755 Phoenix Drive, Ann Arbor MI, 48108, USA
Phone: 734 668-2930 • Fax: 734 668-2877 • Email: info@carsim.com
carsim.com

Scene Tile Creation

Scene tiles are sets of animator shapes and path segments that can be copied and manipulated by the VS Scene Builder tool. The VS Scene Builder can export custom scenarios (maps, animation information, and paths for VS vehicles and moving objects). These tiles are created with a 3D-modeling software called *3DS Max*. The purpose of this document is to go over the proper setup and export of scene tiles, using 3DS Max. This document will *not* cover how to build the geometry, only the proper setup and export of tiles.

> **Note** It is assumed that you have a basic knowledge of 3DS Max, and are familiar with common terms and creation techniques for building and modifying objects using 3DS Max software.

The scene tiles need to be exported in the Open Scene Graph (OSG) format specifically. For information on the OSG format, installing the plugin and export settings please read the Technical Memo: 3DS Max OBJ and OSG Export and Setup.

This document will first walk you through how to create the tiles in 3DS Max, and then onto creating a path, exporting your work, and converting / exporting to the appropriate file type to be imported into your VS product. This document requires the download of the `Scene_Tile_Creation_Tutorial.zip` that can be found on the CarSim download page in the CarSim User Section.

3DS Max Tile Setup

Tiles can be developed into roadway shapes and paths for vehicles to travel down. For this example, we will create a simple tee shape, a straight roadway shape, a 90° turn, and a four-way intersection. Each tile is 300 meters by 300 meters and can snap together in any combination. For this demo we will use the name "**Sample**" as our naming convention. The provided Max file (`Sample_Tiles.max`) is the finished version of the below steps.

> **Note** Tiles can be of any size, however if you would like them to snap to our existing tiles you need to follow two rules. The tile size along the X or Y axis must be an odd multiple of 100m (100m, 300m, 500m…). The path entry/exit points along the edge of a tile must be a multiple of 50m away from the corner, and not a multiple of 100m.

Tile Creation

1. Build the four tiles in 3DS Max each exactly 300 meters by 300 meters centered around 0,0,0 (Figure 1). All the road images should align to the road images on the connecting tiles. For this tutorial, we are using simple geometry. For a more detailed scene, utilize the OSG Level of Detail (LOD) nodes to control mesh density and maintain performance. Each tile is made of multiple objects, but they are exported as a single OSG file.

Note mesh size will affect computer performance. If you are concerned, keep the mesh density low. If you have a more powerful system where this is not an issue, mesh density can be modified / increased.

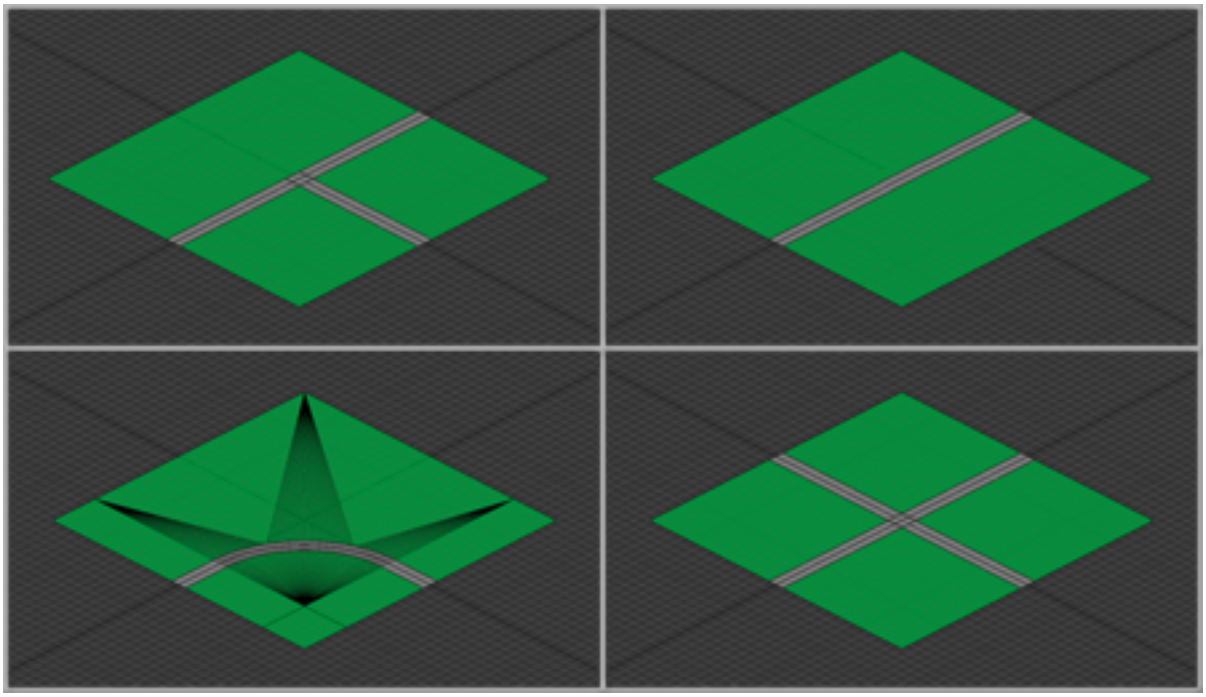

Figure 1. 200m x 200m tiles centered at 0,0,0.

2. Using the **layer system** in 3DS Max, create a layer for each tile below and put the geometry that makes up each tile in the proper layer (Figure 2).
3. All roads and splines should be set to 0 in Z.
    a. `T_01`
    b. `Straight_01`
    c. `90_Turn_01`
    d. `4_Way_01`

Note In this memo, the terms “road” and “road_line” refer to lines and images associated with road surfaces. The first version of the scene builder (2018.0) does not include 3D road surface information of the sort used by the vehicle models in CarSim, TruckSim, and BikeSim (elevation, friction, rolling resistance). The surfaces are all flat and level.

