

**UTAH TRAFFIC LAB DRIVING SIMULATOR:
USER MANUAL AND FLEX LANES SCENARIO DEVELOPMENT**

Dr. Peter T. Martin
Milan Zlatkovic
Ivana Tasic

Department of Civil & Environmental Engineering
University of Utah

June 2012

Disclaimer

The contents of this report reflect the views of the authors, who are responsible for the facts and accuracy of the information presented herein. This document is disseminated under the sponsorship of the Department of Transportation, University Transportation Center program, in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof.

North Dakota State University does not discriminate on the basis of age, color, disability, gender expression/identity, genetic information, marital status, national origin, public assistance status, sex, sexual orientation, status as a U.S. veteran, race or religion. Direct inquiries to the Vice President for Equity, Diversity and Global Outreach, 205 Old Main, (701)231-7708.

ABSTRACT

This report describes the University of Utah Traffic Lab (UTL) driving simulator, and a scenario development for Flex Lanes driving simulation. The first part describes the driving simulator in details. At the time of installation, the UTL driving simulator was unique and the first to join microsimulation with driving simulation. This type of integration offers major possibilities beyond those of a “classic” driving simulator. The second part is dedicated to the development of the Flex Lanes scenario. The Utah Department of Transportation (UDOT) will implement the first Flex Lanes (reversible lanes) project on 5400 S in the City of Taylorsville, Utah, in summer of 2012. The UTL, in cooperation with the AAI Corporation, has developed a real-world driving simulation scenario of the Flex Lanes corridor. The UTL is planning to use this driving simulation to assess the drivers’ performance and compliance with the posted signalization.

TABLE OF CONTENTS

1. INTRODUCTION	1
1.1 VISSIM and ARCHER	1
2. UTAH DRIVING SIMULATOR HAREDWARE COMPONENTS.....	3
2.1 Driver Side Vehicle Station	4
2.2 Three Linux-Based Computers for Rendering.....	4
2.3 One Computer for Operations Console (OpsCon).....	4
2.4 One Computer for Rear View Rendering	4
2.5 One PC for Dynamics, I/O Control and Sound.....	4
2.6 Monitors Dell Professional 22” FPD	5
2.7 Three 55” Full HD TVs	5
2.8 Three Xenarc 7” LCD Displays for Rear View	5
2.9 One 8-Port KVM switch+Cables	5
2.10 Network Switch+Cables	5
2.11 The VISSIM Traffic Modeling Software and Computer Supplied by the University of Utah....	5
2.12 Hardware Preventative Maintenance	6
3. THE ARCHER DRIVING SIMULATOR SYSTEM BY AAI.....	7
3.1 ARCHER Software Layers	7
3.2 ARCHER Requirements	7
3.3 ARCHER System Roles and the ARCHER.exe Main Program	9
3.4 The Packages / Modules of ARCHER.....	10
4. SYSTEM POWER UP, STARTING AND RUNNING ARCHER SCENARIOS	11
4.1 University of Utah Driving Simulator Power On and Quick Start Process	11
4.2 Launcher GUI	12
4.3 Starting ARCHER by Command Line.....	14
4.4 ARCHER GUI Window	15
4.4.1 Advanced View Control Panel.....	16
5. GENERATING ARCHER SCENARIOS – “QUICK START”	19
5.1 ARCHER Scenario File	19
5.2 The Scene File and ARCHER Scene Elements	21
5.3 ARCHER Scenario Types.....	23
6. THIRD-PARTY TOOLS	25
7. SCENARIO AND EXPERIMENTAL CONTROL.....	27
8. ADVANCED SCENARIO CREATION – THE CFI VISSIM SCENARIO, AN EXAMPLE	29

9. FLEX LANES: INTRODUCTION.....	41
10. LITERATURE REVIEW	43
11. THE NEED FOR FLEX LANES ON 5400 SOUTH	45
12. FLEX LANE DESCRIPTIONS FOR TRANSITIONAL LANES	47
13. LEFT TURNS WITHIN THE FLEX LANES SYSTEM.....	51
14. TRANSITION SCENARIOS	53
15. FLEX LANES DRIVING SIMULATOR SCENARIO DEVELOPMENT	57
15.1 VISSIM Model Development.....	57
15.2 Flex Lanes Driving Simulation.....	62
16. CONCLUSIONS.....	65
17. REFERENCES	67
APPENDIX A: Flex Lanes Signal Heads and States for Signalized Intersections.....	69
APPENDIX B: Driving Simulator Users Guide.....	79

LIST OF FIGURES

Figure 1.1	VISSIM & ARCHER Modeling Verses Perception	2
Figure 2.1	System Diagram.....	3
Figure 3.1	Layer Diagram	8
Figure 3.2	ARCHER Software Layer Diagram.....	9
Figure 3.3	ARCHER Package Relationships	10
Figure 4.1	Launcher Interface	13
Figure 4.2	ARCHER GUI Window.....	15
Figure 4.3	Advanced View Control Panel.....	16
Figure 5.1	Template Scenario File Example	20
Figure 12.1	Lane Directions, Lane Assignment and Signal Gantries (Source: UDOT).....	47
Figure 12.2	The Look of Signal Gantries from the Drivers' Perspective.....	49
Figure 14.1	Lane Assignment and Transition	53
Figure 15.1	Overhead Signal Gantry Symbols and Corresponding Signal Groups: a) Eastbound; b) Westbound	59
Figure 15.2	Signal Control Program for Overhead Signal Gantries.....	60
Figure 15.3	Signal Head/Sign and Signal Group Configuration: An Example.....	62
Figure 15.4	Flex Lanes Driving Simulation	64

LIST OF TABLES

Table 14.1	Transition Scenarios	54
Table 14.2	Lane Condition, Intended Driver’s Action and Expected Challenges	55
Table 15.1	VISSIM Main and Transition Periods	57
Table 15.2	VISSIM Links for Reversible Lanes	61

EXECUTIVE SUMMARY

This report describes the University of Utah Traffic Lab (UTL) driving simulator, and a scenario development for Flex Lanes driving simulation. The report includes the user manual of the driving simulator that the UTL acquired in 2011 from the AAI Corporation. Also, a description of the development of the Flex Lanes scenario for 5400 S is given step-by-step.

At the time of installation, the UTL driving simulator was unique and the first to join microsimulation with driving simulation. This type of integration offers major possibilities beyond those of a “classic” driving simulator. The driving scenarios are much more realistic, with controllable traffic operations on multiple levels. It provides inputs of real-world traffic data into driving simulation and detailed outputs of the behavior of driving test-subjects. This expands the level of information that can be obtained from each driving test.

The second part of the report is dedicated to the development of the Flex Lanes driving simulation scenario. The Utah Department of Transportation (UDOT) will implement the first Flex Lanes (reversible lanes) project on 5400 S in the City of Taylorsville, Utah, in summer of 2012. This will be the first implementation of that kind in Utah. A lot of innovative traffic control systems are being used with this installation, which can become a problem for unfamiliarized drivers. The UTL, in cooperation with the AAI Corporation, has developed a real-world driving simulation scenario of the Flex Lanes corridor. The UTL is planning to use this driving simulation to assess the drivers’ performance and compliance with the posted signalization. Any potential problems with drivers’ behavior or traffic control can be pinpointed in a safe, virtual environment and help UDOT with a safe and efficient implementation of the new system.

1. INTRODUCTION

This manual describes the University of Utah Traffic Lab (UTL) driving simulator. The simulator is a 3D visual system of hardware and software produced to allow research of driving behavior. This manual will describe all the components of the simulator, how to start and operate the simulator, how to generate new scenarios, how to troubleshoot, where to get technical support, and what licenses are involved with this product. As an integrated system, the simulator has many features as described below:

- The feel of driving a fully interactive, real car from the driver's perspective
- 3D driving simulator visual system using AAI ARCHER graphics and interactive real time scenario control
- Operator interface that is targeted for the reseARCHER and research environment
- A system that allows for programmable scenarios by the user
- Integrated with VISSIM Traffic Micro simulation application
- Real driver cockpit controls
- Customizable operator graphical user interface, Java-based, controls that can be customized to the specific research scenario
- Event-based dynamic scenario elements and controls
- Roadway infrastructure-based scenario construction
- Driver perspective-based scenario controls

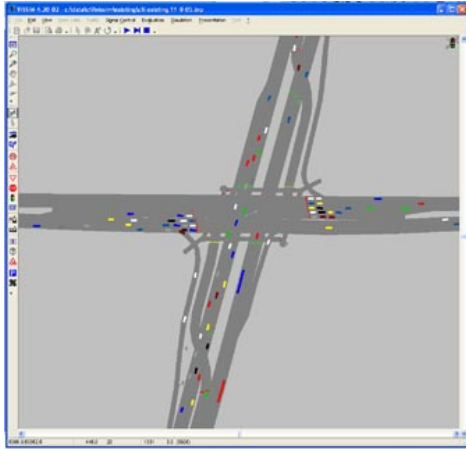
The UTL driving simulator is designed to merge the ARCHER research driving simulator software with the VISSIM traffic micro-simulation software. The current hardware implementation uses a fixed-based driver cockpit (sometimes referred to as a "driving BUC").* The system is hosted on a collection of five Linux-based computers (for scenario control and 3D rendering) and two Windows computers (one to host the vehicle dynamics and driving control I/O and one to host VISSIM). The UTL driving simulator uses the ARCHER 3D Rendering and real-time scenario control software that was originally designed to run the FHWA Highway Driving Simulator. This software and operator interfaces are targeted toward reseARCHERs and research environment. Using the ARCHER software, combined with other third-party tools, different programmable scenarios can be created to match either real world locations or roadway locations specific to research requirements and scenario elements needed for highway and traffic research.

1.1 VISSIM and ARCHER

The traffic generation is accomplished by using the VISSIM micro-traffic simulation software, which has an ARCHER compatible library. This is used to communicate the traffic vehicle positions to the driving simulator at a high update rate. This combination of VISSIM generated traffic and high-end 3D rendering allows reseARCHERs to create customizable virtual roadway scenarios populated with a rich traffic environment. This is accomplished through research-oriented operator controls and event-driven data collection mechanisms. With both ARCHER and VISSIM, the scenario is from the perspective of the roadway and the driver. Figure 1.1 illustrates how this works.

* Driving Buc is a term used for a non-enclosed simulated driver position for a vehicle that includes interactive driving controls such as steering wheel, brake, accelerator, instrument panel, etc.

VISSIM



Traffic Vehicle positions
&
Traffic Signal States



Driven Vehicle Position



ARCHER



Figure 1.1 VISSIM & ARCHER Modeling Verses Perception

2. UTAH DRIVING SIMULATOR HARDWARE COMPONENTS

The UTL driving simulator system is configured complete with a driving BUC, rendering computers, an operators console, a rear view rendering computer, vehicle dynamics and sound computer, 55" LCD panels, monitors, and all required cables (See Figure 2.1, floor layout).

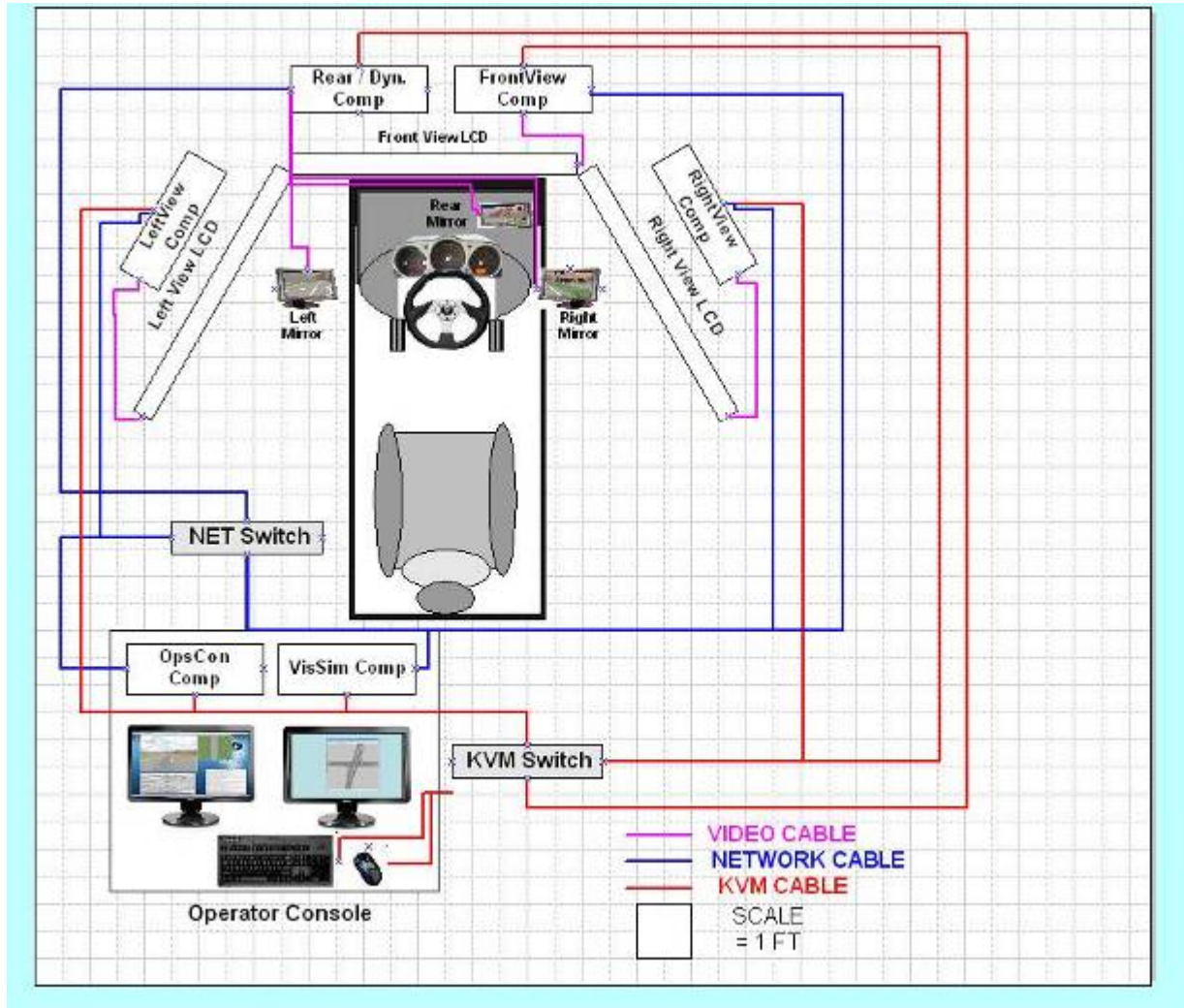


Figure 2.1 System Diagram

2.1 Driver Side Vehicle Station

Included is an open cockpit realistic looking “driving Buc” complete with force feedback steering and foot pedals. All normal console controls and dials such as turn signals, gear shifter (automatic), MPH, and RPM dials will be included. Speakers will add vibration components to the driving Buc for road vibration as well as engine and road noise.



2.2 Three Linux-Based Computers for Rendering

Three AAI supplied 3D graphics rendering computers equipped with Quad Core i7 CPUs and multi-GPU capable (multiple video card on the same motherboard) will be used for primary real-time rendering computers. NVIDIA 480 or better video cards will be used for maximum rendering capability.



The computer case is customized for maximum air cooling at a low noise level. One computer each will be used to drive the front, right and left view displays. Each computer will be connected to one of the 55” full HD LCD TVs. The design of these computers is built to be upgradable and expandable. Each computer will include a 1200 watt power supply to support the high-end computing and graphics components.

2.3 One Computer for Operations Console (OpsCon)

One AAI supplied computer will be used to run the operations console (operator’s scenario controls). This computer will include a NVIDIA 275 (or better) video card for good rendering performance. This computer will utilize a high resolution 22” Dell professional monitor.



2.4 One Computer for Rear View Rendering

One AAI supplied computer with 2 NVIDIA 475 GTX video cards will be utilized to generate the images for the rear view and side mirrors. This computer will be connected to the three Xenarc 7” LCD displays



2.5 One PC for Dynamics, I/O Control and Sound

One PC will be used to run the Vehicle Dynamics software as well as for reading and writing (I/O signals) to the driving controls (steering, foot pedals, etc.). Sound generation will be handled by this computer, including driven vehicle engine, road, and wind sounds. Passing traffic sounds will not be available but will be with a future version of ARCHER.



2.6 Monitors Dell Professional 22" FPD

A Dell 22" professional display with full HD resolution will be used at the operators console as the display for scenario control and vehicle dynamics interfaces with switchable A/B video inputs.

2.7 Three 55" Full HD TVs

Three full HD LCD-based TVs, with LED backlighting for ultra-high contrast, will be used for the forward displays enclosing the driver. An approximate display angle of nearly 180 degrees will be achieved depending on driver's seat positioning.



2.8 Three Xenarc 7" LCD Displays for Rear View

Seven-inch diagonal LCD displays will be used to represent rear and side mirror resolution views. Each mirror display will run at a resolution of 1024x800.



2.9 One 8-Port KVM switch+Cables

An 8-port KVM (keyboard, video, mouse switch) will be used to allow a single wireless keyboard, wireless mouse, and video to be used on all of the computers.

2.10 Network Switch+Cables

An 8-port gigabit network switch with Cat6 cable will be included for high speed network connections between the computers.

2.11 The VISSIM Traffic Modeling Software and Computer Supplied by the University of Utah

A computer will be supplied by the University of Utah to run VISSIM and generate traffic for the ARCHER scenarios. It is suggested that a higher end computer with a minimum of 4 gig of main memory, a quad core i7 CPU running at 2.8 GHz or higher be used.

The University of Utah will contact PTV America (producers of VISSIM) to provide a version of VISSIM (version 5.3 or later) bundled with the software interface, which allows vehicles controlled by the VISSIM application to update the 3D rendering of vehicles used in the ARCHER scenario.

2.12 Hardware Preventative Maintenance

Please see supplied owner's manuals for each hardware device supplied above for care and maintenance requirements. The supplied computers have been built by AAI and have few required maintenance requirements other than cleaning lint and dust from the filters on the bottom of the case. It is recommended that scheduled backup of data created and stored on the OPSCON (scenarios and collected data) and the dynamics/driver cockpit control PC (elevation data) be maintained at regular intervals.

3. THE ARCHER DRIVING SIMULATOR SYSTEM BY AAI

The ARCHER (**Advanced Rendering Cluster for Highway Experimental Research**) application has been designed to run the **Highway Driving Simulator (HDS)** using multiple computers for the roles of 3D rendering (aka image generation), audio generation, vehicle dynamics, scenario control (operators control or Opscon), and car cab/motion base interfacing (through a VME I/O chassis). The diagrams above show some of the typical hardware components of the HDS system. Each of the mentioned roles can be run using different configurations for running on different computer setups, including all of them on a single computer. When running on multiple computers, TCP/IP networking sockets are used to communicate between the various roles.

3.1 ARCHER Software Layers

The ARCHER software has been developed using Open Source software solutions. The layer diagram (Figure 3.1) describes some of the components that ARCHER uses. The software itself is written in C++ and uses the OpenScene Graph (OSG) rendering API (soon to be replaced by the Scenix API by NVIDIA). These rendering APIs are both based on the standard OpenGL graphics library and can use either NVIDIA Cg or OpenGL Shading Language (GLSL) for advanced GPU shading capabilities. All of these software libraries can be run on either the Fedora Core (version 10) Linux or Windows XP operating systems. Generally the Linux OS is utilized for clustered real-time rendering due to its superior software performance on the same hardware and its much smaller memory and resource footprint for the same capability. ARCHER relies on the JAVA programming environment and run-time environment for advanced graphical user interface (GUI) components. New JAVA GUIs can be developed by the end user by following the example GUIs provided with ARCHER and customizing them with new or modified user interfaces. ARCHER also uses SpeedTreeRT for tree modeling and rendering.

The UTL must agree to the sub-packages license agreements in order to use ARCHER. These EULAS (end-user license agreements), license files, and copyrights will be available with the ARCHER license information and terms of use (see **Attachment 4**). In general, the user must not reverse engineer, resell, or repackage for sale any of the ARCHER software or software bundled with ARCHER. Also, the producers of ARCHER and the packages bundled with ARCHER are not responsible for how the software is used or how content and media created with this software is used. Additional information on the other third-party libraries used by ARCHER can be found under the third-party licensing section in **Attachment 5**.

3.2 ARCHER Requirements

The ARCHER system is scalable and will run on a single machine or on multiple machines as described above. ARCHER is tweaked to run most efficiently under Linux (RedHat Fedora Core 10). The Linux version is very efficient at networking, which optimizes clustered operations. (The Windows version is currently in development but is expected to be less efficient at networked clustered applications.) The configuration can be defined at startup by defining the configuration file on the command line. A predefined icon will have this already defined for a given system, but a variant setup can be run as needed by substituting a different configuration file. It is recommended to run the ARCHER GUI from a different machine than the rendering computers. This will take advantage of the operator's console and GUI being able to run at a nonsynchronized rate from the front rendering machines and not slowing them down.

