Visualization Resources for Iowa State University and the Iowa DOT: An Automated Design Model to Simulator Converter

Final Report
November 2012

ctre
Center for Transportation Research and Education

THE NATIONAL ADVANCED DRIVING SIMULATOR

miniSim™

IOWA STATE UNIVERSITY
Institute for Transportation

Sponsored by
Iowa Department of Transportation
Federal Highway Administration
(InTrans Project 10-376)

About CTRE

The mission of the Center for Transportation Research and Education (CTRE) at Iowa State University is to develop and implement innovative methods, materials, and technologies for improving transportation efficiency, safety, and reliability while improving the learning environment of students, faculty, and staff in transportation-related fields.

About NADS

The National Advanced Driving Simulator (NADS) at the University of Iowa is the host, developer, and operator of the world's most advanced ground vehicle simulator. From its inception, the mission of NADS continues to be to help our research partners save lives and reduce the costs of vehicle crashes by better understanding the impact of technology, pharmaceuticals, and other factors on driving performance.

About the MiniSim™ Driving Simulator

The NADS MiniSim™ is high-performance driving simulator software designed for research, development, clinical, and training applications. The core software is based on state-of-the-art driving simulation technology that has been developed through decades of research.

The MiniSim™ core harnesses the technology found in the world's most advanced driving simulator, the NADS-1, into a smaller footprint at a lower cost. The MiniSim™ has been designed to be integrated with a variety of hardware and software components tailored to suit client requirements.

Disclaimer Notice

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. The opinions, findings and conclusions expressed in this publication are those of the authors and not necessarily those of the sponsors.

The sponsors assume no liability for the contents or use of the information contained in this document. This report does not constitute a standard, specification, or regulation.

The sponsors do not endorse products or manufacturers. Trademarks or manufacturers' names appear in this report only because they are considered essential to the objective of the document.

Non-Discrimination Statement

Iowa State University does not discriminate on the basis of race, color, age, religion, national origin, sexual orientation, gender identity, genetic information, sex, marital status, disability, or status as a U.S. veteran. Inquiries can be directed to the Director of Equal Opportunity and Compliance, 3280 Beardshear Hall, (515) 294-7612.

Iowa Department of Transportation Statements

Federal and state laws prohibit employment and/or public accommodation discrimination on the basis of age, color, creed, disability, gender identity, national origin, pregnancy, race, religion, sex, sexual orientation or veteran's status. If you believe you have been discriminated against, please contact the Iowa Civil Rights Commission at 800-457-4416 or the Iowa Department of Transportation affirmative action officer. If you need accommodations because of a disability to access the Iowa Department of Transportation's services, contact the agency's affirmative action officer at 800-262-0003.

The preparation of this report was financed in part through funds provided by the Iowa Department of Transportation through its "Second Revised Agreement for the Management of Research Conducted by Iowa State University for the Iowa Department of Transportation" and its amendments.

The opinions, findings, and conclusions expressed in this publication are those of the authors and not necessarily those of the Iowa Department of Transportation or the U.S. Department of Transportation Federal Highway Administration.

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.	
InTrans Project 10-376			
4. Title and Subtitle Visualization Resources for Iowa State University and the Iowa DOT: An Automated Design Model to Simulator Converter		**5. Report Date** November 2012 **6. Performing Organization Code**	
7. Author(s) Shawn Allen, Yefei He, and Vince Horosewski		**8. Performing Organization Report No.** InTrans Project 10-376	
9. Performing Organization Name and Address Center for Transportation Research and Education Iowa State University 2711 South Loop Drive, Suite 4700 Ames, IA 50010-8664		**10. Work Unit No. (TRAIS)** **11. Contract or Grant No.**	
12. Sponsoring Organization Name and Address Iowa Department of Transportation, 800 Lincoln Way, Ames, IA 50010 Federal Highway Administration, U.S. Department of Transportation, 1200 New Jersey Avenue SE, Washington, DC 20590		**13. Type of Report and Period Covered** Final Report **14. Sponsoring Agency Code** SPR 90-00-RB01-011	
15. Supplementary Notes Color pdfs of this and other InTrans research reports are available at www.intrans.iastate.edu/.			

16. Abstract

This project developed an automatic conversion software tool that takes input a from an Iowa Department of Transportation (DOT) MicroStation three-dimensional (3D) design file and converts it into a form that can be used by the University of Iowa's National Advanced Driving Simulator (NADS) MiniSim. Once imported into the simulator, the new roadway has the identical geometric design features as in the Iowa DOT design file.

The base roadway appears as a wireframe in the simulator software. Through additional software tools, textures and shading can be applied to the roadway surface and surrounding terrain to produce the visual appearance of an actual road.

This tool enables Iowa DOT engineers to work with the universities to create drivable versions of prospective roadway designs. By driving the designs in the simulator, problems can be identified early in the design process. The simulated drives can also be used for public outreach and human factors driving research.

17. Key Words	18. Distribution Statement		
driving simulator—file conversion—MiniSim—roadway design	No restrictions.		
19. Security Classification (of this report)	**20. Security Classification (of this page)**	**21. No. of Pages**	**22. Price**
Unclassified.	Unclassified.	44	NA

Form DOT F 1700.7 (8-72) **Reproduction of completed page authorized**

VISUALIZATION RESOURCES FOR IOWA STATE UNIVERSITY AND THE IOWA DOT: AN AUTOMATED DESIGN MODEL TO SIMULATOR CONVERTER

Final Report
November 2012

Principal Investigator
Omar Smadi
Center for Transportation Research and Education

Authors
Shawn Allen, Yefei He, and Vince Horosewski
The National Advanced Driving Simulator
The University of Iowa

Sponsored by
the Iowa Department of Transportation and
the Federal Highway Administration
State Planning and Research Funding
(SPR 90-00-RB01-011)

Preparation of this report was financed in part
through funds provided by the Iowa Department of Transportation
through its Research Management Agreement with the
Institute for Transportation
(InTrans Project 10-376)

A report from
Center for Transportation Research and Education
Institute for Transportation
Iowa State University
2711 South Loop Drive, Suite 4700
Ames, IA 50010-8664
Phone: 515-294-8103 Fax: 515-294-0467
www.intrans.iastate.edu

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

The authors would like to thank the Iowa Department of Transportation (DOT) for sponsoring this research and the Federal Highway Administration for state planning and research (SPR) funds used for this project. They would also like to thank Reginald Souleyrette, Jeremey Vortherms, Omar Smadi, Brandon Kimble, Brian Smith, and Sue Chrysler.

EXECUTIVE SUMMARY

In 2010, the National Advanced Driving Simulator (NADS) at the University of Iowa contracted with the Center for Transportation Research and Education (CTRE) at Iowa State University (ISU) to provide a portable driving simulator, the MiniSim.

The MiniSim uses a dual database system that includes visual models and computational support data necessary for operation and automatic traffic during simulation. The dual database system relies on simple geometry, texture imagery, and a simple model of the road surface and road network. This is a closed system that does not support importing more complex models, which are the basis for Iowa Department of Transportation (DOT) designs, currently, except through the intervention of a three-dimensional (3D) simulation expert who can modify the design files into a format supported by the simulator. This process is lengthy and therefore expensive, and is one-way only; changes to design files requires re-converting to the simulator form, often starting over from scratch since the data is not contained in a re-useable form.

An alternative method to manual conversion would be to identify the points of similarity between the DOT design file and simulator domains, and to develop a method to extract the required simulator data from the design file as automatically as possible. Once the data is extracted, it can then be used within the existing simulator workflow and tool set with a minimal amount of simulation expertise needed. The conversion process is one-way from design to simulator, but due to the automatic nature of the conversion, multiple iterations of designs are possible with minimal intervention or additional time, unlike the manual construction process.

The project reported here created such an automatic conversion software tool. It takes as input a 3D design file exported from MicroStation—the design package used by the Iowa DOT. Through a series of software steps, this design file is converted into a form that can be used by the MiniSim driving simulator. Once imported into the simulator, the new roadway has the identical geometric design features as in the Iowa DOT design file.

The base roadway appears as a wireframe in the simulator software. Through additional software tools, textures and shading can be applied to the roadway surface and surrounding terrain to produce the visual appearance of an actual road.

Results

This tool enables Iowa DOT engineers to work with the universities to create drivable versions of prospective roadway designs. By driving the designs in the simulator, design problems can be identified early in the process. The simulated drives can also be used for public outreach and human factors driving research.

A video showing the converter tool running and subsequent viewing in the driving simulator is available at www.youtube.com/watch?v=beWGea255s0&feature=youtu.be.

Future Recommendations

Although the conversion of a single, random Iowa DOT design file proved to be successful, some technical challenges remain to create a robust converter. Non-traditional designs, such as crossover lanes, dynamic lanes, and complicated interchange topologies, could prove challenging to the converter in its present form. More extensive testing and extension of the algorithm to include these additional test cases is required to provide a more comprehensive tool for automated conversion of highway designs to simulator-compatible form.

More specifically, additional development is needed to complete the following tasks:

- Create smooth transitions between road segments through junctions
- Increase algorithm efficiency to speed up data processing
- Extend the converter to work with additional simulator platforms, provided those systems use architecture that is easily accessible or well documented at the same level as the LRI/MiniSim architecture
- Generate surfaces with image textures instead of the model data present during conversion

BACKGROUND

Highway design decisions could be enhanced if designers, stakeholders, and the public were able to drive proposed designs before they are built. This converter project was designed to create a tool that facilitates the transfer of computer-aided roadway designs to interactive simulators.

The creation of three-dimensional (3D) drawings for proposed designs for construction, re-construction and rehabilitation activities are becoming increasingly common for highway designers, whether by department of transportation (DOT) employees or consulting engineers. However, technical challenges exist that prevent the use of these 3D drawings/models from being used as the basis of interactive simulation.