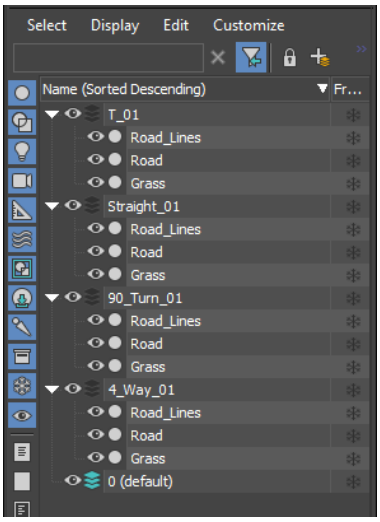

*Figure 2. 3DS Max **layers**.*

Spline (Path) Creation

1. Paths are created in 3DS Max as **spline objects**. Simply divide your road at center line and detach as a spline. Make sure each spline is made from one continuous stroke, not individual segments.
2. The path direction matters, so set the starting point for each path to the correct direction of traffic using the **Make First** button in the **Vertex Sub-Object** panel.
3. Splines should be named logically and should be easily identifiable. For our example, we use the direction and number of path. (Figure 3).

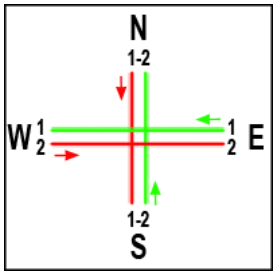

Figure 3. Spline-naming scheme.

4. Create another **layer** named `Splines` and a **sub-layer** for each of the 4 tiles `tee, Straight, 90 and 4-Way` (Figure 4 and Figure 5).
5. Create all the splines for each tile, and place them in the correct sub-layer as shown in Figure 4.
6. In the **vertex sub-panel** make sure that you check the **optimized** box**,** and set the steps to `12`. This is a nice medium for the spline curve, which will maintain good computer performance as well as achieving a nice, smooth curve.

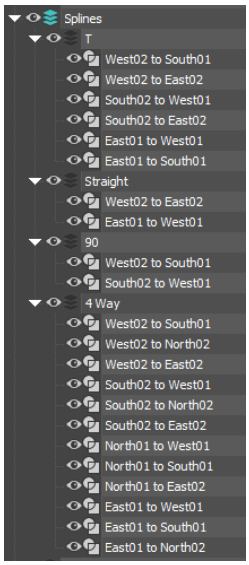

Figure 4. Splines in sub-layers.

Export Nodes

1. We use dummy objects that get exported with the spline data to tell the VS Scene Builder the tile size, tile set and category of the tiles. They are offset along the z-axis for easy selection in the 3DSMAX interface.
2. Create a **standard dummy** object at 0, 0, 10.
3. Rename the dummy object to `*TileSize 300 300 0` (300 x 300-meter tiles)
4. Duplicate the dummy, move it up to 0,0,20 and rename it `*Tileset Sample`.

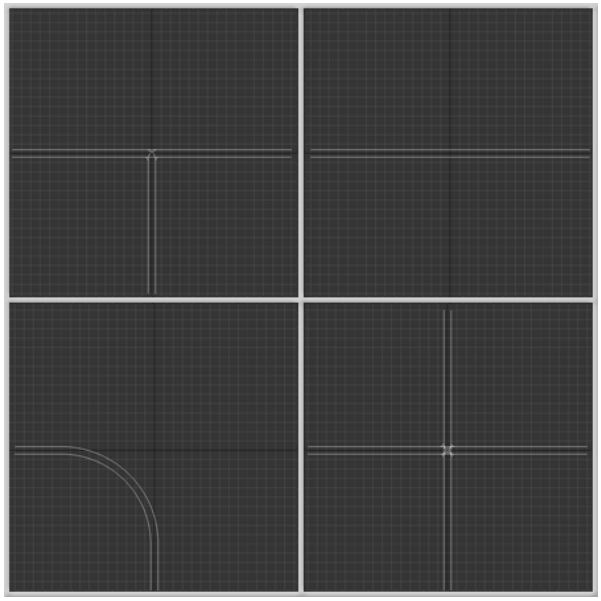

Figure 5. Splines in 3DS Max.

5. You will need to put a copy of these three dummy objects in all four **sub-layers** with the road splines (Figure 6).

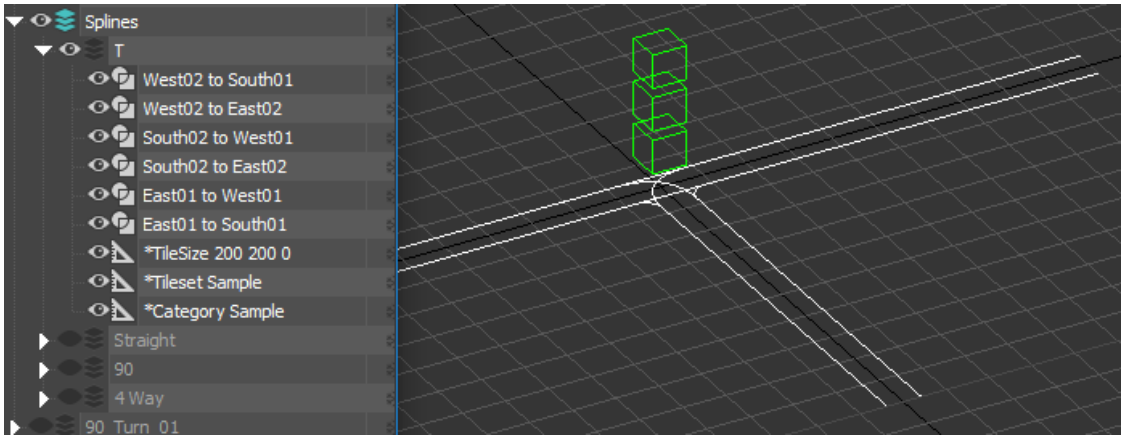

Figure 6. Export nodes (dummy objects).

