Automated Road Segment Creation Process

SAFER SIM

SAFEty REsearch USING SIMulation

UNIVERSITY TRANSPORTATION CENTER

David A. Noyce, PhD, PE
Director and Chair
Traffic Operations and Safety Laboratory
Civil and Environmental Engineering Department

Kelvin R. Santiago, MS, PE
Assistant Researcher
Traffic Operations and Safety Laboratory
Civil and Environmental Engineering Department

Automated Road Segment Creation Process

PI
David A. Noyce, PhD, PE
Department Chair and Director (TOPS)
Civil and Environmental Engineering
Traffic Operations and Safety Laboratory
University of Wisconsin-Madison

Kelvin R. Santiago-Chaparro, MS, PE
Assistant Researcher
Civil and Environmental Engineering
Traffic Operations and Safety
Laboratory
University of Wisconsin-Madison

Co-PI
Madhav V. Chitturi, PhD
Associate Researcher
Civil and Environmental Engineering
Traffic Operations and Safety Laboratory
University of Wisconsin-Madison

Co-Author

A Report on Research Sponsored by SaferSim

August 2016

Table of Contents

List of Figures

List of Tables

Abstract

This report provides a summary of a set of tools that can be used to automate the process of generating roadway surfaces from alignment and texture information. The tools developed were created in Python 3.x and rely on the availability of two data sources: roadway alignment in CAD format and a roadway texture in an image format. The goal behind the creation of the tools is simplifying the model creation process by making it feasible to end users of driving simulators to create roads without the need to necessarily understand 3d modelling. The tools presented can be used to study curve behavior and general alignment characteristics in a driving simulator. Furthermore, the tools presented provides a bridge that can greatly simplify the process of sharing models between different driving simulator platforms. For example, models produced by the tools were shared with users of the MiniSim platform which in turn managed to run the same geometry on their platform by importing the model into their scenario authoring tools.

1 Introduction

In recent years, driving simulators have emerged as powerful tools to support highway design evaluations. This trend, combined with advances in CAD technology, allows the creation of driving simulator scenarios that can be used to conduct virtual road safety audits in order to better understand driver performance on the field while benefiting from a controlled laboratory environment. Regardless of the advances in technology, the scenario creation process remains a lengthy and system-specific one. Time spent creating a scenario for a specific simulator provides limited benefit to those who want to run an experiment using the same scenario on a different platform. The lack of scenario compatibility is a barrier to research. Another barrier is the amount of time that researchers have to spend understanding 3D modeling concepts and software tools required for the creation of scenario tiles not part of the standard libraries provided by simulator vendors.

The creation of custom driving simulator scenario tiles is a process that is often required during the design of driving simulator experiments. This custom tile creation process is usually undertaken by researchers that are not necessarily trained in 3D modeling techniques and software. As a result, the time learning new pieces of software and the concepts associated with those can make the time for project delivery longer. An argument can be made that when time available for the completion of a project is fixed, the time spent learning 3D modeling concepts and software by researchers is time that is taken away from the collection of data (running subjects), interpretation of results, and proper experiment planning stages of the experiment. If a layer of abstraction is introduced into the custom tile creation process, this layer can eliminate or reduce the need for researchers to spend time in the labor-intensive texturing and 3D modeling process.

A set of software scripts written in Python are introduced with the potential to reduce project delivery time while fostering collaborative research. The scripts create a layer of abstraction that eliminates or reduces a significant portion of the complexities associated with the scenario creation process. The scripts allow those interested in the creation of scenarios to rely on software tools that are often familiar to those in the engineering field. In particular, the Python scripts rely on roadway definition data created in computer-aided design (CAD) software and cross-section information in the form of a spreadsheet as an input to generate 3D models and roadway metadata that act as the foundation for scenario tiles compatible with multiple driving simulator platforms.

1.1 Limitations and Benefits of the Tools

The Python scripts created are not intended to replace the scenario assembly tools supplied by simulator manufacturers but instead to act as a tool that reduces the amount of time spent in the creation of the tiles. The use of tools provided by the manufacturers will still play a crucial role in integrating or assembling tiles into a format that can be used by the simulator platform. The tools presented in this report were developed to eliminate barriers that have been previously experienced by the authors during the scenario creation process. For example, time spent creating roadways that are not at the core of the experiment have proven to be time consuming but offer little benefit to the results of the final experiment other than providing seamless transitions between areas in the scenario with experimental value.

Based on existing limitations and the areas where greater benefits from automation can be obtained, the scripts created are focused on automating the generation of transitional roadways/tiles. The use of the scripts presented will allow researchers creating driving simulation scenarios to have more time to focus on detailing the core areas of an experiment. Another example of how the tools presented can provide benefit is in studying different geometric conditions of highways. For example, if an experiment calls for studying different alignment configurations, these roadway models can be created very efficiently by modifying the input data used by the scripts presented.