If designers could interact with their models as a driver would encounter them, certain design flaws and potential safety and operations problems could be identified early in the design process. In essence, one could drive the road before it is built. This capability would greatly enhance design decisions, enable meaningful community involvement, and provide a platform for evaluating safety performance.

During this project, the software engineers at the National Advanced Driving Simulator (NADS) at the University of Iowa developed a prototype file conversion tool in cooperation with the Iowa DOT and the Center for Transportation Research and Education (CTRE) at Iowa State University.

Use of driving simulation to service the needs of the transportation industry in the US lags behind Europe due to several factors, including lack of technical infrastructure at DOTs, cost of maintaining and supporting simulation infrastructure—traditionally done by simulation domain experts—and cost and effort to translate DOT domain data into the simulation domain. However, as costs continue to decrease and the extent of simulation training continues to prove cost-effective value for training and research, simulators have become more widespread and available for general-purpose use. What is required for simulators to become useful for DOT design departments and contractors is the means to exploit the simulation technology in ways that do not require extensive capital outlay or extensive staff training.

Transportation designers face significant challenges as increasing traffic congestion, aging infrastructure, and inadequate investments in innovation compete with increasingly inadequate revenues. Meanwhile, construction costs continue to soar. Innovative designs are being developed that hold the promise of reducing construction, maintenance, and operation costs while preserving safety.

Interactive driving simulation provides an ideal environment for human factors related research and safety evaluations on topics such as speed selection, lane selection, gap acceptance, sign comprehension and compliance, and work-zone driving performance. Driving simulation enables these kinds of safety evaluation before construction begins and provides a test bed to compare design alternatives to each other directly without lifting a shovel.

Historically, roadway design has utilized various forms of visualization during the design process. This could take the form of artist drawings of designs in context, clay models, and, more recently, computer-aided design (CAD). CAD programs, such as MicroStation and AutoCAD Civil3D, provide powerful tools to create engineering drawings for construction.

The engineer must make design decisions considering site-specific characteristics, local conventions, and design standards. Each of these decisions has ramifications for safety, operations, local acceptance, and aesthetics, as well as construction, operations, and maintenance costs. Currently, these drawings are wire-frame models with limited computer rendering of surfaces and textures in a 3D environment.

For visualization purposes, this rendering requires special software components beyond the standard drawing package and specialized training to create. Once rendered, these visualizations can be viewed passively like watching a movie. These animations are typically called a "fly through." While this type of visualization or animation can be quite powerful, it lacks interactivity or user performance measures. Once the movie is rendered, the user can't change the viewpoint or the path through the environment.

While lacking interactivity, visualization has still made great strides in affecting the roadway design process. A Transportation Research Board (TRB) committee and biennial conference exist on the topic. The group has identified several case studies highlighting the role of visualization in decision-making, public outreach, and project planning (*1*).

The visualizations have been used in a limited way to evaluate the safety of design and other software and analytical tools exist to evaluate the safety effects of design decisions. The main software tool in use is the Interactive Highway Safety Design Model (IHSDM), which was developed by the Federal Highway Administration (FHWA) (*2*). This decision-support tool makes predictions about a design's safety by checking it against known engineering standards.

However, the IHSDM does not include any visualization capability. The Roadway Safety Audit process is a checklist approach initially created for use during the design phase, though the process could be performed on virtual roadways using interactive driving simulation scenarios created from 3D roadway design files.

The process of converting transportation design files to simulation models is currently a predominately-manual one that is also scale-dependent, requiring months of effort to translate between the design software and the driving simulator software. The larger the project, the more time required to translate between the design and driving simulator. Changes to the source model require additional time to carry those modifications through to the simulator model. The use of a converter tool is independent of scale, automated with minimal interaction required during conversion, and does not require specialized training to use on the driving simulator beyond a working knowledge of fundamental procedures.

In a driving simulation, the roadways are not only objects to be viewed but also active agents in the software that contain information about rules of the road, such as allowable travel direction and lane width. This enables a user, using an actual steering wheel and pedals, to navigate freely within the environment while interacting with other traffic. Most driving simulators also allow different driver vehicle models (e.g., passenger car versus heavy truck) to be loaded, which change the driver eye height and the vehicle acceleration and braking characteristics.

Driving simulation has been used to evaluate traffic signs, pavement markings, signalized intersection operations, and shoulder width (*3*) (*4*). One example showing how simulation can be used before construction was a study of highway signs in the Boston, Massachusetts Central Artery/Tunnel Project (CA/T), known unofficially as the Big Dig. Researchers at the University of Massachusetts- Amherst created drivable sections of the tunnel in their driving simulator and tested driver reaction to different exit sign layouts (*5*). The results of this study were implemented in the tunnel and are in place today.

Another good example of using driving simulators before construction is provided by the FHWA evaluation of the diverging diamond interchange in cooperation with the Missouri DOT (MoDOT) (*6*). Driver behavior studies conducted in the FHWA simulator resulted in changes to initial alignment to improve sight distance and also in changes in traffic control devices. The creation of the simulator world from the roadway design files was a painstaking manual process taking up to six months. Revisions to the design required extensive re-coding of the simulator program. The converter tool could reduce this process to a matter of hours with revisions handled quickly.

Currently, several large consulting firms and transportation agencies are using interactive driving simulators for public education and outreach. Parsons Brinkerhof created a driving simulator for San Francisco, California's Presidio Parkway redesign and allows the public to visit their office to drive the future roadway to garner support and patience throughout the construction project (*7*). This simulation required extensive programming and is a one-of-a-kind creation for the project.

The Michigan DOT (MDOT) created an "edutaiment" driving simulator to showcase the advantages of Connected Vehicles. This simulator was used at fairs and expos to engage the public in the future of transportation. The simulator was built by a gaming company over four months and was not based on actual roadway designs (*8*). The converter tool can leverage the DOT labor already invested in creating roadway design files and eliminates the need to re-draw these roadways in the simulator software.

INTRODUCTION

NADS is home to one of the three largest motion base simulators in the world, and the largest full motion simulator in the US—the NADS-1. The technologies used to support core simulator architecture such as the visual models, supporting terrain data, behaviors to control scene objects, and authoring of events that occur during simulation, called scenarios, have been developed by the University of Iowa over the past 30 years.

Following the creation of the NADS facility, it was clear that, to advance the fields of driving simulation research and technology significantly, it would be necessary to make driving simulation accessible to a larger audience of researchers, trainers, and practitioners than was practical with the simulators housed at the facility. This was the impetus for creating the MiniSim for CTRE at Iowa State University (Figure 1).

Figure 1. Simulator trailer (left) and view of simulator inside the trailer (right)

The MiniSim uses many of the core simulator technologies developed for the NADS-1 and was created to be licensed for research and training purposes. The MiniSim developed for CTRE is a three-quarter cab simulator consisting of a vehicle seat and steering wheel mounted on a fixed platform with three visual channels and a glass dash instrument panel, which is customized to each driving cab type chosen. The simulator is mounted within the trailer that can be used anywhere with electrical power to it.

The simulator also includes the tool set needed to create custom environments and scenario events using the dataset provided with the simulator (Figure 2). These tools are in use and under active development and support by NADS engineering staff. The tools have been used to support human-in-the-loop driving simulation research successfully since 1997.

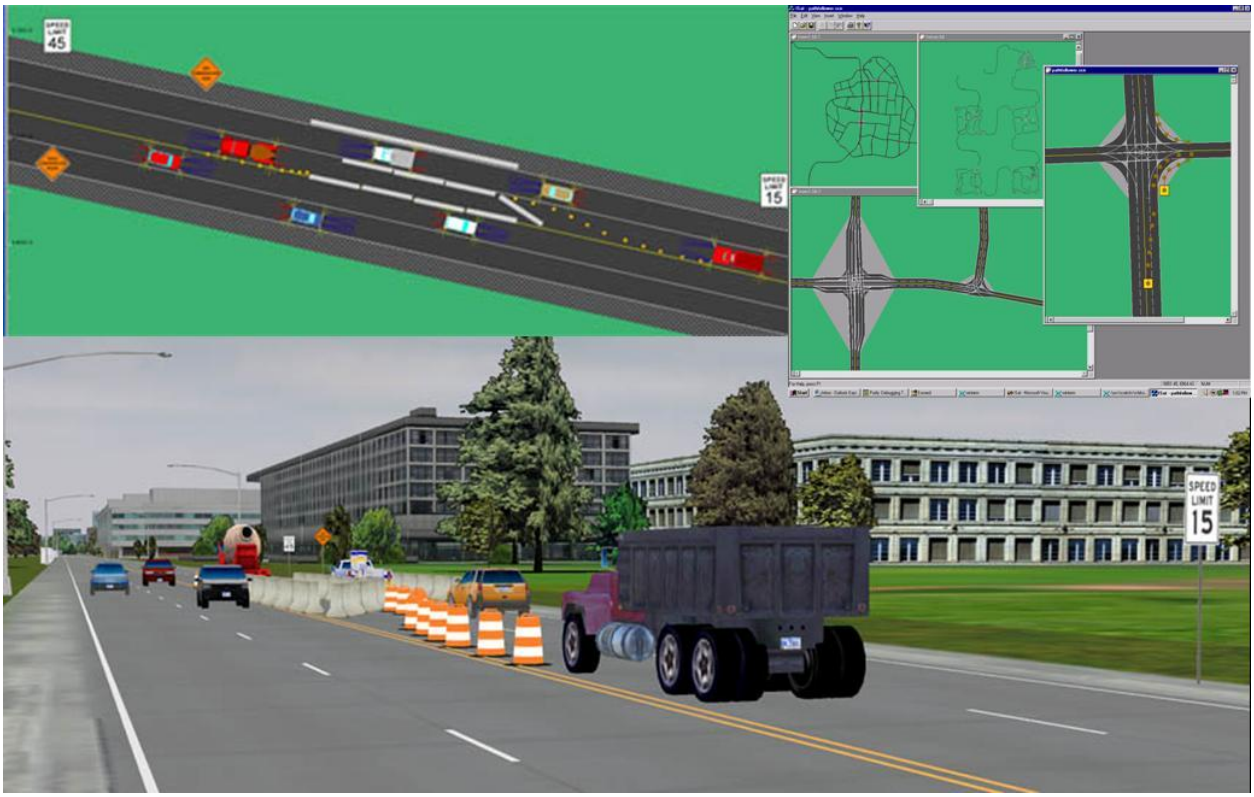


Figure 2. ISAT scenario defined (upper left), various road network environment samples within ISAT (upper right), and resulting simulator scene view from driver eye point (bottom)

The MiniSim licensed to CTRE afforded an opportunity to investigate how this technology might be used by Iowa DOT practitioners in their pursuit of safety and transportation design. The tools used to create 3D models of roadways, terrain, and features in the simulator are relatively common. What is needed for simulation in addition to the visual model is corresponding computational data, known as correlated data. A visual road model consists of geometry and textured imagery to create the necessary visual appearance, but it is the correlated data that permits simulation.