ARCHER Software Layers

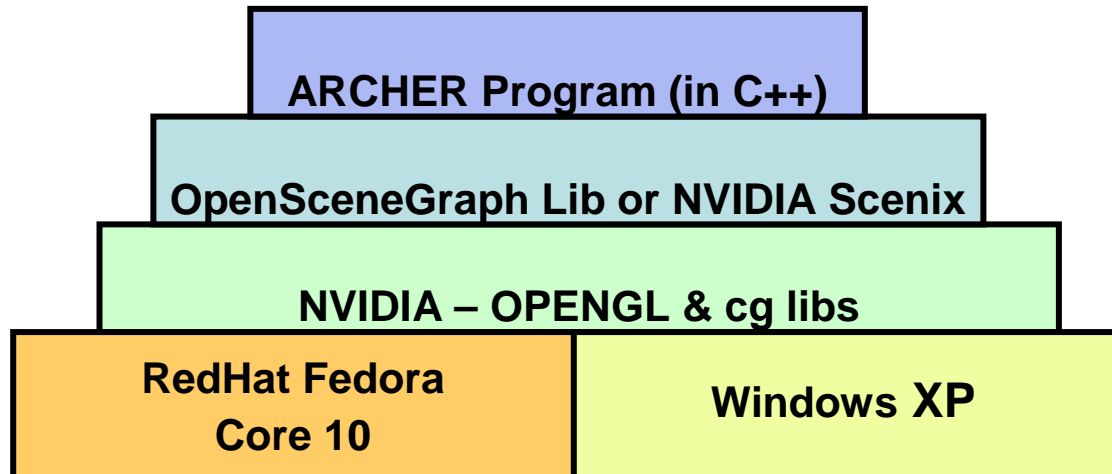


Figure 3.1 Layer Diagram

Ideally, ARCHER should be run on machines with fast CPUs (Intel or AMD) and fast/high end graphics cards. ARCHER can take advantage of NVIDIA SLI mode (multiple graphics cards in a single machine) to render even more complex scenes. At this printing, it is recommended to use NVIDIA Geforce 480 cards or better with at least 1 gig of video memory on the graphics card; however, these graphics cards can be updated. A minimum of 2 gig of main memory should be installed on each computer running ARCHER. Disk drive space sufficient to store the scenarios can be hosted on a separate machine and shared across all ARCHER cluster computers. Typically, this should be at least 100 megabytes but additional space is suggested for growth. To create ARCHER scenarios, it is recommended that a modeling program such as Presagis Creator be used. For generating imagery and textures for models and signs, a program such as Adobe Photoshop CS4/5 is recommended. Other modeling and imaging applications can be used if they support a loadable format, but the programs mentioned have been used very successfully by the HDS team. In addition, to capture, edit, and generate videos, Adobe Premiere Pro CS4/5 is recommended. We have also found that the Google modeling package Sketchup 7 Pro has help in translating CAD packages to formats importable into Creator.

3.3 ARCHER System Roles and the ARCHER.exe Main Program

The **ARCHER** main program, ARCHER.exe, is only one of many programs and utilities contained in the CAR (clustered advanced rendering) library documented here. It does have the distinction of being the main program that starts an ARCHER session. Figure 3.2 illustrates the threads and network processes which are spawned when ARCHER is run. The **ARCHER** program first reads the **Configuration file**, which defines each role (on what computer) and its configuration. For example in the diagram below, **ARCHER** will be started on the OpsCon computer with the **View Controller thread** spawned locally. On the same computer, the **RunARCHER** script will start the scenario specific GUI implemented in Java (defined in the scenario file). Rendering threads will be started on Rendering Computer #1 and #2 and used as Front and a RearViews. The rendering computers run both a **Scene Manager** thread and a **Rendering Manager** thread on each computer. On the computer named Dynamio (Dynamics IO) the **VehicleThread** will be started and run the vehicle dynamics. At the same time, the **CarCab I/O program** is started on the Driver Cockpit computer and it will communicate with the **VehicleThread** for reading the steering, brake, gas pedal, and other control inputs from the car cab. The **VehicleThread** also sets the states for the car dials and gauges (speedometer, etc.) and sends commands to the motion base to change the car cab's orientation as needed by writing commands back through the **CarCab I/O program**.

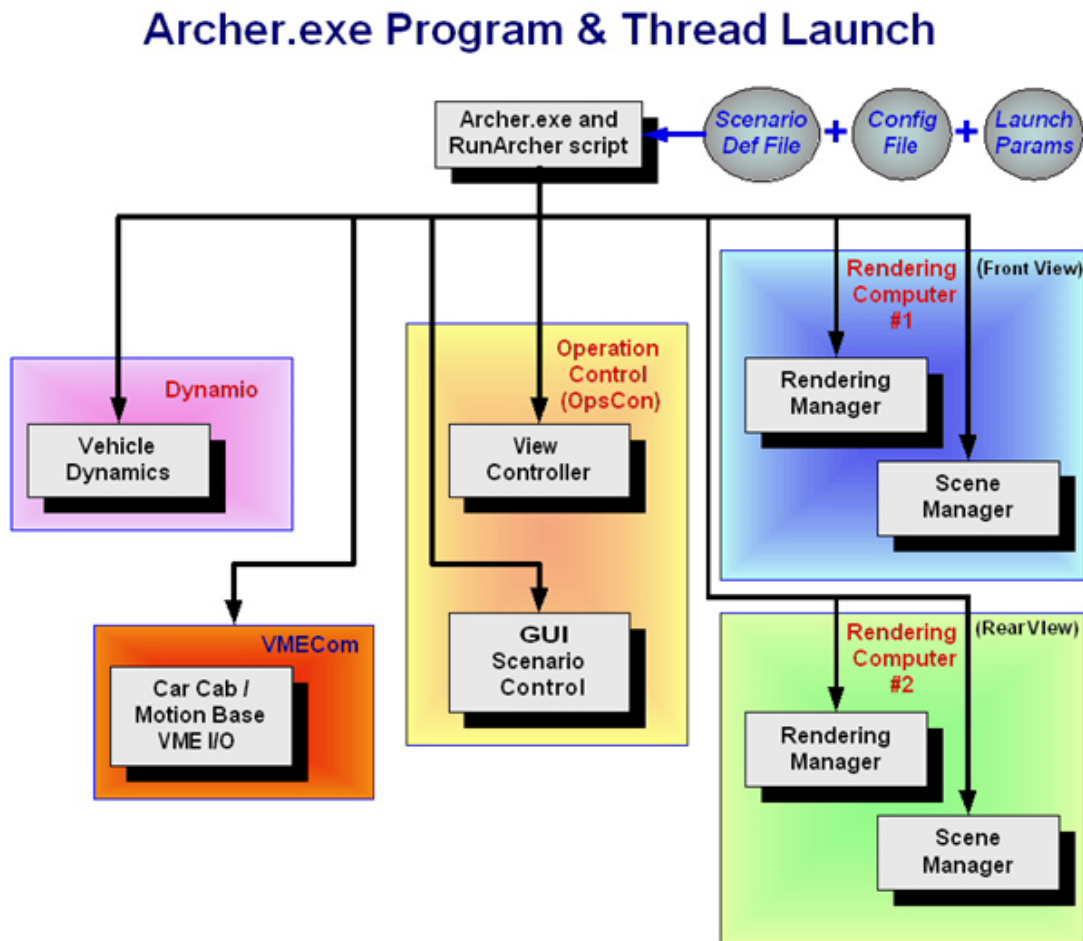


Figure 3.3 ARCHER Software Layer Diagram

3.4 The Packages / Modules of ARCHER

The ARCHER program is organized by **packages** (aka modules) which are grouped by function and design. The ARCHER packages can be divided into five major categories: Vehicle **Dynamics** and updating of the users vehicle position in the scene; **Car Cab I/O**; sound generation for vehicle engine, tire, wind, and other sounds; **View Control** and **Graphical User Interface (GUI)** control for the operations control station (OpsCon); and the majority of the other packages in the ARCHER system are used in **rendering** and updating the scenario and scene elements (**SceneManager**). Many of the packages are defined to allow for better multi-platform support and isolation of hardware level function calls. The following diagram shows this separation by illustrating how these major roles of ARCHER functionality is used during an ARCHER session. The ARCHER.exe program is run on each computer with multiple **threads** sometimes being spawned to execute its function and communicating between computers using **TCP/IP sockets**. Figure 3.3 indicates various communicated params, which are passed through the system across the computer roles. These roles can be run on multiple machines or all of these roles can be run on a single machine as multiple threads.

The ARCHER system utilizes **Scene Graph** technology and currently uses the **Open Scene Graph (OSG)** library as a rendering engine to perform the real time rendering. The ARCHER design uses a logical component called a **SceneObject**. SceneObjects are generic “building” blocks of any scene defined to be rendered in ARCHER. **SceneObjects** are created by **SceneObjCreators**, which are specific to the SceneObject type being created. For example, a tree in the scene will be created by a **TreeCreator**.

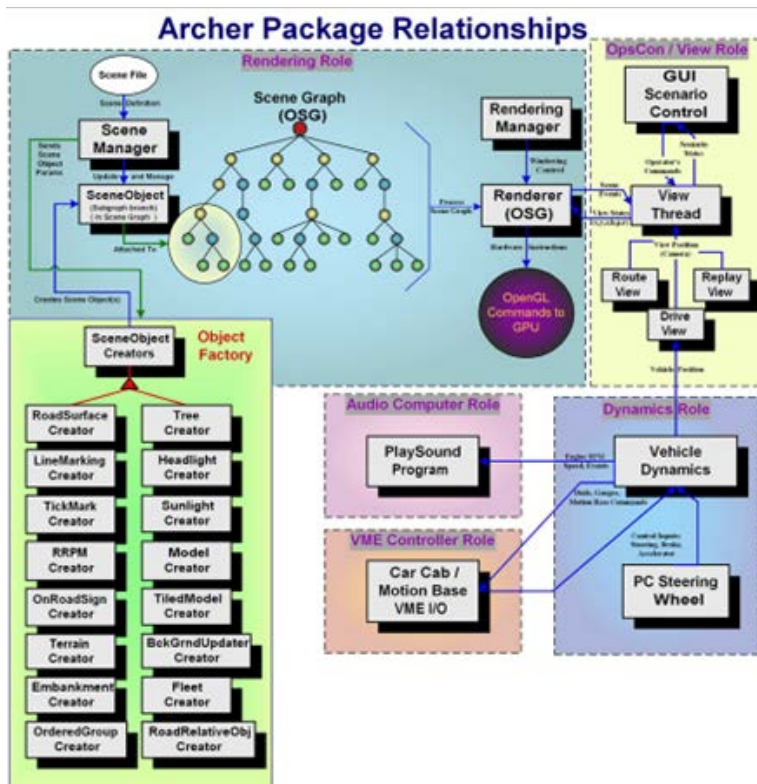


Figure 3.3 ARCHER Package Relationships

4. SYSTEM POWER UP, STARTING AND RUNNING ARCHER SCENARIOS

ARCHER can be run either from the command line, started from the Desktop shortcut ICONS based on scenario or started from the launcher GUI. To start ARCHER from the desktop shortcut icons on AAI OPSCON, follow the steps listed below in Section 4.1.

4.1 University of Utah Driving Simulator Power On and Quick Start Process

- 1) Turn on power strips, Power up **OpsCon** first by touching on/off control button on the top, front of PC
- 2) Wait 1 minute and power up all other PCs with power strip and top, front button
- 3) Power up the Driver Cockpit – left side bottom-fold down cover and press rocker switch, lights on PC and lights on dash board come on

Power up big screens

Use remote(s) or button on bottom left of each display

KVM logon – just hit return twice as there is no password set.

Note the KVM switch layout below:

- KVM – 1** Operators Console PC (**OpsCon**)
- KVM – 2** RightView PC – no display can be seen
- KVM – 3** FrontView PC – no display can be seen
- KVM – 4** LeftView PC – no display can be seen
- KVM – 5** CenterRearView PC – duplicate of rear view display
- KVM – 6** Not Used – no display can be seen
- KVM – 7** Driver Cockpit PC
- KVM – 8** VISSIM PC (Utah supplied)

- 4) **KVM -7** Driving Cockpit PC

Open “My Computer” and under “Network drivers” click connect

“\\155.98.128.128\export\shared” then login archer with password “3dmatrix”.

If this does not appear, then the IP address needs to be reset. Call AAI or an IT person to map drive “T”, username “archer”, password “3dmatrix”.

Click “**Steering Calibration**” icon and wait until complete, approx. 1.5 minutes. Note steering wheel movement.

Click “**RTI Dynamics**” icon

Command line box shell appears

Type ‘**test <elevation file name>**’ and note the selected terrain database must be consistent with the scenario being run in step 6.

CFI_VISSIM_Elevation.wrl

DDI_VISSIM_Elevation.wrl

PA851_Elevation.wrl (note this does not use VISSIM)

Parking_lot_Elevation.wrl (note this does not use VISSIM)

Is done when “**Entering main loop...**” appears in shell window.

5) KVM-8 VISSIM

Password = "3Dmatrix"

VISSIM RUN "archervissim_shortcut" icon

Is done when ""Use server port 0778" appears in shell window.

6) KVM-1 OpsCon – select one of the icons that matched step 4 above.

Run CFI VISSIM Demo load time about 2 minutes

Run DDI VISSIM Demo load time about 5 minutes

Run Parking Lot Training load time about 0.5 minutes

Run PA851 Demo load time about 3 minutes

Wait for screen logos to stop rotating.

After selection of Run items above, then Accept License Terms "Accept to Continue"

Enter "Name" in GUI top left of screen

Click on Start Session

7) KVM-8 VISSIM

Press Keyboard "Ctrl and Q" together, then press it again (twice, first to stop drawing the vehicles in the VISSIM window and second to stop drawing the signal states). This step will drastically improve performance.

8) KVM-1 OpsCon – System is now operable (see Section 4.4 on the ARCHER GUI)

4.2 Launcher GUI

The Launcher GUI is designed to help run experimental studies by applying launch parameters and attributes based on a lookup by participant number. The launcher GUI has been included as part of the ARCHER software but has not been set up for the University of Utah environment, as this will be defined as experimental studies are defined for the Utah program. To setup and configure the launcher GUI, the experimental.xml file must be edited and defined. This file is located in the /aai_archer/car/bin directory, the same location used to run ARCHER from command line (see Section 4.3 below).

/aai_archer/car/bin/experiments.xml – Defines experiment list to show in GUI and what .csv file to use to control the launch parameters.

The xml file will contain the following fields to define the experiments:

<experiments> <list of experiment structures below (see "experiment name=")>

<experiment name=" " description=" " expandbydefault="<true|false>" icon="<icon path name to display icon>" train="<comma separated list of Training scenarios to use">

<mapcode intOrderCode="<Letter>(usually a single code letter from csv file)"

intOrderAbbr="<text to display above participant number after table lookup">

<scenario name="<scenario name to display">

filename="<scenario file name to run from 4.3 below">

description="<a text description to display">

csvfile="<filename.csv>(a comma separated file to use to select ordercodes and launchparams based on participant number lookups">

sessionmod="<an advanced feature maybe useable for some specific scenario types">

launchparam="<a launchparam to pass through">

The launcher GUI can define the launch parameters to use for a given scenario as needed and defined for that given experiment. Figure 4.1 shows the launcher interface.

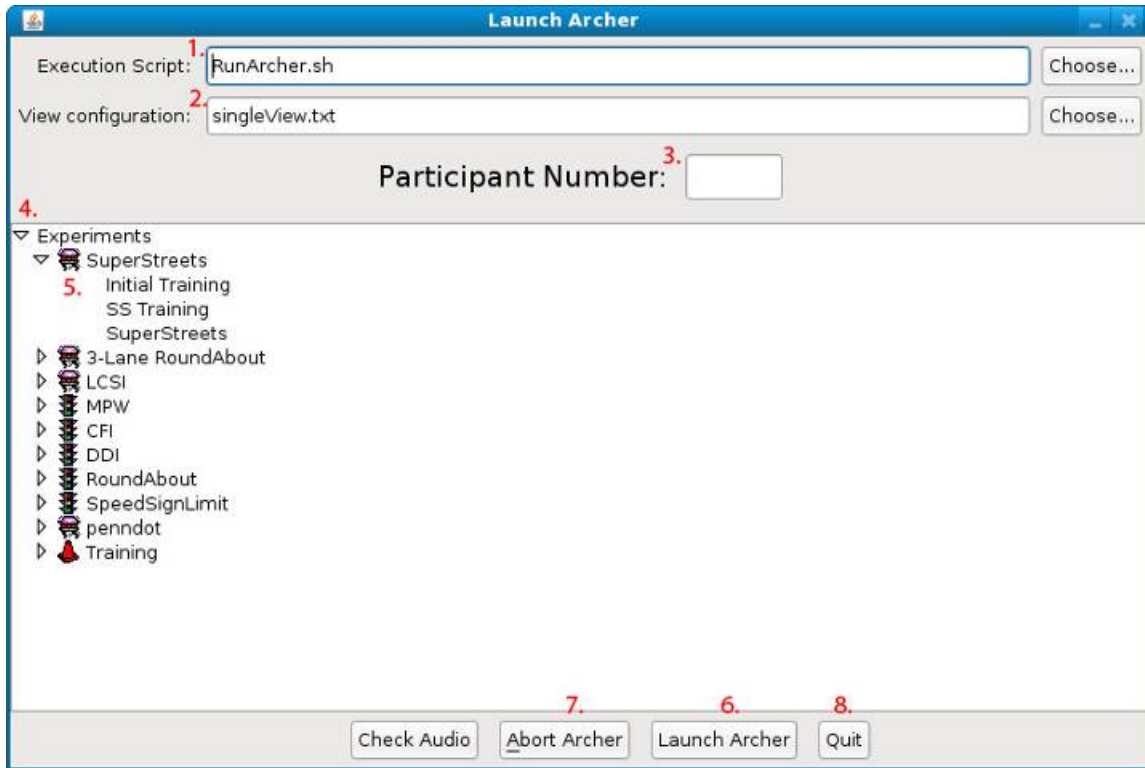


Figure 4.1 Launcher Interface

The “**Execution Script:**” text box (1) is preset to “**RunARCHER.sh**” and only available for debugging purposes and should not be changed. The “**View Configuration:**” text box (2) is also preset and should only be changed by advanced users. The launcher interface will configure what experimental scenario to run by selecting the desired scenario from the “**Experiments**” drop-down list (4). After selecting the given experiment name, the “**Participant Number:**” should be entered in the text box (3) by typing the desired value and left clicking on the experiment name. Entering a participant number will list the proper scenario names to be run in the correct order for the selected experiment. The scenarios are listed just below the participant number text box. These values will be different and specific to each experiment and scenario. Variant scenarios (specific to the selected experiment) are further selected by expanding the experiment list by clicking on the drop-down triangle (5) and selecting the desired session name. ARCHER then can be started with these values by clicking on the “**Launch ARCHER**” button (6). If there is a problem with ARCHER running or if ARCHER does not start correctly, then the “**Abort ARCHER**” button (7) should be clicked to end the ARCHER session. If restarting after aborting, waiting a 10-15 seconds before restarting is recommended to give the previous session time to fully abort all processes. The “**Quit**” button (8) ends the Launcher application and does not stop the ARCHER sessions that may be running. The running ARCHER application should end from the ARCHER OpsCon GUI and the “**Abort ARCHER**” button should be used only if normal closing of the ARCHER application does not work correctly or fully.

4.3 Starting ARCHER by Command Line

ARCHER can be started on the command line from a terminal window as well. To run ARCHER from the command line, bring up a command shell and change the directory (cd command) to the ARCHER top level directory (usually at /hds_ARCHER). Next, set the cvs working directory through the setcvs command. Change the directory to the car/bin directory. From there, any ARCHER scenario can be run from the command line. One item to note is that ARCHER will use the existing launchparams.txt file to set up certain scenario variables which may need to change for different scenarios. This file is automatically created when running ARCHER through the Launcher application.

Then to run, execute the command:

```
./RunARCHER.sh <config_file> <scenario_file>
```

Example: `./RunARCHER.sh singleView.txt pa851_6mi.txt`

Where the <config file> token should be a pre-created configuration file to match the hardware setup for either a single computer or across multiple computers. Typical configurations include: singleView.txt, proj3wMirror.txt, captureFrontView.txt, etc.

The <scenario_file> is any existing scenario file. Some scenario files include; pa851_6mi.txt, ddi_scenario_.txt, CFI_scenario.txt, LCSi_scenario.txt, etc. (see scenario creation section 5)

After the command is executed, several windows should pop up depending on the configuration used. In general a GUI window related to the scenario will pop up on the OpsCon machine or on the same machine if run in single machine mode. Two different plan views will also be displayed, one for lane keeping and one for map-sized route information. If running on a multiple machine configuration, the other windows will generally be full screen rendering windows on given rendering views. For example, front view will be a full screen windowless 3D perspective view out the front view. The left view will be displayed full screen on the left view rendering computer, and so on.



Figure 4.2 ARCHER GUI Window

4.4 ARCHER GUI Window

The ARCHER GUI window is used for scenario control and to display various scenarios-based data. The actual configuration and controls displayed in this window will vary with each experiment. A different GUI window can be constructed and customized for each experiment as needed. The GUI is written in Java and an example template program is included. The version of ARCHER (not the GUI version) is displayed in a small font in the upper left corner of the window. A large text field (1.) lists the scenario state (LOADED / RECORD / SAVED) and scenario name. The SUBJECT ID text field (2.) will display the current subject ID, which is active. If this value differs from what's in the input field (3.), it has not taken effect yet and the enter key should be pressed. A new Subject ID can be typed in the text input field (3.), and will take effect when the enter key is pressed (automatically placing ARCHER into a DRIVE mode for this unused Subject ID and session settings, see 5.). The Start / End Session radio buttons (4.) will start and end the current session using the current Subject ID and other settings that are chosen. A panel of buttons (5.) will allow for various settings to be selected. The simulated driven vehicle lights can be turned ON / OFF with the LIGHTS button. (Note, not all scenarios simulate nighttime materials correctly since in most cases these need to be defined in the models directly). The next 3 adjacent buttons will toggle the scenario between running in AUTO, DRIVE, or REPLAY mode (ARCHER can only be in one of these 3 modes). The AUTO mode is used to automatically run a scenario in a demo or preview mode. The path to be used is predefined by the scenario, but sometimes can be changed by modifying the setup files. A few scenarios are truly fixed and unchangeable, i.e., PA851). DRIVE mode is generally only available when the current selected combination of Subject ID (2.) and session settings (6.) have been unused (scenario has not yet been saved). The unique save name is defined by the scenario name text (1.) above. Selecting End Session while in DRIVE mode will save the session and scenario data and place ARCHER into a REPLAY session of what was just driven. Otherwise, REPLAY is only available when the Subject ID and session settings (6.) have already been driven and saved. The QUIT button will end ARCHER unless the current session (while in DRIVE mode) has not been saved in which case a popup window will be displayed confirming the quit without saving option. The Session Settings (6.) are specific to the scenario but the example above, which has a scenario ordering selection (one of R1_C1, R1_C2, R2_C1, or R2_C2) and a sign setting (one of Min, Low, or High) radio button selections, is fairly typical. These buttons often are used to select the dependent variables of the scenario. Selecting different values can be used to preview the conditions for the scenario. The very large text fields (7.) under the session variable section are used to display various scenario variables, and all except the current MPH and g-force values are scenario specific. Tile-based scenarios will display the current TILE #. This same text field is used for segment name and direction for parametric roadways and segment-based scenarios. The next section of the GUI (8.) will vary widely depending on the specific scenario. Some scenarios will list each road segment (parametric), while others will list each Tile in the experiment. The text in this section will generally depend on what is being studied and can be used to place a subjective rating entry, as in

PA851. A line of text (8.) can be used for directions to be read to the participant. This text can be made to vary based on TILE # or by a distance trigger in the scenario. The GRAPH button (10.) is only used in a few scenarios, and this button is sometimes replaced with other scenario-specific controls (see scenario examples). The VIEW CONTROLS button (9.) pops up an additional panel of Advanced View Control settings (see below).