1.2 Summary of Report Structure

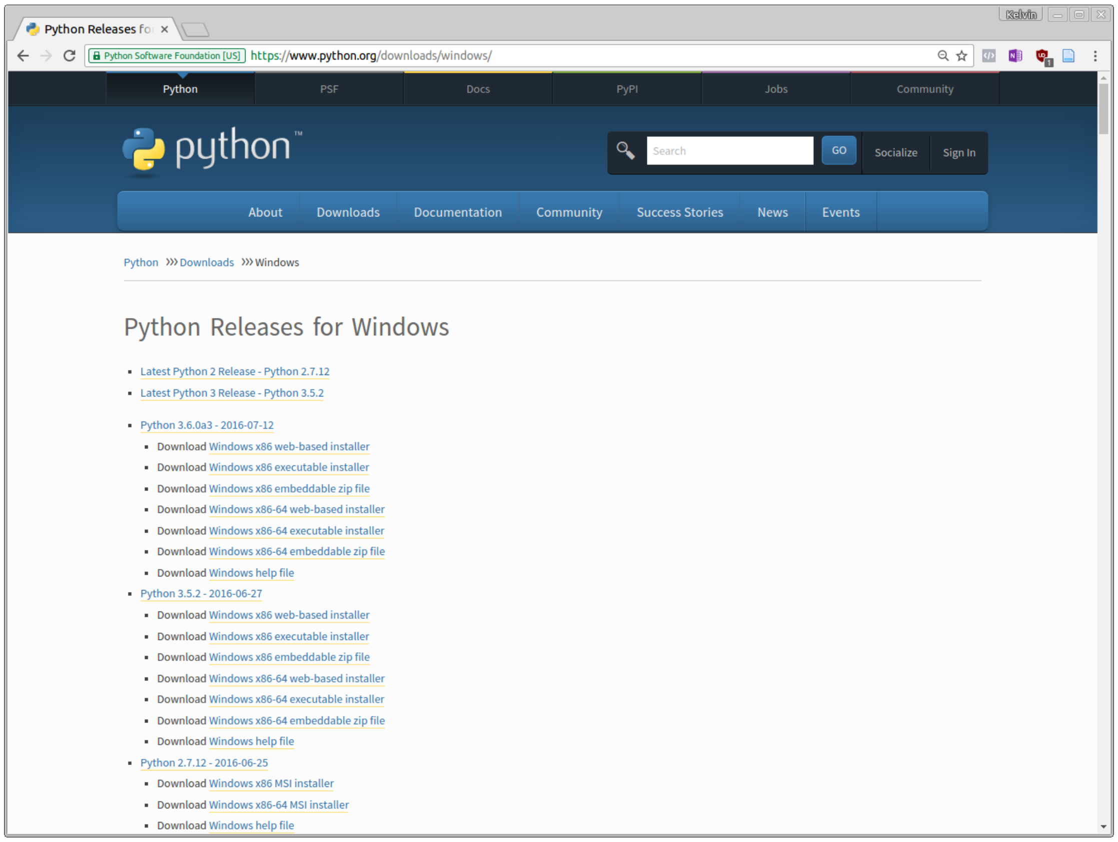
This report focuses on understanding the steps required for preparing the data and on the utilization of the data to streamline and automate critical parts of the driving simulator scenario creation process. Each part of the process is listed as an individual section in order to detail the steps required along with supplemental information. The first part of the report deals with the installation of the Python interpreter, a key component of the procedures described given that each script presented requires the Python interpreter to run. Once a description is provided regarding the installation of Python, two scripts are introduced in the procedure implementation sections. The two scripts introduced can be used to generate the 3D model of the roadway surface as well as supplemental metadata.

2 Environment Preparation

All procedures described in this report rely on the users having Python installed on their computers. The installation of Python is described in the sections ahead along with the installation of a numerical computations package named numpy.

2.1 Downloading and Installing the Python Distribution

The Python distribution can be downloaded from the Python.org website. An assumption in the instructions ahead is that the user will install Python on a Windows computer. Downloading the installer requires visiting http://www.python.org/downloads/windows/ and selecting the latest stable version. A screenshot of the download page is shown in Figure 2.1. As of summer 2015, the latest stable version of Python was 3.5.2.

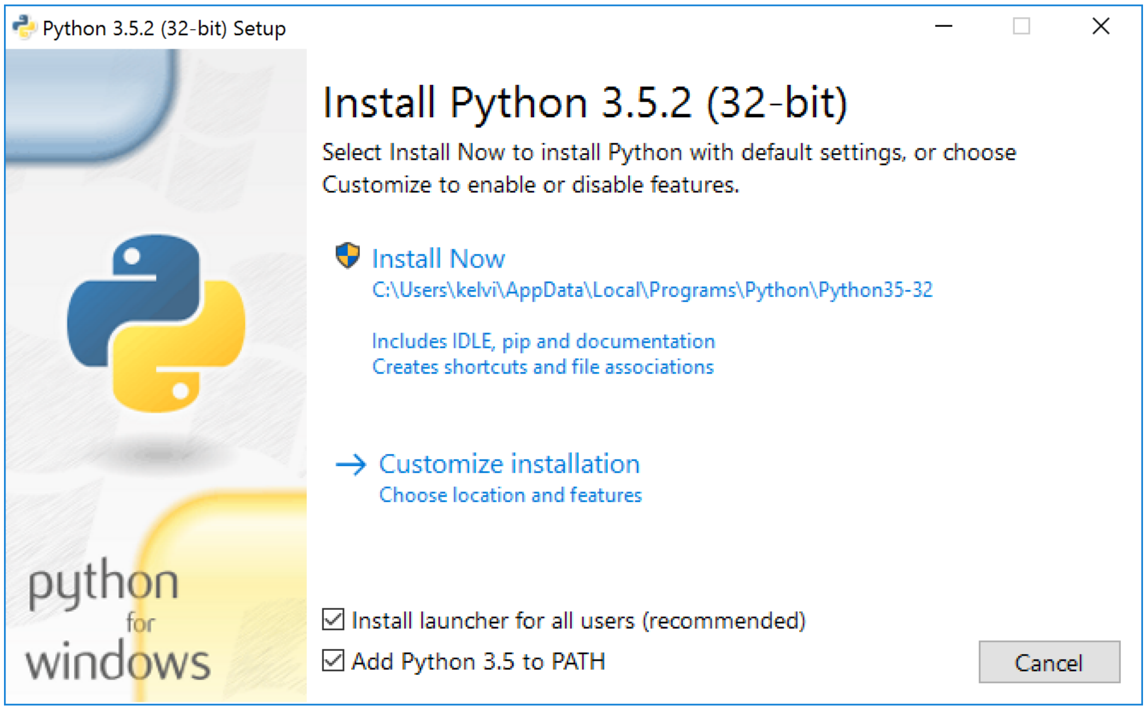


Python Releases for Windows

- Latest Python 2 Release - Python 2.7.12
- Latest Python 3 Release - Python 3.5.2
- Python 3.6.0a3 - 2016-07-12
  - Download Windows x86 web-based installer
  - Download Windows x86 executable installer
  - Download Windows x86 embeddable zip file
  - Download Windows x86-64 web-based installer
  - Download Windows x86-64 executable installer
  - Download Windows x86-64 embeddable zip file
  - Download Windows help file
- Python 3.5.2 - 2016-06-27
  - Download Windows x86 web-based installer
  - Download Windows x86 executable installer
  - Download Windows x86 embeddable zip file
  - Download Windows x86-64 web-based installer
  - Download Windows x86-64 executable installer
  - Download Windows x86-64 embeddable zip file
  - Download Windows help file
- Python 2.7.12 - 2016-06-25
  - Download Windows x86 MSI installer
  - Download Windows x86-64 MSI installer
  - Download Windows help file

Figure 2.1 – Download page for Python.

