Multiuse, High Accuracy, High Density Geospatial Database

Final Report

Prepared by:

Bryan Newstrom
Curtis Olson

Intelligent Vehicles Lab
Department of Mechanical Engineering
University of Minnesota

CTS 09-05

Technical Report Documentation Page

1. Report No. CTS 09-05	2.	3. Recipients Accession No.
4. Title and Subtitle Multiuse, High Accuracy, High Density Geospatial Database		5. Report Date February 2009 6.
7. Author(s) Bryan Newstrom, Curtis Olson		8. Performing Organization Report No.
9. Performing Organization Name and Address Intelligent Vehicles Lab Department of Mechanical Engineering University of Minnesota 1100 Mechanical Engineering 111 Church Street SE Minneapolis, Minnesota 55455		10. Project/Task/Work Unit No. CTS Project # 2005047 11. Contract (C) or Grant (G) No.
12. Sponsoring Organization Name and Address Intelligent Transportation Systems Institute Center for Transportation Studies University of Minnesota 511 Washington Avenue SE, Suite 200 Minneapolis, Minnesota 55455		13. Type of Report and Period Covered Final Report 14. Sponsoring Agency Code
15. Supplementary Notes http://www.cts.umn.edu/Publications/ResearchReports/		

16. Abstract (Limit: 200 words)

High accuracy (2-8 cm) DGPS and high accuracy (5-20 cm) geospatial databases (or to use the term loosely, “enhanced digital maps”) are the primary components of the IV Lab driver assistive systems. In addition to vehicle-based systems, the IV Lab geospatial database has found utility in other applications. For instance, the database has recently been used for a new Intersection Decision Support (IDS) project, where radar sensors are used to determine the state of an intersection as a first step in warning drivers when it is unsafe to enter a thu-Stop intersection. The geospatial database is used in this application to improve the ability of the radar system to determine whether a target represents a legitimate threat at the intersection.

The IV Lab geospatial database was designed and optimized for vehicle applications, and provides real time access to extremely accurate, dense geospatial data. Because of this optimization, its functionality in other applications is somewhat limited. As new applications arise (i.e., the need to integrate high accuracy geospatial data into a driving simulator, the desire by DOTs to more accurately represent roads, rights of way, etc.), a more “global” approach to the design of the existing geospatial database is required. Described herein is a redesign of the geospatial database and database manager and the development of a new “front end” to serve a wide application base.

17. Document Analysis/Descriptors Geographic Information System, Map Database, Terrain Data		18. Availability Statement No restrictions. Document available from: National Technical Information Services, Springfield, Virginia 22161	
19. Security Class (this report) Unclassified	20. Security Class (this page) Unclassified	21. No. of Pages 34	22. Price

Multiuse, High Accuracy, High Density Geospatial Database

Final Report

Prepared by

Bryan Newstrom
Curtis Olson

Intelligent Vehicles Lab
Department of Mechanical Engineering
University of Minnesota

February 2009

Published by

Intelligent Transportation Systems Institute
Center for Transportation Studies
University of Minnesota
511 Washington Avenue SE, Suite 200
Minneapolis, Minnesota 55455

Acknowledgements

The authors wish to acknowledge those who made this research possible. The study was funded by the Intelligent Transportation Systems (ITS) Institute, a program of the University of Minnesota's Center for Transportation Studies (CTS). Financial support was provided by the United States Department of Transportation's Research and Innovative Technologies Administration (RITA).

Table of Contents

List of Figures

Executive Summary

The IV Lab geospatial database was designed and optimized for vehicle applications, and provides real time access to extremely accurate, dense geospatial data. Vehicle applications include (but are not limited to) “machine control” for construction equipment, snowplows which operate in low visibility conditions, and transit buses which operate on a narrow shoulder. Because of this optimization, its functionality in other applications is somewhat limited. As new applications arise (i.e., the need to integrate high accuracy geospatial data into a driving simulator, the desire by DOTs to more accurately represent roads, rights of way, etc.), a more “global” approach to the design of the existing geospatial database is required. Described herein is a redesign of the geospatial database and database manager, and the development of a new “front end” to serve a wide application base.

To accommodate the new applications and subsystems that need access to the spatial database, the database data model has been redesigned. The data model of the database has been restructured to better model the physical layout of roads. The data model defines roads that are subdivided into legs. Legs correspond to each direction of travel for a road; e.g. northbound and southbound lanes for an interstate highway. Road legs are further subdivided into lanes. This new data model is more generic than the original data model but still captures the required data for the new generation of subsystems. A more generic approach will allow better interoptability with other applications outside of the original scope.

The database should conform to open standards used in the GIS world. These standards include geometry models and physical storage formats. The redesign of the database data model was performed within an open source spatial database. Being that the open source spatial database already conforms to open standards, the new data model will then conform as well. Working within the standards has greatly enhanced the interoptability of the IV Lab database to outside applications and new vehicle subsystems.

By using a new database query scheme, the requirement to have fast or real time access to the database has been lessened. This new scheme involves caching data for a larger area, which can be used for a longer period of time before subsequent queries of the database become necessary. In fact, the query speed and latency requirements have been reduced greatly so that a specially designed database manager is no longer needed.

To illustrate the interoptability of the new IV Lab databases and data model, a tool set has been developed to convert the IV Lab database into scenery that can be used within a vehicle simulator. The process of merging and transforming 2D and 3D GIS datasets into a 3D visual scenery database appropriate for real-time simulation and visualization is highly difficult and complex. A process to generate 3D simulator databases was successfully created and demonstrated. This process combined detailed IV Lab map data of actual Minnesota roadways with larger scale, but less accurate GIS terrain and land use data to create an immersive and natural simulator world.

