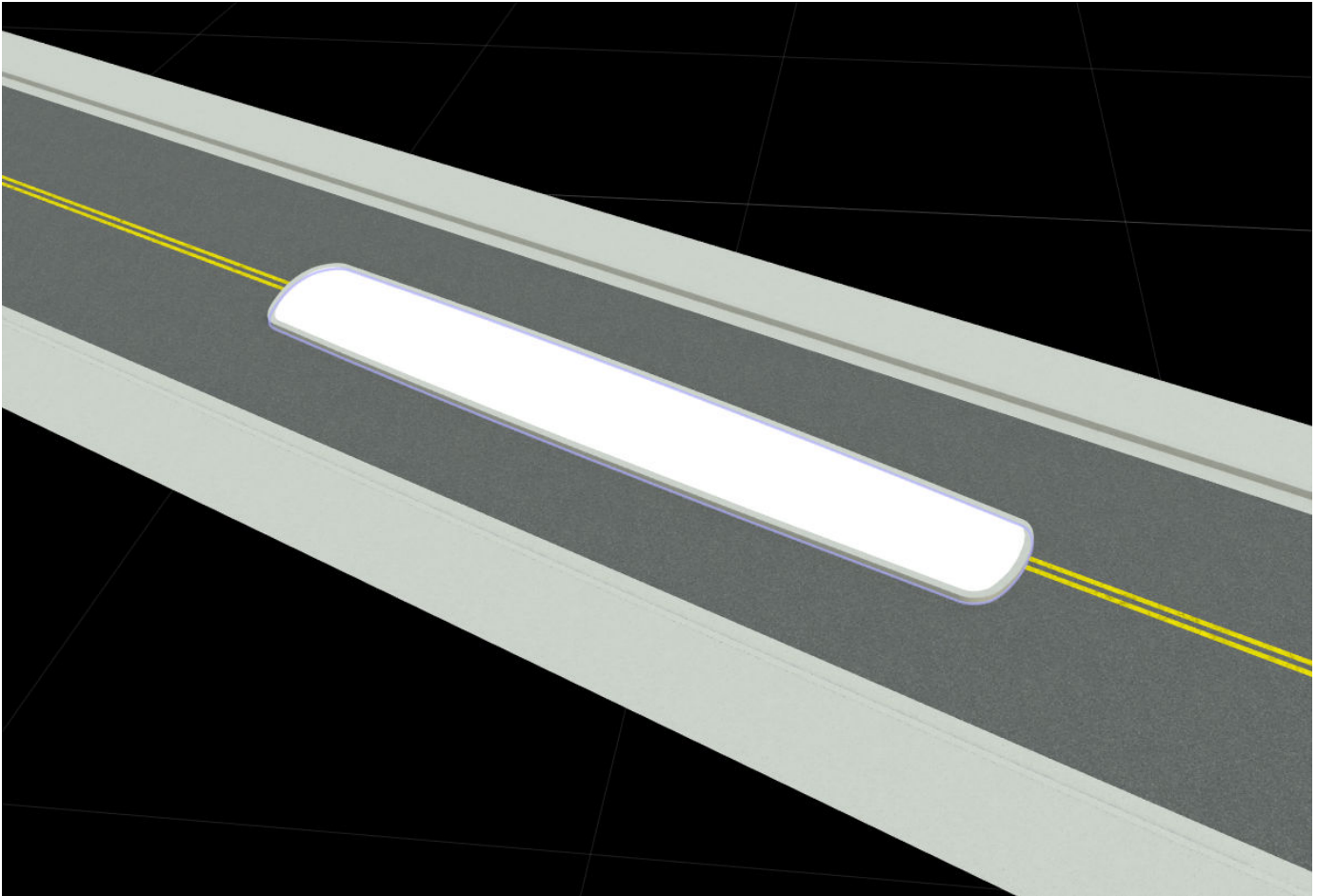
Right-click the road where you want to insert a control point for the traffic island.



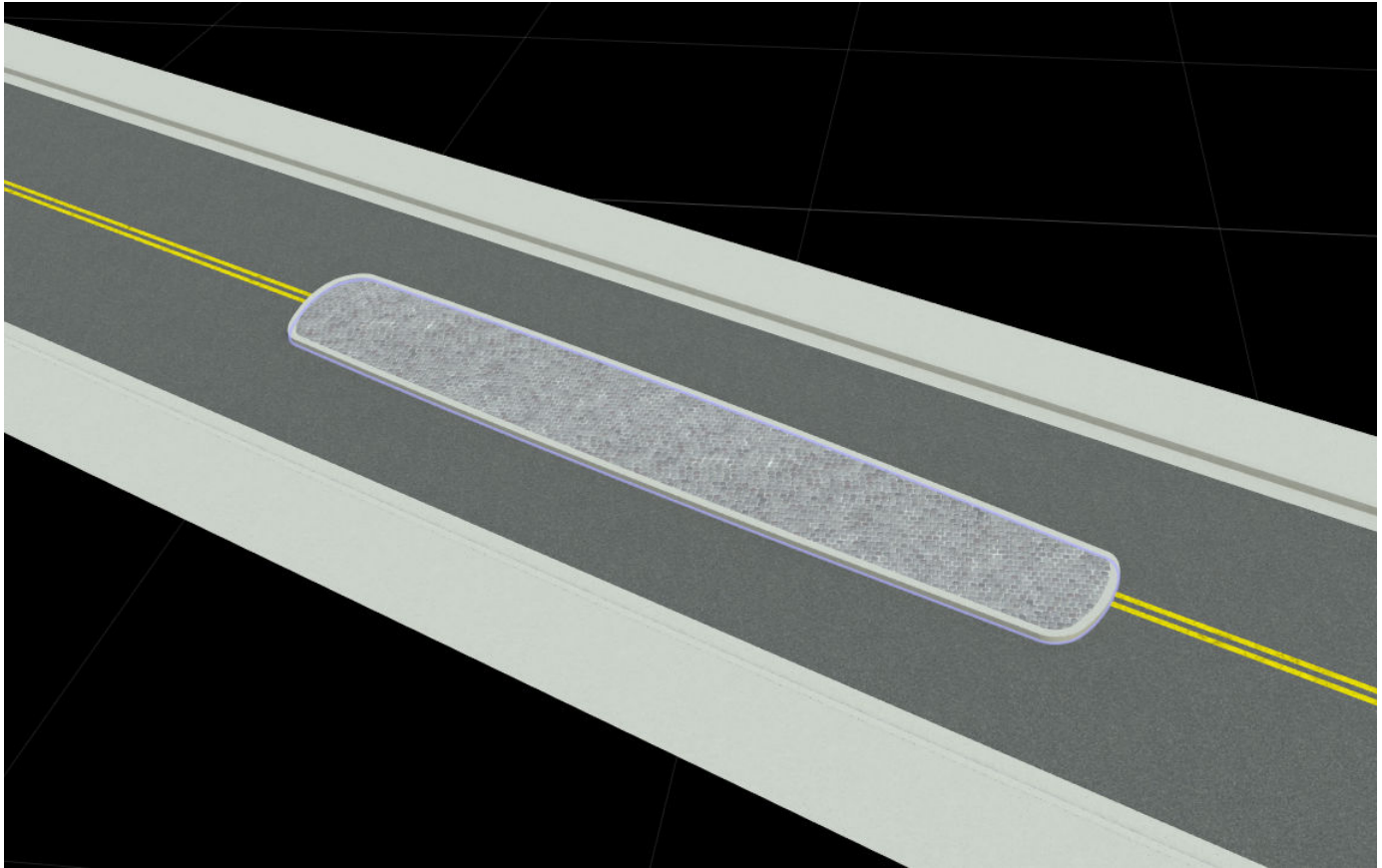
Right-click on the road to insert three additional control points in the form of a rectangle. This creates a rectangular traffic island with sharp corners on the road.



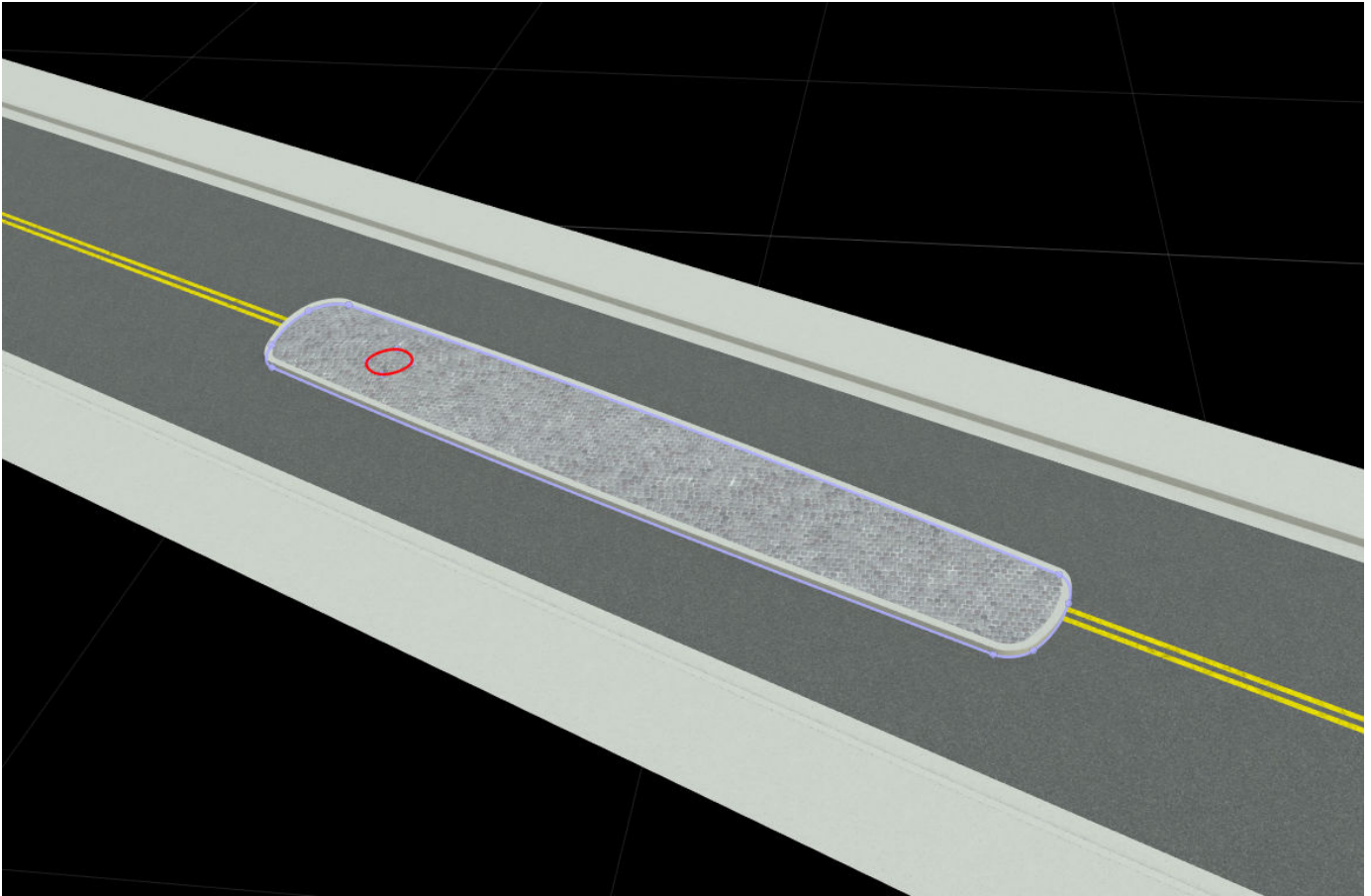
To smooth the sharp corners, first, right-click the corners of the traffic island to insert additional control points. Then, drag the control points towards the interior of the traffic island to adjust the shape of the corners. This creates a traffic island with smooth, rounded corners.



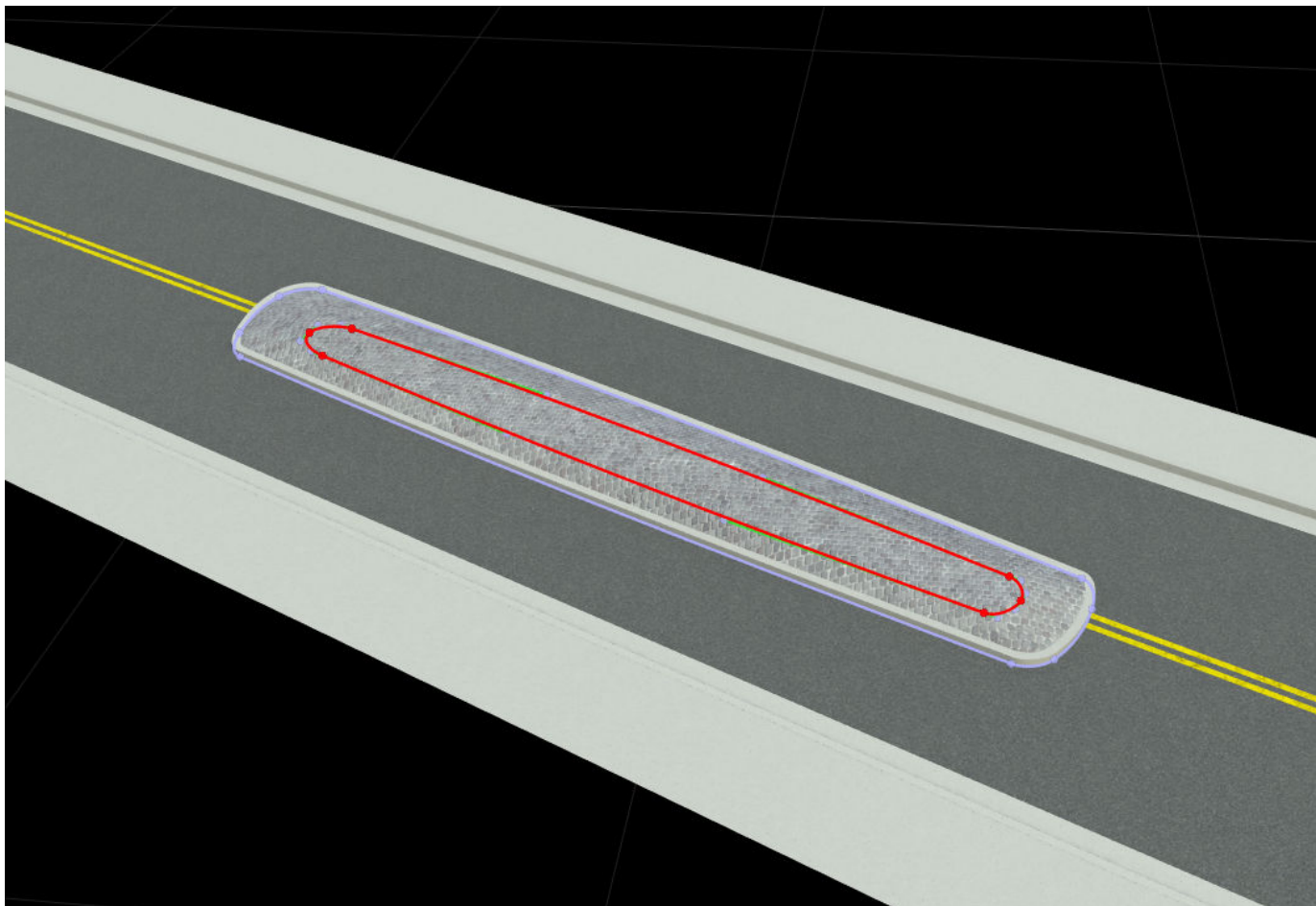
Modify the material of the traffic island. In the **Library Browser**, select the **Assets** folder and navigate to the **Materials** folder. Select the cobblestone material and drag it to the traffic island in the editing canvas.



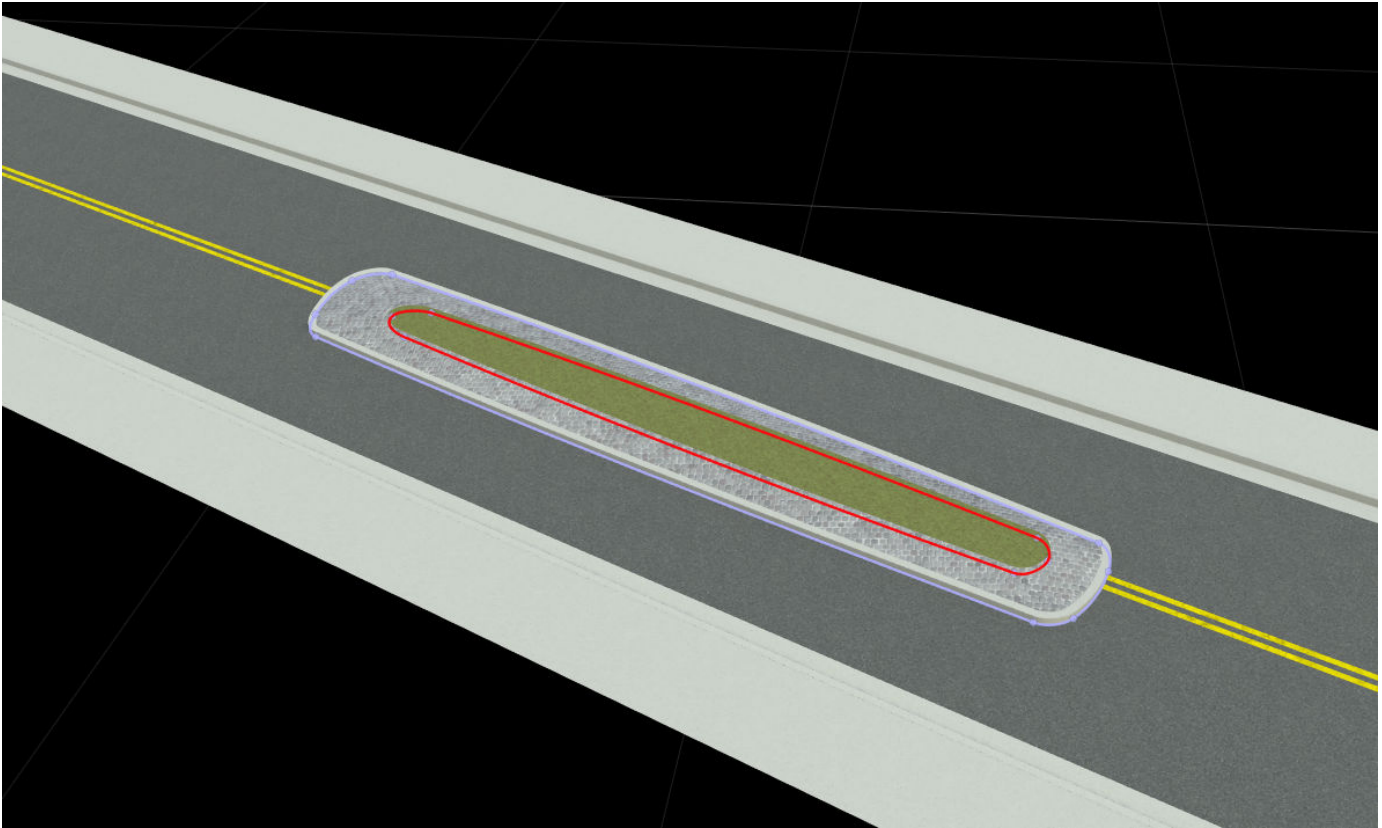
Optionally, you can add an interior shape to the traffic island. Right-click inside the traffic island to insert an interior shape.



Drag the control points to adjust the interior shape of the traffic island.



Modify the material of the interior shape of the traffic island. In the **Library Browser**, select the **Assets** folder and navigate to the **Materials** folder. Select the grass material and drag it to the interior shape of the traffic island in the editing canvas.



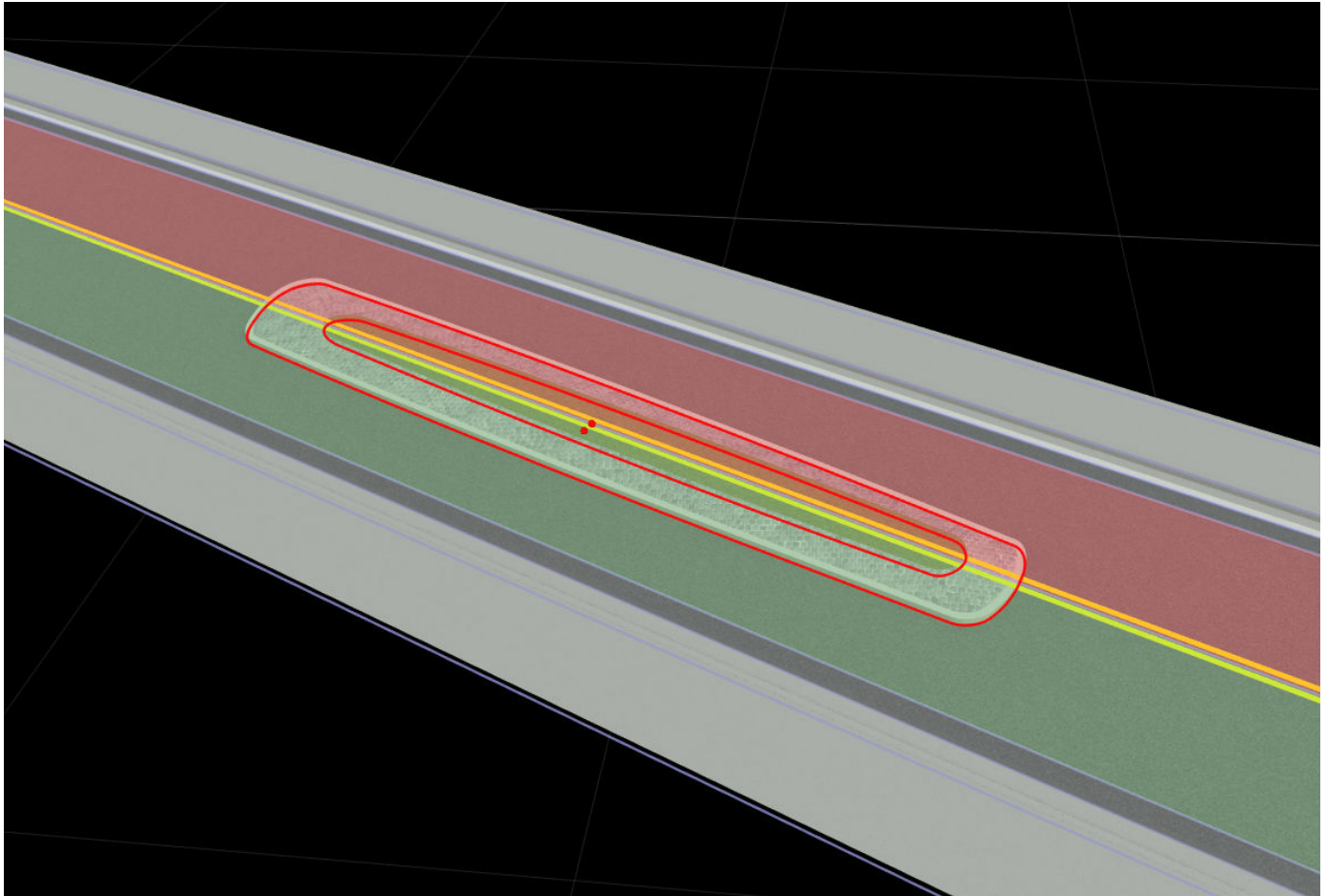
Optionally, you can adjust the properties of the traffic island, such as **Texture Scale** and **Texture Rotation**, from the **Attributes** pane.

The control points of the traffic island are floating above the road surface and are not attached to it.

To adjust this, you must click the **Project Control Points** button  on the left navigation toolbar. This attaches the traffic island to the road surface.

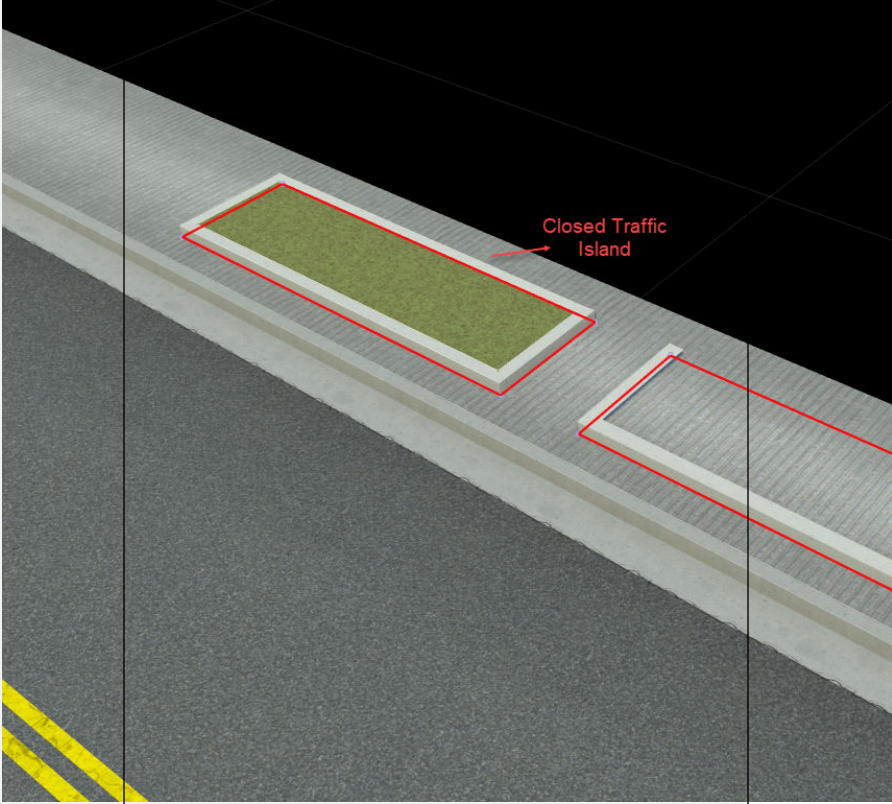
To visualize how the traffic island looks when being exported to the ASAM OpenDRIVE format, click

the **OpenDRIVE Export Preview Tool** button  on the RoadRunner toolbar. In the Export ASAM OpenDRIVE dialog box, specify **Version** as ASAM OpenDRIVE 1.6 and click **Export**. This figure shows the visualization of the traffic island when exported to the ASAM OpenDRIVE format.



Parameters

Traffic islands added to the scene have these attributes.

Attribute	Description
<p>Closed</p>	<p>Specifies if the traffic island has an interior surface. A closed traffic island contains an interior surface that supports a material. A nonclosed traffic island supports only edges or curbs. By default, traffic islands in RoadRunner are closed. Clear the Closed attribute to create an empty traffic island. For example, this figure shows a closed and an empty, nonclosed traffic island.</p> 
<p>Material</p>	<p>Material Assets to use for the traffic island.</p>
<p>Texture Scale</p>	<p>Scale to apply to each dimension of the texture coordinates for the material.</p>
<p>Texture Rotation</p>	<p>If nonzero, specifies an additional rotation to apply to the texture coordinates of the material.</p>

Traffic islands added to the scene are surrounded by a border on all sides. The traffic island border is specified as a sequence of control points and has these attributes.

Attribute	Description
<p>Border Style</p>	<p>Style of the border for the traffic island, specified as one of these options: None, Side, Curb or Concrete.</p>

Attribute	Description
Width	Width of the traffic island border, in meters, specified as a real scalar. The width can be uniform for the entire border, or can vary for different parts of the border of the traffic island.
Material	Material Assets to use for the traffic island border.
Texture Scale	Scale to apply to each dimension of the texture coordinates of the material.
Texture Rotation	If nonzero, specifies an additional rotation to apply to the texture coordinates of the material.

Limitations

- The traffic island in a RoadRunner scene does not automatically stitch onto the road geometry, and can float or sink into the road surface. To adjust this, you must click the **Project Control Points**



button on the left navigation toolbar. This attaches the traffic island to the road surface.

- Modifying road or lane curves does not automatically adjust a traffic island attached to that road or lane. If you modify a road or a lane curve, then you must also modify the traffic island.
- RoadRunner does not clip the interior polygons of a traffic island when exporting a scene. If you want to clip the interior polygons, you must do so manually.

Version History

Introduced in R2022b

See Also

Marking Curve Tool | **Marking Polygon Tool** | **Prop Polygon Tool**

Vector Data Tool

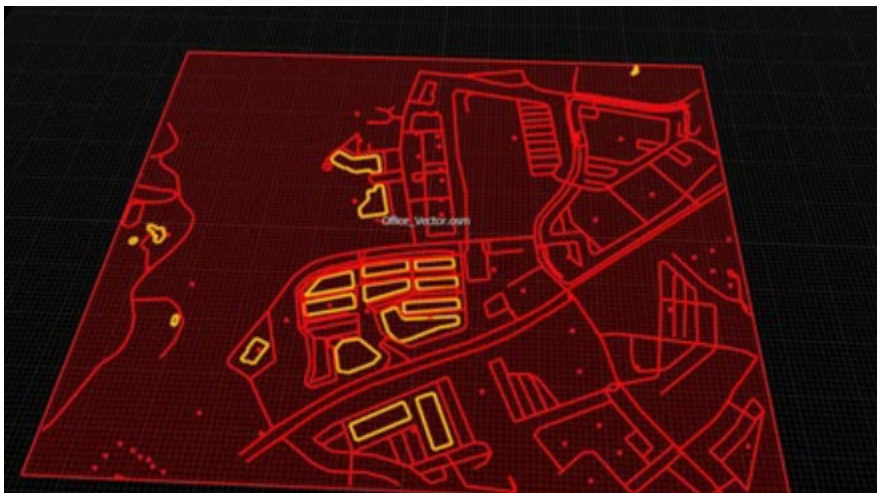
Manage import and configuration of vector data files and explore shape attributes

Description

The **Vector Data Tool** manages the import and configuration of vector data files, and enables exploration of the shape attributes. RoadRunner can load a variety of georeferenced vector map files, such as Shape files (.shp), OpenStreetMap files (.osm), GeoJSON files (.json, .geojson), and GPS Exchange files (.gpx). Vector data such as points, lines, and polygons can be loaded from these files along with their associated attributes.

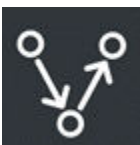
Refer to the **Vector Data Assets** page for a list of supported formats.

Note The **Vector Data Tool** is used to visualize and explore vector file types for use as visual references. Automatic conversion of vector map data into the RoadRunner internal format is not supported.



Open the Vector Data Tool

On the RoadRunner toolbar, click the **Vector Data Tool** button:



Examples

Import a Georeferenced Vector Map

- 1 Click the **Vector Data Tool** button.
- 2 In the **Library Browser**, navigate to the directory containing the vector data asset you want to import. For more details, see **Vector Data Assets**.
- 3 Click and drag the asset from the **Library Browser** into the 3D scene.

Note If the geographic position has not yet been set for this scene, the scene center is set to the latitudinal and longitudinal center of the image. You can change the scene center using the **World Settings Tool**.

If the geographic position has already been set, but the imported image is outside of the maximum range of the scene, an error dialog box appears and cancels the import.

Remove a Vector Map

- 1 Click the **Vector Data Tool** button.
- 2 Click within the bounding box of the vector map you want to delete. Do not click the vector element itself. Instead, click an empty space within the bounding box. When the map is clicked, the bounding box turns red.
- 3 Press the **Delete** key or select **Edit > Delete** from the menu bar.

Adjust the Properties of a Vector Map

- 1 Click the **Vector Data Tool** button.
- 2 Click within the bounding box of the vector map you want to adjust. Do not click the vector element itself. Instead, click an empty space within the bounding box. When the map is clicked, the bounding box turns red.
- 3 Adjust any properties as desired on the **Attributes** pane.

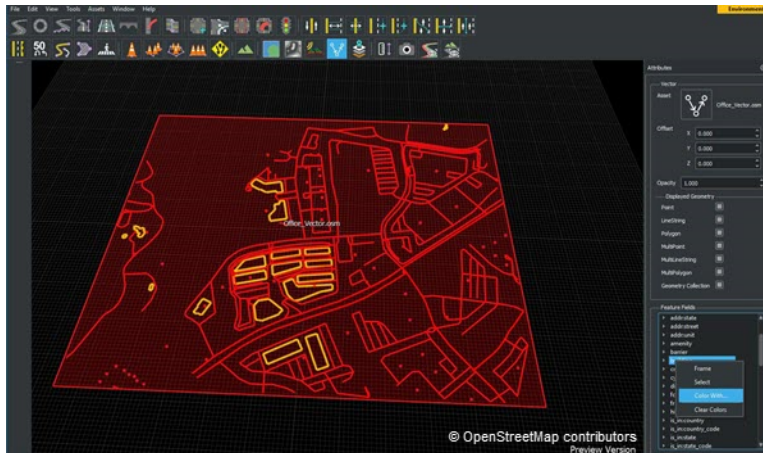
Toggle the Display of Vector Data

Select **View > Vector Data** on the menu bar or press the **F7** key.

View the Attributes of a Feature in a Vector Map

- 1 Click the **Vector Data Tool** button.
- 2 Click the vector feature you want to examine. The properties of the feature are displayed in the **Attributes** pane.
 - You can select multiple features to view the aggregate set of attributes for all selected features.
 - If you click an attribute field name or value in the **Attributes** pane, the main scene highlights the objects with that value.

Color Features Based on Attributes



You can apply custom colors to vector features based on type or other attribute values as follows:

- 1 Click the **Vector Data Tool** button.
- 2 Select the features that you want to recolor (or press **Ctrl+A** to select all).
- 3 In the **Attributes** pane, locate the feature attribute that you want to use for recoloring. If you click an attribute field name or value, the main scene highlights the objects with that value.
- 4 Right-click an attribute field name or value, and select **Color With**.
- 5 Choose a color in the color picker.

Note You can hide the highlighted features by setting the **Alpha** channel value to 0.

Clear Feature Colors

If you assigned a custom color to vector features, you can clear the custom color as follows:

- 1 Click the **Vector Data Tool** button.
- 2 Select the features whose colors you want to reset (or press **Ctrl+A** to select all).
- 3 In the **Attributes** pane, locate the feature attribute whose colors you want to reset.
- 4 Right-click an attribute field name or value and select **Clear Colors**.

Version History

Introduced in R2020a

World Settings Tool

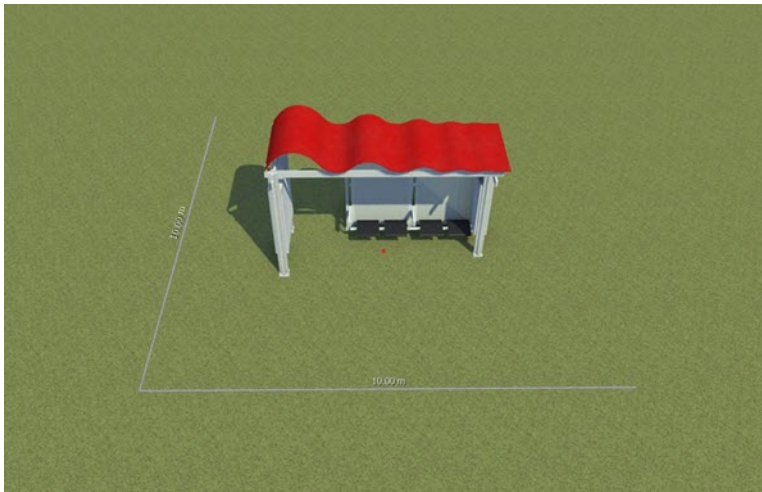
Configure geographic position and size of environment model for data import and export

Description

The **World Settings Tool** is used to configure the geographic position and size of the environment model.

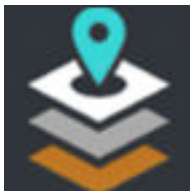
RoadRunner allows models to optionally be georeferenced, meaning they are built to match a particular location on the surface of the Earth.

For more information about the coordinate system used in RoadRunner, refer to “Coordinate Space and Georeferencing”.



Open the World Settings Tool

On the RoadRunner toolbar, click the **World Settings Tool** button:



Examples

Specify an Initial Latitude/Longitude Origin for the Scene

- 1 Click the **World Settings Tool** button.
- 2 Enter the desired latitude and longitude that you want to use as the origin of the scene in the **World Origin** section of the **Attributes** pane.

- 3 Click the **Apply World Changes** button on the **Attributes** pane.

Clear the Geographic Position

- 1 Click the **World Settings Tool** button.
- 2 Press the **Clear World Projection** button in the **Attributes** pane.
- 3 A warning dialog appears, saying that clearing the projection invalidates imported GIS files. To proceed, press the **Clear** button.

Center the Geographic Position on an Existing GIS File

- 1 Click the **World Settings Tool** button.
- 2 Click the GIS file you want to center the projection on.
- 3 Press the **Center World on Selected** button in the **Attributes** pane.
- 4 Press the **Apply World Changes** button in the **Attributes** pane.
- 5 A dialog box appears, asking if you want to just change the world projection (the **Only Change Projection** button) or if you want to transform the model to the new projection (the **Transform Scene** button). See the note on changing world settings to determine the appropriate action and click the appropriate button.

Change the Origin of the Environment to a Specific Latitude/Longitude

- 1 Click the **World Settings Tool** button.
- 2 Enter the desired latitude and longitude that you want to use as the origin of the scene in the **World Origin** section of the **Attributes** pane.
- 3 Click the **Apply World Changes** button on the **Attributes** pane.
- 4 A dialog appears, asking if you want to just change the world projection (the **Only Change Projection** button) or if you want to transform the model to the new projection (the **Transform Scene** button). See the note on changing world settings to determine the appropriate action and press the appropriate button.

Alternatively, follow these steps:

- 1 Click the **World Settings Tool** button. Locate the green circular mark indicating the **World Origin**.
- 2 Click and drag the **World Origin** mark to the desired new origin position.
- 3 Click the **Apply World Changes** button on the **Attributes** pane.
- 4 A dialog appears, asking if you want to just change the world projection (the **Only Change Projection** button) or if you want to transform the model to the new projection (the **Transform Scene** button). See the note on changing world settings to determine the appropriate action and press the appropriate button.

Note To transform a scene, you have two options:

- **Only Change Projection** — Use this option if you want to change the origin without moving any scene objects. This changes the lat/long locations of the objects but preserves their local (XY) coordinates.
 - **Transform Scene** — Use this option if you want to change the origin and move objects to preserve their location on the earth. This changes the local (XY) coordinates of the objects but preserves their geographic (lat/long) coordinates.
-

Change the Workspace Size and Location

- 1 Click the **World Settings Tool** button.
- 2 Enter the desired workspace parameters (**Center** and **Extents**) on the **Attributes** pane.

Alternatively, follow these steps:

- 1 Click and drag inside the blue **World Origin** box in the scene editing canvas to move the center of the workspace.
- 2 Click and drag the edges or corners of the blue **World Origin** box to change the dimensions of the workspace.
- 3 Click the **Apply World Changes** button on the **Attributes** pane.

Note Changing the workspace does not remove or modify any data in the scene. When working with large GIS files, it can be useful to shrink the workspace to view only the portion of the file you want to view at a given time. However, remember to increase the workspace size again before exporting.

More About

Working with GIS Data

Before importing any GIS files or before starting to model any roads, set the desired latitude and longitude in your scene. See “Specify an Initial Latitude/Longitude Origin for the Scene” on page 1-232. This enables you to enter explicit latitude and longitude values.

Another option is to center the origin on the first GIS file imported. RoadRunner does this automatically if you try to drag a GIS file into the scene without having set the origin. Once the origin is set, any additional GIS files brought in are positioned relative to the specified origin.

Note that you can always adjust your world origin later, though it is recommended to make all adjustments prior to creating any roads or other scene elements. Changing the world origin after GIS files have been imported will change the map projection and will require all GIS files to be reloaded and reprojected. This is done automatically when you change the origin.

Scene Workspace

The scene workspace defines the rectangular size of the environment model. This rectangular region is referred to as the workspace and is based on a center *XY* coordinate (in meters relative to the origin) and an *XY* rectangle size, defined in meters.

Because GIS data can get quite large, RoadRunner loads only GIS data that lies within the workspace. The scene workspace is also used for export. The visual scene is geometrically clipped against the scene workspace during export.

To change the workspace dimensions, see “Change the Workspace Size and Location” on page 1-234.

Version History

Introduced in R2020a

Topics

“Create Roads Around Imported GIS Assets”

Assets

Aerial Image Assets

Add GIS satellite and aerial imagery to scene for visual reference

Description



Aerial image assets are used to add geographic information system (GIS) satellite and aerial imagery to a scene, typically for visual reference.

Refer to the **Aerial Imagery Tool** for instructions on adding and adjusting aerial images in your scene.

Creation

Create an asset outside of RoadRunner by using one of the supported file formats shown. Then, drag the file into the RoadRunner **Library Browser**. For more details on creating, editing, and deleting assets, see “Create, Import, and Modify Assets”.

Supported Formats

Image file types that RoadRunner supports:

- Bitmap (.bmp)
- DEM (.dem) — Typically used only for **Aerial Image Assets** or **Elevation Map Assets**
- GIF (.gif)
- GTX (.gtx)
- ICO (.ico)
- IMG (.img)
- JPEG 2000 (.jp2)
- JPEG (.jpg, .jpeg)
- PPM / PGM / PBM (.ppm, .pgm, .pbm)
- PNG (.png)
- RGB (.rgb, .rgba)
- SVG (.svg, .svgz)
- TGA (.tga)
- TIF / GeoTIFF (.tif, .tiff)
- WEBP (.webp)
- X Bitmap Graphic (.xbm)

- X PixMap (.xpm)

GeoTiff, JPEG 2000, and IMG are most common image file types for georeferenced imagery.

See Also

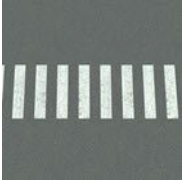
Topics

“Create Roads Around Imported GIS Assets”

Crosswalk Marking Assets

Define crosswalk marking properties, such as color, width, and spacing

Description



Crosswalk marking assets describe the general properties of a crosswalk marking, such as color, width, spacing, and so on. These assets are used by features such as the **Crosswalk And Stop Line Tool** and the **Marking Curve Tool**.

Creation

Create these assets from within the RoadRunner **Library Browser**. For more details on creating, editing, and deleting assets, see “Create, Import, and Modify Assets”.

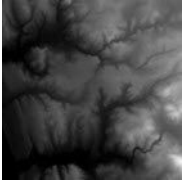
Parameters

Attribute	Description
Width	Width between the centers of the left and right stripes.
Border Width	Width of the left/right stripes.
Dash Length	Length of each dash in ladder-style crosswalks. If zero, no dash will be present.
Dash Gap	Length between dashes in ladder-style crosswalks.
Default Material	Material Assets used by crosswalk instances. This material will be used by any crosswalk instances that do not have an overriding material (that is, crosswalks that do not have an assigned Material attribute in the Marking Curve Tool attributes.) Changing this value will impact any existing crosswalk instances (except those that have an overriding material assigned).
Category	“Segmentation” type for this marking asset.

Elevation Map Assets

Add GIS raster elevation data to scene

Description



Elevation Map Assets are used to add geographic information system (GIS) raster elevation data to a scene.

Refer to the **Elevation Map Tool** for instructions on adding and adjusting elevation maps in your scene.

Creation

Create an asset outside of RoadRunner by using one of the supported file formats shown. Then, drag the file into the RoadRunner **Library Browser**. For more details on creating, editing, and deleting assets, see “Create, Import, and Modify Assets”.

Supported Formats

- DEM (.dem)
- IMG (.img)
- TIF / GeoTIFF (.tif, .tiff)

See Also

Topics

“Create Roads Around Imported GIS Assets”

Extrusion Assets

Define extruded geometry for features such as walls, guard rails, and fences

Description



Extrusion assets are used to create extruded geometry for features such as walls, guard rails, and fences.

Extrusions can combine an extruded cross section (such as the metal railing of a guard rail) with regularly spaced props (such as the wooden posts of the guard rail). These extrusions can be placed along curves using the **Prop Curve Tool**, and along road spans using the **Prop Span Tool**.

Creation

Create these assets from within the RoadRunner **Library Browser**. For more details on creating, editing, and deleting assets, see “Create, Import, and Modify Assets”.

Parameters

Attribute	Description
Default Spacing	Controls how often a prop is placed along the extrusion.
Instance Offset	An offset (in the space of the curve) applied to all props placed along the extrusion.
Item 0, Item 1, and so on	Regularly repeated props within an extrusion. See Add a Repeating Prop to an Extrusion.

You can add a constant offset to all props placed along the extrusion by adjusting the **Instance Offset** attribute in the **Attributes** pane.

Examples

Edit the Shape of an Extrusion

1



Select an extrusion style in the **Library Browser** to view the extrusion profile in the **2D Editor** pane.

2 See "Curve Editing".

The editing controls for extrusions differ from most curve-based tools. For example, there is no explicit tangent control or tangent locking/unlocking. To create a sharp angle, you would create two curves and drag the endpoints of one until it snaps to the end of the other.

Assign Materials to an Extrusion

1 Select an extrusion style in the **Library Browser** to view the extrusion profile in the **2D Editor** pane.

2 Select an extrusion curve in the **2D Editor** pane.

3 Modify any of the following attributes in the **Attributes** pane.

Attribute	Description
Material	Material asset to apply to the selected extrusion curve. For more details, see Material Assets .
Texture Size	Texture coordinate scale for the selected extrusion curve.

Attribute	Description
Two Sided	Whether the material applies to both sides of the selected extrusion curve or a single side. Two-sided is typically only used for thin fences (for example, chain link fences).

More About

Add a Repeating Prop to an Extrusion

You can add regularly spaced props along an extrusion. This is useful to add posts and other supports along the extrusion.

Note that extrusion assets behave much like **Prop Set Assets**. Refer to that documentation for instructions on adding and removing prop items on an extrusion asset.

Lane Marking Assets

Define lane markings, such as color, width, and dash spacing

Description



Lane marking assets describe the properties of lane markings, such as color, width, dash spacing, and so on. These assets are used by features such as the **Lane Marking Tool**, **Marking Curve Tool**, and the **Marking Polygon Tool**. Making a change to a lane marking style will affect all instances that use that style.

Creation

Create these assets from within the RoadRunner **Library Browser**. For more details on creating, editing, and deleting assets, see “Create, Import, and Modify Assets”.

Parameters

Attribute	Description
Marking Type	Line marking pattern. "Solid" is a continuous unbroken line, while "Dashed" refers to a broken line with separate controls for gap and dash lengths.
Width	Width of each marking stripe.
Separation	Gap between marking stripes (for marking types with more than one stripe).
Dash Length	Length of marking dashes (for "Dashed" marking types).
Dash Spacing	Length of gaps between dashes (for "Dashed" marking types).

Attribute	Description
Curve Space Texture	<p>If true, the marking material will be applied in the space of the curve. The vertical axis of the texture will be mapped along the curve, and the horizontal axis will span the total width of the marking.</p> <p>If false, then the marking material will be applied in 'world space', where the vertical axis of the texture will be mapped based on the global 'Y' axis, and the horizontal axis will be mapped based on the global 'X' axis.</p>
Default Color	<p>Initial color to assign to marking instances. Refer to the "Color" attribute in the Marking Curve Tool attributes.</p> <p>Modifying this value in the asset does not impact any existing marking instance.</p>
Default Material	<p>Material Assets used by lane markings in the scene. This material will be used by any marking instances that do not have an overriding material (that is, markings that do not have an assigned Material attribute in the Marking Curve Tool attributes.)</p> <p>Changing this value will impact any existing marking instances (except those that have an overriding material assigned).</p>
Default Start Blend Distance	<p>Initial Start Blend Distance to use for marking instances. Refer to the Start Blend Distance attribute in the Marking Curve Tool attributes.</p> <p>Modifying this value in the asset does not impact any existing marking instance.</p>
Default End Blend Distance	<p>Initial End Blend Distance to use for marking instances. Refer to the End Blend Distance attribute in the Marking Curve Tool attributes.</p> <p>Modifying this value in the asset does not impact any existing marking instance.</p>
Category	<p>"Segmentation" type for this marking asset.</p>

Material Assets

Define visual properties of surfaces, sidewalks, lanes, and other objects

Description



Material assets are used to define the visual properties of surfaces, sidewalks, lanes, and other objects.

Creation

Create these assets from within the RoadRunner **Library Browser**. For more details on creating, editing, and deleting assets, see “Create, Import, and Modify Assets”.

Point Cloud Assets

Add aerial or vehicular point clouds to scene for visual reference

Description



Point cloud assets are used to add aerial or vehicular point clouds to a scene, typically for visual reference.

Refer to the **Point Cloud Tool** for instructions on adding and adjusting point clouds in your scene.

Creation

Create an asset outside of RoadRunner by using one of the supported file formats shown. Then, drag the file into the RoadRunner **Library Browser**. For more details on creating, editing, and deleting assets, see “Create, Import, and Modify Assets”.

Supported Formats

- LAS / LAZ (.las, .laz)
- PCD (.pcd)

Note LAZ files version 1.4 or higher might not load correctly. In these cases, you might need to decompress the files to LAS files. For more details, see “Decompress LAZ Files”.

Polygon Marking Assets

Define space-filling road markings, such as crosshatch and chevron markings

Description



Polygon marking assets describe the properties of space-filling road markings, such as crosshatch and chevron markings.

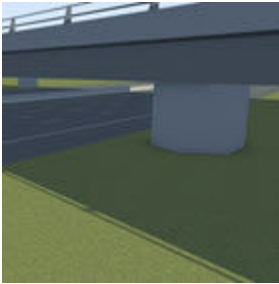
Creation

Create these assets from within the RoadRunner **Library Browser**. For more details on creating, editing, and deleting assets, see “Create, Import, and Modify Assets”.

Post Assets

Define building support posts, such as for bridges and overpasses

Description



Post assets are used for building support posts of varying height, primarily for bridges and overpasses.

You cannot change the material, customize the extrusion profile, or add caps to post assets.

Creation

Create these assets from within the RoadRunner **Library Browser**. For more details on creating, editing, and deleting assets, see “Create, Import, and Modify Assets”.

Prop Assembly Assets

Define collection of prop instances stored as single asset

Description



Prop assembly assets are hierarchical collections of prop instances stored as a single asset that can itself be instantiated within the scene.