From the web page shown in the previous figure, select the Windows x86 executable installer link corresponding to the latest stable version. An executable file will be downloaded. When launched, the executable file will display a window such as the one shown in Figure 2.2. In the window, make sure that the option for “add Python 3.5 to PATH” is checked. This will make Python accessible from the command line by typing Python from any directory. Once the PATH option is selected, proceed to click on the “Install Python 3.5.2” button. Follow the instructions on the screen and the installation will continue.



Python 3.5.2 (32-bit) Setup

Install Python 3.5.2 (32-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\kelvi\AppData\Local\Programs\Python\Python35-32

Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☑ Add Python 3.5 to PATH

Figure 2.2 – Python installer window.

2.2 <u>Using a Python Script</u>

Once Python is installed, all files with a “.py” extension are associated with the Python interpreter. Scripts such as the ones described in this report can then be executed from the command line. For example, to execute a Python script named

"GenRoadSurface.py" once the command line window is active on a directory that contains the desired scripts, the script can be executed by typing "python GenRoadSurface.py". To open a command line prompt window navigate to the directory containing the scripts, click *File* on the File Menu, select *Open command prompt*, and click on *Open command prompt*.

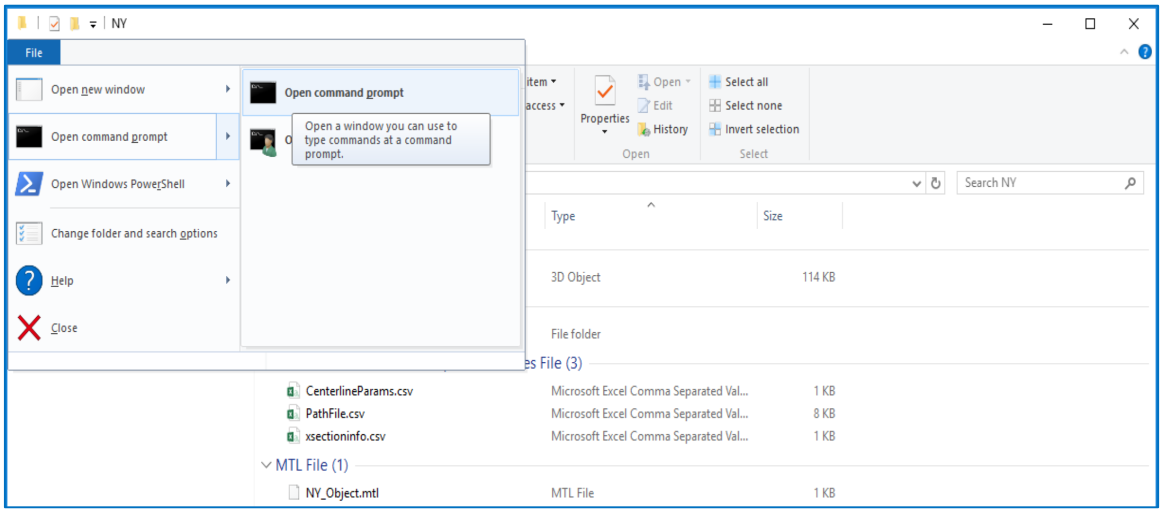
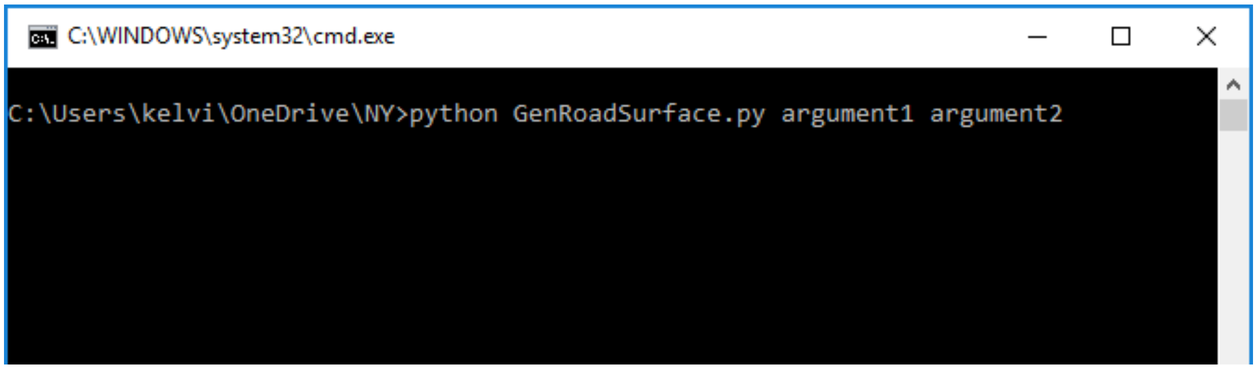


Figure 2.3 – Launching the command line window.

A command line window such as the one shown in Figure 2.4 will open, allowing you to type commands. The structure used for the command is as follows. First, the word *python* is used. Second, the *name of script*. Finally, a list of *startup arguments* is passed to the script by typing their names. In the report, scripts presented perform a specific function. Each script has a specific name and required arguments that make their functionality possible. To use the scripts presented, follow the command structure shown in Figure 2.4.



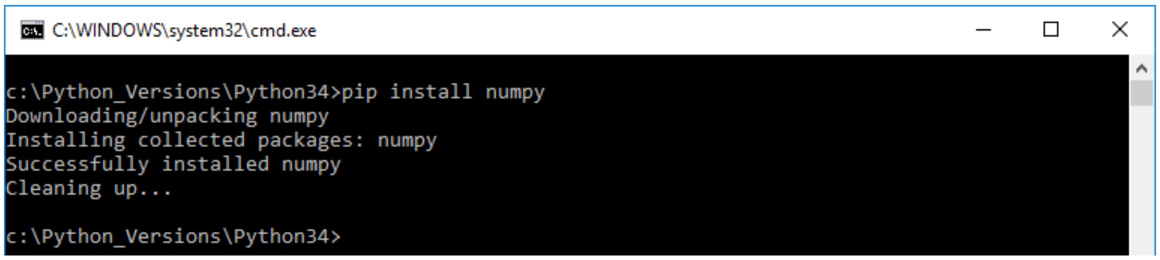
```
C:\WINDOWS\system32\cmd.exe

C:\Users\kelvi\OneDrive\NY>python GenRoadSurface.py argument1 argument2
```

Figure 2.4 – Sample execution of Python script.