Chapter 1: Introduction

Two contrary approaches to handling geospatial data are available. One approach is to package geospatial data in CAD formats. Photogrammetry data is typically made available in CAD file formats. In a CAD file format, raw data is available, but no attributes are assigned to the data. It is the responsibility of the user of that data to review data, extract elements of interest, and assign attributes to the extracted data. This data is easily edited, but difficult to analyze because data is provided without context.

In contrast are geographic information systems (GIS) tools, which include Arcview. GIS typically supports databases which allow the assignment of attributes to geospatial data. Users are able to query the databases using attribute qualifiers, enabling a user to view only desired elements of the database. GIS fails in the ability to easily edit data contained within the database. The inability to easily modify geospatial data renders standard GIS systems ineffective for many design and research applications.

The objective herein is to bridge the gap between these two extremes, enabling a user the ability to edit geospatial data with the precision of a CAD system while preserving the capability to assign and utilize attributes associated with the geospatial data.

Three specific areas are to be addressed through this research project:

- applications,
- the database structure, and
- the database manager.

The research is aimed at creating a database structure and database manager which functionally lies midway between the generic, low accuracy, low density geospatial databases and the specialized, high accuracy, high density databases. The key is “functionally;” the data model will be redesigned to be more “spatially aware,” utilizing a more generic, standard tree structure.

Likewise, the database manager will be redesigned to exploit the new database structure and provide access to high accuracy, high density data in real time. Finally, a set of tools to provide access of the IV Lab database to applications typically using low accuracy, low density databases to applications requiring higher fidelity (driving simulation being a first target) will be developed and demonstrated.

Chapter 2: Original Database Design Limitations

The database structure and database manager was originally designed by the IV Lab for onboard driver assistive systems for use in low visibility driving conditions. The data model used to store data in the database was originally designed specifically for three subsystems:

- the Head Up Display (HUD),
- radar target processing, and
- virtual rumble strip.

Each subsystem has its own requirements that define how the data was modeled and structured within the database. Each subsystem requires data describing some part of a road, or lane of traffic. The HUD requires the lane boundaries, or the paint marking, to display to the operator. Radar target processing needs the extents of the road to determine whether a radar target is on the road to determine whether the target is a threat to the operator. The virtual rumble strip requires the distance between the vehicle and the center of the current lane and the width of the current lane. It uses this distance and width to determine whether a lane departure warning is needed to alert the operator.

A database data model represents how the data is structured within the database. A database includes the data and how it is structured. A database manager is a software program that provides access to the database. The database is comprised of the files on a computer disk, the data model is how the data is structured within those files, and the database manager accesses those files on behalf of other programs. The database manager is the gateway to the data for other programs and software. In the original database system, both the data model and the database manager were custom built.

The database manager was designed to query the database and deliver data to the onboard subsystems in a timely manor. The data was structured in a hierarchal tree that was optimized to process only spatial intersection queries. The subsystems would define an area of interest and the database manager would determine all the data that intersects that area. The results were then packaged by the database manager and delivered to the subsystem.

For an in-depth description of the original database and database manager design, see [[1]].

Within the context that the original database was designed, the database and database manager preformed incredibly well. As more subsystems were developed and more roads were mapped, the limitations of the original design became apparent. The main limitations are how the data was created for the database and how the data was stored. Also, other schemes for querying the database have been developed that reduce the need for extremely fast or real time access to that database.

The original database stored raw data in a database-specific structured text file. This text file would then be parsed and turned into a binary data file. The binary file was then used during

runtime of the database manager. The text file is the result of processing raw data used to create the map. The advantage of the binary data file is that it is faster to read and access than the text file. The limitation is that the text file needs to be converted to load the data into any other program. For example, to edit the data in a GIS application like ArcView, the text file would need to be converted to a shape file. After the editing, the shape file would need to be converted back to the structured text file. Since the text file used a custom format, custom software is needed to do the conversion. The text and binary files were designed for the database manager software, not for the database creator. This is a severe limitation when it comes to editing and creating a geospatial database from raw data. Further development determined that the binary files were unneeded, since the databases were small and were completely loaded into memory by the database manager. The database manager software would load the data from the binary files on startup and never access the binary files again.

The need to query the database at high rates can be lessened by caching data locally by each subsystem. Querying the database can be done at a lower rate but for a larger area. This way the local copy of the data is relevant longer and can be updated only when needed.

Another limitation to the original design was the inclusion of curves into the geometry types. The data modeling the center line of a lane was segmented into lines and curves. A curve was defined as a section of a circle. The original thinking was that this simplified the processing that had to be done by specific subsystems. The problem is that a curve is a nonstandard geometry that is not used in most, if not all, GIS applications. This made editing data in a standard GIS application almost impossible. It also complicated pre-processing of the data, since special algorithms were needed to fit lines and curves to raw data.

Chapter 3: The New Data Model

The data model of the original database was completely defined by the three subsystems (HUD, radar processor, and lane departure warning) that used it. The new data model should include equivalent constructs as in the original data model, but should also be more generic and extensible for new applications and subsystems of the future. The new design should also take into account existing open GIS formats and GIS tools, and be closer in structure to existing digital maps and road network databases.

These requirements are all intertwined. For example; a generic data model makes interoptability with other programs easier and puts the database closer to existing digital map structures. Conforming to open standards for geometry does this as well.

The new data model was designed completely within the existing GIS application PostGIS [[2]]. PostGIS is a spatial extension to the open source database PostgreSQL [[3]]. By doing this, two of the requirements are already met. From the start the database follows open standards. PostGIS follows the OpenGIS "Simple Features Specifications for SQL" specification which includes a definition of standard geometry types [[4]]. Also, since PostGIS follows these standards, it is easy to use tools like GDAL/OGR to convert the data to and from other formats [[5]]. GDAL/OGR is an open source library that allows access to many commonly used spatial and GIS formats, including access to a PostgreSQL database using the PostGIS extension and access to shape files, which are the *defacto* standard GIS format.