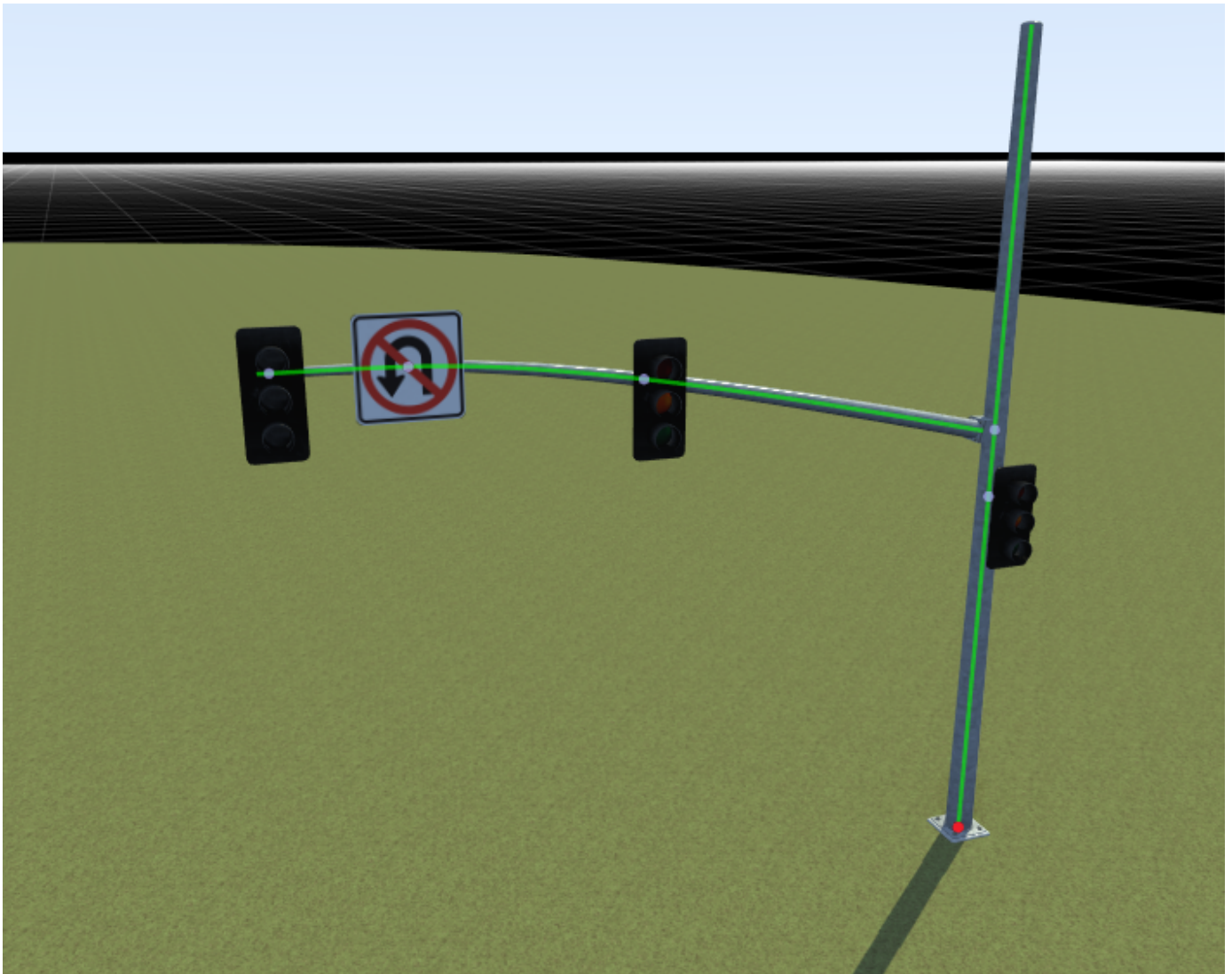
Creation

Create these assets from within the RoadRunner **Library Browser**. For more details on creating, editing, and deleting assets, see “Create, Import, and Modify Assets”.

Examples

Create a Prop Assembly

- 1 Select the **Prop Point Tool**.



- 2 Select the root node of a hierarchical prop (a point prop with one or more props attached to it).
- 3 Click the **Create Assembly** button in the **Attributes** pane.

A new assembly asset appears in the current folder of the **Library Browser** and can be renamed.

The selected prop node is automatically replaced by this new assembly.

Attach a Prop to a Prop Assembly

- 1 Select the **Prop Point Tool**.
- 2 From the **Library Browser**, drag a prop onto the prop assembly. Alternatively, select a prop from the **Library Browser**, and then in the scene, right-click the green attachment curve of an assembly at the point where you want to place the prop.

You can then drag the prop anywhere within the assembly, but you cannot move it off the assembly. You also cannot drag props already in the scene onto an assembly. You must attach new props directly from the **Library Browser**.

Expand a Prop Assembly

- 1** After placing a prop assembly instance in the scene, you can expand it to separate it into its individual components, which breaks the instance's link to the prop assembly asset).

This is useful when you want to modify only a single instance of the assembly. For example, if you want to move a traffic signal head on a traffic assembly.

Select the **Prop Point Tool**.

- 2** Select a prop point with a prop assembly assigned to it.
- 3** Click the **Expand Assembly** button.

Prop Model Assets

Define external 3D model files to add to scene

Description



Prop model assets reference external 3D model files that can be placed within the scene.

Refer to “Props and Signs” for details on adding prop model assets to the scene.

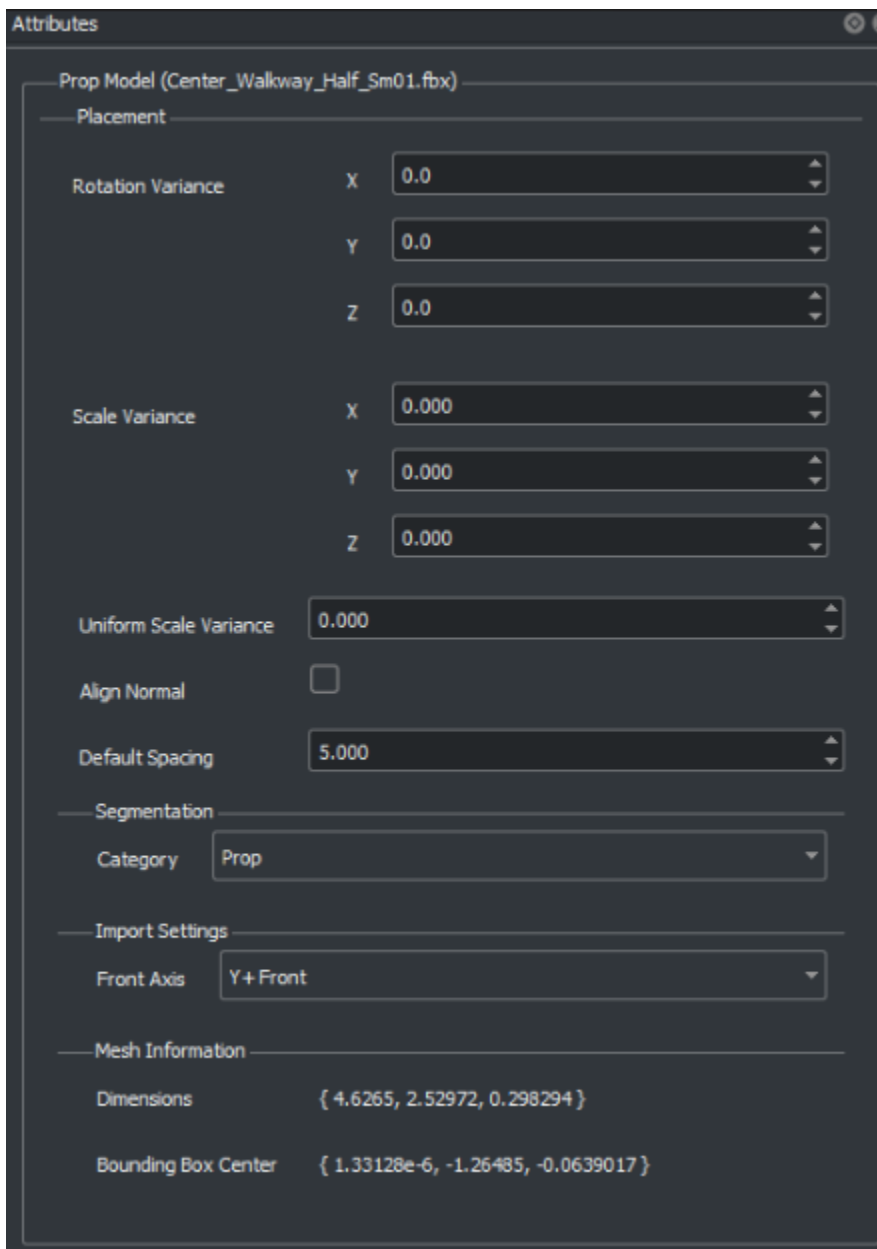
Creation

Create an asset outside of RoadRunner by using one of the supported file formats shown. Then, drag the file into the RoadRunner **Library Browser**. For more details on creating, editing, and deleting assets, see “Create, Import, and Modify Assets”.

Supported Formats

- glTF™ (.gltf, .glb)
- Filmbox (.fbx)
- OpenFlight (.flt)
- OpenSceneGraph (.osg, .osgt, .osgb, .ive)
- Wavefront® (.obj)

Parameters



To set the default attributes for a particular prop asset, first select the prop in the Asset Browser. The prop's attributes will then appear in the **Attributes** pane, where they can be interactively adjusted.

All prop assets have a set of options that affect the way the props are placed. For example, some props will always align vertically, such as a traffic signal. Other props will align to the surface they are placed on, such as a garbage can sitting on a sloped sidewalk. (This option can be toggled on a prop-by-prop basis.) Other options include the ability to randomly rotate the prop around the vertical axis, which is useful for varying the orientation of trees and plants.

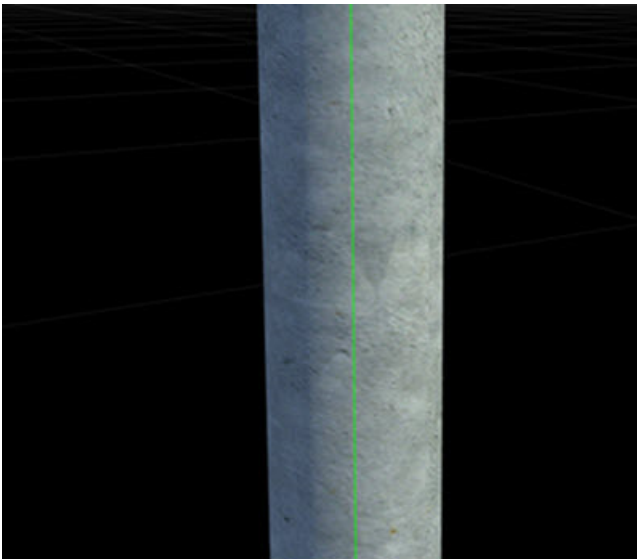
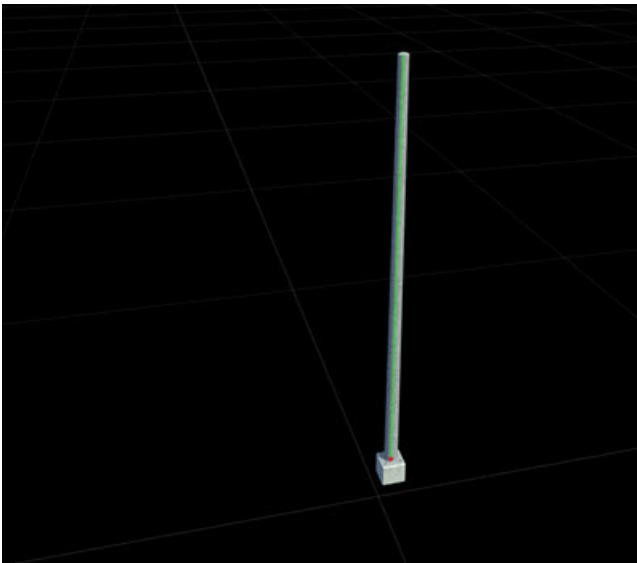
You can view the **Mesh Information** for props in the **Attributes** pane. The **Mesh Information** specifies the values for the **Dimensions** and **Bounding Box Center** for the selected prop. The **Mesh Information** is in meters and the **Bounding Box Center** is relative to the origin of the prop file. The **Dimensions** specify the full length on each axis of the prop.

More About

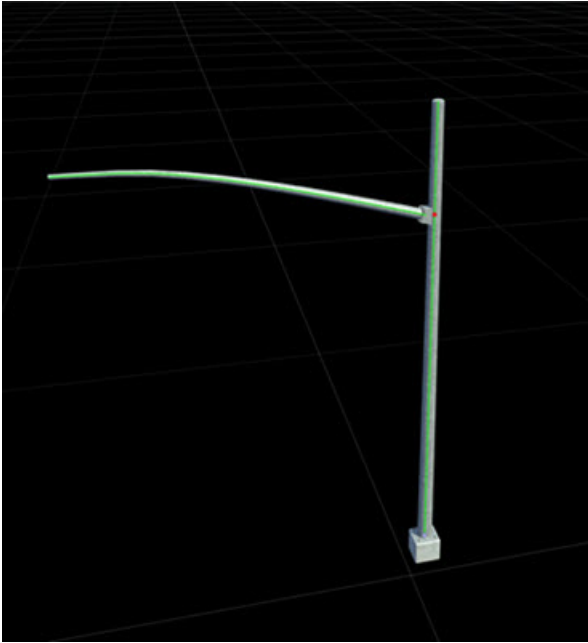
Prop Attachment Curves

An attachment curve is a spline associated with a prop that RoadRunner uses as a cue when attaching objects to each other.

As an example, note the green highlight line in the following signal pole prop. The image on the right is a close-up that shows this detail.

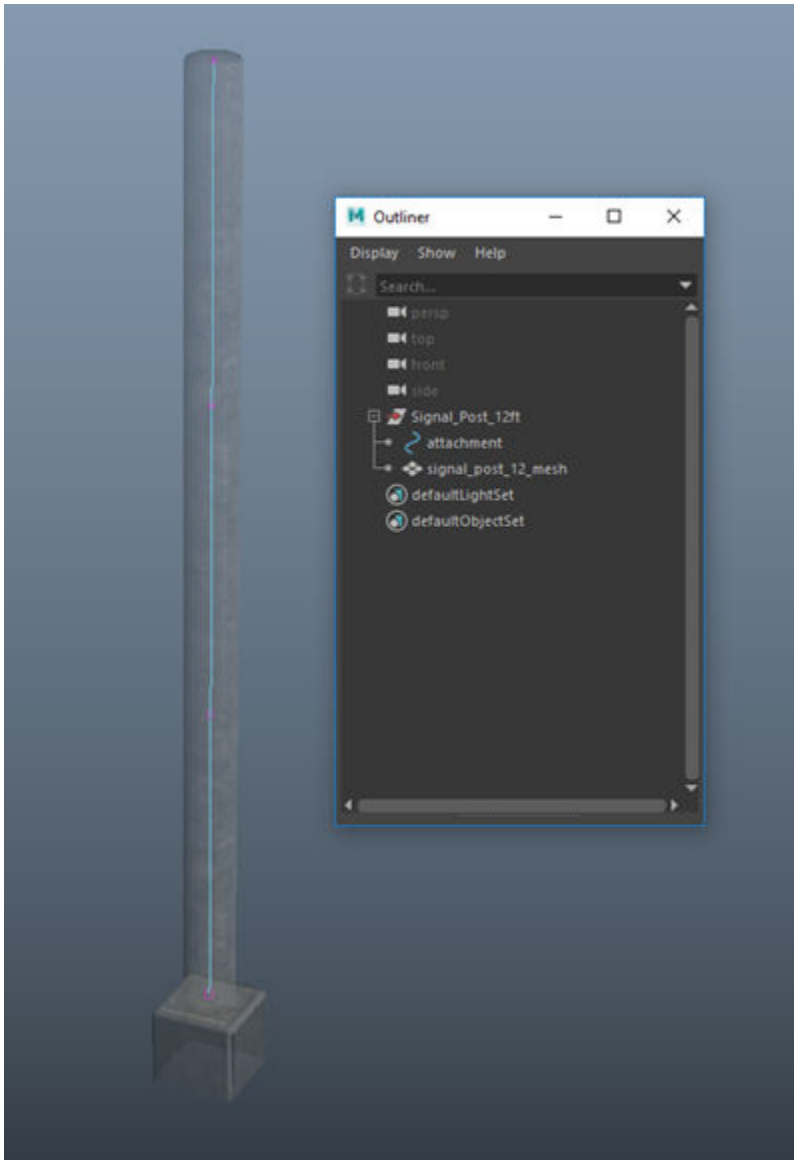


When you pick another prop from the **Library Browser** and bring it sufficiently close to the attachment curve, they snap together at that point. In this case, a signal mast arm has been attached.

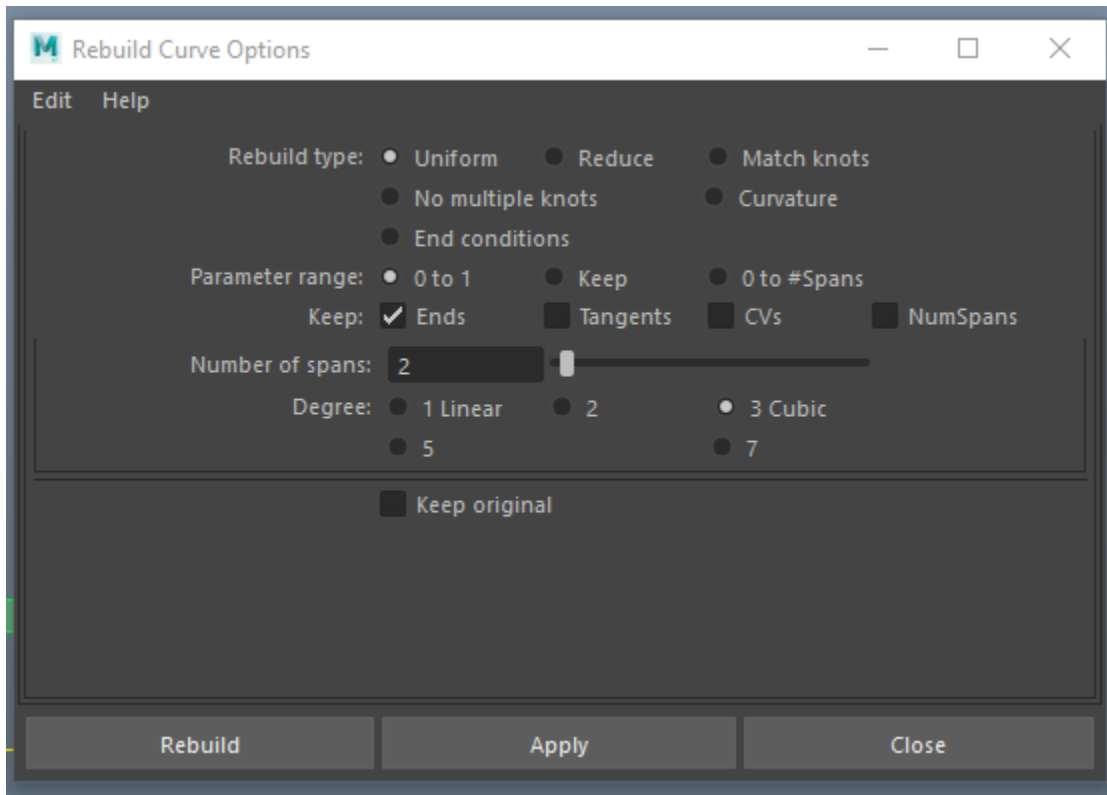


Creating Prop Attachment Curves

RoadRunner does not have a user interface for creating prop attachment curves directly. However, if you have access to a third-party tool like Maya®, you can create these yourself on any prop. Add a spline named `attachment` into the object hierarchy.



If the attachment curve is not appearing in RoadRunner, you may need to add more curve points so that it can be detected. In Maya, this option is under **Curve > Rebuild**.



Format Details

FBX Details

- Layered Textures are not supported and will not be connected to the imported materials.
- Texture transformations (scale, rotation, and translation) are not supported and will be ignored.
- RoadRunner supports importing FBX® Lambertian and Phong materials.
- Multiple UV sets are not supported. These extra UVs must be removed in a program like Maya before being imported into RoadRunner.
- Importing lights from FBX files is not supported.
- Levels of Detail (LODs) are based on the node name. During import, RoadRunner checks if the node name ends with `_med`, `_low`, or `_verylow`.

Note RoadRunner does not render lower LODs, so nodes that end in `_med`, `_low`, or `_verylow` will not be visible.

glTF Details

- Texture sampler settings are not imported and defaults to `repeat` on both axes.
- Sparse accessors are not supported.
- Point and line primitive modes are not supported.

OpenSceneGraph Details

- When importing materials, `osg::Textures` are loaded as materials. These single texture materials will take precedence over `osg::Materials` when generating the mesh.

- Only Overall and Per Vertex color bindings are supported.
- Multiple UVs are not supported. Only the first texture coordinate array is used.
- Point and line PrimitiveSet modes are not supported.

Advanced Details

Textures

Most 3D models have associated image files (such as texture maps or normal maps). Place these image files in the same directory as the prop model itself, or (in certain formats, such as FBX) they can also be embedded into the actual prop file.

Unit Scale and Coordinate System

- RoadRunner uses meter units. Imported FBX files will automatically convert units if needed.
- RoadRunner uses the Maya Z-up coordinate system: +Z is up, +X is right, and -Y is toward the camera.

Note Imported FBX files will automatically rotate to match the coordinate system. However, FBX files created in a left-handed coordinate system will not be converted properly.

Prop Set Assets

Define collections of props that have a random distribution

Description



Prop set assets reference collections of different props that have a random distribution. For example, you can create a prop set of trees that contains a collection of different tree models. Prop sets specify a relative distribution for each item in the set. This influences the likelihood of that item appearing when the prop set is added to the scene. Prop sets can be placed on points, curves, polygons, and spans.

Creation

Create these assets from within the RoadRunner **Library Browser**. For more details on creating, editing, and deleting assets, see “Create, Import, and Modify Assets”.

If you have one or more prop assets selected when you create the new prop set, those assets are automatically added to the new prop set. If no prop assets are selected, an empty prop set is created.

Examples

Add an Item to a Prop Set

- 1 Select a prop set in the **Library Browser**.
- 2 Click the **Add Prop** button in the **Attributes** pane. A new (empty) item is added to the end of the list of items.
- 3 Assign a prop style asset to the prop by dragging an asset from the **Library Browser** into the empty **Prop** asset picker box.

You can use any of the following asset types as a prop set item:

- **Prop Assembly Assets**
- **Prop Model Assets**
- **Prop Set Assets**

Remove an Item from a Prop Set

- 1 Select a prop set in the **Library Browser**.
- 2 Click the **Remove Prop** button below the item you want to remove in the **Attributes** pane.

Road Style Assets

Define templates that specify properties of new roads

Description



Road style assets are templates that specify the properties used when creating new roads. These properties include the number of lanes, lane types, lane widths, lane marking materials, and road cross section information. It also contains information about extrusions, such as barriers and repeated props.

Road style assets are typically used only when initially creating a road. After that point, you can use the road tools described in “Roads, Lanes, and Markings” to customize any aspect of the created road.

A selection of road styles is included with RoadRunner. You can create new road styles and modify existing ones.

Road style creation and editing differs from many other asset types. Refer to the steps listed here.

Creation

You can create a style from any road in your RoadRunner scene.

- 1 Find or create a road and customize it to your liking using the road tools. For more details, see “Roads, Lanes, and Markings”.
- 2 Select the **Cross Section Tool**.
- 3 Select the road.
- 4 Right-click a location along the road to create a cross section (or click an existing cross section).
- 5 Click the **Create Road Style** button on the **Attributes** pane to create a road style in the current **Library Browser** folder.
- 6 Rename the road style asset, if desired.

Examples

Apply Road Style to New Road

Refer to the **Road Plan Tool** documentation.

Apply Road Style to Existing Road

Click and drag the road style from the **Library Browser** onto the road you want to change.

This operation changes the entire road and overwrites any local changes previously made to the road, such as lane edits, marking edits, attribute edits, cross section edits, and more. Therefore, it is recommended that you use it only when first creating a road.

Edit Road Style Asset

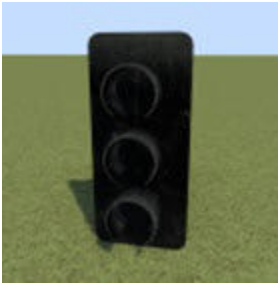
You can replace the contents of a road style asset with a new road style. This action does not affect any previously created roads.

- 1** Find or create a road and customize it to your liking using the road tools. See “Roads, Lanes, and Markings”.
- 2** Select the **Cross Section Tool**.
- 3** Select the road.
- 4** Right-click a location along the road to create a cross section (or click an existing cross section).
- 5** Click the **Update Selected Style** button in the **Attributes** pane.

Signal Assets

Define dynamic traffic signal heads with lights

Description



Signal assets are 3D models that allow for dynamic traffic signal heads with lights.

In many regards, signal assets are similar to **Prop Model Assets**. For example, the same steps are used to add signal assets to the 3D scene.

The “RoadRunner Asset Library Add-On” includes a variety of signals. You can also add your own custom signal models (refer to the Create a Custom Signal Prop section below).

Signal assets include the notion of variants, which allow the same 3D model to be configured with multiple different bulb layouts. For example, a single 3-bulb model could have one variant for green/red/yellow ball lights, and another for green/red/yellow left turn lights. Variants require two texture atlases that define the on and off state for all bulbs available for that signal. For details on creating texture atlases, see **Texture Assets**.

Each variant defines a set of supported turn types that are used to automatically map signals to maneuvers with the **Signal Tool**.

Creation

Create an asset outside of RoadRunner by using one of the supported file formats shown. Then, drag the file into the RoadRunner **Library Browser**. For more details on creating, editing, and deleting assets, see “Create, Import, and Modify Assets”.

Supported Formats

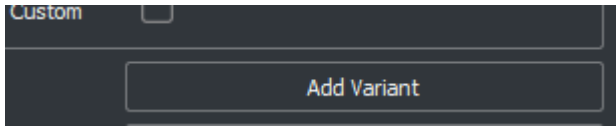
- Filmbox (.fbx)
- Wavefront (.obj)

Examples

Create a Signal Variant

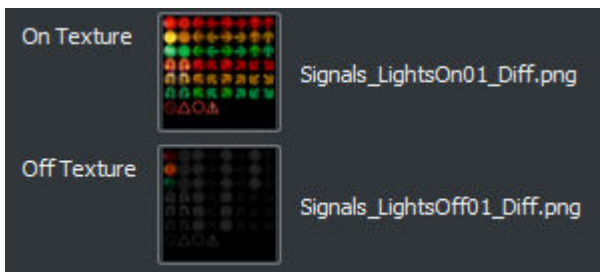
- 1 Select a signal asset in the **Library Browser**.

- 2 Click **Add Variant** in the **Attributes** pane.



Set Bulb Textures

- 1 Select a signal asset in the **Library Browser**.
- 2 Assign texture atlases to the **On Texture** and **Off Texture** asset pickers in the **Attributes** pane.



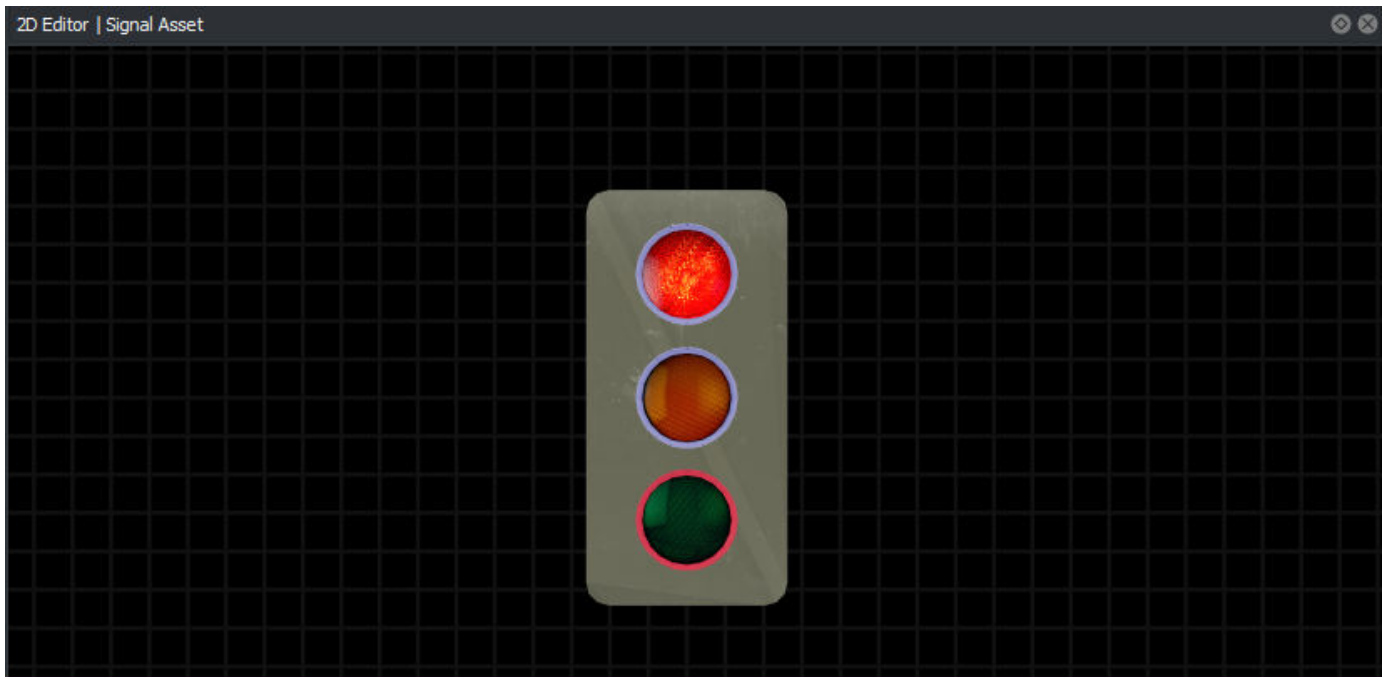
Name or Rename a Signal Variant

- 1 Select a signal asset in the **Library Browser**.
- 2 Choose the variant to edit in the **Variant** drop-down list in the **Attributes** pane.
- 3 Set the **Name** of the variant in the **Attributes** pane.

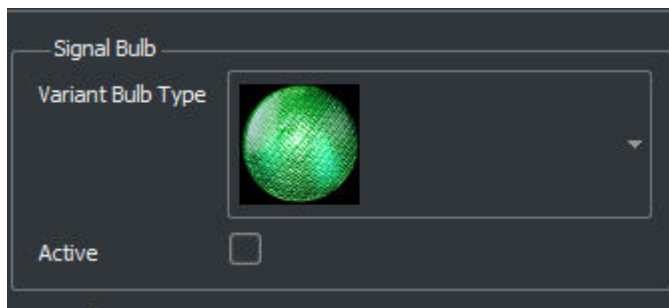


Set or Change a Bulb Type

- 1 Select a signal asset in the **Library Browser**.

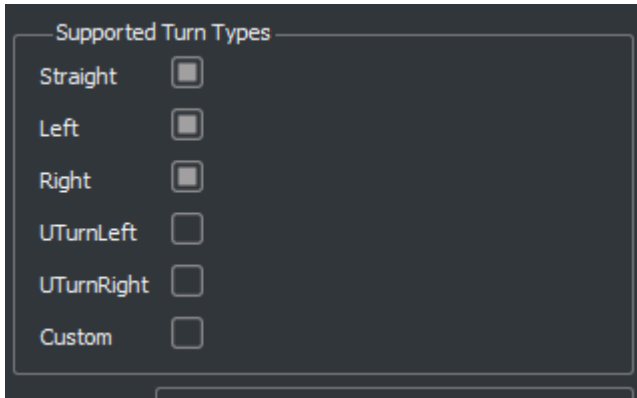


- 2 Select a bulb in the **2D Editor** pane.
- 3 Choose a bulb type (**Variant Bulb Type**) in the **Attributes** pane.



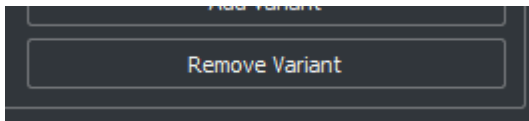
Specify Supported Turn Types

- 1 Select a signal asset in the **Library Browser**.
- 2 Choose the variant to edit in the **Variant** drop-down list in the **Attributes** pane.
- 3 Check the boxes that correspond to the controlled turn types for this signal.



Remove a Signal Variant

- 1 Select a signal asset in the **Library Browser**.
- 2 Choose the variant to remove in the **Variant** drop-down list in the **Attributes** pane.
- 3 Click **Remove Variant**.

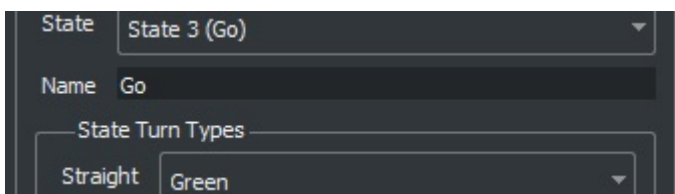


Create a Signal State

- 1 Select a signal asset in the **Library Browser**.
- 2 Click **Add State** in the **Attributes** pane.

Name or Rename a Signal State

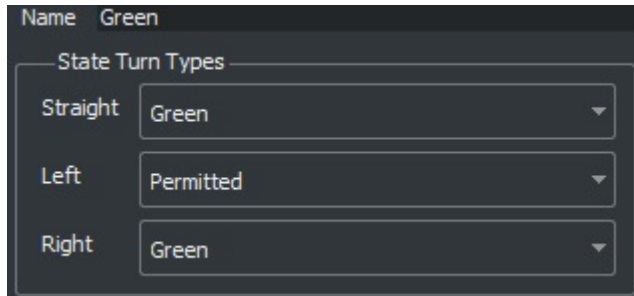
- 1 Select a signal asset in the **Library Browser**.
- 2 Choose the variant to edit in the **Variant** drop-down list in the **Attributes** pane.
- 3 Choose the state to edit in the **State** drop-down list.
- 4 Set the **Name** of the state in the **Attributes** pane.



Specify State Supported Turn Types

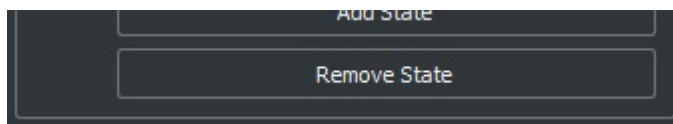
- 1 Select a signal asset in the **Library Browser**.

- 2 Choose the variant to edit in the **Variant** drop-down list in the **Attributes** pane.
- 3 Choose the state to edit in the **State** drop-down list.
- 4 Set the signal mode for each supported turn type in the **State Turn Types** group.



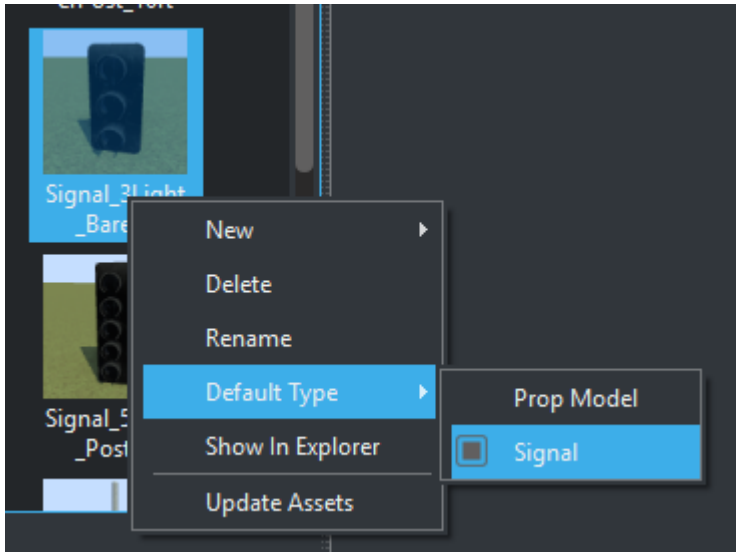
Remove a Signal State

- 1 Select a signal asset in the **Library Browser**.
- 2 Choose the variant to edit in the **Variant** drop-down list in the **Attributes** pane.
- 3 Choose the state to edit in the **State** drop-down list.
- 4 Click **Remove State**.



Create a Custom Signal Prop

- 1 Model the signal in a 3D modeling program.
- 2 Create bulb meshes for each bulb.
- 3 Prefix the name of each bulb mesh with `light_`
- 4 Construct the UVs of the bulb mesh using the full UV grid size (0 to 1).
- 5 Export as FBX.
- 6 Add the asset file using the **Library Browser**.
- 7 Right-click the asset and set the **Default Type** to **Signal**.



Tips

- If automatic detection in the **Signal Tool** does not choose the correct state for a signal, try adjusting the supported turn types for the state or variant.
- Make a red state first, because this will be the default.
- If the signal bulbs do not display correctly, ensure that the on and off textures are set and that both of those textures are texture atlases with the correct grid size. For more details on texture atlases, see **Texture Assets**.

Sign Assets

Define standard and custom street signs

Description



Sign Assets are used to create and edit standard and custom street signs.

Creation

RoadRunner has tools for quickly creating both basic and complex road signs. These signs can be created and stored in the project assets, and then placed throughout the scene in 3D. A basic sign is a sign that is built from a single image file (bitmap or vector graphic), such as a stop or yield sign. A complex sign can have multiple elements, such as text and graphics, such as a custom freeway guide sign. Both basic and complex signs can be created as assets in the asset library and then instanced in the scene as props.

Create Basic Sign Asset from Single Bitmap or Vector Image

- 1 Click in the Asset Browser to select the destination directory in which to import the sign.
- 2 Drag the bitmap file (JPG, PNG, or BMP) or vector graphic file (SVG) from a file browser into the Asset Browser. This action copies the actual file into the project assets. Once the image is in the project assets, RoadRunner automatically generates an icon for the image.
- 3 Right-click the asset icon and select the **Default Type > Sign** menu option. This treats the image file as a sign asset type and is necessary because images can be used as other asset types as well.

Create Complex Sign Asset with Custom Text and Graphics

To create a complex sign asset with custom text, graphics, and so on, follow these steps:

- 1 Click in the Asset Browser to select the destination directory in which to import the sign.
- 2 Right-click in the Asset Browser and select the **New > Sign** menu option. This selection creates a new sign asset that can be further customized with the **Sign Tool**.

Supported Formats

Image file types that RoadRunner supports:

- Bitmap (.bmp)
- DEM (.dem) — Typically used only for **Aerial Image Assets** or **Elevation Map Assets**
- GIF (.gif)
- GTX (.gtx)

- ICO (.ico)
- IMG (.img)
- JPEG 2000 (.jp2)
- JPEG (.jpg, .jpeg)
- PPM / PGM / PBM (.ppm, .pgm, .pbm)
- PNG (.png)
- RGB (.rgb, .rgba)
- SVG (.svg, .svgz)
- TGA (.tga)
- TIF / GeoTIFF (.tif, .tiff)
- WEBP (.webp)
- X Bitmap Graphic (.xbm)
- X PixMap (.xpm)

Examples

Permanently Delete Sign Asset from Project

- 1 Click the sign asset you want to delete in the Asset Browser.
- 2 Press the **Delete** key, or select **Edit > Delete** from the menu bar.
- 3 A dialog box warns you that this operation permanently deletes the file. Click **Yes** if you want to proceed.

Modify Existing Sign Asset

- 1 Click the sign asset you want to edit in the Asset Browser to display the attributes of the sign in the **Attributes** pane.
- 2 Edit the sign attributes.
- 3 In the menu bar, select **File > Save Project**. Changes to assets are not saved until you save the project.

Stencil Marking Assets

Define road paint features, such as arrows, text, and symbols

Description



Stencil marking assets are used to place road paint features, such as arrows, text, and symbols.

Refer to the **Marking Point Tool** for more information about using stencil marking assets as point markings.

Creation

Create an asset outside of RoadRunner by using one of the supported file formats shown. Then, drag the file into the RoadRunner **Library Browser**. For more details on creating, editing, and deleting assets, see “Create, Import, and Modify Assets”.

Supported Formats

- SVG (.svg, .svgz)

Examples

Create a Stencil Marking Asset

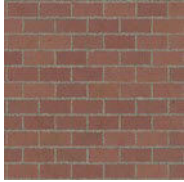
- 1 Add the asset file using the **Library Browser**.
- 2 Right-click the asset and set the **Default Type** to **Stencil Marking**.

The SVG parser used for stencil markings is limited. Many SVG elements are not supported. If your SVG is not supported and you want to use the stencil as a point marking, you can convert it to any of the supported non-SVG image file types and use the **Marking Point Tool** to add the resulting texture assets to the scene. For a list of supported non-SVG image file types, see **Texture Assets**.

Texture Assets

Define texture channels for material assets

Description



Texture assets are image files that are typically used as texture channels for **Material Assets**.

Creation

Create an asset outside of RoadRunner by using one of the supported file formats shown. Then, drag the file into the RoadRunner **Library Browser**. For more details on creating, editing, and deleting assets, see “Create, Import, and Modify Assets”.

Supported Formats

Image file types that RoadRunner supports:

- Bitmap (.bmp)
- DEM (.dem) — Typically used only for **Aerial Image Assets** or **Elevation Map Assets**
- GIF (.gif)
- GTX (.gtx)
- ICO (.ico)
- IMG (.img)
- JPEG 2000 (.jp2)
- JPEG (.jpg, .jpeg)
- PPM / PGM / PBM (.ppm, .pgm, .pbm)
- PNG (.png)
- RGB (.rgb, .rgba)
- SVG (.svg, .svgz)
- TGA (.tga)
- TIF / GeoTIFF (.tif, .tiff)
- WEBP (.webp)
- X Bitmap Graphic (.xbm)
- X PixMap (.xpm)

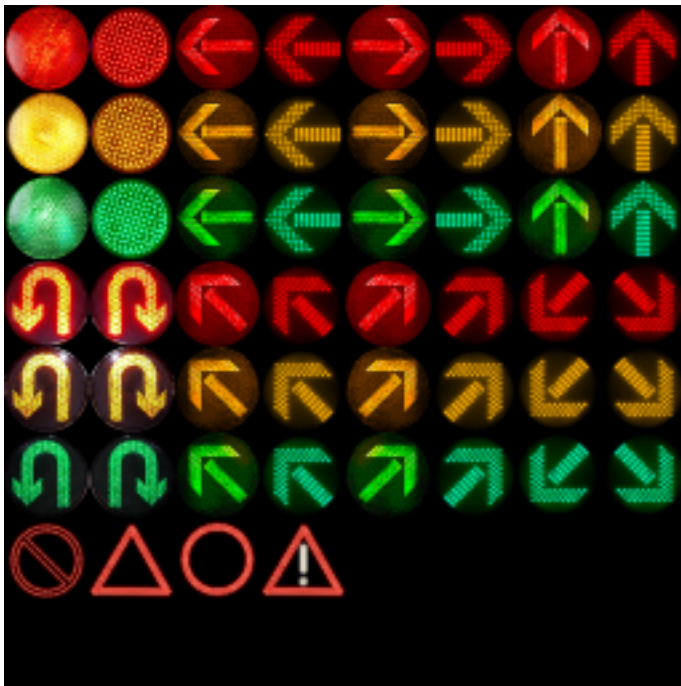
Attributes

Attribute	Description
Default Width	Default spatial size of the texture, in meters. Used, for example, as the default sign size when the texture file is used for Sign Assets .
Is Texture Atlas	See Create a Texture Atlas.

Examples

Create a Texture Atlas

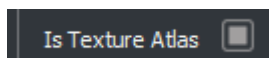
A texture atlas is a single texture image that contains multiple uniformly spaced sub-images, such as this image.



These are used as textures for **Signal Assets** and point markings. For more details on point markings, see the **Marking Point Tool**.

To set texture atlas properties for a texture asset:

- 1 Select one or more texture assets in the Library Browser.
- 2 Select the **Is Texture Atlas** option in the **Attributes** pane.



- 3 Set the grid size of the texture (these specify the number of sub-image rows and columns) in the **Atlas Tiles** option.



See Also
Material Assets

Vector Data Assets

Add GIS shapefiles and other vector data to scene for visual reference

Description



Vector Data Assets are used to add geographic information system (GIS) shapefiles and other vector data to a scene, typically for visual reference.

Refer to **Vector Data Tool** for instructions on adding and adjusting vector data in your scene.

Creation

Create an asset outside of RoadRunner by using one of the supported file formats shown. Then, drag the file into the RoadRunner **Library Browser**. For more details on creating, editing, and deleting assets, see “Create, Import, and Modify Assets”.

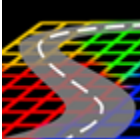
Supported Formats

- GeoJSON (.geojson, .json)
- GPS Exchange (.gpx)
- OpenStreetMap (.osm, .pbf)
- Shapefile (.shp, .dbf, .prj)
- Keyhole Markup Language (.kml, .kmz)

Synthetic CRG Assets

Define road surface data in CRG format

Description



Synthetic CRG assets describe road surface properties, such as surface noise, speed bumps, potholes, and rumble strips, using the curved regular grid (CRG) format defined by ASAM OpenCRG. Using the **Road CRG Tool**, you can assign these assets to a selected road segment and visualize the road surface data.

Creation

Create these assets from within the RoadRunner **Library Browser**. For more details on creating, editing, and deleting assets, see “Create, Import, and Modify Assets”.

Alternatively, you can create these assets from the **Attributes** pane of the **Road CRG Tool**. For more information, see “Create and Apply Synthetic CRG Data from Attributes Pane” on page 2-47.

Parameters

Base Grid

Use these attributes to create the base CRG grid and define noise properties for a road surface.

Attribute	Description
Header Information	Information about the asset file, such as the title of the file, a short description of the contents, and author information.
Samples Per Square Meter	Density of the grid points showing road surface data, specified as a real scalar. When you export synthetic CRG data to ASAM OpenCRG file, this parameter defines the incremental values along u- and v-coordinates of road surface data. Default: 2500
Grid Length	Length of the grid for road surface description, specified as a real scalar. Units are in meters. Default: 10

Attribute	Description
Grid Width	Width of the grid for road surface description, specified as a real scalar. Units are in meters. Default: 2
Noise Strength	Strength of noise in road surface descriptions, specified as a real scalar. Default: 0.01
Noise Scale	Noise scaling factor, specified as a real scalar. Default: 1
Offset Value	Global offset of the grid from the road surface level, specified as a scalar in the range [-0.05, 0.05]. Units are in meters. Default: 0.01

Note

- Synthetic CRG assets do not support the heading angle and slope parameters of the CRG reference line.
 - To visualize base CRG grid data using the **Road CRG Tool**, you must specify **Grid Length** and **Grid Width** values such that the CRG grid overlaps with the underlying road segment. You cannot visualize CRG data outside the extents of the underlying road segment.
-

Element (Speed Bump)

To enable attributes for a speed bump element, add a speed bump element to the base grid. In the **Attributes** pane, click **Add Element**. Then, in the Select Element Type window, select Speed Bumps from the drop-down list and click **OK**.

Attribute	Description
Position > X	X-coordinate of the center of the speed bump, specified as a nonnegative real scalar. This attribute represents the offset of the X-coordinate of the speed bump center from the X-coordinate of the start position of the grid. Units are in meters. Default: 2
Position > Y	Y-coordinate of the center of the speed bump, specified as a real scalar. This attribute represents the offset of the Y-coordinate of the speed bump center from the Y-coordinate of the start position of the grid. Units are in meters. Default: 0

Attribute	Description
Length	Length of the speed bump, specified as a nonnegative real scalar. Units are in meters. Default: 0.25
Width	Width of the speed bump, specified as a nonnegative real scalar. Units are in meters. Default: 2
Step Speed Bump	Render a speed bump as a step. Default: off
Height	Height of the speed bump, specified as nonnegative real scalar. Units are in meters. Default: 0.05
Remove Element	Remove the selected speed bump element from the grid.

Note To visualize a speed bump element using the **Road CRG Tool**, you must specify **Position**, **Length**, and **Width** values such that the speed bump element overlaps with the base CRG grid.

Element (Pothole)

To enable attributes for a pothole element, add a pothole element to the base grid. In the **Attributes** pane, click **Add Element**. Then, in the Select Element Type window, select **Potholes** from the drop-down list and click **OK**.

Attribute	Description
Position > X	X-coordinate of the center of the pothole, specified as a nonnegative real scalar. This attribute represents the offset of the X-coordinate of the pothole center from the X-coordinate of the start position of the grid. Units are in meters. Default: 2
Position > Y	Y-coordinate of the center of the pothole, specified as a real scalar. This attribute represents the offset of the Y-coordinate of the pothole center from the Y-coordinate of the start position of the grid. Units are in meters. Default: 0
Radius	Radius of the pothole, specified as a nonnegative real scalar. Units are in meters. Default: 0.250

Attribute	Description
Depth	Depth of the pothole, specified as a real scalar. Units are in meters. Default: -0.05
Remove Element	Remove the selected pothole element from the grid.

Note To visualize a pothole element using the **Road CRG Tool**, you must specify **Position** and **Radius** values such that the pothole element overlaps with the base CRG grid.

Element (Rumble strip)

To enable attributes for a rumble strip element, add a rumble strip element to the base grid. In the **Attributes** pane, click **Add Element**. Then, in the Select Element Type window, select **Rumble Strips** from the drop-down list and click **OK**.