Exporting ASE and OSG Files

We have now learned how to create the splines which make up the roadways on the tiles, and then constructed everything that we will need for our Sample tile set. Your 3DS Max layers should contain all the elements necessary for proper **vstile** creation. The export steps below will reference the **layer** names for export.

1. Navigate to the directory `CarSim_Prog\Programs\VsSceneBuilder\Resources\Tilesets`. Whether you are using CarSim, TruckSim or BikeSim, the same sub-folders apply.
2. Create a folder named `Sample`.
3. Open the `Sample` folder and create two new folders named `Icons` and `Mesh`.
4. Open the `Mesh` folder and create a folder named `VsV`.
5. Open the `VsV` directory and create an `images` dir. This is where your textures for your scene would go. For this sample tile project, we do not have any textures, just simple colored materials, so this dir will stay empty for the purpose of this tutorial.
6. In 3DS Max turn the **visibility off** on all **layers**.
7. Select the `T_01` layer and turn the **visibility on**. Select all objects that make up `T_01`.
8. Select **File-Export-Selected**.
9. Navigate to the VsV directory you created within the `Sample\Mesh` folder and export `T_01.OSG`.
10. Repeat the above steps for the remaining tiles (`Straight_01.OSG, 90_Turn.OSG and 4_Way_01.OSG`).
11. Turn **visibility off** for all layers, and select the `Splines` **layer**.
12. Turn **visibility on** for the `Splines` **layer** and turn **visibility off** for its **sub-layers**.
13. Select the **T layer** and **select all** splines and the 3 dummy objects.
14. Select **File-Export-Selected**.
15. Navigate to the Sample folder we created.
16. Change the export type to `*.ASE` with these export settings (Figure 6).
17. Save the file as `T_01.ASE`

> **Note** These files must have identical names as the OSG files.

18. Repeat these steps for the remaining spline **sub-layers** `straight_01.ASE, 90_Turn.ASE and 4_Way_01.ASE`.

Creating Icon Files

1. In 3DS Max switch to the top view.
2. **Unhide** and **render** or screen grab an image of each of the four tiles.
3. Open these images in a paint program like photoshop.

4. Crop or size down these images to `300x300 pixels`.
5. Select **File Save As**, change format to **PNG** and navigate to the `Sample\Icons` directory we created.
6. Save each image in this directory with the same name as its matching tile, but keep the correct extension, for example `T_01.png, Straight_01.png, 90_Turn_01.png and 4_Way_01.png`.

Converting the ASE files to vstile files

Now that we've created the tiles, exported the files to ASE format, and also created icons for each tile, we will now use a python script which converts the **ASE** files to a VS Product compatible format, **vstiles**.

> **Note** Before you begin converting the ASE files, first ensure your ASE file export format is set up identical to Figure 7.

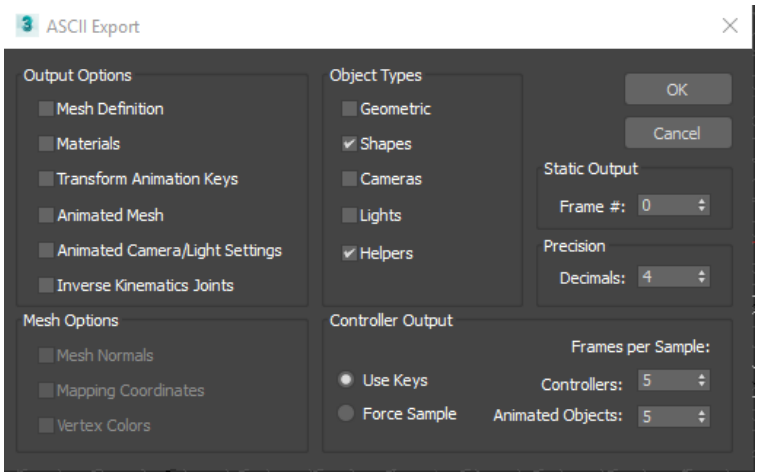

Figure 7. ASE export settings.

The script was developed with **Python 3.6**. If you install this version of Python and associate it with the **.py** extension, you can drag and drop an ASE file onto the script and it will create a **.VSTILE** file with the same base name.

> **Note** If you have the .py file extension registered, which is usually done automatically by the installer, please confirm your working directory is set up correctly. If this is not done, the first time an ASE file is dragged onto the python script it will fail. Subsequent attempts should be fine.

The **VSTILE** file is used by the VS Scene Builder, and contains information about the embedded pathing data, the render data to pass along to the VS Visualizer, the tile icon, and metadata used to categorize and display the tile within the **Tile Palette** in the **VS Scene Builder.**

Using ase2vst.py

1. Navigate to the `Samples` dir with the exported **ASE** files.
2. Drag each **ASE** file individually over the **ase2vst.py** script. This will create the **vstiles** for each **ASE** file.
3. You are now ready to run the VS Scene Builder and create an environment (Figure) that can be used in a VS product (Figure 9). Please access the Scene Builder documentation for more information on how to use the tool.