The simulator data is typically generated simultaneously as the geometry model is developed, and is extracted from the source files in such a way to ensure that the computational and visual models are perfectly synchronized, or correlated, to each other. Failure to correlate properly results in errors on the simulator such as driving above or below a surface, or computer-generated traffic being unable to traverse the scene correctly, which is undesirable in the research world.

NADS uses Presagis Creator for the majority of visual terrain model development due to the road tools that support the creation of simulator roadway data. This data is in the form of a road centerline, and is created one segment at a time (Figure 4). Where complex road surfaces or networks are required, compound segments or complex road network elements (roads and

intersection junctions) are created manually (Figure 4). Figure 4 (right) shows views from the graphical Interactive Scenario Authoring Tool (ISAT).

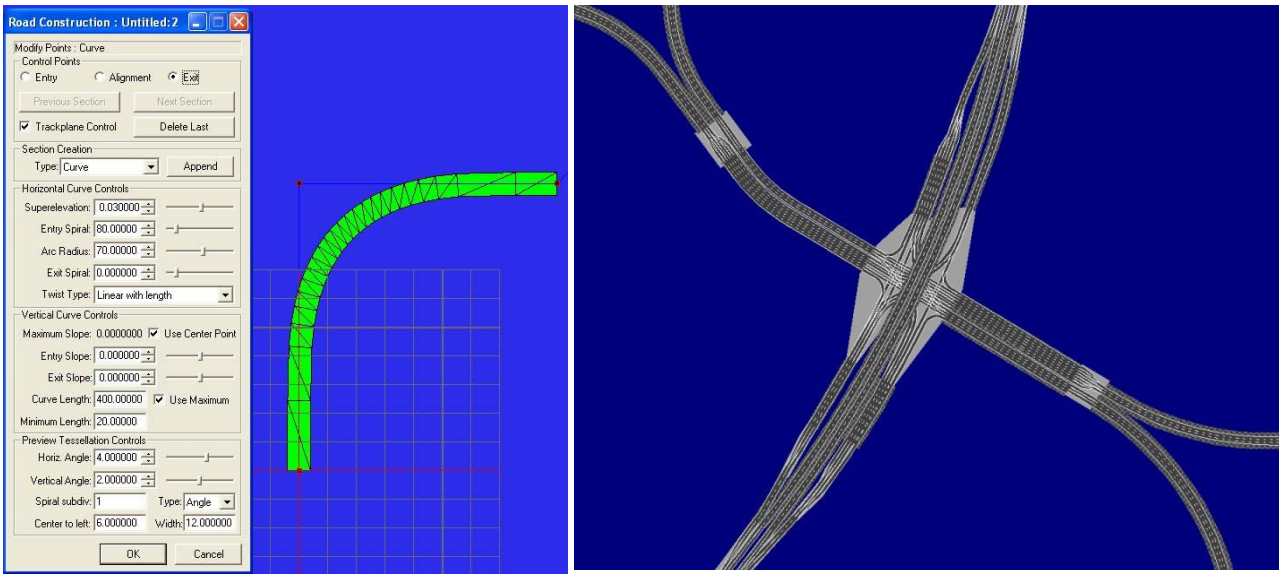


Figure 3. Individual road curve segment (left) and compound road network constructed from multiple segments (right)

There is no concept in Creator of a higher-order structure than the individual road segment, which is a curve, hill, or straight segment. Each road segment retains only the basic parameters that were used to generate it. These road segments are created and then are "baked" into final form by tessellating the road. The settings used for tessellation are not retained, and therefore rely heavily on the conventions and work methods of the individual 3D modeler for consistency.

The road centerline data is generated in the form of centerline data, which is a sorted XYZ ASCII file, containing vertex normal heading-pitch-roll (HPR) data for each centerline point. This centerline data is augmented by additional simulator data libraries, which contain definitions for various road attributes: lateral roadway slices known as lateral profiles, surface materials (e.g., asphalt, terrain, gravel), and road network connectivity.

These collective data files are then used when a terrain database is built and published as a collective model (Figure 4). The individual data files are merged into a collective terrain database that describes the road network for any particular configuration of scene elements. At this point, the terrain database is considered a complete entity and, to add or subtract from it, it is necessary to revert back to the original form of the model, adjust that as needed, and re-publish in the modified form to make changes.

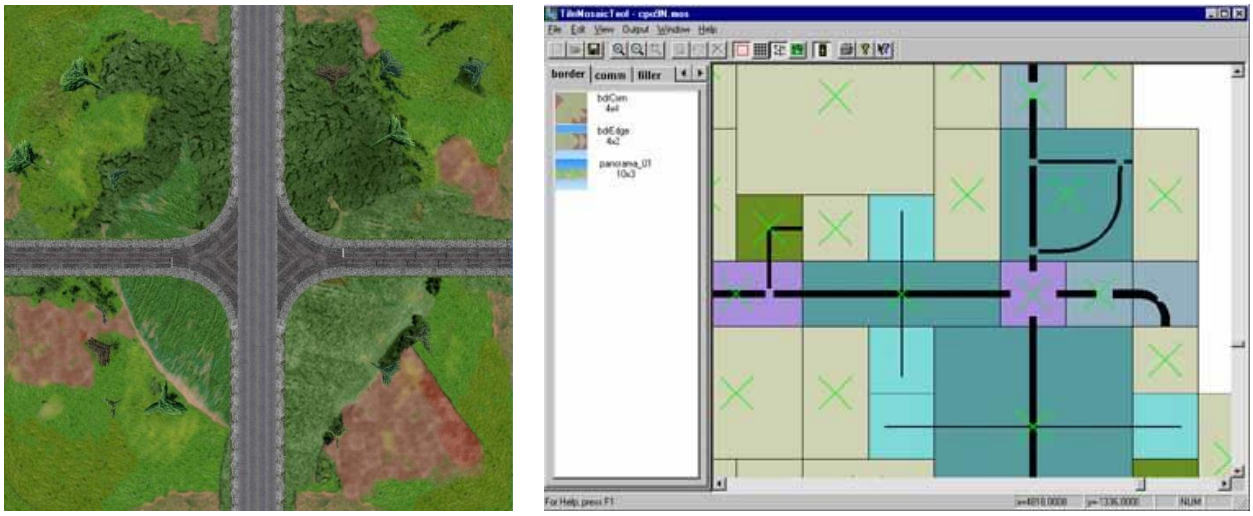


Figure 4. Individual tile model (left) and multiple tiles arranged for a single terrain model (right)

The 3D modeler is responsible for attaching non-visual attributes to the road and visual models. For each road model, it is necessary to identify the minimum characteristics required to support the existing technology. This currently includes marking where each road begins and identifying the road by an alpha-numeric ID, the associated data file that contains centerline data for the road, the lateral profiles that apply to each road segment, the number and widths of each lane, speed limits, passing rules, and intersection junctions that begin and end each road segment.

With this information, it is possible to combine multiple road segments into a complex road network and use that published data in the visual and correlated data to support driving simulation. However, it is still necessary to define what events occur within the simulated environment during simulation. These events are created by authoring them using correlated data using the ISAT. The ISAT reads the correlated data and presents the scenario author with the vehicles, traffic control options, and capability to generate complex chains of events in the simulator, known as scenarios.

To support consolidation of multiple road models into a single terrain model (Figure 4), some conventions are required to ensure the software can manage the consolidation through model attributes. These additional attributes include use of standardized sizes, standard edge definitions for models, unit type, axis orientation and standard location with respect to the local 0,0,0 origin point. The lack of these additional attributes within the Iowa DOT design domain requires each conversion to be implemented as a terrain model in entirety and not as part of a reusable library.

Given the simulator models require only a subset of the data necessary for transportation designs, extracting a portion of data from the Iowa DOT designs, as shown in Table 1, seemed to be an ideal and simple approach.

Table 1. Characteristics of the Iowa DOT design files and the simulator models

	DOT Design file	Simulator Attribute
Visual Model	X	X
Road centerline	X	X
Lateral profile	X	X
Lanes	X	X
Junctions	X	X
Pavement surface	X	X
Surface type	X	X
Shoulders, ROW	X	
Grading (fill, granular, special, side slopes, pavement bottom, earth fill)	X	
Underlayment	X	

METHODOLOGY

The initial approach was intended to utilize generic network development algorithms to extract roadway data from the Iowa DOT design files and then transform the data into a form compatible with the NADS MiniSim. Compatibility with the existing database production workflow was accomplished by following the conventions defined in the NADS Logical Road Interface (LRI) for intermediate data files (see Appendix A).

As long as design data can be mapped onto existing resource types, the tools used to publish simulator models could be used to generate finished simulator models from the design files. The source design was provided in the form of MicroStation DWG, TIN, and terrain project files. The design file is a highway diamond-style interchange with grade-separated junctions (Figure 5).