Attribute	Description
Position > X	X-coordinate of the center of the rumble strips element, specified as a nonnegative real scalar. This attribute represents the offset of the X-coordinate of the rumble strips center from the X-coordinate of the start position of the grid. Units are in meters. Default: 2
Position > Y	Y-coordinate of the center of the rumble strips element, specified as a real scalar. This attribute represents the offset of the Y-coordinate of the rumble strips center from the Y-coordinate of the start position of the grid. Units are in meters. Default: 0
Length of Strip(s)	Length of each rumble strip, specified as a nonnegative real scalar. Units are in meters. Default: 0.3
Width of Strip(s)	Width of each rumble strip, specified as a nonnegative real scalar. Units are in meters. Default: 2
Spacing between strips	Spacing between a pair of consecutive rumble strips, specified as a nonnegative real scalar. Units are in meters. Default: 0.15

Attribute	Description
Number of strips	Number of rumble strips, specified as a nonnegative real scalar. Default: 5
Height	Height of each rumble strip, specified as nonnegative real scalar. Units are in meters. Default: 0.015
Step Rumble Strip	Render each rumble strip as a step. Default: off
Remove Element	Remove the selected rumble strips element from the grid.

Note To visualize rumble strips using the **Road CRG Tool**, you must specify the **Position, Length of Strip(s), Width of Strip(s), Spacing between strips**, and **Number of strips** values such that the rumble strips element overlaps with the base CRG grid.

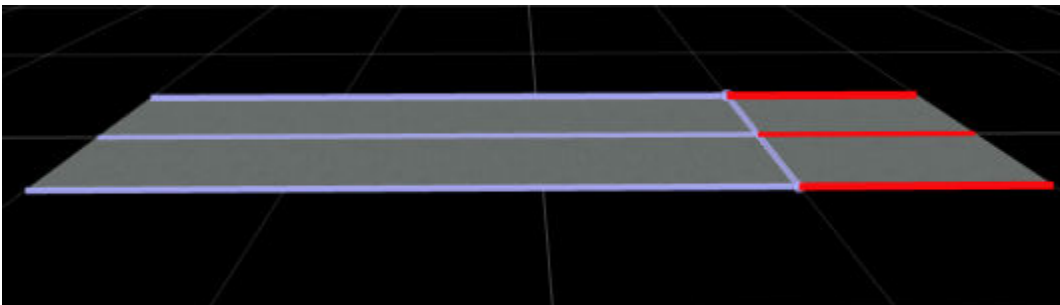
Examples

Create Synthetic CRG Asset

- 1 Navigate to the folder in the **Library Browser** where you want to create the new Synthetic CRG asset.
- 2 Right-click in the **Library Browser** and select **New**, then select **Synthetic CRG**. Alternatively, from the **Assets** menu, select **New**. Then select **Synthetic CRG**.
- 3 Specify the name `example` and press **Enter**.

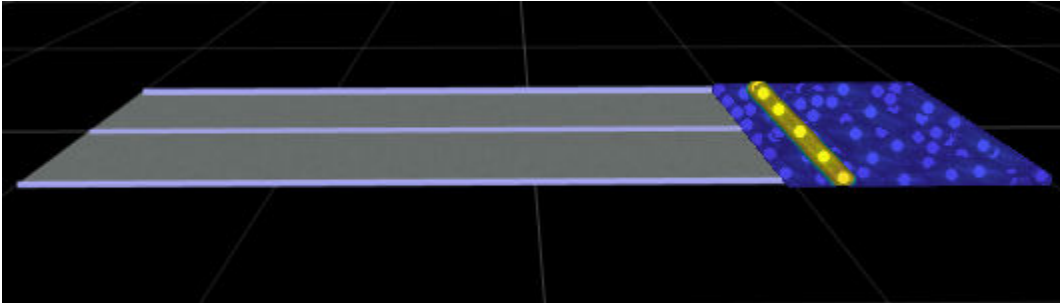
RoadRunner creates the asset `example.rrcrg`. The **Attributes** pane displays various attributes of the generated asset.

Apply Asset to Road Segment and Modify Asset



- 1 Click the **Road CRG Tool** button on the toolbar.
- 2 Select the road to which you want to apply road surface data.
- 3 Right-click a location on the road to insert a span node that divides the road into two segments. Insert more span nodes if you want to divide the road into more segments.
- 4 Select one of the road segments.

- 5 Drag the `example.rrcrg` asset file from the **Library Browser** into the **CRG file** asset holder in the **Attributes** pane.
- 6 RoadRunner visualizes the road surface variations using a colormap.
- 7 Select the `example.rrcrg` file from the **Library Browser** and modify the value of the **Grid Width** attribute so that it matches the road width.
- 8 To add a speed bump element on the grid, in the **Attributes** pane, click **Add Element**. Then, in the Select Element Type window, select **Speed Bumps** from the drop-down list and click **OK**.
- 9 Modify the value of the **Width** attribute of the speed bump element so that it matches the value of the **Grid Width** attribute of the base grid.



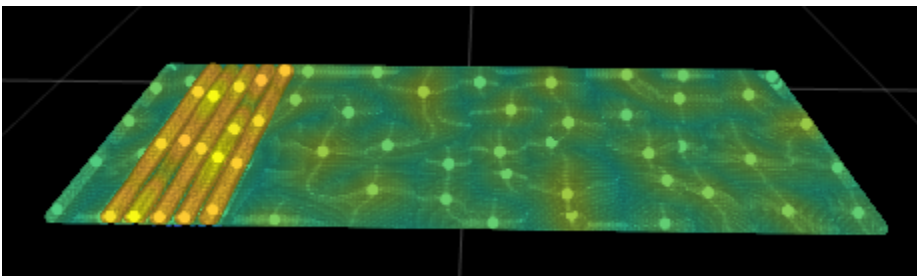
Create and Apply Synthetic CRG Data from Attributes Pane

Follow these steps when you want to create and apply synthetic CRG data to the specified road segment from the **Attributes** pane of the **Road CRG Tool**.

- 1 Click the **Road CRG Tool** button on the toolbar.
- 2 Select the road to which you want to apply road surface data. If you want to select a specific road segment, right-click two desired locations on the road to insert span nodes. Then, select the road segment between the two span nodes.
- 3 In the **Attributes** pane, click **Create Synthetic CRG Asset**. Notice that RoadRunner creates a new CRG asset in the **Library Browser** and automatically applies it to the selected road segment. The grid size of the created asset matches the size of the selected road segment. Therefore, you do not need to manually set the **Grid Length** and **Grid Width** attributes.

You can rename the created asset from the **Library Browser**.

- 4 To add rumble strips on the grid, in the **Attributes** pane, click **Add Element**. Then, in the Select Element Type window, select **Rumble Strips** from the drop-down list and click **OK**.
- 5 Modify the value of the **Width** attribute of the **Rumble Strips** element so that it matches the value of the **Grid Width** attribute of the base grid.



Export Synthetic CRG Data

You can export synthetic CRG data to an ASAM OpenCRG file when you export a scene to an ASAM OpenDRIVE file. For more information, see “Export to ASAM OpenCRG”.

Version History

Introduced in R2022a

See Also

Tools

Road CRG Tool

Topics

“Create, Import, and Modify Assets”

“Importing ASAM OpenCRG Files”

“Export to ASAM OpenCRG”

External Websites

ASAM OpenCRG

Functions

AppRoadRunner

Start RoadRunner application from command line using gRPC

Syntax

```
AppRoadRunner
AppRoadRunner option1 ... optionN
```

Description

AppRoadRunner starts RoadRunner from the command line and opens the application to the start page.

To run AppRoadRunner, first use the command line to navigate to the folder in your RoadRunner installation that contains the AppRoadRunner executable file. Then, call this file from the command line. These are the default installation locations by platform.

- Windows® — C:\Program Files\RoadRunner R2023a\bin\win64
- Linux® Ubuntu® — /usr/local/RoadRunner_R2023a/bin/glnxa64

AppRoadRunner option1 ... optionN starts RoadRunner with the specified startup options.

Examples

Open RoadRunner and Start API Servers

Open RoadRunner to a new scene in an existing project and start running the RoadRunner API server and the scenario co-simulation servers on different network ports. This example shows how to open RoadRunner from the Windows command line, but you can use similar commands in Linux.

Navigate to the folder in your installed version of RoadRunner that contains the AppRoadRunner executable. For example, this path shows the default Windows installation folder.

```
cd "C:\Program Files\RoadRunner R2023a\bin\win64"
```

Open a RoadRunner project located in folder C:\RR\MyProject. For the RoadRunner API server, use IP network port 54321. For the scenario API server, use IP network port 54322.

```
AppRoadRunner --projectPath C:\RR\MyProject --apiPort 54321 --cosimPort 54322
```

RoadRunner opens to a new scene in this project. The **Output** pane displays a message that the RoadRunner API server has been started on port 54321.

```
> Started RoadRunner API server on port 54321.
```

If you have a RoadRunner Scenario license, in the top-right corner of RoadRunner, select **Scene Editing**, then **Scenario Editing** to switch to scenario editing mode. The **Output** pane displays an additional message that the RoadRunner scenario co-simulation server has been started on port 54322.

```
> Started RoadRunner API server on port 54321.
> Started Scenario API server on port 54322.
```

Input Arguments

option1 ... optionN – One or more startup options

strings

One or more startup options, specified as strings corresponding to valid startup options from this table.

Option	Result
<code>--projectPath <i>project</i></code>	<p>Open RoadRunner to a new scene in <i>project</i> and start the RoadRunner API server.</p> <p>Specify <i>project</i> as a path to a valid RoadRunner project folder.</p> <p>If you do not specify this argument, then RoadRunner opens to the start page but does not start the API server.</p> <p>Example: <code>--projectPath C:\RR\MyProject</code></p>
<code>--apiPort <i>port</i></code>	<p>Start the RoadRunner API server on IP network port <i>port</i>. This server receives commands from the RoadRunner service methods for importing and exporting scenes and scenarios.</p> <p>Specify <i>port</i> as an integer in the range [1024, 65535].</p> <p>When calling the RoadRunner APIs, specify this port as the port to use to communicate with the opened instance of RoadRunner. If you are using the APIs with multiple RoadRunner instances open, specify a different port for each instance.</p> <p>Example: <code>--apiPort 54321</code></p> <p>Default: 35707</p>

Option	Result
<code>--cosimPort port</code>	<p>Start the RoadRunner scenario co-simulation server on IP network port <i>port</i>. This server receives commands for co-simulation with MATLAB® and Simulink® and with external simulators such as CARLA (requires RoadRunner Scenario).</p> <p>Specify <i>port</i> as an integer in the range [1024, 65535].</p> <p>Example: <code>--cosimPort 54322</code></p> <p>Default: 35706</p>
<code>--verbose</code>	Display developer messages and additional information in the command-line output.
<code>-v, --version</code>	Display RoadRunner version information.
<code>-, -h, --help</code>	Display the help for the RoadRunner application.
<code>--debugOpenGL</code>	Enable OpenGL® debugging mode to diagnose graphics issues.
<code>--nodisplay</code>	Start RoadRunner in console mode with a non-graphical environment. For more details, see “Control RoadRunner Programmatically in Console Mode”.

Limitations

- AppRoadRunner does not output error logs and other debugging information to the command line. To view such information, open AppRoadRunner in a third-party application that supports debugging information for UI applications.

Tips

- When you open RoadRunner using AppRoadRunner, if you see a message that the port is occupied, check whether you have another instance of RoadRunner open on that port. You can have only one instance of RoadRunner open on a port at one time. To open multiple instances of RoadRunner using AppRoadRunner, specify a different port value for each instance.

Version History

Introduced in R2021b

See Also

`NewProject` | `LoadProject` | `SaveProject` | `NewScene` | `LoadScene` | `SaveScene` | `NewScenario` | `LoadScenario` | `SaveScenario` | `SetScenarioVariable` | `PrepareSimulation` | `SimulateScenario` | `Export` | `Import` | `Exit`

Topics

“Control RoadRunner Programmatically Using gRPC API”

“Convert Scenes Between Formats Using gRPC API”
“Export Multiple Scenes Using gRPC API”

NewProject

Create new RoadRunner project using gRPC

Description

The `NewProject` method creates a new RoadRunner project and discards any unsaved changes.

This RoadRunner API method is a remote procedure call (RPC) that sends a single message to the RoadRunner API service and receives a single message back as a response. The protocol buffer (protobuf) file `roadrunner_service.proto` defines the schema for this method. Using a gRPC® compiler, you can compile the RoadRunner protobuf files into a language supported by gRPC and create client applications to call this method in that language. For more details, see “Compile Protocol Buffers for RoadRunner gRPC API”. For background information on how the RoadRunner API works, see “Control RoadRunner Programmatically Using gRPC API”.

Request

NewProjectRequest — New project request message

New project request, specified as a message with these fields.

Name	Type	Description
<code>folder_path</code> (required)	string	Path at which to create the new project. RoadRunner creates the folder structure recursively. If the folder already exists, then RoadRunner returns an error.
<code>asset_libraries</code> (optional)	repeated string	Asset libraries to include in the new project. The only valid entry is "RoadRunner_Asset_Library", which includes the RoadRunner Asset Library assets in the project. If you do not specify <code>asset_libraries</code> , then RoadRunner includes only the assets that come installed with RoadRunner projects by default.

Response

NewProjectResponse — New project response empty message

New project response, returned as an empty message.

Sample Calls

Command Line

Create a new project. By default, the project includes only the default set of assets included with every RoadRunner project.

```
cd "C:\Program Files\RoadRunner R2023a\bin\win64"
AppRoadRunner --projectPath C:\RR\MyProject --apiPort 54321
CmdRoadRunnerApi "NewProject(folder_path='C:\RR\MyNewProject')" --serverAddress localhost:54321
```

This sample call uses the `CmdRoadRunnerApi` helper command, which is a precompiled version of the RoadRunner service API. For examples that use this command, see:

- “Convert Scenes Between Formats Using gRPC API”
- “Export Multiple Scenes Using gRPC API”

Note `CmdRoadRunnerApi` does not support the `asset_libraries` field, because this helper command does not support repeated field types.

Python

Create a new project. Include the RoadRunner Asset Library assets in the project.

```
newProjectRequest = roadrunner_service_messages_pb2.NewProjectRequest()
newProjectRequest.folder_path = "C:\RR\MyNewProject"
newProjectRequest.asset_libraries.append("RoadRunner_Asset_Library")
api.NewProject(newProjectRequest)
```

This sample call is a snippet of a Python® client. For details on creating complete Python clients, see “Create gRPC Python Client for Controlling RoadRunner Programmatically”.

In this sample call, `api` is a Python stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

C++

Create a new project. Include the RoadRunner Asset Library assets in the project.

```
NewProjectRequest request;
std::string folderPath = "C:\RR\MyNewProject";
request.set_folder_path(folderPath);
request.add_asset_libraries("RoadRunner_Asset_Library");
ClientContext context;
NewProjectResponse reply;
Status status = api->NewProject(&context, request, &reply);
```

This sample call is a snippet of a C++ client. For details on creating complete C++ clients, see “Create gRPC C++ Client for Controlling RoadRunner Programmatically”.

In this sample call, `api` is a C++ stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

Version History

Introduced in R2021b

See Also

LoadProject | SaveProject | NewScene | LoadScene | SaveScene | NewScenario | LoadScenario | SaveScenario | SetScenarioVariable | PrepareSimulation | SimulateScenario | Export | Import | Exit | roadrunner_service.proto | roadrunner_service_messages.proto

Topics

“Control RoadRunner Programmatically Using gRPC API”

LoadProject

Load RoadRunner project using gRPC

Description

The `LoadProject` method loads a specified RoadRunner project and opens RoadRunner to a new scene in that project. If the specified project is already loaded, RoadRunner still opens to a new scene.

This RoadRunner API method is a remote procedure call (RPC) that sends a single message to the RoadRunner API service and receives a single message back as a response. The protocol buffer (protobuf) file `roadrunner_service.proto` defines the schema for this method. Using a gRPC compiler, you can compile the RoadRunner protobuf files into a language supported by gRPC and create client applications to call this method in that language. For more details, see “Compile Protocol Buffers for RoadRunner gRPC API”. For background information on how the RoadRunner API works, see “Control RoadRunner Programmatically Using gRPC API”.

Request

LoadProjectRequest — Load project request message

Load project request, specified as a message with this field.

Name	Type	Description
<code>folder_path</code> (required)	string	Path to existing project folder to load.

Response

LoadProjectResponse — Load project response empty message

Load project response, returned as an empty message.

Sample Calls

Command Line

Open RoadRunner to the project located at `C:\RR\MyProject`. Then, load the project located at `C:\RR\MyOtherProject`.

```
cd "C:\Program Files\RoadRunner R2023a\bin\win64"
AppRoadRunner --projectPath C:\RR\MyProject --apiPort 54321
CmdRoadRunnerApi "LoadProject(folder_path='C:\RR\MyOtherProject')" --serverAddress localhost:54321
```

This sample call uses the `CmdRoadRunnerApi` helper command, which is a precompiled version of the RoadRunner service API. For examples that use this command, see:

- “Convert Scenes Between Formats Using gRPC API”
- “Export Multiple Scenes Using gRPC API”

Python

Load the project located at C:\RR\MyProject.

```
loadProjectRequest = roadrunner_service_messages_pb2.LoadProjectRequest()
loadProjectRequest.folder_path = "C:\RR\MyProject"
api.LoadProject(loadProjectRequest)
```

This sample call is a snippet of a Python client. For details on creating complete Python clients, see “Create gRPC Python Client for Controlling RoadRunner Programmatically”.

In this sample call, `api` is a Python stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

C++

Load the project located at C:\RR\MyProject.

```
LoadProjectRequest request;
std::string folderPath = "C:\RR\MyProject";
request.set_folder_path(folderPath);
ClientContext context;
LoadProjectResponse reply;
Status status = api->LoadProject(&context, request, &reply);
```

This sample call is a snippet of a C++ client. For details on creating complete C++ clients, see “Create gRPC C++ Client for Controlling RoadRunner Programmatically”.

In this sample call, `api` is a C++ stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

Version History

Introduced in R2021b

See Also

NewProject | SaveProject | NewScene | LoadScene | SaveScene | NewScenario | LoadScenario | SaveScenario | SetScenarioVariable | PrepareSimulation | SimulateScenario | Export | Import | Exit | roadrunner_service.proto | roadrunner_service_messages.proto

Topics

“Control RoadRunner Programmatically Using gRPC API”

SaveProject

Save RoadRunner project using gRPC

Description

The `SaveProject` method saves any assets modified in the currently loaded scene to the current project.

This RoadRunner API method is a remote procedure call (RPC) that sends a single message to the RoadRunner API service and receives a single message back as a response. The protocol buffer (protobuf) file `roadrunner_service.proto` defines the schema for this method. Using a gRPC compiler, you can compile the RoadRunner protobuf files into a language supported by gRPC and create client applications to call this method in that language. For more details, see “Compile Protocol Buffers for RoadRunner gRPC API”. For background information on how the RoadRunner API works, see “Control RoadRunner Programmatically Using gRPC API”.

Request

SaveProjectRequest — Save project request
empty message

Save project request, specified as an empty message.

Response

SaveProjectResponse — Save project response
empty message

Save project response, returned as an empty message.

Sample Calls

Command Line

Save the project located at `C:\RR\MyProject`.

```
cd "C:\Program Files\RoadRunner R2023a\bin\win64"  
AppRoadRunner --projectPath C:\RR\MyProject --apiPort 54321  
CmdRoadRunnerApi "SaveProject()" --serverAddress localhost:54321
```

This sample call uses the `CmdRoadRunnerApi` helper command, which is a precompiled version of the RoadRunner service API. For examples that use this command, see:

- “Convert Scenes Between Formats Using gRPC API”
- “Export Multiple Scenes Using gRPC API”

Python

Save the current project.

```
saveProjectRequest = roadrunner_service_messages_pb2.SaveProjectRequest()
api.SaveProject(saveProjectRequest)
```

This sample call is a snippet of a Python client. For details on creating complete Python clients, see “Create gRPC Python Client for Controlling RoadRunner Programmatically”.

In this sample call, `api` is a Python stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

C++

Save the current project.

```
SaveProjectRequest request;
ClientContext context;
SaveProjectResponse reply;
Status status = api->SaveProject(&context, request, &reply);
```

This sample call is a snippet of a C++ client. For details on creating complete C++ clients, see “Create gRPC C++ Client for Controlling RoadRunner Programmatically”.

In this sample call, `api` is a C++ stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

Version History

Introduced in R2021b

See Also

[NewProject](#) | [LoadProject](#) | [NewScene](#) | [LoadScene](#) | [SaveScene](#) | [NewScenario](#) | [LoadScenario](#) | [SaveScenario](#) | [SetScenarioVariable](#) | [PrepareSimulation](#) | [SimulateScenario](#) | [Export](#) | [Import](#) | [Exit](#) | [roadrunner_service.proto](#) | [roadrunner_service_messages.proto](#)

Topics

“Control RoadRunner Programmatically Using gRPC API”

NewScene

Create new RoadRunner scene using gRPC

Description

The `NewScene` method creates a new scene in the current RoadRunner project and discards any unsaved changes.

This RoadRunner API method is a remote procedure call (RPC) that sends a single message to the RoadRunner API service and receives a single message back as a response. The protocol buffer (protobuf) file `roadrunner_service.proto` defines the schema for this method. Using a gRPC compiler, you can compile the RoadRunner protobuf files into a language supported by gRPC and create client applications to call this method in that language. For more details, see “Compile Protocol Buffers for RoadRunner gRPC API”. For background information on how the RoadRunner API works, see “Control RoadRunner Programmatically Using gRPC API”.

Request

NewSceneRequest — New scene request
empty message

New scene request, specified as an empty message.

Response

NewSceneResponse — New scene response
empty message

New scene response, returned as an empty message.

Sample Calls

Command Line

Create a new scene in the project located at `C:\RR\MyProject`.

```
cd "C:\Program Files\RoadRunner R2023a\bin\win64"  
AppRoadRunner --projectPath C:\RR\MyProject --apiPort 54321  
CmdRoadRunnerApi "NewScene()" --serverAddress localhost:54321
```

This sample call uses the `CmdRoadRunnerApi` helper command, which is a precompiled version of the RoadRunner service API. For examples that use this command, see:

- “Convert Scenes Between Formats Using gRPC API”
- “Export Multiple Scenes Using gRPC API”

Python

Create a new scene in the current project.

```
newSceneRequest = roadrunner_service_messages_pb2.NewSceneRequest()
api.NewScene(newSceneRequest)
```

This sample call is a snippet of a Python client. For details on creating complete Python clients, see “Create gRPC Python Client for Controlling RoadRunner Programmatically”.

In this sample call, `api` is a Python stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

C++

Create a new scene in the current project.

```
NewSceneRequest request;
ClientContext context;
NewSceneResponse reply;
Status status = api->NewScene(&context, request, &reply);
```

This sample call is a snippet of a C++ client. For details on creating complete C++ clients, see “Create gRPC C++ Client for Controlling RoadRunner Programmatically”.

In this sample call, `api` is a C++ stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

Version History

Introduced in R2021b

See Also

[NewProject](#) | [LoadProject](#) | [SaveProject](#) | [LoadScene](#) | [SaveScene](#) | [NewScenario](#) | [LoadScenario](#) | [SaveScenario](#) | [SetScenarioVariable](#) | [PrepareSimulation](#) | [SimulateScenario](#) | [Export](#) | [Import](#) | [Exit](#) | [roadrunner_service.proto](#) | [roadrunner_service_messages.proto](#)

Topics

“Control RoadRunner Programmatically Using gRPC API”

LoadScene

Load RoadRunner scene using gRPC

Description

The `LoadScene` method loads a specified scene from the current RoadRunner project and discards any unsaved changes. If the scene that you specify does not belong to the current project, then RoadRunner determines what project the scene belongs to and loads it from that project instead.

This RoadRunner API method is a remote procedure call (RPC) that sends a single message to the RoadRunner API service and receives a single message back as a response. The protocol buffer (protobuf) file `roadrunner_service.proto` defines the schema for this method. Using a gRPC compiler, you can compile the RoadRunner protobuf files into a language supported by gRPC and create client applications to call this method in that language. For more details, see “Compile Protocol Buffers for RoadRunner gRPC API”. For background information on how the RoadRunner API works, see “Control RoadRunner Programmatically Using gRPC API”.

Request

LoadSceneRequest — Load scene request message

Load scene request, specified as a message with this field.

Name	Type	Description
<code>file_path</code> (required)	string	<p>Absolute or relative path to the scene file to load. If you specify a relative path, then the path is relative to the <code>Scenes</code> folder of the current project.</p> <p>The filename in <code>file_path</code> must either end with the <code>.rrscene</code> extension or have no extension. If it has no extension, then RoadRunner appends the <code>.rrscene</code> extension to <code>file_path</code> before loading the scene.</p>

Response

LoadSceneResponse — Load scene response empty message

Load scene response, returned as an empty message.

Sample Calls

Command Line

Load the prebuilt `FourWaySignal` scene from the project located at `C:\RR\MyProject`.

```
cd "C:\Program Files\RoadRunner R2023a\bin\win64"
AppRoadRunner --projectPath C:\RR\MyProject --apiPort 54321
CmdRoadRunnerApi "LoadScene(file_path='FourWaySignal')" --serverAddress localhost:54321
```

This sample call uses the `CmdRoadRunnerApi` helper command, which is a precompiled version of the RoadRunner service API. For examples that use this command, see:

- “Convert Scenes Between Formats Using gRPC API”
- “Export Multiple Scenes Using gRPC API”

Python

Load the prebuilt `FourWaySignal` scene from the current project.

```
loadSceneRequest = roadrunner_service_messages_pb2.LoadSceneRequest()
loadSceneRequest.file_path = "FourWaySignal"
api.LoadScene(loadSceneRequest)
```

This sample call is a snippet of a Python client. For details on creating complete Python clients, see “Create gRPC Python Client for Controlling RoadRunner Programmatically”.

In this sample call, `api` is a Python stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

C++

Load the prebuilt `FourWaySignal` scene from the current project.

```
LoadSceneRequest request;
std::string filePath = "FourWaySignal";
request.set_file_path(filePath);
ClientContext context;
LoadSceneResponse reply;
Status status = api->LoadScene(&context, request, &reply);
```

This sample call is a snippet of a C++ client. For details on creating complete C++ clients, see “Create gRPC C++ Client for Controlling RoadRunner Programmatically”.

In this sample call, `api` is a C++ stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

Version History

Introduced in R2021b

See Also

`NewProject` | `LoadProject` | `SaveProject` | `NewScene` | `SaveScene` | `NewScenario` | `LoadScenario` | `SaveScenario` | `SetScenarioVariable` | `PrepareSimulation` | `SimulateScenario` | `Export` | `Import` | `Exit` | `roadrunner_service.proto` | `roadrunner_service_messages.proto`

Topics

“Control RoadRunner Programmatically Using gRPC API”

SaveScene

Save RoadRunner scene using gRPC

Description

The `SaveScene` method saves a specified RoadRunner scene. If you modified any assets used in the scene, then this method also saves the project to which that scene belongs.

This RoadRunner API method is a remote procedure call (RPC) that sends a single message to the RoadRunner API service and receives a single message back as a response. The protocol buffer (protobuf) file `roadrunner_service.proto` defines the schema for this method. Using a gRPC compiler, you can compile the RoadRunner protobuf files into a language supported by gRPC and create client applications to call this method in that language. For more details, see “Compile Protocol Buffers for RoadRunner gRPC API”. For background information on how the RoadRunner API works, see “Control RoadRunner Programmatically Using gRPC API”.

Request

SaveSceneRequest — Save scene request
message

Save scene request, specified as a message with this field.

Name	Type	Description
file_path (optional)	string	<p>Absolute or relative path to the scene file to save. If you specify a relative path, then the path is relative to the Scenes folder of the current project.</p> <p>If you do not specify <code>file_path</code>, then RoadRunner saves the current scene. If you do not have a current scene loaded, then RoadRunner returns an error.</p> <p>If any folders in the file path are missing, then RoadRunner creates them recursively.</p> <p>The filename in <code>file_path</code> must either end with the <code>.rrscene</code> extension or have no extension. If it has no extension, then RoadRunner appends the <code>.rrscene</code> extension to <code>file_path</code> before saving the scene. If the file being saved already exists, then RoadRunner overwrites it.</p>

Response

SaveSceneResponse — Save scene response

empty message

Save scene response, returned as an empty message.

Sample Calls

Command Line

Save the current scene to the Scenes folder of project `C:\RR\MyProject` and name the scene `MyScene`.

```
cd "C:\Program Files\RoadRunner R2023a\bin\win64"
AppRoadRunner --projectPath C:\RR\MyProject --apiPort 54321
CmdRoadRunnerApi "SaveScene(file_path='MyScene')" --serverAddress localhost:54321
```

This sample call uses the `CmdRoadRunnerApi` helper command, which is a precompiled version of the RoadRunner service API. For examples that use this command, see:

- “Convert Scenes Between Formats Using gRPC API”
- “Export Multiple Scenes Using gRPC API”

Python

Save the current scene to the **Scenes** folder of the current project and name the scene **MyScene**.

```
saveSceneRequest = roadrunner_service_messages_pb2.SaveSceneRequest()
saveSceneRequest.file_path = "MyScene"
api.SaveScene(saveSceneRequest)
```

This sample call is a snippet of a Python client. For details on creating complete Python clients, see “Create gRPC Python Client for Controlling RoadRunner Programmatically”.

In this sample call, **api** is a Python stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

C++

Save the current scene to the **Scenes** folder of the current project and name the scene **MyScene**.

```
SaveSceneRequest request;
std::string filePath = "MyScene";
request.set_file_path(filePath);
ClientContext context;
SaveSceneResponse reply;
Status status = api->SaveScene(&context, request, &reply);
```

This sample call is a snippet of a C++ client. For details on creating complete C++ clients, see “Create gRPC C++ Client for Controlling RoadRunner Programmatically”.

In this sample call, **api** is a C++ stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

Version History

Introduced in R2021b

See Also

NewProject | LoadProject | SaveProject | NewScene | LoadScene | NewScenario | LoadScenario | SaveScenario | SetScenarioVariable | PrepareSimulation | SimulateScenario | Export | Import | Exit | roadrunner_service.proto | roadrunner_service_messages.proto

Topics

“Control RoadRunner Programmatically Using gRPC API”

Export

Export RoadRunner scene or scenario using gRPC

Description

The Export method exports a specified RoadRunner scene or scenario to one of the file formats that RoadRunner supports. For more details on supported export formats, see “Export Scenes”.

This RoadRunner API method is a remote procedure call (RPC) that sends a single message to the RoadRunner API service and receives a single message back as a response. The protocol buffer (protobuf) file `roadrunner_service.proto` defines the schema for this method. Using a gRPC compiler, you can compile the RoadRunner protobuf files into a language supported by gRPC and create client applications to call this method in that language. For more details, see “Compile Protocol Buffers for RoadRunner gRPC API”. For background information on how the RoadRunner API works, see “Control RoadRunner Programmatically Using gRPC API”.

Request

ExportRequest — Export request message

Export request, specified as a message with these fields.

Name	Type	Description
<code>file_path</code> (required)	string	<p>Absolute or relative path to the exported file. If you specify a relative path, then the exported file is saved relative to the Exports folder of the current project.</p> <p>If any folders in the path are missing, RoadRunner tries to create them.</p> <p><code>file_path</code> must include the extension of the exported file.</p>

Name	Type	Description																					
format_name (required)	string	<p>Export format name corresponding to a valid RoadRunner export format. Format name options are case-insensitive.</p> <table border="1" data-bbox="1063 478 1459 1837"> <thead> <tr> <th data-bbox="1063 478 1193 583">format_name Option</th> <th data-bbox="1193 478 1323 583">Description</th> <th data-bbox="1323 478 1459 583">Export Format Details</th> </tr> </thead> <tbody> <tr> <td data-bbox="1063 583 1193 758">"AutoCAD"</td> <td data-bbox="1193 583 1323 758">Export scene to an AutoCAD® file.</td> <td data-bbox="1323 583 1459 758">"Export to AutoCAD"</td> </tr> <tr> <td data-bbox="1063 758 1193 932">"Filmbox"</td> <td data-bbox="1193 758 1323 932">Export scene to a Filmbox file.</td> <td data-bbox="1323 758 1459 932">"Export to FBX"</td> </tr> <tr> <td data-bbox="1063 932 1193 1192">"glTF"</td> <td data-bbox="1193 932 1323 1192">Export scene to a GL Transmission Format (glTF) file.</td> <td data-bbox="1323 932 1459 1192">"Export to glTF"</td> </tr> <tr> <td data-bbox="1063 1192 1193 1367">"OpenFlight"</td> <td data-bbox="1193 1192 1323 1367">Export scene to an OpenFlight file.</td> <td data-bbox="1323 1192 1459 1367">"Export to OpenFlight"</td> </tr> <tr> <td data-bbox="1063 1367 1193 1570">"OpenSceneGraph"</td> <td data-bbox="1193 1367 1323 1570">Export scene to an OpenSceneGraph file.</td> <td data-bbox="1323 1367 1459 1570">"Export to OpenSceneGraph"</td> </tr> <tr> <td data-bbox="1063 1570 1193 1837">"USD"</td> <td data-bbox="1193 1570 1323 1837">Export scene to a Universal Scene Description (USD) file.</td> <td data-bbox="1323 1570 1459 1837">"Export to USD"</td> </tr> </tbody> </table>	format_name Option	Description	Export Format Details	"AutoCAD"	Export scene to an AutoCAD® file.	"Export to AutoCAD"	"Filmbox"	Export scene to a Filmbox file.	"Export to FBX"	"glTF"	Export scene to a GL Transmission Format (glTF) file.	"Export to glTF"	"OpenFlight"	Export scene to an OpenFlight file.	"Export to OpenFlight"	"OpenSceneGraph"	Export scene to an OpenSceneGraph file.	"Export to OpenSceneGraph"	"USD"	Export scene to a Universal Scene Description (USD) file.	"Export to USD"
format_name Option	Description	Export Format Details																					
"AutoCAD"	Export scene to an AutoCAD® file.	"Export to AutoCAD"																					
"Filmbox"	Export scene to a Filmbox file.	"Export to FBX"																					
"glTF"	Export scene to a GL Transmission Format (glTF) file.	"Export to glTF"																					
"OpenFlight"	Export scene to an OpenFlight file.	"Export to OpenFlight"																					
"OpenSceneGraph"	Export scene to an OpenSceneGraph file.	"Export to OpenSceneGraph"																					
"USD"	Export scene to a Universal Scene Description (USD) file.	"Export to USD"																					

Name	Type	Description		
		format_ name Option	Descript ion	Export Format Details
		"Apollo "	Export scene to the Baidu Apollo file format.	"Export to Apollo"
		"GeoJSON"	Export scene to a GeoJSON file.	"Export to GeoJSON"
		"OpenDRIVE"	Export scene to an ASAM OpenDRIVE file.	"Export to ASAM OpenDRIVE"
		"OpenScenario"	Export scenario to an ASAM OpenScenario® file.	"Export to ASAM OpenScenario" (RoadRunner Scenario)
		"OpenScenario 2.0"	Export scenario to an ASAM OpenScenario 2.0 file.	"Export to ASAM OpenScenario" (RoadRunner Scenario)
		"CARLA"	Export scene to the CARLA format.	"Export to CARLA"
		"Metamoto"	Export scene to the Metamoto® format.	"Export to Metamoto"
		"Unity"	Export scene to the	"Export to Unity"

Name	Type	Description		
		format_ name Option	Descript ion	Export Format Details
			Unity® format.	
		"Unreal "	Export scene to the Unreal Engine® format.	"Export to Unreal Using Filmbox (.fbx) File"
		"VTD"	Export scene to the VTD format.	"Export to VTD"
<p>You can also specify custom exporters that you define in the <code>ExportConfigurations.xml</code> file. For more details, see "Export Custom Formats".</p>				

Name	Type	Description																								
export_settings (optional)	oneof	<p>Export settings configuration, specified as one of the export settings messages defined in the export_settings.proto file.</p> <p>If you specify export_settings, then the export settings type must be compatible with the format name specified in the format_name field.</p> <table border="1" data-bbox="1063 640 1456 1808"> <thead> <tr> <th data-bbox="1063 640 1193 745">export_settings Field</th> <th data-bbox="1193 640 1323 745">Type</th> <th data-bbox="1323 640 1456 745">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="1063 745 1193 882">auto_cad_settings</td> <td data-bbox="1193 745 1323 882">AutoCadExportSettings message</td> <td data-bbox="1323 745 1456 882">AutoCAD export settings</td> </tr> <tr> <td data-bbox="1063 882 1193 1018">filmbox_settings</td> <td data-bbox="1193 882 1323 1018">FilmboxExportSettings message</td> <td data-bbox="1323 882 1456 1018">Filmbox export settings</td> </tr> <tr> <td data-bbox="1063 1018 1193 1155">gltf_settings</td> <td data-bbox="1193 1018 1323 1155">GltfExportSettings message</td> <td data-bbox="1323 1018 1456 1155">glTF export settings</td> </tr> <tr> <td data-bbox="1063 1155 1193 1291">open_flight_settings</td> <td data-bbox="1193 1155 1323 1291">OpenFlightExportSettings message</td> <td data-bbox="1323 1155 1456 1291">OpenFlight export settings</td> </tr> <tr> <td data-bbox="1063 1291 1193 1428">open_scene_graph_settings</td> <td data-bbox="1193 1291 1323 1428">OpenSceneGraphExportSettings message</td> <td data-bbox="1323 1291 1456 1428">OpenSceneGraph export settings</td> </tr> <tr> <td data-bbox="1063 1428 1193 1564">usd_settings</td> <td data-bbox="1193 1428 1323 1564">UsdExportSettings message</td> <td data-bbox="1323 1428 1456 1564">Universal Scene Description export settings</td> </tr> <tr> <td data-bbox="1063 1564 1193 1808">apollo_settings</td> <td data-bbox="1193 1564 1323 1808">ApolloExportSettings message</td> <td data-bbox="1323 1564 1456 1808">Apollo export settings</td> </tr> </tbody> </table>	export_settings Field	Type	Description	auto_cad_settings	AutoCadExportSettings message	AutoCAD export settings	filmbox_settings	FilmboxExportSettings message	Filmbox export settings	gltf_settings	GltfExportSettings message	glTF export settings	open_flight_settings	OpenFlightExportSettings message	OpenFlight export settings	open_scene_graph_settings	OpenSceneGraphExportSettings message	OpenSceneGraph export settings	usd_settings	UsdExportSettings message	Universal Scene Description export settings	apollo_settings	ApolloExportSettings message	Apollo export settings
export_settings Field	Type	Description																								
auto_cad_settings	AutoCadExportSettings message	AutoCAD export settings																								
filmbox_settings	FilmboxExportSettings message	Filmbox export settings																								
gltf_settings	GltfExportSettings message	glTF export settings																								
open_flight_settings	OpenFlightExportSettings message	OpenFlight export settings																								
open_scene_graph_settings	OpenSceneGraphExportSettings message	OpenSceneGraph export settings																								
usd_settings	UsdExportSettings message	Universal Scene Description export settings																								
apollo_settings	ApolloExportSettings message	Apollo export settings																								

Name	Type	Description		
		export_settings Field	Type	Description
		geo_json_settings	GeoJsonExportSettings message	GeoJSON export settings
		open_drive_settings	OpenDriveExportSettings message	ASAM OpenDRIVE export settings
		open_scenario_settings	OpenScenarioExportSettings message	ASAM OpenSCENARIO export settings
		carla_settings	CarlaExportSettings message	CARLA export settings
		metamoto_settings	MetamotoExportSettings message	Metamoto export settings
		unity_settings	UnityExportSettings message	Unity export settings
		unreal_settings	UnrealExportSettings message	Unreal Engine export settings
		vtd_settings	VtdExportSettings message	VTD export settings

Response

ExportResponse — Export response
empty message

Export response, returned as an empty message.

Sample Calls

Command Line

Export the current scene in project `C:\RR\MyProject` to ASAM OpenDRIVE version 1.6. Export the file to the Exports folder of the current project and name it `MyExportedScene.xodr`.

```
cd "C:\Program Files\RoadRunner R2023a\bin\win64"
AppRoadRunner --projectPath C:\RR\MyProject --apiPort 54321
CmdRoadRunnerApi "Export(file_path='MyExportedScene.xodr' format_name='opendrive' ^
open_drive_settings.open_drive_version='1.6')" --serverAddress localhost:54321
```

This sample call uses the `CmdRoadRunnerApi` helper command, which is a precompiled version of the RoadRunner service API. For examples that use this command, see:

- “Convert Scenes Between Formats Using gRPC API”
- “Export Multiple Scenes Using gRPC API”

Python

Export the current scene in the current project to an ASAM OpenDRIVE version 1.6 file. Export the file to the Exports folder of the current project and name it `MyExportedScene.xodr`.

```
exportRequest = roadrunner_service_messages_pb2.ExportRequest()
exportRequest.file_path = "MyExportedScene.xodr"
exportRequest.format_name = "opendrive"
exportRequest.open_drive_settings.open_drive_version = 1.6
api.Export(exportRequest)
```

This sample call is a snippet of a Python client. For details on creating complete Python clients, see “Create gRPC Python Client for Controlling RoadRunner Programmatically”.

In this sample call, `api` is a Python stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

C++

Export the current scene in the current project to an ASAM OpenDRIVE version 1.6 file. Export the file to the Exports folder of the current project and name it `MyExportedScene.xodr`.

```
ExportRequest request;
std::string filePath = "MyExportedScene.xodr";
std::string formatName = "opendrive";
request.set_file_path(filePath);
request.set_format_name(formatName);
OpenDriveExportSettings * openDRIVESettings(request.mutable_open_drive_settings());
openDRIVESettings->set_open_drive_version(1.6);
ClientContext context;
ExportResponse reply;
Status status = api->Export(&context, request, &reply);
```

This sample call is a snippet of a C++ client. For details on creating complete C++ clients, see “Create gRPC C++ Client for Controlling RoadRunner Programmatically”.

In this sample call, `api` is a C++ stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

Version History

Introduced in R2021b

See Also

NewProject | LoadProject | SaveProject | NewScene | LoadScene | SaveScene | NewScenario | LoadScenario | SaveScenario | SetScenarioVariable | PrepareSimulation | SimulateScenario | Import | Exit | roadrunner_service.proto | roadrunner_service_messages.proto | export_settings.proto

Topics

“Control RoadRunner Programmatically Using gRPC API”
“Convert Scenes Between Formats Using gRPC API”
“Export Multiple Scenes Using gRPC API”

Import

Import file into RoadRunner scene or scenario using gRPC

Description

The `Import` method imports a file that is in a format that RoadRunner supports into the currently loaded scene or scenario. You can import ASAM OpenDRIVE and ASAM OpenSCENARIO files.

This RoadRunner API method is a remote procedure call (RPC) that sends a single message to the RoadRunner API service and receives a single message back as a response. The protocol buffer (protobuf) file `roadrunner_service.proto` defines the schema for this method. Using a gRPC compiler, you can compile the RoadRunner protobuf files into a language supported by gRPC and create client applications to call this method in that language. For more details, see “Compile Protocol Buffers for RoadRunner gRPC API”. For background information on how the RoadRunner API works, see “Control RoadRunner Programmatically Using gRPC API”.

Request

ImportRequest — Import request message

Import request, specified as a message with these fields.

Name	Type	Description
<code>file_path</code> (required)	string	Absolute or relative path to the file to import. If you specify a relative path, then you must specify a path to a file in the Assets folder of the current project.
<code>open_drive_settings</code> (optional)	OpenDRIVEImportSettings message	ASAM OpenDRIVE import settings.
<code>csv_trajectory_settings</code> (optional)	CsvTrajectoryImportSettings message	CSV trajectory import settings.
<code>roadrunner_hd_map_settings</code> (optional)	RoadRunnerHdMapImportSettings message	RoadRunner HD map import settings.

Response

ImportResponse — Import response empty message

Import response, returned as an empty message.

Sample Calls

Command Line

Import an ASAM OpenDRIVE file named `MyOpenDRIVEFile.xodr` from the `Assets` folder of a project located at `C:\RR\MyProject`. Do not import traffic signals from the file.

```
cd "C:\Program Files\RoadRunner R2023a\bin\win64"
AppRoadRunner --projectPath C:\RR\MyProject --apiPort 54321
CmdRoadRunnerApi "Import(file_path='MyOpenDRIVEFile.xodr' ^
open_drive_settings.import_signals='false')" --serverAddress localhost:54321
```

This sample call uses the `CmdRoadRunnerApi` helper command, which is a precompiled version of the RoadRunner service API. For examples that use this command, see:

- “Convert Scenes Between Formats Using gRPC API”
- “Export Multiple Scenes Using gRPC API”

Python

Import an ASAM OpenDRIVE file named `MyOpenDRIVEFile.xodr` from the `Assets` folder of the current project. Do not import traffic signals from the file.

```
importRequest = roadrunner_service_messages_pb2.ImportRequest()
importRequest.file_path = "MyOpenDRIVEFile.xodr"
importRequest.open_drive_settings.import_signals.value = False
api.Import(importRequest)
```

This sample call is a snippet of a Python client. For details on creating complete Python clients, see “Create gRPC Python Client for Controlling RoadRunner Programmatically”.

In this sample call, `api` is a Python stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

C++

Import an ASAM OpenDRIVE file named `MyOpenDRIVEFile.xodr` from the `Assets` folder of the current project. Do not import traffic signals from the file.

```
ImportRequest request;
std::string filePath = "MyOpenDRIVEFile.xodr";
request.set_file_path(filePath);
OpenDriveExportSettings * openDRIVESettings(request.mutable_open_drive_settings());
openDRIVESettings->mutable_import_signals()->set_value(false);
ClientContext context;
ImportResponse reply;
Status status = api->Import(&context, request, &reply);
```

This sample call is a snippet of a C++ client. For details on creating complete C++ clients, see “Create gRPC C++ Client for Controlling RoadRunner Programmatically”.

In this sample call, `api` is a C++ stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

Version History

Introduced in R2021b

R2022b: Import HD Map data programmatically using gRPC

Import HD map data into RoadRunner programmatically using the `RoadRunnerHdMapImportSettings` setting in the gRPC `Import` request. You need a RoadRunner license to load HD maps and a RoadRunner Scene Builder license to both load and build HD maps into editable RoadRunner scenes.

See Also

`NewProject` | `LoadProject` | `SaveProject` | `NewScene` | `LoadScene` | `SaveScene` | `NewScenario` | `LoadScenario` | `SaveScenario` | `SetScenarioVariable` | `PrepareSimulation` | `SimulateScenario` | `Export` | `Exit` | `roadrunner_service.proto` | `roadrunner_service_messages.proto` | `import_settings.proto`

Topics

“Control RoadRunner Programmatically Using gRPC API”

“Convert Scenes Between Formats Using gRPC API”

Exit

Exit RoadRunner application using gRPC

Description

The `Exit` method exits the RoadRunner application and shuts down the RoadRunner gRPC server.

This RoadRunner API method is a remote procedure call (RPC) that sends a single message to the RoadRunner API service and receives a single message back as a response. The protocol buffer (protobuf) file `roadrunner_service.proto` defines the schema for this method. Using a gRPC compiler, you can compile the RoadRunner protobuf files into a language supported by gRPC and create client applications to call this method in that language. For more details, see “Compile Protocol Buffers for RoadRunner gRPC API”. For background information on how the RoadRunner API works, see “Control RoadRunner Programmatically Using gRPC API”.

Request

ExitRequest — Exit request

empty message

Exit request, specified as an empty message.

Response

ExitResponse — Exit response

empty message

Exit response, returned as an empty message.

Sample Calls

Command Line

Exit RoadRunner and shut down the API server.

```
cd "C:\Program Files\RoadRunner R2023a\bin\win64"
AppRoadRunner --projectPath C:\RR\MyProject --apiPort 54321
CmdRoadRunnerApi "Exit()" --serverAddress localhost:54321
```

This sample call uses the `CmdRoadRunnerApi` helper command, which is a precompiled version of the RoadRunner service API. For examples that use this command, see:

- “Convert Scenes Between Formats Using gRPC API”
- “Export Multiple Scenes Using gRPC API”

Python

Exit RoadRunner and shut down the API server.


```
exitRequest = roadrunner_service_messages_pb2.ExitRequest()
api.Exit(exitRequest)
```

This sample call is a snippet of a Python client. For details on creating complete Python clients, see “Create gRPC Python Client for Controlling RoadRunner Programmatically”.

In this sample call, `api` is a Python stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

C++

Exit RoadRunner and shut down the API server.

```
ExitRequest request;
ClientContext context;
ExitResponse reply;
Status status = api->Exit(&context, request, &reply);
```

This sample call is a snippet of a C++ client. For details on creating complete C++ clients, see “Create gRPC C++ Client for Controlling RoadRunner Programmatically”.

In this sample call, `api` is a C++ stub of the RoadRunner service API. For details on generating these stubs, see “Compile Protocol Buffers for RoadRunner gRPC API”.

Version History

Introduced in R2021b

See Also

NewProject | LoadProject | SaveProject | NewScene | LoadScene | SaveScene | NewScenario
| LoadScenario | SaveScenario | SetScenarioVariable | PrepareSimulation |
SimulateScenario | Export | Import | roadrunner_service.proto |
roadrunner_service_messages.proto

Topics

“Control RoadRunner Programmatically Using gRPC API”

Objects

roadrunner

Start RoadRunner application using MATLAB

Description

A `roadrunner` object enables you to perform common workflow tasks in the RoadRunner application, such as opening, closing, and saving scenes and projects, from the MATLAB command line. You can also use object functions to import data from files and export scenes from RoadRunner to other formats.

Before you create a `roadrunner` object for the first time, you must install RoadRunner and activate your RoadRunner license interactively. For more information, see “Install and Activate RoadRunner”.

The `roadrunner` object requires a license for Automated Driving Toolbox™.

Creation

Syntax

```
rrApp = roadrunner(projectFolder)
rrApp = roadrunner(projectFolder,Name=Value)
```

Description

`rrApp = roadrunner(projectFolder)` starts RoadRunner from the default installation folder location, and opens a new scene in an existing project at the location specified by `projectFolder`.

`rrApp = roadrunner(projectFolder,Name=Value)` also uses a name-value argument to set the Properties of the `roadrunner` object.