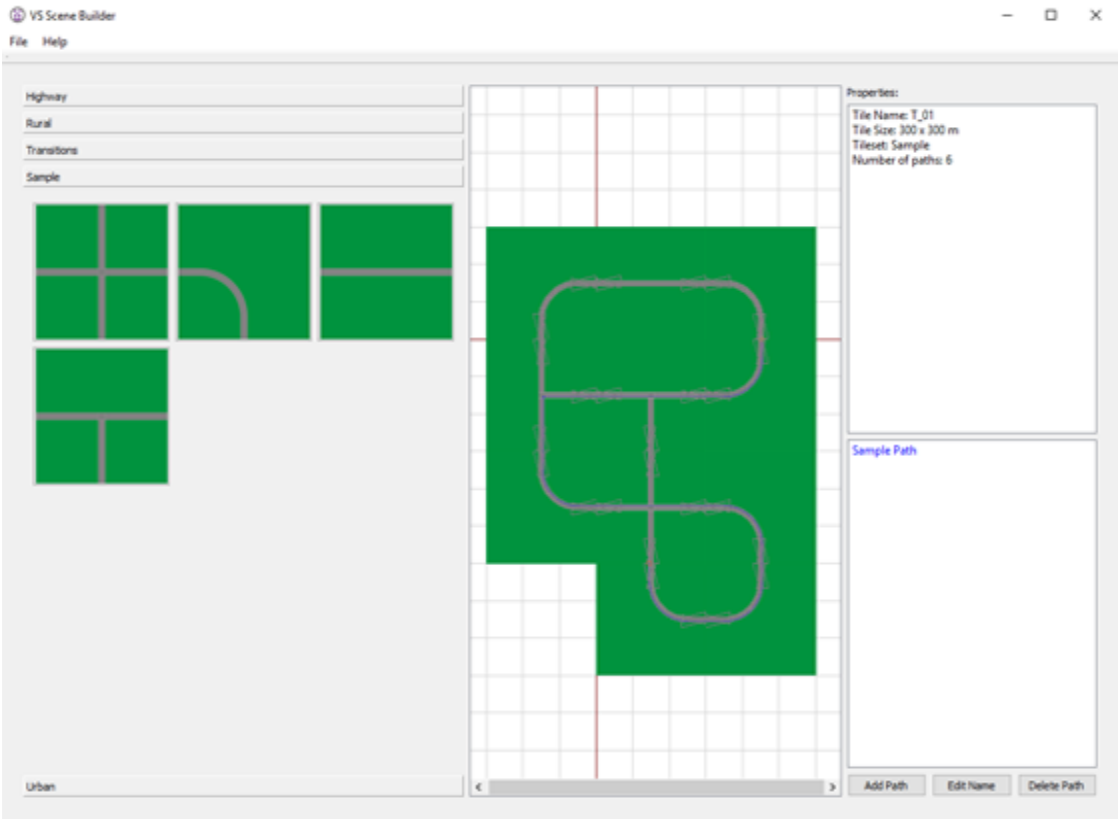

Figure 8. Sample tiles in VS scene builder.