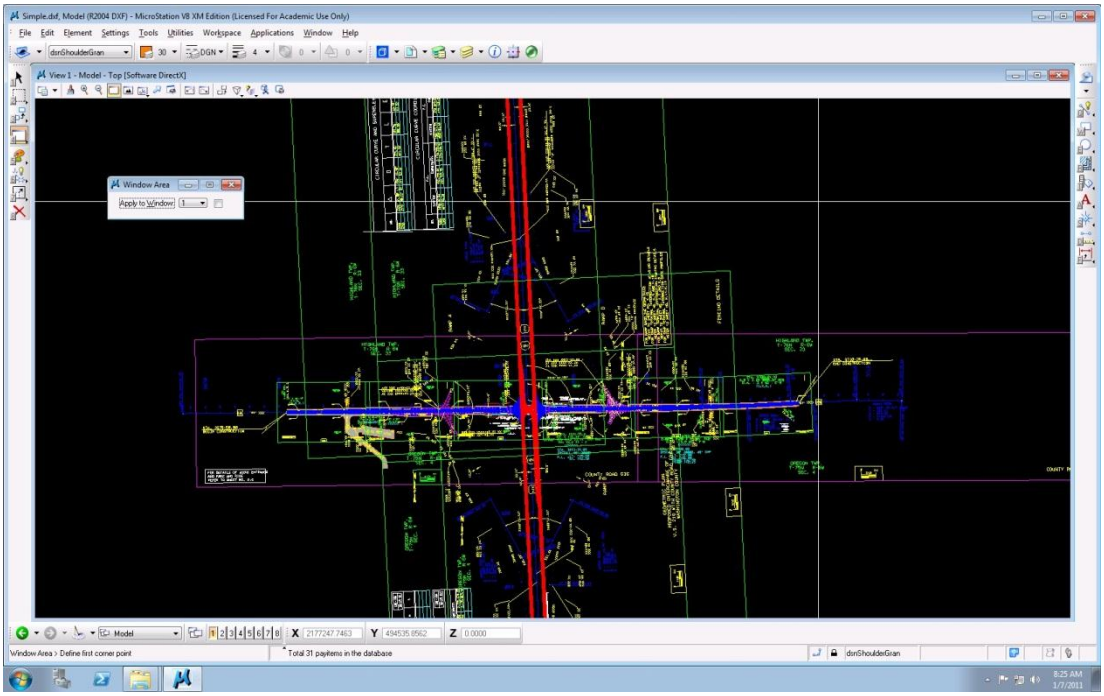


Figure 5. MicroStation view of sample interchange model data

Due to the apparent complexity of the source design file, it proved necessary to convert the source data into a form that could be read by the NADS standard 3D package, Creator, to investigate it in detail. This proved to be difficult, as the DXF data exported from MicroStation would crash Creator or import with significant anomalies that were not visible from the native MicroStation environment or with pieces of the source file missing.

Eventually, it was discovered that MicroStation uses themes for exporting data and, after some investigation, a theme was used to export Creator-compatible data in a form that resolved the anomalies. Unfortunately this path did not correct the inability of Creator to import the data, as

pieces of the design file were still missing from the import. Some other method of accessing the data was required.

During a trial of the software Rhino 3D, we discovered that Rhino would read the MicroStation DWG and could display all layers and all elements present within the source design file (Figure 6).

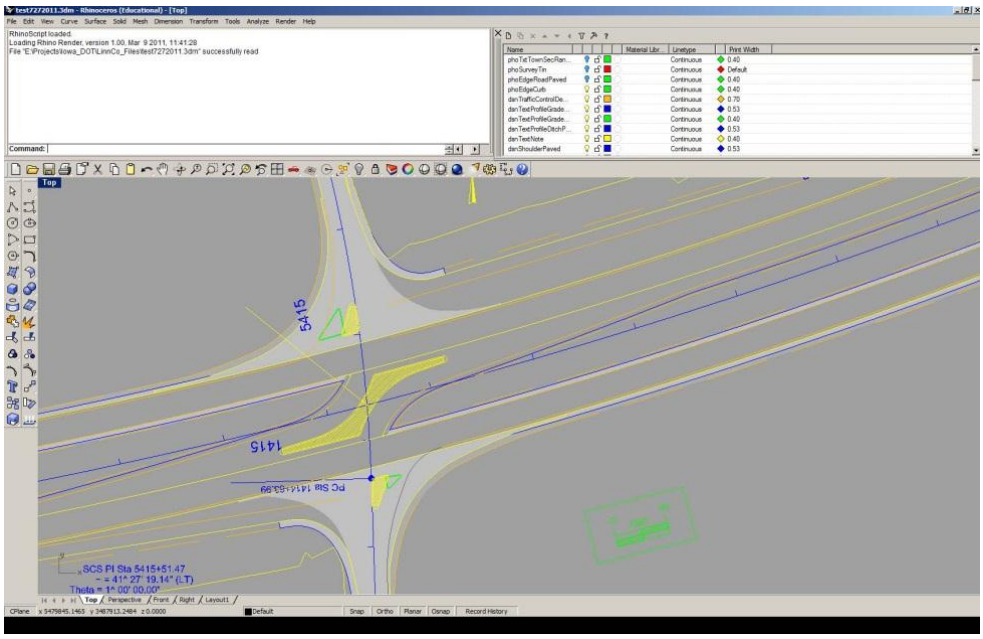


Figure 6. Rhino 3D view of sample design file with all source data retained

NADS then investigated the layer conventions used in the source MicroStation design file. Although the Iowa DOT confirmed that layer conventions were part of the design process, we discovered some layers that did not seem to fit within the normal conventions. Some data from a specific layer would be found on a different layer, and some data was actually missing from the design file altogether. It was unclear on visual inspection whether this missing or anomalous data would affect the development of algorithms to extract simulator data.

This finding led to a fundamental shift in the project from developing an algorithm to extract data. The decision was made not to rely on automated algorithms, but on a human observer to identify congruent data and to identify the data necessary for simulation (road centerlines and intersection junctions). This process is manually intensive but relatively simple; visual inspection of the design file shows complex road/junction relationships that are evident to a human observer (Figure 7).

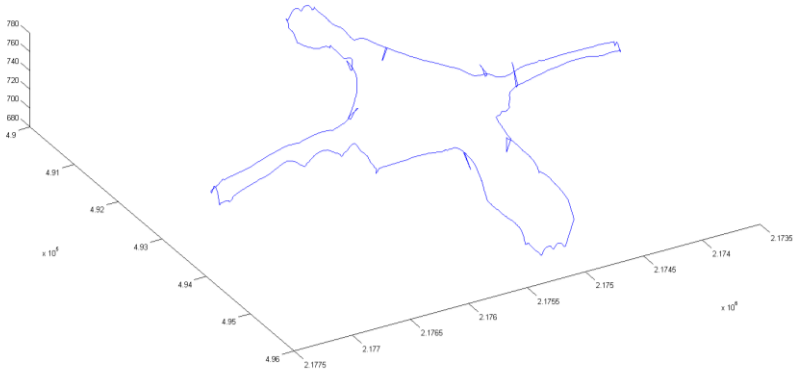


Figure 7. Boundary data "spike" anomalies present in the source data

Significant resources were expended before it became clear that, despite re-use of existing technology to develop the user interface, re-modeling the road network and connectivity was not cost-effective nor a desired outcome by the Iowa DOT. As it was expressed to the project team, manual interaction needed to be kept to an absolute minimum, although it was agreed that minor interventions—such as inserting simulator-specific attributes—would be acceptable (Figure 8).

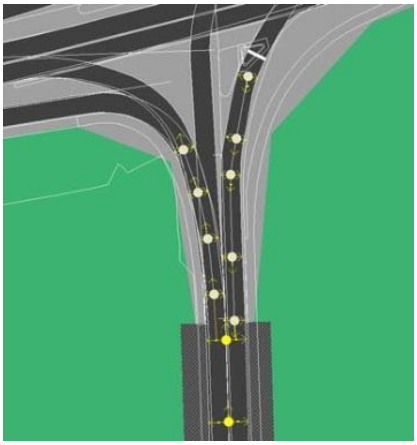


Figure 8. Manual insertion of corridor nodes to create road-to-road connections, tool view

At this point, a re-assessment of the project was required. Was it possible to access the design data in another form other than the visual-oriented layers of the source DWG/DXF? Once access was determined, would it be possible to extract that data? Finally, would it be possible to convert the extracted data to a form compatible with the simulator workflow such that creation of a drivable simulation using design data would require minimum user interaction?

An assessment of possible data extraction was made using Civil 3D, which is an industry competitor to MicroStation and the road design package for the University of Iowa College of Engineering. It appeared that both Civil 3D and MicroStation could export data in extensible markup language (XML). XML is a markup language that defines a set of rules for encoding documents in a format that is both human- and machine-readable.

Initially, it seemed that MicroStation XML was not a complete description of the design and would therefore prove unworkable. However, in the case of the design file provided, that proved not to be the case and all data necessary was present within the XML file after consultation with the Iowa DOT and transmission of a second XML dataset.

The conversion process involves several data processing stages as illustrated in Figure 9 and described below.