4.4.1 Advanced View Control Panel

The View Controls panel consists of 7 distinct clusters of buttons, each with different functions related to controlling either the view or what is seen in the scene. In Figure 4.3, the set of buttons outlined in red (1.) will rotate the view of the scene (as an offset to the driven vehicles view) as incremental additions or subtractions to the current heading. Pitch, PAN LEFT, PAN RIGHT will +/- heading in degrees, PAN UP, PAN DOWN, will +/- pitch in degrees. CENTER will re-center the view on the driven vehicle's heading. The default amount moved with each press is 2.86 degrees unless the SLOW LOOK or FAST LOOK buttons have been pushed (see 2. below). The set of buttons outlined in green (3.), allow the view position to be moved UP / DOWN or +/- in Z, LEFT / RIGHT or +/- in Y, or to zero out the position offset with the ZERO button. The default amount moved with each press is .5 meters unless the SLOW LOOK or FAST LOOK buttons have been pushed (see 2. below). The first two rows of buttons outlined in blue (2.) are only effective in the AUTO and REPLAY modes of ARCHER and have no effect in the DRIVE mode. When first entering AUTO or REPLAY mode ARCHER will be paused and will not start following the predefined or loaded path until placed into the FORWARD mode. The path will be followed in reverse when the REVERSE button is pushed. Only one of these three buttons will be active at a time, and they act together modally. The speed of replay (or path following) defaults to a real-time rate (interpolation is used to match replay time) and can be adjusted by (x2) twice as fast for each press of the FASTER >> button. The SLOWER << button, will divide the speed in half (x.5) for each press. The playback rate can be reset to real-time again by pressing the x1 SPEED button.



Figure 4.3 Advanced View Control Panel

The text labels below this section (line at 4.) will display the current speed in mph, distance down the road in meters (scenario specific), the current relative time since the start session button was pressed, current view X position, Y position, Z position (elevation), current view heading (H), pitch (P), and roll (R) in degrees. The texsize value is only used for development and can be ignored.

The next line outlined in yellow (5.) will toggle various rendering options. The Sequence Capture button will allow for either single screen captures (default) or for a timed set of sequential image frame captures (i.e., a movie sequence) to be saved. The Frame Rate button will toggle between several frame rate displays on each rendering window (upper left corner). The first number displayed on the left is in frames per second (fps) averaged over 100 frames. The second, third, and fourth numbers represent the cull, draw, and update timing information in terms of fps as well. The Wire-Frame View button will toggle the scene between wire and solid rendering. The Day/Night button will toggle between using the nighttime

lighting setting and the daytime lighting setting. The sun direction values for these can be changed in the scene file (see section 6, Day/Night parameters). The High/Low Beam button will toggle between the high and low beam headlight models defined in the scene file (see section 6, Headlight parameters). The 6-digit text field (6.) allows the user to move the view to any arbitrary position (X, Y, Z) and orientation (heading - H, pitch - P, roll - R) in global coordinates (decimal number), each separated by spaces. The bottom set of buttons (7.) will toggle between rendering various parts of the scene based on category listed.

5. GENERATING ARCHER SCENARIOS – “QUICK START”

Creating an ARCHER scenario must start with understanding what is required in the scenario. For research-based scenarios, a good place to start is defining what the independent and dependent variables will be. The independent variables represent the things in the scenario that must be changed while the dependent variables will be the things that should be measured. ARCHER has been created to control what the driver experiences from the perspective of these independent variables instead of a virtual world model. In other words, we display different roadways and signs to the driver that are controlled by the independent variables in some controlled sequence. Many other simulators commonly model the world as a re-creation of our current world using graphical techniques, such as dividing the existing world into predefined Tile subdivisions. ARCHER has the capability of using “Tiles” as well, but these tiles are not fixed to a predefined terrain surface, but rather are descriptions of the driver’s current location (and experimental conditions) with the subsequent tiles defining the next location (and experimental conditions) that we want the driver to experience. It is often very useful to define the experimental conditions and place them in a table or matrix to understand all the variations that need to be controlled in the scenario. These variables combined with any optional GUI controls and other scenario-based triggers will form the basis for the scenario controls for your experiment (see Section 9.0 for detailed examples of scenario control methods).

Scenario Creation

Scenarios for the Traffic Lab Driving Simulator will be created through a combination of populating the ARCHER scene file, creating a matching VISSIM scenario, using 3D modeling tools such as Presagis Creator and imaging tools such as Adobe Photoshop. Database synchronization between VISSIM and ARCHER will be partially achieved through ARCHER reading in the VISSIM input file into ARCHER as ARCHER will read the road definitions from VISSIM. It will be important for the VISSIM roadway geometry to be created with the level of detail (number of links) that allow for sufficient detail to be used in the ARCHER geometry. Some of the steps used to create scene content in ARCHER are listed below.

- Scenario files for ARCHER will be defined by the ARCHER scene data file. The VISSIM road definition will be collaborated to the ARCHER roadway definition through utility programs.
- The scene file is a text file which can be edited to incorporate any ARCHER element including road definitions (Pavement), driveways, side roads, pavement markers, signal lights, signs, shoulders, embankments, medians, trees, elevation (terrain), buildings (models), traffic /vehicles, and events.
- OpenFlt 3D Models created with the Presagis Creator modeling tool can be included in the scene file.
- Traffic will be generated by synching the ARCHER scenario with VISSIM. The specific VISSIM data files are defined in the scene file.

5.1 ARCHER Scenario File

To create a new scenario, a new scenario file must be created. The scenario file can be created by copying the template scenario file (see Figure 5.1) and editing the following lines marked in green. Values marked in red should not be changed unless directed to by the ARCHER developers. Values in orange can be changed by advanced users. The experiment name is marked in yellow as <ExpName> and should be replaced with the new experiment name to be used. Several lines are commented out by the “#” symbol and can be uncommented to take effect. The main definition of what elements are in the scenario is defined in the scene file. Examples and methods for creating scene files are defined in Sections 10 and 11.

```

1. #####
2. #          ARCHER Scenario File template
3. #####
4. ScenarioName=<ExpName>
5. GuiProg=CarAdvancedGui
6. GVDSModel=$DVLPATH/data/common/LikeSaturn
7. WashoutModel=$DVLPATH/data/common/md150
8. GVDSStartXYZH=-1990.0 -3.0 0.53 0.0
9. InitOffsetX=-0.354
10. InitOffsetY=-0.26
11. InitOffsetZ=0.55
12. TileSize=804.672
13. NumTiles=7
14. NumSessions=191
15. DirectionsSetupFile=$DVLPATH/data/<ExpName>/<ExpName>_directionsDef.txt
16. RouteFile=$DVLPATH/data/<ExpName>/DemoRoute.csv
17. SceneFile=$DVLPATH/data/<ExpName>/<ExpName>_scene.txt
18. SubjectDataPath=$DVLPATH/data/<ExpName>/expRunData
19. #FogParameters=0.0024 0.75 0.75 0.75 0.0 FogSky
20. #FogParameters=0.0018 0.0 0.15 0.0 0.0 ClearSky
21. #TestSessions=1
22. Night=yes

```

Figure 5.1 Template Scenario File Example

- **GuiProg:** This value can be set to a new Java program that can be made custom and specific to a given experiment. The generic Gui is represented by the CarAdvancedGui value. Alternatively the CarDemoGui value can be set for a simplified interface.
- **GVDSModel:** This value should not be changed as it represents which vehicle dynamics model to use. Only the LikeSaturn model is currently supported, and AAI must be contacted to change this.
- **WashoutModel:** This value should not be changed as only the md150 value is supported.
- **GVDSStartXYZH:** This value is set for scenario start position when in drive mode.
- **InitOffsetX:** The view offset relative position X to the vehicles center point from the vehicle dynamics.
- **InitOffsetY:** The view offset relative position Y to the vehicles center point from the vehicle dynamics.
- **InitOffsetZ:** The view offset relative position Z to the vehicles center point from the vehicle dynamics.
- **GuiHostPort:** The GUI host port can be specified to allow the GUI to run on a different machine from the machine where ARCHER is launched.
- **SubjectDataPath:** The Subject Data Path specifies where collected data and saved data should be stored.
- **ScenarioName:** This is used as the prefix for save file names.
- **TestSessions:** This can be set to allow restricted settings in the GUI to be changed after start session is set (such as changing caseIDs during a session).
- **ScenarioControlFile:** (or *ScenarioControlFile:) This value will setup advanced scenario trial controls defined in Section **9.0 SCENARIO & EXPERIMENTAL CONTROL**.
- **DirectionsSetupFile:** (or *DirectionsSetupFile:) This value will setup directions that will be displayed in the GUI (See Section **9.0 SCENARIO & EXPERIMENTAL CONTROL**.)
- **NoDrive:** Setting this value will not allow Drive mode to be used during the session. This can be used to force replay only (a.k.a. read only) mode.
- **SceneFile:** The Scene file defines what SceneObjects are in the scenario. The scene file also defines scenario controls and events to be defined during the scenario.

- **NumTiles:** Tile-based scenarios need the number of tiles defined here for GUI operation.
- **TileSize:** Tile-based scenarios need to define the size of the Tile here for correctly reporting tile information during execution.
- **NumSessions:** Multi-Session-based scenarios will define the number of sessions used here for GUI control.
- **CaptureTime:** The capture time variable will define how many seconds to capture a sequence of images for when the sequence capture button is selected. A value of zero used here will capture a single image.
- **ColumnIconMode:** This value will setup a Button box populated with image-based icons. This mode is used in conjunction with a scenario control file for operator scoring a scenario run based on trials (See **Section 9.0 SCENARIO & EXPERIMENTAL CONTROL.**)
- **ColumnIcon:** This defines a button icon to be used in Column Icon mode (See **Section 9.0 SCENARIO & EXPERIMENTAL CONTROL.**)
- **NumPracticeTrials:** This value is used with the ColumnIconMode to define a set of practice trials to use.
- ***<any text=>:** Any parameter which begins with an "*" will be treated as a launch parameter. If the value is not otherwise interpreted, it will be recorded in the saved data file. For example, a value of *Gender=Male will print out Gender = Male in the saved data file. Below are listed several special launch params, which are further interpreted.
- ***Participant #:** Set participant # to this value.
- ***SessionId:** Set the sessionId value to this #.
- ***Maneuver Order:** A deprecated way of setting the CaseID (used in some of the older GUIs)

5.2 The Scene File and ARCHER Scene Elements

An ARCHER scenario uses the ARCHER scene file to define what should be rendered and how to control the scenario (the scene file is defined in the scenario, see Section 5.1 above). The scene file is organized by lists of scene object types (definitions for types of scene objects, i.e., a Ford Mustang type model) and scene object instances (i.e., actual objects to render, my Ford Mustang parked on the side of the road). Each definition or set of parameters in the scene file will be preceded by a keyword to define the Scene Object or attributes which follow. The list of Registered Types of Scene Objects is defined below. Additional definitions for the parameters will be defined in each types section later.

Model	A 3D model file to be loaded and positioned / rotated by given parameters
Embankment	A continuous, and road relative cross-section, which can use an image
Fleet	A set of paths used to generate vehicle traffic
Headlight	Light parameters to define headlight modeling
LineMarking	Line marking for a specific RoadSurface (below), requires the RoadSurface
OnRoadSign	Horizontal signage on the given RoadSurface, e.g., Arrow signs painted on the road
OrderedGroups	A scenario method of defining a random ordering of objects such as road segments
RoadSurface	A parametrically defined roadway based on a cross section data
RRPM	Retro-Reflective Pavement Markings setup on a RoadSurface
Sunlight	Parameters to define Day and Night lighting levels
Terrain	Elevation Terrain patch parameters, can be USGS data or other digital elevation data
TickMark	Transverse pavement markings used for speed calming
TiledModel	A scenario method of defining a tile that will be repeatedly driven through
Tree	A road relative object using the SpeedTreeRT library
RoadRelativeObj	Placement of one or many objects (random patches) relative to the road

ARCHER Scene Elements (images for section 5.3)

ARCHER scenarios can have various scene elements, such as the ones defined below, added to any scenario through the scene construction process described above. A quick start (one-day) training session for the University of Utah Traffic Lab staff will be presented after delivery and setup. Various examples of these scene elements are shown graphically in the images below.

**Roads (Pavement),
driveways, side roads
Pavement Markers
Signal Lights
Signs
Shoulders
Embankments
Medians
Trees
Elevation (Terrain)
Buildings
Traffic / Vehicles
Events**



Besides the registered types listed above, certain keywords can either modify a scene object or define additional parameters for special use. Below are listed these other keywords that can be used.

TYPE: The TYPE: keyword can be used for some types of Scene Objects (e.g., Model type) to define a class of object that can then be referenced for specific instances. In this way, the general parameters, which are reused for a scene object, will be defined by the type (Model filename), but the instance parameters such as position and orientation will be defined for the specific instance.

STATE: The STATE: keyword can be used after any defined Scene Object to define multiple states a scene object can be in. This will allow for different state representations to be selected as needed in the scenario control. An example of this is for a traffic light that exists in one of three states: green, amber, or red. Then an event can be defined that refers to these states.

EVENT: An event defines processing to occur based on something being triggered. Different kinds of triggers can be based on time, distance, or by pushing a GUI button. Events, when triggered, will change states based on a timing diagram.

MATERIAL_TABLE: The Material Table can define materials to use for various rendering parameters used primarily with the Embankment, LineMarking, RRPM, RoadSurface, and OnRoadSign Registered Scene Object Types. Refer to each type's parameter description for how they are used.

STARTUP_PARAM: The Startup Parameters will define various startup defaults such as Time of Day, Headlights On/OFF, and some other initial rendering states.

5.3 ARCHER Scenario Types

ARCHER scenario controls can be based on one of three different mechanisms: parametric-roadway-based (using a centerline), trial-based tiling (as defined above), or road segment-based. Any of these methods can be combined with the other methods to form a hybrid approach. Each of these approaches has different advantages depending on what is required for the scenario.

6. THIRD-PARTY TOOLS

Various third-party tools are required for creating ARCHER scene elements. These tools are modeling tools as well as image-based and other terrain geo-specific applications, which can assist in generating the scenario as required. 3D modeling tools such as Presages Creator, Maya, 3D Studio, and Google Sketchup can be used to create models defined in the scene file. The Presages Creator tool is most commonly used in military and civilian real-time simulations while Maya and 3D Studio are most commonly used in game creation. Image generation and editing tools such as Photoshop, Corel Draw, and Adobe Illustrator are used to create image-based textures that are used both by the 3D modeling tools listed above as well as in several of the other ARCHER non-model-based scene elements such as the embankment or road scene objects. Specialized tools such as Google Earth and other GIS tools (i.e., ArcGIS) can be used to help define geo-specific locations and can aid in generating elevation terrain objects. These tools can also be used in checking for geo-typical look and feel for an ARCHER scenario.

7. SCENARIO AND EXPERIMENTAL CONTROL

Research-based scenarios generally require parameters that define the trials to be driven by the participants. These parameters can be represented by using several techniques to control the roadway or scene variations; the specific method will depend on the research requirements. First, a decision must be made on the main scenario control mechanism, which can be categorized as either trial-based tiling, segment-based road variations, road relative-based, or some combination. Secondary to this will be defining different cases to represent elements of the scene. These cases can be used to represent different trials or control cases needed by the study. The cases are often changed between participants or can be used to control the trials presented to the driver in a specific order. For trial-based tiles, the cases can be control-based on the number of tiles the driver has driven. For example, on Tile 10 we want to have a special emergency signal go off for the intersection in the middle of the tile. Additionally, a scenario control file can be defined to help set various states of scene elements.

8. ADVANCED SCENARIO CREATION – THE CFI VISSIM SCENARIO, AN EXAMPLE

The syntax for defining the CFI VISSIM scene is defined below, and the following text shows the breakdown of the contents of this scene file, section by section. The full scene file can be found in Attachment 4. The scene will be self documenting through comments as any line starting with a C++ comment “//”, a double slash will be ignored for processing. Additionally, a “#” symbol at as the first character of a line will be ignored. In general, these comments must be started at the beginning of the line. It is recommended that a comment block at the beginning of the file should identify the scenario name and the organization producing the scenario. A date stamp and author field can also be defined. Other syntax requirements are that, unless otherwise specified, no parameters should contain spaces and names should be single words or joined by an underscore. In general, similar syntax rules to C/C++ coding should be followed with strict definitions for what parameters are expected for specific scene object definitions.

```
// *****
// Highway Driving Simulator Team, AAI Engineering Support, Inc.
// Developed for Human Center Systems, Federal Highway Administration.
// *****
// This file is for defining the scene for at runtime for an Adaptable
// Rendering Cluster for Highway Environment Research or ARCHER.
// *****
// *****
// This file will list definitions of SceneObjectTypes and instances of Scene
// Objects. The formats are as follows:...
//
// Valid types and params:
// For prototype useage. Later use Prototype_name instead of a SceneObj Type
//
// TYPE: "SceneObj_Type_Name" "Prototype_name" "model FileSpec" XX_offset YY_offset ZZ_offset Heading_offset
Pitch_offset Roll_offset
// or ...
// TYPE: "SceneObj_Type_Name" "Prototype_name" "model FileSpec" XX_offset YY_offset ZZ_offset (h,p,r defaults
to 0.0 0.0 0.0)
// or ...
// TYPE: "SceneObj_Type_Name" "Prototype_name" "model FileSpec" (x,y,z,h,p,r defaults to 0.0 0.0 0.0 0.0 0.0
0.0)
//
// Valid SceneObject types (based on registered types in subclasses of CarSceneObjCreator.
// MODEL ---- for CarModelCreator class
// MODEL "Instance name" "Prototype_name" "model FileSpec" XX_position YY_position ZZ_position Heading Pitch
Roll
// or ...
// MODEL "Instance name" "Prototype_name" "model FileSpec" XX_position YY_position ZZ_position (h,p,r defaults
to 0.0 0.0 0.0)
// or ...
// MODEL "Instance name" "Prototype_name" "model FileSpec" (x,y,z,h,p,r defaults to 0.0 0.0 0.0 0.0 0.0 0.0)
//
// STATE formats:
// STATE <STATE_TYPE> <state group name> <switchnode name> <default state>
// stateName IndexValue
//
// (NOTE: scaling is not currently implemented)
// *****
// PROTOTYPES (TYPES:) need to come before they are used
// -----
```

The beginning of this file lists the syntaxes and examples used for defining scene object prototypes and scene object instances. Following that, and before any object uses them, the material table must be defined. The material table is based on OpenGL material modeling, and the OpenGL programming guide can be referenced for what the specific mathematical model of what the different parameters do; but in general, the **Ambient** component represents the global ambient light in the environment. The **Diffuse** component represents how light reflects from the material with direct sources shining on it. The **Emission** (emmissive light) component represents material that gives off its own light. The **Specular** component represents the material having a reflective specular hotspot (highlights). The **Alpha** will adjust the

apparent transparency of the material, with a value of 1.0 being totally opaque and 0.0 being totally transparent.

```

// *****
// -----
// Define shared materials. (This is not a scene-object.)
// -----
// type name: MATERIAL_TABLE:
// -----
// parameters:
//
// MaterialName=name
// Ambient rr gg bb [default 0.2 0.2 0.2]
// Diffuse rr gg bb [default 0.8 0.8 0.8]
// Emission rr gg bb [default 0.0 0.0 0.0]
// Specular rr gg bb [default 0.0 0.0 0.0]
// Alpha ff [default 1.0]
// Shininess ff [default 0.0]
//
// Note: all values are within [0,1].
// -----

```

The next section of the scene file includes prototype definitions that will be used as individual scene object instances later in the file. A prototype can potentially be of any scene object type, but in general, prototypes are mostly **Model** types as are used here. A **Model** prototype can be defined by specifying the keyword **TYPE:** followed by the **ARCHER** scene object type name, then the prototype name (later used to create an instance of the prototype), followed by the additional specific parameters for that scene object type. In the case of the **Model** type, the path to the model file should be defined. Then optionally an **X-Offset, Y-Offset, Z-Offset, Heading, Pitch,** and **Roll** offsets can be specified. Every instance of this **TYPE** (prototype).

```

// -----
// Prototype section:
//
// TYPE: "SceneObj_Type_Name" "Prototype_name" "model FileSpec" XX_offset
// YY_offset ZZ_offset Heading_offset Pitch_offset Roll_offset
// -----
TYPE: MODEL CFI_E2W /hdsshare/data/CFI/FlightFolderModels/CFI_E2W14.flt
TYPE: MODEL ALT_CFI_N2S /hdsshare/data/CFI/FlightFolderModels/ALT_CFI_S2N14.flt
TYPE: MODEL CFI_N2S /hdsshare/data/CFI/FlightFolderModels/CFI_S2N27.flt
TYPE: MODEL CFI_Signals /hdsshare/data/CFI/FlightFolderModels/Signals/CFI_TrafficSignalHousingCFI.flt
TYPE: MODEL CFI_Signals_Hi /hdsshare/data/CFI/FlightFolderModels/Signals/CFI_TrafficSignalHousingCFI_hi.flt
TYPE: MODEL Conv_Signals /hdsshare/data/CFI/FlightFolderModels/Signals/TrafficSignalHousingconv.flt
TYPE: MODEL CFI_Signs /hdsshare/data/CFI/FlightFolderModels/Signs/CFIsigns.flt
TYPE: MODEL CFI_Signs_Alt /hdsshare/data/CFI/FlightFolderModels/Signs/cfi_signs_alt.flt
TYPE: MODEL BLDsigns /hdsshare/data/CFI/FlightFolderModels/Signs/cfi_BuildingSigns.flt
TYPE: MODEL BLDMOD /hdsshare/data/CFI/FlightFolderModels/buildings/buildings_placed.flt
TYPE: MODEL W4_2L_W /hdsshare/data/CFI/FlightFolderModels/Signs/cfi_W_jersey.flt
TYPE: MODEL W4_2L_E /hdsshare/data/CFI/FlightFolderModels/Signs/cfi_E_jersey.flt
#TYPE: MODEL W4_2L_N /hdsshare/data/CFI/FlightFolderModels/Signs/cfi_N_jersey.flt
#TYPE: MODEL W4_2L_S /hdsshare/data/CFI/FlightFolderModels/Signs/cfi_S_jersey.flt

```


The next model instance is of the ownership driven Saturn car. This is visible only in the mirror displays and in the plan views/lane view rendering windows.

```
// *****
// SCENE OBJECT INSTANCES
// -----
MODEL CLOUDS /hdsshare/data/CFI/FlightFolderModels/facadeMT.flt 0 0 0 0 0 0
Background
TerrainButton
MODEL SaturnCar /hdsshare/data/RCars/SaturnLPv8.flt 0.33 0.08 -2.97 90 0 0
UnderOwnship
// Tagging Scene Objects to Turn On / Off with these GUI buttons.
//
// RoadButton
// TreeButton
// TerrainButton
// RelObjectButton
// EmbankmentButton
// Signs1Button
// Signs2Button
// SignalsButton
// PMButton
// OtherButton
```

In the next set of comments, list tags can be used to associate with some of the GUI buttons for turning on/off categories of objects. This tag should be placed after the full scene object definition, and is optional. Many of the predefined scene object types will automatically be classified as terrain, trees, roadway, relative objects, or embankment geometry. Model geometry has to be tagged, as it can be used for any of these purposes.