2.3 Installing the Numpy Package

The installation of Python previously described provides core functionality. Python scripts to generate roadway metadata rely on a numerical computation package capable of performing matrix operations. A package in Python can be treated as a library of functions and tools that expand the functionality of the language by providing additional functions that simplify the script-writing process. The package used by the roadway metadata generation script is called numpy. Numpy is a popular computation package for Python. The simplest method to install numpy is via the built-in Python package manager. Once Python is installed, to install numpy you can open a command line window and type “pip install numpy”. Figure 2.5 shows a screen of a successful installation process.



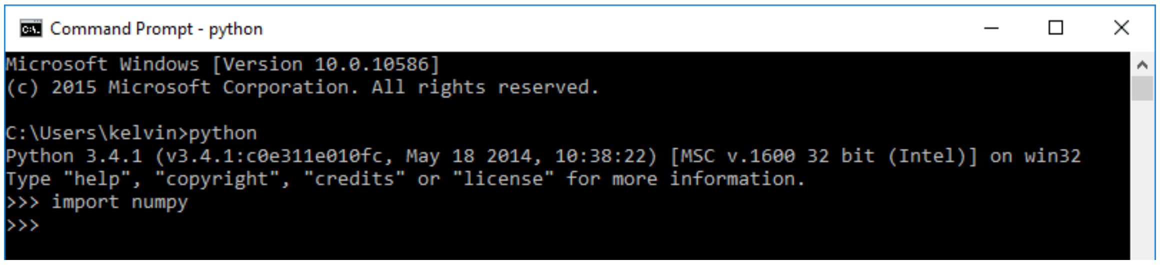
```
C:\WINDOWS\system32\cmd.exe

c:\Python_Versions\Python34>pip install numpy
Downloading/unpacking numpy
Installing collected packages: numpy
Successfully installed numpy
Cleaning up...

c:\Python_Versions\Python34>
```

Figure 2.5 – Installation of numpy.

To confirm the numpy installation, open a command line window and type "python". If successful, a message should appear displaying the version of Python installed followed by a ">>>" prompt. Type "import numpy" and press enter. If another ">>>" prompt follows, the installation was successful. Figure 2.6 shows the confirmation process.



```
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\kelvin>python
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:38:22) [MSC v.1600 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>>
```

Figure 2.6 – Confirming the installation of numpy.

3 Process Implementation

The sections ahead summarize the steps that need to be followed in order to go from the concept of a road section to a 3D model of that section through the use of an automated set of tools. Steps are presented as Python scripts that require specific input data in order to generate 3D models and metadata used in the scenario creation process.

3.1 Break the Scenario into Candidate Segments

As previously mentioned, automating the entire scenario creation process is not a feasible solution and is not the goal of this project. Instead, the goal is streamlining the scenario creation process. In order to streamline the process and take advantage of the Python scripts that will be described in the sections ahead, the first step in the process is identifying parts of a desired driving simulator scenario that can be created with the scripts described in this report. Segments of the scenario that are candidates for the automation process are: segments typically considered warmup sections, segments located in between experimental areas of the scenario, or segments that call for the use of a continuous cross-section along an alignment.

An experimental area is a part of the scenario that calls for detailed data analysis and perhaps an area during which events will be triggered in order to study the reaction of subjects. These are areas characterized by requiring scripts to triggering events, areas that contain complex geometric changes, and intersections, among others. When viewed from the perspective of the overall scenario, these areas are not large and can be connected with road segments that can be automatically created. Segments that are candidates for being automatically created need to have one key characteristic: the cross-section is similar along the entire segment. The aforementioned cross-section continuity is a realistic requirement because real-world roads are characterized by a cross-section that spans for miles followed by a transition (which could be thought of as an experimental area) and then a new continuous cross-section that follows.

Figure 3.1 shows a schematic of a driving simulator scenario in which there are three experimental areas (A, B, and C) connected by road segments that can be automatically generated. In the hypothetical scenario structure, the connections between test areas could be automatically generated. Contrary to what the schematic shows, the connection segments don’t have to be straight ones. These automatically generated roadway segments could have changes in the vertical profile, contain horizontal curves, or a combination of both. As the methodology outlined stands, the geometry of the segments that connect the experimental areas can be defined in a CAD program by representing the inner and outer edges of a roadway surface along with the corresponding roadside as polylines in the CAD model.

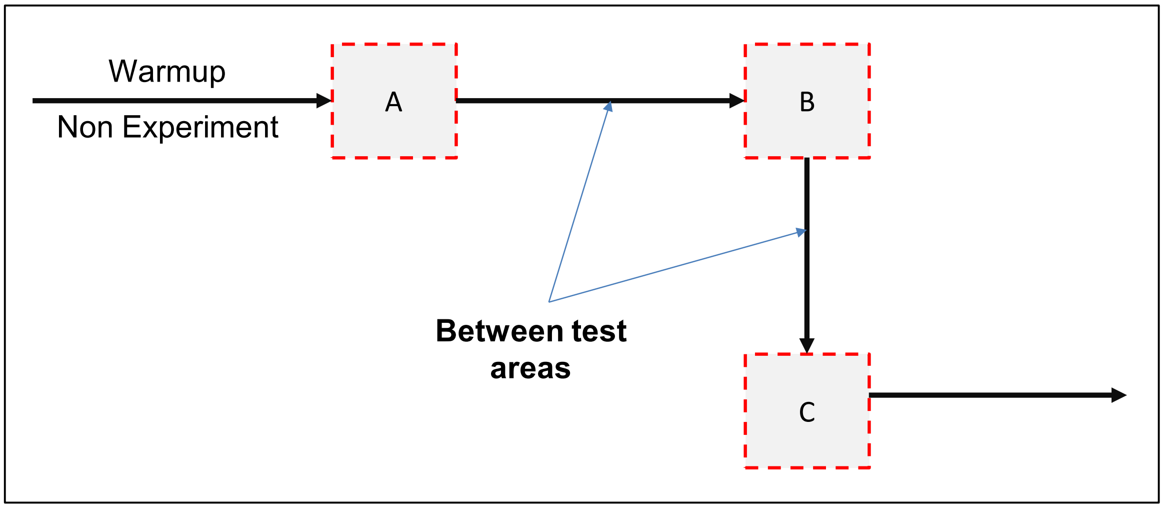


Figure 3.1 – Example of segments with potential for automation.