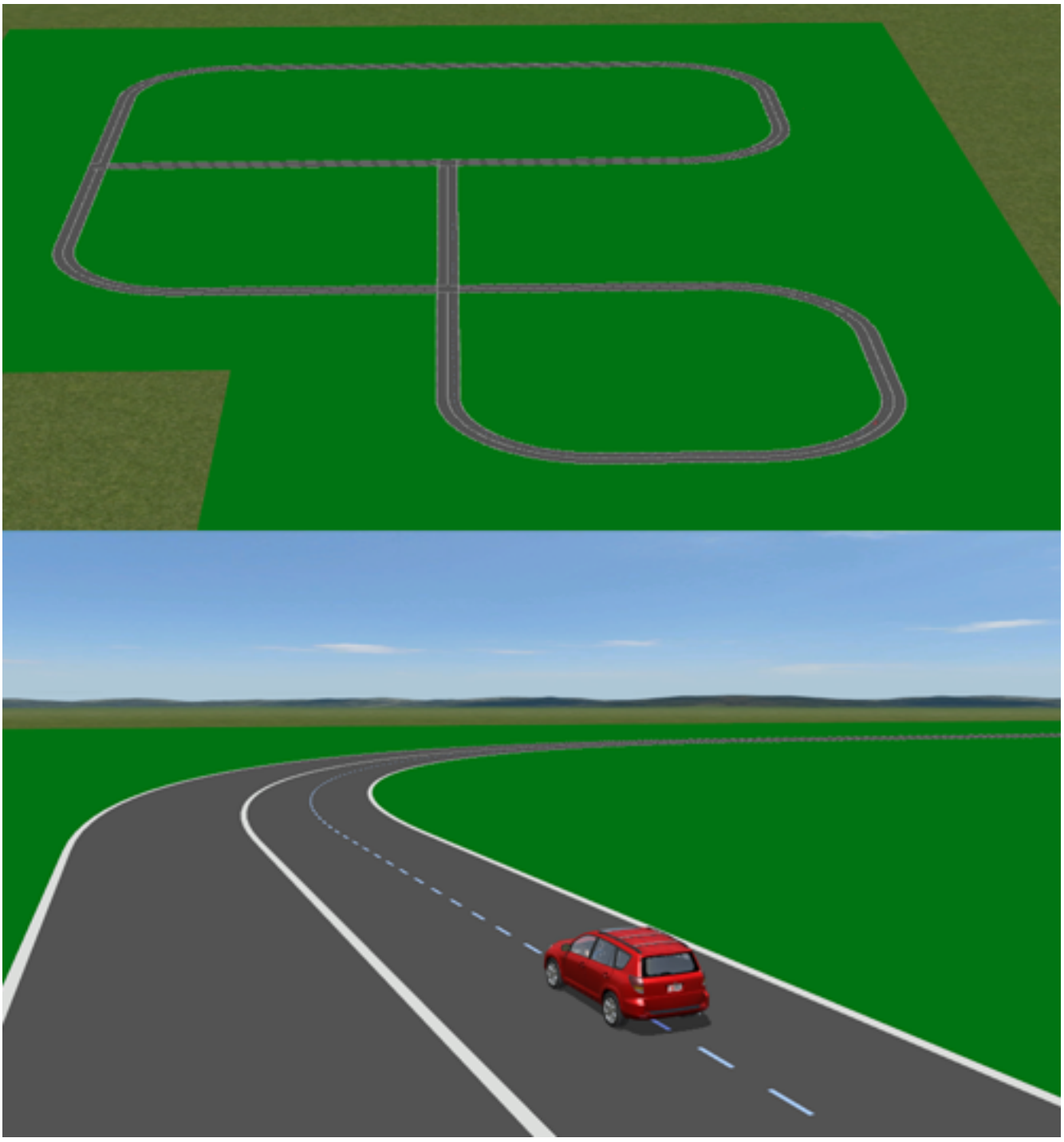

Figure 9. Sample Tiles in CarSim.

Props

Props can be created and added as individual objects to be placed and added to the tiles in the builder. Any object that you would like to add as a prop should be built around 0,0,0 and built to scale. Props can be exported as OSG or OBJ files. For this example we will create a new MAIN ST OBJ road sign. Follow these steps to create a new prop.

1. Copy an existing asset as a template:
    a. Navigate to your CarSim_Prog\Programs\VsSceneBuilder\Resources\Props dir. If you are using TruckSim or BikeSim it's the same sub folders.
    b. Depending on what type of prop you are creating navigate to the appropriate sub folder. For this example open the Road_Signs folder.
    c. Create a new subfolder call Street_Names.
    d. Open an existing prop folder and copy the contents. Lets use Stop01 for this example. CarSim_Prog\Programs\VsSceneBuilder\Resources\Props\Road_Signs\US\Stop01
    e. Paste the contents into the newly created Street_Names folder.
    f. Rename Stop_01.vssceneprop to MAIN_ST.vssceneprop.
    g. Open the subdirectory Mesh, then Vsv and delete the obj, mtl and textures associated with the copied stop sign prop.
2. Create your own versions of the assets:
    a. Create and save your textures in the Props\Road_signs\US\Street_Names\Mesh\Vsv directory. For this example I created MainSt.dds and Sign_Pole_di.dds.
    b. In max create your street sign and assign your textures. Build it to scale, located at 0,0,0 and facing the negative Y axis, as seen in (Figure 10.).
    c. Select the sign and export it out as MainStreet.obj in the Street_Names\Mesh\Vsv directory
    d. Open a texture editing program and create an image to use as an icon for the sign.
    e. Save the image as MainStreet.png in the Street_Names\Icons folder.
3. Edit the JSON file that informs the VS Scene Builder where to locate the prop:
    a. Open the MAIN_ST.vssceneprop file, in the Street_Names folder, in a text editor.
    b. Change the "Name": to "MAIN ST"
    c. Change "Thumbnail": to "Icons/MainStreet.png
    d. Change "Bounds": to [1.0, 1.0, 2.0]
    e. Change "Render Data": to "Mesh/VsV/MainStreet.obj"
4. Validate and test:
    a. Your vssceneprop file should look like this (Figure 11).
    b. Now when you open the VS Scene Builder your new prop will show up in the Props-Signs (US) folder (Figure 12).
    c. Drag the icon onto your road layout and position it where you would like, double tap to rotate (Figure 12).
    d. With a path created export the scene and load it in CarSim, TruckSim or BikeSim.
    e. The new street sign should show up exactly where you placed it (Figure 13).