Input Arguments

projectFolder — RoadRunner project folder path

character vector | string scalar

RoadRunner project folder path, specified as a character vector or string scalar. For details about the RoadRunner project folder structure, see “RoadRunner Project and Scene System”.

Example: `roadrunner("C:\My Project")` opens the project located in the C:\My Project folder in the RoadRunner application on a Windows machine.

Data Types: `char` | `string`

Properties

InstallationFolder — Location of local RoadRunner installation folder

character vector | string scalar

Location of local RoadRunner installation folder, specified as a character vector or string scalar.

These are the default RoadRunner installation locations on Windows and Linux platforms:

- Windows - C:\Program Files\RoadRunner *R20NNx*\bin\win64
- Linux, Ubuntu - /usr/local/RoadRunner_*R20NNx*/bin/glnxa64

R20NNx is the release version you are using. To customize the default value of the RoadRunner installation folder, use the MATLAB settings API.

Data Types: `char` | `string`

NoDisplay — Start RoadRunner in console mode

`false` or `0` (default) | `true` or `1`

Start RoadRunner in console mode in a non-graphical environment, specified as `logical 0` (`false`) or `logical 1` (`true`). For more details on starting RoadRunner in console mode, see “Control RoadRunner Programmatically in Console Mode” .

Data Types: `logical`

Ports — Start RoadRunner on specified apiPort and cosimPort

`matrix`

Start RoadRunner on specified `apiPort` and `cosimPort`, specified as a matrix of size 1-by-2. RoadRunner API server port (`apiPort`) and RoadRunner Scenario simulation API server port (`cosimPort`) are automatically assigned by default when you launch the RoadRunner application. The server runs locally at `localhost:port` after you open a RoadRunner project. If the ports assigned are not free, a connection error occurs. In this case, you can explicitly assign values to `apiPort` and `cosimPort` using the `roadrunner` object or the `connect` function.

Example: `rrApp= roadrunner(projectFolder,Ports= [54321, 54322])` starts RoadRunner and RoadRunner Scenario on `apiPort` and `cosimPort` 54321 and 54322 respectively.

Data Types: `int`

Object Functions

Project

<code>newProject</code>	Create new RoadRunner project using MATLAB
<code>openProject</code>	Open RoadRunner project using MATLAB
<code>saveProject</code>	Save RoadRunner project using MATLAB

Scenes

<code>newScene</code>	Create new RoadRunner scene using MATLAB
<code>openScene</code>	Open RoadRunner scene using MATLAB
<code>saveScene</code>	Save RoadRunner scene using MATLAB
<code>exportScene</code>	Export RoadRunner scene using MATLAB
<code>exportCustomFormat</code>	Export RoadRunner scene to custom format using MATLAB
<code>importScene</code>	Import scene into RoadRunner using MATLAB

Scenarios

<code>newScenario</code>	Create new scenario in RoadRunner Scenario using MATLAB
<code>openScenario</code>	Open scenario in RoadRunner Scenario using MATLAB

saveScenario	Save scenario in RoadRunner Scenario using MATLAB
exportScenario	Export scenario from RoadRunner Scenario using MATLAB
importScenario	Import file into RoadRunner Scenario using MATLAB
createSimulation	Create RoadRunner Scenario simulation using MATLAB
getScenarioVariable	Get the value of RoadRunner scenario variable using MATLAB
setScenarioVariable	Set RoadRunner scenario variable using MATLAB
remapAnchor	Remap road anchor in RoadRunner Scenario in MATLAB

Connect, Close and Status

roadrunner.connect	Connect to open instance of RoadRunner using MATLAB
close	Close RoadRunner using MATLAB
status	Get current status of RoadRunner using MATLAB

Examples

Start RoadRunner Application from Default Installation Folder

Specify the path to an existing project using the `projectFolder` variable. For example, this code shows the path to a project, on a Windows® machine, located at `C:\RR\MyProject`.

```
projectFolder = "C:\RR\MyProject";
```

Create a `roadrunner` object and open RoadRunner by specifying your project as the location where you want to create a scene. This example assumes that RoadRunner is installed in its default location.

```
rrApp = roadrunner(projectFolder);
```

Change Default RoadRunner Installation Folder Settings

Modify the default RoadRunner installation folder settings by using the MATLAB `settings` function.

You can specify a value for the `InstallationFolder` property of a `roadrunner` object that persists across MATLAB sessions, or for a given MATLAB session, using the `settings` function.

To set a persistent default value for the `InstallationFolder` property, edit the value of `PersonalValue`. The specified value persists across multiple MATLAB sessions for an individual user.

```
s = settings;  
s.roadrunner.application.InstallationFolder.PersonalValue = "C:\Program Files\RoadRunner R2022a\
```

You can also set a temporary default value for the `InstallationFolder` property. The specified value persists for only the current MATLAB session, and is cleared at the end of the session.

```
s = settings;  
s.roadrunner.application.InstallationFolder.TemporaryValue = "C:\MyRoadRunner Install\bin\win64"
```

Start Multiple RoadRunner Application Instances Simultaneously

Open multiple RoadRunner applications by creating multiple instances of the `roadrunner` object. You can use each instance to programmatically interact with the RoadRunner application that it opens. The scenes used in this code are included in RoadRunner projects by default.

Open a first instance of RoadRunner by specifying the `Lane Keep` project. In this example, the project is located on the path `C:\RR\Lane Keep`.

```
rrApp1 = roadrunner("C:\RR\Lane Keep");
```

Open the `FourWaySignal` scene in the first project.

```
openScene(rrApp1, "FourWaySignal.rrscene");
```

Open a second instance of RoadRunner by specifying the `USCity` project. In this example, the project is located on the path `C:\RR\USCity`.

```
rrApp2 = roadrunner("C:\RR\USCity");
```

Open the `SanAntonio` scene in the second project

```
openScene(rrApp2, "SanAntonio.rrscene");
```

Start RoadRunner Application in Console Mode

Specify the path to an existing project using the `projectFolder` variable. For example, this code shows the path to a project, on a Windows® machine, located at `C:\RR\MyProject`.

```
projectFolder = "C:\RR\MyProject";
```

Create a `roadrunner` object and open RoadRunner by specifying your project as the location where you want to create a scene. This example assumes that RoadRunner is installed in its default location. Specify the `NoDisplay` property to launch the application in console mode using a non-graphical terminal.

```
rrApp = roadrunner(projectFolder, InstallationFolder="C:\Program Files\RoadRunner R2022b\bin\win64");
```

Limitations

- The `roadrunner` object and its associated functions are supported only in RoadRunner R2022a and later.
- The project you specify using the `projectFolder` argument, when creating the `roadrunner` object must already exist.
- The `projectFolder` argument and “`InstallationFolder`” on page 4-0 property do not support Unicode characters in the path.

Tips

- Deleting the `roadrunner` object from the MATLAB workspace does not close the RoadRunner application associated with it. You must manually close the RoadRunner application. Also, exiting

from the current MATLAB session does not close any RoadRunner applications created by `roadrunner`.

Version History

Introduced in R2022a

See Also

`roadrunner.connect` | `newProject` | `openProject` | `newScenario` | `openScenario` | `close`

Topics

“RoadRunner Project and Scene System”

“RoadRunner Scenario Fundamentals” (RoadRunner Scenario)

“Export Multiple Scenes Using MATLAB”

“Convert Scenes Between Formats Using MATLAB Functions”

roadrunner.connect

Connect to open instance of RoadRunner using MATLAB

Syntax

```
rrApp = roadrunner.connect  
rrApp = roadrunner.connect(apiPort)  
rrApp = roadrunner.connect(apiPort,cosimPort)
```

Description

`rrApp = roadrunner.connect` returns an `rrApp` object for the most recently opened instance of RoadRunner.

`rrApp = roadrunner.connect(apiPort)` returns an `rrApp` object for an open RoadRunner application running an API server on the specified IP network port, `apiPort`.

`rrApp = roadrunner.connect(apiPort,cosimPort)` returns an `rrApp` object for an open RoadRunner application. It optionally specifies the scenario simulation API server port using `cosimPort`.

Examples

Connect to Open RoadRunner Instance

Connect to the last opened instance of RoadRunner application using MATLAB.

Use the `connect` function to connect to the last opened instance of RoadRunner. This call returns an object `rrApp` that provides functions for performing basic workflow tasks such as opening, closing, and saving scenes and projects.

```
rrApp = roadrunner.connect();
```

Connect to RoadRunner Instance on Specified IP Network Port

Connect to an open instance of RoadRunner application running on the specified IP Network Port.

Call the `connect` function and pass it the `apiPort` as an argument. This call returns an `rrApp` object for an open RoadRunner application running on API server on the specified IP network port.

```
apiPort = 28703;  
rrApp = roadrunner.connect(apiPort);
```

Connect to RoadRunner Instance Using IP Network and Scenario Simulation Ports

Connect to an open instance of RoadRunner application running on the specified IP network and scenario simulation API server ports.

Call the `connect` function and pass it the `apiPort` and `cosimPort` as arguments. The `apiPort` specifies the IP network port and the `cosimPort` specifies the scenario simulation API server port respectively for the RoadRunner application. This call returns an `rrApp` object for an open RoadRunner application running on API server on the specified IP network and scenario simulation ports.

```
apiPort = 35707;  
cosimPort = 35706;  
rrApp = roadrunner.connect(apiPort,cosimPort);
```

Input Arguments

apiPort — IP network port for RoadRunner API server

35707 (default)

IP network port for RoadRunner API server, specified as a default value of 35707. This server receives commands from the RoadRunner service methods for importing and exporting scenes and scenarios. The API server runs locally at `localhost:apiPort` after you open a RoadRunner project. Specify `apiPort` as an integer in the range [1024, 65535]. If `apiPort` is unspecified, a default value of 35707 is used for connecting.

Example: `roadrunner.connect(54321)`

Data Types: `int`

cosimPort — IP network port for RoadRunner Scenario co-simulation server

35706 (default)

IP network port for RoadRunner Scenario co-simulation server, specified as a default value of 35706. This server receives commands for co-simulation with MATLAB and Simulink and with external simulators such as CARLA (requires RoadRunner Scenario). Specify `cosimPort` as an integer in the range [1024, 65535]. If `apiPort` is unspecified, a default value of 35706 is used for connecting.

Example: `roadrunner.connect(54321,54322)`

Data Types: `int`

Output Arguments

rrApp — RoadRunner application

`roadrunner` object

RoadRunner application associated with a project, returned as a `roadrunner` object. This object provides functions for performing common workflow tasks such as opening, closing, and saving scenes and projects. `rrApp` provides functions that support importing data from files and exporting scenes to other formats from RoadRunner.

Version History

Introduced in R2022a

See Also

roadrunner | newProject | newScene | newScenario

Topics

“Export Multiple Scenes Using MATLAB”

“Convert Scenes Between Formats Using MATLAB Functions”

close

Close RoadRunner using MATLAB

Syntax

```
close(rrApp)
```

Description

`close(rrApp)` closes the specified RoadRunner instance `rrApp` from the MATLAB command line.

Examples

Open and Close RoadRunner Instance

Open a project in RoadRunner using the `roadrunner` function by specifying the location in which to create a project. This example assumes that RoadRunner is installed in its default location in Windows.

Specify the path to an existing project. For example, this code shows the path to a project located on `C:\RR\MyProject`. The function returns a `roadrunner` object, `rrApp`, that provides functions for performing basic workflow tasks such as opening, closing, and saving scenes and projects.

```
projectFolder = "C:\RR\MyProject";  
rrApp = roadrunner(projectFolder);
```

Close the RoadRunner instance `rrApp`.

```
close(rrApp);
```

Input Arguments

rrApp — RoadRunner application

`roadrunner` object

RoadRunner application associated with a project, specified as a `roadrunner` object. This object provides functions for performing common workflow tasks such as opening, closing, and saving scenes and projects. `rrApp` provides functions that support importing data from files and exporting scenes to other formats from RoadRunner.

Tips

- If you open a RoadRunner instance using a `roadrunner` object and close it interactively, the object continues to exist in the MATLAB workspace. You must delete it manually.
- Function calls made to a `roadrunner` object after closing the associated RoadRunner instance return an error.

- Closing a RoadRunner instance by using the `close` function deletes the associated `roadrunner` object.

Version History

Introduced in R2022a

See Also

`roadrunner` | `newProject` | `newScene` | `newScenario`

Topics

“Export Multiple Scenes Using MATLAB”

“Convert Scenes Between Formats Using MATLAB Functions”

exportCustomFormat

Export RoadRunner scene to custom format using MATLAB

Syntax

```
exportCustomFormat(rrApp, filename, formatname)
```

Description

`exportCustomFormat(rrApp, filename, formatname)` exports the current RoadRunner scene to a custom format specified using the `ExportConfigurations.xml` file. To specify a custom configuration, follow these steps:

- 1 Create an XML file that configures the details of the custom export format.
- 2 Save the XML file to the Project folder of the RoadRunner project that you want to contain the export option and name it `ExportConfigurations.xml`.

For more details on creating the `ExportConfigurations.xml` file, see “Export Custom Formats”.

Examples

Export RoadRunner Scene using Custom Format

Open a project in RoadRunner using the `roadrunner` function by specifying the location in which to create a project. This example assumes that RoadRunner is installed in its default location in Windows.

Specify the path to an existing project. For example, this code shows the path to a project located on `C:\RR\MyProject`. The function returns a `roadrunner` object, `rrApp`, that provides functions for performing basic workflow tasks such as opening, closing, and saving scenes and projects.

```
projectFolder = "C:\RR\MyProject";  
rrApp = roadrunner(projectFolder);
```

Open an existing scene in RoadRunner.

```
filename = "FourWaySignal.rrscene";  
openScene(rrApp, filename);
```

Create an `ExportConfigurations.xml` file to configure custom export format. Then, use the `exportCustomFormat` function to export the currently open RoadRunner scene to the custom format specified in the `ExportConfigurations.xml` file.

```
customfilename = "customExport.zip";  
formatname = "My Custom Format";  
exportCustomFormat(rrApp, customfilename, formatname);
```

Input Arguments

rrApp — RoadRunner application

roadrunner object

RoadRunner application associated with a project, specified as a `roadrunner` object. This object provides functions for performing common workflow tasks such as opening, closing, and saving scenes and projects. `rrApp` provides functions that support importing data from files and exporting scenes to other formats from RoadRunner.

filename — File path for exported RoadRunner scene

character vector | string scalar

File path for exported RoadRunner scene, specified as a character vector or string scalar. This argument specifies the absolute or relative file path for to the exported file. If you specify a relative path, then the exported file is saved relative to the `Exports` folder of the current project. If any folders in the specified path do not exist, RoadRunner tries to create them. `filename` must include the extension of the exported file.

Example: "customExport.zip"

Data Types: `char` | `string`

formatname — Custom export format name

character vector | string scalar

Custom export format name, specified as a character vector or string scalar. This argument specifies the configuration name of your custom export format in the `ExportConfigurations.xml` file. The format name is case sensitive.

Example: "My Custom Format"

Data Types: `char` | `string`

Version History

Introduced in R2022a

See Also

`roadrunner` | `exportScene` | `close`

Topics

"Export Multiple Scenes Using MATLAB"

"Convert Scenes Between Formats Using MATLAB Functions"

exportScene

Export RoadRunner scene using MATLAB

Syntax

```
exportScene(rrApp, filename, formatname)
exportScene(rrApp, filename, formatname, exportoptions)
```

Description

`exportScene(rrApp, filename, formatname)` exports the RoadRunner scene file to one of the file formats supported by RoadRunner.

`exportScene(rrApp, filename, formatname, exportoptions)` specifies an export option configuration, `exportoptions`.

Examples

Export Scene from RoadRunner to ASAM OpenDRIVE®

Export a scene from a RoadRunner project to the ASAM OpenDRIVE format using MATLAB®.

Open a project in RoadRunner using the `roadrunner` function by specifying the location in which to create a project. This example assumes that RoadRunner is installed in its default location in Windows.

Specify the path to an existing project. For example, this code shows the path to a project located on `C:\RR\MyProject`. The function returns a `roadrunner` object, `rrApp`, that provides functions for performing basic workflow tasks such as opening, closing, and saving scenes and projects.

```
projectFolder = "C:\RR\MyProject";
rrApp = roadrunner(projectFolder);
```

Open a scene in the project by using the `openScene` function with the `roadrunner` object and the RoadRunner scene you wish to open as input arguments. This example uses the `FourWaySignal.rrscene` scene, which is one of the scenes included by default in RoadRunner projects, and is located in the `Scenes` folder of the project.

```
sceneName = "FourWaySignal.rrscene";
openScene(rrApp, sceneName);
```

Set export options by creating an `openDriveExportOptions` object to enable export of signals and objects from the file.

```
options = openDriveExportOptions(OpenDriveVersion=1.5, ExportSignals=true, ExportObjects=true);
```

Use the `exportScene` function to export the scene to ASAM OpenDRIVE. Specify your `roadrunner` object, the name of the file to which you want to export the scene, the export format, and the export options as input arguments to the `exportScene` function.


```
filename = "FourWaySignal.xodr";
formatname = "OpenDRIVE";
exportScene(rrApp, filename, formatname, options);
```

Input Arguments

rrApp — RoadRunner application

roadrunner object

RoadRunner application associated with a project, specified as a roadrunner object. This object provides functions for performing common workflow tasks such as opening, closing, and saving scenes and projects. rrApp provides functions that support importing data from files and exporting scenes to other formats from RoadRunner.

filename — File path for the exported RoadRunner scene

character vector | string scalar

File path for the exported RoadRunner scene, specified as a character vector or string scalar. This argument specifies the absolute or relative file path of the file to which you are exporting the scene from the RoadRunner project. If you specify a relative path, then the path is relative to the Exports folder of the current project. If you specify an absolute path, then RoadRunner exports the scene to a file in the exact location specified. If any folders in the path are missing, RoadRunner tries to create them. filename can include the extension for the exported file or have no extension. If it has no extension, then RoadRunner appends the extension of the format specified by the formatname to the file name before exporting the scene.

Example: `exportScene(rrApp, "FourWaySignal.xodr", "OpenDRIVE", options)`, "FourWaySignal.xodr" represents the file name of the exported file, which is relative to the Exports folder of the current project.

Data Types: char | string

formatname — Export format name

character vector | string scalar

Export format name, specified as a character vector or string scalar. This argument specifies the format name corresponding to a valid RoadRunner format. Format name options are case-insensitive. Supported formats are: AutoCAD, Filmbox, glTF, OpenFlight, OpenSceneGraph, ASAM OpenDRIVE, USD, Apollo, CARLA, Metamoto, Unity, Unreal®, GeoJSON, and VTD.

Example: While calling `exportScene(rrApp, "FourWaySignal.xodr", "OpenDRIVE", options)`, "OpenDRIVE" specifies that the file will be exported to ASAM OpenDRIVE format.

Data Types: char | string

exportoptions — Export options configuration

valid export options object

Export options configuration, specified as one of the export options objects compatible with the specified format formatname. This argument specifies the options used when exporting the specified scene.

Export Formats	Description	Properties	
apolloExportOptions	<p>Options for exporting a RoadRunner scene to Apollo.</p> <p>apolloExportOptions (Name=Value) creates an export options configuration object for the Apollo format, with properties specified as one or more name-value arguments.</p> <p>If a default property value is "auto", the RoadRunner application determines what value to use and sets the property to that value.</p>	ApolloVersion	<p>Apollo version to export to, specified as 3 or 5, respectively.</p> <p>Default: "auto"</p>
		DatabaseVersion	<p>Identifier for the exported scene, specified as a numeric scalar. This property is useful for versioning exports of the same scene.</p> <p>Default: "auto"</p>
		DatabaseName	<p>Name of the exported scene, specified as a string scalar or character vector.</p> <p>Default: "auto"</p>
		DrivingSide	<p>Driving side of the exported scene, specified as: "Unspecified", "Left", or "Right".</p> <p>Default: "auto"</p>
		ExportSignals	<p>Export all signals and signs mapped to junctions as <signal> entries, specified as a logical 1</p>

Export Formats	Description	Properties	
			(true) or 0 (false). Default: "auto"
		ExportObjects	Export all props as <object> entries, specified as a logical 1 (true) or 0 (false). Default: "auto"
		ClampDistances	Clamp distances in the RoadRunner scene to multiples of 1 cm, specified as a logical 1 (true) or 0 (false). Specifying this property as true can prevent RoadRunner from exporting very short roads. Default: "auto"
		Example: options = apolloExportOptions(DrivingSide="Right");	

Export Formats	Description	Properties	
autocadExportOptions	Options for exporting a RoadRunner scene to AutoCAD. autocadExportOptions(Name=Value) creates an export options configuration object for the AutoCAD format, with properties specified as one or more name-value arguments. If a default property value is "auto", the RoadRunner application determines what value to use and sets the property to that value.	SplitMeshes	Split meshes by their segmentation type, specified as a logical 1 (true) or 0 (false). Default: "auto"
		ResizeTextures	Resize the dimensions of exported textures by rounding them up to the next highest power of two, specified as a logical 1 (true) or 0 (false). Default: "auto"
		TilesExportOptions	Options for splitting meshes per tile, specified as a tilesExportOptions object. Default: "auto"
		Example: options = autocadExportOptions(SplitMeshes=true);	

Export Formats	Description	Properties	
tilesExportOptions	<p>Options for exporting tiles.</p> <p>tilesExportOptions(Name=Value) creates an options configuration object for exporting tiles, with properties specified as one or more name-value arguments. These fields correspond to options in the AutoCAD, Filmbox, glTF, OpenFlight, OpenSceneGraph, and USD export options.</p> <p>If a default property value is "auto", the RoadRunner application determines what value to use and sets the property to that value.</p>	TileSize	<p>The size of exported tiles, specified as a two element double vector.</p> <p>Default: "auto"</p>
		TileCenter	<p>The center location of exported tiles, specified as a two element double vector.</p> <p>Default: "auto"</p>
		ExportIndividualTiles	<p>Export each tile as a separate file, specified as a logical 1 (true) or 0 (false).</p> <p>Default: "auto"</p>
		<p>Example: options = tilesExportOptions(TileSize=[100 100]);</p>	

Export Formats	Description	Properties	
carlaExportOptions	<p>Options for exporting a RoadRunner scene to CARLA.</p> <p><code>carlaExportOptions(Name=Value)</code> creates an export options configuration object for the CARLA format, with properties specified as one or more name-value arguments.</p> <p>If a default property value is "auto", the RoadRunner application determines what value to use and sets the property to that value.</p>	<p>UnrealDatasmithOptions</p>	<p>Options for exporting a UnrealDatasmith file, specified as a <code>unrealDatasmithExportOptions</code> object.</p> <p>Default: "auto"</p>
		<p>OpenDriveOptions</p>	<p>Options for exporting an ASAM OpenDRIVE file, specified as an <code>openDriveExportOptions</code> object.</p> <p>Default: "auto"</p>
		<p>Example: <code>options = carlaExportOptions(UnrealDatasmithOptions=unrealDatasmithExportOptions(SplitMeshes=true));</code></p>	

Export Formats	Description	Properties	
carlaFilmboxExportOptions	<p>Options for exporting a RoadRunner scene to CARLA.</p> <p><code>carlaFilmboxExportOptions(Name=Value)</code> creates an export options configuration object for the CARLA format, with properties specified as one or more name-value arguments.</p> <p>If a default property value is "auto", the RoadRunner application determines what value to use and sets the property to that value.</p>	OpenDriveOptions Options for exporting an ASAM OpenDRIVE file, specified as an <code>openDriveExportOptions</code> object. Default: "auto"	
		FilmboxOptions Options for exporting a Filmbox file, specified as a <code>filmboxExportOptions</code> object. Default: "auto"	
		Example: <code>options = carlaFilmboxExportOptions(FilmboxOptions=filmboxExportOptions(SplitMeshes=true));</code>	

Export Formats	Description	Properties	
<p>datasmithRoadExportOptions</p>	<p>Options for exporting a RoadRunner scene to DatasmithRoad.</p> <p><code>datasmithRoadExportOptions(Name=Value)</code> creates an export options configuration object for the DatasmithRoad format, with properties specified as one or more name-value arguments.</p> <p>If a default property value is "auto", the RoadRunner application determines what value to use and sets the property to that value.</p>	<p>UnrealDatasmithOptions</p>	<p>Options for exporting a Unreal Datasmith file, specified as a <code>unrealDatasmithExportOptions</code> object.</p> <p>Default: "auto"</p>
		<p>OpenDriveOptions</p>	<p>Options for exporting an ASAM OpenDRIVE file, specified as an <code>openDriveExportOptions</code> object.</p> <p>Default: "auto"</p>
		<p>Example: <code>options = datasmithRoadExportOptions(UnrealDatasmithOptions=unrealDatasmithExportOptions(SplitMeshes=true));</code></p>	

Export Formats	Description	Properties	
filmboxExportOptions	<p>Options for exporting a RoadRunner scene to Filmbox.</p> <p>filmboxExportOptions(Name=Value) creates an export options configuration object for the Filmbox format, with properties specified as one or more name-value arguments.</p> <p>If a default property value is "auto", the RoadRunner application determines what value to use and sets the property to that value.</p>	SplitMeshes	Split meshes by their segmentation type, specified as a logical 1 (true) or 0 (false). Default: "auto"
		ResizeTextures	Resize the dimensions of exported textures by rounding them up to the next highest power of two, specified as a logical 1 (true) or 0 (false). Default: "auto"
		EmbedTextures	Embed the exported textures inside the exported file, specified as a logical 1 (true) or 0 (false). Default: "auto"
		TilesExportOptions	Options to split meshes per tile, specified as a tilesExportOptions object. Default: "auto"
		Example: options = filmboxExportOptions(Res	

Export Formats	Description	Properties	
		izeTextureDimensions=true);	
geoJSONExportOptions	<p>Options for exporting a RoadRunner scene to GeoJSON.</p> <p>geoJSONExportOptions(Name=Value) creates an export options configuration object for the GeoJSON format, with properties specified as one or more name-value arguments.</p> <p>If a default property value is "auto", the RoadRunner application determines what value to use and sets the property to that value.</p>	ReduceFileSize	<p>Removes new lines from the exported GeoJSON file and decreases its size, specified as a logical 1 (true) or 0 (false).</p> <p>Default: "auto"</p>
		<p>Example: options = geoJSONExportOptions(ReduceFileSize=true);</p>	

Export Formats	Description	Properties	
glTFExportOptions	<p>Options for exporting a RoadRunner scene to glTF.</p> <p><code>glTFExportOptions(Name=Value)</code> creates an export options configuration object for the glTF format, with properties specified as one or more name-value arguments.</p> <p>If a default property value is "auto", the RoadRunner application determines what value to use and sets the property to that value.</p>	SplitMeshes	<p>Split meshes by their segmentation type, specified as a logical 1 (true) or 0 (false).</p> <p>Default: "auto"</p>
		ResizeTextures	<p>Resize the dimensions of exported textures by rounding them up to the next highest power of two, specified as a logical 1 (true) or 0 (false).</p> <p>Default: "auto"</p>
		TilesExportOptions	<p>Options for splitting split meshes per tile, specified as a <code>TilesExportOptions</code> object.</p> <p>Default: "auto"</p>
		<p>Example: <code>options = glTFExportOptions(SplitMeshes=true);</code></p>	

Export Formats	Description	Properties	
<p>metamotoExportOptions</p>	<p>Options for exporting a RoadRunner scene to Metamoto.</p> <p>metamotoExportOptions(Name=Value) creates an export options configuration object for the Metamoto format, with properties specified as one or more name-value arguments.</p> <p>If a default property value is "auto", the RoadRunner application determines what value to use and sets the property to that value.</p>	<p>FilmboxOptions</p>	<p>Options for exporting a Filmbox file, specified as a filmboxExportOptions object.</p> <p>Default: "auto"</p>
		<p>OpenDriveOptions</p>	<p>Options for exporting an ASAM OpenDRIVE file, specified as an openDriveExportOptions object.</p> <p>Default: "auto"</p>
		<p>GeoJSONOptions</p>	<p>Options for exporting a GeoJSON file, specified as a geoJSONExportOptions object.</p> <p>Default: "auto"</p>
		<p>Example: options = metamotoExportOptions(GeoJSONOptions=geoJSONExportOptions(ReduceFileSize=true));</p>	

Export Formats	Description	Properties	
openDriveExportOptions	<p>Options for exporting a RoadRunner scene to ASAM OpenDRIVE.</p> <p><code>openDriveExportOptions(Name=Value)</code> creates an export options configuration object for the ASAM OpenDRIVE format, with properties specified as one or more name-value arguments.</p> <p>If a default property value is "auto", the RoadRunner application determines what value to use and sets the property to that value.</p>	OpenDriveVersion	<p>ASAM OpenDRIVE version to export to, specified as 1.4, 1.5, or 1.6, specified as a scalar double.</p> <p>Default: 1.6</p>
		DatabaseVersion	<p>Identifier for the exported scene, specified as a numeric scalar. This property is useful for versioning exports of the same scene.</p> <p>Default: 0.</p>
		DatabaseName	<p>Name of the exported scene, specified as a string scalar or character vector.</p> <p>Default: "auto"</p>
		DrivingSide	<p>Driving side of the exported scene, specified as: "Unspecified", "Left", or "Right".</p> <p>Default: "auto"</p>
		ExportMarkingsAsLine	<p>Export additional lane marking data (spacing, dash length, and individual paint strip</p>

Export Formats	Description	Properties	
			widths) using the <line> definition in ASAM OpenDRIVE, specified as a logical 1 (true) or 0 (false). Default: "auto"
		ExportSignals	Export all signals and signs mapped to junctions as <signal> entries, specified as a logical 1 (true) or 0 (false). Default: "auto"
		ExportOpenCRG	Export to ASAM OpenCRG file, specified as a logical 1 (true) or 0 (false). Default: "auto"
		ExportObjects	Export all props as <object> entries, specified as a logical 1 (true) or 0 (false). Default: "auto"
		ExportHeadingOffsetRel	Export the <hoffset> (heading

Export Formats	Description	Properties	
		ativeToOrientation	offset) values of <signal> entries as being relative to <orientation>, which is the direction of travel of the road that the signal applies to, specified as a logical 1 (true) or 0 (false). Default: "auto"
		ExportConflictPoints	Export an <object> entry for every point in a junction where two roads intersect, specified as a logical 1 (true) or 0 (false). Default: "auto"
		ExportSceneOriginReference	Export a reference point origin at 0,0 in the scene, specified as a logical 1 (true) or 0 (false). Default: "auto"
		ClampDistances	Clamp distances in the RoadRunner scene to multiples of 1 cm, specified

Export Formats	Description	Properties	
			<p>as a logical 1 (true) or 0 (false). Specifying this property as true can prevent RoadRunner from exporting very short roads.</p> <p>Default: "auto"</p>
		<p>Example: options = openDriveExportOptions(ExportSignals=true);</p>	

Export Formats	Description	Properties	
openFlightExportOptions	<p>Options for exporting a RoadRunner scene to OpenFlight.</p> <p><code>openFlightExportOptions(Name=Value)</code> creates an export options configuration object for the OpenFlight format, with properties specified as one or more name-value arguments.</p> <p>If a default property value is "auto", the RoadRunner application determines what value to use and sets the property to that value.</p>	SplitMeshes	<p>Split meshes by their segmentation type, specified as a logical 1 (true) or 0 (false).</p> <p>Default: "auto"</p>
		ResizeTextu reDimension s	<p>Resize the dimensions of exported textures by rounding them up to the next highest power of two, specified as a logical 1 (true) or 0 (false).</p> <p>Default: "auto"</p>
		TilesExport Options	<p>Options for splitting meshes per tile, specified as a <code>tilesExportOptions</code> object.</p> <p>Default: "auto"</p>
		<p>Example: <code>options = openFlightExportOptions(SplitMeshes=true);</code></p>	

Export Formats	Description	Properties	
<p>openSceneGraphExportOptions</p>	<p>Options for exporting a RoadRunner scene to OpenSceneGraph.</p> <p>openSceneGraphExportOptions (Name=Value) creates an export options configuration object for the OpenSceneGraph format, with properties specified as one or more name-value arguments.</p> <p>If a default property value is "auto", the RoadRunner application determines what value to use and sets the property to that value.</p>	SplitMeshes	<p>Split meshes by their segmentation type, specified as a logical 1 (true) or 0 (false).</p> <p>Default: "auto"</p>
		ResizeTextures	<p>Resize the dimensions of exported textures by rounding them up to the next highest power of two, specified as a logical 1 (true) or 0 (false).</p> <p>Default: "auto"</p>
		EmbedTextures	<p>Embed the exported textures inside the exported file, specified as a logical 1 (true) or 0 (false).</p> <p>Default: "auto"</p>
		TilesExportOptions	<p>Options for splitting meshes per tile, specified as a tilesExportOptions object.</p> <p>Default: "auto"</p>

Export Formats	Description	Properties	
		Example: options = openSceneGraphExportOptions(SplitMeshes=true);	
unityExportOptions	<p>Options for exporting a RoadRunner scene to Unity.</p> <p>unityExportOptions(Name=Value) creates an export options configuration object for the Unity format, with properties specified as one or more name-value arguments.</p> <p>If a default property value is "auto", the RoadRunner application determines what value to use and sets the property to that value.</p>	FilmboxOptions	<p>Options for exporting a Filmbox file, specified as a filmboxExportOptions object.</p> <p>Default: "auto"</p>
		OpenDriveOptions	<p>Options for exporting an ASAM OpenDRIVE file, specified as an openDriveExportOptions object.</p> <p>Default: "auto"</p>
		Example: options = unityExportOptions(FilmboxOptions=filmboxExportOptions(EmbedTextures=true));	

Export Formats	Description	Properties	
<p>unrealExportOptions</p>	<p>Options for exporting a RoadRunner scene to Unreal.</p> <p>unrealExportOptions (Name=Value) creates an export options configuration object for the Unreal format, with properties specified as one or more name-value arguments.</p> <p>If a default property value is "auto", the RoadRunner application determines what value to use and sets the property to that value.</p>	<p>FilmboxOptions</p>	<p>Options for exporting a Filmbox file, specified as a filmboxExportOptions object.</p> <p>Default: "auto"</p>
		<p>OpenDriveOptions</p>	<p>Options for exporting an ASAM OpenDRIVE file, specified as an openDriveExportOptions object.</p> <p>Default: "auto"</p>
		<p>Example: options = unrealExportOptions(FilmboxOptions=filmboxExportOptions(EmbedTextures=true));</p>	

Export Formats	Description	Properties	
unrealDatasmithExportOptions	<p>Options for exporting a RoadRunner scene to UnrealDatasmith.</p> <p>unrealDatasmithExportOptions(Name=Value) creates an export options configuration object for the UnrealDatasmith format, with properties specified as one or more name-value arguments.</p> <p>If a default property value is "auto", the RoadRunner application determines what value to use and sets the property to that value.</p>	SplitMeshes	<p>Split meshes by their segmentation type, specified as a logical 1 (true) or 0 (false).</p> <p>Default: "auto"</p>
		ResizeTextureDimensions	<p>Resize the dimensions of exported textures by rounding them up to the next highest power of two, specified as a logical 1 (true) or 0 (false).</p> <p>Default: "auto"</p>
		TilesExportOptions	<p>Options for splitting meshes per tile, specified as a tilesExportOptions object.</p> <p>Default: "auto"</p>
		<p>Example: options = unrealDatasmithExportOptions(ResizeTextureDimensions=true);</p>	

Export Formats	Description	Properties	
usdExportOptions	<p>Options for exporting a RoadRunner scene to USD.</p> <p><code>usdExportOptions(Name=Value)</code> creates an export options configuration object for the USD format, with properties specified as one or more name-value arguments.</p> <p>If a default property value is "auto", the RoadRunner application determines what value to use and sets the property to that value.</p>	SplitMeshes	<p>Split meshes by their segmentation type, specified as a logical 1 (true) or 0 (false).</p> <p>Default: "auto"</p>
		ResizeTextu reDimension s	<p>Resize the dimensions of exported textures by rounding them up to the next highest power of two, specified as a logical 1 (true) or 0 (false).</p> <p>Default: "auto"</p>
		TilesExport Options	<p>Options for splitting meshes per tile, specified as a <code>tilesExportOptions</code> object.</p> <p>Default: "auto"</p>
		<p>Example: <code>options = usdExportOptions(SplitMeshes=true);</code></p>	

Export Formats	Description	Properties	
vtdExportOptions	<p>Options for exporting a RoadRunner scene to VTD.</p> <p>vtdExportOptions(Name=Value) creates an export options configuration object for the VTD format, with properties specified as one or more name-value arguments.</p> <p>If a default property value is "auto", the RoadRunner application determines what value to use and sets the property to that value.</p>	OpenSceneGraphOptions	Options for export an OpenSceneGraph file, specified as an openSceneGraphExportOptions object. Default: "auto"
		OpenDriveOptions	Options for export an ASAM OpenDRIVE file, specified as an openDriveExportOptions object. Default: "auto"
		Example: options = vtdExportOptions(OpenDriveOptions=openDriveExportOptions(ExportObjects=true));	

Data Types: char | string

Version History

Introduced in R2022a

See Also

roadrunner | openScene | exportCustomFormat | importScene | close

Topics

"Export Multiple Scenes Using MATLAB"

"Convert Scenes Between Formats Using MATLAB Functions"

importScene

Import scene into RoadRunner using MATLAB

Syntax

```
importScene(rrApp, filename, formatname)  
importScene(rrApp, filename, formatname, importoptions)
```

Description

`importScene(rrApp, filename, formatname)` imports data from a file specified by `filename` into the currently open scene. The specified file must be in a format RoadRunner supports.

`importScene(rrApp, filename, formatname, importoptions)` sets options for import using the `importoptions` argument.

Examples

Import Scene from ASAM OpenDRIVE® File to RoadRunner

Open a project in RoadRunner using the `roadrunner` function by specifying the location in which to create a project. This example assumes that RoadRunner is installed in its default location in Windows.

Specify the path to an existing project. For example, this code shows the path to a project located on `C:\RR\MyProject`. The function returns a `roadrunner` object, `rrApp`, that provides functions for performing basic workflow tasks such as opening, closing, and saving scenes and projects.

```
projectFolder = "C:\RR\MyProject";  
rrApp = roadrunner(projectFolder);
```

Open a new scene in the current project. The `newScene` function opens a blank scene in the currently open project.

```
newScene(rrApp);
```

Set import options by creating an `openDriveImportOptions` object that disables the import of signals and objects from the file.

```
options = openDriveImportOptions(ImportSignals=false, ImportObjects=false);
```

Specify the path to an ASAM OpenDRIVE file. Then, use the `importScene` function to import data from the specified ASAM OpenDRIVE file into the blank open scene.

```
filename = "C:\RR\MyProject\Exports\FourWaySignal.xodr";  
formatname = "OpenDRIVE";  
importScene(rrApp, filename, formatname, options);
```


Input Arguments

rrApp — RoadRunner application

roadrunner object

RoadRunner application associated with a project, specified as a roadrunner object. This object provides functions for performing common workflow tasks such as opening, closing, and saving scenes and projects. rrApp provides functions that support importing data from files and exporting scenes to other formats from RoadRunner.

filename — Path of file to import into RoadRunner

character vector | string scalar

Path of file to import into RoadRunner, specified as a character vector or string scalar. This argument specifies the absolute or relative file path of the file to import. If you specify a relative path, then the path is relative to the Assets folder of the current project.

Example: `importScene(rrApp, "C:\RR\MyProject\Exports\FourWaySignal.xodr", "OpenDRIVE", options)`, `"C:\RR\MyProject\Exports\FourWaySignal.xodr"` represents the absolute file path of the file to be imported.

Data Types: char | string

formatname — Import format name

character vector | string scalar

Import format name, specified as a character vector or string scalar. This value must correspond to a valid import format supported by RoadRunner. Format name options are case-insensitive. RoadRunner supports only ASAM OpenDRIVE format.

Example: `importScene(rrApp, "FourWaySignal.xodr", "OpenDRIVE", options)`
"OpenDRIVE" specifies that the imported file `FourWaySignal.xodr` is in the ASAM OpenDRIVE format.

Data Types: char | string

importoptions — Import options configuration

valid export options object

Import options configuration, specified as one of the import options objects compatible with the file specified in the filename argument. Supported objects are:

- `openDRIVEImportOptions`
- `roadrunnerHDMMapImportOptions`

Import Formats	Description	Properties	
openDriveImportOptions	<p>Options for importing ASAM OpenDRIVE files into RoadRunner scene.</p> <p>openDriveImportOptions(Name=Value) creates an import options configuration object for the ASAM OpenDRIVE format, with properties specified as one or more name-value arguments.</p> <p>If a default property value is "auto", the RoadRunner application determines what value to use and sets the property to that value.</p>	ImportObjects	<p>Map all <object> entries in the imported file to props or markings, specified as a logical 1 (true) or 0 (false).</p> <p>Default: 0</p>
		ImportHeadingOffsetRelativeToOrientation	<p>Import the <hOffset> (heading offset) values of <signal> entries as being relative to <orientation>, which is the direction of travel of the road that the signal applies to. By default, the heading offset is relative to the heading of the road, regardless of the direction of travel, specified as a logical 1 (true) or 0 (false)</p> <p>Default: auto</p>
		LaneOptions	<p>Lane import options specified as a laneImportOptions object.</p>
		Offset	<p>Offset of the imported ASAM</p>

Import Formats	Description	Properties	
			OpenDRIVE scene, relative to the center of the RoadRunner scene, specified as a three-element vector of form [x y z]. Values are in meters.
		Projection	Projection of the imported ASAM OpenDRIVE scene, specified as a projectionImportOptions. If the file does not have projection information, then RoadRunner uses the projection of the scene. If both the scene and the file do not have projection information, then RoadRunner uses the Transverse Mercator projection centered at 0 degrees latitude and longitude.
		ImportSignals	Map all <signal> entries in the imported file to signals or signs, specified as a logical 1

Import Formats	Description	Properties	
			(true) or 0 (false). Default: "auto"
		ProjectionMode	Projection mode, specified as : "Unspecified", "FullProjection", "TranslateOnly", "NoProjection".
		Example: options = openDriveImportOptions(ImportSignals=true);	

Import Formats	Description	Properties	
laneImportOptions	<p>laneImportOptions (Name=Value) creates an options configuration object, with properties specified as one or more name-value arguments.</p> <p>If a default property value is "auto", the RoadRunner application determines what value to use and sets the property to that value.</p>	ConvertShoulderToCurb	<p>Specify this property as logical 1 (true) to import lanes with "curb" markings and type "shoulder" as type "curb" if this option is enabled. Otherwise, the function imports these lanes as type "shoulder", specified as a logical 1 (true) or 0 (false)</p> <p>Default: "auto"</p>
		ConvertLaneHeights	<p>Map all <height> entries to imported lanes, specified as a logical 1 (true) or 0 (false)</p> <p>Default: "auto"</p>
		MedianLaneType	<p>Map all <lane> types as "median" or "raised median", specified as: "Unspecified", "Median", "RaisedMedian".</p> <p>Default: "auto"</p>

Import Formats	Description	Properties	
		<p>Example:</p> <pre>options = laneImportOptions(Median LaneType="RaisedMedian") ; creates a lane import options object that maps all lanes as a raised median lane type.</pre>	
<p>projectionImportOptions</p>	<p>projectionImportOptions(Name=Value) creates an options configuration object for a map projection with properties specified as one or more name-value arguments.</p> <p>If a default property value is "auto", the RoadRunner application determines what value to use and sets the property to that value.</p>	<p>Projection</p>	<p>Geospatial projection and datum used to represent spatial coordinates in the map. This property supports valid WKT (including ESRI WKT) or PROJ.4 projection strings. Specified as a string scalar or character vector.</p> <p>Default: "auto"</p>
		<p>Example:</p> <pre>options = projectionImportOptions(Projection="utm +zone=11 +datum=WGS84 + units=m +no_defs +ellps=WGS84 +towgs84=0,0,"); creates projection a projection import options object that uses a PROJ.4 projection string.</pre>	

Import Formats	Description	Properties	
roadrunnerHDMapImportOptions	<p>Options for importing RoadRunner HD Map files into RoadRunner scene.</p> <p>roadrunnerHDMapImportOptions(Name=Value) creates an import options configuration object for the RoadRunner HD Map format with properties specified as one or more name-value arguments.</p> <p>If a default property value is "auto", the RoadRunner application determines what value to use and sets the property to that value.</p>	ImportStep	Import step, specified as one of the following strings: "Unspecified" or "Load". Default: "auto"
		LoadOptions	Load options, specified as roadrunnerHDMapLoadOptions. Default: "auto"
		BuildOptions	Load options, specified as roadrunnerHDMapBuildOptions. Default: "auto"
		Example: <pre>options = roadrunnerHDMapImportOptions(ImportStep="Load");</pre> creates a RoadRunner HD map import options that loads the map.	

Import Formats	Description	Properties	
<p>roadrunnerHDMapLoadOptions</p>	<p>Options for loading RoadRunner HD Map files into RoadRunner scene.</p> <p>roadrunnerHDMapLoadOptions (Name=Value) creates an options configuration object for loading the RoadRunner HD Map format with properties specified as one or more name-value arguments.</p> <p>If a default property value is "auto", the RoadRunner application determines what value to use and sets the property to that value.</p>	<p>Offset</p>	<p>Offset of the imported HD map, relative to the center of the RoadRunner scene, specified as a 3-element vector.</p>
		<p>Projection</p>	<p>Projection of the imported HD map scene, specified as a projectionImportOptions. If the projection is not set, then RoadRunner uses the file's projection. If the file does not have projection information, then RoadRunner uses the projection of the scene. If both the scene and the file do not have projection information, then RoadRunner uses the Transverse Mercator projection centered at 0 degrees latitude and longitude.</p>
		<p>Example:</p>	

Import Formats	Description	Properties
		<code>options = roadrunnerHDMapLoadOptions(Offset=[0 10 0]);</code> creates a RoadRunner HD map load options with the specified offset.

Import Formats	Description	Properties	
roadrunnerHDMaPBuildOptions	<p>Options for building RoadRunner HD Map files into RoadRunner scene.</p> <p>roadrunnerHDMaPBuildOptions (Name=Value) creates an options configuration object for building the RoadRunner HD Map format with properties specified as one or more name-value arguments.</p> <p>If a default property value is "auto", the RoadRunner application determines what value to use and sets the property to that value.</p>	FitCrossSections	Fit cross sections, specified as a logical 1 (true) or 0 (false) . Default: "auto"
		DetectAsphaltSurfaces	Detect asphalt surfaces, specified as a logical 1 (true) or 0 (false) . Default: "auto"
		ClearSceneOfExistingData	Clear the scene of existing data , specified as a logical 1 (true) or 0 (false) . Default: "auto"
		CurvatureBlend	Position of fit arcs used from transition from line to arc, specified as a scalar double value. Default: "auto"
		AutoDetectBridgesOptions	Options to auto detect bridges, specified as a autoDetectBridgesOptions. Default: "auto"
		Example:	

Import Formats	Description	Properties				
		options = roadrunnerHdMapBuildOptions(DetectAsphaltSurface=true); creates a RoadRunner HD map build options that detects asphalt surfaces.				
autoDetectBridgesOptions	<p>Options for auto detecting bridges.</p> <p>autoDetectBridgesOptions(Name=Value) creates an options configuration object for auto detecting bridges when building the RoadRunner HD Map. Properties specified as one or more name-value arguments.</p> <p>If a default property value is "auto", the RoadRunner application determines what value to use and sets the property to that value.</p>	<table border="1"> <tr> <td>BridgeSpanInflation</td> <td>Length of the bridge span specified as a scalar double value.</td> </tr> <tr> <td colspan="2">Default: "auto"</td> </tr> </table> <p>Example:</p> <p>options = autoDetectBridgesOptions(BridgeSpanInflation=35); creates options that set bridge span inflation.</p>	BridgeSpanInflation	Length of the bridge span specified as a scalar double value.	Default: "auto"	
BridgeSpanInflation	Length of the bridge span specified as a scalar double value.					
Default: "auto"						

Version History

Introduced in R2022a

See Also

roadrunner | openScene | exportScene | close

Topics

"Export Multiple Scenes Using MATLAB"

"Convert Scenes Between Formats Using MATLAB Functions"

newProject

Create new RoadRunner project using MATLAB

Syntax

```
newProject(rrApp,projectFolder)
newProject(rrApp,projectFolder,Name=Value)
```

Description

`newProject(rrApp,projectFolder)` creates a new RoadRunner project folder in the location specified by `projectFolder`. RoadRunner creates the specified folder, also attempting to create any folders in the specified file path that do not already exist. If a folder with the same name already exists in the specified location, then RoadRunner returns an error.

`newProject(rrApp,projectFolder,Name=Value)` sets options using one or more name-value arguments.

Examples

Create New RoadRunner Project

Open a project in RoadRunner using the `roadrunner` function by specifying the location in which to create a project. This example assumes that RoadRunner is installed in its default location in Windows.

Specify the path to an existing project. For example, this code shows the path to a project located on `C:\RR\MyProject`. The function returns a `roadrunner` object, `rrApp`, that provides functions for performing basic workflow tasks such as opening, closing, and saving scenes and projects.

```
projectFolder = "C:\RR\MyProject";
rrApp = roadrunner(projectFolder,InstallationFolder='C:\Program Files\RoadRunner R2023a\bin\win64');
```

Specify a location for a new project folder. Use the `newProject` function to create a new RoadRunner project in the specified location. The name of the project folder determines the name of the new project.

```
newProjectFolder = "C:\My New Project";
newProject(rrApp,newProjectFolder);
```

Input Arguments

rrApp — RoadRunner application

`roadrunner` object

RoadRunner application associated with a project, specified as a `roadrunner` object. This object provides functions for performing common workflow tasks such as opening, closing, and saving

scenes and projects. `rrApp` provides functions that support importing data from files and exporting scenes to other formats from RoadRunner.

projectFolder — RoadRunner project folder path

character vector | string scalar

RoadRunner project folder path, specified as a character vector or string scalar. Use this argument to specify the project folder in which to create a new scene when opening RoadRunner. For details about the RoadRunner project folder structure, see “RoadRunner Project and Scene System”.