3.2 <u>Roadway Data Preparation and Generation</u>

The creation of the data required for a 3D model starts with the identification of edges on the sections of roadway that need to be created. Figure 3.2 shows an example of a roadway and highlights some of the edges that need to be defined as part of the outlined process. As the edges highlighted suggest, the lines required are those lines that would be included in a top-view representation of the model, including those that are masked by lines at a higher elevation. It should be noted that while the term *line* is used, for purposes of the CAD model creation discussed in the next section, the aforementioned lines need to be created as polyline objects.

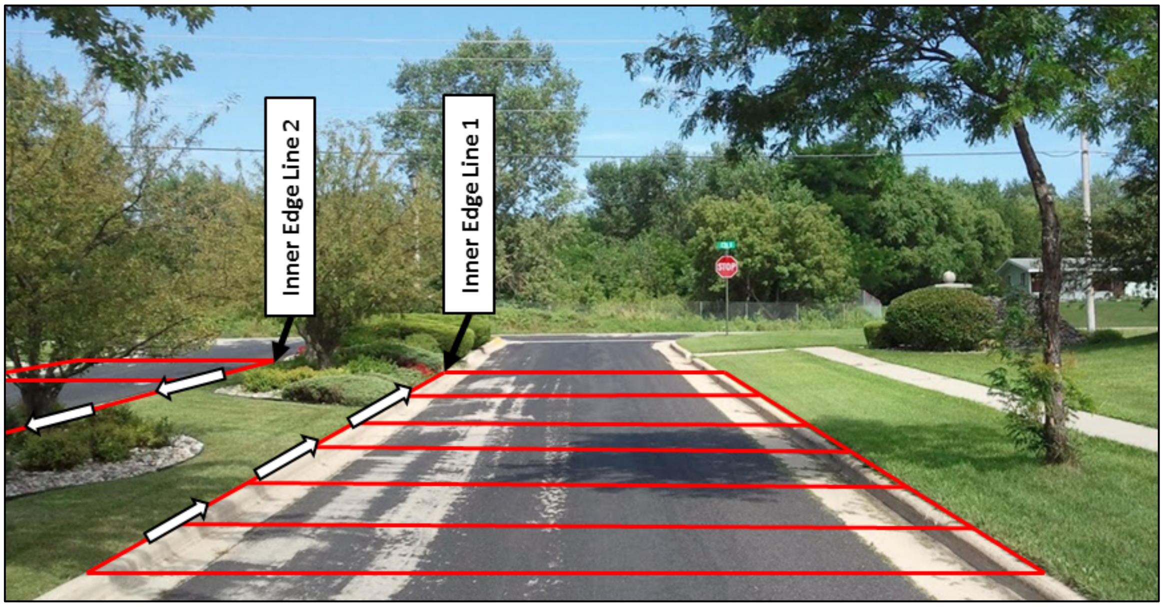


Figure 3.2 – Identification of required components on a real roadway.

3.2.1 Creation of a CAD model

The first step in the process of generating the necessary data once the lines that need to be created are identified is the creation of a CAD model representing the top view of the desired roadway section. For purposes of the explanation it will be assumed that the elevation of each vertex of the lines is zero; however, it can have any elevation value and can work with scenarios that include vertical curves and gradients. The actual creation of the polylines can be done in a user's software of choice, e.g., AutoCAD^{TM} or

MicroStation^{TM}, among others. One requirement that should be taken into consideration is that each edge of the roadway represented in the CAD model needs to be defined as a polyline since this allows the edge as a continuous object with multiple vertices instead of a collection of multiple objects, each with two vertices. The use of polylines makes the process of including changes in elevation simpler because each vertex can be assigned a specific elevation that represents the profile of the road. The specific elevation profile can be obtained by standard engineering computations or by using specific software capable of modifying the polyline elevations to represent the profile of a road.

3.2.2 Generation of files defining model polylines

Each polyline that is included in the CAD model needs to be exported as a CSV file that contains rows of data, each row containing the x, y, and z coordinate that defines the vertices of the line. Before exporting, the user should make sure that the polylines to be exported are considered “clean” based on the guidelines provided in the next section. Also, tools discussed in this report assume that z coordinates represent elevation, that x coordinates grow from left to right, and that the y axis is perpendicular to the x-z plane. A representation of the aforementioned axis directions is shown in Figure 3.3.