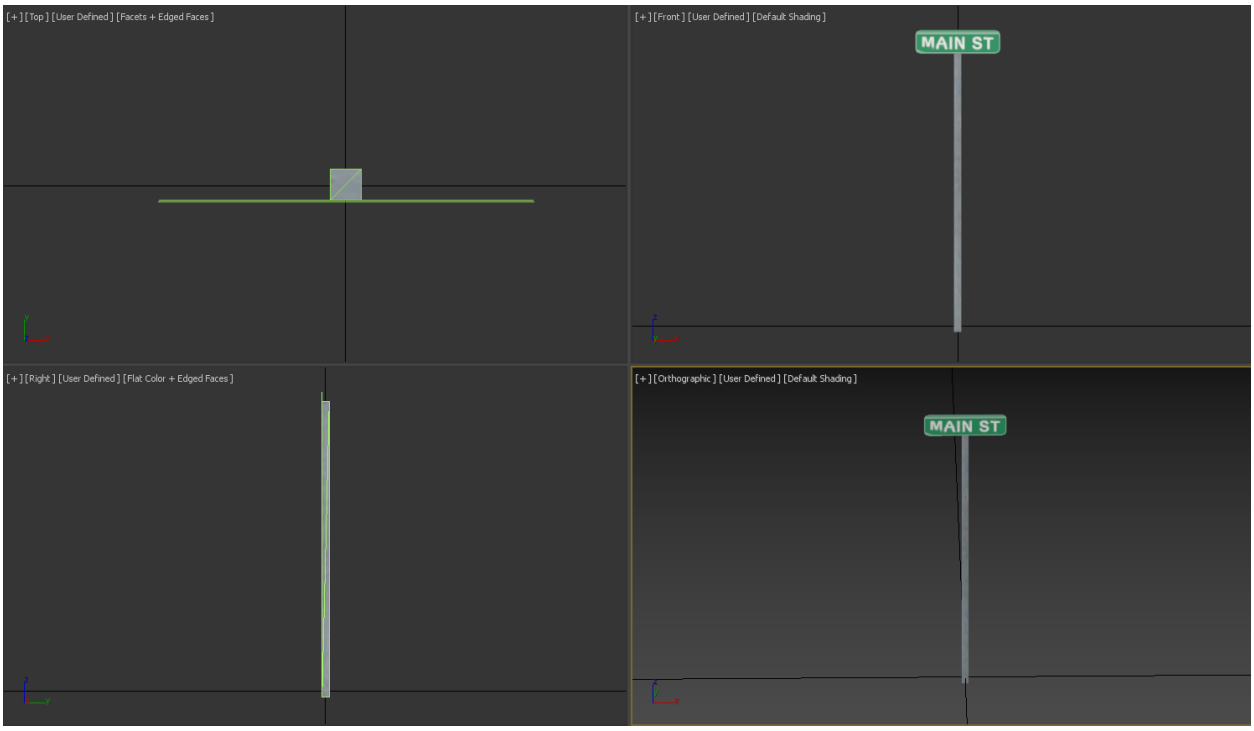

Figure 10. Orientation of new street sign

```
{
  "VsSceneProp": {
    "Version": 1.0,
    "Name": "MAIN ST",
    "Propset": "Signs (US)",
    "Thumbnail": "Icons/MainStreet.png",
    "Units": "m",
    "Coordinates": [ "East", "North", "Up" ],
    "Bounds": [ 1.0, 1.0, 2.0 ],
    "Render Data": "Mesh/VsV/MainStreet.obj",
    "Sensor Object Type": "STOP_SIGN",
    "Sensor Object Message": 4,
    "Sensor Object Diameter": 0.01,
    "Sensor Object Height": 1.8
  }
}
```

Figure 11. Vssceneprop file

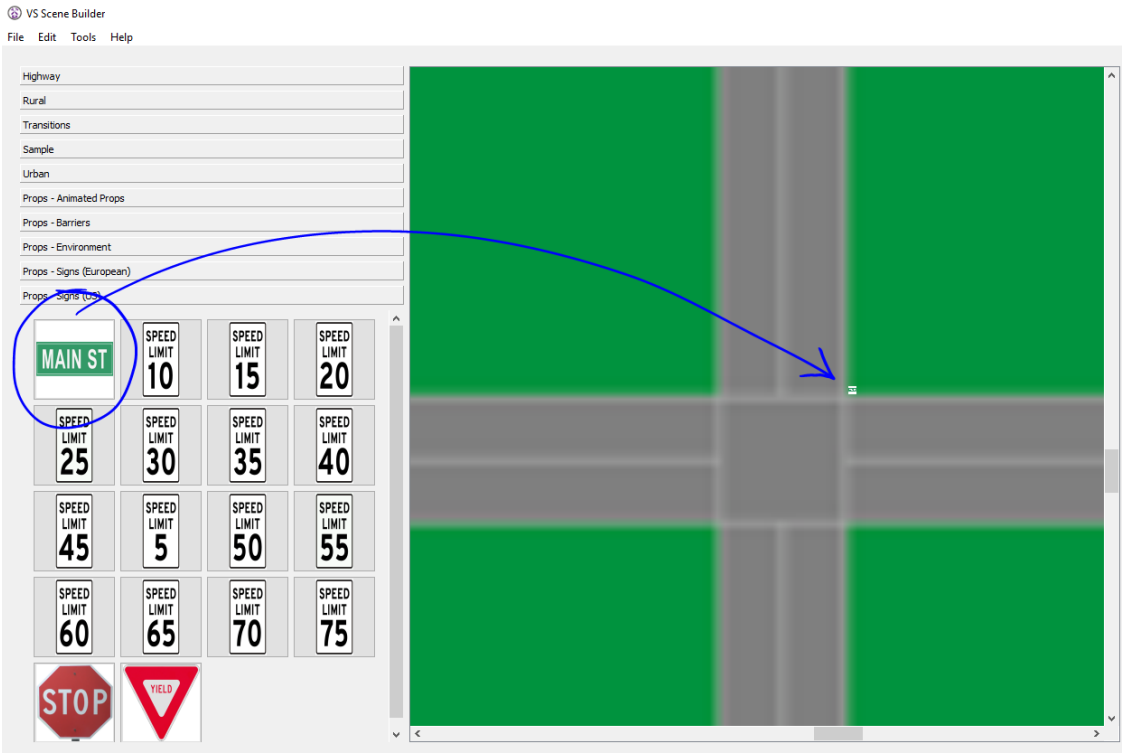

Figure 12. New prop in scene

Figure 13. New prop in scene

VS Terrain API

The new VS Terrain API provides an interface connecting external terrain files to the VS solver tire models. A new `VSTERRAIN` file format is used for representing external data files using a universal terrain data model.

The VS Terrain API allows external wrapper programs to provide basic ground elevation and friction given X-Y coordinates of the vehicle's tires. The VS Solver uses terrain queries that are computationally efficient over large areas. The technology provided as part of VS Terrain is at its best when paired with a dense and large 3D mesh representation of the ground.

Starting with 2019.0, use of the VS Terrain API and accompanying file format is deployed as part of Unreal Engine VehicleSim Dynamics plugin, VS Scene Builder, and the VS Solver.

1. A new triangle-based binary terrain is included within the solvers as the default. Using the VS Terrain module, a user can populate and save a `VSTERRAIN` file. This file can then be passed to the solver to be used for the driving surface.
2. VS Terrain API provides a user-defined terrain callback within the solvers. A user can set an alternative callback via the VS API. This approach is useful if the user is interfacing with other simulation tools that provide terrain information.
3. Individual `VSTERRAIN` files can be attached to tiles in the VS Scene Builder. The data in these files will be merged on export and supplied to the solver to be used with the default callback mentioned above.