Example: `roadrunner("C:\My New Project")` specifies to create a new scene in the `My New Project` folder when opening RoadRunner on a Windows machine.

Data Types: `char` | `string`

Name-Value Pair Arguments

Example: `newProject(rrApp,"C:\RR\nMyProject",AssetLibraries="RoadRunner_Asset_Library")`

Specify optional pairs of arguments as `Name1=Value1, ..., NameN=ValueN`, where `Name` is the argument name and `Value` is the corresponding value. Name-value arguments must appear after other arguments, but the order of the pairs does not matter.

AssetLibraries — RoadRunner asset libraries

character vector | string scalar

RoadRunner asset libraries to include in the new project, specified as a character vector or string scalar. The only valid entry is `"RoadRunner_Asset_Library"`, which includes the RoadRunner Asset Library assets in the project. If you do not specify `AssetLibraries`, then RoadRunner includes only the assets that come installed with RoadRunner projects by default.

Data Types: `char` | `string`

Version History

Introduced in R2022a

See Also

`roadrunner` | `openProject` | `saveProject`

Topics

“Export Multiple Scenes Using MATLAB”

“Convert Scenes Between Formats Using MATLAB Functions”

newScene

Create new RoadRunner scene using MATLAB

Syntax

```
newScene(rrApp)
```

Description

`newScene(rrApp)` creates a new scene in the current RoadRunner project.

Examples

Create New Scene

Open a project in RoadRunner using the `roadrunner` function by specifying the location in which to create a project. This example assumes that RoadRunner is installed in its default location in Windows.

Specify the path to an existing project. For example, this code shows the path to a project located on `C:\RR\MyProject`. The function returns a `roadrunner` object, `rrApp`, that provides functions for performing basic workflow tasks such as opening, closing, and saving scenes and projects.

```
projectFolder = "C:\RR\MyProject";  
rrApp = roadrunner(projectFolder);
```

Create an empty scene in the current project.

```
newScene(rrApp);
```

Input Arguments

rrApp — RoadRunner application

`roadrunner` object

RoadRunner application associated with a project, specified as a `roadrunner` object. This object provides functions for performing common workflow tasks such as opening, closing, and saving scenes and projects. `rrApp` provides functions that support importing data from files and exporting scenes to other formats from RoadRunner.

Version History

Introduced in R2022a

See Also

`roadrunner` | `openScene` | `saveScene`

Topics

“Export Multiple Scenes Using MATLAB”

“Convert Scenes Between Formats Using MATLAB Functions”

openProject

Open RoadRunner project using MATLAB

Syntax

```
openProject(rrApp,projectFolder)
```

Description

`openProject(rrApp,projectFolder)` opens RoadRunner to a new scene in the specified project `projectFolder`. The project must already exist. If the specified project is already open, RoadRunner still opens a new scene.

Examples

Open Project

Open a project in RoadRunner using the `roadrunner` function by specifying the location in which to create a project. This example assumes that RoadRunner is installed in its default location in Windows.

Specify the path to an existing project. For example, this code shows the path to a project located on `C:\RR\MyProject`. The function returns a `roadrunner` object, `rrApp`, that provides functions for performing basic workflow tasks such as opening, closing, and saving scenes and projects.

```
projectFolder = "C:\RR\MyProject";  
rrApp = roadrunner(projectFolder);
```

Specify a different project folder. Use the `openProject` function to create a new scene in the second RoadRunner project.

```
newProjectFolder = "C:\My New Project";  
openProject(rrApp,newProjectFolder);
```

Input Arguments

rrApp — RoadRunner application

`roadrunner` object

RoadRunner application associated with a project, specified as a `roadrunner` object. This object provides functions for performing common workflow tasks such as opening, closing, and saving scenes and projects. `rrApp` provides functions that support importing data from files and exporting scenes to other formats from RoadRunner.

projectFolder — RoadRunner project folder path

character vector | string scalar

RoadRunner project folder path, specified as a character vector or string scalar. Use this argument to specify the project folder in which to create a new scene when opening RoadRunner. For details about the RoadRunner project folder structure, see “RoadRunner Project and Scene System”.

Example: `roadrunner("C:\My New Project")` specifies to create a new scene in the My New Project folder when opening RoadRunner on a Windows machine.

Data Types: `char` | `string`

Version History

Introduced in R2022a

See Also

`roadrunner` | `newProject` | `saveProject`

Topics

“Export Multiple Scenes Using MATLAB”

“Convert Scenes Between Formats Using MATLAB Functions”

openScene

Open RoadRunner scene using MATLAB

Syntax

```
openScene(rrApp, filename)
```

Description

`openScene(rrApp, filename)` opens a specified scene file from the current RoadRunner project. If the specified scene does not belong to the current project, then RoadRunner determines the project to which the scene belongs and opens the scene from that project instead.

Examples

Open Scene in Current RoadRunner Project

Open a project in RoadRunner using the `roadrunner` function by specifying the location in which to create a project. This example assumes that RoadRunner is installed in its default location in Windows.

Specify the path to an existing project. For example, this code shows the path to a project located on `C:\RR\MyProject`. The function returns a `roadrunner` object, `rrApp`, that provides functions for performing basic workflow tasks such as opening, closing, and saving scenes and projects.

```
projectFolder = "C:\RR\MyProject";  
rrApp = roadrunner(projectFolder);
```

Open an existing scene in the current RoadRunner project.

```
filename = "FourWaySignal.rrscene";  
openScene(rrApp, filename);
```

Input Arguments

rrApp — RoadRunner application

`roadrunner` object

RoadRunner application associated with a project, specified as a `roadrunner` object. This object provides functions for performing common workflow tasks such as opening, closing, and saving scenes and projects. `rrApp` provides functions that support importing data from files and exporting scenes to other formats from RoadRunner.

filename — RoadRunner scene file to open

character vector | string scalar

RoadRunner scene file to open, specified as a character vector or string scalar. This argument specifies the absolute or relative path to the scene file you want to open. If you specify a relative path,

then the path is relative to the Scenes folder of the current project. `filename` must end with `.rrscene` or have no extension. If it has no extension, then RoadRunner appends the `.rrscene` extension to the file name before opening the scene.

Example: `openScene(rrApp, "FourWaySignal.rrscene")` open the `FourWaySignal.rrscene` scene file from the Scenes folder of the current project.

Data Types: `char` | `string`

Version History

Introduced in R2022a

See Also

`roadrunner` | `newScene` | `saveScene`

Topics

“Export Multiple Scenes Using MATLAB”

“Convert Scenes Between Formats Using MATLAB Functions”

saveProject

Save RoadRunner project using MATLAB

Syntax

```
saveProject(rrApp)
```

Description

`saveProject(rrApp)` saves the currently open scene in the specified RoadRunner instance to the current project.

Examples

Save Project

Open a project in RoadRunner using the `roadrunner` function by specifying the location in which to create a project. This example assumes that RoadRunner is installed in its default location in Windows.

Specify the path to an existing project. For example, this code shows the path to a project located on `C:\RR\MyProject`. The function returns a `roadrunner` object, `rrApp`, that provides functions for performing basic workflow tasks such as opening, closing, and saving scenes and projects..

```
projectFolder = "C:\RR\MyProject";  
rrApp = roadrunner(projectFolder);
```

Import an ASAM OpenDRIVE® file into RoadRunner by using the `importScene` function.

```
filename = "C:\RR\MyProject\Exports\FourWaySignal.xodr";  
importScene(rrApp, filename, "OpenDRIVE");
```

Save the modified assets in the project by using the `saveProject` function.

```
saveProject(rrApp);
```

Input Arguments

rrApp — RoadRunner application

`roadrunner` object

RoadRunner application associated with a project, specified as a `roadrunner` object. This object provides functions for performing common workflow tasks such as opening, closing, and saving scenes and projects. `rrApp` provides functions that support importing data from files and exporting scenes to other formats from RoadRunner.

Version History

Introduced in R2022a

See Also

roadrunner | newProject | openProject | importScene

Topics

“Export Multiple Scenes Using MATLAB”

“Convert Scenes Between Formats Using MATLAB Functions”

saveScene

Save RoadRunner scene using MATLAB

Syntax

```
saveScene(rrApp)
saveScene(rrApp, filename)
```

Description

`saveScene(rrApp)` saves any modifications to the scene currently open in the specified RoadRunner instance. If you have modified any assets in the scene, then this function also saves the current project.

`saveScene(rrApp, filename)` saves any modifications to the specified scene filename. If you have modified any assets in the scene, then this function also saves the project to which the scene belongs.

Examples

Save RoadRunner Scene

Open a project in RoadRunner using the `roadrunner` function by specifying the location in which to create a project. This example assumes that RoadRunner is installed in its default location in Windows.

Specify the path to an existing project. For example, this code shows the path to a project located on `C:\RR\MyProject`. The function returns a `roadrunner` object, `rrApp`, that provides functions for performing basic workflow tasks such as opening, closing, and saving scenes and projects.

```
projectFolder = "C:\RR\MyProject";
rrApp = roadrunner(projectFolder);
```

Open an existing scene in the current RoadRunner project.

```
filename = "FourWaySignal.rrscene";
openScene(rrApp, filename);
```

Save the RoadRunner scene to a new location.

```
newFilename = "FourWaySignal1.rrscene";
saveScene(rrApp, newFilename);
```

Input Arguments

rrApp — RoadRunner application

`roadrunner` object

RoadRunner application associated with a project, specified as a `roadrunner` object. This object provides functions for performing common workflow tasks such as opening, closing, and saving

scenes and projects. `rrApp` provides functions that support importing data from files and exporting scenes to other formats from RoadRunner.

filename — RoadRunner scene file to open

character vector | string scalar

RoadRunner scene file to open, specified as a character vector or string scalar. This argument specifies the absolute or relative path to the scene file you want to save. If you specify a relative path, then the path is relative to the Scenes folder of the current project. If you do not have a scene open, then RoadRunner returns an error. If any folders in the specified file path do not exist, then RoadRunner creates them recursively. `filename` must end with `.rrscene` or have no extension. If it has no extension, then RoadRunner appends the `.rrscene` extension to the file name before opening the scene. If the file being saved already exists, then RoadRunner overwrites it.

Example: `saveScene(rrApp, "FourWaySignal1.rrscene")` saves "FourWaySignal1.rrscene" located in the Scenes folder of the current project.

Data Types: `char` | `string`

Version History

Introduced in R2022a

See Also

`roadrunner` | `newScene` | `openScene`

Topics

"Export Multiple Scenes Using MATLAB"

"Convert Scenes Between Formats Using MATLAB Functions"

status

Get current status of RoadRunner using MATLAB

Syntax

```
rrStatus = status(rrApp)
```

Description

`rrStatus = status(rrApp)` gets the status of the RoadRunner application. The function returns the project, the scene or scenario that is currently loaded and any unsaved changes in the project, scene, or scenario.

Examples

Retrieve Status of RoadRunner Application

Get the status of RoadRunner application.

Open a project in RoadRunner using the `roadrunner` function by specifying the location in which to create a project. This example assumes that RoadRunner is installed in its default location in Windows.

Specify the path to an existing project. For example, this code shows the path to a project located on `C:\RR\MyProject`. The function returns a `roadrunner` object, `rrApp`, that provides functions for performing basic workflow tasks such as opening, closing, and saving scenes and projects.

```
projectFolder = "C:\RR\MyProject";  
rrApp = roadrunner(projectFolder, InstallationFolder='C:\Program Files\RoadRunner R2022b\bin\win
```

Open an existing scenario in RoadRunner Scenario by calling the `openScenario` function and passing it the `rrApp` object and the specific scenario filename that you want to open. This call opens the desired scenario in the RoadRunner Scenario application through MATLAB.

```
filename = "TrajectoryCutIn.rrscenario";  
openScenario(rrApp, filename);
```

Get the status of the RoadRunner application by calling the `status` function and passing it the `rrApp` object. This call returns the current project, scenario and any unsaved changes in the project.

```
rrStatus = status(rrApp)  
  
rrStatus = struct with fields:  
    Project: [1x1 struct]  
    Scene: [1x1 struct]  
    Scenario: [1x1 struct]
```

View project, scene and scenario details from the status of the RoadRunner application.

```
rrStatus.Project
```



```
ans = struct with fields:
  UnsavedChanges: 0
  Filename: 'C:/RR/MyProject'
```

`rrStatus.Scene`

```
ans = struct with fields:
  UnsavedChanges: 1
  Filename: 'C:/RR/MyProject/Scenes/ScenarioBasic.rrscene'
```

`rrStatus.Scenario`

```
ans = struct with fields:
  UnsavedChanges: 0
  Filename: 'C:/RR/MyProject/Scenarios/TrajectoryCutIn.rrscenario'
```

Input Arguments

`rrApp` — RoadRunner application

roadrunner object

RoadRunner application associated with a project, specified as a `roadrunner` object. This object provides functions for performing common workflow tasks such as opening, closing, and saving scenes and projects. `rrApp` provides functions that support importing data from files and exporting scenes to other formats from RoadRunner.

Output Arguments

`rrStatus` — Status of RoadRunner application

struct

Status of the RoadRunner application, specified as a struct. The function returns the project, the scene or scenario that is currently loaded and any unsaved changes in the project.

Tips

- RoadRunner makes changes automatically to the scene or scenario in the following cases:
 - When you switch to **Scenario Editing**, RoadRunner quantizes the road network to eliminate small roads and lanes.
 - When you load a scene or a scenario from a previous release, RoadRunner updates the internal data.

In these cases, you will see unsaved changes in the response output of the `status` function call, despite having made no changes to the scene or scenario.

Version History

Introduced in R2022b

See Also

roadrunner | newProject | newScene | newScenario

Topics

“RoadRunner Project and Scene System”

“RoadRunner Scenario Fundamentals” (RoadRunner Scenario)

“Export Multiple Scenes Using MATLAB”

“Convert Scenes Between Formats Using MATLAB Functions”

roadrunnerHDMap

Create RoadRunner HD Map using MATLAB

Description

A `roadrunnerHDMap` object enables you to create a RoadRunner HD Map using MATLAB. You can also populate the map with lanes, lane boundaries, lane markings, and junctions.

RoadRunner HD Map is a road data model for representing high-definition (HD) map data in a RoadRunner scene. This model defines a simple data structure to represent road layouts using lanes, lane boundaries, lane markings, and junctions. You can use the RoadRunner HD Map to create a binary file for your custom HD map data, import the binary file into RoadRunner, and build scenes.

Creation

Syntax

```
rrMap = roadrunnerHDMap()
rrMap = roadrunnerHDMap(Name=Value)
```

Description

`rrMap = roadrunnerHDMap()` creates an empty RoadRunner HD Map. You can populate the map with lanes, lane boundaries, lane markings, and junctions using `roadrunnerHDMap` object properties. The data in the map is serialized using protocol buffers and saved in a binary format with the `.rrhd` extension using the `write` method. You can use the `read` method to read a file with `.rrhd` extension and populate an empty `roadrunnerHDMap` object.

`rrMap = roadrunnerHDMap(Name=Value)` sets the properties using name-value pairs.

Properties

Author — Name of map author

character vector | string scalar

Name of the map author, specified as a string scalar.

Example: `rrMap = roadrunnerHDMap(Author="Map Author")` creates a RoadRunner HD Map owned by "Map Author".

Data Types: `char` | `string`

GeoReference — Geographic coordinates of road network origin

row vector

Geographic coordinates of road network origin, specified as a row vector of the form `[lat, lon]` with two elements. `lat` represents the latitude of the coordinates in degrees and `lon` represents the longitude of the coordinates in degrees. These values are with respect to the WGS84 reference

ellipsoid, which is a standard ellipsoid used by GPS data. For more details, see “Coordinate Space and Georeferencing”.

Data Types: `double`

GeographicBoundary — Spatial bounds of geometric data

`matrix`

Spatial bounds of the geometric data represented in the 3D coordinate space of the RoadRunner HD Map projection, specified as a 2-by-3 matrix of double precision. The elements in the matrix represent the minimum and the maximum position of a 3D axis-aligned box.

Example: `rrMap = roadrunnerHdMap(Author="Map Author", GeographicBoundary=[-0.782 -3.13 0; 101.565 50 0])` creates a RoadRunner HD Map owned by "Map Author". The spatial bounds of the geometric data in the 3D coordinate space of the map projection is specified by the double precision values of the parameter `GeographicBoundary`.

Data Types: `double`

Lanes — Lane properties of road

array of `roadrunner.hdmap.Lane` objects

Lane properties of a road, specified as an array of `roadrunner.hdmap.Lane` objects.

Example: `rrMap.Lanes(1) = roadrunner.hdmap.Lane(ID="Lane1", Geometry=[-0.782 -1.56; 50.78 23.43], LaneType="Driving", TravelDirection="Forward")` adds this information about the lanes for the map: lane id, coordinates defining lane geometry, lane type, and driving direction.

LaneBoundaries — Boundaries of lane

array of `roadrunner.hdmap.LaneBoundary` objects

Boundaries of the lane, specified as an array of `roadrunner.hdmap.LaneBoundary` objects.

Example: `rrMap.LaneBoundaries(1) = roadrunner.hdmap.LaneBoundary(ID="LaneBoundary1", Geometry=[0 0; 50 25])` adds the lane boundaries to the lane and defines the geometry for each lane boundary.

LaneGroups — Group of lanes with similar geometry

array of `roadrunner.hdmap.LaneGroup` objects

Group of lanes with similar geometry, specified as an array of `roadrunner.hdmap.LaneGroup` objects.

Example: `rrMap.LaneGroups = roadrunner.hdmap.LaneGroup(ID="LaneGroup1")` adds a lane group with an id "LaneGroup1" to the map.

LaneMarkings — Lane marking definitions

array of `roadrunner.hdmap.LaneMarking` objects

Lane marking definitions, specified as an array of `roadrunner.hdmap.LaneMarking` objects.

Example: `rrMap.LaneMarkings = roadrunner.hdmap.LaneMarking(ID="Dashed1", AssetPath="Assets/Markings/DashedSingleWhite.rrlms")` adds the "Dashed1" lane marking asset from its relative location to the lane.

Junctions — Road junction

array of `roadrunner.hdmap.Junction` objects

Road junctions in the map, specified as an array of `roadrunner.hdmap.Junction` objects.

Example: `rrMap.Junctions= roadrunner.hdmap.Junction(ID="Junction1")` adds a junction with id "Junction1" to the map.

BarrierTypes — Barrier types for barrier extrusion and type information

array of `roadrunner.hdmap.BarrierType` objects

Barrier types for barrier extrusion and type information, specified as an array of `roadrunner.hdmap.BarrierType` objects.

Example: `rrMap.BarrierTypes= roadrunner.hdmap.BarrierType(ID="BarrierType1", ExtrusionPath=path)` adds a barrier type with id "BarrierType1" for a barrier extrusion with the asset path, `path`, pointing to the extrusion information of this barrier type.

Barrier — Barriers in map

array of `roadrunner.hdmap.Barrier` objects

Barriers in the map, specified as an array of `roadrunner.hdmap.Barrier` objects.

Example: `rrMap.Barrier= roadrunner.hdmap.Barrier(ID="Barrier1")` adds a barrier with id "Barrier1" to the map.

SignTypes — Sign types of sign features and type information

array of `roadrunner.hdmap.SignType` objects

Sign types of sign features and type information, specified as an array of `roadrunner.hdmap.SignType` objects.

Example: `rrMap.SignTypes= roadrunner.hdmap.SignType(ID="SignType1", ExtrusionPath=AssetPath="Assets/Signs/UK/Sign_Parking.svg")` adds a sign type with id "SignType1" for a parking sign located on the relative path "Assets/Signs/UK/Sign_Parking.svg".

Signs — Signs in map

array of `roadrunner.hdmap.Sign` objects

Signs in the map, specified as an array of `roadrunner.hdmap.Sign` objects.

Example: `rrMap.Sign= roadrunner.hdmap.Sign(ID="Sign1")` adds a sign with id "Sign1" to the map.

StaticObjectType — Attributes of physical objects

array of `roadrunner.hdmap.StaticObjectType` objects

Attributes of physical objects (includes road furniture and props), specified as an array of `roadrunner.hdmap.StaticObjectType` objects.

Example: `rrMap.StaticObjectType= roadrunner.hdmap.StaticObjectType(ID="StaticObjectType1")` adds a static object type with id "StaticObjectType1" to the map.

StaticObject — Static objects in map

array of `roadrunner.hdmap.StaticObject` objects

Attributes of physical objects (includes road furniture and props), specified as an array of `roadrunner.hdmap.StaticObject` objects.

Example: `rrMap.StaticObject= roadrunner.hdmap.StaticObject(ID="StaticObject1")`
 adds a static object with id "StaticObject1" to the map.

Object Functions

`write` Write HD Map to binary file using MATLAB
`read` Read HD Map from binary file using MATLAB
`plot` Plot RoadRunner HD Map using MATLAB
`readCRS` Read coordinate reference system (CRS) data from RoadRunner HD map using MATLAB

Examples

Build a RoadRunner HD Map using Synthetic Data

Create a RoadRunner HD Map by calling the `roadrunnerHDMap` object. The object returns a map with an author and spatial bounds of geometric data attributes.

```
rrMap = roadrunnerHDMap(Author="Map Author",GeographicBoundary=[-0.782 -3.13 0;101.565 50 0]);
```

Create the road lanes using the `roadrunner.hdmap.Lane` object. Specify the lane information for the lane id, coordinates defining the lane geometry, driving direction, and lane type.

```
rrMap.Lanes(2) = roadrunner.hdmap.Lane(ID="Lane2",Geometry=[50.78 23.43;100.78 48.43],LaneType="D",Direction="F");
rrMap.Lanes(1) = roadrunner.hdmap.Lane(ID="Lane1",Geometry=[0.782 -1.56;50.78 23.43],LaneType="D",Direction="F");
```

Add connectivity information for Lane1 and Lane2. Specify alignment between the lanes by defining information about their predecessor and successor relationship.

```
addPredecessor(rrMap.Lanes(2),"Lane1",Alignment="Forward");
addSuccessor(rrMap.Lanes(1),"Lane2",Alignment="Forward");
```

Create the lane boundaries of the road using the `roadrunner.hdmap.LaneBoundary` object. Specify the lane boundary information for the lane id and the coordinates defining the lane geometry.

```
rrMap.LaneBoundaries(4) = roadrunner.hdmap.LaneBoundary(ID="LaneBoundary4",Geometry=[50 25; 100 50]);
rrMap.LaneBoundaries(3) = roadrunner.hdmap.LaneBoundary(ID="LaneBoundary3",Geometry=[51.565 21.875; 101.565 48.43]);
rrMap.LaneBoundaries(2) = roadrunner.hdmap.LaneBoundary(ID="LaneBoundary2",Geometry=[1.565 -3.13; 50.78 23.43]);
rrMap.LaneBoundaries(1) = roadrunner.hdmap.LaneBoundary(ID="LaneBoundary1",Geometry=[0 0; 50 25]);
```

Link the lane boundaries to the lanes. Define the left and the right lane boundaries for each lane, and specify alignment between lanes and lane boundaries.

```
leftBoundary(rrMap.Lanes(1),"LaneBoundary1",Alignment="Forward");
rightBoundary(rrMap.Lanes(1),"LaneBoundary2",Alignment="Forward");
leftBoundary(rrMap.Lanes(2),"LaneBoundary4",Alignment="Forward");
rightBoundary(rrMap.Lanes(2),"LaneBoundary3",Alignment="Forward");
```

Version History

Introduced in R2022b

See Also

`roadrunner`

Topics

“Build Simple Roads Programmatically Using RoadRunner HD Map”

“Build Scenes from Custom Data Using RoadRunner HD Map”

write

Write HD Map to binary file using MATLAB

Syntax

```
write(rrMap,filename)
write(rrMap,filename,Name=Value)
```

Description

`write(rrMap,filename)` writes the RoadRunner HD Map in a binary file `filename`. Specify the extension of the file as `.rrhd`. You can import the file into RoadRunner by adding the file to a folder in the **Library Browser** and dragging it into the scene. You can then build the scene using the **Scene Builder Tool**.

`write(rrMap,filename,Name=Value)` sets options using one or more name-value arguments.

Examples

Write Map to a Binary File

Create a RoadRunner HD Map by calling the `roadrunnerHDMMap` object. Populate the map using your custom data. For example, this code specifies the author and the spatial bounds of the geometric data attributes for the map. The call to the `roadrunnerHDMMap` object returns a HD map with author and spatial bounds attributes.

```
rrMap = roadrunnerHDMMap(Author="Map Author",GeographicBoundary=[-0.782 -3.13 0;101.565 50 0])
```

```
rrMap =
```

```
roadrunnerHDMMap with properties:
```

```
    Author: "Map Author"
  Projection: ""
GeographicBoundary: [2x3 double]
        Lanes: [0x1 roadrunner.hdmap.Lane]
LaneBoundaries: [0x1 roadrunner.hdmap.LaneBoundary]
   LaneGroups: [0x1 roadrunner.hdmap.LaneGroup]
LaneMarkings: [0x1 roadrunner.hdmap.LaneMarking]
      Junctions: [0x1 roadrunner.hdmap.Junction]
BarrierTypes: [0x1 roadrunner.hdmap.BarrierType]
      Barriers: [0x1 roadrunner.hdmap.Barrier]
   SignTypes: [0x1 roadrunner.hdmap.SignType]
        Signs: [0x1 roadrunner.hdmap.Sign]
```

Specify the name of the binary file you want to write the map to. Use the `write` function to write the map data to the file specified by the argument `filename`.

```
filename = "MyMap.rrhd";
write(rrMap,filename)
```


Input Arguments

rrMap — RoadRunner HD Map road data model

roadrunnerHDMMap object

RoadRunner HD Map road data model, specified as a `roadrunnerHDMMap` object. `rrMap` defines a simple data structure to represent road layouts using lanes, lane boundaries, lane markings, and junctions. This object provides functions that support reading, writing, and plotting HD map data.

filename — File name to write map data to

character vector | string scalar

File name to write the map data to, specified as a character vector or string scalar. This argument specifies the absolute or relative path to the binary file you want to write the data to. If you specify a relative path, then the path is relative to the `Scenes` folder of the current project. `filename` must end with `.rrhd`.

Example: `write(rrMap, "MyMap.rrhd")` writes the map data to the `MyMap.rrhd` binary file.

Data Types: `char` | `string`

Name-Value Pair Arguments

Specify optional pairs of arguments as `Name1=Value1, ..., NameN=ValueN`, where `Name` is the argument name and `Value` is the corresponding value. Name-value arguments must appear after other arguments, but the order of the pairs does not matter.

Example:

WKT — WKT projection string

character vector | string scalar

WKT projection string, specified as a character vector or string scalar.

PROJ4 — PROJ.4 projection string

character vector | string scalar

A PROJ.4 projection string, specified as a character vector or string scalar.

Version History

Introduced in R2022b

See Also

Objects

`roadrunnerHDMMap` | `roadrunner`

Functions

`read` | `plot`

Topics

“Build Simple Roads Programmatically Using RoadRunner HD Map”

read

Read HD Map from binary file using MATLAB

Syntax

```
read(rrMap, filename)
```

Description

`read(rrMap, filename)` reads the RoadRunner HD Map from a binary file specified by `filename` and populates the `rrMap` object with the map data. Specify the extension of the file as `.rrhd`.

Examples

Read HD Map from Binary File

Create an empty RoadRunner HD Map by calling the `roadrunnerHDMMap` object.

```
rrMap = roadrunnerHDMMap;
```

Specify the name of the binary file you want to read. Use the `read` function to read the file specified by the argument `filename`.

```
filename = "C:\RR\MyProject\Scenes\MyMap.rrhd";  
read(rrMap, filename);
```

Input Arguments

rrMap — RoadRunner HD Map road data model

`roadrunnerHDMMap` object

RoadRunner HD Map road data model, specified as a `roadrunnerHDMMap` object. `rrMap` defines a simple data structure to represent road layouts using lanes, lane boundaries, lane markings, and junctions. This object provides functions that support reading, writing, and plotting HD map data.

filename — File name to read map data from

character vector | string scalar

File name to read the map data from, specified as a character vector or string scalar. This argument specifies the absolute or relative path to the binary file you want to read. If you specify a relative path, then the path is relative to the `Scenes` folder of the current project. `filename` must end with `.rrhd`.

Example: `read(rrMap, "MyMap.rrhd")` reads the map data from the `MyMap.rrhd` binary file.

Data Types: `char` | `string`

Version History

Introduced in R2022b

See Also

Objects

roadrunnerHDMap | roadrunner

Functions

write | plot

Topics

“Build Simple Roads Programmatically Using RoadRunner HD Map”

plot

Plot RoadRunner HD Map using MATLAB

Syntax

```
plot(rrMap)
plot(rrMap,Name=Value)
```

Description

`plot(rrMap)` plots the RoadRunner HD Map's lane centers, lane boundaries, lane group centers, and barriers.

`plot(rrMap,Name=Value)` specifies additional name-value pair arguments for the HD map.

Examples

Plot HD Map from Binary File

Create an empty RoadRunner HD Map by calling the `roadrunnerHDMMap` object.

```
rrMap = roadrunnerHDMMap;
```

Specify the name of the binary file you want to read. Use the `read` function to read the file specified by the argument `filename`.

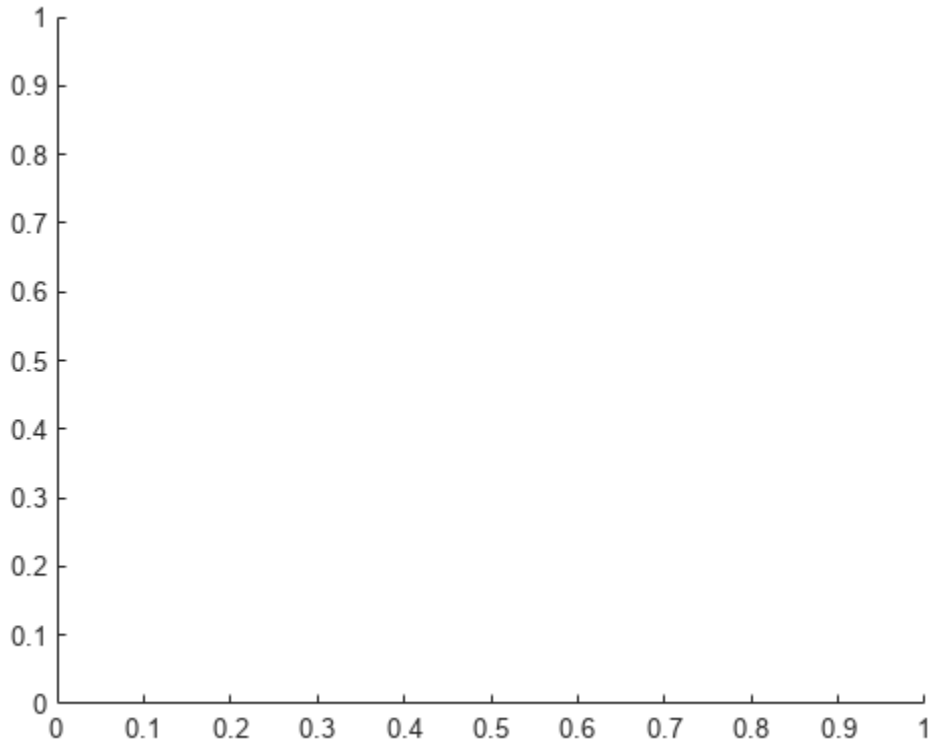
```
filename = "C:\RR\MyProject\Scenes\MyMap.rrhd";
read(rrMap,filename)
rrMap
```

```
rrMap =
  roadrunnerHDMMap with properties:
```

```
    Author: "Map Author"
  Projection: ""
GeographicBoundary: [2×3 double]
      Lanes: [0×1 roadrunner.hdmap.Lane]
LaneBoundaries: [0×1 roadrunner.hdmap.LaneBoundary]
  LaneGroups: [0×1 roadrunner.hdmap.LaneGroup]
LaneMarkings: [0×1 roadrunner.hdmap.LaneMarking]
   Junctions: [0×1 roadrunner.hdmap.Junction]
BarrierTypes: [0×1 roadrunner.hdmap.BarrierType]
   Barriers: [0×1 roadrunner.hdmap.Barrier]
  SignTypes: [0×1 roadrunner.hdmap.SignType]
      Signs: [0×1 roadrunner.hdmap.Sign]
```

Plot the map.

```
plot(rrMap)
```



Input Arguments

rrMap — RoadRunner HD Map road data model

roadrunnerHDMap object

RoadRunner HD Map road data model, specified as a `roadrunnerHDMap` object. `rrMap` defines a simple data structure to represent road layouts using lanes, lane boundaries, lane markings, and junctions. This object provides functions that support reading, writing, and plotting HD map data.

Name-Value Pair Arguments

Specify optional pairs of arguments as `Name1=Value1, ..., NameN=ValueN`, where `Name` is the argument name and `Value` is the corresponding value. Name-value arguments must appear after other arguments, but the order of the pairs does not matter.

Example:

Parent — Handle to axes object

handle object

Handle to an axes object, specified as a handle object to contain the plot. If this property is not specified, the `plot` function generates a new figure.

ShowLaneGroups — Display lane group centers

false or 0 (default) | true or 1

Display lane group centers in the plot, specified as logical 0 (false) or logical 1 (true). If you set `ShowLaneGroups` to true, then the `plot` function displays the lane group centers in the plot.

ShowBarriers — Display barrier centers

false or 0 (default) | true or 1

Display barrier centers in the plot, specified as logical 0 (false) or logical 1 (true). If you set `ShowBarriers` to true, then the `plot` function displays the barrier centers in the plot.

ShowSigns — Display signs

false or 0 (default) | true or 1

Display signs in the plot, specified as logical 0 (false) or logical 1 (true). If you set `ShowSigns` to true, then the `plot` function displays the signs in the plot.

ShowStaticObjects — Display static objects

false or 0 (default) | true or 1

Display static objects in the plot, specified as logical 0 (false) or logical 1 (true). If you set `ShowStaticObjects` to true, then the `plot` function displays the static objects in the plot.

ShowLegend — Display legend

true or 1 (default) | false or 0

Display legend in the plot, specified as logical 1 (true) or logical 0 (false). If you set `ShowLegend` to false, then the `plot` function does not display the legend for the plot. A legend is not shown when the Parent name-value argument is specified.

ShowLineMarkers — Display line markers

true or 1 (default) | false or 0

Display line markers in the plot, specified as logical 1 (true) or logical 0 (false). If you set `ShowLineMarkers` to true, then the `plot` function displays the line markers for the plot.

Version History

Introduced in R2022b

See Also

Objects

`roadrunnerHDMMap` | `roadrunner`

Functions

`write` | `read`

Topics

“Build Simple Roads Programmatically Using RoadRunner HD Map”

roadrunner.hdmap.Lane

Create lanes in RoadRunner HD Map using MATLAB

Description

A `roadrunner.hdmap.Lane` object enables you to define the structure for representing lanes in a RoadRunner HD Map scene model. You can create an empty lane in a RoadRunner HD Map and assign properties to the lane to describe its geometry, travel direction, lane boundaries, and lane type.

Creation

Syntax

```
aLane = roadrunner.hdmap.Lane()  
aLane = roadrunner.hdmap.Lane(Name=Value)
```

Description

`aLane = roadrunner.hdmap.Lane()` creates an empty lane.

`aLane = roadrunner.hdmap.Lane(Name=Value)` sets the properties of the lane using name-value pairs.

Properties

ID — ID of lane

character vector | string scalar

ID of the lane, specified as a character vector or string scalar.

Example: `rrMap.Lanes = roadrunner.hdmap.Lane(ID="Lane1")` creates a lane with id `Lane1` in a RoadRunner HD Map.

Data Types: `char` | `string`

Geometry — 3D coordinates of points representing center line of lane

three element vector

3D coordinates of points representing the center line of the lane, specified as a three element vector, with double precision values.

Example: `rrMap.Lanes(1) = roadrunner.hdmap.Lane(ID="Lane1", Geometry=[-0.782 -1.56; 50.78 23.43])` creates a lane with id `"Lane1"` and defines the coordinates for the lane geometry.

Data Types: `double`

TravelDirection — Direction of travel along the lane

character vector | string scalar

Direction of travel along the lane, specified as a character vector or string scalar. The direction of travel along the lane is specified as one of the following strings: "Unspecified", "Undirected", "Forward", "Backward", or "Bidirectional".

Example: `rrMap.Lanes(1) = roadrunner.hdmap.Lane(ID="Lane1", Geometry=[-0.782 -1.56;50.78 23.43], TravelDirection="Forward")` creates a lane with a lane id, coordinates defining lane geometry, and driving direction.

Data Types: `char` | `string`

LeftLaneBoundary — Linkage information for left boundary of lane

`roadrunner.hdmap.AlignedReference` object

Linkage information for the left boundary of the lane, specified as a `roadrunner.hdmap.AlignedReference` object.

RightLaneBoundary — Linkage information for right boundary of lane

`roadrunner.hdmap.AlignedReference` object

Linkage information for the right boundary of the lane, specified as a `roadrunner.hdmap.AlignedReference` object.

Predecessors — Linkage information for preceding lanes

`roadrunner.hdmap.AlignedReference` object

Linkage information for preceding lanes, specified as a `roadrunner.hdmap.AlignedReference` object.

Successors — Linkage information for succeeding lanes

`roadrunner.hdmap.AlignedReference` object

Linkage information for succeeding lanes, specified as a `roadrunner.hdmap.AlignedReference` object.

LaneType — Type of lane

character vector | string scalar

Type of lane, specified as a character vector or string scalar. The type of lane is specified as one of the following strings: "Unspecified", "Driving", "Shoulder", "Border", "Restricted", "Parking", "Curb", "Sidewalk", "CenterTurn" or "Biking".

Example: `rrMap.Lanes(1) = roadrunner.hdmap.Lane(ID="Lane1", Geometry=[-0.782 -1.56;50.78 23.43], LaneType="Driving", TravelDirection="Forward")` creates a lane with a lane id, coordinates defining lane geometry, lane type, and driving direction.

Data Types: `char` | `string`

Object Functions

<code>addPredecessor</code>	Add predecessor to lane in RoadRunner HD Map using MATLAB
<code>addSuccessor</code>	Add successor to lane in RoadRunner HD Map using MATLAB
<code>leftBoundary</code>	Set left boundary of lane in RoadRunner HD Map using MATLAB
<code>rightBoundary</code>	Set right boundary of lane in RoadRunner HD Map using MATLAB

Examples

Add Lane to RoadRunner HD Map

Create an empty RoadRunner HD Map by calling the `roadrunnerHDMMap` object.

```
rrMap = roadrunnerHDMMap()

rrMap =
  roadrunnerHDMMap with properties:

    Author: ""
    Projection: ""
    GeographicBoundary: [0×3 double]
    Lanes: [0×1 roadrunner.hdmap.Lane]
    LaneBoundaries: [0×1 roadrunner.hdmap.LaneBoundary]
    LaneGroups: [0×1 roadrunner.hdmap.LaneGroup]
    LaneMarkings: [0×1 roadrunner.hdmap.LaneMarking]
    Junctions: [0×1 roadrunner.hdmap.Junction]
    BarrierTypes: [0×1 roadrunner.hdmap.BarrierType]
    Barriers: [0×1 roadrunner.hdmap.Barrier]
    SignTypes: [0×1 roadrunner.hdmap.SignType]
    Signs: [0×1 roadrunner.hdmap.Sign]
```

Create the road lane using the `roadrunner.hdmap.Lane` object. Specify the lane information for the lane id, coordinates defining the lane geometry, driving direction, and lane type.

```
rrmap.Lane = roadrunner.hdmap.Lane(ID="Lane",Geometry=[-0.782 -1.56;50.78 23.43],TravelDirection=)

rrmap = struct with fields:
  Lane: [1×1 roadrunner.hdmap.Lane]
```

Version History

Introduced in R2022b

See Also

`roadrunnerHDMMap` | `roadrunner.hdmap.LaneBoundary` | `roadrunner.hdmap.LaneGroup` | `roadrunner.hdmap.LaneMarking` | `roadrunner.hdmap.Junction`

Topics

“Build Simple Roads Programmatically Using RoadRunner HD Map”

“Build Scenes from Custom Data Using RoadRunner HD Map”

roadrunner.hdmap.Reference

Defines id of referenced object in RoadRunner HD map using MATLAB

Description

The `roadrunner.hdmap.Reference` object holds the id to the object being referred to in RoadRunner HD Maps. For example, a Junction A can refer to Lane B by holding Lane B's id in its reference.

Creation

Syntax

```
ref = roadrunner.hdmap.Reference()  
ref = roadrunner.hdmap.Reference(Name=Value)
```

Description

`ref = roadrunner.hdmap.Reference()` creates an empty reference.

`ref = roadrunner.hdmap.Reference(Name=Value)` sets the properties of the reference using name-value pairs.

Properties

ID — ID of referenced object

character vector | string scalar

ID of the referenced object, specified as a character vector or string scalar object.

Example: `ref = roadrunner.hdmap.Reference(ID="Lane2")` creates a referenced object with id Lane2.

Data Types: `char` | `string`

Examples

Add Lane with Reference to RoadRunner HD Map

Create an empty RoadRunner HD Map by calling the `roadrunnerHDMMap` object.

```
rrMap = roadrunnerHDMMap()
```

```
rrMap =  
    roadrunnerHDMMap with properties:
```

```
    Author: ""
```

```

    Projection: ""
    GeographicBoundary: [0×3 double]
        Lanes: [0×1 roadrunner.hdmap.Lane]
    LaneBoundaries: [0×1 roadrunner.hdmap.LaneBoundary]
    LaneGroups: [0×1 roadrunner.hdmap.LaneGroup]
    LaneMarkings: [0×1 roadrunner.hdmap.LaneMarking]
    Junctions: [0×1 roadrunner.hdmap.Junction]
    BarrierTypes: [0×1 roadrunner.hdmap.BarrierType]
    Barriers: [0×1 roadrunner.hdmap.Barrier]
    SignTypes: [0×1 roadrunner.hdmap.SignType]
    Signs: [0×1 roadrunner.hdmap.Sign]

```

Create the road lane using the `roadrunner.hdmap.Lane` object. Specify the lane information for the lane id, coordinates defining the lane geometry, driving direction, and lane type.

```
rLane = roadrunner.hdmap.Lane(ID="Lane1", Geometry=[-0.782 -1.56;50.78 23.43], TravelDirection="Fo
```

```
rLane =
```

```
    Lane with properties:
```

```

        ID: "Lane1"
        Geometry: [2×2 double]
        TravelDirection: Forward
        LeftLaneBoundary: [0×0 roadrunner.hdmap.AlignedReference]
        RightLaneBoundary: [0×0 roadrunner.hdmap.AlignedReference]
        Predecessors: [0×1 roadrunner.hdmap.AlignedReference]
        Successors: [0×1 roadrunner.hdmap.AlignedReference]
        LaneType: Driving

```

Create a reference to `Lane1` using the `roadrunner.hdmap.Reference` object.

```
ref = roadrunner.hdmap.Reference(ID="Lane1")
```

```
ref =
```

```
    Reference with properties:
```

```

        ID: "Lane1"

```

Version History

Introduced in R2022b

See Also

`roadrunnerHDMAP` | `roadrunner.hdmap.Lane` | `roadrunner.hdmap.AlignedReference`

Topics

“Build Simple Roads Programmatically Using RoadRunner HD Map”

“Build Scenes from Custom Data Using RoadRunner HD Map”

roadrunner.hdmap.AlignedReference

Provided information for linking objects in RoadRunner HD Maps using MATLAB

Description

The `roadrunner.hdmap.AlignedReference` object enables you to specify information for linking objects in RoadRunner HD Maps. This information is used to specify lane navigation. You can link a lane with lane boundaries and other lane. To specify the alignment between linked objects, you must follow these rules:

- Alignment must always be relative to the orientation of the object of interest.
- When the directions of two objects match, the alignment is forward. Otherwise, the alignment is backward.

Creation

Syntax

```
alref = roadrunner.hdmap.AlignedReference()  
alref = roadrunner.hdmap.AlignedReference(Name=Value)
```

Description

`alref = roadrunner.hdmap.AlignedReference()` creates an empty aligned reference.

`alref = roadrunner.hdmap.AlignedReference(Name=Value)` sets the properties of the aligned reference using name-value pairs.

Properties

Reference — ID of referenced object

`roadrunner.hdmap.Reference` object

ID of the referenced object, specified as a `roadrunner.hdmap.Reference` object.

Example: `ref= roadrunner.hdmap.Reference(ID="Lane2")` creates a referenced object with id Lane2.

Alignment — Type of alignment

character vector | string scalar

Type of alignment between linked objects, specified as a character vector or string scalar. The type of alignment is specified as one of the following strings: "Unspecified", "Forward", or "Backward".

Example: `alref= roadrunner.hdmap.AlignedReference(Reference=ref, Alignment="Forward")` creates the aligned reference for Lane2 with "Forward" alignment.

Data Types: `char` | `string`

Examples

Add Lane with Aligned Reference to RoadRunner HD Map

Create an empty RoadRunner HD Map by calling the `roadrunnerHDMAP` object.

```
rrMap = roadrunnerHDMAP()
rrMap =
  roadrunnerHDMAP with properties:
      Author: ""
      Projection: ""
      GeographicBoundary: [0x3 double]
      Lanes: [0x1 roadrunner.hdmap.Lane]
      LaneBoundaries: [0x1 roadrunner.hdmap.LaneBoundary]
      LaneGroups: [0x1 roadrunner.hdmap.LaneGroup]
      LaneMarkings: [0x1 roadrunner.hdmap.LaneMarking]
      Junctions: [0x1 roadrunner.hdmap.Junction]
      BarrierTypes: [0x1 roadrunner.hdmap.BarrierType]
      Barriers: [0x1 roadrunner.hdmap.Barrier]
      SignTypes: [0x1 roadrunner.hdmap.SignType]
      Signs: [0x1 roadrunner.hdmap.Sign]
```

Create the road lane using the `roadrunner.hdmap.Lane` object. Specify the lane information for the lane id, coordinates defining the lane geometry, driving direction, and lane type.

```
rLane = roadrunner.hdmap.Lane(ID="Lane1",Geometry=[-0.782 -1.56;50.78 23.43],TravelDirection="Forward")
rLane =
  Lane with properties:
      ID: "Lane1"
      Geometry: [2x2 double]
      TravelDirection: Forward
      LeftLaneBoundary: [0x0 roadrunner.hdmap.AlignedReference]
      RightLaneBoundary: [0x0 roadrunner.hdmap.AlignedReference]
      Predecessors: [0x1 roadrunner.hdmap.AlignedReference]
      Successors: [0x1 roadrunner.hdmap.AlignedReference]
      LaneType: Driving
```

Create an aligned reference to `Lane1` using the `roadrunner.hdmap.AlignedReference` object. For this example, specify the type of alignment as `Forward`.

```
ref = roadrunner.hdmap.Reference(ID="Lane1")
ref =
  Reference with properties:
      ID: "Lane1"

alref = roadrunner.hdmap.AlignedReference(Reference=ref,Alignment="Forward")
alref =
  AlignedReference with properties:
```

Reference: [1×1 roadrunner.hdmap.Reference]
Alignment: Forward

Version History

Introduced in R2022b

See Also

roadrunnerHDMMap | roadrunner.hdmap.Lane | roadrunner.hdmap.Reference

Topics

“Build Simple Roads Programmatically Using RoadRunner HD Map”

“Build Scenes from Custom Data Using RoadRunner HD Map”

addPredecessor

Add predecessor to lane in RoadRunner HD Map using MATLAB

Syntax

```
aPredecessor = addPredecessor(aLane,predecessorID)
aPredecessor = addPredecessor(aLane,predecessorID,Name=Value)
```

Description

`aPredecessor = addPredecessor(aLane,predecessorID)` adds a predecessor with ID `predecessorID` to the `Predecessors` property of the lane and returns it.

`aPredecessor = addPredecessor(aLane,predecessorID,Name=Value)` sets options using one or more name-value arguments.

Examples

Add Predecessor to a Lane in RoadRunner HD Map

Create an empty RoadRunner HD Map by calling the `roadrunnerHDMMap` object.

```
rrMap = roadrunnerHDMMap()
```

```
rrMap =
  roadrunnerHDMMap with properties:
```

```

    Author: ""
  Projection: ""
GeographicBoundary: [0x3 double]
      Lanes: [0x1 roadrunner.hdmap.Lane]
LaneBoundaries: [0x1 roadrunner.hdmap.LaneBoundary]
  LaneGroups: [0x1 roadrunner.hdmap.LaneGroup]
LaneMarkings: [0x1 roadrunner.hdmap.LaneMarking]
  Junctions: [0x1 roadrunner.hdmap.Junction]
BarrierTypes: [0x1 roadrunner.hdmap.BarrierType]
   Barriers: [0x1 roadrunner.hdmap.Barrier]
  SignTypes: [0x1 roadrunner.hdmap.SignType]
     Signs: [0x1 roadrunner.hdmap.Sign]
```

Create two road lanes using the `roadrunner.hdmap.Lane` object. Specify the lane information for the lane id, coordinates defining the lane geometry, driving direction, and lane type.

```
rrMap.Lanes(2) = roadrunner.hdmap.Lane(ID="Lane2",Geometry=[-0.782 -1.56;50.78 23.43],TravelDire
```

```
rrMap =
  roadrunnerHDMMap with properties:
```

```

    Author: ""
  Projection: ""
```

```

GeographicBoundary: [0×3 double]
    Lanes: [2×1 roadrunner.hdmap.Lane]
LaneBoundaries: [0×1 roadrunner.hdmap.LaneBoundary]
LaneGroups: [0×1 roadrunner.hdmap.LaneGroup]
LaneMarkings: [0×1 roadrunner.hdmap.LaneMarking]
Junctions: [0×1 roadrunner.hdmap.Junction]
BarrierTypes: [0×1 roadrunner.hdmap.BarrierType]
Barriers: [0×1 roadrunner.hdmap.Barrier]
SignTypes: [0×1 roadrunner.hdmap.SignType]
Signs: [0×1 roadrunner.hdmap.Sign]

```

```
rrMap.Lanes(1) = roadrunner.hdmap.Lane(ID="Lane1",Geometry=[-0.534 -2.43;25.78 46.43],TravelDire
```

```
rrMap =
```

```
roadrunnerHDMAP with properties:
```

```

    Author: ""
    Projection: ""
GeographicBoundary: [0×3 double]
    Lanes: [2×1 roadrunner.hdmap.Lane]
LaneBoundaries: [0×1 roadrunner.hdmap.LaneBoundary]
LaneGroups: [0×1 roadrunner.hdmap.LaneGroup]
LaneMarkings: [0×1 roadrunner.hdmap.LaneMarking]
Junctions: [0×1 roadrunner.hdmap.Junction]
BarrierTypes: [0×1 roadrunner.hdmap.BarrierType]
Barriers: [0×1 roadrunner.hdmap.Barrier]
SignTypes: [0×1 roadrunner.hdmap.SignType]
Signs: [0×1 roadrunner.hdmap.Sign]

```

Add Lane1 as a predecessor to Lane2.

```
addPredecessor(rrMap.Lanes(2), "Lane1")
```

```
ans =
```

```
AlignedReference with properties:
```

```

Reference: [1×1 roadrunner.hdmap.Reference]
Alignment: Unspecified

```

Input Arguments

aLane — Lane properties of road

roadrunner.hdmap.Lane object

Lane properties of a road, specified as a roadrunner.hdmap.Lane object.