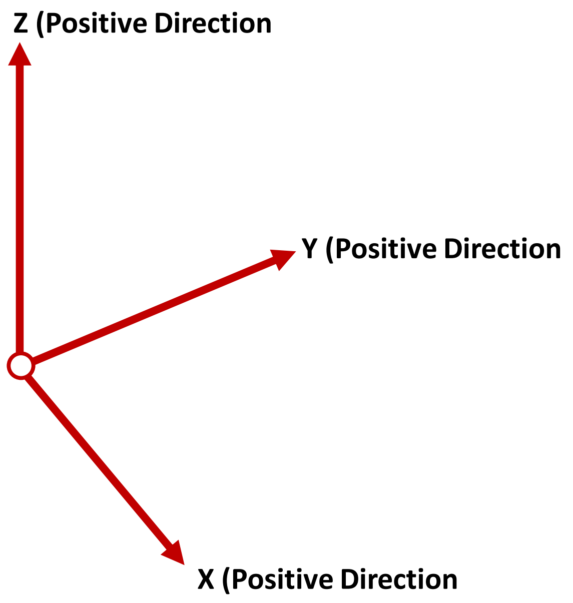
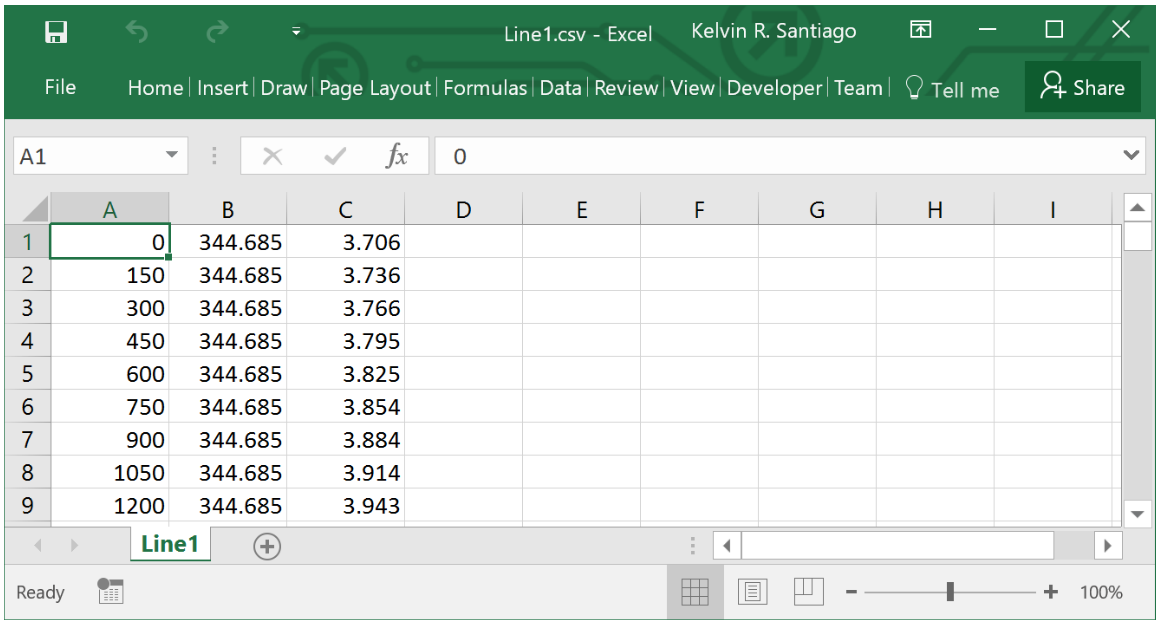


Figure 3.3 – Assumed axis direction.

The procedure to extract the coordinates that define the poly lines from the CAD model are dependent on the CAD software used. As a reference, the procedure used by the authors to convert the polylines in the model into CSV files relied on the CADTools software (www.glamsen.se/CadTools.htm), which expands the functionality of AutoCAD^{TM}. Each polyline in the model was converted into individual CSV files that are used by the tool as described in the sections ahead. After extra columns and rows are removed, the resulting CSV files should look as the one shown in Figure 3.4. In the figure, Column A contains the x coordinates of the line, Column B the y coordinates, and Column C the z coordinates.



	A	B	C
1	0	344.685	3.706
2	150	344.685	3.736
3	300	344.685	3.766
4	450	344.685	3.795
5	600	344.685	3.825
6	750	344.685	3.854
7	900	344.685	3.884
8	1050	344.685	3.914
9	1200	344.685	3.943

Figure 3.4 – Example of CSV File Defining Edge of Roadway

3.3 Polylines Cleanup

Polylines created from the CAD model representing a top view based on the road segment and on aerial images, blueprints, or from scratch, should be cleaned before export. The "cleaning" process is required in order to eliminate vertices that are too close to each other or that in some cases virtually overlap with another. The final goal of the cleanup process is to have CSV files that define each edge of the surface with an equal amount of points. The equal amount of points is the result of each edge line segment having a corresponding segment, at a given offset, on every other edge line of the road. For road segments such as those that are the focus of automation in this project, e.g., the one in Figure 3.2, the aforementioned same amount of points requirements can be achieved without problems.

Cleanup of the polylines should and can be achieved in a CAD environment, and may not be required if proper drafting techniques are followed. Therefore, the best practice for avoiding having to perform a cleanup of the edge lines that define the road is to plan the edge line definition early in the process before line-definition data is exported.

3.4 Texture Preparation

The preparation of textures for the project is a relatively simple process that can grow in complexity if photo-realistic textures are desired for the segment. However, the quality of the textures, and their creation, is something left to the end user.

Requirements for textures that can work with the scripts presented are as follows:

- The y axis of the texture (in an x and y coordinate system) should be thought of as parallel to the roadway alignment.
- The bottom of the texture should align with the top of the texture. In other words, there should be some mirroring effect across an imaginary x axis in the middle of the texture.
- While not a requirement for all video platforms, the dimensions of the texture should have a size that represents a power of two. This does not mean that textures should be square, but that the dimension of each side should be a power of 2.

All textures used in the creation of a model should be inside a folder called *TextureLibrary*, and the name should be unique regardless of the format. For example, if there is a file called Image1.png, there should not be another file called Image1.jpg. The role played by texture and how a texture is applied to each face of the model is better described in Section 3.5.2.

3.5 Roadway Surface Creation

A 3D model of a roadway surface is the desired output of the process, if the need for metadata is not considered. The 3D model enables sharing the environment across multiple platforms and provides the foundation for further detailing of the model. Using the procedure described ahead, a 3D model can be obtained by relying on the previously discussed line definitions and the required texture.

3.5.1 Process Definition

The generation of roadway 3D model (in OBJ format) requires the use of a script that receives an object name, the top view coordinates of the desired model, and a configuration file that defines desired textures. Parameter values required for the operation of the script are further summarized in Table 3.1. The name of the object (param1) should be a short name, without spaces, that describe the object that was just created. The cross-section configuration file (param2) is a CSV file that provides information about how the cross-section of the road will be textured. Finally, the lines that define the edges of the mode (param3 and beyond) are the csv files discussed in the previous section that contain the coordinates of the surface edges.

Table 3.1 – Roadway surface generation: parameter structure.

```
python GenRoadSurface.py   param1    param2    param3 … paramN
```

Where:
    param1: name of the desired output object
    param2: file defining the configuration of the cross section
    param3: params3 and beyond are the CSV files containing the coordinates of the lines
            that define the cross section of the road.

The end result of the script described in Table 3.1 is a 3D model such as the one shown in Figure 3.5, which can be opened by virtually all 3D modeling software, due to the use of the OBJ format. As the figure shows, the coordinates included on each csv file have been used to generate 3D faces and texture the faces accordingly based on information included in the cross-section definition file (param2). Texturing is achieved by generating a UV map that makes sure that textures are properly placed and offset in reference to model edges.