VS Terrain for Tiles

All tiles can have an associated VS Terrain file that will define the shape of the tile's terrain. For tiles that do not have an elevation profile, the Scene Builder will create a flat VS Terrain section for the tile during scene export.

Specify a custom VS Terrain by including an Export Node "dummy object" with a reference to the VS Terrain file. The Export Node should be renamed to `*TerrainData T_01.vsterrain`.

For the following example we are going to add a couple of items to the tile to demonstrate what the VS Terrain Utility can do. One item will be part of the road surface and the other will be a cosmetic scene item. The first item we will add is a speed bump near the intersection. This will be useful road information for the VS Terrain. The second item will be a bench on the side of the road. Since it is only cosmetic, the extra polygons in the shape will only slow down the solver when searching for road contact points. The VS Terrain Utility can remove certain meshes based on the materials that are found in the FBX. To use the tile with the VS Terrain Utility export the mesh to an FBX.

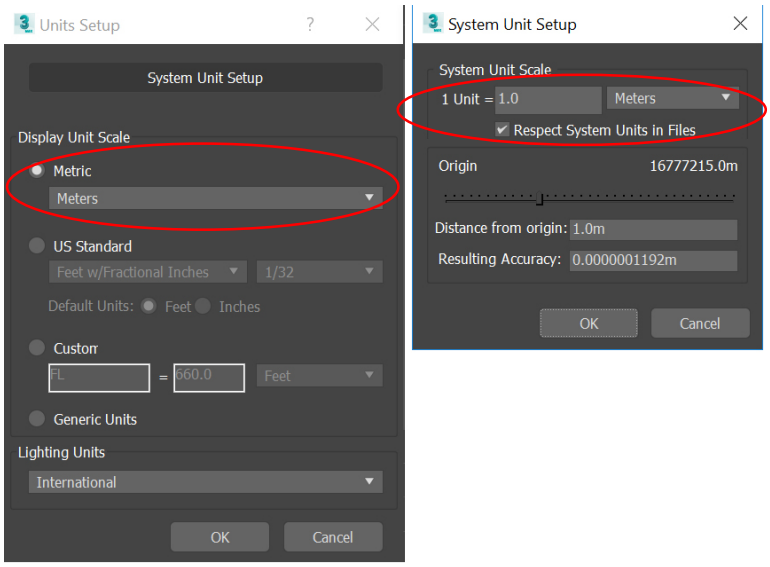

Figure 8: 3DS Max system settings using meters for 1 unit.

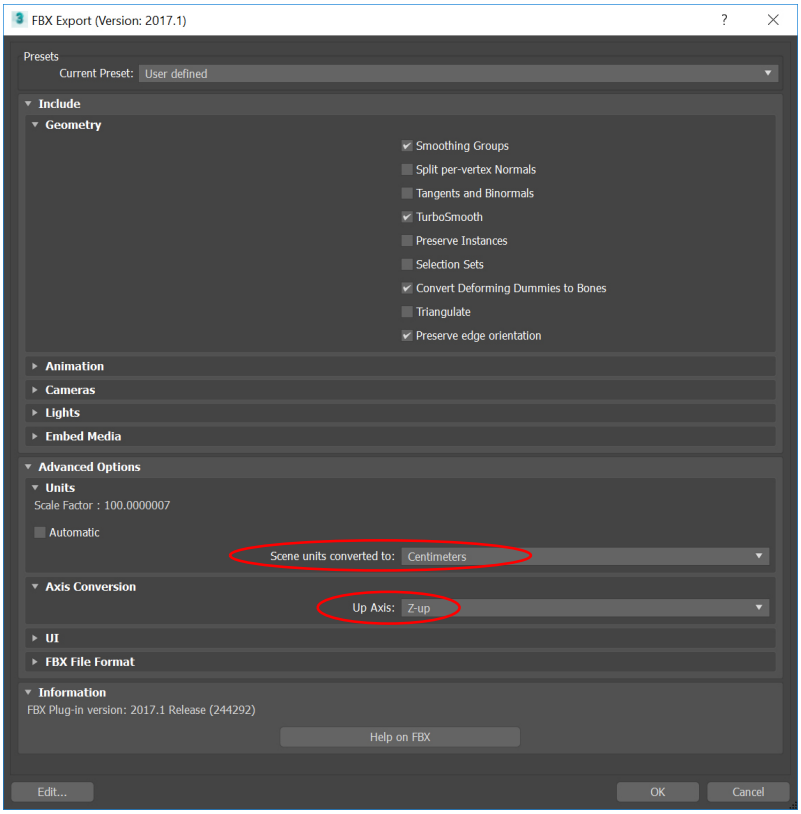

Figure 9: Important FBX export settings and system settings in 3DS Max.

Note If the FBX is going to be used by the VS Visualizer instead of an OSG shape file. The file will need to be scaled to meters. This can be done in the FBX export settings,

VehicleSim shape file link screen, or the VS Terrain Utility will convert and save a file in meters next to the original file.

Using VS Terrain Utility tool

The following steps use the VS Terrain Utility to create a VS Terrain file from the exported tile FBX.

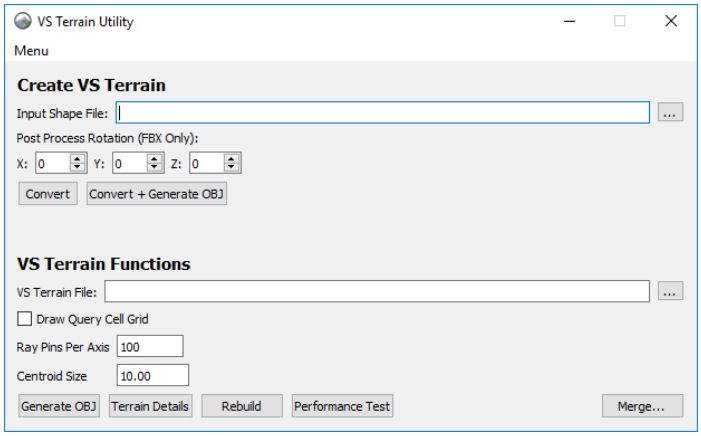

Figure 10: VS Terrain Utility program for creating VS Terrain from FBX files.