Example: aLane= roadrunner.hdmap.Lane(ID="Lane1", Geometry=[-0.782 -1.56;50.78 23.43], LaneType="Driving", TravelDirection="Forward") creates a lane with properties lane id, coordinates defining lane geometry, lane type, and driving direction.

predecessorID — ID of predecessor lane

character vector | string scalar

ID of the predecessor lane, specified as a string.

Example: `addPredecessor(rrMap.Lanes(2),"Lane1")` assigns Lane1 as the predecessor of Lane2 in a RoadRunner HD Map.

Data Types: `char` | `string`

Name-Value Pair Arguments

Specify optional pairs of arguments as `Name1=Value1, . . . ,NameN=ValueN`, where `Name` is the argument name and `Value` is the corresponding value. Name-value arguments must appear after other arguments, but the order of the pairs does not matter.

Example:

Alignment — Type of alignment

`character vector` | `string scalar`

Type of alignment between linked objects, specified as a character vector or string scalar. The type of alignment is specified as one of the following strings: "Unspecified", "Forward", or "Backward".

Example: `alref= roadrunner.hdmap.AlignedReference(Reference=ref, Alignment="Forward")` creates the aligned reference for Lane2 with "Forward" alignment.

Data Types: `char` | `string`

Output Arguments

aPredecessor — Linkage information for preceding lanes

`roadrunner.hdmap.AlignedReference` object

Linkage information for preceding lanes, specified as a `roadrunner.hdmap.AlignedReference` object.

Version History

Introduced in R2022b

See Also

`roadrunnerHDMAP` | `roadrunner.hdmap.Lane` | `roadrunner.hdmap.AlignedReference` | `addSuccessor`

Topics

"Build Simple Roads Programmatically Using RoadRunner HD Map"
 "Build Scenes from Custom Data Using RoadRunner HD Map"

addSuccessor

Add successor to lane in RoadRunner HD Map using MATLAB

Syntax

```
aSuccessor = addSuccessor(aLane, successorID)
aSuccessor = addSuccessor(aLane, successorID, Name=Value)
```

Description

`aSuccessor = addSuccessor(aLane, successorID)` adds a successor with ID `successorID` to the `Successors` property of the lane and returns it.

`aSuccessor = addSuccessor(aLane, successorID, Name=Value)` sets options using one or more name-value arguments.

Examples

Add Successor to a Lane in RoadRunner HD Map

Create an empty RoadRunner HD Map by calling the `roadrunnerHDMAP` object.

```
rrMap = roadrunnerHDMAP()
```

```
rrMap =
  roadrunnerHDMAP with properties:
```

```

    Author: ""
  Projection: ""
GeographicBoundary: [0×3 double]
        Lanes: [0×1 roadrunner.hdmap.Lane]
LaneBoundaries: [0×1 roadrunner.hdmap.LaneBoundary]
   LaneGroups: [0×1 roadrunner.hdmap.LaneGroup]
LaneMarkings: [0×1 roadrunner.hdmap.LaneMarking]
   Junctions: [0×1 roadrunner.hdmap.Junction]
BarrierTypes: [0×1 roadrunner.hdmap.BarrierType]
    Barriers: [0×1 roadrunner.hdmap.Barrier]
   SignTypes: [0×1 roadrunner.hdmap.SignType]
        Signs: [0×1 roadrunner.hdmap.Sign]
```

Create two road lanes using the `roadrunner.hdmap.Lane` object. Specify the lane information for the lane id, coordinates defining the lane geometry, driving direction, and lane type.

```
rrMap.Lanes(2) = roadrunner.hdmap.Lane(ID="Lane2", Geometry=[-0.782 -1.56;50.78 23.43], TravelDire
```

```
rrMap =
  roadrunnerHDMAP with properties:
```

```

    Author: ""
  Projection: ""
```

```

GeographicBoundary: [0×3 double]
  Lanes: [2×1 roadrunner.hdmap.Lane]
LaneBoundaries: [0×1 roadrunner.hdmap.LaneBoundary]
  LaneGroups: [0×1 roadrunner.hdmap.LaneGroup]
  LaneMarkings: [0×1 roadrunner.hdmap.LaneMarking]
  Junctions: [0×1 roadrunner.hdmap.Junction]
BarrierTypes: [0×1 roadrunner.hdmap.BarrierType]
  Barriers: [0×1 roadrunner.hdmap.Barrier]
  SignTypes: [0×1 roadrunner.hdmap.SignType]
  Signs: [0×1 roadrunner.hdmap.Sign]

```

```
rrMap.Lanes(1) = roadrunner.hdmap.Lane(ID="Lane1",Geometry=[-0.534 -2.43;25.78 46.43],TravelDire
```

```
rrMap =
```

```
roadrunnerHDMAP with properties:
```

```

  Author: ""
  Projection: ""
GeographicBoundary: [0×3 double]
  Lanes: [2×1 roadrunner.hdmap.Lane]
LaneBoundaries: [0×1 roadrunner.hdmap.LaneBoundary]
  LaneGroups: [0×1 roadrunner.hdmap.LaneGroup]
  LaneMarkings: [0×1 roadrunner.hdmap.LaneMarking]
  Junctions: [0×1 roadrunner.hdmap.Junction]
BarrierTypes: [0×1 roadrunner.hdmap.BarrierType]
  Barriers: [0×1 roadrunner.hdmap.Barrier]
  SignTypes: [0×1 roadrunner.hdmap.SignType]
  Signs: [0×1 roadrunner.hdmap.Sign]

```

Add Lane2 as a successor of Lane1.

```
addSuccessor(rrMap.Lanes(1), "Lane2")
```

```
ans =
```

```
AlignedReference with properties:
```

```

  Reference: [1×1 roadrunner.hdmap.Reference]
  Alignment: Unspecified

```

Input Arguments

aLane — Lane properties of road

roadrunner.hdmap.Lane object

Lane properties of a road, specified as a roadrunner.hdmap.Lane object.

Example: aLane= roadrunner.hdmap.Lane(ID="Lane1", Geometry=[-0.782 -1.56;50.78 23.43], LaneType="Driving", TravelDirection="Forward") creates a lane with properties lane id, coordinates defining lane geometry, lane type, and driving direction.

successorID — ID of successor lane

character vector | string scalar

ID of the successor lane, specified as a string.

Example: `addPredecessor(rrMap.Lanes(2),"Lane1")` assigns `Lane1` as the predecessor of `Lane2` in a RoadRunner HD Map.

Data Types: `char` | `string`

Name-Value Pair Arguments

Specify optional pairs of arguments as `Name1=Value1, ..., NameN=ValueN`, where `Name` is the argument name and `Value` is the corresponding value. Name-value arguments must appear after other arguments, but the order of the pairs does not matter.

Example:

Alignment — Type of alignment

`character vector` | `string scalar`

Type of alignment between linked objects, specified as a character vector or string scalar. The type of alignment is specified as one of the following strings: `"Unspecified"`, `"Forward"`, or `"Backward"`.

Example: `alref= roadrunner.hdmap.AlignedReference(Reference=ref, Alignment="Forward")` creates the aligned reference for `Lane2` with `"Forward"` alignment.

Data Types: `char` | `string`

Output Arguments

aSuccessor — Linkage information for succeeding lanes

`roadrunner.hdmap.AlignedReference` object

Linkage information for succeeding lanes, specified as a `roadrunner.hdmap.AlignedReference` object.

Version History

Introduced in R2022b

See Also

`roadrunnerHDMAP` | `roadrunner.hdmap.Lane` | `roadrunner.hdmap.AlignedReference` | `addPredecessor`

Topics

"Build Simple Roads Programmatically Using RoadRunner HD Map"

"Build Scenes from Custom Data Using RoadRunner HD Map"

leftBoundary

Set left boundary of lane in RoadRunner HD Map using MATLAB

Syntax

```
aLeftBoundary = leftBoundary(aLane, laneBoundaryID)
aLeftBoundary = leftBoundary(aLane, laneBoundaryID, Name=Value)
```

Description

`aLeftBoundary = leftBoundary(aLane, laneBoundaryID)` sets a boundary with ID `laneBoundaryID` to the `LeftLaneBoundary` property of the lane and returns it.

`aLeftBoundary = leftBoundary(aLane, laneBoundaryID, Name=Value)` sets options using one or more name-value arguments.

Examples

Set Left Boundary to a Lane in RoadRunner HD Map

Create an empty RoadRunner HD Map by calling the `roadrunnerHDMMap` object.

```
rrMap = roadrunnerHDMMap()

rrMap =
  roadrunnerHDMMap with properties:

    Author: ""
  Projection: ""
GeographicBoundary: [0x3 double]
    Lanes: [0x1 roadrunner.hdmap.Lane]
LaneBoundaries: [0x1 roadrunner.hdmap.LaneBoundary]
  LaneGroups: [0x1 roadrunner.hdmap.LaneGroup]
LaneMarkings: [0x1 roadrunner.hdmap.LaneMarking]
  Junctions: [0x1 roadrunner.hdmap.Junction]
BarrierTypes: [0x1 roadrunner.hdmap.BarrierType]
  Barriers: [0x1 roadrunner.hdmap.Barrier]
  SignTypes: [0x1 roadrunner.hdmap.SignType]
    Signs: [0x1 roadrunner.hdmap.Sign]
```

Add lane and lane boundary to the map.

```
rrMap.Lanes = roadrunner.hdmap.Lane(ID="Lane1")

rrMap =
  roadrunnerHDMMap with properties:

    Author: ""
  Projection: ""
GeographicBoundary: [0x3 double]
```

```

        Lanes: [1x1 roadrunner.hdmap.Lane]
LaneBoundaries: [0x1 roadrunner.hdmap.LaneBoundary]
    LaneGroups: [0x1 roadrunner.hdmap.LaneGroup]
    LaneMarkings: [0x1 roadrunner.hdmap.LaneMarking]
        Junctions: [0x1 roadrunner.hdmap.Junction]
    BarrierTypes: [0x1 roadrunner.hdmap.BarrierType]
        Barriers: [0x1 roadrunner.hdmap.Barrier]
    SignTypes: [0x1 roadrunner.hdmap.SignType]
        Signs: [0x1 roadrunner.hdmap.Sign]

```

```
rrMap.LaneBoundaries = roadrunner.hdmap.LaneBoundary(ID="LaneBoundaryL")
```

```
rrMap =
    roadrunnerHDMAP with properties:
```

```

        Author: ""
        Projection: ""
    GeographicBoundary: [0x3 double]
        Lanes: [1x1 roadrunner.hdmap.Lane]
    LaneBoundaries: [1x1 roadrunner.hdmap.LaneBoundary]
        LaneGroups: [0x1 roadrunner.hdmap.LaneGroup]
    LaneMarkings: [0x1 roadrunner.hdmap.LaneMarking]
        Junctions: [0x1 roadrunner.hdmap.Junction]
    BarrierTypes: [0x1 roadrunner.hdmap.BarrierType]
        Barriers: [0x1 roadrunner.hdmap.Barrier]
    SignTypes: [0x1 roadrunner.hdmap.SignType]
        Signs: [0x1 roadrunner.hdmap.Sign]

```

Set the left lane boundary of the lane using the `leftBoundary` function.

```
leftBoundary(rrMap.Lanes, "LaneBoundaryL");
```

Input Arguments

aLane — Lane properties of road

roadrunner.hdmap.Lane object

Lane properties of a road, specified as a `roadrunner.hdmap.Lane` object.

Example: `aLane= roadrunner.hdmap.Lane(ID="Lane1", Geometry=[-0.782 -1.56;50.78 23.43], LaneType="Driving", TravelDirection="Forward")` creates a lane with properties lane id, coordinates defining lane geometry, lane type, and driving direction.

laneBoundaryID — ID of left lane boundary

character vector | string scalar

ID of the left lane boundary, specified as a string.

Example: `leftBoundary(rrMap.Lanes, "LaneBoundaryL")` assigns the left boundary with an id `LaneBoundaryL` to a lane in a RoadRunner HD Map.

Data Types: char | string

Name-Value Pair Arguments

Specify optional pairs of arguments as `Name1=Value1, . . . , NameN=ValueN`, where `Name` is the argument name and `Value` is the corresponding value. Name-value arguments must appear after other arguments, but the order of the pairs does not matter.

Example:

Alignment — Type of alignment

character vector | string scalar

Type of alignment between linked objects, specified as a character vector or string scalar. The type of alignment is specified as one of the following strings: "Unspecified", "Forward", or "Backward".

Example: `aLref= roadrunner.hdmap.AlignedReference(Reference=ref, Alignment="Forward")` creates the aligned reference for Lane2 with "Forward" alignment.

Data Types: `char` | `string`

Output Arguments

aLeftBoundary — Linkage information for left boundary of lane

`roadrunner.hdmap.AlignedReference` object

Linkage information for the left boundary of the lane, specified as a `roadrunner.hdmap.AlignedReference` object.

Version History

Introduced in R2022b

See Also

`roadrunnerHDMAP` | `roadrunner.hdmap.Lane` | `roadrunner.hdmap.AlignedReference` | `rightBoundary`

Topics

“Build Simple Roads Programmatically Using RoadRunner HD Map”

“Build Scenes from Custom Data Using RoadRunner HD Map”

rightBoundary

Set right boundary of lane in RoadRunner HD Map using MATLAB

Syntax

```
aRightBoundary = rightBoundary(aLane, laneBoundaryID)
aRightBoundary = rightBoundary(aLane, laneBoundaryID, Name=Value)
```

Description

`aRightBoundary = rightBoundary(aLane, laneBoundaryID)` sets a boundary with ID `laneBoundaryID` to the `RightLaneBoundary` property of the lane and returns it.

`aRightBoundary = rightBoundary(aLane, laneBoundaryID, Name=Value)` sets options using one or more name-value arguments.

Examples

Set Right Boundary to a Lane in RoadRunner HD Map

Create an empty RoadRunner HD Map by calling the `roadrunnerHDMMap` object.

```
rrMap = roadrunnerHDMMap()

rrMap =
  roadrunnerHDMMap with properties:
    Author: ""
    Projection: ""
    GeographicBoundary: [0x3 double]
    Lanes: [0x1 roadrunner.hdmap.Lane]
    LaneBoundaries: [0x1 roadrunner.hdmap.LaneBoundary]
    LaneGroups: [0x1 roadrunner.hdmap.LaneGroup]
    LaneMarkings: [0x1 roadrunner.hdmap.LaneMarking]
    Junctions: [0x1 roadrunner.hdmap.Junction]
    BarrierTypes: [0x1 roadrunner.hdmap.BarrierType]
    Barriers: [0x1 roadrunner.hdmap.Barrier]
    SignTypes: [0x1 roadrunner.hdmap.SignType]
    Signs: [0x1 roadrunner.hdmap.Sign]
```

Add lane and lane boundary to the map.

```
rrMap.Lanes = roadrunner.hdmap.Lane(ID="Lane1")

rrMap =
  roadrunnerHDMMap with properties:
    Author: ""
    Projection: ""
    GeographicBoundary: [0x3 double]
```



```

    Lanes: [1x1 roadrunner.hdmap.Lane]
  LaneBoundaries: [0x1 roadrunner.hdmap.LaneBoundary]
    LaneGroups: [0x1 roadrunner.hdmap.LaneGroup]
  LaneMarkings: [0x1 roadrunner.hdmap.LaneMarking]
    Junctions: [0x1 roadrunner.hdmap.Junction]
  BarrierTypes: [0x1 roadrunner.hdmap.BarrierType]
    Barriers: [0x1 roadrunner.hdmap.Barrier]
  SignTypes: [0x1 roadrunner.hdmap.SignType]
    Signs: [0x1 roadrunner.hdmap.Sign]

```

```
rrMap.LaneBoundaries = roadrunner.hdmap.LaneBoundary(ID="LaneBoundaryR")
```

```
rrMap =
  roadrunnerHDMAP with properties:
```

```

    Author: ""
    Projection: ""
  GeographicBoundary: [0x3 double]
    Lanes: [1x1 roadrunner.hdmap.Lane]
  LaneBoundaries: [1x1 roadrunner.hdmap.LaneBoundary]
    LaneGroups: [0x1 roadrunner.hdmap.LaneGroup]
  LaneMarkings: [0x1 roadrunner.hdmap.LaneMarking]
    Junctions: [0x1 roadrunner.hdmap.Junction]
  BarrierTypes: [0x1 roadrunner.hdmap.BarrierType]
    Barriers: [0x1 roadrunner.hdmap.Barrier]
  SignTypes: [0x1 roadrunner.hdmap.SignType]
    Signs: [0x1 roadrunner.hdmap.Sign]

```

Set the right lane boundary of the lane using the `rightBoundary` function.

```
rightBoundary(rrMap.Lanes, "LaneBoundaryR");
```

Input Arguments

aLane — Lane properties of road

roadrunner.hdmap.Lane object

Lane properties of a road, specified as a `roadrunner.hdmap.Lane` object.

Example: `aLane= roadrunner.hdmap.Lane(ID="Lane1", Geometry=[-0.782 -1.56;50.78 23.43], LaneType="Driving", TravelDirection="Forward")` creates a lane with properties lane id, coordinates defining lane geometry, lane type, and driving direction.

laneBoundaryID — ID of left lane boundary

character vector | string scalar

ID of the left lane boundary, specified as a string.

Example: `leftBoundary(rrMap.Lanes, "LaneBoundaryL")` assigns the left boundary with an id `LaneBoundaryL` to a lane in a RoadRunner HD Map.

Data Types: char | string

Name-Value Pair Arguments

Specify optional pairs of arguments as `Name1=Value1, . . . , NameN=ValueN`, where `Name` is the argument name and `Value` is the corresponding value. Name-value arguments must appear after other arguments, but the order of the pairs does not matter.

Example:

Alignment — Type of alignment

character vector | string scalar

Type of alignment between linked objects, specified as a character vector or string scalar. The type of alignment is specified as one of the following strings: "Unspecified", "Forward", or "Backward".

Example: `alref = roadrunner.hdmap.AlignedReference(Reference=ref, Alignment="Forward")` creates the aligned reference for `Lane2` with "Forward" alignment.

Data Types: `char` | `string`

Output Arguments

aRightBoundary — Linkage information for right boundary of lane

`roadrunner.hdmap.AlignedReference` object

Linkage information for the right boundary of the lane, specified as a `roadrunner.hdmap.AlignedReference` object.

Version History

Introduced in R2022b

See Also

`roadrunnerHDMAP` | `roadrunner.hdmap.Lane` | `roadrunner.hdmap.AlignedReference` | `leftBoundary`

Topics

"Build Simple Roads Programmatically Using RoadRunner HD Map"

"Build Scenes from Custom Data Using RoadRunner HD Map"

roadrunner.hdmap.LaneBoundary

Create lane boundary in RoadRunner HD Map using MATLAB

Description

A `roadrunner.hdmap.LaneBoundary` object enables you to define the border of a lane in a RoadRunner HD Map scene model. One or more lanes can optionally contain a reference to lane boundaries on either sides of the lane. Lane boundaries optionally contain definitions for parametric attributes. These attributes define the lane markings over a span. Multiple lane markings can be spanned over a lane boundary as a parametric range between [0,1] of the boundary's length.

Creation

Syntax

```
aLaneBoundary = roadrunner.hdmap.LaneBoundary()  
aLaneBoundary = roadrunner.hdmap.LaneBoundary(Name=Value)
```

Description

`aLaneBoundary = roadrunner.hdmap.LaneBoundary()` creates an empty lane boundary.

`aLaneBoundary = roadrunner.hdmap.LaneBoundary(Name=Value)` sets the properties of the lane boundary using name-value pairs.

Properties

ID — ID of lane boundary

character vector | string scalar

ID of the lane boundary, specified as a character vector or string scalar.

Example: `rrMap.LaneBoundaries = roadrunner.hdmap.LaneBoundary(ID="LaneBoundary1")` creates a lane boundary with id `Lane1Boundary` in a RoadRunner HD Map.

Data Types: `char` | `string`

Geometry — 3D coordinates of lane boundary points

three element vector

3D coordinates of the lane boundary points, specified as a three element vector, with double precision values.

Example: `rrMap.LanesBoundaries = roadrunner.hdmap.LaneBoundary(ID="LaneBoundary1", Geometry=[-0.782 -1.56;50.78 23.43])` creates a lane boundary with id `"LaneBoundary1"` and defines the coordinates for the lane boundary geometry.

Data Types: `double`

ParametricAttributes — Span-based parametric attributes to create lane markings

array of `roadrunner.hdmap.ParametricAttribution` objects

Span-based parametric attributes to create lane markings, specified as an array of `roadrunner.hdmap.ParametricAttribution` objects.

Examples**Add Lane Boundary to RoadRunner HD Map**

Create an empty RoadRunner HD Map by calling the `roadrunnerHDMMap` object.

```
rrMap = roadrunnerHDMMap()
```

```
rrMap =  
  roadrunnerHDMMap with properties:  
  
    Author: ""  
    Projection: ""  
    GeographicBoundary: [0×3 double]  
    Lanes: [0×1 roadrunner.hdmap.Lane]  
    LaneBoundaries: [0×1 roadrunner.hdmap.LaneBoundary]  
    LaneGroups: [0×1 roadrunner.hdmap.LaneGroup]  
    LaneMarkings: [0×1 roadrunner.hdmap.LaneMarking]  
    Junctions: [0×1 roadrunner.hdmap.Junction]  
    BarrierTypes: [0×1 roadrunner.hdmap.BarrierType]  
    Barriers: [0×1 roadrunner.hdmap.Barrier]  
    SignTypes: [0×1 roadrunner.hdmap.SignType]  
    Signs: [0×1 roadrunner.hdmap.Sign]
```

Create the road lane boundary using the `roadrunner.hdmap.LaneBoundary` object. Specify the lane information for the lane id and coordinates defining the lane geometry.

```
rrMap.LaneBoundaries = roadrunner.hdmap.LaneBoundary(ID="LaneBoundary1", Geometry=[50 25; 100 50])
```

```
rrMap =  
  roadrunnerHDMMap with properties:  
  
    Author: ""  
    Projection: ""  
    GeographicBoundary: [0×3 double]  
    Lanes: [0×1 roadrunner.hdmap.Lane]  
    LaneBoundaries: [1×1 roadrunner.hdmap.LaneBoundary]  
    LaneGroups: [0×1 roadrunner.hdmap.LaneGroup]  
    LaneMarkings: [0×1 roadrunner.hdmap.LaneMarking]  
    Junctions: [0×1 roadrunner.hdmap.Junction]  
    BarrierTypes: [0×1 roadrunner.hdmap.BarrierType]  
    Barriers: [0×1 roadrunner.hdmap.Barrier]  
    SignTypes: [0×1 roadrunner.hdmap.SignType]  
    Signs: [0×1 roadrunner.hdmap.Sign]
```

Version History

Introduced in R2022b

See Also

roadrunnerHDMap | roadrunner.hdmap.Lane

Topics

“Build Simple Roads Programmatically Using RoadRunner HD Map”

“Build Scenes from Custom Data Using RoadRunner HD Map”

roadrunner.hdmap.MarkingReference

Defines the information for linking lane boundary and lane marking in in RoadRunner HD Map using MATLAB

Description

A `roadrunner.hdmap.MarkingReference` object enables you to define the information for linking lane boundary and lane marking in a RoadRunner HD Map scene model.

Creation

Syntax

```
mrkref = roadrunner.hdmap.MarkingReference()  
mrkref = roadrunner.hdmap.MarkingReference(Name=Value)
```

Description

`mrkref = roadrunner.hdmap.MarkingReference()` creates an empty marking reference.

`mrkref = roadrunner.hdmap.MarkingReference(Name=Value)` sets the properties of the marking reference using name-value pairs.

Properties

MarkingID — ID of lane marking element

`roadrunner.hdmap.Reference` object

ID of the lane marking element, specified as a `roadrunner.hdmap.Reference` object.

FlipLaterally — Set flip marking orientation

`true` or `1` | `false` or `0`

Set flip marking orientation, specified as logical `0` (`false`) or logical `1` (`true`). Set this field to logical `1` (`true`) to flip the order of the lane marking stripes.

Examples

Create Marking Reference in RoadRunner HD Map

Create an reference for a marking asset.

```
ref = roadrunner.hdmap.Reference(ID="Dashed1")
```

```
ref =  
    Reference with properties:
```

```
ID: "Dashed1"
```

Create a marking reference for the marking to link the lane boundary and the lane marking.

```
mrkref = roadrunner.hdmap.MarkingReference(MarkingID=ref,FlipLaterally=true)
```

```
mrkref =
```

```
  MarkingReference with properties:
```

```
    MarkingID: [1x1 roadrunner.hdmap.Reference]
```

```
    FlipLaterally: 1
```

Version History

Introduced in R2022b

See Also

[roadrunnerHDMAP](#) | [roadrunner.hdmap.Lane](#) | [roadrunner.hdmap.LaneBoundary](#) | [roadrunner.hdmap.ParametricAttribution](#)

Topics

["Build Simple Roads Programmatically Using RoadRunner HD Map"](#)

["Build Scenes from Custom Data Using RoadRunner HD Map"](#)

roadrunner.hdmap.ParametricAttribution

Span-based parametric attributes

Description

A `roadrunner.hdmap.ParametricAttribution` object enables you to define the span-based parametric attributes for lane boundaries in a RoadRunner HD Map scene model.

Creation

Syntax

```
prmAttr = roadrunner.hdmap.ParametricAttribution()  
prmAttr = roadrunner.hdmap.ParametricAttribution(Name=Value)
```

Description

`prmAttr = roadrunner.hdmap.ParametricAttribution()` creates an empty parametric attribution for a lane boundary.

`prmAttr = roadrunner.hdmap.ParametricAttribution(Name=Value)` sets the properties of the parametric attribution for lane boundary using name-value pairs.

Properties

Span — Range for span-based attributes

two element vector

Range for span based attributes, specified as a two element vector [`spanStart` `spanEnd`]. Using this property, you can insert span nodes that define a range along a curve-based parent object. You can change attributes along the specified range. For example, you can specify a range for which to change a lane marking along a lane boundary. `spanStart` is the normalized distance in the range [0, 1] between the start of the current span and the start of the parent object. Use the length of the parent object to get the normalized value. `spanEnd` is the normalized distance in the range [0, 1] between the end of the current span and the start of the parent object. Use the length of the parent object to get the normalized value.

Data Types: double

MarkingReference — Reference to marking defined in RoadRunner HD Map

`roadrunner.hdmap.MarkingReference` object

Reference to marking defined in a RoadRunner HD Map, specified as a `roadrunner.hdmap.MarkingReference` object. This property defines the linking information for lane marking. You can specify ID, and offset attributes to place lane markings.

Examples

Create Parametric Attributes in RoadRunner HD Map

Create an reference for a marking asset.

```
ref = roadrunner.hdmap.Reference(ID="Dashed1")
```

```
ref =
  Reference with properties:
```

```
    ID: "Dashed1"
```

```
mrkref = roadrunner.hdmap.MarkingReference(MarkingID=ref)
```

```
mrkref =
  MarkingReference with properties:
```

```
    MarkingID: [1x1 roadrunner.hdmap.Reference]
    FlipLaterally: 0
```

Create a parametric attribution for the marking with reference id Dashed1

```
prmAttr = roadrunner.hdmap.ParametricAttribution(Span=[0.1 0.3], MarkingReference=mrkref)
```

```
prmAttr =
  ParametricAttribution with properties:
```

```
    Span: [0.1000 0.3000]
    MarkingReference: [1x1 roadrunner.hdmap.MarkingReference]
```

Version History

Introduced in R2022b

See Also

roadrunnerHDMAP | roadrunner.hdmap.Lane | roadrunner.hdmap.LaneBoundary

Topics

“Build Simple Roads Programmatically Using RoadRunner HD Map”

“Build Scenes from Custom Data Using RoadRunner HD Map”

roadrunner.hdmap.LaneGroup

Create lane groups in RoadRunner HD Map using MATLAB

Description

A `roadrunner.hdmap.LaneGroup` object enables you to define a group of lanes with similar geometry in a RoadRunner HD Map scene model. The `LaneGroup` geometry is used as a general approximation for the geometry of the lanes contained within that group. The hierarchy of the lane elements is defined as:

- A lane group contains references to lanes.
- A lane has references to its boundaries and its connected lanes.
- A lane boundary has references to the lane markings.

Creation

Syntax

```
aLaneGroup = roadrunner.hdmap.LaneGroup()  
aLaneGroup = roadrunner.hdmap.LaneGroup(Name=Value)
```

Description

`aLaneGroup = roadrunner.hdmap.LaneGroup()` creates an empty lane group collection.

`aLaneGroup = roadrunner.hdmap.LaneGroup(Name=Value)` sets the properties of the lane group collection using name-value pairs.

Properties

ID — ID of lane group

character vector | string scalar

ID of the lane group, specified as a character vector or string scalar.

Example: `rrMap.LaneGroups = roadrunner.hdmap.LaneGroup(ID="LaneGroup1")` creates a lane group with id `LaneGroup1` in a RoadRunner HD Map.

Data Types: `char` | `string`

Geometry — 3D coordinates of geometric shape of lane group

three element vector

3D coordinates of the geometric shape of the lane group, specified as a three element vector, with double precision values.

Example: `rrMap.LaneGroups = roadrunner.hdmap.LaneGroup(ID="LaneGroup1", Geometry=[-0.782 -1.56;50.78 23.43])` creates a lane group with id `"LaneGroup1"` and defines the coordinates for the geometric shape of the lane group.

Data Types: double

Lanes — Collection of lane references

array of `roadrunner.hdmap.AlignedReference` objects

Collection of lane references, specified as an array of `roadrunner.hdmap.AlignedReference` objects. This property defines the linkage information for lanes within the lane group.

Examples

Add Lane Group to RoadRunner HD Map

Create an empty RoadRunner HD Map by calling the `roadrunnerHDMAP` object.

```
rrMap = roadrunnerHDMAP()
```

```
rrMap =
  roadrunnerHDMAP with properties:
```

```

    Author: ""
    Projection: ""
    GeographicBoundary: [0x3 double]
        Lanes: [0x1 roadrunner.hdmap.Lane]
    LaneBoundaries: [0x1 roadrunner.hdmap.LaneBoundary]
    LaneGroups: [0x1 roadrunner.hdmap.LaneGroup]
    LaneMarkings: [0x1 roadrunner.hdmap.LaneMarking]
    Junctions: [0x1 roadrunner.hdmap.Junction]
    BarrierTypes: [0x1 roadrunner.hdmap.BarrierType]
    Barriers: [0x1 roadrunner.hdmap.Barrier]
    SignTypes: [0x1 roadrunner.hdmap.SignType]
    Signs: [0x1 roadrunner.hdmap.Sign]
```

Create the road lane group using the `roadrunner.hdmap.LaneGroup` object. Specify the lane information for the lane id.

```
rrMap.LaneGroups = roadrunner.hdmap.LaneGroup(ID="LaneGroup1")
```

```
rrMap =
  roadrunnerHDMAP with properties:
```

```

    Author: ""
    Projection: ""
    GeographicBoundary: [0x3 double]
        Lanes: [0x1 roadrunner.hdmap.Lane]
    LaneBoundaries: [0x1 roadrunner.hdmap.LaneBoundary]
    LaneGroups: [1x1 roadrunner.hdmap.LaneGroup]
    LaneMarkings: [0x1 roadrunner.hdmap.LaneMarking]
    Junctions: [0x1 roadrunner.hdmap.Junction]
    BarrierTypes: [0x1 roadrunner.hdmap.BarrierType]
    Barriers: [0x1 roadrunner.hdmap.Barrier]
    SignTypes: [0x1 roadrunner.hdmap.SignType]
    Signs: [0x1 roadrunner.hdmap.Sign]
```

Version History

Introduced in R2022b

See Also

roadrunnerHDMAP | roadrunner.hdmap.Lane | roadrunner.hdmap.LaneBoundary |
roadrunner.hdmap.LaneMarking

Topics

“Build Simple Roads Programmatically Using RoadRunner HD Map”

“Build Scenes from Custom Data Using RoadRunner HD Map”

roadrunner.hdmap.LaneMarking

Create lane markings in RoadRunner HD Map using MATLAB

Description

A `roadrunner.hdmap.LaneMarking` object enables you to define the road lane marking object in a RoadRunner HD Map scene model.

Creation

Syntax

```
aLaneMarking = roadrunner.hdmap.LaneMarking()  
aLaneMarking = roadrunner.hdmap.LaneMarking(Name=Value)
```

Description

`aLaneMarking = roadrunner.hdmap.LaneMarking()` creates an empty lane marking.

`aLaneMarking = roadrunner.hdmap.LaneMarking(Name=Value)` sets the properties of the lane marking using name-value pairs.

Properties

ID — ID of lane marking element

character vector | string scalar

ID of the lane marking element, specified as a character vector or string scalar. The ID can be referenced by the lane elements using `roadrunner.hdmap.MarkingReference` object.

Data Types: `char` | `string`

AssetPath — Relative path for lane marking asset

`roadrunner.hdmap.RelativeAssetPath` object

Relative path for lane marking asset, specified as a `roadrunner.hdmap.RelativeAssetPath` object. To define the lane markings on a span, `RelativeAssetPath` can be used to reference existing marking assets in the library.

Examples

Add Lane Marking to RoadRunner HD Map

Create an empty RoadRunner HD Map by calling the `roadrunnerHDMAP` object.

```
rrMap = roadrunnerHDMAP()
```

```
rrMap =
  roadrunnerHDMAP with properties:
    Author: ""
    GeoReference: [0 0]
    GeographicBoundary: []
    Lanes: [0x1 roadrunner.hdmap.Lane]
    LaneBoundaries: [0x1 roadrunner.hdmap.LaneBoundary]
    LaneGroups: [0x1 roadrunner.hdmap.LaneGroup]
    LaneMarkings: [0x1 roadrunner.hdmap.LaneMarking]
    Junctions: [0x1 roadrunner.hdmap.Junction]
    BarrierTypes: [0x1 roadrunner.hdmap.BarrierType]
    Barriers: [0x1 roadrunner.hdmap.Barrier]
    SignTypes: [0x1 roadrunner.hdmap.SignType]
    Signs: [0x1 roadrunner.hdmap.Sign]
    StaticObjectTypes: [0x1 roadrunner.hdmap.StaticObjectType]
    StaticObjects: [0x1 roadrunner.hdmap.StaticObject]
```

Create a relative path to an asset. In this example, we create a relative path to the dashed single white asset. This path is relative to the `Assets` folder of your RoadRunner project.

```
path = roadrunner.hdmap.RelativeAssetPath(AssetPath="/Assets/Markings/DashedSingleWhite.rrlms")
```

```
path =
  RelativeAssetPath with properties:
    AssetPath: "/Assets/Markings/DashedSingleWhite.rrlms"
```

Create the road lane marking using the `roadrunner.hdmap.LaneMarking` object. Specify the lane marking information for the lane marking id and the asset path.

```
rrMap.LaneMarkings = roadrunner.hdmap.LaneMarking(ID="Dashed1", AssetPath=path)
```

```
rrMap =
  roadrunnerHDMAP with properties:
    Author: ""
    GeoReference: [0 0]
    GeographicBoundary: []
    Lanes: [0x1 roadrunner.hdmap.Lane]
    LaneBoundaries: [0x1 roadrunner.hdmap.LaneBoundary]
    LaneGroups: [0x1 roadrunner.hdmap.LaneGroup]
    LaneMarkings: [1x1 roadrunner.hdmap.LaneMarking]
    Junctions: [0x1 roadrunner.hdmap.Junction]
    BarrierTypes: [0x1 roadrunner.hdmap.BarrierType]
    Barriers: [0x1 roadrunner.hdmap.Barrier]
    SignTypes: [0x1 roadrunner.hdmap.SignType]
    Signs: [0x1 roadrunner.hdmap.Sign]
    StaticObjectTypes: [0x1 roadrunner.hdmap.StaticObjectType]
    StaticObjects: [0x1 roadrunner.hdmap.StaticObject]
```

Version History

Introduced in R2022b

See Also

[roadrunnerHDMAP](#) | [roadrunner.hdmap.Lane](#) | [roadrunner.hdmap.LaneBoundary](#) | [roadrunner.hdmap.LaneGroup](#)

Topics

[“Build Simple Roads Programmatically Using RoadRunner HD Map”](#)

[“Build Scenes from Custom Data Using RoadRunner HD Map”](#)

roadrunner.hdmap.RelativeAssetPath

Defines relative path for lane marking asset in RoadRunner HD Map using MATLAB

Description

A `roadrunner.hdmap.RelativeAssetPath` object enables you to define the road lane marking object in a RoadRunner HD Map scene model. The `RelativeAssetPath` refers to the assets in the asset library by using the directory path relative to the project directory.

Creation

Syntax

```
assetPath = roadrunner.hdmap.RelativeAssetPath()  
assetPath = roadrunner.hdmap.RelativeAssetPath(Name=Value)
```

Description

`assetPath = roadrunner.hdmap.RelativeAssetPath()` creates an empty relative asset path.

`assetPath = roadrunner.hdmap.RelativeAssetPath(Name=Value)` sets the properties of the relative asset path using name-value pairs.

Properties

AssetPath — Relative path for asset in asset library

character vector | string scalar

Relative path for an asset in the asset library, specified as a character vector or string scalar.

Example: `"Assets/Markings/DashedSingleWhite.rrlms"` sets the path for the white dashed lane marking asset.

Data Types: `char` | `string`

Examples

Add an Asset to a Lane Marking in RoadRunner HD Map

Create an empty RoadRunner HD Map by calling the `roadrunnerHDMMap` object.

```
rrMap = roadrunnerHDMMap()
```

```
rrMap =  
    roadrunnerHDMMap with properties:
```

```
    Author: ""
```



```

    Projection: ""
  GeographicBoundary: [0x3 double]
    Lanes: [0x1 roadrunner.hdmap.Lane]
  LaneBoundaries: [0x1 roadrunner.hdmap.LaneBoundary]
  LaneGroups: [0x1 roadrunner.hdmap.LaneGroup]
  LaneMarkings: [0x1 roadrunner.hdmap.LaneMarking]
  Junctions: [0x1 roadrunner.hdmap.Junction]
  BarrierTypes: [0x1 roadrunner.hdmap.BarrierType]
  Barriers: [0x1 roadrunner.hdmap.Barrier]
  SignTypes: [0x1 roadrunner.hdmap.SignType]
  Signs: [0x1 roadrunner.hdmap.Sign]

```

Create a relative path to an asset. In this example, we create a relative path to the dashed single white asset. This path is relative to the `Assets` folder of your RoadRunner project.

```
path = roadrunner.hdmap.RelativeAssetPath(AssetPath="/Asset/Markings/DashedSingleWhite.rrlms")
```

```
path =
```

```
  RelativeAssetPath with properties:
```

```
    AssetPath: "/Asset/Markings/DashedSingleWhite.rrlms"
```

Create the road lane marking using the `roadrunner.hdmap.LaneMarking` object and add the asset as a lane marking.

```
rrMap.LaneMarkings = roadrunner.hdmap.LaneMarking(ID="Dashed1", AssetPath=path)
```

```
rrMap =
```

```
  roadrunnerHDMAP with properties:
```

```

    Author: ""
    Projection: ""
  GeographicBoundary: [0x3 double]
    Lanes: [0x1 roadrunner.hdmap.Lane]
  LaneBoundaries: [0x1 roadrunner.hdmap.LaneBoundary]
  LaneGroups: [0x1 roadrunner.hdmap.LaneGroup]
  LaneMarkings: [1x1 roadrunner.hdmap.LaneMarking]
  Junctions: [0x1 roadrunner.hdmap.Junction]
  BarrierTypes: [0x1 roadrunner.hdmap.BarrierType]
  Barriers: [0x1 roadrunner.hdmap.Barrier]
  SignTypes: [0x1 roadrunner.hdmap.SignType]
  Signs: [0x1 roadrunner.hdmap.Sign]

```

Version History

Introduced in R2022b

See Also

`roadrunnerHDMAP` | `roadrunner.hdmap.LaneMarking`

Topics

“Build Simple Roads Programmatically Using RoadRunner HD Map”

“Build Scenes from Custom Data Using RoadRunner HD Map”

roadrunner.hdmap.Junction

Create junctions in RoadRunner HD Map using MATLAB

Description

A `roadrunner.hdmap.Junction` object enables you to define the structure for representing junctions in a RoadRunner HD Map scene model. Junctions represent a logical grouping of a set of lanes, typically involving a crossing (or convergence or divergence) between multiple roadways. Use this object to define the geometry and connectivity of lanes at a crossing between multiple roadways.

Creation

Syntax

```
aJunction = roadrunner.hdmap.Junction()
aJunction = roadrunner.hdmap.Junction(Name=Value)
```

Description

`aJunction = roadrunner.hdmap.Junction()` creates an empty lane junction.

`aJunction = roadrunner.hdmap.Junction(Name=Value)` sets the properties of the junction using name-value pairs.

Properties

ID — ID of junction

character vector | string scalar

ID of the junction, specified as a character vector or string scalar.

Example: `rrMap.Junctions = roadrunner.hdmap.Junction(ID="Junction1")` creates a junction with id `Junction1` in a RoadRunner HD Map.

Data Types: `char` | `string`

Geometry — Geometry of junction object

matrix

Geometry of the junction object, specified as a matrix, with double precision values. The geometry of the junction object is represented by a set of non-overlapping polygons. The non-overlapping set can include polygons within holes of another polygon. A polygon has one exterior boundary, but it can have multiple holes within it. The matrix contains three-element vectors representing the polygons. For each polygon, the first vector contains the coordinates for the outer boundary of the polygon. The second vector contains the coordinates for the boundaries of holes within the polygon.

Data Types: `double`

Lanes — ID of connecting lanes

array of `roadrunner.hdmap.Reference` objects

ID of connecting lanes, specified as an array of `roadrunner.hdmap.Reference` objects. These lanes may connect topologically to other lanes enclosed in the junction. For example, a small roundabout (mini-circle) could be comprised of a single junction whose enclosed lanes include a ring formation.

Examples**Add Junction to RoadRunner HD Map**

Create an empty RoadRunner HD Map by calling the `roadrunnerHDMAP` object.

```
rrMap = roadrunnerHDMAP()
```

```
rrMap =
```

```
roadrunnerHDMAP with properties:
```

```
    Author: ""
    Projection: ""
    GeographicBoundary: [0×3 double]
        Lanes: [0×1 roadrunner.hdmap.Lane]
    LaneBoundaries: [0×1 roadrunner.hdmap.LaneBoundary]
    LaneGroups: [0×1 roadrunner.hdmap.LaneGroup]
    LaneMarkings: [0×1 roadrunner.hdmap.LaneMarking]
    Junctions: [0×1 roadrunner.hdmap.Junction]
    BarrierTypes: [0×1 roadrunner.hdmap.BarrierType]
    Barriers: [0×1 roadrunner.hdmap.Barrier]
    SignTypes: [0×1 roadrunner.hdmap.SignType]
    Signs: [0×1 roadrunner.hdmap.Sign]
```

Create the road lane marking using the `roadrunner.hdmap.Junction` object. Specify the junction information for the junction id.

```
rrMap.Junctions = roadrunner.hdmap.Junction(ID="Junction1")
```

```
rrMap =
```

```
roadrunnerHDMAP with properties:
```

```
    Author: ""
    Projection: ""
    GeographicBoundary: [0×3 double]
        Lanes: [0×1 roadrunner.hdmap.Lane]
    LaneBoundaries: [0×1 roadrunner.hdmap.LaneBoundary]
    LaneGroups: [0×1 roadrunner.hdmap.LaneGroup]
    LaneMarkings: [0×1 roadrunner.hdmap.LaneMarking]
    Junctions: [1×1 roadrunner.hdmap.Junction]
    BarrierTypes: [0×1 roadrunner.hdmap.BarrierType]
    Barriers: [0×1 roadrunner.hdmap.Barrier]
    SignTypes: [0×1 roadrunner.hdmap.SignType]
    Signs: [0×1 roadrunner.hdmap.Sign]
```

Version History

Introduced in R2022b

See Also

roadrunnerHDMAP | roadrunner.hdmap.Lane

Topics

“Build Simple Roads Programmatically Using RoadRunner HD Map”

“Build Scenes from Custom Data Using RoadRunner HD Map”

roadrunner.hdmap.Barrier

Create barrier in RoadRunner HD Map using MATLAB

Description

A `roadrunner.hdmap.Barrier` object enables you to define the structure for representing barriers in a RoadRunner HD Map scene model.

Creation

Syntax

```
aBarrier = roadrunner.hdmap.Barrier()  
aBarrier = roadrunner.hdmap.Barrier(Name=Value)
```

Description

`aBarrier = roadrunner.hdmap.Barrier()` creates an empty barrier.

`aBarrier = roadrunner.hdmap.Barrier(Name=Value)` sets the properties of the barrier using name-value pairs.

Properties

ID — ID of barrier element

character vector | string scalar

ID of the barrier element, specified as a character vector or string scalar.

Example: `rrMap.Barriers = roadrunner.hdmap.Barrier(ID="Barrier1")` creates a barrier with id `Lane1` in a RoadRunner HD Map.

Data Types: `char` | `string`

Geometry — Base of barrier that extrusion is repeated over

three element vector

Base of barrier that extrusion is repeated over, specified as a three element vector, with double precision values.

Data Types: `double`

BarrierTypeReference — Type of barrier

`roadrunner.hdmap.Reference` object

Type of barrier, specified as a `roadrunner.hdmap.Reference` object.

FlipLaterally — Set flip barrier orientation

`true` or `1` | `false` or `0`

Set flip barrier orientation, specified as `logical 0 (false)` or `logical 1 (true)`. Set this field to `logical 1 (true)` to flip the orientation of barrier along the lateral direction. Use this field to flip the primary face of the extrusion geometry, such as the guard rails shown in these images.

Examples

Add Barrier to RoadRunner HD Map

Create an empty RoadRunner HD Map by calling the `roadrunnerHDMMap` object.

```
rrMap = roadrunnerHDMMap()

rrMap =
  roadrunnerHDMMap with properties:
    Author: ""
    Projection: ""
    GeographicBoundary: [0x3 double]
    Lanes: [0x1 roadrunner.hdmap.Lane]
    LaneBoundaries: [0x1 roadrunner.hdmap.LaneBoundary]
    LaneGroups: [0x1 roadrunner.hdmap.LaneGroup]
    LaneMarkings: [0x1 roadrunner.hdmap.LaneMarking]
    Junctions: [0x1 roadrunner.hdmap.Junction]
    BarrierTypes: [0x1 roadrunner.hdmap.BarrierType]
    Barriers: [0x1 roadrunner.hdmap.Barrier]
    SignTypes: [0x1 roadrunner.hdmap.SignType]
    Signs: [0x1 roadrunner.hdmap.Sign]
```

Create the barrier using the `roadrunner.hdmap.Barrier` object. Specify the barrier information for the barrier id.

```
rrMap.Barriers = roadrunner.hdmap.Barrier(ID="Barrier1")

rrMap =
  roadrunnerHDMMap with properties:
    Author: ""
    Projection: ""
    GeographicBoundary: [0x3 double]
    Lanes: [0x1 roadrunner.hdmap.Lane]
    LaneBoundaries: [0x1 roadrunner.hdmap.LaneBoundary]
    LaneGroups: [0x1 roadrunner.hdmap.LaneGroup]
    LaneMarkings: [0x1 roadrunner.hdmap.LaneMarking]
    Junctions: [0x1 roadrunner.hdmap.Junction]
    BarrierTypes: [0x1 roadrunner.hdmap.BarrierType]
    Barriers: [1x1 roadrunner.hdmap.Barrier]
    SignTypes: [0x1 roadrunner.hdmap.SignType]
    Signs: [0x1 roadrunner.hdmap.Sign]
```

Version History

Introduced in R2022b

See Also

roadrunnerHDMAP | roadrunner.hdmap.BarrierType

Topics

“Build Simple Roads Programmatically Using RoadRunner HD Map”

“Build Scenes from Custom Data Using RoadRunner HD Map”

roadrunner.hdmap.BarrierType

Create barrier types in RoadRunner HD Map using MATLAB

Description

A `roadrunner.hdmap.BarrierType` object enables you to define the barrier extrusion and the type information for a barrier in a RoadRunner HD Map scene model.

Creation

Syntax

```
aBarrierType = roadrunner.hdmap.BarrierType()
aBarrierType = roadrunner.hdmap.BarrierType(Name=Value)
```

Description

`aBarrierType = roadrunner.hdmap.BarrierType()` creates an empty barrier type.

`aBarrierType = roadrunner.hdmap.BarrierType(Name=Value)` sets the properties of the barrier type using name-value pairs.

Properties

ID — ID of barrier type

character vector | string scalar

ID of the barrier type, specified as a character vector or string scalar. The `Barrier` object refers to this field to describe a barrier based on its extruded geometry.