The next section of the file defines the first model instance to be placed in the scene. This object is the street signs as they are located in the scene (hence the 0,0,0,0,0,0 position, orientation attributes as the model file is already positioned to line up with roadway geometry correctly. The instance name given to this object is streetsign1 and we give it CaseId's of 1, 5, and 9. This means the instance will be displayed only when CaseId's 1, 5, or 9 are active. These caseId's can be set through the GUI interface, launchparams or through a trial-based scenarioControl file. CaseId's can also be used in trial-based tiling that are active when you are on a given tile, for example, show a given sign only when the driver is on Tile 4. In this case, the CaseId is set by a launchparam. The instance is tagged to be toggles on / off with the OtherButton on the GUI. A second model instance defines a grass plane to draw, which is a flat terrain piece that will underlay the roadway and other off-road features.

```
MODEL streetsigns1 /hdsshare/data/CFI/FlightFolderModels/Signs/cfi_streetsigns_sl.flt 0.0 0.0 0.0 0.0 0.0 0.0
CaseId 1 5 9 CaseEnd
OtherButton
MODEL grass /hdsshare/data/CFI/FlightFolderModels/grassplane.flt 0.0 0.0 0.0 0.0 0.0 0.0
EmbankmentButton
```

Next, we define a roadway to be used to place trees along. SpeedTree objects can be placed along a roadway as a road relative object. We define an invisible road definition and use it for each roadway section in which we want to place trees.

```
// -----
// Road
// -----
// type name: RoadSurface
// -----
// valid properties:
// RoadId=name
// DmvRoadDefinition=file roadWidth offCenter ## deprecated
// DmvRoadTexture=pnm-file refCoordX refCoordY resolX resolY
// DhMRoadDefinition=file numLanes
// MaxEdgeWidth=meters
// MaxSegmentsPerNode=number
// -----
//TREES East (Tiles 8-1-2) along SR 123 Madison
```

```

// This road surface allows us to have road relative tress created
RoadSurface
RoadId=CFI_RD_E
DhmRoadDefinition=/hdsshare/data/CFI/Centerline_CFI_1_meter.dhm 2
MaxEdgeWidth=1.0
DmvRoadTexture=/hdsshare/DSCN008Xct3.ppm 0 0 0.004 0.0055
RoadTextureScheme=Tiling 0 0
MaxSegmentsPerNode=300
TYPE: TREE TREEROAD
TREEROAD CFI_RD_E INDIVIDUAL 6.5
TYPE: TREE TREE_6_2XL
Model=/hdsshare/data/PennDOT/scene/Models/trees/S_TREES/Tree3XXL/firtree_RT2_XXL.spt
TexLeaf=/hdsshare/data/PennDOT/scene/Models/trees/S_TREES/Tree3XXL/CompositeMap.tga
TexBark=/hdsshare/data/PennDOT/scene/Models/trees/S_TREES/Tree3XXL/CedarBark02.jpg
ObjMode=REPEATING
Size=6.0
SizeVariance=5.0
LeafLODs=450,650
BranchLODs=450,600
//Left is Right and Right is Left - The L and R are REVERSED
//Distance going down the road, randomness value, Side L / R
//Distance from the road, Rotation, Size, Z - Height
//TREE_6_2XL tree62XL 0 400 70 11111 L 0.0 0.5 1.0 // REPEATING MODE
//TREE_6_2XL tree62XLa1 000 400 20 88523 L 24.0 1.0 1.0 0.5
//TREE_6_2XL tree62XLb1 000 400 20 21433 R 8.0 1.0 1.0 0.5
TREE_6_2XL tree62XLa2 1050 1750 65 22222 L 24.0 5.0 1.0 0.5
TREE_6_2XL tree62XLa2 850 1750 75 23332 L 64.0 1.0 2.0 0.25
TREE_6_2XL tree62XLa2 750 1750 99 23232 L 108.0 1.0 2.5 0.0
TREE_6_2XL tree62XLb2 1050 1750 65 44444 R 8.0 1.0 1.0 0.5
TREE_6_2XL tree62XLb2 850 1750 75 43334 R 56.0 1.0 2.0 0.25
TREE_6_2XL tree62XLa2 750 1750 99 23432 R 104.0 1.0 2.5 0.0
TREE_6_2XL tree62XLa3 1900 2750 65 33333 L 24.0 1.0 1.0 0.5
TREE_6_2XL tree62XLa3 1900 2750 75 34443 L 64.0 1.0 2.0 0.25
TREE_6_2XL tree62XLa3 1900 2750 99 34343 L 108.0 1.0 2.5 0.0
TREE_6_2XL tree62XLb3 1900 2750 65 66566 R 16.0 1.0 1.0 0.5
TREE_6_2XL tree62XLb3 1900 2750 75 65556 R 56.0 1.0 2.0 0.25
TREE_6_2XL tree62XLa3 1900 2750 99 65566 R 104.0 1.0 2.5 0.0
TYPE: TREE TREEROAD
TREEROAD CFI_RD_E INDIVIDUAL 2.5
TYPE: TREE TREE_61_2XL
Model=/hdsshare/data/PennDOT/scene/Models/trees/S_TREES/Tree3XXL/firtree_RT2_XXL.spt
TexLeaf=/hdsshare/data/PennDOT/scene/Models/trees/S_TREES/Tree3XXL/CompositeMap.tga
TexBark=/hdsshare/data/PennDOT/scene/Models/trees/S_TREES/Tree3XXL/CedarBark02.jpg
ObjMode=REPEATING
Size=6.0
SizeVariance=5.0
LeafLODs=100560,100650
BranchLODs=450,600
TREE_61_2XL tree62XLa2 750 1750 39 23232 L 258.0 1.0 2.5 0.0
TREE_61_2XL tree62XLa2 750 1750 99 23432 R 254.0 1.0 2.5 0.0
TREE_61_2XL tree62XLa3 1900 2750 39 34343 L 258.0 1.0 2.5 0.0
TREE_61_2XL tree62XLa3 1900 2750 99 65566 R 254.0 1.0 2.5 0.0
//TREES North (Tiles 2-3-4) along SR 309 Dominion
// This road surface allows us to have road relative tress created
RoadSurface
RoadId=CFI_RD_N
DhmRoadDefinition=/hdsshare/data/CFI/Centerline_CFI_2_meter.dhm 2
MaxEdgeWidth=1.0
DmvRoadTexture=/hdsshare/DSCN008Xct3.ppm 0 0 0.004 0.0055
RoadTextureScheme=Tiling 0 0
MaxSegmentsPerNode=300
TYPE: TREE TREEROAD
TREEROAD CFI_RD_N INDIVIDUAL 6.5
TYPE: TREE TREE_6_3XL
Model=/hdsshare/data/PennDOT/scene/Models/trees/S_TREES/Tree3XXL/firtree_RT2_XXL.spt
TexLeaf=/hdsshare/data/PennDOT/scene/Models/trees/S_TREES/Tree3XXL/CompositeMap.tga
TexBark=/hdsshare/data/PennDOT/scene/Models/trees/S_TREES/Tree3XXL/CedarBark02.jpg
ObjMode=REPEATING
Size=7.0
SizeVariance=6.0
LeafLODs=450,700
BranchLODs=450,600
//TREE_6_2XL tree62XL 0 400 70 11111 L 0.0 0.5 1.0 // REPEATING MODE
//Left Side
//TREE_6_3XL tree63XLa2 0 350 60 22222 L 24.0 5.0 1.0 0.5
TREE_6_3XL tree63XLa2 0 565 70 23332 L 64.0 1.0 2.0 0.25
TREE_6_3XL tree63XLa2 0 565 99 23232 L 118.0 1.0 1.5 0.0
//Right Side
//TREE_6_3XL tree63XLb2 0 250 60 44444 R 12.0 1.0 1.0 0.5
TREE_6_3XL tree63XLb2 0 540 70 43334 R 56.0 1.0 2.0 0.25
TREE_6_3XL tree63XLa2 0 540 99 23432 R 124.0 1.0 2.5 0.0

```

```

//TREE_6_3XL tree63XLa2 750 1750 60 22222 L 24.0 5.0 1.0 0.5
TREE_6_3XL tree63XLa2 630 1750 70 23332 L 64.0 1.0 2.0 0.25
TREE_6_3XL tree63XLa2 630 1950 99 23232 L 118.0 1.0 1.5 0.0
//TREE_6_3XL tree63XLb2 750 1750 60 44444 R 8.0 1.0 1.0 0.5
TREE_6_3XL tree63XLb2 600 1750 70 43334 R 56.0 1.0 2.0 0.25
TREE_6_3XL tree63XLa2 600 1800 99 23432 R 124.0 1.0 2.5 0.0
//TREE_6_3XL tree63XLb2 2150 2750 60 45554 L 24.0 1.0 1.0 0.5
TREE_6_3XL tree63XLb2 2050 2750 70 43534 L 64.0 1.0 2.0 0.25
TREE_6_3XL tree63XLa2 2050 2850 99 45454 L 108.0 1.0 2.5 0.0
//TREE_6_3XL tree63XLb2 2150 3000 60 45554 R 12.0 1.0 1.0 0.5
TREE_6_3XL tree63XLb2 2050 3000 70 43534 R 56.0 1.0 2.0 0.25
TREE_6_3XL tree63XLa2 2050 3000 99 45454 R 124.0 1.0 2.5 0.0
TYPE: TREE TREEROAD
TREEROAD CFI_RD_N INDIVIDUAL 2.5
TYPE: TREE TREE_61_3XL
Model=/hdsshare/data/PennDOT/scene/Models/trees/S_TREES/Tree3XXL/firtree_RT2_XXL.spt
TexLeaf=/hdsshare/data/PennDOT/scene/Models/trees/S_TREES/Tree3XXL/CompositeMap.tga
TexBark=/hdsshare/data/PennDOT/scene/Models/trees/S_TREES/Tree3XXL/CedarBark02.jpg
ObjMode=REPEATING
Size=6.0
SizeVariance=5.0
LeafLODs=100560,100650
BranchLODs=450,600
//TREE_6_2XL tree62XL 0 400 70 11111 L 0.0 0.5 1.0 // REPEATING MODE
//Left Side
TREE_61_3XL tree63XLa2 0 565 39 23232 L 258.0 1.0 1.5 0.0
//Right Side
TREE_61_3XL tree63XLa2 0 540 99 23432 R 124.0 1.0 2.5 0.0
TREE_61_3XL tree63XLa2 660 1950 39 23232 L 278.0 1.0 1.5 0.0
TREE_61_3XL tree63XLa2 600 1800 99 23432 R 124.0 1.0 2.5 0.0
TREE_61_3XL tree63XLa2 2050 2850 39 45454 L 268.0 1.0 2.5 0.0
TREE_61_3XL tree63XLa2 2050 3000 99 45454 R 124.0 1.0 2.5 0.0

```

Next are roadway section definitions. The scenario controls required a different ordering for counterbalancing the study. That is done with the scene order CaseId as documented below. Each Tile is named as an instance name of Tile1, Tile2, ..., Tile8. Each of these roadway models are tagged with the RoadButton for the GUI to manipulate. The road was divided into N2S, north to south, and E2W, east to west, sections for modeling and reuse as most of the E2W road definition and lanes are common to both types of roadways.

```

// Now we define the roadway sections Tile by Tile
// SIGNALS
// SCENE ORDER 1 RegOrder=1, CFIOrder=1
// ScenarioCases=1,5,9 RegOrder=1(Bangerter first), CFIOrder=1(CrossOver 1st)
// Where 1=Min Signage, 5=Low Cost Signage, 9=Hi / Aggressive Signage

CFI_E2W tile1_e2w 277.5031 0.0 0.0 180.0 0.0 0.0
CaseId 1 5 9 CaseEnd
RoadButton
ALT_CFI_N2S tile1_n2s 277.5031 0.0 0.0 180.0 0.0 0.0
CaseId 1 5 9 CaseEnd
RoadButton

```

Next, we document the signals from both the VISSIM signal controller ids and the Signal Groups for each signal group controlled by VISSIM.

```

// VISSIM Signal ControllerID = 3
// Left from EB. Forest to NB. Madison Signal Group = 7 (2 Lanes) (Index 6 is green here) (7 is Yellow)
(others = red)
// Straight on EB. Forest Signal Group = 4 (3 lanes) (Index 4 is green here) (5 is Yellow)
(others = red)
// Right from EB. Forest to SB. Madison Signal Group = ???
//-----
// Left from WB. Forest to SB. Madison Signal Group = 3 (2 Lanes) (Index 6 is green here) (7 is Yellow)
(others = red)
// Straight on WB. Forest Signal Group = 8 (3 Lanes) (Index 4 is green here) (5 is Yellow)
(others = red)
// Right from WB. Forest to NB. Madison Signal Group = ???
//-----
// Left from NB Madison to WB Forest Signal Group = 5 (2 Lanes) (Index 2 is green here) (3 is Yellow)
(others = red)
// Straight on NB Madison Signal Group = 2 (3 Lanes) (Index 0 is green here) (1 is Yellow)
(others = red)
// Right from NB Madison to EB Forest Signal Group = ???
//-----
// Left from SB Madison to EB Forest Signal Group = 1 (2 Lanes) (Index 2 is green here) (3 is Yellow)
(others = red)
// Straight on SB Madison Signal Group = 6 (3 Lanes) (Index 0 is green here) (1 is Yellow)
(others = red)
// Right from SB Madison to WB Forest Signal Group = ???

```

Now we create the instance of the tile1 signal model. This model contains switches in the geometry that we connect to via the STATE: scene object instance. The states are defined by a single letter to represent each signal group value of R|G|Y as a value string. Each of these value strings are attached to the model switches by setting the switches index. We name the STATE lights and connect the scene graph switch node, which is named switch in the flt file.

Next, we create an event type for how and when to switch the Lights named STATE. There are multiple types of events available, timed, distance-based, as well as the VISSIM_SIG type event used here. The VISSIM signal type event will map the states listed above to the VISSIM signal controller and group by specifying the Signal Group after the instance name of this event (here it is 3). The last parameter is used for data collection and tells which host to use for data collection. Any host name can be used, or all will collect data on all computers that have a scene. The listed values map the STATE by name (Lights here) to its Bit position to the VISSIM group #. This is down for each tile.

```

Conv_Signals tile1_signals 277.5031 0.0 0.0 180.0 0.0 0.0
CaseId 1 5 9 CaseEnd
SignalsButton

STATE: MODEL Lights switch 10
RRRRRRRR 0
RRYRRYRR 1
RRRRRRGG 2
RRRRRRYY 3
RGRRRRRR 4
RYRRYRRR 5
GRRRRRRR 6
YRRYRRRR 7
RRRRRRRR 8
GGGGGGGG 9

//Signal Tile 1
// EVENT: VISSIM_SIG <NAME> <SC_id>
EVENT: VISSIM_SIG SIGNAL1 3 all
// <StateName> SG# bit to set
// Expected mappings to
Lights 1 7
Lights 2 8
Lights 3 2
Lights 4 3
Lights 5 4
Lights 6 6
Lights 7 1
Lights 8 5

```

The next section defines some additional signs and the buildings in the tile.

```

CFI_Signs_Alt tile1_signs 277.5031 0.0 0.0 180.0 0.0 0.0
CaseId 1 5 9 CaseEnd
Signs1Button
BLDsigns tile1_bldsigns 277.5031 0.0 0.0 180.0 0.0 0.0
CaseId 1 5 9 CaseEnd
RelObjectButton
BLDMOD tile1_bldgs 277.5031 0.0 0.0 180.0 0.0 0.0
CaseId 1 5 9 CaseEnd
RelObjectButtonCFI_E2W tile2_e2w 415.6597 1100.0 0.0 180.0 0.0 0.0
CaseId 1 5 9 CaseEnd
RoadButton

```

The command below will preload all models so that we do not have caching in while driver and experience hiccups while doing I/O during the drive.

```

// define these so that they are in view when pre-rendering
STOP_RENDERING
T_Scene1_Gaps sgl_1 300.0 -1070.0 -5.0 0.0 0.0 0.0
T_Scene1_Gaps sgl_2 -620.0 -180.0 -5.0 90.0 0.0 0.0
T_PM8_Low pl_f -10 80 -5 -90 0 0
.....
.....
CFI_E2W tile1_e2w_r 240 240 -20 45 0 0
RESUME_RENDERING

```

Below are some definitions for the headlight modeling. This scenario does not try to do any night modeling and therefore is not used.

```

// -----
// Headlight
// -----
// type name: Headlight
// -----
// valid properties:
// HeadlightId=name
// UnderOwnership # if appears, must be the only token in a line
// LowBeamColor=red(%f) green(%f) blue(%f) alpha(%f)
// LowBeamAmbientColor=red(%f) green(%f) blue(%f) alpha(%f)
// LowBeamSpecularColor=red(%f) green(%f) blue(%f) alpha(%f)
// LowBeamPosition=x(%f) y(%f) z(%f)
// LowBeamDirection=vx(%f) vy(%f) vz(%f)

```

```

// LowBeamConeAngle=degreeAngle(%f)
// LowBeamFocusAttenuation=0-128(%f)
// LowBeamDistanceAttenuation=constant(%f) linear(%f) quadric(%f)
// LowBeamFalloff=0-1(%f)
// LowBeamReflectColor=red(%f) green(%f) blue(%f) alpha(%f)
// LowBeamReflectAmbientColor=red(%f) green(%f) blue(%f) alpha(%f)
// LowBeamReflectSpecularColor=red(%f) green(%f) blue(%f) alpha(%f)
// LowBeamReflectPosition=x(%f) y(%f) z(%f)
// LowBeamReflectDirection=vx(%f) vy(%f) vz(%f)
// LowBeamReflectConeAngle=degreeAngle(%f)
// LowBeamReflectFocusAttenuation=0-128(%f)
// LowBeamReflectDistanceAttenuation=constant(%f) linear(%f) quadric(%f)
// LowBeamReflectFalloff=0-1(%f)
// HighBeamColor=red(%f) green(%f) blue(%f) alpha(%f)
// HighBeamAmbientColor=red(%f) green(%f) blue(%f) alpha(%f)
// HighBeamSpecularColor=red(%f) green(%f) blue(%f) alpha(%f)
// HighBeamPosition=x(%f) y(%f) z(%f)
// HighBeamDirection=vx(%f) vy(%f) vz(%f)
// HighBeamConeAngle=degreeAngle(%f)
// HighBeamFocusAttenuation=0-128(%f)
// HighBeamDistanceAttenuation=constant(%f) linear(%f) quadric(%f)
// HighBeamFalloff=0-1(%f)
// HighBeamReflectColor=red(%f) green(%f) blue(%f) alpha(%f)
// HighBeamReflectAmbientColor=red(%f) green(%f) blue(%f) alpha(%f)
// HighBeamReflectSpecularColor=red(%f) green(%f) blue(%f) alpha(%f)
// HighBeamReflectPosition=x(%f) y(%f) z(%f)
// HighBeamReflectDirection=vx(%f) vy(%f) vz(%f)
// HighBeamReflectConeAngle=degreeAngle(%f)
// HighBeamReflectFocusAttenuation=0-128(%f)
// HighBeamReflectDistanceAttenuation=constant(%f) linear(%f) quadric(%f)
// HighBeamReflectFalloff=0-1(%f)
// -----
Headlight
HeadlightId=OwnshipHeadlight
UnderOwnship
LowBeamColor=0.6 0.6 0.5 1.0
//LowBeamColor=0.0 0.0 0.8 1.0
LowBeamAmbientColor=0.0 0.0 0.0 1.0
LowBeamSpecularColor=0.0 0.0 0.0 1.0
LowBeamPosition=0.5 -10.0 2.5
LowBeamDirection=0.0 0.998 -0.06162
LowBeamConeAngle=110.0
LowBeamFocusAttenuation=128.0
LowBeamDistanceAttenuation=0.02 0.00009 0.000008
LowBeamFalloff=.2
HighBeamColor=0.7 0.7 0.6 1.0
//HighBeamColor=0.0 0.0 1.0 1.0
HighBeamAmbientColor=0.0 0.0 0.0 1.0
HighBeamSpecularColor=0.0 0.0 0.0 1.0
HighBeamPosition=0.5 -10.0 2.5
HighBeamDirection=0.0 0.998 -0.06162
HighBeamConeAngle=110.0
HighBeamFocusAttenuation=128.0
HighBeamDistanceAttenuation=0.02 0.00009 0.000008
HighBeamFalloff=.2
LowBeamReflectColor=0.0 0.0 0.0 1.0
LowBeamReflectAmbientColor=0.0 0.0 0.0 1.0
LowBeamReflectSpecularColor=1.0 1.0 0.9 1.0
//LowBeamReflectSpecularColor=1.0 0.0 0.0 1.0
LowBeamReflectPosition=0.5 -10.0 2.5
LowBeamReflectDirection=0.0 0.9848 -0.1737
LowBeamReflectConeAngle=100.0
LowBeamReflectFocusAttenuation=128.0
LowBeamReflectDistanceAttenuation=0.00015 0.0005 0.0000025
LowBeamReflectFalloff=.2
HighBeamReflectColor=0.0 0.0 0.0 1.0
HighBeamReflectAmbientColor=0.0 0.0 0.0 1.0
HighBeamReflectSpecularColor=1.0 1.0 1.0 1.0
//HighBeamReflectSpecularColor=1.0 0.0 0.0 1.0
HighBeamReflectPosition=0.5 -10.0 3.5
HighBeamReflectDirection=0.0 0.9848 -0.1737
HighBeamReflectConeAngle=130.0
HighBeamReflectFocusAttenuation=128.0
HighBeamReflectDistanceAttenuation=0.0001 0.00035 0.0000025
HighBeamReflectFalloff=.2