The first thing to tackle with the new data model was handling the curve geometry used in the original database. The original database defined curves as a section of a circle and this type of geometry is not supported by the standard geometry types. The solution was to use a linestring and define a curve using linear referencing. A linestring is simply a series of points. The definition of a linestring in the new database is the same as that used in the original database. The center of a lane can be modeled by a linestring. The curve would be defined by the start and end distance along a linestring, along with other curve attributes like its center point, radius, or curvature. The start and end distance are a rectilinear distances along a reference linestring. PostGIS supports linear references and they are easily converted to regular geometry buy eliminating the specified section of the reference linestring. SeeFigure 1 for the differences between the original database geometry and the new database geometry regarding curves. Also, existing digital maps and road databases rely heavily on linear referencing. Using this scheme does not remove the need to fit curves to data, but makes this optional. The linestring defining the overall geometry does not depend on the curve sections and can exist without the curves.

The requirements of the new data model should include the original requirements but be more general and flexible to handle new challenges. In an attempt to handle the unknowns of the future requirements, the new data model leverages the current GIS tools. The original data model has been simplified and reorganized into a more tree like hierarchy that better reflects road structure. This new model can be thought of as the core model by which other object types can be added and removed as needed. Also, the database has been purposely designed to be simple and flexible, knowing that it will change in the future. Leveraging the current GIS applications, like PostGIS, and GIS processes will make changes easier to implement.

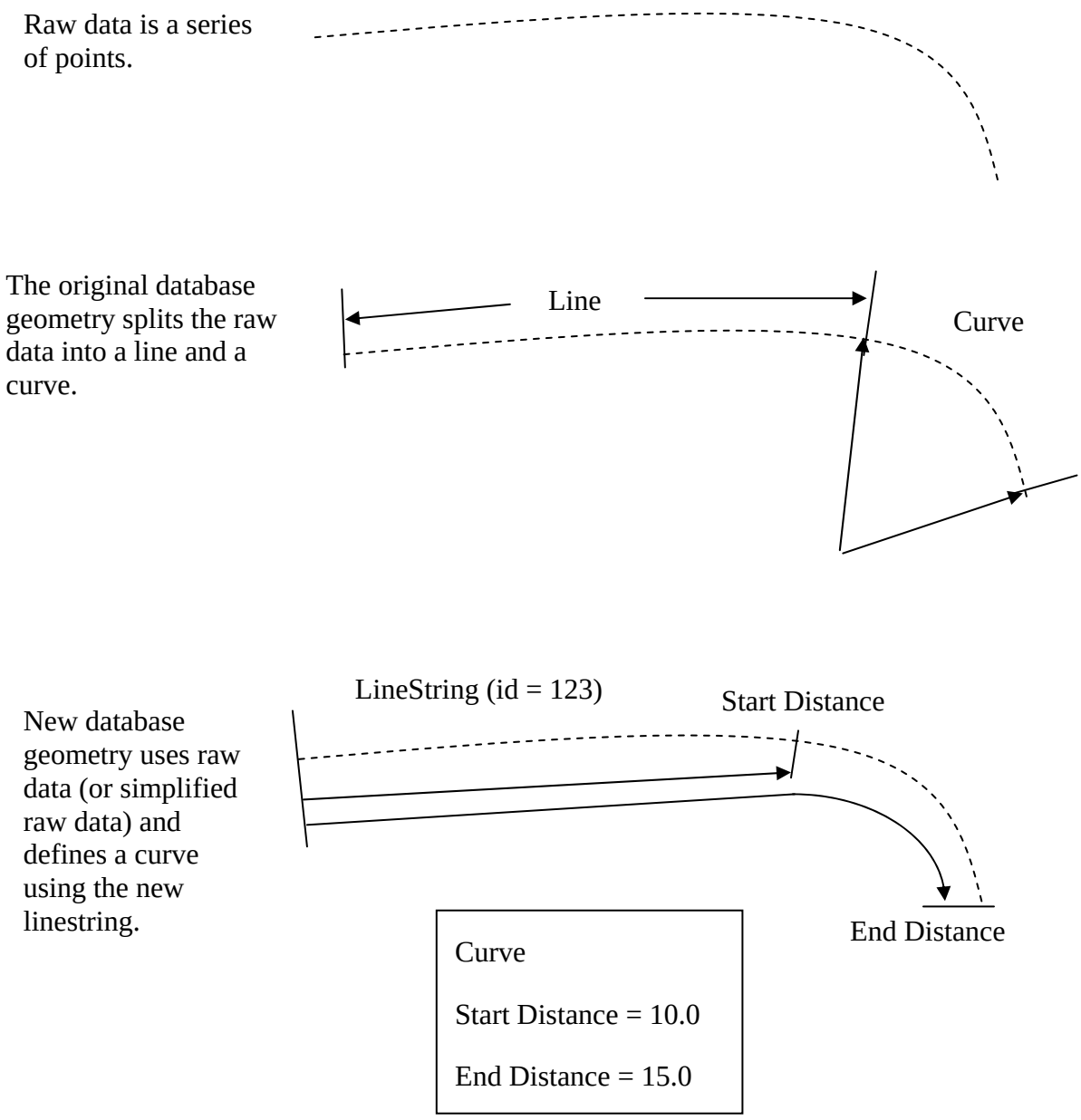

Figure 1. Redefining a curve.

The new core data model better models the hierarchal structure for roads. It defines a road as being composed of legs. Each leg is in turn is composed of lanes. Lanes can then be further decomposed into straight sections and curves, by using linear referencing as described before.

See Figure 2 for a graphical depiction of the data model and Figure 3 for the complete data model and hierarchy relationships.