Data Types: `char` | `string`

ExtrusionPath — Relative path for extrusion asset that defines type of barrier

`roadrunner.hdmap.RelativeAssetPath` object

Relative path for extrusion asset that defines type of barrier, specified as a `roadrunner.hdmap.RelativeAssetPath` object. The asset path points to the extrusion information of this barrier type.

Examples

Add Barrier Type to RoadRunner HD Map

Create an empty RoadRunner HD Map by calling the `roadrunnerHDMAP` object.

```
rrMap = roadrunnerHDMAP()
```

```

rrMap =
  roadrunnerHDMAP with properties:
    Author: ""
    GeoReference: [0 0]
    GeographicBoundary: [0x3 double]
    Lanes: [0x1 roadrunner.hdmap.Lane]
    LaneBoundaries: [0x1 roadrunner.hdmap.LaneBoundary]
    LaneGroups: [0x1 roadrunner.hdmap.LaneGroup]
    LaneMarkings: [0x1 roadrunner.hdmap.LaneMarking]
    Junctions: [0x1 roadrunner.hdmap.Junction]
    BarrierTypes: [0x1 roadrunner.hdmap.BarrierType]
    Barriers: [0x1 roadrunner.hdmap.Barrier]
    SignTypes: [0x1 roadrunner.hdmap.SignType]
    Signs: [0x1 roadrunner.hdmap.Sign]
    StaticObjectTypes: [0x1 roadrunner.hdmap.StaticObjectType]
    StaticObjects: [0x1 roadrunner.hdmap.StaticObject]

```

Create an extrusion path to an asset. In this example, create a relative path to a `constantSlopeBarrier` asset file. This path is relative to the Assets folder of your RoadRunner project

```
path = roadrunner.hdmap.RelativeAssetPath(AssetPath="/Assets/Extrusions/ConstantSlopeBarrier.rrect")
```

```

path =
  RelativeAssetPath with properties:
    AssetPath: "/Assets/Extrusions/ConstantSlopeBarrier.rrect"

```

Create the barrier type using the `roadrunner.hdmap.BarrierType` object. Specify the barrier type information for the barrier type id and the extrusion path.

```
rrMap.BarrierTypes = roadrunner.hdmap.BarrierType(ID="BarrierType1", ExtrusionPath=path)
```

```

rrMap =
  roadrunnerHDMAP with properties:
    Author: ""
    GeoReference: [0 0]
    GeographicBoundary: [0x3 double]
    Lanes: [0x1 roadrunner.hdmap.Lane]
    LaneBoundaries: [0x1 roadrunner.hdmap.LaneBoundary]
    LaneGroups: [0x1 roadrunner.hdmap.LaneGroup]
    LaneMarkings: [0x1 roadrunner.hdmap.LaneMarking]
    Junctions: [0x1 roadrunner.hdmap.Junction]
    BarrierTypes: [1x1 roadrunner.hdmap.BarrierType]
    Barriers: [0x1 roadrunner.hdmap.Barrier]
    SignTypes: [0x1 roadrunner.hdmap.SignType]
    Signs: [0x1 roadrunner.hdmap.Sign]
    StaticObjectTypes: [0x1 roadrunner.hdmap.StaticObjectType]
    StaticObjects: [0x1 roadrunner.hdmap.StaticObject]

```

Version History

Introduced in R2022b

See Also

roadrunnerHDMAP | roadrunner.hdmap.Barrier

Topics

“Build Simple Roads Programmatically Using RoadRunner HD Map”

“Build Scenes from Custom Data Using RoadRunner HD Map”

roadrunner.hdmap.Sign

Create signs in RoadRunner HD Map using MATLAB

Description

A `roadrunner.hdmap.Sign` object enables you to define the road sign objects in a RoadRunner HD Map scene model.

Creation

Syntax

```
aSign = roadrunner.hdmap.Sign()  
aSign = roadrunner.hdmap.Sign(Name=Value)
```

Description

`aSign = roadrunner.hdmap.Sign()` creates an empty sign.

`aSign = roadrunner.hdmap.Sign(Name=Value)` sets the properties of the sign using name-value pairs.

Properties

ID — ID of sign element

character vector | string scalar

ID of the sign element, specified as a character vector or string scalar.

Example: `rrMap.Signs = roadrunner.hdmap.Sign(ID="Sign1")` creates a sign with id `Sign1` in a RoadRunner HD Map.

Data Types: `char` | `string`

Geometry — Geometry of 3D stationary object representing the sign

three element vector

Geometry of the 3D stationary object representing the sign, specified as a three element vector, with double precision values. Sign is represented as an oriented bounding box.

Data Types: `double`

SignTypeReference — Type of sign

`roadrunner.hdmap.Reference` object

Type of sign, specified as a `roadrunner.hdmap.Reference` object.

Examples

Add Sign to RoadRunner HD Map

Create an empty RoadRunner HD Map by calling the `roadrunnerHDMMap` object.

```
rrMap = roadrunnerHDMMap()

rrMap =
  roadrunnerHDMMap with properties:

      Author: ""
      Projection: ""
      GeographicBoundary: [0×3 double]
      Lanes: [0×1 roadrunner.hdmap.Lane]
      LaneBoundaries: [0×1 roadrunner.hdmap.LaneBoundary]
      LaneGroups: [0×1 roadrunner.hdmap.LaneGroup]
      LaneMarkings: [0×1 roadrunner.hdmap.LaneMarking]
      Junctions: [0×1 roadrunner.hdmap.Junction]
      BarrierTypes: [0×1 roadrunner.hdmap.BarrierType]
      Barriers: [0×1 roadrunner.hdmap.Barrier]
      SignTypes: [0×1 roadrunner.hdmap.SignType]
      Signs: [0×1 roadrunner.hdmap.Sign]
```

Create the sign using the `roadrunner.hdmap.Sign` object. Specify the sign information for the sign id.

```
rrMap.Signs = roadrunner.hdmap.Sign(ID="Sign1")

rrMap =
  roadrunnerHDMMap with properties:

      Author: ""
      Projection: ""
      GeographicBoundary: [0×3 double]
      Lanes: [0×1 roadrunner.hdmap.Lane]
      LaneBoundaries: [0×1 roadrunner.hdmap.LaneBoundary]
      LaneGroups: [0×1 roadrunner.hdmap.LaneGroup]
      LaneMarkings: [0×1 roadrunner.hdmap.LaneMarking]
      Junctions: [0×1 roadrunner.hdmap.Junction]
      BarrierTypes: [0×1 roadrunner.hdmap.BarrierType]
      Barriers: [0×1 roadrunner.hdmap.Barrier]
      SignTypes: [0×1 roadrunner.hdmap.SignType]
      Signs: [1×1 roadrunner.hdmap.Sign]
```

Version History

Introduced in R2022b

See Also

`roadrunnerHDMMap` | `roadrunner.hdmap.SignType`

Topics

“Build Simple Roads Programmatically Using RoadRunner HD Map”

“Build Scenes from Custom Data Using RoadRunner HD Map”

roadrunner.hdmap.SignType

Create sign types in RoadRunner HD Map using MATLAB

Description

A `roadrunner.hdmap.SignType` object enables you to define the sign features and the type information for a sign in a RoadRunner HD Map scene model.

Creation

Syntax

```
aSignType = roadrunner.hdmap.SignType()  
aSignType = roadrunner.hdmap.SignType(Name=Value)
```

Description

`aSignType = roadrunner.hdmap.SignType()` creates an empty sign type.

`aSignType = roadrunner.hdmap.SignType(Name=Value)` sets the properties of the sign type using name-value pairs.

Properties

ID — ID of sign type

character vector | string scalar

ID of the sign type, specified as a character vector or string scalar. The `Sign` object refers to this field to describe the type of a sign.

Data Types: `char` | `string`

AssetPath — Relative path for image of sign or sign asset file

`roadrunner.hdmap.RelativeAssetPath` object

Relative path for an image of the sign or a sign asset file, specified as a `roadrunner.hdmap.RelativeAssetPath` object. The asset path points to the information about the sign type.

Examples

Add Sign Type to RoadRunner HD Map

Create an empty RoadRunner HD Map by calling the `roadrunnerHDMMap` object.

```
rrMap = roadrunnerHDMMap()
```

```

rrMap =
  roadrunnerHDMAP with properties:
    Author: ""
    GeoReference: [0 0]
    GeographicBoundary: [0x3 double]
    Lanes: [0x1 roadrunner.hdmap.Lane]
    LaneBoundaries: [0x1 roadrunner.hdmap.LaneBoundary]
    LaneGroups: [0x1 roadrunner.hdmap.LaneGroup]
    LaneMarkings: [0x1 roadrunner.hdmap.LaneMarking]
    Junctions: [0x1 roadrunner.hdmap.Junction]
    BarrierTypes: [0x1 roadrunner.hdmap.BarrierType]
    Barriers: [0x1 roadrunner.hdmap.Barrier]
    SignTypes: [0x1 roadrunner.hdmap.SignType]
    Signs: [0x1 roadrunner.hdmap.Sign]
    StaticObjectTypes: [0x1 roadrunner.hdmap.StaticObjectType]
    StaticObjects: [0x1 roadrunner.hdmap.StaticObject]

```

Create a relative path to an asset. In this example, create a relative path to a parking sign asset file from the RoadRunner Asset Library. This path is relative to the Assets folder of your RoadRunner project.

```
path = roadrunner.hdmap.RelativeAssetPath(AssetPath="/Assets/Signs/UK/Sign_Parking.svg")
```

```

path =
  RelativeAssetPath with properties:
    AssetPath: "/Assets/Signs/UK/Sign_Parking.svg"

```

Create the sign type using the `roadrunner.hdmap.SignType` object. Specify the sign type information for the sign type id and the asset path.

```
rrMap.SignTypes = roadrunner.hdmap.SignType(ID="SignType1", AssetPath=path)
```

```

rrMap =
  roadrunnerHDMAP with properties:
    Author: ""
    GeoReference: [0 0]
    GeographicBoundary: [0x3 double]
    Lanes: [0x1 roadrunner.hdmap.Lane]
    LaneBoundaries: [0x1 roadrunner.hdmap.LaneBoundary]
    LaneGroups: [0x1 roadrunner.hdmap.LaneGroup]
    LaneMarkings: [0x1 roadrunner.hdmap.LaneMarking]
    Junctions: [0x1 roadrunner.hdmap.Junction]
    BarrierTypes: [0x1 roadrunner.hdmap.BarrierType]
    Barriers: [0x1 roadrunner.hdmap.Barrier]
    SignTypes: [1x1 roadrunner.hdmap.SignType]
    Signs: [0x1 roadrunner.hdmap.Sign]
    StaticObjectTypes: [0x1 roadrunner.hdmap.StaticObjectType]
    StaticObjects: [0x1 roadrunner.hdmap.StaticObject]

```

Version History

Introduced in R2022b

See Also

roadrunnerHDMAP | roadrunner.hdmap.Sign

Topics

“Build Simple Roads Programmatically Using RoadRunner HD Map”

“Build Scenes from Custom Data Using RoadRunner HD Map”

roadrunner.hdmap.StaticObject

Create static objects for RoadRunner HD Map using MATLAB

Description

A `roadrunner.hdmap.StaticObject` object enables you to define the static object instances in a RoadRunner HD Map scene model. This object is used for props and road furniture that do not move.

Creation

Syntax

```
aStaticObject = roadrunner.hdmap.StaticObject()
aStaticObject = roadrunner.hdmap.StaticObject(Name=Value)
```

Description

`aStaticObject = roadrunner.hdmap.StaticObject()` creates an empty `StaticObject`.

`aStaticObject = roadrunner.hdmap.StaticObject(Name=Value)` sets the properties of the `StaticObject` using name-value pairs.

Properties

ID — ID of StaticObjectelement

character vector | string scalar

ID of the `StaticObject` element, specified as a character vector or string scalar.

Example: `rrMap.StaticObjects = roadrunner.hdmap.StaticObject(ID="StaticObject1")` creates a `StaticObject` with id `StaticObject1` in a RoadRunner HD Map.

Data Types: `char` | `string`

Geometry — Geometry of the StaticObject

`GeoOrientedBoundingBox`

Geometry of the `StaticObject`, specified as a `GeoOrientedBoundingBox`. Static objects fit within an oriented bounding box. Objects should be scaled to fit the bound and then rotated. See `GeoOrientation3` for how to handle rotations of objects.

Data Types: `vector`

ObjectTypeReference — Type of StaticObject

`roadrunner.hdmap.Reference` object

Type of `StaticObject`, specified as a `roadrunner.hdmap.Reference` object.

Examples

Add Static Object to RoadRunner HD Map

Create an empty RoadRunner HD Map by calling the `roadrunnerHDMAP` object.

```
rrMap = roadrunnerHDMAP()
```

```
rrMap =
```

```
roadrunnerHDMAP with properties:
```

```
    Author: ""
    Projection: ""
    GeographicBoundary: [0×3 double]
        Lanes: [0×1 roadrunner.hdmap.Lane]
    LaneBoundaries: [0×1 roadrunner.hdmap.LaneBoundary]
    LaneGroups: [0×1 roadrunner.hdmap.LaneGroup]
    LaneMarkings: [0×1 roadrunner.hdmap.LaneMarking]
    Junctions: [0×1 roadrunner.hdmap.Junction]
    BarrierTypes: [0×1 roadrunner.hdmap.BarrierType]
    Barriers: [0×1 roadrunner.hdmap.Barrier]
    SignTypes: [0×1 roadrunner.hdmap.SignType]
    Signs: [0×1 roadrunner.hdmap.Sign]
    StaticObjectTypes: [0×1 roadrunner.hdmap.StaticObjectType]
    StaticObjects: [0×1 roadrunner.hdmap.StaticObject]
```

Create a static object using the `roadrunner.hdmap.StaticObject` object. Specify the static object information for the object id.

```
rrMap.StaticObjects = roadrunner.hdmap.StaticObject(ID="StaticObject1")
```

```
rrMap =
```

```
roadrunnerHDMAP with properties:
```

```
    Author: ""
    Projection: ""
    GeographicBoundary: [0×3 double]
        Lanes: [0×1 roadrunner.hdmap.Lane]
    LaneBoundaries: [0×1 roadrunner.hdmap.LaneBoundary]
    LaneGroups: [0×1 roadrunner.hdmap.LaneGroup]
    LaneMarkings: [0×1 roadrunner.hdmap.LaneMarking]
    Junctions: [0×1 roadrunner.hdmap.Junction]
    BarrierTypes: [0×1 roadrunner.hdmap.BarrierType]
    Barriers: [0×1 roadrunner.hdmap.Barrier]
    SignTypes: [0×1 roadrunner.hdmap.SignType]
    Signs: [0×1 roadrunner.hdmap.Sign]
    StaticObjectTypes: [0×1 roadrunner.hdmap.StaticObjectType]
    StaticObjects: [1×1 roadrunner.hdmap.StaticObject]
```

Version History

Introduced in R2022b

See Also

roadrunnerHDMAP | roadrunner.hdmap.StaticObjectType

Topics

“Build Simple Roads Programmatically Using RoadRunner HD Map”

“Build Scenes from Custom Data Using RoadRunner HD Map”

readCRS

Read coordinate reference system (CRS) data from RoadRunner HD map using MATLAB

Syntax

```
crs = readCRS(rrMap)
```

Description

`crs = readCRS(rrMap)` reads the coordinate reference system data from the RoadRunner HD map `rrMap`.

This function requires Mapping Toolbox™.

Examples

Read CRS Data from RoadRunner HD Map

Create a RoadRunner HD map, represented by a `roadrunnerHDMap` object. Specify an author and spatial bounds for the geometric data attributes of the map.

```
rrMap = roadrunnerHDMap(Author="Map Author",GeographicBoundary=[-0.782 -3.13 0; 101.565 50 0]);
```

Read the CRS data from the RoadRunner HD map file using the `readCRS` function and generate a `projcrs` object.

```
crs = readCRS(rrMap);  
disp(crs)
```

```
projcrs with properties:
```

```
          Name: "WGS 84 / Transverse Mercator"  
    GeographicCRS: [1x1 geocrs]  
    ProjectionMethod: "Transverse Mercator"  
          LengthUnit: "meter"  
    ProjectionParameters: [1x1 map.crs.ProjectionParameters]
```

Input Arguments

rrMap — RoadRunner HD Map road data model

`roadrunnerHDMap` object

RoadRunner HD Map road data model, specified as a `roadrunnerHDMap` object. `rrMap` defines a simple data structure to represent road layouts using lanes, lane boundaries, lane markings, and junctions. This object provides functions that support reading, writing, and plotting HD map data.

Output Arguments

crs — Coordinate reference system

geocrs object | projcrs object

Coordinate reference system (CRS), returned as one of these objects:

- `geocrs` — Returned by a file that contains geographic CRS data.
- `projcrs` — Returned by a file that contains projected CRS data.

Version History

Introduced in R2023a

See Also

Objects

roadrunnerHDMap | roadrunner

Functions

read | plot | write

Topics

“Build Simple Roads Programmatically Using RoadRunner HD Map”

roadrunner.hdmap.StaticObjectType

Create static object types in RoadRunner HD Map using MATLAB

Description

A `roadrunner.hdmap.StaticObject` object enables you to define the static object and the type information for a static object in a RoadRunner HD Map scene model.

Creation

Syntax

```
aStaticObjectType = roadrunner.hdmap.StaticObjectType()  
aStaticObjectType = roadrunner.hdmap.StaticObjectType(Name=Value)
```

Description

`aStaticObjectType = roadrunner.hdmap.StaticObjectType()` creates an empty `StaticObject` type.

`aStaticObjectType = roadrunner.hdmap.StaticObjectType(Name=Value)` sets the properties of the `StaticObject` type using name-value pairs.

Properties

ID — ID of StaticObject type

character vector | string scalar

ID of the `StaticObject` type, specified as a character vector or string scalar. The `StaticObject` object refers to this field to describe the type of a static object.

Data Types: `char` | `string`

AssetPath — Relative path for 3D mesh or prop asset file

`roadrunner.hdmap.RelativeAssetPath` object

Relative path for 3D mesh or prop asset file, specified as a `roadrunner.hdmap.RelativeAssetPath` object. The asset path points to the information about the static object type.

Examples

Add Static Object Type to RoadRunner HD Map

Create an empty RoadRunner HD Map by calling the `roadrunnerHDMAP` object.

```
rrMap = roadrunnerHDMAP()
```

```

rrMap =
  roadrunnerHDMAP with properties:
    Author: ""
    Projection: ""
    GeographicBoundary: [0x3 double]
    Lanes: [0x1 roadrunner.hdmap.Lane]
    LaneBoundaries: [0x1 roadrunner.hdmap.LaneBoundary]
    LaneGroups: [0x1 roadrunner.hdmap.LaneGroup]
    LaneMarkings: [0x1 roadrunner.hdmap.LaneMarking]
    Junctions: [0x1 roadrunner.hdmap.Junction]
    BarrierTypes: [0x1 roadrunner.hdmap.BarrierType]
    Barriers: [0x1 roadrunner.hdmap.Barrier]
    SignTypes: [0x1 roadrunner.hdmap.SignType]
    Signs: [0x1 roadrunner.hdmap.Sign]
    StaticObjectTypes: [0x1 roadrunner.hdmap.StaticObjectType]
    StaticObjects: [0x1 roadrunner.hdmap.StaticObject]

```

Create a relative path to an asset. In this example, we create a relative path to a prop asset file from the RoadRunner Asset Library. This path is relative to the `Assets` folder of your RoadRunner project.

```
path = roadrunner.hdmap.RelativeAssetPath(AssetPath="/Asset/Props/Trees/Eucalyptus_Sm01.fbx")
```

```

path =
  RelativeAssetPath with properties:
    AssetPath: "/Asset/Props/Trees/Eucalyptus_Sm01.fbx"

```

Create the static object type using the `roadrunner.hdmap.StaticObjectType` object. Specify the static object type information for the static object type id and the asset path.

```
rrMap.StaticObjectTypes = roadrunner.hdmap.StaticObjectType(ID="StaticObjectType1", AssetPath=path)
```

```

rrMap =
  roadrunnerHDMAP with properties:
    Author: ""
    Projection: ""
    GeographicBoundary: [0x3 double]
    Lanes: [0x1 roadrunner.hdmap.Lane]
    LaneBoundaries: [0x1 roadrunner.hdmap.LaneBoundary]
    LaneGroups: [0x1 roadrunner.hdmap.LaneGroup]
    LaneMarkings: [0x1 roadrunner.hdmap.LaneMarking]
    Junctions: [0x1 roadrunner.hdmap.Junction]
    BarrierTypes: [0x1 roadrunner.hdmap.BarrierType]
    Barriers: [0x1 roadrunner.hdmap.Barrier]
    SignTypes: [0x1 roadrunner.hdmap.SignType]
    Signs: [0x1 roadrunner.hdmap.Sign]
    StaticObjectTypes: [1x1 roadrunner.hdmap.StaticObjectType]
    StaticObjects: [0x1 roadrunner.hdmap.StaticObject]

```

Version History

Introduced in R2022b

See Also

roadrunnerHDMAP | roadrunner.hdmap.StaticObject

Topics

“Build Simple Roads Programmatically Using RoadRunner HD Map”

“Build Scenes from Custom Data Using RoadRunner HD Map”

Protocol Buffers

roadrunner_service.proto

RoadRunner service protobuf schema

Description

The `roadrunner_service.proto` protocol buffer (protobuf) file defines the schema for the RoadRunner service methods used to programmatically control RoadRunner.

Using a protobuf compiler, you can compile this file and other RoadRunner protobuf files into a language supported by gRPC and generate RoadRunner API code in that language. For more details, see “Compile Protocol Buffers for RoadRunner gRPC API”. For background information on how the RoadRunner API works, see “Control RoadRunner Programmatically Using gRPC API”.

Protobuf File Location	<i>RRInstallFolder/bin/platform/Proto/mathworks/roadrunner</i>
Protobuf Syntax	proto3
Package	mathworks.roadrunner

Services

RoadRunnerService — RoadRunner service methods
service of remote procedure call methods

RoadRunner service methods, specified as a service of remote procedure call (RPC) methods. Use these methods to send message requests to RoadRunner to perform operations programmatically. The `roadrunner_service_messages.proto` protobuf file defines the schema for these messages.

This table describes the RPC methods you can use.

NewProject	Create new RoadRunner project
LoadProject	Load RoadRunner project
SaveProject	Save RoadRunner project
NewScene	Create new RoadRunner scene
LoadScene	Load RoadRunner scene
SaveScene	Save RoadRunner scene
NewScenario	Create new RoadRunner scenario
LoadScenario	Load RoadRunner scenario
SaveScenario	Save RoadRunner scenario
SetScenarioVariable	Set RoadRunner scenario variable
getAllScenarioVariables	Retrieve all RoadRunner scenario variables
PrepareSimulation	Prepare RoadRunner simulation for scenario simulation engine
SimulateScenario	Simulate RoadRunner scenario

RemapAnchor	Remap RoadRunner scenario anchors
Export	Export RoadRunner scene
Import	Import file into RoadRunner scene
Exit	Exit RoadRunner application

Version History

Introduced in R2021b

See Also

[roadrunner_service_messages.proto](#) | [import_settings.proto](#) | [export_settings.proto](#)

Topics

“Control RoadRunner Programmatically Using gRPC API”

“Compile Protocol Buffers for RoadRunner gRPC API”

roadrunner_service_messages.proto

RoadRunner service messages protobuf schema

Description

The `roadrunner_service_messages.proto` protocol buffer (protobuf) file defines the schema for messages that you can send using the RoadRunner service methods. These methods are defined in the `roadrunner_service.proto` protobuf file.

Using a protobuf compiler, you can compile this file and other RoadRunner protobuf files into a language supported by gRPC and generate RoadRunner API code in that language. For more details, see “Compile Protocol Buffers for RoadRunner gRPC API”. For background information on how the RoadRunner API works, see “Control RoadRunner Programmatically Using gRPC API”.

Protobuf File Location	<i>RRInstallFolder/bin/platform/Proto/mathworks/roadrunner</i>
Protocol Buffer Syntax	proto3
Package	mathworks.roadrunner

Messages

NewProjectRequest — New project request message

New project request, specified as a message with these fields.

Name	Type	Description
folder_path (required)	string	Path at which to create the new project. RoadRunner creates the folder structure recursively. If the folder already exists, then RoadRunner returns an error.
asset_libraries (optional)	repeated string	Asset libraries to include in the new project. The only valid entry is "RoadRunner_Asset_Library", which includes the RoadRunner Asset Library assets in the project. If you do not specify <code>asset_libraries</code> , then RoadRunner includes only the assets that come installed with RoadRunner projects by default.

NewProjectResponse — New project response empty message

New project response, returned as an empty message.

LoadProjectRequest — Load project request
message

Load project request, specified as a message with this field.

Name	Type	Description
folder_path (required)	string	Path to existing project folder to load.

LoadProjectResponse — Load project response
empty message

Load project response, returned as an empty message.

SaveProjectRequest — Save project request
empty message

Save project request, specified as an empty message.

SaveProjectResponse — Save project response
empty message

Save project response, returned as an empty message.

NewSceneRequest — New scene request
empty message

New scene request, specified as an empty message.

NewSceneResponse — New scene response
empty message

New scene response, returned as an empty message.

LoadSceneRequest — Load scene request
message

Load scene request, specified as a message with this field.

Name	Type	Description
file_path (required)	string	Absolute or relative path to the scene file to load. If you specify a relative path, then the path is relative to the Scenes folder of the current project. The filename in file_path must either end with the .rrscene extension or have no extension. If it has no extension, then RoadRunner appends the .rrscene extension to file_path before loading the scene.

LoadSceneResponse — Load scene response
empty message

Load scene response, returned as an empty message.

SaveSceneRequest — Save scene request
message

Save scene request, specified as a message with this field.

Name	Type	Description
file_path (optional)	string	<p>Absolute or relative path to the scene file to save. If you specify a relative path, then the path is relative to the Scenes folder of the current project.</p> <p>If you do not specify <code>file_path</code>, then RoadRunner saves the current scene. If you do not have a current scene loaded, then RoadRunner returns an error.</p> <p>If any folders in the file path are missing, then RoadRunner creates them recursively.</p> <p>The filename in <code>file_path</code> must either end with the <code>.rrscene</code> extension or have no extension. If it has no extension, then RoadRunner appends the <code>.rrscene</code> extension to <code>file_path</code> before saving the scene. If the file being saved already exists, then RoadRunner overwrites it.</p>

SaveSceneResponse — Save scene response
empty message

Save scene response, returned as an empty message.

SetVariableRequest — Set variable request
message

Set variable request, specified as a message with these fields.

Name	Type	Description
name (required)	string	Name of the variable to set.
value (required)	string	New value to assign to the variable. Even if the variable value is numeric, you must specify <code>value</code> as a string.

SetVariableResponse — Set variable response
empty message

Set variable response, returned as an empty message.

GetAllVariablesRequest — Get all variables request
empty message

Get all variables request, returned as an empty message.

GetAllVariablesResponse — Get all variables response message

Get all variables response, returned as a message with these fields.

Name	Type	Description
variables	repeated Variable	Names and values of all variables in the scenario. For more information on the Variable message, see the Variable section of the roadrunner_service_messages.proto page.

Variable — Scenario variables message

Variables in the scenario, returned as a message with these fields.

Name	Type	Description
name	string	Name of the variable.
value	string	Value of the variable.

NewScenarioRequest — New scenario request empty message

New scenario request, specified as an empty message.

NewScenarioResponse — New scenario response empty message

New scenario response, returned as an empty message.

LoadScenarioRequest — Load scenario request message

Load scenario request, specified as a message with these fields

Name	Type	Description
file_path (required)	string	<p>Absolute or relative path to the scenario file to load. If you specify a relative path, then the path is relative to the Scenarios folder of the current project.</p> <p>The filename specified for file_path must either end with the .rrscenario extension or have no extension. If the input has no extension, then RoadRunner appends the .rrscenario extension to the specified value before loading the scenario.</p>
keep_current_scene (optional)	bool	<p>Option to load the scenario into the current scene. If you set keep_current_scene to true and there is no current scene, the LoadScenario method call fails.</p> <p>If you set keep_current_scene to false, then RoadRunner loads the scene that the scenario was previously saved with and loads the scenario into that scene.</p> <p>Default: false</p>

LoadScenarioResponse — Load scenario response
empty message

Load scenario response, returned as an empty message.

SaveScenarioRequest — Save scenario request
message

Save scenario request, specified as a message with this field.

Name	Type	Description
file_path (optional)	string	<p>Absolute or relative path to the scenario file to save. If you specify a relative path, then the path is relative to the Scenarios folder of the current project.</p> <p>RoadRunner saves the scenario to the path specified by file_path. If you do not specify file_path, then RoadRunner saves the current scenario. If you do not have a current scenario loaded or have not previously saved the current scenario, then RoadRunner returns an error.</p> <p>RoadRunner recursively creates any folders missing from the file path.</p> <p>The filename specified for file_path must either end with the .rrscenario extension or have no extension. If the input has no extension, then RoadRunner appends the .rrscenario extension to the specified value before loading the scenario.</p>

SaveScenarioResponse — Save scenario response
empty message

Save scenario response, returned as an empty message.

PrepareSimulationRequest — Prepare simulation request
empty message

Prepare simulation request, specified as an empty message.

PrepareSimulationResponse — Prepare simulation response
empty message

Prepare simulation response, returned as an empty message.

SimulateScenarioRequest — Simulate scenario request
message

Simulate scenario request, specified as a message with these fields.

Name	Type	Description
padding (optional)	google.protobuf.DoubleValue	<p>Simulation padding to control how fast the simulation runs, specified as a nonnegative number.</p> <ul style="list-style-type: none"> • A value between 0 and 1 is slower than real time. • A value of 1 equates to real time. • A value greater than 1 is faster than real time. <p>If you omit this value, then the simulation runs as fast as possible, given the performance of the CPU and the complexity of the scenario.</p>
simulation_end_time (optional)	google.protobuf.DoubleValue	<p>Maximum amount of time, in seconds, that the simulation runs. The simulation ends when it reaches <code>simulation_end_time</code>, if it does not stop earlier. If you omit this value, then the simulation runs until it stops for another reason, such as if an end condition is met, a collision occurs, or you manually stop the simulation.</p>
blocking (optional)	google.protobuf.BoolValue	<p>Option to block calls to the RoadRunner application during simulation. If <code>true</code>, then calls to RoadRunner are blocked until the simulation ends. If <code>false</code>, you can make calls to the RoadRunner application immediately after starting the simulation.</p> <p>Default: <code>true</code></p>

SimulateScenarioResponse — Simulate scenario response
empty message

Simulate scenario response, returned as an empty message.

ExportRequest — Export request
message

Export request, specified as a message with these fields.

Name	Type	Description
file_path (required)	string	<p>Absolute or relative path to the exported file. If you specify a relative path, then the exported file is saved relative to the Exports folder of the current project.</p> <p>If any folders in the path are missing, RoadRunner tries to create them.</p> <p>file_path must include the extension of the exported file.</p>

Name	Type	Description																					
format_name (required)	string	<p>Export format name corresponding to a valid RoadRunner export format. Format name options are case-insensitive.</p> <table border="1" data-bbox="1062 478 1458 1837"> <thead> <tr> <th data-bbox="1062 478 1192 583">format_name Option</th> <th data-bbox="1192 478 1321 583">Description</th> <th data-bbox="1321 478 1458 583">Export Format Details</th> </tr> </thead> <tbody> <tr> <td data-bbox="1062 583 1192 758">"AutoCAD"</td> <td data-bbox="1192 583 1321 758">Export scene to an AutoCAD file.</td> <td data-bbox="1321 583 1458 758">"Export to AutoCAD"</td> </tr> <tr> <td data-bbox="1062 758 1192 932">"Filmbox"</td> <td data-bbox="1192 758 1321 932">Export scene to a Filmbox file.</td> <td data-bbox="1321 758 1458 932">"Export to FBX"</td> </tr> <tr> <td data-bbox="1062 932 1192 1192">"glTF"</td> <td data-bbox="1192 932 1321 1192">Export scene to a GL Transmission Format (glTF) file.</td> <td data-bbox="1321 932 1458 1192">"Export to glTF"</td> </tr> <tr> <td data-bbox="1062 1192 1192 1367">"OpenFlight"</td> <td data-bbox="1192 1192 1321 1367">Export scene to an OpenFlight file.</td> <td data-bbox="1321 1192 1458 1367">"Export to OpenFlight"</td> </tr> <tr> <td data-bbox="1062 1367 1192 1570">"OpenSceneGraph"</td> <td data-bbox="1192 1367 1321 1570">Export scene to an OpenSceneGraph file.</td> <td data-bbox="1321 1367 1458 1570">"Export to OpenSceneGraph"</td> </tr> <tr> <td data-bbox="1062 1570 1192 1837">"USD"</td> <td data-bbox="1192 1570 1321 1837">Export scene to a Universal Scene Description (USD) file.</td> <td data-bbox="1321 1570 1458 1837">"Export to USD"</td> </tr> </tbody> </table>	format_name Option	Description	Export Format Details	"AutoCAD"	Export scene to an AutoCAD file.	"Export to AutoCAD"	"Filmbox"	Export scene to a Filmbox file.	"Export to FBX"	"glTF"	Export scene to a GL Transmission Format (glTF) file.	"Export to glTF"	"OpenFlight"	Export scene to an OpenFlight file.	"Export to OpenFlight"	"OpenSceneGraph"	Export scene to an OpenSceneGraph file.	"Export to OpenSceneGraph"	"USD"	Export scene to a Universal Scene Description (USD) file.	"Export to USD"
format_name Option	Description	Export Format Details																					
"AutoCAD"	Export scene to an AutoCAD file.	"Export to AutoCAD"																					
"Filmbox"	Export scene to a Filmbox file.	"Export to FBX"																					
"glTF"	Export scene to a GL Transmission Format (glTF) file.	"Export to glTF"																					
"OpenFlight"	Export scene to an OpenFlight file.	"Export to OpenFlight"																					
"OpenSceneGraph"	Export scene to an OpenSceneGraph file.	"Export to OpenSceneGraph"																					
"USD"	Export scene to a Universal Scene Description (USD) file.	"Export to USD"																					

Name	Type	Description		
		format_ name Option	Description	Export Format Details
		"Apollo"	Export scene to the Baidu Apollo file format.	"Export to Apollo"
		"GeoJSON"	Export scene to a GeoJSON file.	"Export to GeoJSON"
		"OpenDRIVE"	Export scene to an ASAM OpenDRIVE file.	"Export to ASAM OpenDRIVE"
		"OpenScenario"	Export scenario to an ASAM OpenSCENARIO file.	"Export to ASAM OpenSCENARIO" (RoadRunner Scenario)
		"OpenScenario 2.0"	Export scenario to an ASAM OpenSCENARIO 2.0 file.	"Export to ASAM OpenSCENARIO" (RoadRunner Scenario)
		"CARLA"	Export scene to the CARLA format.	"Export to CARLA"
		"Metamoto"	Export scene to the Metamoto format.	"Export to Metamoto"
		"Unity"	Export scene to the Unity format.	"Export to Unity"

Name	Type	Description		
		format_ name Option	Descript ion	Export Format Details
		"Unreal"	Export scene to the Unreal Engine format.	"Export to Unreal Using Filmbox (.fbx) File"
		"VTD"	Export scene to the VTD format.	"Export to VTD"
		You can also specify custom exporters that you define in the <code>ExportConfigurations.xml</code> file. For more details, see "Export Custom Formats".		

Name	Type	Description																								
export_settings (optional)	oneof	Export settings configuration, specified as one of the export settings messages defined in the export_settings.proto file. If you specify export_settings, then the export settings type must be compatible with the format name specified in the format_name field.																								
		<table border="1"> <thead> <tr> <th>export_settings Field</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>auto_cad_settings</td> <td>AutoCadExportSettings message</td> <td>AutoCAD export settings</td> </tr> <tr> <td>filmbox_settings</td> <td>FilmboxExportSettings message</td> <td>Filmbox export settings</td> </tr> <tr> <td>gltf_settings</td> <td>GltfExportSettings message</td> <td>glTF export settings</td> </tr> <tr> <td>open_flight_settings</td> <td>OpenFlightExportSettings message</td> <td>OpenFlight export settings</td> </tr> <tr> <td>open_scene_graph_settings</td> <td>OpenSceneGraphExportSettings message</td> <td>OpenSceneGraph export settings</td> </tr> <tr> <td>usd_settings</td> <td>UsdExportSettings message</td> <td>Universal Scene Description export settings</td> </tr> <tr> <td>apollo_settings</td> <td>ApolloExportSettings message</td> <td>Apollo export settings</td> </tr> </tbody> </table>	export_settings Field	Type	Description	auto_cad_settings	AutoCadExportSettings message	AutoCAD export settings	filmbox_settings	FilmboxExportSettings message	Filmbox export settings	gltf_settings	GltfExportSettings message	glTF export settings	open_flight_settings	OpenFlightExportSettings message	OpenFlight export settings	open_scene_graph_settings	OpenSceneGraphExportSettings message	OpenSceneGraph export settings	usd_settings	UsdExportSettings message	Universal Scene Description export settings	apollo_settings	ApolloExportSettings message	Apollo export settings
export_settings Field	Type	Description																								
auto_cad_settings	AutoCadExportSettings message	AutoCAD export settings																								
filmbox_settings	FilmboxExportSettings message	Filmbox export settings																								
gltf_settings	GltfExportSettings message	glTF export settings																								
open_flight_settings	OpenFlightExportSettings message	OpenFlight export settings																								
open_scene_graph_settings	OpenSceneGraphExportSettings message	OpenSceneGraph export settings																								
usd_settings	UsdExportSettings message	Universal Scene Description export settings																								
apollo_settings	ApolloExportSettings message	Apollo export settings																								

Name	Type	Description		
		export_settings Field	Type	Description
		geo_json_settings	GeoJsonExportSettings message	GeoJSON export settings
		open_drive_settings	OpenDriveExportSettings message	ASAM OpenDRIVE export settings
		open_scenario_settings	OpenScenarioExportSettings message	ASAM OpenSCENARIO export settings
		carla_settings	CarlaExportSettings message	CARLA export settings
		metamoto_settings	MetamotoExportSettings message	Metamoto export settings
		unity_settings	UnityExportSettings message	Unity export settings
		unreal_settings	UnrealExportSettings message	Unreal Engine export settings
		vtd_settings	VtdExportSettings message	VTD export settings

ExportResponse — Export response
empty message

Export response, returned as an empty message.

ImportRequest — Import request
message

Import request, specified as a message with these fields.

Name	Type	Description
file_path (required)	string	Absolute or relative path to the file to import. If you specify a relative path, then you must specify a path to a file in the <code>Assets</code> folder of the current project.
open_drive_settings (optional)	OpenDRIVEImportSettings message	ASAM OpenDRIVE import settings.
csv_trajectory_settings (optional)	CsvTrajectoryImportSettings message	CSV trajectory import settings.
roadrunner_hd_map_settings (optional)	RoadRunnerHdMapImportSettings message	RoadRunner HD map import settings.

ImportResponse — Import response
empty message

Import response, returned as an empty message.

ExitRequest — Exit request
empty message

Exit request, specified as an empty message.

ExitResponse — Exit response
empty message

Exit response, returned as an empty message.

RemapAnchorRequest — Remap anchor request
message

Remap anchor request, specified as a message with these fields.

Name	Type	Description
source_anchor (required)	string	Name of the source anchor from the scenario. If empty or not found in the scenario, the operation fails.

Name	Type	Description									
remap_type (required)	oneof	<p>Type of anchor remap to execute. Remap to an existing anchor by specifying an anchor name, or create a new anchor to remap to by specifying a 3D position.</p> <table border="1"> <thead> <tr> <th>remap_type Field</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>name_remap</td> <td>AnchorRemapToName Message</td> <td>Name of target anchor to remap to.</td> </tr> <tr> <td>position_remap</td> <td>AnchorRemapToPosition Message</td> <td>Position relative to the center of the RoadRunner scene to create a new anchor to remap to.</td> </tr> </tbody> </table> <p>If empty, the operation fails.</p> <p>For more information on the <code>AnchorRemapToName</code> and <code>AnchorRemapToPosition</code> messages, see the <code>AnchorRemapToName</code> and <code>AnchorRemapToPosition</code> sections of the <code>roadrunner_service_messages.proto</code> page.</p>	remap_type Field	Type	Description	name_remap	AnchorRemapToName Message	Name of target anchor to remap to.	position_remap	AnchorRemapToPosition Message	Position relative to the center of the RoadRunner scene to create a new anchor to remap to.
remap_type Field	Type	Description									
name_remap	AnchorRemapToName Message	Name of target anchor to remap to.									
position_remap	AnchorRemapToPosition Message	Position relative to the center of the RoadRunner scene to create a new anchor to remap to.									

RemapAnchorResponse — Remap anchor response
empty message

Remap anchor response, returned as an empty message.

AnchorRemapToName — Remap anchor to target name
message

Name of the target anchor to remap to, specified as a message with these fields.

Name	Type	Description
target_anchor (required)	string	Name of target anchor to remap to. If not found in the scenario, the operation fails.

AnchorRemapToPosition — Remap anchor to position message

Position to remap to, specified as a message with these fields.

Name	Type	Description
position (required)	scenario.common.Vector3	Position relative to the center of the RoadRunner scene at which to create a new anchor to remap to. If the distance between this position and the closest road in the scene is greater than 25 meters, the operation fails.
new_anchor_name (optional)	string	Name of the new anchor. If left empty, RoadRunner assigns a default anchor name.

Version History

Introduced in R2021b

R2022b: Import HD Map data programmatically using gRPC

Import HD map data into RoadRunner programmatically using the `RoadRunnerHdMapImportSettings` setting in the gRPC `Import` request. You need a RoadRunner license to load HD maps and a RoadRunner Scene Builder license to both load and build HD maps into editable RoadRunner scenes.

See Also

`roadrunner_service.proto` | `import_settings.proto` | `export_settings.proto`

Topics

“Control RoadRunner Programmatically Using gRPC API”

“Compile Protocol Buffers for RoadRunner gRPC API”

import_settings.proto

Import settings protobuf schema

Description

The `import_settings.proto` protocol buffer (protobuf) file defines the schema for RoadRunner import settings. These settings correspond to options in the RoadRunner import dialog boxes for the supported file formats.

Using a protobuf compiler, you can compile this file and other RoadRunner protobuf files into a language supported by gRPC and generate RoadRunner API code in that language. For more details, see “Compile Protocol Buffers for RoadRunner gRPC API”. For background information on how the RoadRunner API works, see “Control RoadRunner Programmatically Using gRPC API”.

Protobuf File Location	<i>RRInstallFolder/bin/platform/Proto/mathworks/roadrunner</i>
Protobuf Syntax	proto3
Package	mathworks.roadrunner

Enumerations

ProjectionMode — Projection mode enumeration

Projection mode, specified as an enumeration containing these options.

Name	Value	Description
PROJECTION_MODE_UNSPECIFIED	0	No projection mode specified
PROJECTION_MODE_FULL_PROJECTION	1	Full projection
PROJECTION_MODE_TRANSLATE_ONLY	2	Translate-only projection
PROJECTION_MODE_NO_PROJECTION	3	No projection mode

MedianLaneType — Median lane type enumeration

Median lane type, specified as an enumeration containing these options.

Name	Value	Description
MEDIAN_LANE_TYPE_UNSPECIFIED	0	No median lane type specified
MEDIAN_LANE_TYPE_MEDIAN	1	Median lane type

Name	Value	Description
MEDIAN_LANE_TYPE_RAISED_MEDIAN	2	Raised median lane type

These fields correspond to the Lane Options in the Import ASAM OpenDRIVE dialog box. For more details, see “Importing ASAM OpenDRIVE Files”.

ImportStep — Import step type enumeration

Import step type, specified as an enumeration containing these options.

Name	Value	Description
IMPORT_STEP_UNSPECIFIED	0	No import step type specified
IMPORT_STEP_LOAD	1	Load import step type
IMPORT_STEP_BUILD	2	Build import step type

Messages

OpenDriveImportSettings — ASAM OpenDRIVE import settings message

ASAM OpenDRIVE import settings, specified as a message containing these fields.

Name	Type	Description
import_signals	google.protobuf.BoolValue	Map all <signal> entries in file to signals or signs.
import_props	google.protobuf.BoolValue	Map all <object> entries in file to props or markings.
import_hoffset_relative_to_orientation	google.protobuf.BoolValue	Import the <hOffset> (heading offset) values of <signal> entries as being relative to <orientation>, which is the direction of travel of the road that the signal applies to. By default, the heading offset is relative to the heading of the road, regardless of its direction of travel.
lane_options	LaneOptions message	Lane import options.
offset	scenario.common.Vector3 message	Offset of the imported ASAM OpenDRIVE scene, relative to the center of the RoadRunner scene.

Name	Type	Description
projection	scenario.common.Projection message	Projection of the imported ASAM OpenDRIVE scene. <ul style="list-style-type: none"> If the file does not have projection information, then RoadRunner uses the projection of the scene. If both the scene and file do not have projection information, then RoadRunner uses the Transverse Mercator projection centered at 0 degrees latitude and longitude.
projection_mode	ProjectionMode message	Projection mode.

These fields correspond to options in the Import ASAM OpenDRIVE dialog box. For more details on these options, see “Importing ASAM OpenDRIVE Files”.

LaneOptions — Lane options

message

Lane options, specified as a message containing these fields.

Name	Type	Description
curb_lane_markings_to_curb_lanes	google.protobuf.BoolValue	Convert curb lane markings to curb lanes. Lanes with the "curb" marking and type "shoulder" are imported as type "curb" if this option is enabled. Otherwise, lanes are imported as type "shoulder".
convert_lane_heights	google.protobuf.BoolValue	Map all <height> entries to imported lanes.
median_lane_type	MedianLaneType enumeration	Map all <lane> types as "median" or "raised median".

These fields correspond to options in the Import ASAM OpenDRIVE dialog box. For more details, see “Importing ASAM OpenDRIVE Files”.

CsvTrajectoryImportSettings — CSV trajectory import settings

message

CSV trajectory import settings, specified as a message containing these fields.

Name	Type	Description
actor_attributes	ActorAttributes message	Optional attributes to apply to the actor.
spawn_time	google.protobuf.DoubleValue	Time to delay (in seconds) before adding the actor to the scenario.
remove_time	google.protobuf.DoubleValue	Time to delay (in seconds) before removing the actor from the scenario.

These fields correspond to attributes you can specify when programmatically importing CSV files. For more details, see “Import CSV Files Programmatically” (RoadRunner Scenario).

ActorAttributes — Actor Attributes message

Actor attributes, specified as a message containing these fields. When you specify actor attributes, follow the same conventions as when specifying variables through the RoadRunner UI or API.

Name	Type	Description
name	string	Name to be given to the new actor.
id	string	Actor ID to be given to the new actor.
color	string	Color to be given to the new actor.
asset_path	string	Asset path for the actor in the RoadRunner asset library.
behavior_asset_path	string	Behavior asset path in the RoadRunner asset library.

These fields correspond to attributes you can specify when programmatically importing CSV files. For more details, see “Import CSV Files Programmatically” (RoadRunner Scenario).

RoadRunnerHdMapImportSettings — RoadRunner HD map import settings message

RoadRunner HD map import settings, specified as a message containing these fields.

Name	Type	Description
import_step	ImportStep enumeration	Specifies the import step to be performed. If this parameter is unset, then both load and build are performed.
load_settings	RoadRunnerHdMapLoadSettings message	Settings for loading the HD map data into RoadRunner.

Name	Type	Description
build_settings	RoadRunnerHdMapBuildSettings message	Settings for building the loaded HD map in RoadRunner. This requires RoadRunner Scene Builder license.

These fields correspond to options in the Scene Builder dialog box. For more details on these options, see “Import HD Map File into RoadRunner”.

RoadRunnerHdMapLoadSettings — RoadRunner HD map load settings message

RoadRunner HD map load settings, specified as a message containing these fields.

Name	Type	Description
offset	scenario.common.Vector3 message	Offset of the imported HD map, relative to the center of the RoadRunner scene.
projection	scenario.common.Projection message	Projection of the imported HD map. <ul style="list-style-type: none"> • If the projection is not set, then RoadRunner uses the projection of the file. • If the file does not have projection information, then RoadRunner uses the projection of the scene. • If both the scene and file do not have projection information, then RoadRunner uses the Transverse Mercator projection centered at 0 degrees latitude and longitude.

These fields correspond to options in the Scene Builder dialog box. For more details on these options, see “Import HD Map File into RoadRunner”.

RoadRunnerHdMapBuildSettings — RoadRunner HD map build settings message

RoadRunner HD map build settings, specified as a message containing these fields.

Name	Type	Description
fit_cross_sections	google.protobuf.BoolValue	Import flat cross-sections of the HD map. By default, the Scene Builder Tool imports cross-section information such as super elevation (banking) and crowning. To import flat cross-sections, clear this option. For more information about cross-sections, see the Cross Section Tool .
detect_asphalt_surfaces	google.protobuf.BoolValue	Detect asphalt surfaces and apply an asphalt texture to them.
clear_scene_of_existing_data	google.protobuf.BoolValue	Removes all the data from the scene.
curvature_blend	google.protobuf.DoubleValue	Specifies the curvature blend along the road.
auto_detect_bridges	AutoDetectBridges message	Auto detect bridge attributes as applied to road intersections in the imported HD map.

These fields correspond to options in the Scene Builder dialog box. For more details on these options, see “Build Scenes Using HERE HD Live Map Data”.

AutoDetectBridges — Auto detect bridges settings message

Auto detect bridges settings, specified as a message containing these fields.

Name	Type	Description
bridge_span_inflation	google.protobuf.DoubleValue	Amount of extension (in meters) of the bridge on either side of the intersection in the imported HD map. By default, the Scene Builder Tool creates bridges at road intersections when the roads have different elevations. The tool extends the bridges by 20 meters on either side of the intersection. You can change the amount of extension by changing the <code>bridge_span_inflation</code> value.

These fields correspond to options in the Scene Builder dialog box. For more details on these options, see “Build Scenes Using HERE HD Live Map Data”.