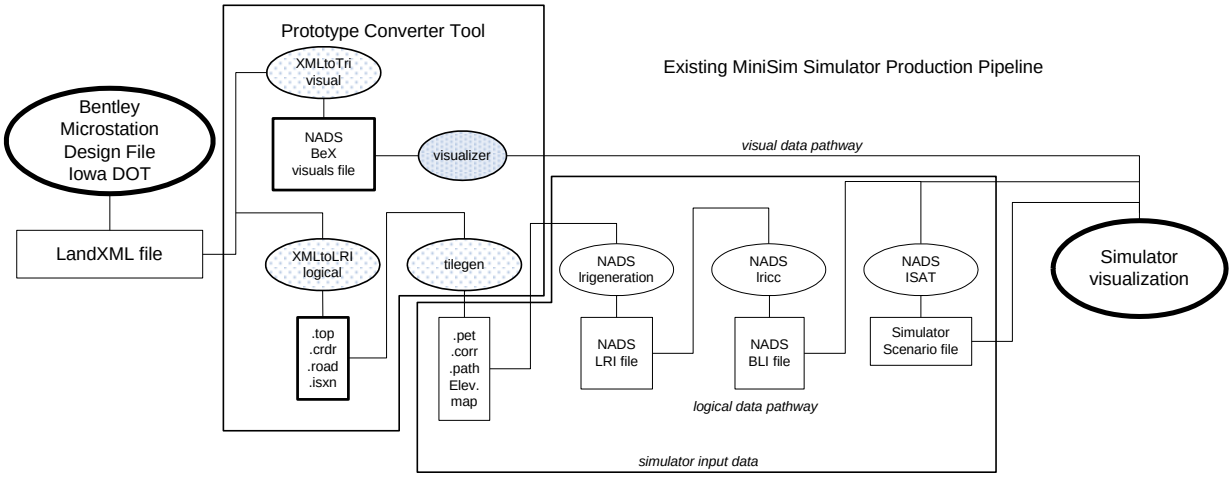


Figure 9. Converter processing and simulator workflow

XMLtoTRI: Convert XML source to the proprietary NADS Basic geometric EXtract (BEX) file format

XMLtoTRI pulls information from the input XML files and reformats that into a form that the MiniSim can interpret, namely BEX format. This involves finding the pertinent information within the XML file and stripping off the XML tags from it. In addition, XMLtoTRI writes a short header into each output file. No further analysis is done. Three categories of data may be extracted: TIN, boundary, and breaklines.

TIN data defines the triangulated surface geometry of the road design. Boundary data defines the boundary lines of the design. Breaklines are lateral profile curves obtained from slicing the road geometry perpendicular to the centerline at fixed intervals and then connecting the matching points between consecutive lateral profile curves. These three categories of data when combined together are able to preserve the full surface geometry contained in the original design data, as well as road-aligned grid lines that provide excellent visual cues for lane definition and direction of travel, given the absence of surface textures.

Using the BEX files as the source for real-time visualization in the NADS simulator, the visual appearance of the road structure in the simulator is almost identical to that in the design software such as MicroStation or Civil 3D.

XMLtoTRI Inputs:

XMLtoTRI draws data from particular points within a LandXML file, described below in XPath format (see Table 2). To explain, XML documents are organized hierarchically. Data is stored in nodes, and given XML nodes can have children just like directories on a hard disk, XPath syntax resembles traditional file system paths, such as C:\Windows\System32\config. However, as there is only one root node, the drive letter is unnecessary.

In addition, attributes, which further describe nodes, are indicated with the @ symbol. Finally, XPath provides some pattern matching capability, which is typically offset with square brackets, so Definition[@surfType = 'TIN'], for example, means find the Definition node with a surfType attribute that has a value of TIN.

Table 2. XMLtoTRI conversion using XPath

TIN	
Vertices	/LandXML/Surfaces/Surface/Definition[@surfType = 'TIN']/Pnts
Connectivity	/LandXML/Surfaces/Surface/Definition[@surfType = 'TIN']/Faces

Boundary	
Surface name	/LandXML/Surfaces/Surface/@name
Boundary name	/LandXML/Surfaces/Surface/SourceData/Boundaries/Boundary/@name
Vertices	/LandXML/Surfaces/Surface/SourceData/Boundaries/Boundary/PntList3D

Breaklines	
Breakline name	/LandXML/Surfaces/Surface/SourceData/Breaklines/Breakline/@name
Vertices	/LandXML/Surfaces/Surface/SourceData/Breaklines/Breakline/PntList3D
Style	/LandXML/Surfaces/Surface/SourceData/Breaklines/Breakline/Feature/Property[@label = 'style']

Note: All child nodes of /LandXML/Surfaces/Surface/SourceData/Breaklines are processed, and there will typically be many of these.

XMLtoLRI: Convert XML source to NADS MiniSim LRI file format

The chief difficulty in converting design files into logical descriptions usable by the MiniSim lies in divining how various roads join together. While this is a trivial task for a human being, it must be remembered that computers work from large stores of essentially undifferentiated numbers and, as such, must be instructed how to proceed in great detail.

As the MiniSim simulator software entails specialized definitions of “road” and “intersection,” which do not necessarily match those in general Iowa DOT use, the word “route” is used to describe an un-analyzed segment of pavement derived from LandXML files. A route will,

therefore, after processing, typically give rise to both roads and intersections, which can then be driven on the MiniSim.

According to the MiniSim logical definition, a road is a section of drivable pavement that does not vary in width and does not intersect with any other roads. In the same context, an intersection connects two or more roads together and can be of irregular shape. Finally, a corridor is a path through an intersection that connects two lanes on different roads and is used principally to help computer-generated traffic negotiate intersections.

Initially, XMLtoLRI determines which routes correspond to which centerline alignments, as these data come from different sources and are not guaranteed to be named in a consistent manner or otherwise linked within the design file set. This processing is necessary because, while roads and intersections fundamentally derive from routes, XMLtoLRI must analyze the centerlines to find connectivity between routes in certain cases. Finding matching routes and centerlines is accomplished by calculating the distance between pairs and then selecting the pair having the minimum distance.

Next, XMLtoLRI traverses each route looking for bulges or constrictions. These areas indicate intersections; whereas, the consistent sections indicate roads. In this manner, a candidate set of roads and intersections is constructed, along with a subset of the total connectivity between them. In certain cases, several routes will fall along a single centerline. This presents an easy opportunity to infer connections between these routes as they must necessarily be linked head to tail.

Although some connectivity will have been detected by this point, some portion will have defied analysis also. To find all missing connectivity, XMLtoLRI consults the centerlines extracted earlier and, once again, measures distances from one to another looking for places where the nearness falls below a certain threshold. These instances indicate intersections, which may or may not already be present. When XMLtoLRI finds that an additional intersection is needed, it creates one and links it appropriately. Similarly, XMLtoLRI can combine intersections that are found to overlap. At the conclusion of this step, XMLtoLRI will have produced a complete set of roads, intersections, and their linkages that logically represents the initial design.

At this point, XMLtoLRI fleshes out the logical description to include required ancillary simulator attributes. First, however, the directionality of traffic flow must be defined. XMLtoLRI does this largely by assuming the rule that drivers stay right within their own frame of reference. In the case of single-lane roads, XMLtoLRI looks at the general flow of traffic at adjoining intersections to decide between options, specifically whether driving onto the road in question from a certain direction will require an extremely hard turn.

This done, each lane within each road is designated with a code identifying the direction of traffic flow upon it as well as a width. With this information, XMLtoLRI can now generate corridors. Within each intersection, each lane is considered in terms of where it might lead. XMLtoLRI relies on a basic set of rules to answer this question. In particular, only right-most

lanes can turn right, only left-most lanes can turn left, and there can only be as many straight-through corridors as the smallest number of lanes in corresponding roads.

Currently, XMLtoLRI draws a straight-line linear path from incoming lane to outgoing lane, but it is envisioned that a smoothed spline will replace this in later revisions. The use of linear corridors has no impact on the drivability of an intersection by the simulator driver.

In the last step, XMLtoLRI analyzes the geometry found within the TIN section of the design XML files to generate a grid-post data structure akin to a contour map for elevation data of each of the intersections within the logical description. The MiniSim uses these to determine the height of the pavement throughout an intersection. This analysis proceeds in several steps.

First, all triangles are scanned to determine if they fall within the bounds of any intersection. All irrelevant triangles are discarded at this point. Then, for each intersection, XMLtoLRI considers its size and complexity to determine an adequate resolution granularity to be used during the scanning performed in the next step. Scanning proceeds sequentially along axis-aligned columns, which subdivide the intersection. At each column index, a "cutting" plane is constructed and the constituent triangles are examined for the intersection. This yields a two-dimensional height map, which, after a change of coordinate basis, can be scanned to provide an elevation reading at any given point. This processing step generates an output file that contains a matrix of elevation data.

Table 3 provides a summary of XMLtoLRI inputs.

Table 3. XMLtoLRI inputs

Alignments	
Alignment	/LandXML/Alignments/Alignment
Name	/LandXML/Alignments/Alignment/@name
Length	/LandXML/Alignments/Alignment/@length
Start station	/LandXML/Alignments/Alignment/@staStart
Coordinate geometry	/LandXML/Alignments/Alignment/CoordGeom
Lines	/LandXML/Alignments/Alignment/CoordGeom/Line
Curves	/LandXML/Alignments/Alignment/CoordGeom/Curve
Elevation profile	/LandXML/Alignments/Alignment/CoordGeom/Profile/ProfAlign[@name = '{alignment_name}_P']

Breaklines	
Name	/LandXML/Surfaces/Surface/SourceData/Breaklines/Breakline/@name
Route edges	/LandXML/Surfaces/Surface/SourceData/Breaklines/Breakline[@name = '{route_name}-R_EOP']
Lane edges	/LandXML/Surfaces/Surface/SourceData/Breaklines/Breakline[@name = '{route_name}-R_LN{lane_index}']

Notes: The XPath for the right boundary of the breakline is given; the left boundary is obtained similarly. Also, the XPath for route edges can vary somewhat. In the case of divided highways, R_EOPM and R_EOP form the boundaries of the rightmost route and L_EOP and L_EOPM form the boundaries of the leftmost route. In addition, certain irregular road segments are defined by L-L_EOP and R-R_EOP or some variation on this rule.