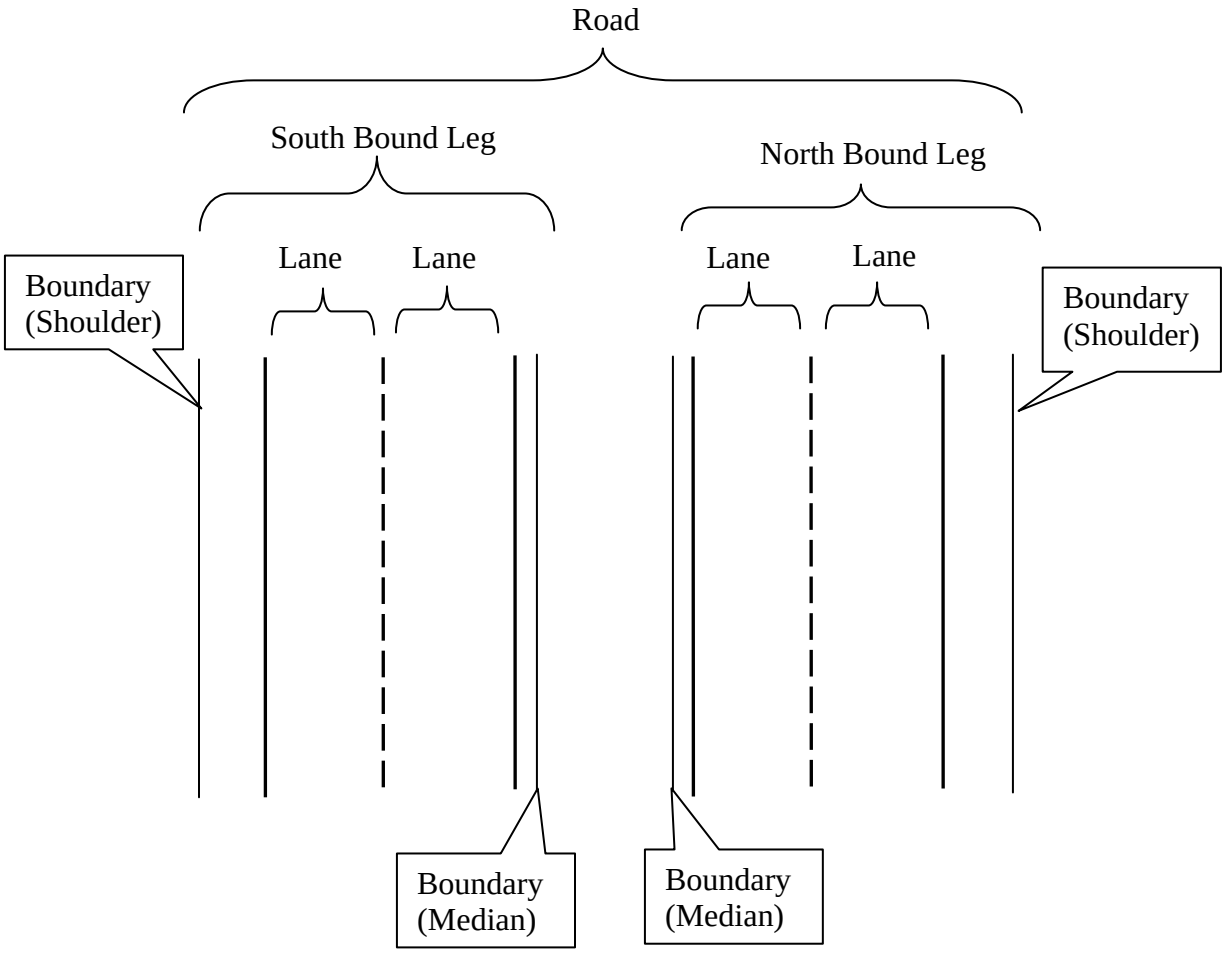

Figure 2. Graphical depiction of data model.