```

Sunlight modeling levels are set in the following section. The values here can be changed to represent different sun directions and light levels. Two different levels can be represented. Here a night-time level is defined along with a daytime near 1 PM value.

```
// -----
// Sunlight
// -----
// type name: Sunlight
// -----
// valid properties:
// SunlightId=name
// SunDirection=vx(%f) vy(%f) vz(%f)
// SunLightAmbient=red(%f) green(%f) blue(%f) alpha(%f)
// SunLightDiffuse=red(%f) green(%f) blue(%f) alpha(%f)
// SunLightSpecular=red(%f) green(%f) blue(%f) alpha(%f)
// GlobalAmbient=red(%f) green(%f) blue(%f) alpha(%f)
// SkyColor=red(%f) green(%f) blue(%f) alpha(%f)
// SunDirectionNight=vx(%f) vy(%f) vz(%f)
// SunLightAmbientNight=red(%f) green(%f) blue(%f) alpha(%f)
// SunLightDiffuseNight=red(%f) green(%f) blue(%f) alpha(%f)
// SunLightSpecularNight=red(%f) green(%f) blue(%f) alpha(%f)
// GlobalAmbientNight=red(%f) green(%f) blue(%f) alpha(%f)
// SkyColorNight=red(%f) green(%f) blue(%f) alpha(%f)
// -----
Sunlight
SunlightId=TheSun
PlanViewNever
//For Day
SunDirection=-0.6 -0.6 0.8
SunLightAmbient=1.0 1.0 1.0 1.0
SunLightDiffuse=1.0 1.0 1.0 1.0
SunLightSpecular=1.0 1.0 1.0 1.0
GlobalAmbient=1.0 1.0 1.0 1.0
SkyColor=.25 .25 .25 1.0
//For twilight
//SunDirection=-0.3 -0.8 0.8
//SunLightAmbient=0.3 0.3 0.3 1.0
//SunLightDiffuse=0.3 0.3 0.3 1.0
//SunLightSpecular=0.4 0.4 0.4 1.0
//GlobalAmbient=0.25 0.25 0.25 1.0
//SkyColor=0.2 0.33 0.55 1.0
//SunLightAmbientNight=0.1 0.1 0.1 1.0
//SunLightDiffuseNight=0.5 0.5 0.5 1.0
SunLightAmbientNight=0.05 0.05 0.05 1.0
SunLightDiffuseNight=0.05 0.05 0.05 1.0
SunLightSpecularNight=0.05 0.05 0.05 1.0
GlobalAmbientNight=0.05 0.05 0.1 1.0
SkyColorNight=0.00637 0.038 0.09441 1.0
Sunlight
SunlightId=TheSunInPlanView
PlanViewOnly
SunDirection=-0.6 -0.6 0.8
SunLightAmbient=1.0 1.0 1.0 1.0
SunLightDiffuse=1.0 1.0 1.0 1.0
SunLightSpecular=1.0 1.0 1.0 1.0
GlobalAmbient=1.0 1.0 1.0 1.0
SkyColor=0.6837 0.8978 1.0 1.0
// TheSunInPlanView must be alphabetically after TheSun
// due to CarSunlightCreator::SetDaylight
```

The next section defines the scenario start positions to be used while in driving mode. This value is used on reset as well as initialization of a scenario. Replay and Auto play modes will use the first location in the file definition. Multiple start positions are used for the multiple sessionIds set by the GUI or launchparam.

```
// -----
ScenarioStartPosition=211.428 -542.24 0.53 0.0
ScenarioStartPosition=70.41 -550.1 0.53 0.0
ScenarioStartPosition=140.8 -548.6 0.53 0.0
ScenarioStartPosition=1.5 -548.6 0.53 0.0
//ScenarioStartPosition=680.0 460.0 0.0 0.0
//ScenarioStartPosition=240.0 240.0 0.0 0.0
//ScenarioStartPosition=0.0 0.0 0.0 0.0
//ScenarioStartPosition=-1995.0 -1.9 1.08 90.0
```


Various STARTUP_PARAMS can be set in this section. The PreRender startup parameter will preload the listed models to improve loading behavior.

```
// -----
STARTUP_PARAM:
DayTime
Headlight=off
PreRenderBegin
sgl_1 sgl_2
pl_f pl_r pl_l pl_fl pl_fr
ss3_f ss3_r ss3_l ss3_fl ss3_fr
ss2_f ss2_r ss2_l ss2_fl ss2_fr
tile4_W4_2LE_f tile4_W4_2LE_r tile4_W4_2LE_l tile4_W4_2LE_fl tile4_W4_2LE_fr
tile2_W4_2L_f tile2_W4_2L_r tile2_W4_2L_l tile2_W4_2L_fl tile2_W4_2L_fr
tile1_bldgs_f tile1_bldgs_r tile1_bldgs_l tile1_bldgs_fl tile1_bldgs_fr
tile1_bldsigns_f tile1_bldsigns_r tile1_bldsigns_l tile1_bldsigns_fl tile1_bldsigns_fr
tile1_signs_f tile1_signs_r tile1_signs_l tile1_signs_fl tile1_signs_fr
tile4_signs_f tile4_signs_r tile4_signs_l tile4_signs_fl tile4_signs_fr
tile1_signals_f tile1_signals_r tile1_signals_l tile1_signals_fl tile1_signals_fr
tile8_signals_x_f tile8_signals_x_r tile8_signals_x_l tile8_signals_x_fl tile8_signals_x_fr
tile4_signals_f tile4_signals_r tile4_signals_l tile4_signals_fl tile4_signals_fr
tile4_n2s_f tile4_n2s_r tile4_n2s_l tile4_n2s_fl tile4_n2s_fr
tile1_n2s_f tile1_n2s_r tile1_n2s_l tile1_n2s_fl tile1_n2s_fr
tile1_e2w_f tile1_e2w_r tile1_e2w_l tile1_e2w_fl tile1_e2w_fr
PreRenderEnd
```

The **TrafficManager** section is where the VISSIM scenario used for traffic is defined. The traffic host can be defined here, but can also be defined in the configuration file. This is the same for the tcp port to use for the traffic socket connection. The next variable will define the VISSIM .inp file to use for the VISSIM scenario. This location is defined from the VISSIM's computers relative location and must be specified as the VISSIM computer needs it. It is suggested not to change the **VisSimSubjOffsetZ=0.5** value, as VISSIM is somewhat sensitive when this value is changed but is supposed to represent the height of the ownership driven vehicle. The **VisSimResol** value should be kept at 10, as it represents the number of sim steps per second and 10 is the highest available resolution. The **VehicleTurnSwitchName** is what switch flt node to find for turn signal settings. This should not be changed while using the supplied vehicle models.

The **VehicleModel** list should list the 3D models used within VISSIM and linking those to flt file models to be used in ARCHER. The model numbers are arbitrarily set, but we use the same numbers as VISSIM for making debugging easier, but they should be unique. Each vehicle can have an offset of X, Y, Z, H, P, R and the vehicle should be positioned to be in the middle of the front bumper as the origin.

The **VisSimGlobalBound** attribute defines the total play area of the VISSIM scenario.

The **VissimToArcher** attribute will map two points from the VISSIM coordinates to the ARCHER coordinates.

```
TrafficManager:
TrafficHost=169.135.246.54
#TrafficHost=hrd01-vissim2
TrafficPort=8778
VisSimFile=t:\data\CFI\Utah_VisSim_v2.0\VISSIM_model\Network_1_dsims_test.inp
#VisSimFile=t:\data\CFI\Utah_VisSim_v2.0\VISSIM_model\Network_1_dsims.inp
#VisSimSubjOffsetZ=InitOffsetZ+VehicleDynamicsOffsetZ
#VisSimSubjOffsetZ=1.077
VisSimSubjOffsetZ=0.5
VisSimResol=10
VehicleTurnSwitchName=turnsignals
VehicleModel=car1.v3d 101 /hdsshare/data/Vehicles/BMW_330_2010_white.flt -0.024 -0.18 0.0 0 0 0
VehicleModel=car2.v3d 102 /hdsshare/data/Vehicles/Toyota_Prius_2010_gray.flt -0.024 -0.18 0.0 0 0 0
VehicleModel=car3.v3d 103 /hdsshare/data/Vehicles/Porsche_Cayenne_2011_red.flt -0.024 -0.18 0.0 0 0 0
VehicleModel=car4.v3d 104 /hdsshare/data/Vehicles/Volkswagen_Routan_2010_black.flt -0.024 -0.18 0.0 0 0 0
VehicleModel=car5.v3d 105 /hdsshare/data/Vehicles/Mini_CooperS_2006_gray.flt -0.024 -0.18 0.0 0 0 0
VehicleModel=car6.v3d 106 /hdsshare/data/Vehicles/Dodge_Challenger_2010_red.flt -0.024 -0.18 0.0 0 0 0
VehicleModel=BMW.v3d 107 /hdsshare/data/Vehicles/BMW_330_2010_red.flt -0.024 -0.18 0.0 0 0 0
```

```

VehicleModel=Car_Cadillac_CTS_2003.v3d 108 /hdsshare/data/Vehicles/Cadillac_CTS_2008_tea.flt -0.024 -0.18 0.0 0
0 0
VehicleModel=Car_Cadillac_STS_2007.v3d 109 /hdsshare/data/Vehicles/Cadillac_Escalade_2004_green.flt -0.024 -0.18
0.0 0 0 0
VehicleModel=Car_Chrysler_300C_2005.v3d 110 /hdsshare/data/Vehicles/Chrysler_300C_2010_gold.flt -0.024 -0.18 0.0
0 0 0
VehicleModel=Car_Mazda_Miata_2004.v3d 111 /hdsshare/data/Vehicles/Mazda_Miata_2008_silver.flt -0.024 -0.18 0.0 0
0 0
VehicleModel=Car_Mini_Cooper_2004.v3d 112 /hdsshare/data/Vehicles/Mini_CooperS_2006_white.flt -0.024 -0.18 0.0 0
0 0
VehicleModel=Car_Porsche_Cayman_2008.v3d 113 /hdsshare/data/Vehicles/Porsche_Cayenne_2011_biege.flt -0.024 -0.18
0.0 0 0 0
VehicleModel=Car_Saturn_LS_2003.v3d 114 /hdsshare/data/Vehicles/Saturn_LS_1998_purple.flt -0.024 -0.18 0.0 0 0 0
VehicleModel=Car_Toyota_Prius_2007.v3d 115 /hdsshare/data/Vehicles/Toyota_Prius_2010_blue.flt -0.024 -0.18 0.0 0
0 0
VehicleModel=Van_Dodge_Caravan_2005.v3d 116 /hdsshare/data/Vehicles/Volkswagen_Routan_2010_yellow.flt -0.024 -
0.18 0.0 0 0 0
VehicleModel=truck.v3d 201 /hdsshare/data/Vehicles/Box_Truck_2003_blue.flt -0.024 -0.18 0.0 0 0 0
VehicleModel=truck4.v3d 202 /hdsshare/data/Vehicles/Box_Truck_2003_white.flt -0.024 -0.18 0.0 0 0 0
VehicleModel=truck5.v3d 203 /hdsshare/data/Vehicles/Box_Truck_2003_gray.flt -0.024 -0.18 0.0 0 0 0
VehicleModel=bus.v3d 301 /hdsshare/data/Vehicles/Metro_Bus_2005_blue.flt -0.024 -0.18 0.0 0 0 0
VehicleModel=Bus_Double_red.v3d 302 /hdsshare/data/Vehicles/Metro_Bus_2005_gray.flt -0.024 -0.18 0.0 0 0 0
VehicleModel=Bus_Doubledecker.v3d 303 /hdsshare/data/Vehicles/Metro_Bus_2005_orange.flt -0.024 -0.18 0.0 0 0 0
VehicleModel=Bus_greyhound.v3d 304 /hdsshare/data/Vehicles/Metro_Bus_2005_yellow.flt -0.024 -0.18 0.0 0 0 0
VehicleModel=Bus_school_36ft.v3d 305 /hdsshare/data/Vehicles/Metro_Bus_2005_white.flt -0.024 -0.18 0.0 0 0 0
VehicleModel=Bus_city_transit.v3d 306 /hdsshare/data/Vehicles/Metro_Bus_2005_black.flt -0.024 -0.18 0.0 0 0 0
VisSimGlobalBound=-2610.0 -1830.0 3590.0 3590.0
VissimToArcher=(-4282.2,1147.7)->(-2079.16,1158.30),(-2054.6,-1263.1)->(149.53,-1255.47)
:End

```

The MapView section defines what image to use for the map-based plan view. In the case below, multiple CaseIds are being used since the roadway geometry is different and has four different representations.

```

<MapView>
  <Display orientation='180' />
  <Background worldcoord='-2612,-1788,966,1719'>
    /hdsshare/data/CFI/plan_views/scenel_1k.png
    case:1:5:9
  </Background>
  <Background worldcoord='-2681,-1926,826,1650'>
    /hdsshare/data/CFI/plan_views/scene2_1k.png
    case:2:6:10
  </Background>
  <Background worldcoord='-2680,-1651,828,1787'>
    /hdsshare/data/CFI/plan_views/scene3_1k.png
    case:3:7:11
  </Background>
  <Background worldcoord='-2750,-1789,688,1719'>
    /hdsshare/data/CFI/plan_views/scene4_1k.png
    case:4:8:12
  </Background>
  <ViewerIcon size='200,200'>
    /hdsshare/data/CFI/plan_views/saturn_2d.png
  </ViewerIcon>
</MapView>
/-- end

```

9. FLEX LANES: INTRODUCTION

Reversible lanes are an effective strategy that leads to a more balanced volume-to-capacity ratio on peak and off-peak directed traffic lanes on urban roads. For urban roads with two or more traffic lanes per direction that carry heavy commuter traffic, it was noticed that the capacity usage of the peak and off-peak directed traffic lanes varies greatly. In the direction of the peak traffic, congestions are common and the lane capacity is not enough to carry all the traffic. On the other hand, the capacity of the off-peak directed traffic lanes is underutilized, since the traffic demand is much lower. The idea behind reversible lanes was to increase the capacity of the peak directed lanes by alternating the direction of the middle lanes in the roadway. In this way, the peak traffic would get an additional lane (or lanes), while the off-peak traffic would lose it (or them), creating a better volume-to-capacity ratio in peak and off-peak lanes. This was seen as a better option for increasing the capacity where and when it was needed, than widening the roadway and adding additional lanes.

Even with the long history of reversible lanes, which have been used for more than 80 years, there are few practices that guide their application compared with numerous other techniques of traffic management. Some applications of this system in the United States were successful, while others failed due to various reasons. The major concerns of reversible lane systems are safety (changing the direction of lanes can lead to direct conflicts among vehicles if the drivers don't respond properly), and left turns along the corridor (left-turn lanes also have to change their position; a bigger problem is mid-block left turns). In order for reversible lanes to function properly, it is very important to have adequate traffic signalization, and to educate the drivers how to use the system and how to respond to lane transitions.

As a strategy to alleviate congestions along 5400 South route in the City of Taylorsville, where the peak direction traffic congestions are common, the Utah Department of Transportation (UDOT) proposes to install a reversible lane system, called Flex lanes. This report describes the Flex lanes system, explains solutions for the most common problems of reversible lanes, and introduces the main means of driver education, evaluation, and feedback through a computer-based simulator.

10. LITERATURE REVIEW

Reversible lane systems (RLS) have been in use for more than 80 years (1, 2, 3). Their application on roadways have varied greatly; ranging from high-density freeways to local neighborhood streets. Even with the long history of RLS, there are few practices that guide their application compared with the numerous other techniques of traffic management (1).

Several studies were conducted prior to 2003 on the benefit/safety aspects of existing RLS. These published studies and surveys analyzed existing reversible lane systems with the objective of understanding why some systems work (4, 5) while other ones fail (6, 7, 8).

In 2003, a review of every reversible lane system in the United States was evaluated and published as the National Cooperative Highway Research Program (NCHRP) Synthesis 340 (3). The synthesis was created to address the need to enhance the understanding within the transportation community relative to convertible and reversible lane use. Production of the synthesis was needed by the transportation community due to the limited amount of published information available on issues related to reversible lane planning, design, operation, control, management, and enforcement. Forty-nine states and local transportation, law enforcement, and emergency management agencies replied with survey information, demonstrating that 23 of these agencies were using one or more forms of reversible lane operations. A review of published literature and other “difficult-to-access” reports and studies was carried out, as well as a survey of current and recent practices. Lastly, field visits and dialogues with practitioners gave light to additional information that was used to establish seven specific examples of the variety of design and control characteristics that can make reversible/lane operations successful. This synthesis found that many of the actual cost and benefits of reversible lane systems remain largely unexplored, which if known, may have been able to contribute to a decrease in traffic accidents, increased efficiency, and better use of resources (1).

A study published by Bretherton and Elhaj reviewed the U.S. 78 RLS in Georgia and found that part of the public’s negative perception of RLS was due to an increase in the number of injuries and fatalities, and injury and fatality rates along the corridor, which were mostly caused by driver confusion. This negative perception was received even though the installation of the RLS showed an increase in level-of-service (LOS) and operation speeds on U.S 78 (6). A survey conducted by the Arizona State University School of Planning found an overwhelming negative sentiment across all stakeholders of the installed RLS along 7th Avenue and 7th Street. It stated that users felt they are unsafe, had reduced accessibility due to the limited left-turn opportunities, and had a reduced effectiveness due to driver confusion (7).

Conversely, the NCHRP Synthesis 340 found that nearly all the agencies surveyed did not report any significant safety problems or driver confusion (3). It also found that even though there have not been many analyses of the safety effect of reversible operations, a large amount of empirical evidence gained from past experience indicates that drivers readily adapt to them. An RLS installed in downtown Calgary, Canada, received great praise from its residents, with the local agency citing its success due to clear, realistic presentations of the facility, accurate traffic analysis and modeling, and public input, which was addressed and incorporated into the design when possible (5). Even with the synthesis stating that agencies haven’t had any significant problems with safety or confusion; it reports that these same agencies still remain hesitant to implement reversible operations if they can be avoided. This tepidness is due in part to a generally held belief that reversible operations may be confusing to drivers not familiar with this type of operation, and that the agencies will require additional staffing to manage and enforce the operation (3).

There is some direction given by the American Association of State Highway and Transportation Officials (AASHTO), the Manual on Uniform Traffic Control Devices (MUTCD), the Institute of Transportation Engineers (ITE), and the Federal Highway Association (FHWA) on proposed criteria and general practices of RLS, although the details are not as specific or consistent as for other forms of traffic management and control (1). While not having a wide latitude in RLS design can be seen as negative, some see this as a good thing, as many publications have shown that successful, publicly supported systems are generally designed specifically for the area in which it is used, such as the RLS in Calgary. That “ground up” design promotes thoughtful and rational decision making, leading to results that fit that particular region.

Wolshon and Lambert state that during their evaluation of RLS (1, 2, 3,9), they found no significant number of published studies to evaluate the safety effect of reversible operations, and it was clear that the subject of reversible lane systems represents a gap in knowledge, particularly in areas of assessment and evaluation. The review showed that the performance of the vast number of these segments has never been quantitatively evaluated, and although some performance reviews have been undertaken, their results are not widely disseminated in the general literature. This lack of study indicates that many aspects of their costs and benefits remain largely unknown. Such information would be very valuable to use in developing criteria and planning/design/management guidelines that can be used to determine the conditions that might warrant their use as well as their potential impact on safety and mobility versus more conventional techniques (2).