Version History

Introduced in R2021b

R2022b: Import HD Map data programmatically using gRPC

Import HD map data into RoadRunner programmatically using the `RoadRunnerHdMapImportSettings` setting in the gRPC `Import` request. You need a RoadRunner license to load HD maps and a RoadRunner Scene Builder license to both load and build HD maps into editable RoadRunner scenes.

See Also

[roadrunner_service.proto](#) | [roadrunner_service_messages.proto](#) | [export_settings.proto](#) | [Import](#) | [Export](#)

Topics

[“Control RoadRunner Programmatically Using gRPC API”](#)

[“Compile Protocol Buffers for RoadRunner gRPC API”](#)

export_settings.proto

Export settings protobuf schema

Description

The `export_settings.proto` protocol buffer (protobuf) file defines the schema for RoadRunner export settings. These settings generally correspond to options in the RoadRunner export dialog boxes.

Using a protobuf compiler, you can compile this file and other RoadRunner protobuf files into a language supported by gRPC and generate RoadRunner API code in that language. For more details, see “Compile Protocol Buffers for RoadRunner gRPC API”. For background information on how the RoadRunner API works, see “Control RoadRunner Programmatically Using gRPC API”.

Protobuf File Location	<i>RRInstallFolder/bin/platform/Proto/mathworks/roadrunner</i>
Protobuf Syntax	proto3
Package	mathworks.roadrunner

Enumerations

CrgRoadDataFormat — Data format of roads exported to ASAM OpenCRG enumeration

Data format of roads exported to ASAM OpenCRG, specified as an enumeration containing these options.

Name	Value	Description
CRG_ROAD_DATA_FORMAT_UNSPECIFIED	0	No road data format specified
CRG_ROAD_DATA_FORMAT_LRFI	1	Long, real, formatted, interchangeable data format specified
CRG_ROAD_DATA_FORMAT_LDFI	2	Long, double, formatted, interchangeable data format specified

These options correspond to the **Road Data Format** options in the Export OpenCRG Options dialog box, which you can access from **Export**, then **ASAM OpenDRIVE**. For more details on these options, see “Export to ASAM OpenCRG” and “Export to ASAM OpenDRIVE”.

DrivingSide — Driving side enumeration

Driving side of the roads exported to ASAM OpenDRIVE, specified as an enumeration containing these options.

Name	Value	Description
DRIVING_SIDE_UNSPECIFIED	0	No driving side specified
DRIVING_SIDE_LEFT	1	Left-hand driving side specified
DRIVING_SIDE_RIGHT	2	Right-hand driving side specified

These options correspond to the **Left Side** and **Right Side** options in the Export ASAM OpenDRIVE dialog box, which you can access from the **File** menu, under **Export**, then **ASAM OpenDRIVE**. For more details on these options, see “Export to ASAM OpenDRIVE” and “Left-Hand Drive Export to ASAM OpenDRIVE”.

Note DRIVING_SIDE_UNSPECIFIED is the default value for the **Driving Side**. If you change the driving side in the RoadRunner application, then DRIVING_SIDE_LEFT is the default value.

Messages

AutoCadExportSettings — AutoCAD export settings
message

AutoCAD export settings, specified as a message containing these fields.

Name	Type	Description
split_by_segmentation	google.protobuf.BoolValue	Split meshes by their segmentation type. For more details, see “Segmentation”.
power_of_two_textures	google.protobuf.BoolValue	Resize the dimensions of exported textures by rounding them up to the next highest power of two.
export_only_highest_lods	google.protobuf.BoolValue	Export only the highest LODs for all props in the scene.
export_to_tiles	ExportToTiles message	Split meshes per tile.

These fields correspond to options in the Export AutoCAD dialog box, which you can access from the **File** menu, under **Export**, then **AutoCad**. For more details on these options, see “Export to AutoCAD”.

FilmbboxExportSettings — FBX export settings
message

FBX export settings, specified as a message containing these fields.

Name	Type	Description
split_by_segmentation	google.protobuf.BoolValue	Split meshes by their segmentation type. For more details, see “Segmentation”.

Name	Type	Description
power_of_two_textures	google.protobuf.BoolValue	Resize the dimensions of exported textures by rounding them up to the next highest power of two.
embed_textures	google.protobuf.BoolValue	Embed the exported textures inside the exported file.
export_only_highest_lods	google.protobuf.BoolValue	Export only the highest LODs for all props in the scene.
export_to_tiles	ExportToTiles message	Split meshes per tile.

These fields correspond to options in the Export Filmbox dialog box, which you can access from the **File** menu, under **Export**, then **Filmbox**. For more details on these options, see “Export to FBX”.

GlTFExportSettings — glTF export settings message

glTF export settings, specified as a message containing these fields.

Name	Type	Description
split_by_segmentation	google.protobuf.BoolValue	Split meshes by their segmentation type. For more details, see “Segmentation”.
power_of_two_textures	google.protobuf.BoolValue	Resize the dimensions of exported textures by rounding them up to the next highest power of two.
export_only_highest_lods	google.protobuf.BoolValue	Export only the highest LODs for all props in the scene.
export_to_tiles	ExportToTiles message	Split meshes per tile.

These fields correspond to options in the Export glTF dialog box, which you can access from the **File** menu, under **Export**, then **glTF**. For more details on these options, see “Export to glTF”.

OpenFlightExportSettings — OpenFlight export settings message

OpenFlight export settings, specified as a message containing these fields.

Name	Type	Description
split_by_segmentation	google.protobuf.BoolValue	Split meshes by their segmentation type. For more details, see “Segmentation”.
power_of_two_textures	google.protobuf.BoolValue	Resize the dimensions of exported textures by rounding them up to the next highest power of two.

Name	Type	Description
export_only_highest_lods	google.protobuf.BoolValue	Export only the highest LODs for all props in the scene.
export_to_tiles	ExportToTiles message	Split meshes per tile.

These fields correspond to options in the Export OpenFlight dialog box, which you can access from the **File** menu, under **Export**, then **OpenFlight**. For more details on these options, see “Export to OpenFlight”.

OpenSceneGraphExportSettings — OpenSceneGraph export settings message

OpenSceneGraph export settings, specified as a message containing these fields.

Name	Type	Description
split_by_segmentation	google.protobuf.BoolValue	Split meshes by their segmentation type. For more details, see “Segmentation”.
power_of_two_textures	google.protobuf.BoolValue	Resize the dimensions of exported textures by rounding them up to the next highest power of two.
embed_textures	google.protobuf.BoolValue	Embed the exported textures inside the exported file
export_only_highest_lods	google.protobuf.BoolValue	Export only the highest LODs for all props in the scene.
export_to_tiles	ExportToTiles message	Split meshes per tile.

These fields correspond to options in the Export OpenSceneGraph dialog box, which you can access from the **File** menu, under **Export**, then **OpenSceneGraph**. For more details on these options, see “Export to OpenSceneGraph”.

UnrealDatasmithExportSettings — Unreal Datasmith export settings message

Unreal Datasmith export settings, specified as a message containing these fields.

Name	Type	Description
split_by_segmentation	google.protobuf.BoolValue	Split meshes by their segmentation type. For more details, see “Segmentation”.
power_of_two_textures	google.protobuf.BoolValue	Resize the dimensions of exported textures by rounding them up to the next highest power of two.
export_only_highest_lods	google.protobuf.BoolValue	Export only the highest LODs for all props in the scene.
export_to_tiles	ExportToTiles message	Split meshes per tile.

These fields correspond to options in the Export Unreal Datasmith dialog box, which you can access from the **File** menu, under **Export**, then **Unreal Datasmith**. For more details on these options, see “Export to Unreal Using Datasmith (.udatasmith) File”.

UsdExportSettings — USD export settings message

Universal Scene Description (USD) export settings, specified as a message containing these fields.

Name	Type	Description
split_by_segmentation	google.protobuf.BoolValue	Split meshes by their segmentation type. For more details, see “Segmentation”.
power_of_two_textures	google.protobuf.BoolValue	Resize the dimensions of exported textures by rounding them up to the next highest power of two.
export_only_highest_lods	google.protobuf.BoolValue	Export only the highest LODs for all props in the scene.
export_to_tiles	ExportToTiles message	Split meshes per tile.

These fields correspond to options in the Export USD dialog box, which you can access from the **File** menu, under **Export**, then **USD**. For more details on these options, see “Export to USD”.

ApolloExportSettings — Apollo export settings message

Apollo export settings, specified as a message containing these fields.

Name	Type	Description
apollo_version	double	The version of Apollo to export. Valid versions are 3.0 (default) and 5.0.
database_version	double	An identifier for the exported scene, which is useful for versioning exports of the same scene. For more details, see “Export Custom Formats”.
database_name	string	The name of the exported scene. For more details, see “Export Custom Formats”.
driving_side	DrivingSide enumeration	The driving side of the exported scene.
export_signals	google.protobuf.BoolValue	Export all signals and signs mapped to junctions as <signal> entries. For more details, see “Export Custom Formats”.

Name	Type	Description
export_objects	google.protobuf.BoolValue	Export all props as <object> entries. For more details, see “Export Custom Formats”.
clamp_distances	google.protobuf.BoolValue	Clamp distances in the RoadRunner scene to multiples of 1 cm. This prevents exporting very short roads. Note RoadRunner clamps the scene itself, which can cause small changes to the roads in the scene.

These fields correspond to options in the Export Apollo dialog box, which you can access from the **File** menu, under **Export**, then **Apollo**. For more details on these options, see “Export to Apollo”.

GeoJsonExportSettings — GeoJSON export settings message

GeoJSON export settings, specified as a message containing this field.

Name	Type	Description
reduce_file_size	google.protobuf.BoolValue	Removes new lines from the exported GeoJSON file and decreases its size. For more details, see “Export Options”.

This field corresponds to an option in the Export GeoJSON dialog box, which you can access from the **File** menu, under **Export**, then **GeoJSON**. For more details on this option, see “Export to GeoJSON”.

OpenDriveExportSettings — ASAM OpenDRIVE export settings message

ASAM OpenDRIVE export settings, specified as a message containing these fields.

Name	Type	Description
open_drive_version	double	The version of ASAM OpenDRIVE to export. Valid versions are 1.4 (default), 1.5, or 1.6.
database_version	double	An identifier for the exported scene, which is useful for versioning exports of the same scene.
database_name	string	The name of the exported scene.
driving_side	DrivingSide enumeration	The driving side of the exported scene.

Name	Type	Description
export_markings_as_line	google.protobuf.BoolValue	Export additional lane marking data (spacing, dash length, and individual paint strip widths) using the <line> definition in ASAM OpenDRIVE.
export_signals	google.protobuf.BoolValue	Export all signals and signs mapped to junctions as <signal> entries.
export_objects	google.protobuf.BoolValue	Export all props as <object> entries.
export_hoffset_relative_to_orientation	google.protobuf.BoolValue	Export the <hOffset> (heading offset) values of <signal> entries as being relative to <orientation>, which is the direction of travel of the road that the signal applies to.
export_conflict_points	google.protobuf.BoolValue	Export an <object> entry for every point in a junction where two roads intersect.
export_scene_origin_reference	google.protobuf.BoolValue	Export a reference point origin at (0,0) in the scene.
clamp_distances	google.protobuf.BoolValue	Clamp distances in the RoadRunner scene to prevent exporting very short roads. Note RoadRunner clamps the scene itself, which can cause small changes to the roads in the scene.
synthetic_open_crg	SyntheticOpenCrg	Export road surface data assigned to different road segments to the ASAM OpenCRG file format. For more details, see “Export to ASAM OpenCRG”.

Name	Type	Description
enforce_connected_road_continuity	double	<p>Specifies the road data format for the output ASAM OpenCRG file. The two data formats supported are:</p> <ul style="list-style-type: none"> • LRFI (default) — Long, real, formatted, interchangeable data format • LDFI — Long, double, formatted, interchangeable data format <p>Dependencies</p> <p>To enable this attribute, <code>export_synthetic_open_crg</code> and <code>export_open_crg</code> options must be selected.</p>

These fields correspond to options in the Export ASAM OpenDRIVE dialog box, which you can access from the **File** menu, under **Export**, then **ASAM OpenDRIVE**. For more details on these options, see “Export to ASAM OpenDRIVE”.

OpenScenarioExportSettings — ASAM OpenSCENARIO export settings message

ASAM OpenSCENARIO export settings, specified as a message containing these fields.

Name	Type	Description
open_scenario_version	double	The version of ASAM OpenSCENARIO to export. Valid versions are 1.0, 1.1, or 2.0.

Name	Type	Description
new_or_existing_scene_graph (optional)	oneof	<p>Generate a new scene graph file during export or reuse an existing scene graph. If you do not specify this field, then RoadRunner exports the scene associated with the scenario as an OpenSceneGraph file.</p> <p>Specify <code>new_or_existing_scene_graph</code> as one of these fields:</p> <ul style="list-style-type: none"> • <code>scene_graph_format_name</code> — Format name of the scene graph file exported with the scenario, specified as a string. Only "OpenSceneGraph" is supported. Currently, setting this field to "OpenSceneGraph" is equivalent to leaving <code>new_or_existing_scene_graph</code> unset. • <code>path_to_existing_scene_graph</code> — Path to an existing scene graph file used with the exported scenario, specified as a string. If you are exporting scenarios that all use the same scene, specify this option to reuse the same scene graph file after the first export, which can reduce export time significantly. For an example that shows how to reuse a scene graph file, see "Generate Scenario Variations Using gRPC API" (RoadRunner Scenario).
open_scene_graph_settings (optional)	OpenSceneGraphExportsSettings message	OpenSceneGraph export settings. For more details, see "Export Options (OpenSceneGraph)" and "Export to OpenSceneGraph".
open_drive_settings (optional)	OpenDriveExportSettings message	Options in the ASAM OpenDRIVE tab to export an ASAM OpenDRIVE file. For more details, see "Export to ASAM OpenDRIVE".

These fields correspond to options in the Export ASAM OpenSCENARIO dialog box, which you can access from the **File** menu, under **Export**, then **ASAM OpenSCENARIO 1.x**. For more details on these options, see "Export to ASAM OpenSCENARIO" (RoadRunner Scenario).

OpenScenario2ExportSettings — ASAM OpenSCENARIO 2.0 export settings message

ASAM OpenSCENARIO 2.0 export settings, specified as a message containing these fields.

Name	Type	Description
open_drive_settings (optional)	OpenDriveExportSettings message	Options in the ASAM OpenDRIVE tab to export an ASAM OpenDRIVE file. For more details, see “Export to ASAM OpenDRIVE”.

These fields correspond to options in the Export ASAM OpenSCENARIO 2.0 dialog box, which you can access from the **File** menu, under **Export**, then **ASAM OpenSCENARIO 2.0**. For more details on these options, see “Export to ASAM OpenSCENARIO” (RoadRunner Scenario).

CarlaExportSettings — CARLA export settings message

CARLA export settings, specified as a message containing these fields.

Name	Type	Description
unreal_datasmith_settings	UnrealDatasmithExportSettings message	Options in Unreal tab to export a Unreal Datasmith file. For more details, see “CARLA Export Overview”.
open_drive_settings	OpenDriveExportSettings message	Options in the ASAM OpenDRIVE tab to export an ASAM OpenDRIVE file. For more details, see “CARLA Export Overview” and “Export to ASAM OpenDRIVE”.

These fields correspond to options in the Export CARLA dialog box, which you can access from the **File** menu, under **Export**, then **CARLA**. For more details on these options, see “Export to CARLA”. The **CARLAExportSettings** are compatible with the **CARLA** export option that exports Unreal Datasmith and ASAM OpenDRIVE files. If you try to use the Filmbox option, RoadRunner would crash immediately.

CarlaFilmboxExportSettings — CARLA Filmbox export settings message

CARLA Filmbox export settings, specified as a message containing these fields.

Name	Type	Description
open_drive_settings	OpenDriveExportSettings message	Options in the ASAM OpenDRIVE tab to export an ASAM OpenDRIVE file. For more details, see “CARLA Export Overview” and “Export to ASAM OpenDRIVE”.

Name	Type	Description
filmbox_settings	FilmboxExportSettings message	Options in the Filmbox tab to export a Filmbox file. For more details, see “CARLA Export Overview” and “Export to FBX”.

These fields correspond to options in the Export CARLA Filmbox dialog box, which you can access from the **File** menu, under **Export**, then **CARLA Filmbox**. For more details on these options, see “Export to CARLA”.

MetamotoExportSettings — Metamoto export settings message

Metamoto export settings, specified as a message containing these fields.

Name	Type	Description
filmbox_settings	FilmboxExportSettings message	Options in the Filmbox tab to export a Filmbox file. For more details, see “Export to FBX”.
open_drive_settings	OpenDriveExportSettings message	Options in the ASAM OpenDRIVE tab to export an ASAM OpenDRIVE file. For more details, see “Export to ASAM OpenDRIVE”.
geo_json_settings	GeoJsonExportSettings message	Options in the GeoJSON tab to export a GeoJSON file. For more details, see “Export to GeoJSON”.

These fields correspond to options in the Export Metamoto dialog box, which you can access from the **File** menu, under **Export**, then **Metamoto**. For more details on these options, see “Export to Metamoto”.

UnityExportSettings — Unity export settings message

Unity export settings, specified as a message containing these fields.

Name	Type	Description
filmbox_settings	FilmboxExportSettings message	Options in the Filmbox tab to export an Filmbox file. For more details, see “Export to FBX”.
open_drive_settings	OpenDriveExportSettings message	Options in the ASAM OpenDRIVE tab to export a ASAM OpenDRIVE file. For more details, see “Export to ASAM OpenDRIVE”.

These fields correspond to options in the Export Unity dialog box, which you can access from the **File** menu, under **Export**, then **Unity**. For more details on these options, see “Export to Unity”.

UnrealExportSettings — Unreal export settings

message

Unreal export settings, specified as a message containing these fields.

Name	Type	Description
filmbox_settings	FilmboxExportSettings message	Options in the Filmbox tab to export a Filmbox file. For more details, see “Export to FBX”.
open_drive_settings	OpenDriveExportSettings message	Options in the ASAM OpenDRIVE tab to export an ASAM OpenDRIVE file. For more details, see “Export to ASAM OpenDRIVE”.

These fields correspond to options in the Export Unreal dialog box, which you can access from the **File** menu, under **Export**, then **Unreal**. For more details on these options, see “Export to Unreal Using Filmbox (.fbx) File”.

DatasmithRoadExportSettings — Datasmith road export settings

message

Datasmith road export settings, specified as a message containing these fields.

Name	Type	Description
unreal_datasmith_settings	UnrealDatasmithExportSettings message	Options in Unreal tab to export a Unreal Datasmith file. For more details, see “CARLA Export Overview”.
open_drive_settings	OpenDriveExportSettings message	Options in the ASAM OpenDRIVE tab to export an ASAM OpenDRIVE file. For more details, see “CARLA Export Overview” and “Export to ASAM OpenDRIVE”.

These fields correspond to options in the Export Datasmith Road dialog box, which you can access from the **File** menu, under **Export**, then **Datasmith**. For more details on these options, see “Export to Unreal Using Datasmith (.udatasmith) File”.

VtdExportSettings — VTD export settings

message

VTD export settings, specified as a message containing these fields.

Name	Type	Description
open_scene_graph_settings	OpenSceneGraphExportSettings message	Options in the OpenSceneGraph tab to export an OpenSceneGraph file. For more details, see “Export Options (OpenSceneGraph)” and “Export to OpenSceneGraph”.
open_drive_settings	OpenDriveExportSettings message	Options in the ASAM OpenDRIVE tab to export an ASAM OpenDRIVE file. For more details, see “Export Options (ASAM OpenDRIVE)” and “Export to ASAM OpenDRIVE”.

These fields correspond to options in the Export VTD dialog box, which you can access from the **File** menu, under **Export**, then **VTD**. For more details on these options, see “Export to VTD”.

ExportToTiles — Export to tiles settings message

Export to tiles settings, specified as a message containing these fields.

Name	Type	Description
tile_size	scenario.common.Vector2	Size of the exported tiles. For more details, see “Export to Tiles”.
tile_center	scenario.common.Vector2	Center location of the exported tiles. For more details, see “Export to Tiles”.
export_individual_tiles	google.protobuf.BoolValue	Export each tile as a separate file. For more details, see “Export to Tiles”.

These fields correspond to options in the AutoCAD, FBX, glTF, OpenFlight, OpenSceneGraph, and USD export dialog boxes. For more details on these options, see “Export to Tiles”.

SyntheticOpenCrg — Synthetic ASAM OpenCRG settings message

Synthetic ASAM OpenCRG settings, specified as a message containing these fields.

Name	Type	Description
road_data_format	CrgRoadDataFormat enumeration	The data format of the road in the exported scene.

Version History

Introduced in R2021b

R2022b: Export highest level of details

The gRPC `export_settings.proto` now supports the ability to export only the highest LODs programmatically for these exporters: AutoCAD, Filmbox, glTF, OpenFlight, OpenSceneGraph, USD, CARLA, Metamoto, Unity, Unreal, Datasmith, and VTD.

See Also

[roadrunner_service.proto](#) | [roadrunner_service_messages.proto](#) | [import_settings.proto](#) | [Import](#) | [Export](#)

Topics

[“Control RoadRunner Programmatically Using gRPC API”](#)
[“Compile Protocol Buffers for RoadRunner gRPC API”](#)

geometry.proto

Scene geometry protobuf schema

Description

The `geometry.proto` protocol buffer (protobuf) file defines the schema for RoadRunner scene geometry.

Using a protobuf compiler, you can compile this file and other RoadRunner protobuf files into a language supported by gRPC and generate RoadRunner API code in that language. For more details, see “Compile Protocol Buffers for RoadRunner gRPC API”. For background information on how the RoadRunner API works, see “Control RoadRunner Programmatically Using gRPC API”.

Protobuf File Location	<i>RRInstallFolder/bin/platform/Proto/mathworks/roadrunner/scenario/common</i>
Protobuf Syntax	proto3
Package	mathworks.scenario.common

Messages

Vector2 — Two-element vector
message

Two-element vector of double-precision values, specified as a message with these fields.

Name	Type	Description
x	double	First vector element
y	double	Second vector element

Vector3 — Three-element vector
message

Three-element vector of double-precision values, specified as a message with these fields.

Name	Type	Description
x	double	First vector element
y	double	Second vector element
z	double	Third vector element

Box3 — 3D axis-aligned box
message

3D axis-aligned box, specified as a message with these fields.

Name	Type	Description
min	Vector3 message	Minimum position of the box, which is the center of the bottom face of the box.
max	Vector3 message	Maximum position of the box, which is the center of the top face of the box.

Vector3List — List of three-element vectors
message

List of three-element vectors, specified as a message with this field.

Name	Type	Description
values	repeated Vector3 message	3D coordinates of points representing geometric objects.

Polygon — Geometry of polygon objects
message

Geometry of polygon objects, specified as a message with these fields. A polygon has one exterior boundary, but it can have multiple holes within it.

Name	Type	Description
exterior_ring	Vector3List message	Outer boundary of the polygon object
interior_rings	repeated Vector3List message	Boundaries of holes within the polygon

MultiPolygon — Set of non-overlapping polygons
message

Set of non-overlapping polygons, specified as a message with this field. The non-overlapping set can include polygons within holes of another polygon.

Name	Type	Description
polygons	repeated Polygon message	Geometry of non-overlapping polygon objects

Projection — Map projection
message

Map projection, specified as a message with this field.

Name	Type	Description
projection	string	Geospatial projection and datum used to represent spatial coordinates in the map. Valid WKT (including ESRI WKT) and PROJ.4 projection strings are supported. For more details on projection strings, see "Coordinate Space and Georeferencing".

Dimension3 — Size of 3D object message

Size of the 3D object, specified as a message with these fields. Each message that uses a Dimension3 message defines the units of its dimensions.

Name	Type	Description
length	double	Extent of the 3D object along the x-axis. You must specify a nonnegative value for the length.
width	double	Extent of the 3D object along the y-axis. You must specify a nonnegative value for the width.
height	double	Extent of the 3D object along the z-axis. You must specify a nonnegative value for the height.

GeoAngle3 — Angles of 3D rotation message

Angles of 3D rotation, specified as a message with these fields. A GeoAngle3 (roll, pitch, heading) of (0, 0, 0) faces the positive x-direction in the projection space of the input file, with left in the positive y-direction and up in the positive z-direction.

To convert these values into a transformation matrix, you must first convert the heading from the projection space of the file into the target projection space. Then, apply the rotations in this order: heading (about the z-axis), pitch (about the y'-axis), and roll (about the x''-axis). The y'-axis is the new y-axis that results after the first rotation, and the x''-axis is the new x-axis that results after the second rotation.

Name	Type	Description
roll	double	Clockwise rotation of the 3D object about the x''-axis in local Cartesian space. Units are in radians.

Name	Type	Description
pitch	double	Clockwise rotation of the 3D object about the y'-axis in local Cartesian space. Units are in radians.
heading	double	Clockwise rotation of the 3D object about the z-axis in projection space of input file. Units are in radians.

GeoOrientation3 — 3D orientation of object message

3D orientation of the object, specified as a message with this field.

Name	Type	Description
geo_angle	GeoAngle3 message	Angles of 3D rotation of the object.

GeoOrientedBoundingBox — Geometry of 3D stationary object message

Geometry of the 3D stationary object, specified as a message with these fields.

Name	Type	Description
center	scenario.common.Vector3 message	Coordinates of the center of the 3D object.
dimension	Dimension3 message	Dimensions for each local oriented axis of the bounding box in Cartesian space. Units are in meters. Each value represents the distance from the center of the box to the face of the box in the corresponding local axis. As such, each value is half of the total length of the box in the corresponding local axis. For example, if a box is 8 meters wide, the dimension for that axis is 4 meters.
orientation	GeoOrientation3 message	Orientation of the 3D object.

Version History

Introduced in R2021b

See Also

Topics

[“Control RoadRunner Programmatically Using gRPC API”](#)

[“Compile Protocol Buffers for RoadRunner gRPC API”](#)

[“Import Custom Data Using RoadRunner HD Map”](#)

hd_map_header.proto

RoadRunner HD Map header message

Description

hd_map_header.proto is a protocol buffer (protobuf) file that defines the schema for the header message of the RoadRunner HD Map scene model.

Protobuf File Location	<i>RRInstallFolder/bin/platform/Proto/mathworks/scenario/scene/hd</i>
Protobuf Syntax	proto3
Package	mathworks.scenario.scene.hdmap

Messages

Header — Header message
message

Header message of the RoadRunner HD Map, specified as a message with these fields.

Name	Type	Description
author	string	Name of the file author.
projection	scenario.common.Projection message	Map projection, specified as WKT (including ESRI WKT) or PROJ.4 projection strings. For more details, see “Coordinate Space and Georeferencing”.
geographic_boundary	DataBounds message	Spatial bounds of the geometric data represented in the 3D coordinate space of the RoadRunner HD Map projection.

DataBounds — Spatial bounds of geometric data
message

Spatial bounds of the geometric data stored in the RoadRunner HD Map, specified as a message with this field.

Name	Type	Description
bounds	scenario.common.Box3 message	Minimum and maximum position of the 3D, axis-aligned box.

Version History

Introduced in R2021b

See Also

hd_map.proto | hd_lanes.proto | hd_lane_markings.proto | hd_junctions.proto |
hd_barriers.proto | hd_signs.proto | hd_static_objects.proto |
common_attributes.proto | geometry.proto

Topics

“Compile Protocol Buffers for RoadRunner gRPC API”
“Build Scenes from Custom Data Using RoadRunner HD Map”

hd_map.proto

RoadRunner HD Map message

Description

hd_map.proto is a protocol buffer (protobuf) file that defines the schema for the RoadRunner HD Map scene model.

Protobuf File Location	<i>RRInstallFolder/bin/platform/Proto/mathworks/scenario/scene/hd</i>
Protobuf Syntax	proto3
Package	mathworks.scenario.scene.hdmap

Messages

HDMap — RoadRunner HD Map scene model message

RoadRunner HD Map scene model, specified as a message with these fields. You must define the lanes and lane_boundaries fields to create this scene model. Other fields are optional.

Name	Type	Description
lanes	repeated Lane message	Lane properties of a road
lane_boundaries	repeated LaneBoundary message	Boundaries of a lane
lane_groups	repeated LaneGroup message	Group of lanes with similar geometry
lane_markings	repeated LaneMarking message	Properties of lane markings
junctions	repeated Junction message	Properties of junctions
barrier_types	repeated BarrierTypeDefinition message	Type of barrier
barriers	repeated Barrier message	List of Barrier instances
sign_types	repeated SignTypeDefinition message	Type of sign
signs	repeated Sign message	List of Sign instances
static_object_types	repeated StaticObjectTypeDefinition message	Type of static object
static_objects	repeated StaticObject message	List of StaticObject instances

Version History

Introduced in R2021b

See Also

hd_lanes.proto | hd_lane_markings.proto | hd_junctions.proto | hd_barriers.proto |
hd_signs.proto | hd_static_objects.proto | hd_map_header.proto |
common_attributes.proto | geometry.proto

Topics

“Compile Protocol Buffers for RoadRunner gRPC API”

“Build Scenes from Custom Data Using RoadRunner HD Map”

hd_lanes.proto

Lane representation messages for the RoadRunner HD Map

Description

hd_lanes.proto is a protocol buffer (protobuf) file that defines the schema for representing lanes in the RoadRunner HD Map scene model.

Protobuf File Location	<i>RRInstallFolder/bin/platform/Proto/mathworks/scenario/scene/hd</i>
Protobuf Syntax	proto3
Package	mathworks.scenario.scene.hdmap

Enumerations

TravelDir — Direction of travel along lane
enumeration

Direction of travel along the lane, specified as an enumeration containing these options.

Name	Value	Description
TRAVEL_DIR_UNSPECIFIED	0	Unspecified travel direction
TRAVEL_DIR_UNDIRECTED	1	Undirected travel
TRAVEL_DIR_FORWARD	2	Forward travel direction
TRAVEL_DIR_BACKWARD	3	Backward travel direction
TRAVEL_DIR_BIDIRECTIONAL	4	Bidirectional travel direction

LaneType — Type of lane
enumeration

Type of lane, specified as an enumeration containing these options.

Name	Value	Description
LANE_TYPE_UNSPECIFIED	0	Unspecified type of lane
LANE_TYPE_DRIVING	1	Normal driving lanes
LANE_TYPE_SHOULDER	2	Lanes reserved for emergency stopping
LANE_TYPE_BORDER	3	Lanes at the road borders
LANE_TYPE_RESTRICTED	4	Lanes reserved for high occupancy vehicles
LANE_TYPE_PARKING	5	Lanes alongside driving lanes, intended for parking vehicles
LANE_TYPE_CURB	6	Curb at the edge of the road

Name	Value	Description
LANE_TYPE_SIDEWALK	7	Lanes reserved for pedestrians
LANE_TYPE_CENTER_TURN	8	Lanes in the middle of a two-way street, intended for making a turn
LANE_TYPE_BIKING	9	Lanes reserved for cyclists

Messages

Lane — Properties of lane message

Properties of the lane, specified as a message with these fields.

Name	Type	Description
id	string	ID of the lane
geometry	scenario.common.Vector3List message	3D coordinates of points representing the center line of the lane
travel_dir	TravelDir enumeration	Direction of travel along the lane
left_lane_boundary	AlignedReference message	Linkage information for the left boundary of the lane
right_lane_boundary	AlignedReference message	Linkage information for the right boundary of the lane
predecessors	repeated AlignedReference message	Linkage information for preceding lanes
successors	repeated AlignedReference message	Linkage information for succeeding lanes
lane_type	LaneType enumeration	Type of lane

ParametricAttribution — Span-based parametric attributes message

Span-based parametric attributes, specified as a message with these fields

Name	Type	Description
span	ParametricRange message	Range for span-based attributes

Name	Type	Description				
attributes	oneof	Span-based attributes, specified as a message with this field.				
		<table border="1"> <thead> <tr> <th>Attribute</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>marking_reference</td> <td>MarkingReference message</td> <td>Linking information for lane marking. You can specify ID, and offset attributes to place lane markings.</td> </tr> </tbody> </table>	Attribute	Type	Description	marking_reference
Attribute	Type	Description				
marking_reference	MarkingReference message	Linking information for lane marking. You can specify ID, and offset attributes to place lane markings.				

LaneBoundary — Boundaries of lane message

Boundaries of the lane, specified as a message with these fields.

Name	Type	Description
id	string	ID of the lane boundary
geometry	scenario.common.Vector3List message	3D coordinates of the lane boundary points
parametric_attributes	repeated ParametricAttribution	Span-based parametric attributes to create lane markings

LaneGroup — Group of lanes with similar geometry message

Group of lanes with similar geometry, specified as a message with these fields.

Name	Type	Description
id	string	ID of the lane group
geometry	scenario.common.Vector3List message	3D coordinates of the geometric shape of lane group
lanes	repeated AlignedReference message	Linkage information for lanes within the lane group

Version History

Introduced in R2021b

See Also

hd_map_header.proto | hd_map.proto | hd_lane_markings.proto | hd_junctions.proto |
hd_barriers.proto | hd_signs.proto | hd_static_objects.proto |
common_attributes.proto | geometry.proto

Topics

“Compile Protocol Buffers for RoadRunner gRPC API”
“Build Scenes from Custom Data Using RoadRunner HD Map”

hd_lane_markings.proto

Lane marking representation messages for RoadRunner HD Map

Description

hd_lane_markings.proto is a protocol buffer (protobuf) file that defines the schema for representing lane markings in the RoadRunner HD Map scene model.

Protobuf File Location	<i>RRInstallFolder/bin/platform/Proto/mathworks/scenario/scene/hd</i>
Protobuf Syntax	proto3
Package	mathworks.scenario.scene.hdmap

Messages

LaneMarking — Lane marking definitions

message

Lane marking definitions, specified as a message with these fields.

Name	Type	Description
id	string	ID of the lane marking element
asset_path	RelativeAssetPath message	Relative path for the lane marking asset

MarkingReference — Information for linking lane boundary and lane marking

message

Information for linking lane boundary and lane marking, specified as a message with these fields.

Name	Type	Description
marking_id	Reference message	ID of the lane marking element
flip_laterally	bool	Set this field to true to flip the order of the lane marking stripes.

Version History

Introduced in R2021b

See Also

hd_map_header.proto | hd_map.proto | hd_lanes.proto | hd_junctions.proto |
hd_barriers.proto | hd_signs.proto | hd_static_objects.proto |
common_attributes.proto | geometry.proto

Topics

“Compile Protocol Buffers for RoadRunner gRPC API”

“Build Scenes from Custom Data Using RoadRunner HD Map”

hd_junctions.proto

Junction representation for RoadRunner HD Map

Description

hd_junctions.proto is a protocol buffer (protobuf) file that defines the schema for representing junctions in the RoadRunner HD Map scene model.

Protobuf File Location	<i>RRInstallFolder/bin/platform/Proto/mathworks/scenario/scene/hd</i>
Protobuf Syntax	proto3
Package	mathworks.scenario.scene.hdmap

Messages

Junction — Road Junction
message

Road junction, specified as a message with these fields. Use this message to define the geometry and connectivity of lanes at a crossing between multiple roadways.

Name	Type	Description
id	string	Junction ID
geometry	scenario.common.MultiPolygon message	Geometry involving multiple non-overlapping polygons.
lanes	repeated Reference message	ID of the connecting lanes

Version History

Introduced in R2021b

See Also

hd_map_header.proto | hd_map.proto | hd_lanes.proto | hd_lane_markings.proto | hd_barriers.proto | hd_signs.proto | hd_static_objects.proto | geometry.proto

Topics

“Compile Protocol Buffers for RoadRunner gRPC API”

“Build Scenes from Custom Data Using RoadRunner HD Map”

common_attributes.proto

Common attributes for RoadRunner HD Map protocol buffer files

Description

`common_attributes.proto` is a protocol buffer (protobuf) file that defines the schema for common attributes of the RoadRunner HD Map scene model.

Protobuf File Location	<i>RRInstallFolder/bin/platform/Proto/mathworks/scenario/scene/hd</i>
Protobuf Syntax	proto3
Package	mathworks.scenario.scene.hdmap

Enumerations

Alignment — Type of alignment between linked objects
enumeration

Type of alignment between linked objects, specified as an enumeration containing these options. For more information, see “Alignment Types in RoadRunner HD Map” on page 5-59.

Name	Value	Description
ALIGNMENT_UNSPECIFIED	0	Alignment between linked objects is not specified.
ALIGNMENT_FORWARD	1	Specifies that the directions of linked objects are matching.
ALIGNMENT_BACKWARD	2	Specifies that the directions of linked objects are not matching.

Messages

ParametricRange — Range for span-based attributes
message

Range for span-based attributes, specified as a message with these fields. Using this message, you can insert span nodes that define a range along a curve-based parent object. You can change attributes along the specified range. For example, you can specify a range for which to change a lane marking along a lane boundary.

Name	Type	Description
span_start	double message	Normalized distance in the range [0, 1] between the start of the current span and the start of the parent object. Use the length of the parent object to get the normalized value.
span_end	double	Normalized distance in the range [0, 1] between the end of the current span and the start of the parent object. Use the length of the parent object to get the normalized value.

AlignedReference — Information for linking objects
message

Information for linking objects, specified as a message with these fields.

Name	Type	Description
reference	Reference message	ID of a referenced object
alignment	Alignment enumeration	Type of alignment between linked objects

Reference — ID of referenced object
message

ID of a referenced object, specified as a message with this field.

Name	Type	Description
id	string	ID of a referenced object

RelativeAssetPath — Asset file path
message

Asset file path relative to the project directory, specified as a message with this field.

Name	Type	Description
asset_path	string	Relative path for an asset in the asset library.

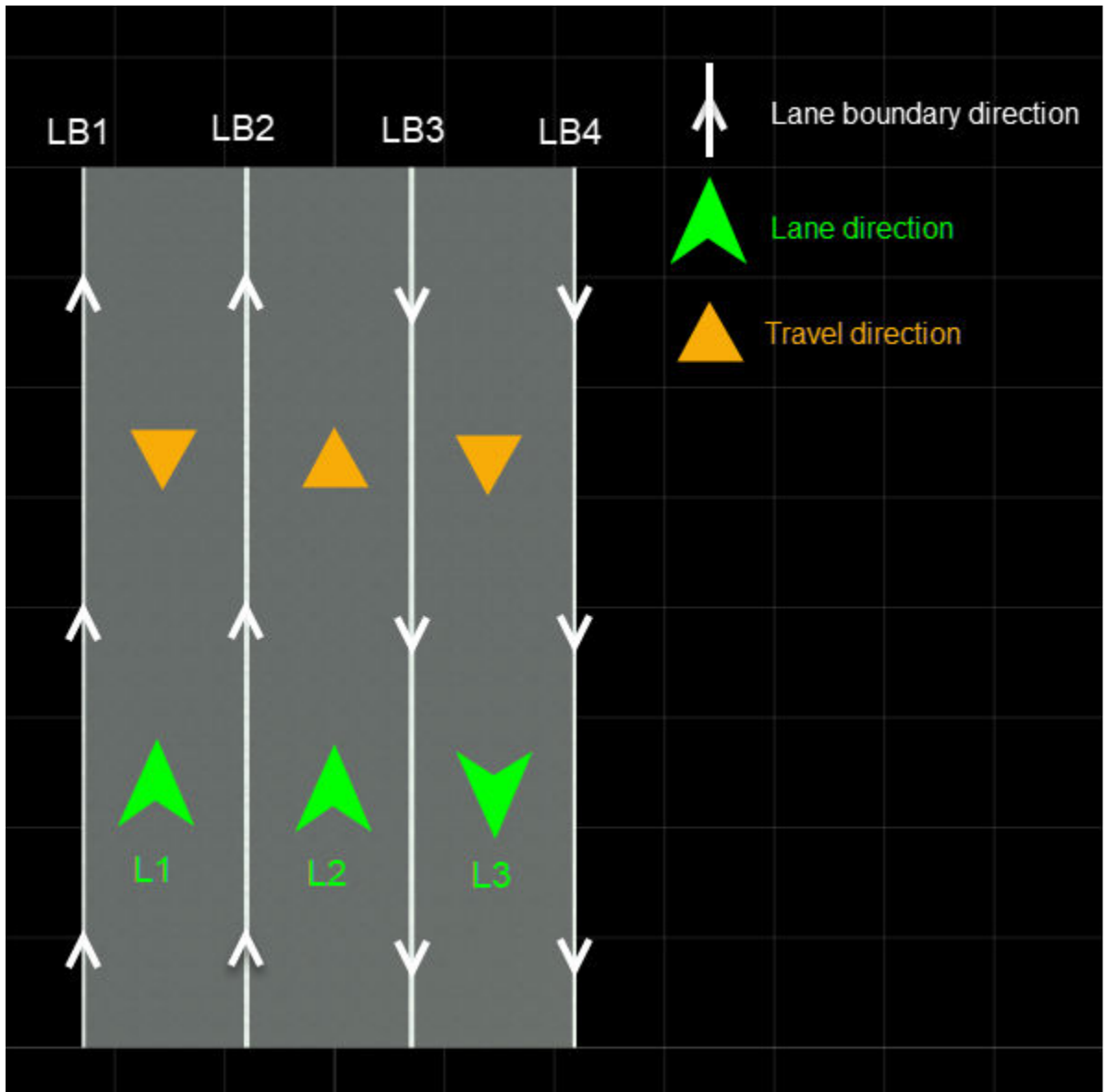
Example: "Assets/Markings/DashedSingleWhite.rrlms" sets the path for the white dashed lane marking asset.

More About

Alignment Types in RoadRunner HD Map

The `AlignedReference` message enables you to specify lane navigation. You can link a lane with lane boundaries and other lanes. As shown in this diagram, the geometry of a lane or lane boundary

is independent of other lanes or lane boundaries. You must specify the appropriate type of alignment when you link these objects.



To specify alignment between linked objects, follow these rules:

- Alignment must always be relative to the orientation of the object of interest.
- When the directions of two objects match, the alignment is forward. Otherwise, the alignment is backward.

This table shows alignment between the lanes and lane boundaries shown in the diagram. For completeness, the table also shows the travel direction for lanes.

Lane Number	Left Lane Boundary		Right Lane boundary		Travel Direction
	Number	Alignment	Number	Alignment	
L1	LB1	Forward	LB2	Forward	Backward
L2	LB2	Forward	LB3	Backward	Forward
L3	LB4	Forward	LB3	Forward	Forward

- Lane L1 is oriented from bottom-to-top. Since the bottom-to-top direction matches the directions of two lane boundaries (LB1 and LB2) their alignments are forward. However, the travel direction is backward as it runs from top-to-bottom.
- Lane L2 is oriented from bottom-to-top. As the left lane boundary (LB2), and travel direction of the lane also run from bottom-to-top, these objects have forward alignment. The right lane boundary (LB3) runs from top-to-bottom, so its alignment is backward.
- Lane L3 is oriented from top-to-bottom, which matches the directions of two lane boundaries and the travel direction of the lane. These objects have forward alignment. Note that, due to the top-to-bottom orientation of the lane, the left boundary for this lane is LB4, and the right boundary is LB3.

Version History

Introduced in R2021b

See Also

hd_map_header.proto | hd_map.proto | hd_lanes.proto | hd_lane_markings.proto |
hd_junctions.proto | hd_barriers.proto | hd_signs.proto | hd_static_objects.proto |
geometry.proto

Topics

“Compile Protocol Buffers for RoadRunner gRPC API”

“Build Scenes from Custom Data Using RoadRunner HD Map”

hd_barriers.proto

Barrier representation messages for RoadRunner HD Map

Description

The `hd_barriers.proto` protocol buffer (protobuf) file defines the schema for representing barriers in the RoadRunner HD Map scene model.

Protobuf File Location	<i>RRInstallFolder/bin/platform/Proto/mathworks/scenario/scene/hd</i>
Protobuf Syntax	proto3
Package	<code>mathworks.scenario.scene.hdmap</code>

Messages

BarrierTypeDefinition — Type of barrier message

Type of barrier, specified as a message with these fields.

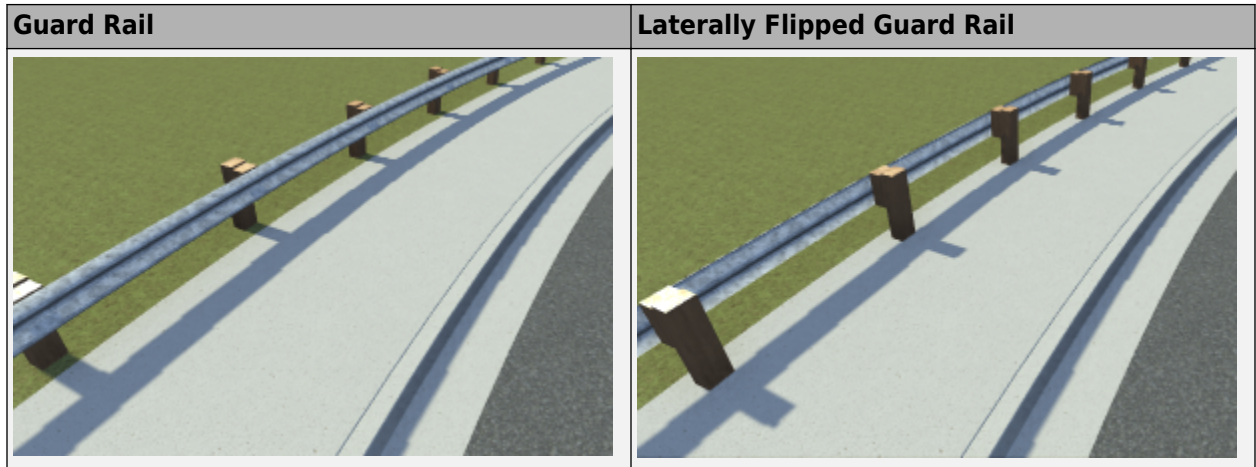
Name	Type	Description
<code>id</code>	string	ID of barrier type. The <code>Barrier</code> message refers to this field to describe a barrier based on its extruded geometry.
<code>extrusion_path</code>	RelativeAssetPath message	Relative path for the extrusion asset that describes the type of barrier.

Barrier — Properties of barrier message

Properties of the barrier, specified as a message with these fields.

Name	Type	Description
<code>id</code>	string	ID of the barrier element.
<code>geometry</code>	<code>scenario.common.Vector3List</code> message	3D coordinates of the barrier center points.
<code>barrier_type_ref</code>	Reference message	Type of barrier.

Name	Type	Description
flip_laterally	bool	Set this field to true to flip the orientation of barrier along the lateral direction. Use this field to flip the primary face of the extrusion geometry, such as the guard rails shown in these images.



Version History

Introduced in R2022a

See Also

hd_map_header.proto | hd_map.proto | hd_lanes.proto | hd_junctions.proto |
hd_signs.proto | hd_static_objects.proto | common_attributes.proto |
geometry.proto

Topics

“Compile Protocol Buffers for RoadRunner gRPC API”

“Build Scenes from Custom Data Using RoadRunner HD Map”

hd_signs.proto

Sign representation messages for RoadRunner HD Map

Description

The `hd_signs.proto` protocol buffer (protobuf) file defines the schema for representing signs in the RoadRunner HD Map scene model.

Protobuf File Location	<i>RRInstallFolder/bin/platform/Proto/mathworks/scenario/scene/hd</i>
Protobuf Syntax	proto3
Package	<code>mathworks.scenario.scene.hdmap</code>

Messages

SignTypeDefinition — Type of sign message

Type of sign, specified as a message with these fields.

Name	Type	Description
<code>id</code>	string	ID of sign type. The <code>Sign</code> message refers to this field to describe a type of sign.
<code>asset_path</code>	RelativeAssetPath message	Relative path for an image of the sign or a sign asset file.

Sign — Properties of sign message

Properties of the sign, specified as a message with these fields.

Name	Type	Description
<code>id</code>	string	ID of the sign element.
<code>geometry</code>	<code>scenario.common.GeoOrientedBoundingBox</code> message	Geometry of the 3D stationary object representing the sign.
<code>sign_type_ref</code>	Reference message	Type of sign.

Version History

Introduced in R2022a

See Also

`hd_map_header.proto` | `hd_map.proto` | `hd_lanes.proto` | `hd_junctions.proto` | `hd_barriers.proto` | `hd_static_objects.proto` | `common_attributes.proto` | `geometry.proto`

Topics

“Compile Protocol Buffers for RoadRunner gRPC API”

“Build Scenes from Custom Data Using RoadRunner HD Map”

hd_static_objects.proto

Static object representation messages for RoadRunner HD Map

Description

The `hd_static_objects.proto` protocol buffer (protobuf) file defines the schema for representing static objects in the RoadRunner HD Map scene model. Static objects refer to physical objects such as props and road furniture that do not move.

Protobuf File Location	<i>RRInstallFolder/bin/platform/Proto/mathworks/scenario/scene/hd</i>
Protobuf Syntax	proto3
Package	mathworks.scenario.scene.hdmap

Messages

StaticObjectTypeDefintion — Type of static object message

Type of static object, specified as a message with these fields.

Name	Type	Description
id	string	ID of the static object type. The <code>StaticObject</code> message refers to this field to describe a type of static object.
asset_path	RelativeAssetPath message	Relative path for a 3D mesh or a prop asset file that contains information about the static object type.

StaticObject — Properties of static object message

Properties of the static object, specified as a message with these fields.

Name	Type	Description
id	string	ID of the static object instance.
geometry	scenario.common.GeoOrientedBoundingBox message	Geometry of the 3D static object that fits within an oriented bounding box. Objects are scaled to fit the bounds and are rotated using a <code>GeoOrientation3</code> message.
object_type_ref	Reference message	Type of the static object.

Version History

Introduced in R2022b

See Also

[hd_map_header.proto](#) | [hd_map.proto](#) | [hd_lanes.proto](#) | [hd_junctions.proto](#) |
[hd_signs.proto](#) | [hd_barriers.proto](#) | [common_attributes.proto](#) | [geometry.proto](#)

Topics

“Compile Protocol Buffers for RoadRunner gRPC API”

“Build Scenes from Custom Data Using RoadRunner HD Map”