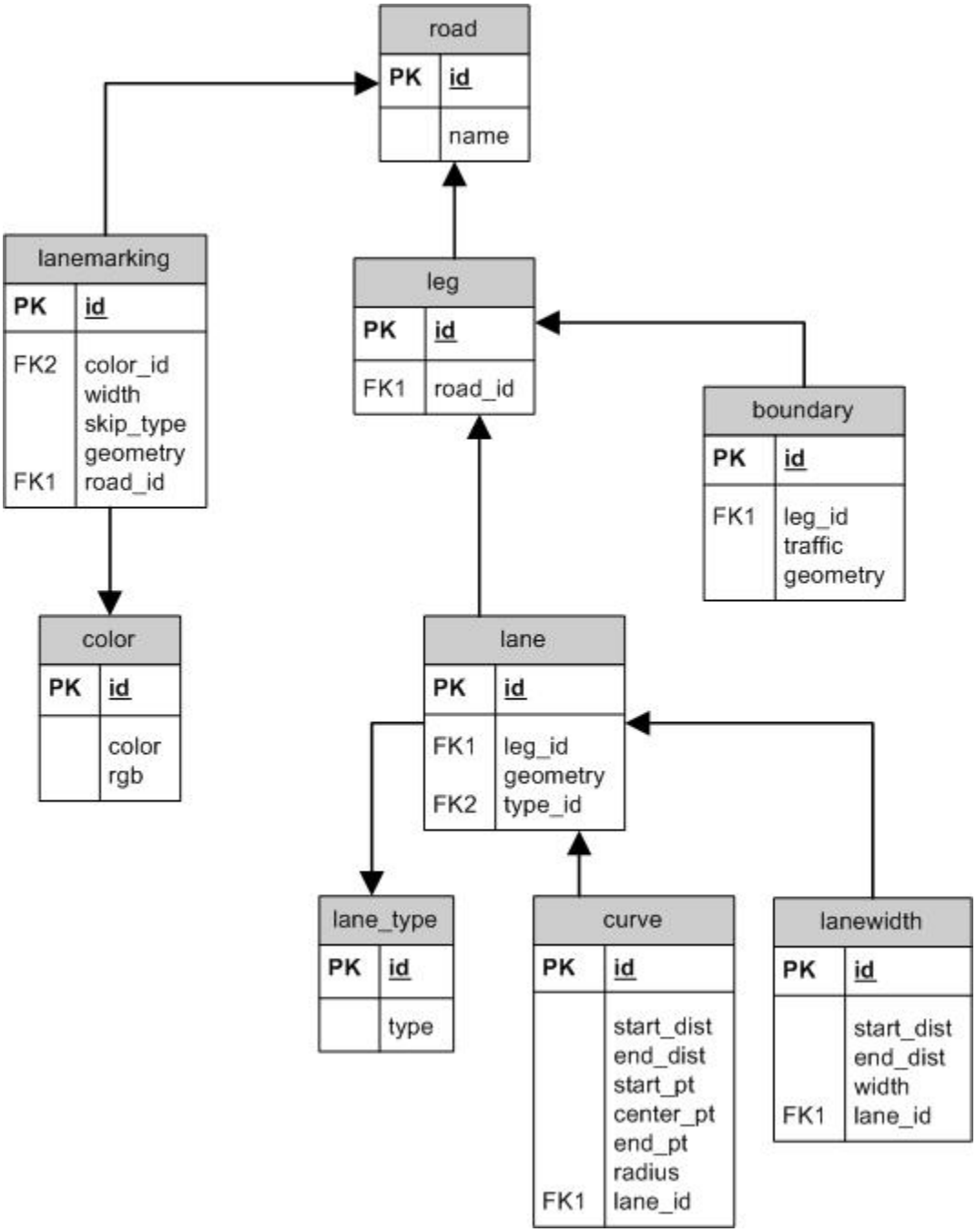

Figure 3. Core road data model. PK is Primary Key and FK is Foreign Key. In a SQL database, primary and foreign keys are used to link different rows of different tables and show relationships in the data model. The arrows show the relationships; e.g., a lanemarking object references a road object.

The Road object contains a unique id number and a road name. This is to uniquely identify the road within the database. A coarse geometry, or geometry from an existing digital map, could be included. The Road object can serve as a common link between this database and existing digital maps and road databases. Current consumer digital maps are used primarily for route planning and turn by turn directions and do not need more detail than roads.

The Lanemarking object contains the geometry of any paint markings applied to the road. This object is used to generate the scenery used within the Head Up Display subsystem.

A Leg object defines one direction of a Road object. Generally, an individual road has two directions; for example, a northbound and a southbound leg of an interstate highway. The Leg object does not contain geometry but does link together lanes that do have geometry. Leg objects aide in radar target processing; always showing oncoming traffic as a threat is considered to be a false positive. Oncoming traffic is only a threat if it is in your lane or leg.

Along with the Leg object; the Laneboundary object is used to filter radar targets. The Laneboundary object is defined as the extents of a Leg. This would correspond to the edge of pavement or the center line of a 2-way road.

The Lane object depicts the geometry of the center of any lane. The Lane has a type qualifier specifying the type of lane. Examples include a normal traffic lane, a turn lane, and a shoulder lane. The width of the lane is specified by linear referencing the distance at which the lane is the given width. Currently, a Lane object is the only linestring geometry that the knowledge of curves is required.

This data model covers all needed information about roads that the current driver assistive subsystems and intersection decision support subsystems require. It models the physical properties and relationships for real roads better that the original data model and still allows for future development. Since the data model requires no non-standard geometry types, the database can be stored in a variety of standard GIS data formats. The ability to use standard GIS data formats greatly expands the number of programs and applications that can access, edit, or generate the database. As an example, Figure 4 shows a portion of an IV Lab database overlaying satellite imagery within Google Earth. In this case, using an application from the GDLA/OGR library, the database was converted from shape files to the KML format used by Google Earth.

It is important to note that the accuracy of Google Earth data varies both between states and even within a single state; population centers are generally represented by more accurate data. To provide equal comparisons across the entire US, US Geological Survey (USGS) digital ortho-imagery would provide consistent comparisons.