One of the largest challenges still facing RLS is the gap in knowledge of assessment and evaluation of safety and operational effects of left-hand turns. Mitigation procedures, such as additional signage and better traffic control devices, have been shown to successfully reduce accidents on RLS; however, Wolshon and Lambert found that there is not a significant number of published studies to evaluate the quantitative safety effects of these procedures (3). This publication attempts to analyze the effects of left-hand turns on RLS, specifically by quantifying the effects of traffic flows and accident rates (safety) in two ways. One, by varying the signalized spacing (intersection to intersection) of controlled left-hand turns; and two, by allowing/disallowing mid-block (uncontrolled) left had turns, on RLS corridors.

By discovering what conditions left turns have on traffic flows and accident rates, engineers may be able to use these data to design reversible lanes that more fully optimize the cost of the system, and increase the level of safety along the RLS corridor. This optimization of capacity and safety may help dampen drivers’ uneasiness with RLS, and improve the frequency of their use among transportation agencies.

11. THE NEED FOR FLEX LANES ON 5400 SOUTH

The western side of the Salt Lake Valley has experienced a tremendous amount of residential and commercial development in the past decade. This led to an immense increase in traffic, especially during the peak commuting hours, causing traffic congestion on many West-East routes. Delays at intersections and lack of road capacity contribute to this congestion. UDOT has developed a coordinated strategy involving several innovative projects to alleviate traffic congestion on West-East routes in and around the City of Taylorsville, especially during peak hours. These strategies are aimed at reducing congestion, and at the same time improve travel time, increase safety, increase roadway capacity and lifespan, and use the existing infrastructure in a more efficient manner.

5400 South is one of the major West-East arterials in the area, carrying most of the commuter traffic. It also represents a connection with major North-South routes, such as Bangerter Highway, I-215, Redwood Road, and I-15. During the morning and afternoon peak hours, this route experiences heavy traffic congestions in the peak hour direction: eastbound in the morning and westbound in the afternoon. This route already has three traffic lanes per direction (plus the median lane), so widening the roadway wouldn't be an effective option. Evaluating different options for increasing capacity, UDOT has found that the reversible lane system (Flex lanes) would work best for the given corridor. Implementing this system, the existing roadway configuration will be utilized in the most effective manner. During the peak hours, the capacity of the off-peak direction lanes is underutilized, while the peak direction lanes are congested. Alternating the number of lanes in the AM, PM, and off-peak periods would bring a better balance of the volume-to-capacity ratio in the peak and off-peak direction. Flex lanes are planned to be built along the most congested segment of 5400 South, between Bangerter Highway and Redwood Road. In order to overcome all the concerns of this concept, to introduce the system to the users and get support, UDOT actively involves the public and invites people to participate in the process.

5. Off-Peak - this lane is used as a WESTBOUND THROUGH LANE. A vehicle in this lane may continue straight through, or merge left or right into the adjacent lanes.
6. Off-Peak to Evening Peak - this lane is used as a WESTBOUND THROUGH LANE. A vehicle in this lane may continue straight through, or merge left or right into the adjacent lanes.
7. Evening Peak - this lane is used as a WESTBOUND THROUGH LANE. A vehicle in this lane may continue straight through, or merge left or right into the adjacent lanes.
8. Evening Peak to Off-Peak - this lane is used as a WESTBOUND THROUGH LANE. A vehicle in this lane may continue straight through, or merge left or right into the adjacent lanes.
9. Off-Peak - this lane is used as a WESTBOUND THROUGH LANE. A vehicle in this lane may continue straight through, or merge left or right into the adjacent lanes.

LANE 4

1. Off-Peak - this lane becomes a LEFT TURN LANE for both directions of traffic. A vehicle in this lane may make a left turn or merge right into the adjacent though lane.
2. Off-Peak to Morning Peak - this lane is cleared of all traffic. A vehicle in this lane must turn left or merge right into the adjacent thru lane.
3. Morning Peak – this lane is used as an EASTBOUND THROUGH LANE. A vehicle in this lane may continue straight through, or merge left or right into the adjacent lanes.
4. Morning Peak to Off-Peak – this lane is cleared of all traffic. A vehicle in this lane must merge left into the clearing left turn lane, or merge right into the adjacent through lane.
5. Off-Peak - this lane becomes a LEFT TURN LANE for both directions of traffic. A vehicle in this lane may make a left turn or merge right into the adjacent though lane.
6. Off-Peak to Evening Peak - this lane is cleared of all traffic.
7. Evening Peak - this lane is used as a WESTBOUND THROUGH LANE. A vehicle in this lane may continue straight through, or merge left or right into the adjacent lanes.
8. Evening Peak to Off-Peak - this lane is cleared of all traffic. A vehicle in this lane must turn left or merge right into the adjacent thru lane.
9. Off-Peak - this lane becomes a LEFT TURN LANE for both directions of traffic. A vehicle in this lane may make a left turn or merge right into the adjacent though lane.

LANE 5

1. Off-Peak - this lane is used as an EASTBOUND THROUGH LANE. A vehicle in this lane may continue straight through, or merge left or right into the adjacent lanes.
2. Off-Peak to Morning Peak - this lane is used as an EASTBOUND THROUGH LANE. A vehicle in this lane may continue straight through, or merge left or right into the adjacent lanes.
3. Morning Peak - this lane is used as an EASTBOUND THROUGH LANE. A vehicle in this lane may continue straight through, or merge left or right into the adjacent lanes.
4. Morning Peak to Off-Peak - this lane is used as an EASTBOUND THROUGH LANE. A vehicle in this lane may continue straight through, or merge left or right into the adjacent lanes.
5. Off-Peak - this lane is used as an EASTBOUND THROUGH LANE. A vehicle in this lane may continue straight through, or merge left or right into the adjacent lanes.
6. Off-Peak to Evening Peak - this lane is CLEARING of all traffic. A vehicle in this lane must exit this lane by either merging right into the adjacent through lane, or merge left into the clearing left turn lane.
7. Evening Peak - this lane becomes a LEFT TURN LANE for both directions of traffic. A vehicle in this lane may make a left turn or merge right into the adjacent though lane.
8. Evening Peak to Off-Peak – this lane is CLEARING of all traffic. A vehicle in this lane must exit this lane by either merging right into the adjacent through lane, or turn left.

- 9. Off-Peak - this lane is used as an EASTBOUND THROUGH LANE. A vehicle in this lane may continue straight through or merge left or right into the adjacent lanes.

LANE 6

- 1. Not Reversible

LANE 7

- 1. Not Reversible

Signal gantries are the most delicate part of the Flex lanes, because they communicate directly to the drivers informing them about lane assignments, so that drivers will know what action to take and to prepare for the oncoming transition. The look of the signal gantries from the drivers' perspective for different periods is given in Figure 12.2.



Figure 12.2 The Look of Signal Gantries from the Drivers' Perspective

13. LEFT TURNS WITHIN THE FLEX LANES SYSTEM

The best places to implement the reversible lanes system are bridges, because there are no turning movements. Implementation on bridges is more straightforward than other corridors of traffic that have turning movements. When reversible lanes are used in systems with turning movements, their complexity increases, and with that increased complexity comes more driver confusion. This can lead to more frequent vehicle incidents, negative public opinions of reversible lanes, and an overall increase in frustrated drivers.

A main cause of concern with both the jurisdictional agency and drivers of the reversible lane corridor is how left-turn movements will be handled, both turning onto and off of the reversible lanes. The simplest way to handle left-turn movements is to eliminate them altogether. This will lead to higher capacity with the elimination of the left-turn lane and left-turn movements at intersections. The no-left option favors commuters who are passing through the area, but not local residents or businesses who see the no-left option as an impediment to their accessibility.

The problem of the left-turn lanes within a reversible lane system has been a difficult one to solve for many cities that want to implement this system. The city of Omaha decided to abandon the use of reversible lanes because a study found that the number of crashes increased after the reverse lanes were put into use (3). This led many to believe that reversible lanes were unsafe and not worth the price of increased capacity. This notion of RLSs being unsafe is a common misconception among the population. A study conducted after the reversible lanes were removed found that all of the “increase” in accidents was related to drivers illegally attempting to make mid-block left turns. The accidents either involved a motorist attempting to cross the opposing lanes of traffic and being blindsided, or a motorist stopping in the turn lane halfway through the block and being rear ended by a driver following behind. The problem was not that reversible lanes were “unsafe,” but that either the population was not educated well enough on the rules of the system or the no-left-turn law was not being enforced.

With the Flex lanes system implementation, UDOT is trying to solve the left-turn problem in the best possible way. The public is actively involved in this procedure. Within this system, left turns will be allowed at all signalized intersections. Special four-light traffic signal displays that can display both circles and arrows will be used for lanes that change from through to left and vice versa. Conditions for left turns at signalized intersections can be found in Appendix C. A bigger problem is the left turns at non-signalized intersections and mid-block left turns. So far, a compromise between UDOT and the public has been made on the following turning options:

Left turns will be allowed onto 5400 South at all hours from Harvey Heights Road and Jordan Canal Road. The initial plan was to restrict left turns during rush hours. However, these two roads have no other viable options for merging onto 5400 South.

Left turns will be allowed at all non-signalized intersections if vehicles will cross only two lanes of traffic. This will allow motorists on the north side of 5400 South to make a left turn and enter the eastbound Flex lanes during the morning rush hour. Likewise, motorists on the south side of 5400 South will be able to cross 5400 South to enter the westbound Flex lanes during the evening rush hour. The initial plan called for disallowing all left turns from all intersections without traffic signals.

The outside (or curb) lanes on 5400 South will be approximately 13 feet wide, instead of the originally planned 11 feet, to allow more room for buses and right turns on the road.

As for law enforcement and education, UDOT proposes to install signs stating that left turns are not allowed as well as impose demanding fines on violators of the “no-left turn” law within the RLS. To solve the education problem, UDOT will introduce a simulation video to educate drivers on the laws and workings of the Flex lanes.

14. TRANSITION SCENARIOS

The transition periods (off-peak and vice versa) are the most critical periods for Flex lanes operations. During these periods, lanes are changing their direction and assignment, creating possibilities for conflicts among vehicles. This is where the signal gantries and four-light traffic signal displays play a major role in informing the drivers about the oncoming changes. Eight possible options for vehicles traveling in the middle lanes are identified, depending on the lane in which the vehicles travel and their intention at the next intersection (going through or turning left). Refer to Figure 14.1 and Table 14.1 for a detailed description. This example shows vehicles travelling eastbound, but the same goes for the opposite direction.

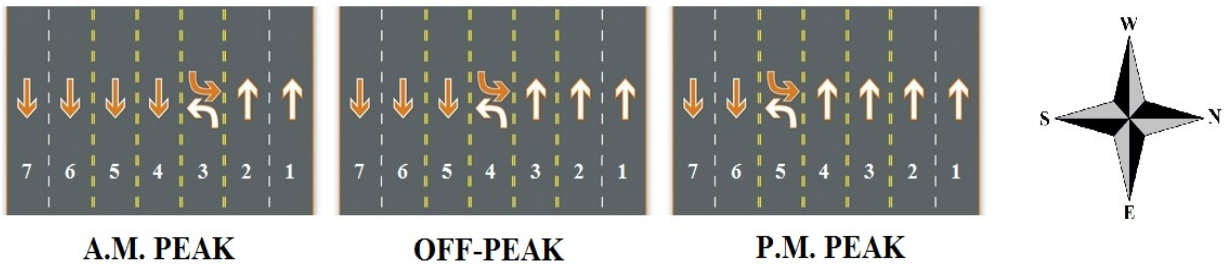


Figure 14.1 Lane Assignment and Transition

Table 14.1 Transition Scenarios

Direction	Transition Period	Lane/Origin	Intent	Lane/Destination	Transition	Vehicle Test
EB	1) Off peak to AM peak	5	left	3	Gaining a lane	Car needs to move 2 lanes to turn left at next intersection
EB	2a) AM peak to Off peak	4	thru	5	Losing a lane	Car at inside thru lane and wants to go THRU next intersection but must move out of dropping lane.
EB	2b) AM peak to Off peak	4	left	4	Losing a lane	Car at inside thru lane and wants to go LEFT next intersection but must move out of dropping lane.
EB	2c) AM peak to Off peak	5	left	4	Losing a lane	Car at inside thru lane and wants to go LEFT next intersection but must move out of dropping lane.
EB	3a) Off peak to PM peak	5	thru	6	Losing a lane	Car at inside thru lane and wants to go THRU next intersection but must move out of dropping lane.
EB	3b) Off peak to PM peak	5	left	5	Losing a lane	Car at inside thru lane and wants to go LEFT next intersection but must move out of dropping lane.
EB	3c) Off peak to PM peak	6	left	5	Losing a lane	Car at inside thru lane and wants to go LEFT next intersection but must move out of dropping lane.
EB	4) PM peak to Off peak	6	left	4	Gaining a lane	Car needs to move 2 lanes to turn left at next intersection

Table 14.2 shows lane transitions and some challenges for the drivers traveling in the middle lanes, depending on their intended action at the next intersection.

Table 14.2 Lane Condition, Intended Driver’s Action and Expected Challenges

Lane Condition	Intended Action	Expected Challenges
Lanes Transitioning with Yellow X	Turn Left at Intersection	Drivers feel that they need to leave the lanes with yellow X, missing their intended turn.
Lanes Transitioning with Yellow X	Go Thru Intersection	Drivers don’t merge to the right. They will be “trapped” in the turn lane.
Green Thru Arrow	Turn Left at Intersection	Drivers may turn into the Turn lane too soon, either hitting or blocking a vehicle going the opposing direction turning into a side street.
Green Thru Arrow	Go Thru Intersection	Drivers may feel that they want to be in the right most lane possible instead of using the Flex lane. This may decrease the utilization of the lane capacity.
Yellow X turns to Red X	Turn Left at Intersection	Drivers need to move out of the Flex lanes at this point. Some drivers may try to reach the intersection to make a left turn and become stuck in the wrong lane.
Yellow X turns to Red X	Go Thru Intersection	Some drivers may believe that the lane that they are in will switch back to a green arrow (the lanes in their direction is expanding) or others may choose to take advantage of the empty lane to move ahead of traffic.

15. FLEX LANES DRIVING SIMULATOR SCENARIO DEVELOPMENT

Utah Traffic Lab, in collaboration with AAI Corporation, has developed a driving simulator scenario for the 5400 S Flex Lanes project. The scenario consists of a VISSIM microsimulation model, developed by Utah Traffic Lab, and a 3D rendering model developed by AAI Corporation. These two models are integrated into a fully functional driving simulation scenario with all elements of the real-world Flex Lanes corridor.

15.1 VISSIM Model Development

The initial VISSIM model was developed in 2010 for then existing PM peak traffic conditions (4:00 – 6:00 pm) and it encompassed the corridor between Bangerter Highway and Redwood Road. Traffic counts and signal timing data were obtained from AECOM, along with basic geometric features and signalization through aerial and street view maps. This initial VISSIM model was calibrated based on turning movement counts at all intersections on this corridor.

The initial model is redesigned for the new Flex Lanes configuration. The Flex Lanes VISSIM model is developed based on UDOT’s plans and documents for the 5400 S corridor. This model is developed in such a way to encompass all main and transition periods within one hour of simulation, to make it more appropriate for use with the driving simulator.

The Flex Lanes model consists of five signalized intersections (3600 W, 3200 W, 2700 W, 2200 W and 1900 W), and eight 3-leg and 4-leg stop controlled intersections. There are seven traffic lanes along the corridor, with the lane assignment as described in the previous chapter. There are a total of eighteen overhead signal gantries (lane indicators) on the corridor. Each gantry is driven by a separate fixed-time traffic controller, with the timing set to correspond to the main and transition periods. The main periods last for 14 minutes (840 seconds), with one minute (60 seconds) transition periods. The periods coded into the one-hour VISSIM model are given in Table 15.1 in seconds.

Table 15.1 VISSIM Main and Transition Periods

Period/Transition	From (s)	To (s)
AM	0	840
AM – Off (1)	840	900
Off (1)	900	1740
Off (1) – PM	1740	1800
PM	1800	2640
PM – Off (2)	2640	2700
Off (2)	2700	3540
Off (2) – AM	3540	3600

Each fixed-time traffic control program that operates overhead lane indicators consists of 24 signal groups, 12 for each direction. These signal groups are programmed to operate in green-red mode to indicate which of the symbols are to be displayed. The corresponding signal heads do not exist in the VISSIM model, but the signal group outputs are used for integration with the driving simulator. The driving simulator needs specific and unique VISSIM outputs to indicate different combinations of symbols on overhead gantries (this is also the case for intersection signal heads). The meaning of each signal group green time is given in Figure 15.1, separately for eastbound and westbound directions.

EB Lane #	1	2	3	4	5	6	7
AM							
Signal Group	1	1	2	3	4	5	5
Transition AM - OFF							
Signal Group	1	1	6	7	4	5	5
OFF							
Signal Group	1	1	8	9	4	5	5
Transition OFF - PM							
Signal Group	1	1	8	7	10	5	5
PM							
Signal Group	1	1	8	11	12	5	5
Transition PM - OFF							
Signal Group	1	1	8	7	10	5	5
OFF							
Signal Group	1	1	8	9	4	5	5
Transition OFF - AM							
Signal Group	1	1	6	7	4	5	5



a)

WB Lane #	7	6	5	4	3	2	1
AM							
Signal Group	13	13	14	15	16	17	17
Transition AM - OFF							
Signal Group	13	13	14	18	19	17	17
OFF							
Signal Group	13	13	14	20	21	17	17
Transition OFF - PM							
Signal Group	13	13	22	18	21	17	17
PM							
Signal Group	13	13	23	24	21	17	17
Transition PM - OFF							
Signal Group	13	13	22	18	21	17	17
OFF							
Signal Group	13	13	14	20	21	17	17
Transition OFF - AM							
Signal Group	13	13	14	18	19	17	17



b)

Figure 15.1 Overhead Signal Gantry Symbols and Corresponding Signal Groups:
a) Eastbound; b) Westbound

The signal control programs for overhead gantries are dynamic, meaning that they can easily be upgraded for any duration of periods or transitions. Figure 15.2 shows the signal control program coded into VISSIM. It should be noted that the VISSIM fixed-time control program does not support more than two green/red time starts/ends, so signal groups 7 and 18 could not be coded as given in Figure 15.1. However, since some of the periods are redundant (same signal groups are active), a unique combination of signal phases could be achieved for integration with the driving simulator at any given time.

Table 15.2 VISSIM Links for Reversible Lanes

Period	EB links	WB links	TWLT links
AM	Two	None	One
Off	One	One	One
PM	None	Two	One

EB: Eastbound

WB: Westbound

TWLT: Two-way Left Turn Lane

A special feature of the 5400 S Flex Lanes system is the use of multi-purpose signal heads at signalized intersections. The signal heads are designed in accordance with the travel lane designations. The intersection of 1900 W and 5400 S does not belong to the Flex Lanes segment, so this is a regular four-leg signalized intersection with split-phasing traffic control. Along the main corridor, in either direction, there are three different allocations of through lanes and three allocations of left-turn lanes at signalized intersections. According to the design plans, there are five signal heads at each main approach (except 3600 W eastbound), with two standard three-ball signal heads for through movements, one three-light dual-purpose signal head (for protected left and through movements), one dual-purpose four-light signal head for protected left and through movements, and one four-light signal head for protected left turns. The four-light signal heads are not active during some periods, depending on the lane assignment. In addition to the signal heads, there are also electronic signs that show lane purpose during different periods. Signal heads and signs are driven by specially designed traffic signal plans, which are in this case actuated to allow for a greater flexibility. VISSIM's Ring Barrier Controllers (RBC) are used to program intersections signals. It should be noted that the signal timings operate in a non-coordinated actuated mode, and the actual field signal timings were not available during the creation of the model. However, they can easily be upgraded when the actual signal timings become available. An example of the signal head/sign configuration is given in Figure 15.3. The complete configurations are provided in Appendix A. The figures also show the allocation and status of signal groups used in VISSIM signal control programs, and the same signal groups are used to connect the VISSIM model with the driving simulator.

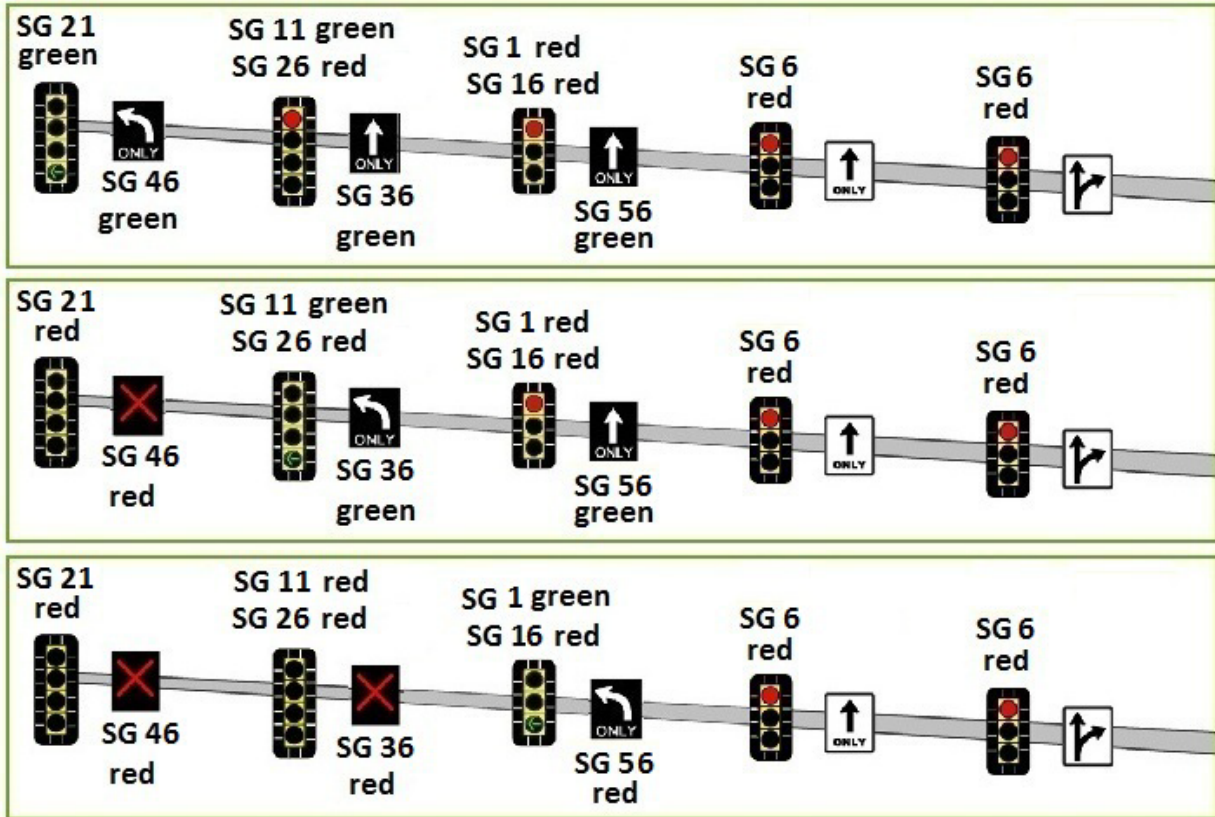


Figure 15.3 Signal Head/Sign and Signal Group Configuration: An Example

The configuration of signal states in the driving simulator corresponds to the VISSIM signal control outputs. VISSIM provides a unique combination of signal groups for each time period, which enables the exact mapping of signal states between VISSIM and the driving simulator. The intersection signal programs are flexible, and they do not require any upgrades even if the simulation periods are changed. The only thing that should be checked in this case is the state of different signal groups during transition periods, and any new changes should be coded in the driving simulator (ARCHER’s “scene file”) to incorporate any unexpected signal group combinations.