TIN	
Vertices	/LandXML/Surfaces/Surface/Definition[@surfType = 'TIN']/Pnts
Connectivity	/LandXML/Surfaces/Surface/Definition[@surfType = 'TIN']/Faces

XMLtoLRI Outputs:

The road and intersection information that is extracted from the LandXML files is organized into intermediate data files in text format that are easy to read and edit as follows:

- Overall layout (.top) file, which lists the roads, intersections, and intersection corridors
- Road definition (.road) files with centerline data and lane definition
- Intersection definition (.isxn) files with intersection boundary definition, road-to-corridor connectivity information, and elevation and material map of the area bound by the intersection
- Intersection corridor definition (.crdr) files with corridor curve data

TILEGEN: Convert intermediate files output from the XMLtoLRI process into NADS tile (road segment) definition files

These intermediate files match well in form and functionality with the NADS tile definition files. Therefore, the conversion process is very straightforward. Included in the tile definition files are the following:

- Top level layout (.pet) file, which includes road and intersection information, lateral profiles, lane definitions, as well as objects placed on the road and intersection such as traffic signs, traffic lights, and other statically-placed entities
- Road path (.path) files that contain the centerline data
- Intersection corridor (.corr) files that contain the corridor curve data
- Intersection elevation map (.map) file that contains the elevation and material maps of the surface areas covered by intersections

The output files of TILEGEN conform to the NADS tile definition file format. They are fully compatible with existing NADS tools to generate lri and binary bli files that are used by ISAT and NADS simulators. In the future, they can be further standardized, after which they can be added to the tile definition library and combined with other existing tiles to construct larger, more-complex road networks.

CONCLUSIONS

The end result of this project was a converter process that can transform the sample Iowa DOT interchange design file into MiniSim-compatible data, which was then integrated into the simulator. This process was validated by using the sample data to build a simulator scenario, which included computer-controlled traffic, and then driving on the resulting model. A video showing the converter tool running and subsequent viewing in the driving simulator is available (*9*).

RECOMMENDATIONS FOR FUTURE DEVELOPMENT

Although the conversion of a single, random Iowa DOT design file proved to be successful, some technical challenges remain to create a robust converter. Non-traditional designs, such as crossover lanes, dynamic lanes, and complicated interchange topologies, could prove challenging to the converter in its present form. More extensive testing and extension of the algorithm to include these additional test cases is required to provide a more comprehensive tool for automated conversion of highway designs to simulator-compatible form.

Additional development is needed to create smooth transitions between road segments through junctions. The current tool is capable of associating roads and corridors but the corridors are linear and excessively coarse in comparison to the standard, manually-generated intersection corridor (Figure 10).

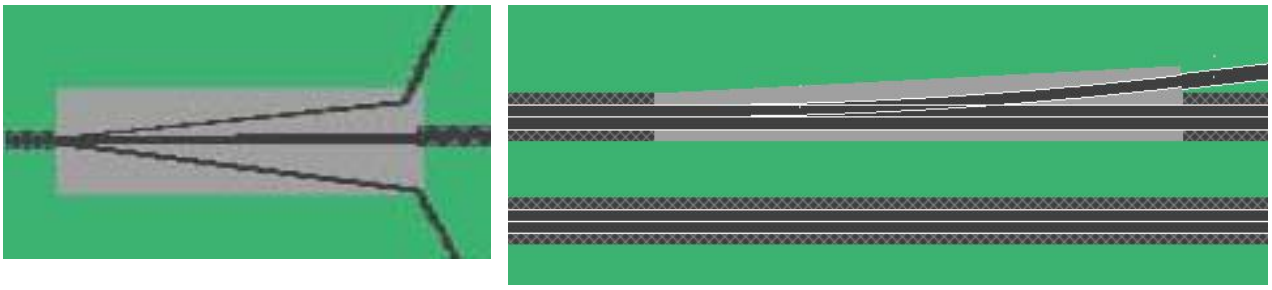


Figure 10. Linear corridor from converter (left) and smooth corridor from manual construction (right)

The tile (road segment) definition files generated by the converter, although perfectly usable for building a simulator database on its own, is not standardized with the library of tile models on edge definitions. Therefore, the resulting road segment tiles cannot be connected with those pre-existing tiles to create a larger driving environment than the source file defines. Additional development is needed to be able to automatically standardize the tile and make it compatible with existing road segments.

Additional development is needed to increase algorithm efficiency to speed up data processing. The test sample file requires 24 minutes to process, and a significant increase in speed could be anticipated by algorithm improvement such as parallel processing.

Additional development is needed to extend the converter to work with additional simulator platforms, provided those systems use architecture that is easily accessible or well documented at the same level as the LRI/MiniSim architecture. This would provide benefit to other DOTs that have already invested in driving simulators and would like to leverage the benefits of automating transportation design models to simulation models.

Additional development is needed to generate surfaces with image textures instead of the model data present during conversion. Additional user interface options would be required to facilitate texture application, which is typically accomplished through associating image picture elements (pixels) with geometry through UV mapping the image onto the geometry within a 3D modeling package.

REFERENCES

1. TRB Committee on Visualization http://www.trbvis.org/MAIN/CASE_STUDIES.html
2. FHWA Interactive Highway Safety Design Model http://www.fhwa.dot.gov/research/tfhrc/projects/safety/comprehensive/ihsdm/
3. Chrysler and Nelson (2011). Ch. 36. Design and Evaluation of Signs and Pavement Markings Using Driving Simulators. Fisher, D. L., Rizzo, M., Caird, J. K., and Lee, J. D. (Eds). *Handbook of Driving Simulation for Engineering, Medicine, and Psychology*. Boca Raton, FL: CRC Press.
4. Kelly, Lassacher, and Shipstead (2007). *A High-Fidelity Driving Simulator as a Tool for Design and Evaluation of Highway Infrastructure Upgrades*. Montana State University Technical Report. http://www.mdt.mt.gov/other/research/external/docs/research_proj/high_fidelity/final_report.pdf
5. Upchurch, Fisher, Carpenter, and Dutta (2002) "Freeway Guide Sign Design with Driving Simulator for Central Artery-Tunnel Boston, Massachusetts." *Transportation Research Record*. No. 1801. Transportation Research Board, Washington DC. http://trb.metapress.com/content/880h534617x10755/fulltext.pdf
6. Bared, Edara, and Jagannathan (2005). "Design and operational Performance of Double Crossover Intersection and Diverging Diamond Interchange." *Transportation Research Record*. No. 1912. Transportation Research Board, Washington, DC.
7. Presidio Parkway Project. Accessed April 15, 2012. http://www.presidioparkway.org/press_center/articles/presidio_parkway_project_debuts_20678.aspx
8. Epic Games Intellidrive Interactive. Accessed April 15, 2012. http://forums.epicgames.com/threads/727296-Intellidrive-Interactive-Triple-Monitor-Driving-Sim-Now-playing-in-San-Francisco!?s=2906b8477d26069818e7020990242e64
9. He, Y., Horosewski, V., Chrysler, S., and Allen, S. (2012). Demonstration video for automatic design model to simulator converter. http://www.youtube.com/watch?v=beWGea255s0&feature=youtu.be

APPENDIX A – NADS LOGICAL ROAD INTERCHANGE (LRI) INFORMATION

This appendix contains a detailed description of LRI information.

LRI Introduction

This appendix contains the specifications for the LRI file. The specification includes the layout of the file, presented as BNF grammar, and the semantics of the various parts of the file. The LRI file defines all the physical aspects of the virtual environment database and in addition, presents selectable options regarding the location and/or visual appearance of objects. The LRI file is tightly coupled to the visual database, and as such, reflects information that is fixed by choices made by the 3D modeler/terrain model designer.

LRI Conventions

The BNF grammar and associated semantics for the LRI file use the following conventions:

Items in **bold** are terminal symbols and appear verbatim in the file.

Items in `plain courier` are non-terminal rules that refer to simple identifiers or numbers.

Items enclosed in angle rackets such as `<rule3>` are non-terminals and refer to a separate rule in the grammar.

Items inside braces, i.e., `{ name }` must appear one or more times

Items inside brackets, i.e., `[ name ]` must appear zero or more times.

The vertical bar | is used to indicate alternatives so that `a | b` means either a or b but not both.

The following non-terminals are pre-defined and not shown explicitly:
string: a plain string with no white space, uses convention similar to C, max length is 31 chars
float: a floating point number using the XX.YYY format or exponential notation
integer: an integer represented in decimal or hex notation using C conventions (i.e., 17 = 0x11)
EOL: an end of line character

Example:

Consider the following rules:

<obj_list> = <obj_type> { string }

<obj_type> = **SIGN** | **CAR** | **TRUCK**

The following is a valid input string for this grammar:

SIGN stop1 stop2 stop3
CAR car1
TRUCK trailer1 large_truck

The following is not a valid input string for this grammar:

CAR1 car1 car2
TRUCK
SIGN TRUCK
SIGN This_sign_name_is_more_than_32_chars_in_length_thus_invalid

LRI Overall Layout

The file consists of four major sections: the header, the road definitions, the intersection definitions, and the objects. An optional attribute dictionary can appear at the end of the file. White space can appear between any lines and C style comment can be used anywhere with the exception of sections within the header that define long messages.

LRI Header

The LRI header contains a few pieces of information that apply to the whole LRI file. The following lists the format.

<header> = **HEADER {** [{ header_items }] **}**

<header_items> = **zdown =** <integer> |
solchecksum = <integer> |
comment = " any_text_within_quotes **"**

The **zdown** flag indicates that the global coordinate system uses a convention where the gravity force vector has a positive z component and the up vector has a negative z component. The default is the opposite. In general, CVED and the contents of the LRI file are coordinate system independent; however, there are a few places where CVED has to assign a normal vector automatically, and this is where this flag makes a difference.

The **solchecksum** entry contains the check sum of the SOL library that is referenced by the contents of the LRI file. This information is used to verify that the version of the SOL used at runtime is consistent with the version of the SOL used when the LRI file was created. CVED will report any inconsistencies.