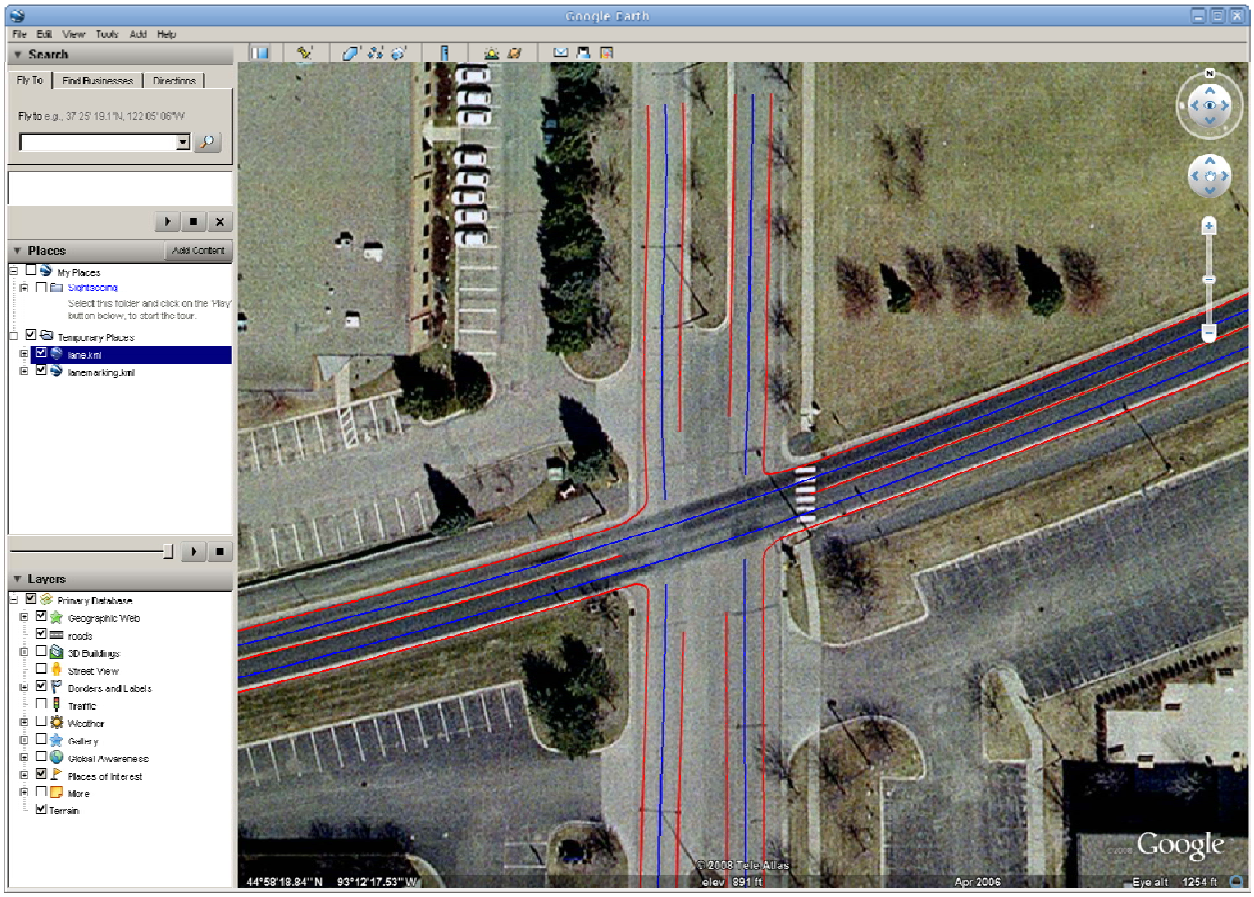

Figure 4. IV Lab database overlaying satellite imagery in Google Earth. The red lines are Lanemarking objects and the blue lines are Lane objects.

Chapter 4: The New Database Manager

The original database manager was designed and optimized to process intersection queries. The database manager would determine all data that intersected a given area. The client or subsystem requesting road data would query the database though the database manager.

The database manager would read in a binary format data file and create a tree structure of the entire database in memory. The database was "tiled" into discrete sections that were further segregated into lists of distinct data types. This was done so that query processing would be as fast as possible. The database manager did not do the tiling; the person creating the database was responsible for cutting the database into sections. This simplified the code base of the database manager but complicated the database creation process.

To process a query, the database manager determines which tiles the query polygon intersects. The database manager would then determine which objects, within the selected tiles, intersect the query polygon. (See Figure 5 for a depiction of the query processing.) Using the tiles the database manager could very quickly disregard sections of the database that it would not have to process in more detail. This was all done to make query processing very fast so the subsystem could query the database at a high rate.

Using a temporary local cache, the frequency of queries can be lowered. If a subsystem were to query the database using a larger area; it would not have to query the database as frequently and could tolerate longer query times. For a vehicle-based subsystem the lifetime of the local cache is dependent on how large the query area is and the speed at which the vehicle moves through the local cache. The subsystem needs to maintain the local cache by watching the distance between the original query position and the current position. If this distance is outside of some tolerance, then the database needs to be re-queried and the local cache rebuilt.

Readily available open source tools, like the GDAL/OGR library and the PostgreSQL database using the PostGIS extension, can be used as the database manager. Both of these systems implement spatial indexing analogous to the tiling scheme of the original database manager. Spatial indexing is a generic name for arranging the data in some sort of hierarchical tree based on spatial properties. The end result is that the number of objects to be queried can be quickly reduced to only the objects that are relevant to the query.

Using either the GDAL/OGR library or the PostgreSQL database as the database manager in conjunction with the caching query scheme, the IV Lab has found little or no performance degradation when used for on-board vehicle subsystems. This is now the preferred method to access road databases for current and future IV Lab development.