15.2 Flex Lanes Driving Simulation

The Utah Traffic Lab has a fully functional driving simulation model for the 5400 S Flex Lanes corridor. The simulation duration is one hour, with all the periods and transitions. The VISSIM model that drives the simulation is not calibrated nor validated for use with the driving simulator, but it was created in such a way to represent different traffic conditions. The model is ready for testing drivers’ compliance and understanding of the system. Figure 15.4 shows the current look of the Flex Lanes driving simulation.





Figure 15.4 Flex Lanes Driving Simulation

16. CONCLUSIONS

A reversible lanes system is an effective strategy of increasing capacity where and when it is needed, without widening the roadway and adding more lanes. It also brings more balanced capacity utilization, where volume-to-capacity ratios on peak and off-peak directed traffic lanes are balanced.

UDOT plans to activate a reversible lanes system on critical segments of the 5400 South route in the City of Taylorsville. This system should help alleviate peak direction congestions, which are very common along this route. Along with public involvement, UDOT works to solve crucial problems of the new system and provide an adequate education to the system users.

This report describes the Flex lanes system and explains the solutions to the common problems associated with reversible lanes. It also presents a development of a Flex lanes driving simulation. The simulation can be used to record drivers' responses and compliance with the posted signalization. The next step is to develop a testing module and perform testing on selected subjects.

17. REFERENCES

1. Wolshon, B. and L. Lambert, "Planning and Operational Practices for Reversible Roadways," *Institute of Transportation Engineers ITE Journal*, Aug. 2006a.
2. Wolshon, B. and L. Lambert, "Reversible Lane Systems: Synthesis of Practice," *Journal of Transportation Engineering*, Vol. 132, No. 12, Dec. 2006b, pp. 933–944.
3. Wolshon, B. and L. Lambert, NCHRP Synthesis 340: Convertible Lanes and Roadways. Transportation Research Board, National Research Council, Washington, D.C., 2004, 92 pp.
4. Hemphill, J. and V.H. Surti. A Feasibility Study of a Reversible-Lane Facility for a Denver Street Corridor. In *Transportation Research Record: Journal of the Transportation Research Board*, No. 514, Transportation Research Board of the National Academies, Washington, D.C., 1974, pp. 29-32.
5. Logan, M., T. McLeod, and C. Jordan. 5th Avenue Connector Contra-flow Facility: From Concept to Completion. 2006 Annual Conference of the Transportation Association of Canada, Charlottetown, Prince Edward Island
6. Bretherton Jr., W. M. and M. Elhaj. Is a Reversible Lane System Safe? 1996 Compendium of Technical Papers, 66th Annual Meeting of the Institute of Transportation Engineers, Minneapolis, Minn., 1996. pp. 277-281.
7. Golub, A. Quality-of-Life Study of the 7th Avenue and 7th Street Reverse Lanes. School of Planning, Arizona State University, Phoenix Urban Research Laboratory, Arizona State University. May, 2008.
8. TransCore. Grant Road Reversible Lane Traffic Flow and Crash Analysis with an Update of the Broadway Reversible Lane Study. Prepared for City of Tucson, September 2005.
9. Wolshon, B., and L. Lambert. Comparative Review of Reversible Roadway Termini Design. 3rd International Symposium on Highway Geometric Design, 2005.

APPENDIX A: FLEX LANES SIGNAL HEADS AND STATES FOR SIGNALIZED INTERSECTIONS



1900 W Signal States



2200 W Signal States: AM Peak



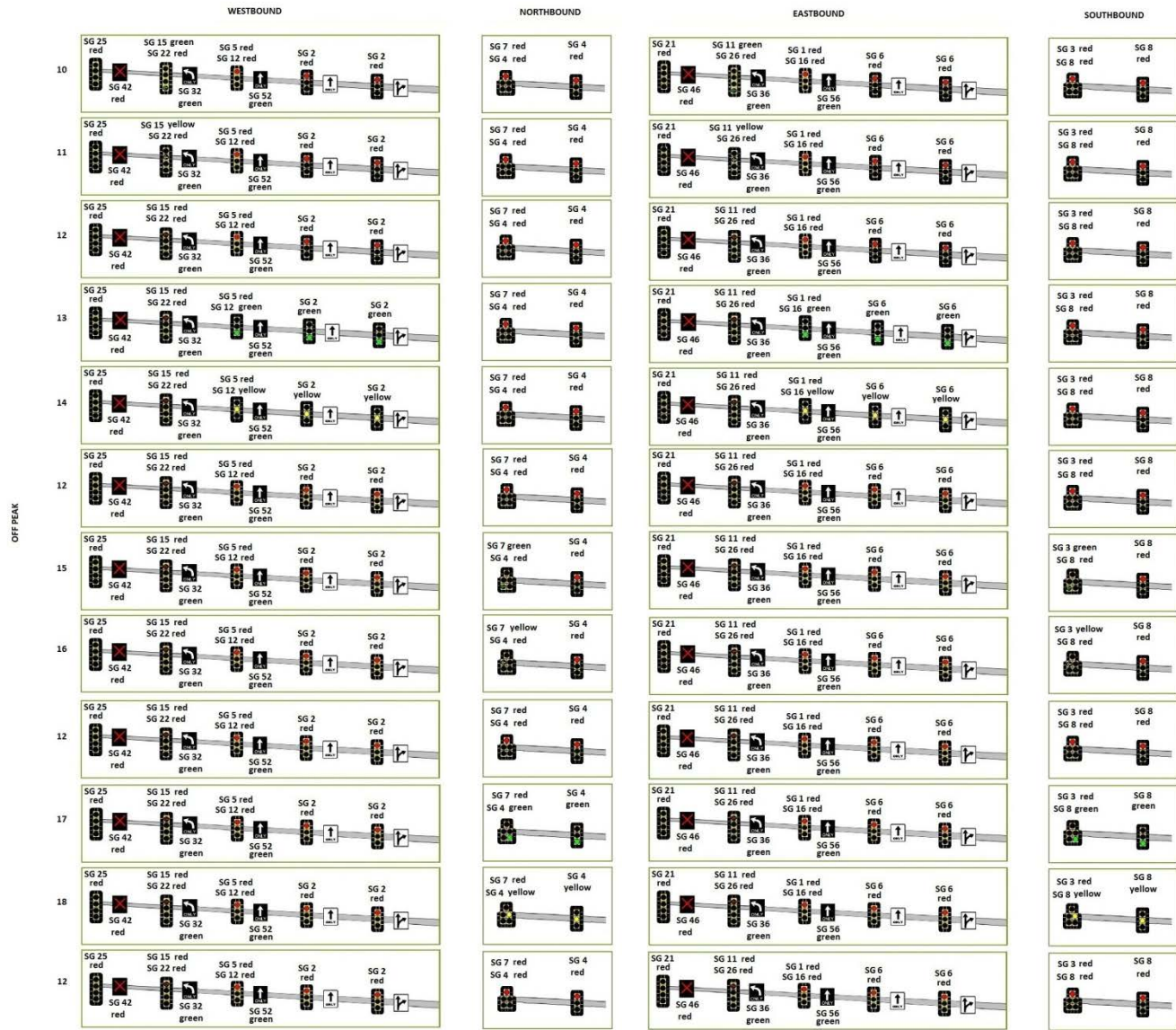
2200 W Signal States: OFF Peak



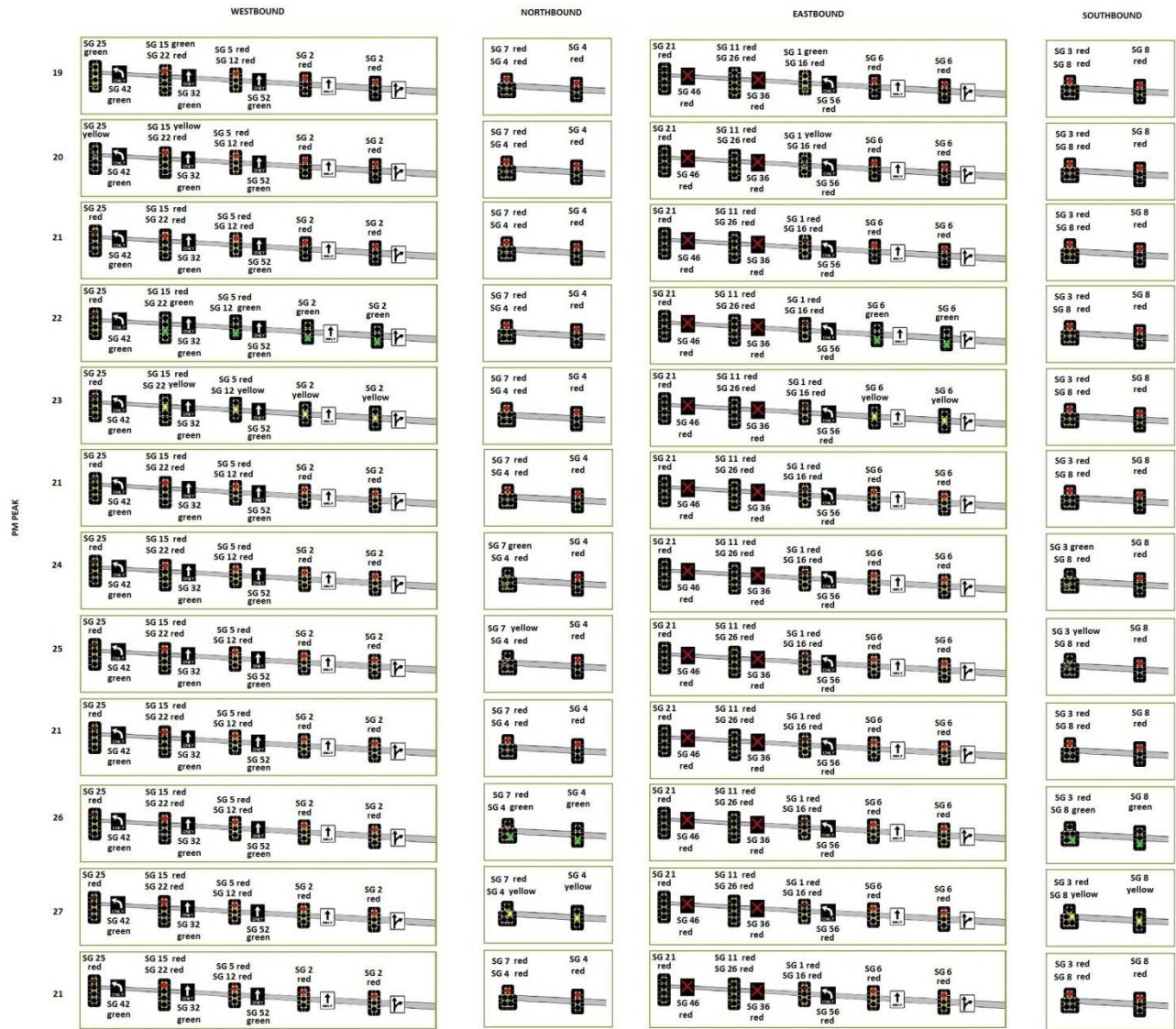
2200 W Signal States: PM Peak



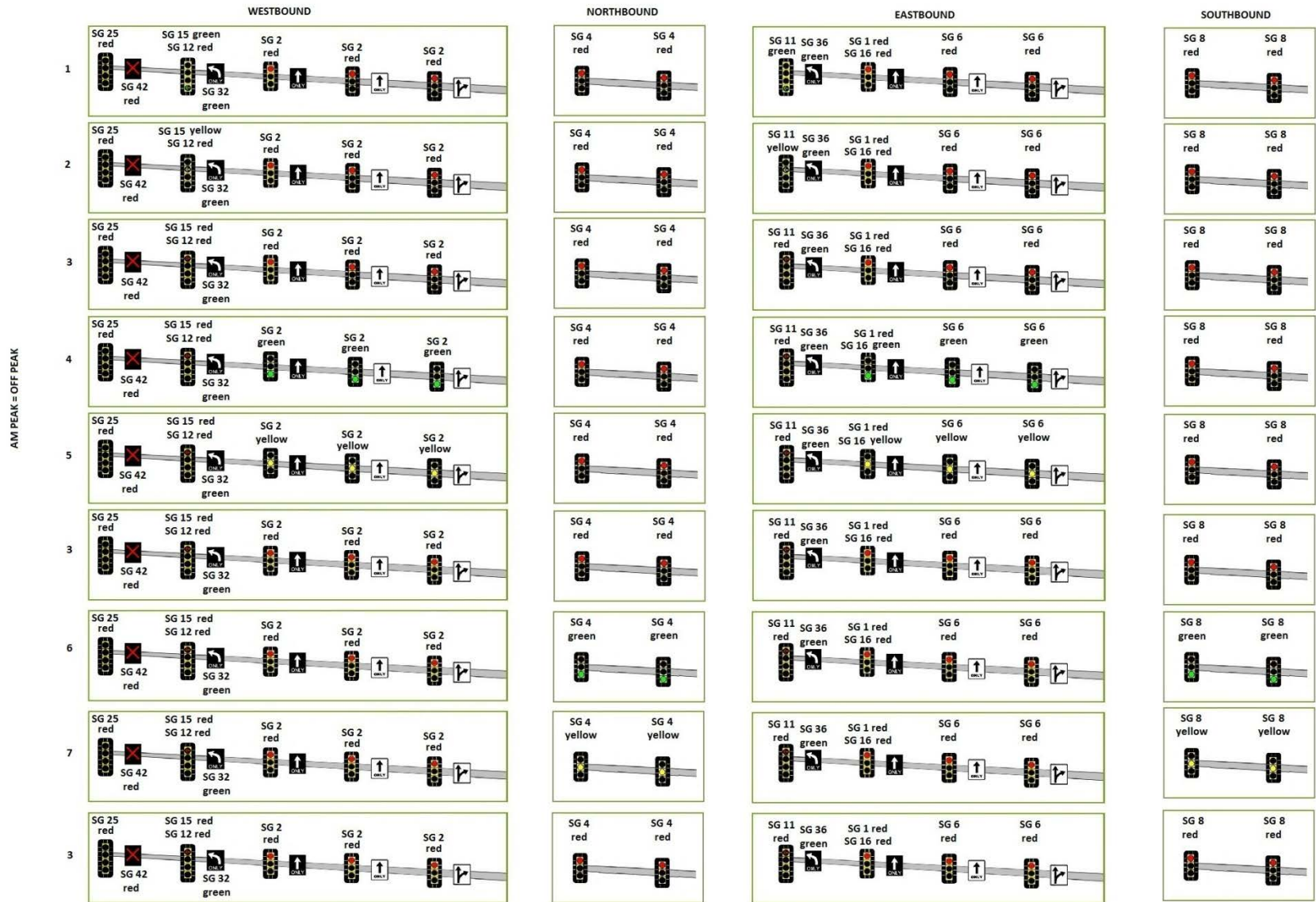
2700 W and 3200 W Signal States: AM Peak



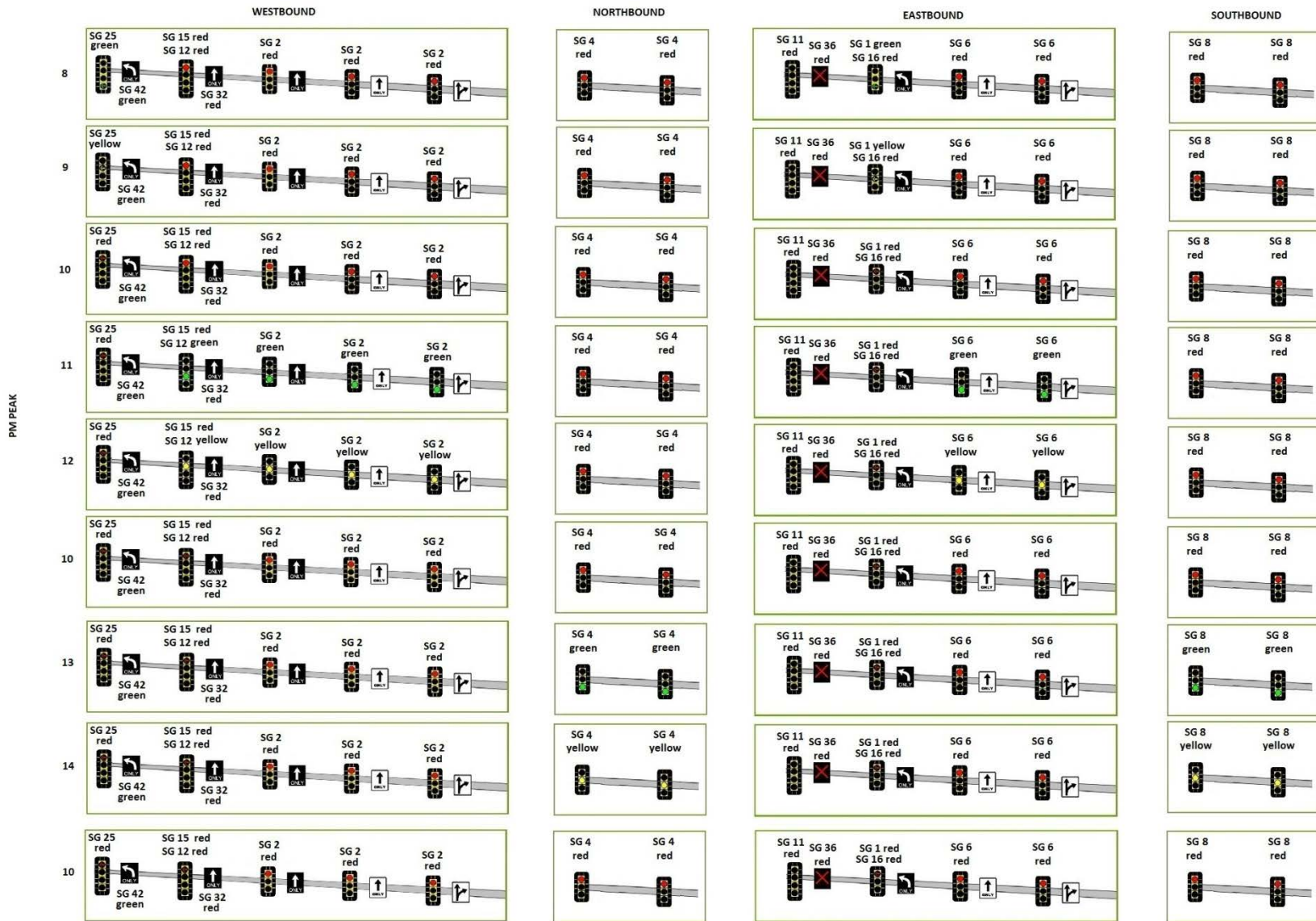
2700 W and 3200 W Signal States: OFF Peak



2700 W and 3200 W Signal States: PM Peak



3600 W Signal States: AM and OFF Peak



3600 W Signal States: PM Peak

APPENDIX B: DRIVING SIMULATOR USERS GUIDE

Driving Simulator User Guide



Ivana Tasic

INTRODUCTION

Transportation research that cannot be conducted in the real world due to safety and costs uses Driving Simulators. A simple version of a Driving Simulator can be described as a car model with driver's seat, steering wheel, driving commands and simulated dynamic traffic and roadway environment around.

The Utah Traffic Lab Driving Simulator is designed to provide highly accurate data on driver's behavior in traffic. The simulator gives the driver visual, auditory, and motion experience that closely matches the real world. It includes day and night driving scenarios, vegetation, road signs, pavement markings, and traffic control devices. It collects data on vehicle speed and position, orientation, lane changing, vehicle controls and traffic signal states.

This driving simulator is unique because of the way its scenarios are built. Traffic conditions in each scenario are created in VISSIM micro-simulation software. VISSIM defines the number of vehicles in traffic, vehicle speeds, vehicle routes and traffic signal states. Traffic conditions in VISSIM are based on real-time data. So the simulator driver is driving in close-to-realistic traffic environment. Software ARCHER incorporates the driving simulator in created traffic conditions as one of the vehicles and upgrades created traffic conditions into 3D scenarios with roadway environment. ARCHER creates scenes that driver sees at a real-time rate, and the environment on the screens around changes while driver is driving. This is how driver gets the impression of movement in close-to-real traffic environment.