LRI Road Section

The road section contains information about roads in the virtual environment. Each road contains references to adjacent intersections, attributes, longitudinal control points, and any number of lateral profile curves. The following details the format:

<road_list> = **ROADS {** [<lateral_curve_list>] { <road_def_list> } **}**

<lateral_curve_list> = **LAT_CURVES {** { <curve_name> **{** <curve_def> **}** } **}**

<curve_def> = <width> { <lat_cpoint> <lat_cpoint> }

<lat_cpoint> = <height> <lateral_dist> <material_ref>

<road_def_list> = <road_name> <intersection1> <intersection2> **{**
[**ATTR** { <attribute > }]
[**REPOBJS {** { <obj_name> <lateral_dist> <start> <period> <end> [**aligned**] } **}**]
LANES { { <width> <direction> } **}**
LONGCURVE {
{ [<cpoint_name>] <x> <y> <z> <i> <j> <k> [{ <cpoint_info> } **;**] EOL }
}
}

<cpoint_info> = **LATCURVE** <curve_name> |
ATTR { attribute } |
LANEWIDTH { <width> }

<attribute> = inum <lane_mask> float float float float

<material_ref> = integer

<curve_name> = string
<road_name> = string
<intersection1> = string
<intersection2> = string
<obj_name> = string
<lateral_dist> = float
<start> = float
<period> = float
<end> = float
<width> = float
<height> = float
<lane_mask> = integer
<direction> = **P** | **N**
cpoint_name = string

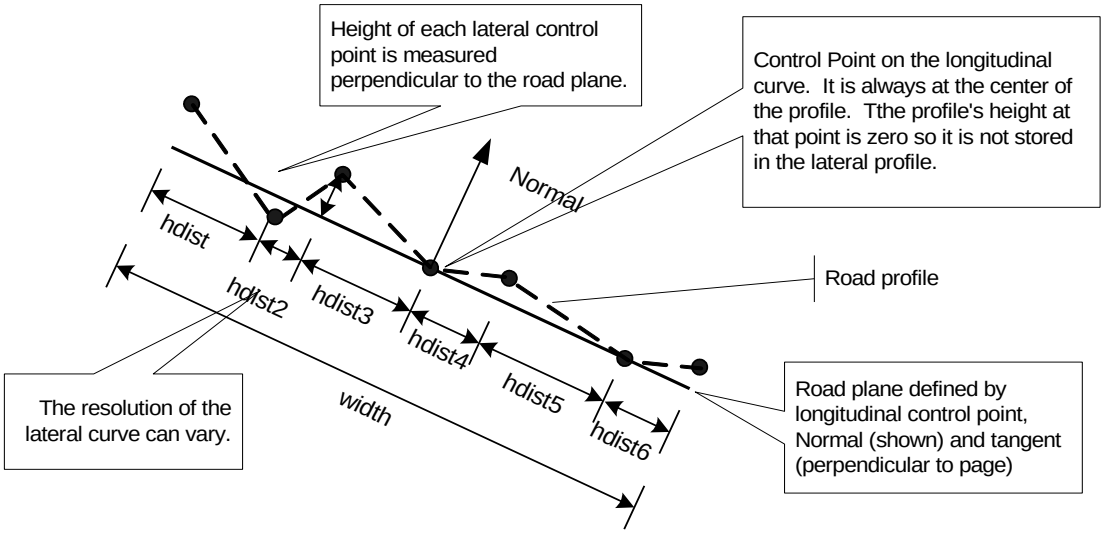
<x> = float
<y> = float
<z> = float
<i> = float
<j> = float
<k> = float

LRI Lateral Curve

The rule <lateral_curve_list> represents an optional list of lateral profile curves. If none is given, then roads can only have flat profiles. A global list is used because most of the time, classes of roads share profiles and thus space is minimized by simply referencing a road profile multiple times as opposed to providing it in the profile inline. Within the list, one or more curve definitions can be given, and all should have a name (<curve_name>) by which they can be referenced later.

A lateral curve is fully defined by an underlying width and an odd number of control points that provide the height and surface properties at distances along the profile. By definition, the height of the profile at the defining longitudinal control point, which is always the center control point, is zero so it is not included in the profile. Note that the syntax only allows an even number of control points to be specified in the LRI file. Adding the implicit control point at the center of the plane provides an odd number of elevations along the road's width. Each control point contains the elevation, the lateral distance from the road's centerline, and the material code applicable at that location.

Each profile control point represents the elevation of the road's surface at that distance from the center control point. That distance is perpendicular from the plane defined by the longitudinal control point and normal vector. The following figure illustrates the basic concept.



In addition to an elevation, each control point also references a surface property record. The detailed format of these is to be decided.

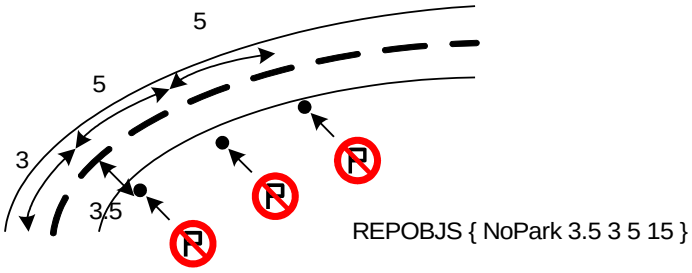
LRI Road Definition

LRI Repeated Objects

The REPOBJS line, which is optional, indicates the existence of one or more repeated objects associated with this road. A repeated object is an object whose definition appears only once in the list of global objects, but represents an arbitrary number of instances along the road centerline, starting at <start> units from the beginning of the road and appearing at a period specified by <period>, ending at a distance <end> units from the beginning of the road. The <obj_name> rule refers to the name of an object definition in the scenario object library, and the <lateral_dist> refers to the implicit position of the object laterally from the road centerline.

If the **aligned** flag is specified, the object's orientation will be aligned with the coordinate system defined by the road's normal, tangent, and right vector at the corresponding centerline position defined by the start/period parameter. If the aligned flag is not specified, then the normal vector of the object's orientation will be vertical. Examples of objects for which the aligned flag would be specified are pot holes or seams on the pavement. Examples of objects for which the aligned flag would not be specified are mile markers or signs.

The following figure illustrates an example. A single repeated object has been defined under the name NoPark. That object is placed at a lateral distance of 3.5 units from the road centerline. On the longitudinal axis, the first instance of it will appear at 3.0 units from the start of the road and appear each five units following that. Finally, after a distance of 15 units, the object will not appear any more. As shown in the figure, three objects will appear in CVED even though only one object needs to be stored. The savings in object storage space can be tremendous for objects that repeat thousands of times along roadways such as seams, pole markers etc.



Note that repeated objects are expected to increase the computational load of various CVED access functions, so their usage should be balanced against the space savings.

Attributes

Attributes are used to associate user-defined numeric values with roads and lanes in the virtual environment. Attributes contained in the LRI file cannot change at runtime, but the user can add

additional attributes through the CVED API. Attributes have a wide range of usage, including but not limited to speed limits, road classes (rural, highway, merge …), and other lane properties (turn lanes, HOV lanes etc.). Attributes provide a simple interface to extending the modeling capabilities of the LRI file without changes to the underlying software.

An attribute is identified and referenced by an integer. To ease debugging and error messages, the LRI file contains an optional attribute dictionary that associates a textual name with a numeric attribute identifier. See later section for information on the attribute dictionary.

Each attribute is further defined by four floating point values. The first two floating point numbers represent the values of the attribute and have no particular meaning to the parser. The LRI file simply contains that information, and interpretation is left to the user. The last two numbers are the applicability range of the attribute. The applicability range is the distance along the road for which the attribute applies. The first number is the distance along the road centerline for which the attribute starts applying, and the second number is the distance along the road centerline on which the attribute ends applying. The distance is measured linearly along the road's centerline control points.

If the first startpoint is set to -1 (or any negative number), then the start of the road is implied as the beginning of the attribute's application. Similarly, if the end point is set to -1, then the end of the road is implied as the end of the attributes application. When attributes are specified within a control point and the start end point value is -1, then this attribute will be applicable starting at that control point. The end point can be a value or -1 and the meaning is similar to when the control point is specified in the road section.

A lane mask allows association of an attribute with any subset of the lanes on a particular road. The lane mask is an integer representing a bitmask. If a particular bit is turned on, then the attribute applies to the corresponding lane. The least significant bit references lane 0. A negative integer in place of the lane mask indicates that the attribute applies to all lanes.

LRI Intersections Section

The intersections section contains information about all intersections in the virtual environment. In addition, it contains the list of elevation maps used by any intersections for terrain guidance. The elevation map syntax is similar to the one used for defining the elevation profile within terrain objects. The following details the format:

intersec_defs = **INTERSECTIONS {** [{ <elev_map> }] { <intersec_list> } **}**

<elev_map> = **ELEVMAP** <name> <nrows> <ncols> <res> **{** {<z> <mat_ref>} **}**

<intersec_list> = <intersect_name> <elev_info> **{**
ROADS { <road_name> }
[**BORDER** <x> <y> <lflag> <x> <y> <lflag> { <x> <y> <lflag> }]
[{ **CRDR** <road_name> <lane> <road_name> <lane> **{**

```
        [ { HOLDOFS <dist> <clrnc> <crdr_reason> [ hline_info ] [ {cr_sign_info} ] } ]
        CRDRCURVE {
          { <x> <y> <width> [ <crdr_line_info> ] EOL }
        }
        [ ATTR { <crdrs_attr> } ]
       } } ]
      [ ATTR { <intrs_attr> } ]
}

<crdrs_attr >          = inum float float float float
<intrs_attr >          = inum float float

<elev_info>            = <height> | <elev_map_name> <xorig> <yorig>
<hline_info>           = HOLDLINE = <thick> <angle>
<cr_sign_info>         = HOLDSIGN = <name>
<crdr_line_info>       = LINES <lflag> <lstyle> <lflag> <lstyle>
<name>                         = string
<nrows>                        = integer
<ncols>                = integer
<res>                  = float
<lane>                 = integer
<intersect_name>       = string
<height>                       = float
<elev_map_name>                = string
<road_name>            = string
<lflag>                = y | n
<dist>                 = float
<clrnc>                = float
<crdr_reason>          = SIGN | TLIGHT | OVLAP
<width>                        = float
<xorig>                = float
<yorig>                = float
<x>                    = float
<y>                    = float
<thick>                = float
<angle>                = float
<mat_ref>                      = integer
<lstyle>               = SOLID | DOTTED
```

Each intersection is given a name and information about its elevation (<intersect_name>, <elev_info>). The elevation can be specified as a floating point number, in which case the intersection is flat, or as a

string that is taken as a reference to an existing elevation map. If an elevation map is given, it defines the elevation when inside the intersection.