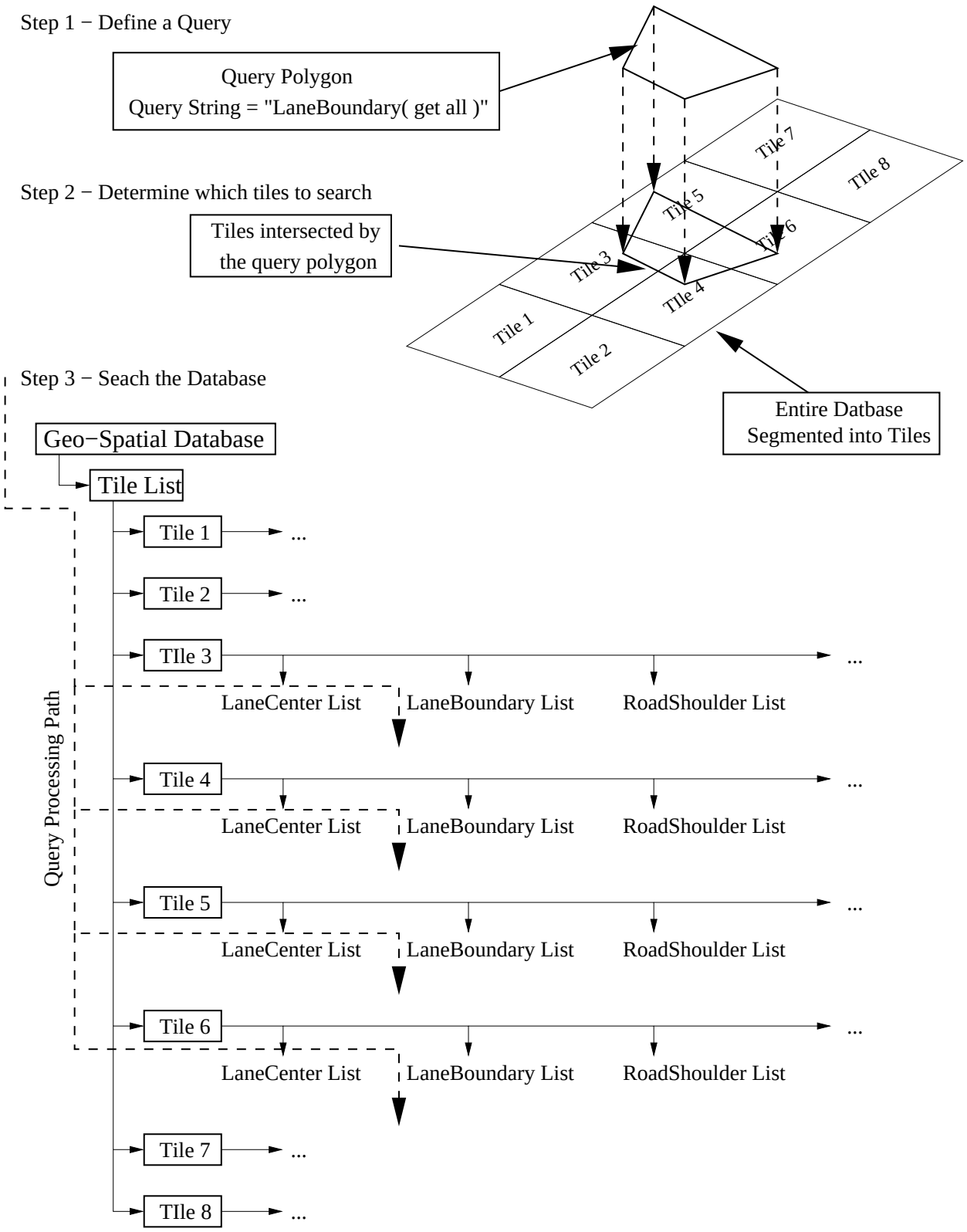

Figure 5. The original database manager query process.

Chapter 5: Tool Set Development

Background

The final task of this project is to develop a tool set that transforms the largely two dimensional (2D) IV Lab database data into a three dimensional (3D) virtual environment for use in the HumanFIRST immersive driving simulator. This transformation process poses several significant challenges.

1. A real time simulation or visualization system uses triangles as its rendering primitive. All of the input database data must be converted to triangle structures for real time rendering.

2. Triangles must not overlap. Overlapping triangles lead to an unwanted effect called z-buffer fighting which creates flashing and distracting artifacts in the visual scene.

3. When building a 3D scene from a variety of data sources, it is very possible that the original 2D data will overlap with itself in a variety of areas and in a variety of ways. These overlaps need to be removed from the data in a systematic way.

4. Most objects and surfaces need to be textured (overlaid with patterns, art work, or digital photos) in order for the result to appear natural and realistic in a real time simulator.

5. When dealing with real world data and automated algorithms, it is very easy to overload the available structure sizes and rendering capacity, so various simplification algorithms need to be employed.

6. Real world data often includes inconsistencies or incompatibilities that can cause problems downstream in the tool chain. These inconsistencies could be introduced in the data collection stage, the database management stage, or even when combining multiple databases from different sources. For example, a data collection mistake (or even a lower resolution data set) could place a feature in a slightly incorrect location in the database. This could lead to things like a road passing through the middle of a lake, or a building in the middle of a road. A large amount of effort goes into identifying inconsistencies within a data set or in the combination of data sets and finding various ways to work around the issues.

7. A simulator is an immersive environment, whereas the IV Lab high-accuracy, road mapping data covers only small portions of the world and only deals with specific aspects of those areas. A system designed to produce realistic 3D worlds needs to be able to fill in the space around the precise data with less precise data in all the surrounding areas so as to create an immersive and continuous world.

This chapter describes an approach for building a software tool chain that can

1. take an arbitrary set of 2D input data layers,

2. resolve any overlapping areas to create a seamless surface with no overlaps or cracks,

3. drape this 2D data over three dimensional terrain data, and

4. robustly texture map the resulting structures.

This process can create a large scale, continuous 3D visual database suitable for real time simulation and visualization, beginning with a precisely mapped roadway dataset. This system is able to pull in larger contextual data sets to fill in the world around the precisely mapped data.

Coordinate Systems, Tiling, and Numerical Precision

Historically GIS data has been stored in a wide variety of formats and coordinate systems. To combine data from a variety of sources into one unified scenery database, a single consistent coordinate system and geoid must be chosen. This project uses longitude and latitude coordinates (in degrees) in the WGS84 world geodetic system. This is the coordinate system used to manipulate data in the toolset pipeline. A coordinate system and corresponding geoid is shown in Figure 6 below.