1. Select the FBX file exported by 3DS Max.
2. Click `Convert` to create a VS Terrain only or click `Convert + Generate` to create an extra Obj shape file. The Obj shape file will only contain geometry for purpose of debugging and visualizing the VS Terrain. The Obj uses a single material to show the shape of the VS Terrain. If the source of your FBX produces a rotated VS Terrain. Set a rotation that will apply once the FBX is loaded and reoriented by the FBX SDK. Compare the VS Terrain and the FBX using the generated OBJ.
3. After clicking `Convert`, a popup will come up with the detected materials found in the FBX. This is used to set the driving surface properties for different materials. It can also be used to exclude certain meshes that are not necessary for driving within the scene, ie trees, buildings, or signs.

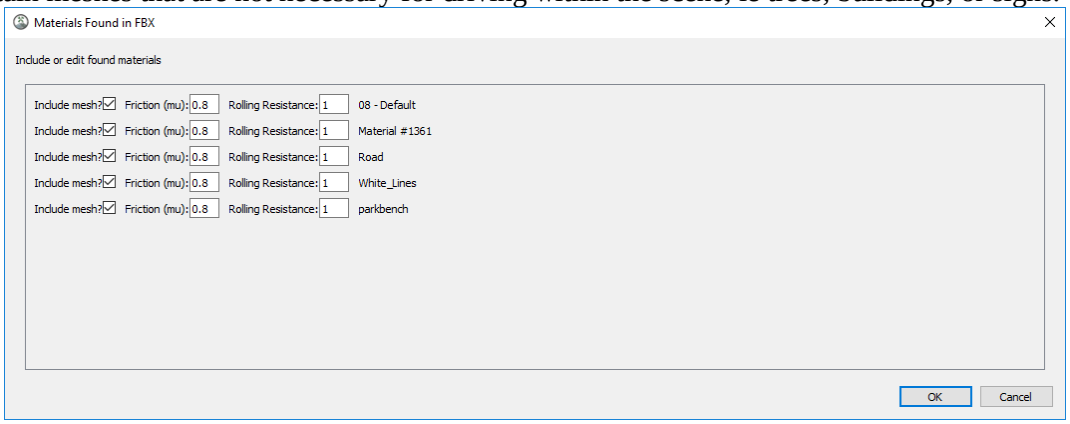

Figure 11: Material Options for materials found in the FBX. Include/exclude meshes and set road surface properties.

4. Once this is done, we can visualize the Debug Obj to check what terrain is in the tile. In the image on the left we can see the `parkbench` mesh is in the terrain. We can quickly recreate the VS Terrain and Obj without the `parkbench` by clicking `Convert` and unchecking the "Include mesh?" option for the `parkbench` when the materials popup is shown.

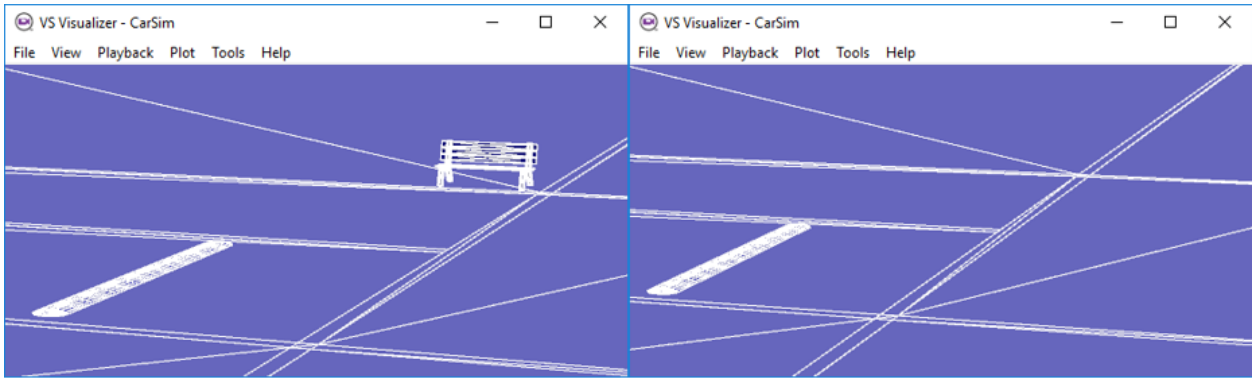

Figure 12: VS Terrain visualized with Obj shape. Image on the left includes the parkbench. Image on the right excluded the parkbench mesh when creating the VS terrain. VS Visualizer scribe render mode.

Below is a picture showing the result in VS Visualizer. The speed bump was included correctly in the VS Terrain. Also, the geometry for the `parkbench` was properly excluded from the terrain as shown above. This is a small example of removing items from the terrain. In a larger scene with many buildings, excluding the buildings from the VS Terrain can improve performance in the vehicle solver.

Figure 13: D-Class sedan driving over the speed bump using VS Terrain in VS Visualizer

For more details on the **VS Terrain Utility**, see the **VS Terrain** technical memo.

VS Terrain Coordinate System

The VS Terrain Utility will use the FBX Axis System `FBXAxisSystem::ZMayaUp` to create the VS Terrain files. All incoming FBX files will be converted to the `ZMayaUp` coordinate system. When creating

a VS Terrain file the summary will notify the user if the FBX file axis system was converted. The `ZMayaUp` coordinate system is compatible with the VehicleSim math solvers.