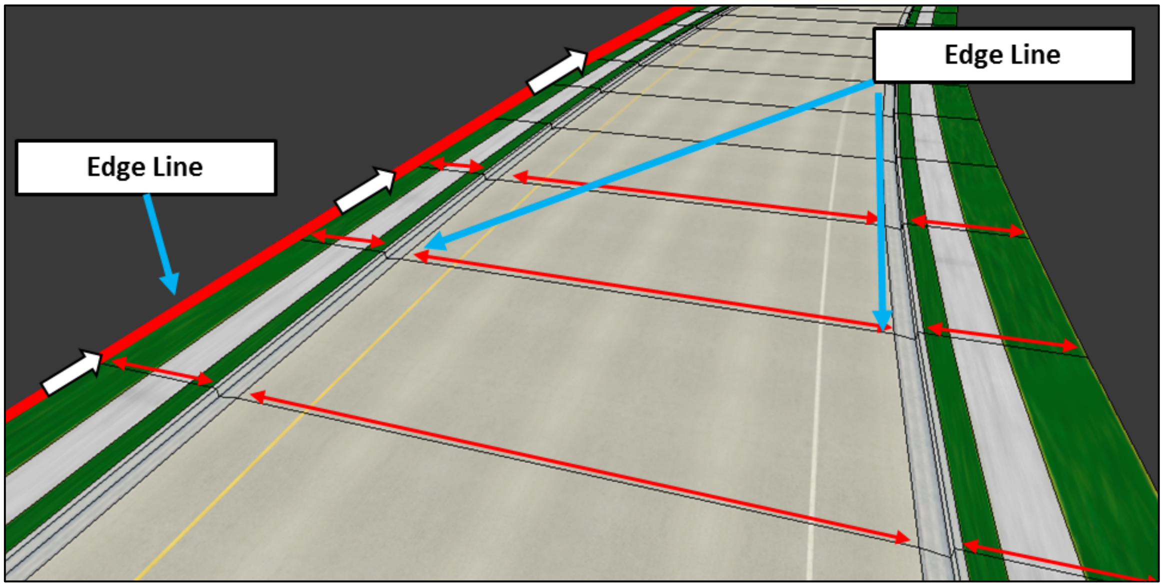
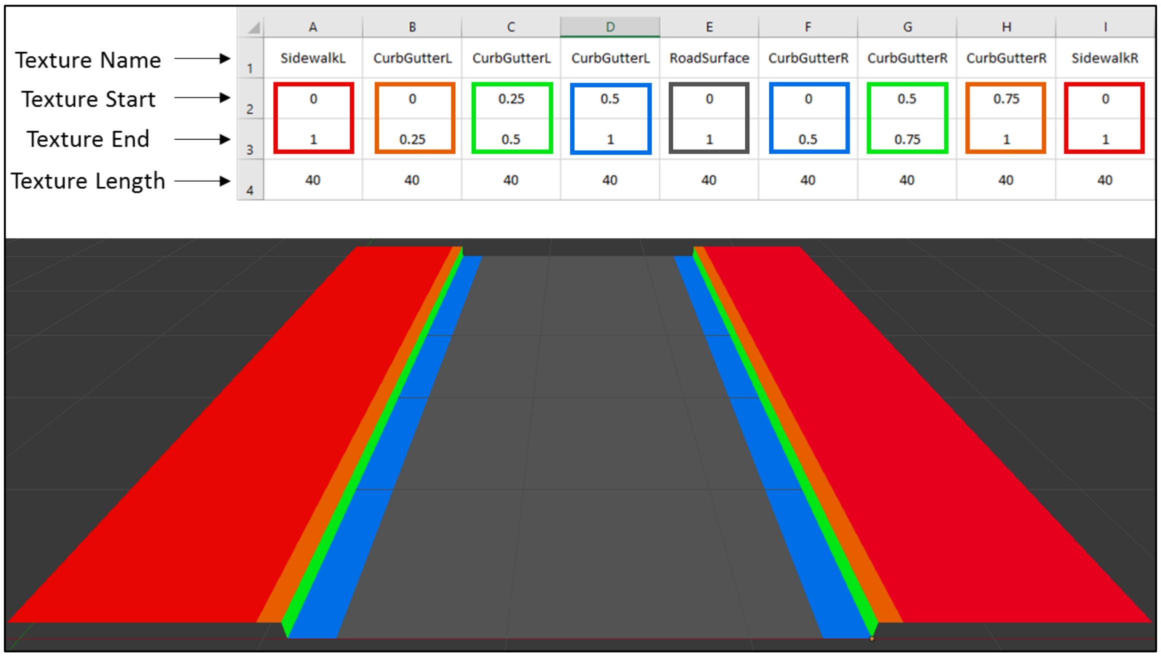


Figure 3.5 – Sample roadway surface produced by script.

3.5.2 Cross-section definition file details

The param2 file specified in Table 3.1 defines the characteristics of the cross-section that will be extruded and textured along the road edge lines specified by param3 and beyond. The structure of the file is comma separated value (CSV). The contents of a sample param2 file are shown in Figure 3.6. As shown, each column represents a texture that will be applied to a group of faces in the model. For example, column A applies a texture named *SidewalkL* to faces between the boundaries of the first two lines passed as edge lines to the surface generation script.

The name of the texture is specified in Row 1. The name of the texture is a reference to an image file located in a folder called *TextureLibrary* located in the same directory of the Python scripts. Rows 2 and 3 of each column represent the portion of the texture (measured along the x axis and expressed in values that range from 0.0 to 1.0) that is applied to the face. Row 4 of each column contains the real-world length represented by the texture name and corresponding file.



		A	B	C	D	E	F	G	H	I
Texture Name →	1	SidewalkL	CurbGutterL	CurbGutterL	CurbGutterL	RoadSurface	CurbGutterR	CurbGutterR	CurbGutterR	SidewalkR
Texture Start →	2	0	0	0.25	0.5	0	0	0.5	0.75	0
Texture End →	3	1	0.25	0.5	1	1	0.5	0.75	1	1
Texture Length →	4	40	40	40	40	40	40	40	40	40

Figure 3.6 – Characteristics of the CSV file.