DRIVING SIMULATOR COMPONENTS

Driving Simulator has Hardware and Software Components. They are listed in the following sections in the manner that matches hardware component with adequate software component and illustration. Figure 1 represents current allocation of Driving Simulator components in the Utah Traffic Lab. Use Figure 1 as a "map" for introduction to each component and Figure 2 to learn about the functions of each component.

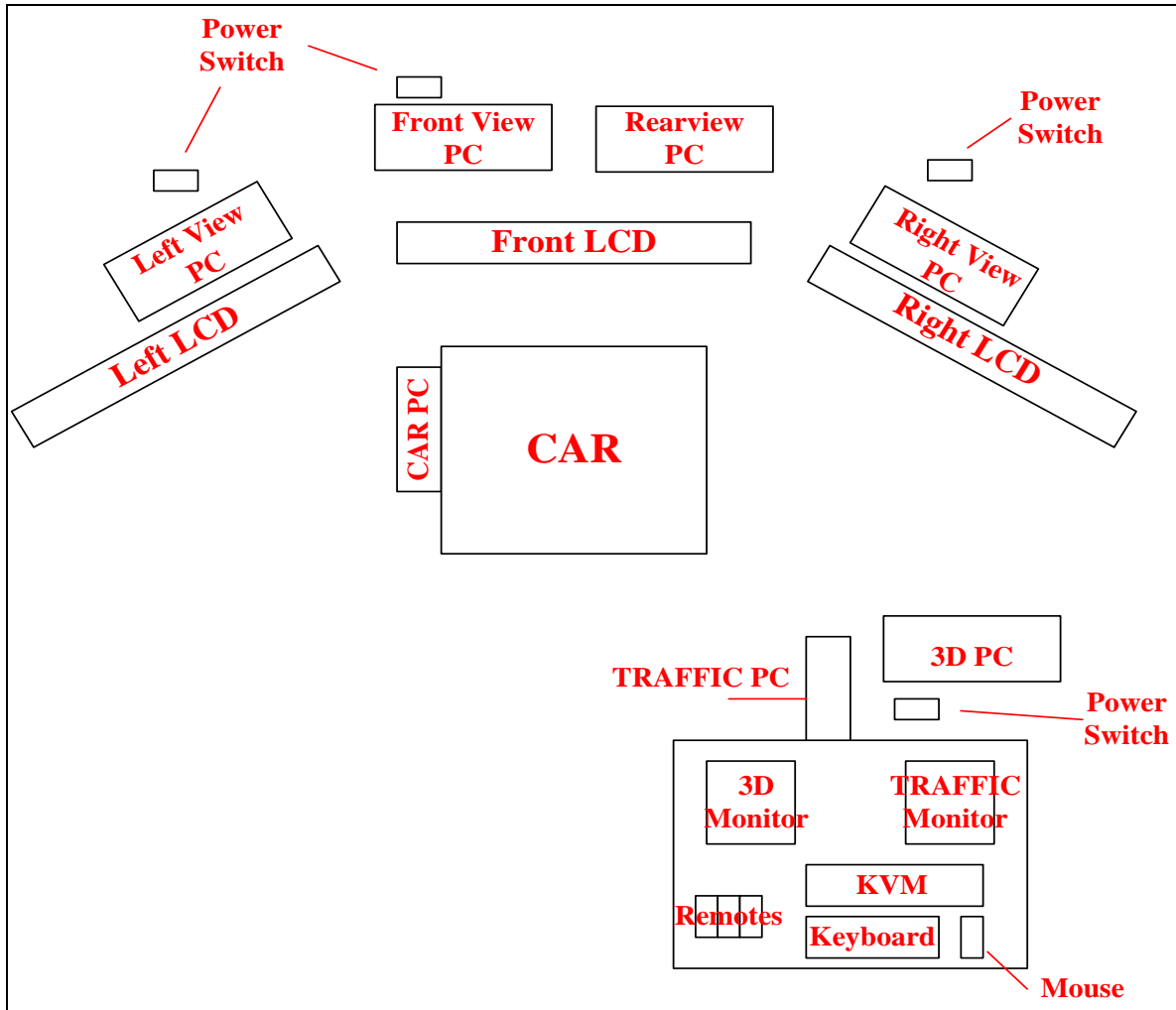



Figure 1: Driving Simulator Hardware Components Allocation

Hardware Component	Software Function	Illustration
<p>CAR</p> <p>The driver's side of the vehicle with driver's seat, steering wheel and driving controls, gas and brake, and gear shifter</p>	<p>Vehicle Control</p>	
<p>Power Switches and Cables</p> <p>Power and connectivity for entire system</p>		

<p>3D PC Computer with the sign “OPSCON” on the top</p>	<p>Operations Control: 3D Scenarios; Vehicle Control: CAR’s position</p>	
<p>TRAFFIC PC Dell computer</p>	<p>Traffic Control: Generation of traffic conditions</p>	
<p>Front, Left, Right and Rear View PCs Computers that have the names FRONT VIEW, LEFT VIEW, RIGHT VIEW and REARVIEW on the top, respectively</p>	<p>Display Control: Front, Left, Right and Rear View</p>	
<p>CAR PC Computer located in the driver’s seat, on the left side</p>	<p>Vehicle Control</p>	
<p>3D Monitor Monitor connected 3D PC and CAR PC</p>	<p>Vehicle Control, Operations Control, Display Control</p>	
<p>TRAFFIC Monitor Monitor connected to TRAFFIC PC</p>	<p>Traffic Control</p>	
<p>Hardware Component</p>	<p>Software Function</p>	<p>Illustration</p>
<p>Front, Left, and Right LCD</p>	<p>Display Control: Front, Left and Right PC</p>	
<p>Rearview Mirrors</p>	<p>Display Control: Rear View PC</p>	

<p>Remotes</p> <p>Remotes that turn on Front, Left and Right PC.</p>		
<p>KVM (Keyboard Video Mouse) Control board with keyboard, mouse and 8 switches:</p> <p>KVM 1 KVM 2 KVM 3 KVM 4 KVM 5 KVM 6 – Divides VISSIM and ARCHER KVM 7 KVM 8</p>	<p>Display Control</p> <p>3D PC</p> <p>Right View PC</p> <p>Front View PC</p> <p>Left View PC</p> <p>Rear View PC</p> <p>CAR PC</p> <p>TRAFFIC PC</p>	
<p>KVM Switches:</p> 		

Figure 2: Matching Hardware and Software Components of the Driving Simulator

START UP PROCESS

Three major tasks could be identified in the Driving Simulator Start-Up Process:

- 1) Power Check
- 2) System Power Up
- 3) Start Driving Scenario

Each one of these tasks will have several sub-tasks. Each sub-task will be addressed in the following manner: What should you do; How do you know if it is done right; Illustration if needed.

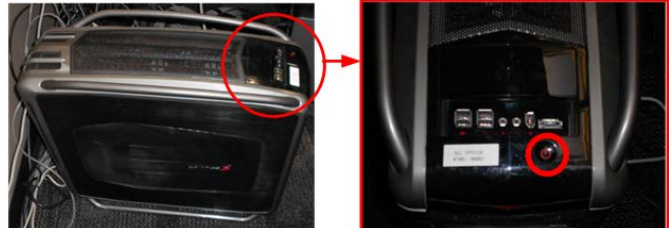
Power Check

Press the button on the front side of the power switch. Repeat this for all four power switches. If done properly a green light will show up on the power switches.

System Power Up

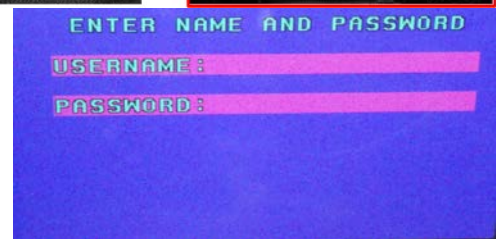
1) Press the red button on the top of 3D PC (the one with the sign “OPSCON” on top). The button should turn red light as in Figure 3.

Figure 3: 3D PC power up



3D Monitor screen should be as in Figure 4. Press the key “Enter” on the keyboard three times.

Figure 4: 3D Monitor that shows that 3D PC is on



2) Press the button on the front side of TRAFFIC PC. Press the red button on the top of each of these PCs just as you did for 3D PC. You will check if these PCs are on later. Keep going with the process.

3) Go to the CAR. You will see a key on the left side of the bottom of the driver seat. This is where the CAR PC is. Unlock it. Press the button on the CAR PC as shown on Figure 5. Go to the 3D Monitor and see if the screen is on.

Figure 5: How to Power Up CAR PC



4) Use the Remotes or the buttons at the bottom of the LCDs to turn on these screens. This is where you check if the Front, Left, Right and Rear PCs are on. If they are the LCD screens should look exactly like on Figure 6.

Figure 6: LCD Screens are on



Start Driving Scenario

1) Go to KVM and press switch KVM 7. 3D Monitor will display CAR PC on the screen. Locate icon “My Computer” on this screen and click on it twice. A “My Computer” window will open. Locate “Network drivers” on this window and click twice on “\\155.98.128.128\export\shared”. A new window will open as shown on Figure 7. Use the password “3Dmatrix” to connect.

Figure 7: Archer Login - Screen Shot

Look back to the main screen of 3D Monitor and locate icon “Steering calibration” right below the icon “My computer”. Click twice on “Steering calibration”. A window will open as shown on Figure 8. Wait for about 1.5 minutes and note the steering wheel movement. After you see “Press any key to continue...” the steering calibration is over. You should then press any key on the keyboard.

Figure 8: Steering Calibration Window on 3D Monitor Display for CAR PC

2) Locate the icon “RTI Dynamics” right below the “My Computer” icon and click twice on it. A window will open as on Figure 9:

Figure 9: RTI Dynamics

You should type in with the space included: “test <elevation file name>”. Instead of “<elevation file name>” you should type in one of the following four names:

CFI_VISSIM_Elevation.wrl
DDI_VISSIM_Elevation.wrl
PA851_lot_Elevation.wrl
Parking_lot_Elevation.wrl

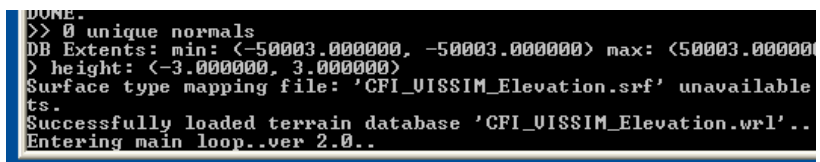
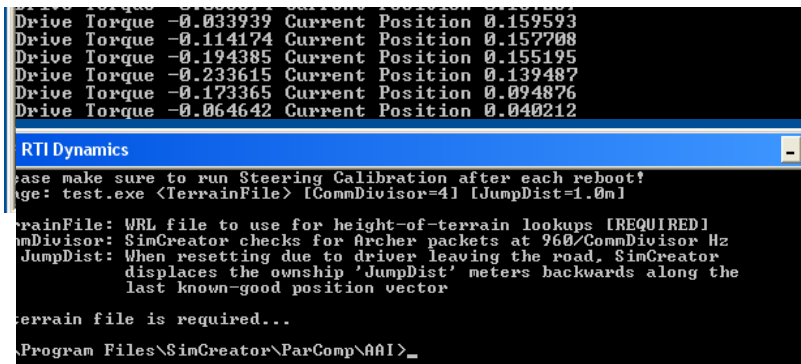
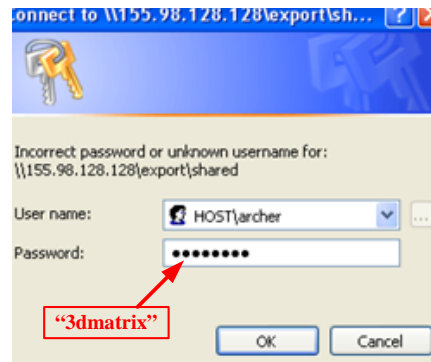
Figure 10: Loading of Elevation File Finished

Elevation file must match the scenario you want to run on the simulator. Wait until “Entering the main loop...” appears on the screen, as shown on Figure 10.

3) Press KVM 8 switch and look at the TRAFFIC Monitor. Login to “archer” with the password “3Dmatrix”, as shown on Figure 11. Pay attention on capital letters in the password.

Figure 11: TRAFFIC Monitor Login

Locate icon “VISSIM RUN “archervissim_shortcut” on the screen and double click on that icon. Wait for “Use server port 8778” to appear on the window, as shown on Figure 12. After it appears, go to the next sub-task.



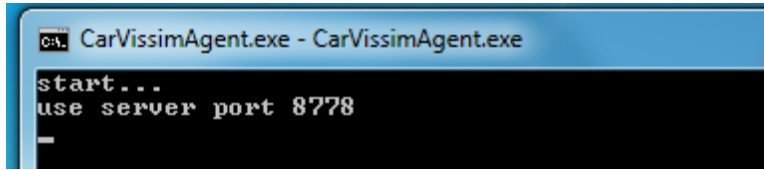


Figure 12: VISSIM and ARCHER are Connected

4) Press KVM 1 switch. Look at the 3D Monitor. Select one of the icons so that the name matches elevation file name from sub-task 2 in this task (see Figure 13). Click twice on corresponding icon (see Figure 14).

Elevation File Name	Corresponding Scenario Icon/ Name
CFI_VISSIM_Elevation.wrl	Run CFI VISSIM Demo
DDI_VISSIM_Elevation.wrl	Run DDI VISSIM Demo
PA851_lot_Elevation.wrl	Run PA851 Demo
Parking_lot_Elevation.wrl	Run Parking Lot Training Demo

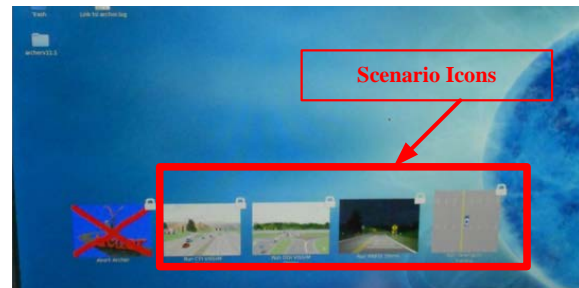
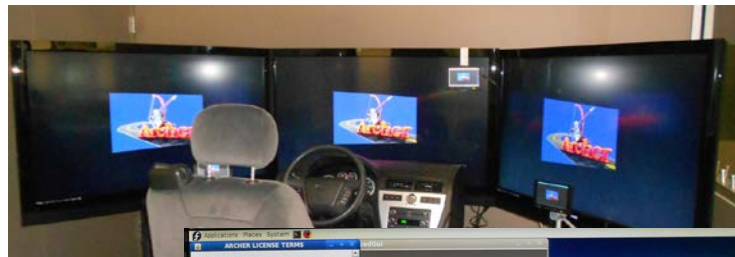


Figure 13: Elevation Files and Scenario Names

Figure 14: Possible Scenarios – Screen Shot

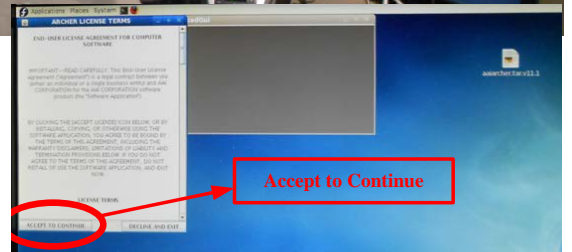
After you click on adequate scenario icon, the screen logos “Archer” will start rotating on LCDs and Monitors as shown on Figure 15:

Figure 15: Archer Starting the Chosen Scenario



5) Wait for the screen logos to stop rotating. Look at the 3D Monitor. A window will open offering you to accept license terms. Click once on “Accept to Continue” as shown on the Figure 16.

Figure 16: Accepting the License Terms



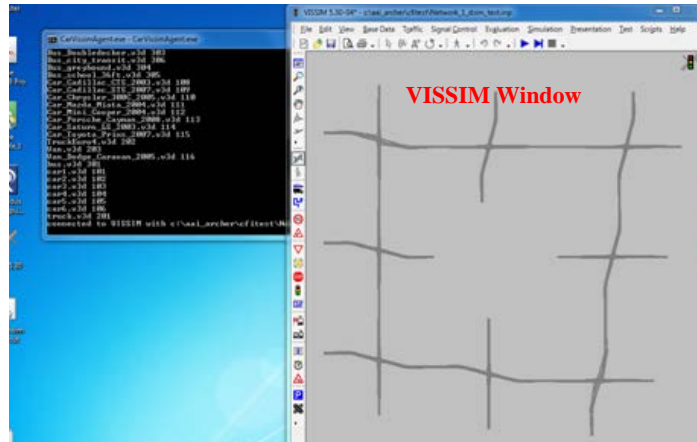
Pay attention to the panel that will appear on the screen with the name of the scenario you loaded, as shown on Figure 17. Type in the name you want to in the “name” field. Wait for the engine noise. Once you hear the engine noise, click one on “Start Session”, as shown on the Figure 17. All LCD screens should be displaying traffic movements if everything is done correctly.

Figure 17: Panel for Operations Control



6) Press KVM 8 switch. Look at the TRAFFIC Monitor. Click on VISSIM window as shown on Figure 18, and press “CTRL+Q” twice on the keyboard.

Figure 18: VISSIM Window on TRAFFIC Monitor Display



Press KVM 1 switch and look at the 3D Monitor. If you see the speed/distance recordings on the screen changing colors the system is on (see Figure 19)

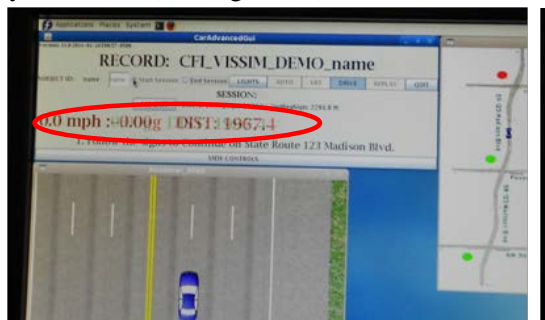


Figure 19: The Simulator is Ready for Driving

7) Go to the CAR and have a seat. Use the gear shifter to put the car into drive and drive!!!

START UP PROCES SUMMARY

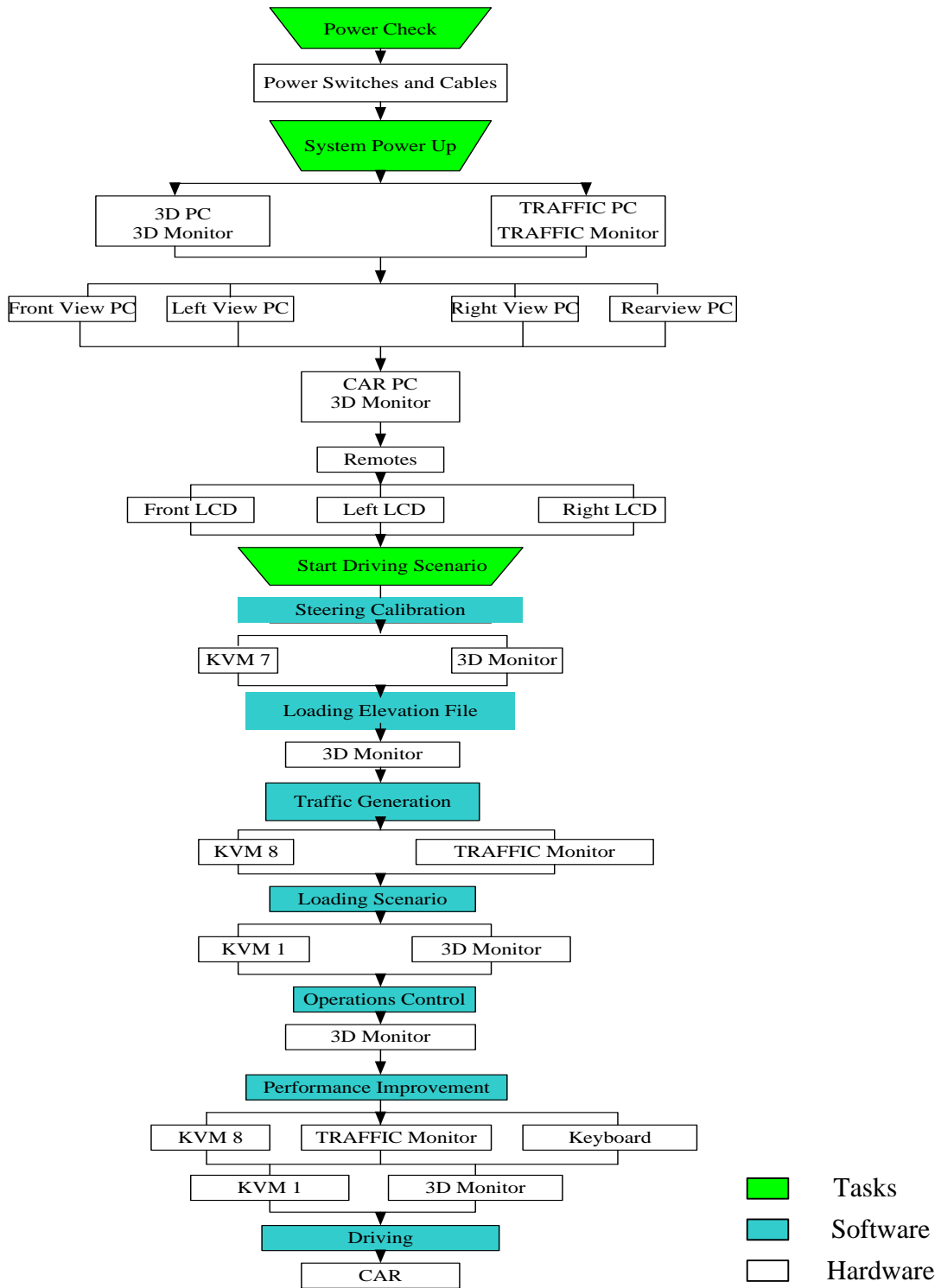


Figure 20: Start Up Process Summary: Hardware and Software - Flowchart