Each intersection block contains an optional border that provides an outline marking the intersection's boundary along with a list of corridors. Finally, any number of attributes can be associated with an intersection (rule <intrs_attr>).

A corridor block begins with the definition of the source and destination roads and lanes (<road_name> <lane>). Within the block any number of hold offsets can be specified. Each hold offset contains the distance (along the corridor's centerline) that it is located along with the clearance parameter and the reason for the hold offset. The reason parameter can be one of SIGN/TLIGHT or OVLAP indicating a traffic sign, a traffic light or a conflicting corridor. When the reason is OVLAP, indicating that the corridor exists because of the intersection of this corridor with another one, the clearance parameter indicates how far behind the hold offset a vehicle would have to stop to avoid colliding with any other object traversing the conflicting corridor. The optional <hline_info> rule specifies the layout of the horizontal line that is found in front of hold offsets by providing the line's thickness (<thick>) and orientation (<angle>). A thickness of 0 indicates that there is no line. The <cr_sign_info> rule references the controlling object, if the reason is SIGN or TLIGHT.

The corridor curve is defined by a 2D path and associated width. Note that the third dimension and normal information is derived by the elevation map or flat elevation, which is specified at the intersection header. The curve is linearly interpolated in two dimensions to define the corridor's curve. At each line segment, one can specify if a line exists to the left or to the right of the corridor and if any, specify if it's a solid line or dotted line. The syntax requires that a corridor contains at least one control point; however, it does take at least two control points to define a curve, so it is expected that semantically valid lri files will contain at least two points per curve.

Finally, attributes can be associated with each corridor (<crdrs_attr>). Attributes are specified by providing their identifier along with their two user specific parameters and the start/end ranges for their applicability along the corridor's curve. Similarly to the attribute specification method for roads, a –1 indicates that the attribute applies from the start or end of the road respectively.

LRI Objects

The objects section lists all static or reconfigurable objects in the virtual environment. Once an object is listed in this section of the LRI file, it cannot be removed nor have its attributes changed by the runtime API, with the only exception been the limited changes allowed through the reconfigurable object model.

Each object is identified by its type, a unique object name, the SOL identifier and its position and orientation. If an object name appears in the LRI file more than once, then it is considered to be a reconfigurable object and CVED will contain a single definition of an object but with as many options as there are duplicate entries. The default entry for the object's option list will be the first entry encountered in the LRI file. Duplicate object names must have SOL identifiers that belong to the same

SOL category, their location must be the same and their bounding boxes (as defined by the width and length fields) must also be the same size.

The next four parameters place an instance of an object in the specified coordinates with the specified yaw angle. A value of 0.0 for yaw would have the object along the x axis. To allow "planting" objects on the terrain, the z coordinate could be specified as **plant** in which case the object's Z will be defined by any underlying terrain as defined by roads or intersections. The following lists the syntax of this section of the LRI file.

The last parameter specifies the unique id of the object in the LRI file. It is to facilitate controlling objects using numbers instead of strings. A valid id is a positive integer. If an object does not need to be controlled, its id is left to -1. Usually the tool that generates the visual database assigns a unique number to each controllable object, and that number is correlated to the LRI file.

object_defs = **OBJECTS {** [{ obj_def }] **}**
<obj_def> = <obj_type> <obj_name> <sol_id> <x> <y> <zval> <yaw> <id>
<obj_type> = string
<obj_name> = string
<sol_id> = string
<x> = float
<y> = float
<zval> = float | **plant**
<yaw> = float
<id> = integer

LRI Attribute Dictionary

The attribute dictionary, which is optional, allows the association of strings with attribute identifiers. It is primarily used to allow readable error or status messages from software that is using the attributes.

This section consists of a list of integers followed by a string defining the attribute's name. Duplicate attributes are allowed, the latest values simply overwrite the earlier values. The following lists the syntax of this section of the LRI file.

attr_dictionary = [**ATTR_DICT {** { attr_dict_entry } **}**]
<attr_dict_entry> = <attr_name> <attr_id>

<attr_name> = string
<attr_id> = integer

APPENDIX B – COMPARISON OF TRADITIONAL VISUALIZATION METHODS AND DRIVING SIMULATION

Visualization type	Key technical differences	Technical differences show how converter tool is more effective	Source of information
Visualization through artistic software (e.g. 3DStudioMax)	- Costly to develop in terms of time or personnel; specialized skills required for best results - Visuals may not be based on actual engineering design models - Visualization are non-interactive animations - a user can "fly through" but not interact with environmental elements - Final visualization product not easily changed when source design is modified	- 95% automated processing, results are available quickly; no 3D technical expertise required - Eliminates re-creating road specifications by using engineering drawings and models directly - Driving simulation can provide detailed interactive environments with working traffic elements - Simulation model can be re-created easily by processing model updates	Autodesk Simulator (Parsons Brinkerhof Presidio Parkway project); FHWA Exploratory Advanced Research project (NADS 2011) Iowa DOT/ISU converter project
Driving simulators	- Most driving simulators require core competencies that include 3D modeling, scenario programming, integration and operation - Model scenes and scenarios are not compatible between different simulators	- Converter removes the need for additional 3D modeling; within a simulator platform, integration skills are reduced - Converter could be expanded to a common data format compatible with identified simulators	Handbook of Driving Simulation for Engineering, Medicine, and Psychology, 2011. Chapter 39. Roadway Visualization Current FHWA project
Visualization through roadway design software	-Lacks capability to evaluate test drive performance -Lacks capability to import 3rd party designs -Lacks compatibility with 3rd party driving simulators	Converter accepts roadway design model files and converts to driving simulator compatible format	FORUM8 UC-Win Bentley MicroStation

APPENDIX C – TILE TERRAIN MODEL CONCEPTS

As described earlier, the NADS MiniSim uses a library of reusable terrain model pieces to construct larger detailed terrain environments for simulation. These models are known collectively as tiles. Tiles are standalone individual models that typically contain roads, terrain and features. Tiles are grouped to allow easy selection when using the database publishing software to create a terrain configuration. The categories are intended to provide a coarse level of differentiation between tiles with different features (*culture*). Generally, tiles will contain roads, roadway markings, terrain and vegetation. More complex tiles include signage or traffic control devices such as traffic signals, railroad crossing signals, and general signage. In most cases, signs and traffic control devices are designed to be operated during simulation, which means they may be *authored* to change appearance either during initialization (i.e., changing a stop sign to a yield sign), or change dynamically during simulation (i.e., traffic light state changes from red to green to yellow).

Standard tile categories* include:

a. City

b. Comm (commercial features)

c. Filler (borders, panoramas and fillers)

d. Fwy (freeway)

e. Ind (Industrial features)

f. Mtn (Mountain)

g. Railroad (includes a rail line)

h. Res (Residential)

i. Rural (Countryside)

j. Urban

k. Special (Tiles which are non-standard in some way, or project-specific and not intended for general-purpose use)

l. Suburb

**Tile categories are customized to meet project needs and can be extended to include additional categories as needed.*

Due to the nature of the visual and virtual correlated databases, special effects such as wet or snow environments have been constructed as specialized tiles, with the intended effect built-in. Time of day is a more general simulation effect, although for some complex tiles, a lighting effect has also been built into the tile. These special-function tiles are identified by keyword within the tile name (wet, day, night).

APPENDIX D – TILE ASSOCIATED FILE SET

The Tile Library consists of a number of related files, all of which are necessary to the use of the standard terrain model publishing workflow for MiniSim simulator environments:

1. allTiles.txt – this text file contains a list of the tile library tiles and categories; the configuration file for the Tile Mosaic Tool (TMT).

2. tileSizes.txt – this text file contains a list of tile library tiles and their dimensions, given in Tile Units (increments of 660 ft; a 1 x 1 tile = 660ft x 660ft). Secondary configuration file for the TMT.

3. LatProfileList.lat – this text file contains all the lateral specifications for every road type. It includes a cross-section profile and a material code index.

4. Intersection.map – a text file that contains terrain data and specifications for unique elevation maps applied inside intersections.

5. SurfaceMaterialSpecifications.xlsx – an Excel document that contains surface material codes for road surfaces.

6. FLT – These are binary OpenFlight files that contain geometry and references to image texture files.

7. FTR – This is a TMT support file that contains all the tile references necessary for a terrain configuration including XYZ offset, rotation, and tile category type.

8. RGB, RGBA, DDS –binary image texture files. The RGB and RGBA files may also have associated .attr files: these files are produced from the modeling environment tools and are not currently used by the TMT or MiniSim. However, the RGB, RGBA and DDS files are necessary for MiniSim simulator operation.

9. PATH/CORR – these are text data files that contain coordinate data describing either roads or intersection corridors.

10. ICN – this is a binary file that is used for the TMT tile icon.

11. PET – this text file contains logical database tag information.

12. JPG – this is a binary snapshot view of the tile from an overhead view.

13. TXT – this text file contains TMT information. The TXT file will be updated when the binary FLT file changes, and should remain read/write. If you want to force the TMT to re-process the tile library or any particular tile, delete the TXT files and the TMT will re-create them.

14. MOS – this binary file is a TMT project file, also referred to as a terrain configuration file.

15. SUP – this binary file is used by the TMT.

16. CD1 – A text file containing a list of the Tile Library Tiles used in the MOS.

17. CD2 – A text file that defines the connective edges between tiles.