In the case of a texture 1000 pixels wide by 2000 pixels tall, if the start and end values specified for the texture are 0.25 and 0.75, respectively, then the parts of the texture file that will be used for each face are the image portion between 250 pixels and 750 pixels on the x axis (wide) and 0 pixels to 2000 pixels on the y axis (tall). The portion of the texture file that is used along the y axis is determined by the Python script by using the real-world texture length specified in Row 4. UV maps are created that guarantee that seams of adjacent model faces start and end at the correct position of the texture file along the y axis of the image.

3.6 Road Metadata Creation

Roadway metadata enables turning surfaces that are simply 3D surfaces into road alignments that are understood by the driving simulators. Roadway metadata allows autonomous traffic to be displayed, and, in platforms such as the MiniSim, the metadata is required to allow subjects to navigate through the scenario, i.e., “drive the scenario.” The metadata provides information about the characteristics of the road such as width and lane definition as well as superelevation information. In the case of the MiniSim platform, the roadway metadata relies on a PATH file, while in the case of the RTI simulator, a PATH file can be used by relying on tools provided by the manufacturer. However, in the case of the RTI platform, the default option is the definition of a roadway surface as part of the VRML code that defines the scenario.

3.6.1 Process definition

The process of generating the necessary roadway metadata for a scenario relies on a Python script that receives two edge line definitions: one for the inner edge of the roadway surface and another for the outer edge of the roadway surface. The Python scripts go through each of the lines and compute the necessary parameters to create a road definition. Produced metadata is stored in an output file. The content of the output file can then be embedded into the desired scenario. A description of the script utilization and parameter definition is shown below and includes the parameters required for operation. The metadata output includes VRML code containing the definition of a road as well as a PATH file containing normal for the same road.

Table 3.2 – Roadway metadata generation: parameter structure.

```
python GenRoadMeta.py   param1    param2   param3

Where:
     param1 : file that defines the inner edge of the road
     param2 : file that defines the outer edge of the road
     param3 : name of the file where the roadway definition will be saved
```

3.6.2 Underlying theory

The roadway edge line files passed as parameters to the roadway metadata generation script include the points that define each edge line as shown in Figure 3.7. In the case of the edge lines passed to the GenRoadMeta.py script, these lines should be the ones that represent the driving surfaces and are not the same as for the GenRoadSurface.py. Instead, the lines are a subset of all edge lines that define the entire model since only two are used. Two different metadata outputs are produced by the script: first, PATH file data, and second, a VRML definition of a roadway used by RTI simulators.

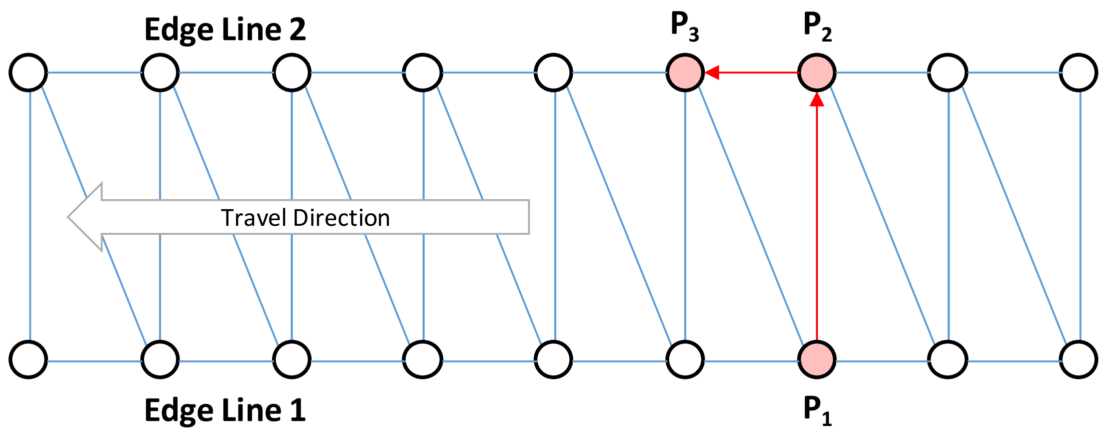


Figure 3.7 – Structure of vectors used to define PATH file.

To generate a PATH file, the normal vector along the roadway needs to be computed by taking into account the points that define each line. A normal vector is produced for each pair of points between Edge Line 1 and Edge Line 2. The produced vector relies on the pair of points between the lines (e.g., P_1 and P_2) and the next point in Edge Line 2 (e.g., P_3). Edge Line 2 is always assumed to represent the rightmost edge along the travel direction.

3.7 Implemental Considerations and Future Work

Work presented requires the use of the Python interpreter to execute the scripts and generate a desired 3D model compatible with driving simulation scenario authoring tools. And while the layer of abstraction presented can reduce the amount of time spent during the scenario creation process, there are areas where improvement is still possible. For example, the scripts presented can be implemented as a web service, thus eliminating the need for Python and thus further simplifying the process. Future work will look at implementing the tools described as web service that do not require the installation of software.

4 Conclusions

The creation of custom driving simulator scenario tiles is a process that is often required during the design of driving simulator experiments. This custom tile creation process is usually undertaken by researchers that are not necessarily trained in 3D modeling techniques and software. As a result, the time spent learning new pieces of software and the concepts associated with those can make the time for project delivery longer. When time available for the completion of a project is fixed, time spent learning 3D modeling concepts and software by researchers is time taken away from the stages of data collection (running subjects), interpretation of results, and proper experiment planning. Python scripts described in this report introduce a layer of abstraction into the custom tile creation process. The layer can eliminate or reduce the need for researchers to spend time in the labor-intensive texturing and 3D modeling process.

Scripts presented should be treated as an aid in the scenario creation process, not as a replacement. As discussed, the scripts can automate the process of creating roads that connect critical parts of the scenarios or parts that include intersections. Inputs required for the scripts include the coordinates of a line defining the centerline of a road, a file defining the cross section characteristics, and the desired texture for each component of the road. All input information required can be generated with software tools that are typically available to those in the engineering field. For example, CAD and spreadsheet programs can be used to generate the roadway geometry information, while an image editing program can be used to manage or create desired texture materials. Once implemented, the result of the scripts is a 3D model and supporting metadata that can be used to support the scenario creation process. The output can be used in multiple scenario creation tools provided by simulator manufacturers, thus creating opportunities for collaborative research.