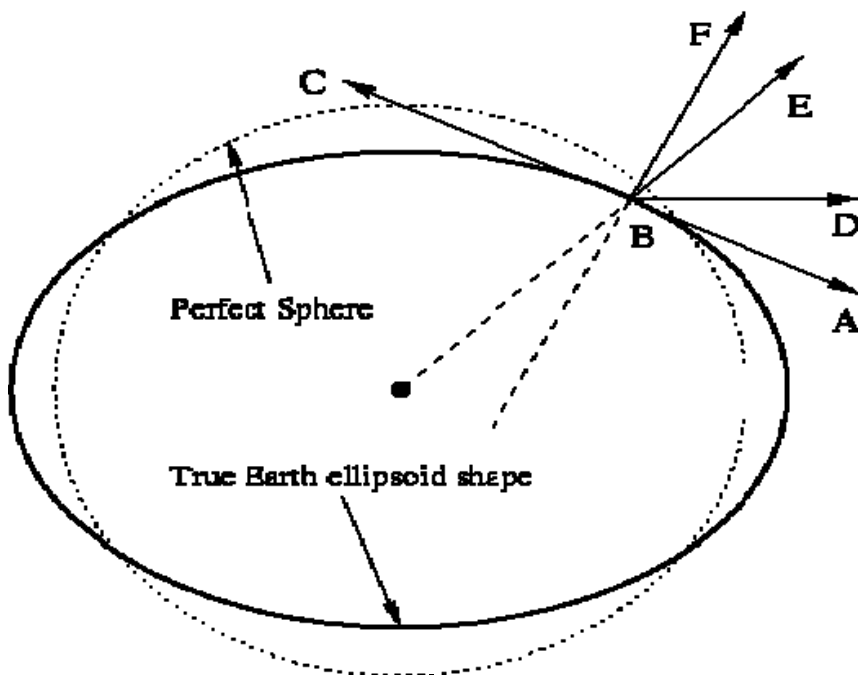

Figure 6. Oblate ellipsoid Earth cross section.

In order to efficiently handle large areas of data, the world must be broken up into sections called tiles. Conceptually, a tile is simply a section of the surface skin of the Earth. Tile surfaces maintain the original Earth curvature and orientation relative to an Earth centered Cartesian

coordinate system. Tile edges run parallel to the longitude and latitude lines on a map. A representation of a tile is shown in Figure 7 below.

For this project, a tile size of 1/8th of a degree longitude and 1/4 of a degree in latitude has been chosen. Tile sizes will vary by latitude, but for Minnesota, this scheme yields tiles that are approximately 6.4 miles East-West and 18 miles North-South.

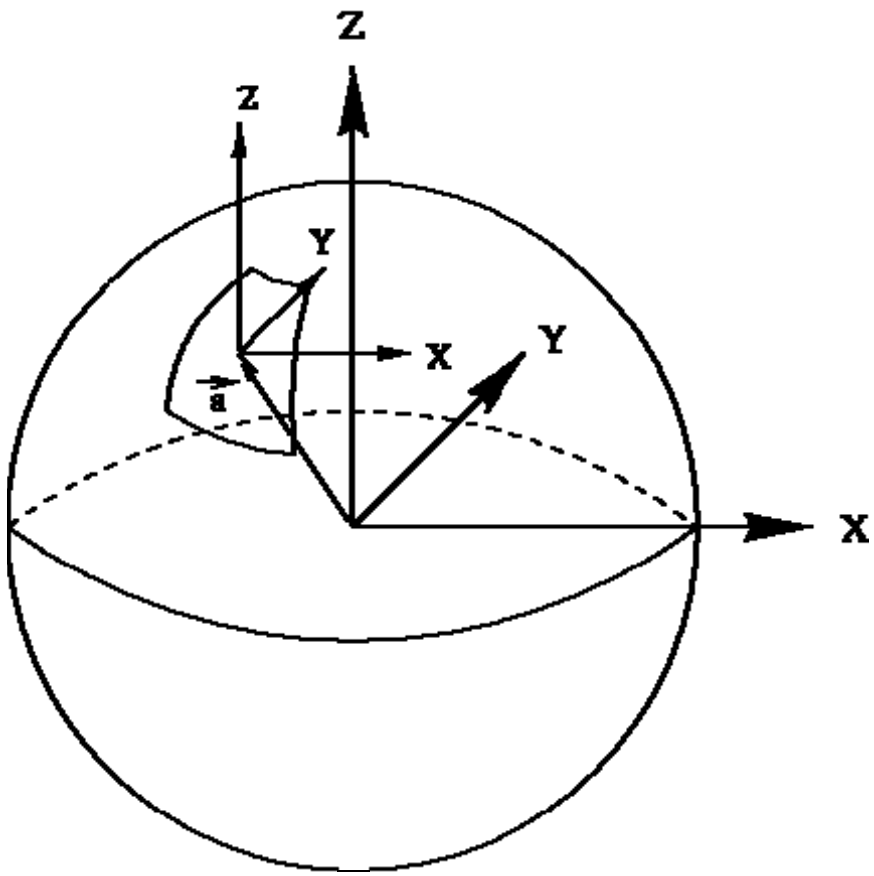

Figure 7. Representation of a tile.

An important consideration when dealing with GIS data sets is the subject of numerical precision and how computers represent real numbers internally. One limitation of traditional computer hardware is that software is forced to represent arbitrary real numbers with a limited set of bytes. This leads to a number of potential problems and influences many decisions later in the pipeline.

For example a standard 4 byte IEEE floating point number can represent about 7 decimal points of accuracy. The radius of the Earth at the equator is ~6,378,138 meters. If the Earth's radius is represented using a standard 4 byte floating point number, due to precision limitations, the radius can only be known to the closest 10 meters. There are a number of methods employed to represent world coordinates more accurately; however, the basic limitations of a computer's ability to represent real numbers is a constant challenge and concern throughout each stage of this toolset.

The effects of these errors can be reduced by using an 8 or 16 byte floating point representation, but even though these errors can be reduced, the errors and the issues they introduce never go away. Any time that floating point numbers are used to represent GIS data, the limits and challenges imposed by floating point numbers need to be considered.

At a top level it would not seem like this would be a critical factor, but deep inside the critical algorithms of this tool chain (polygon clipping, Delaunay triangulation, edge matching, and surface smoothing, etc.) there are significant sensitivities to numerical precision. Arbitrary real world GIS data has a propensity for exposing these issues and forcing the developer to deal with them.

Input Data

GIS data is often organized in one of several common high-level constructs. The data can be represented as a series of points (such as a set of waypoints in a handheld GPS system). Points are dimensionless, but they may represent something with real size, such as the location of a town. We may want to generate a simple town outline from a single point location in order to represent the town visually. GIS data can also be in the represented as a series of line segments (such as the center line of a river or the track record from a handheld GPS system). In this case the lines mark important features but the width, length, and height of the feature may not be known. There is often a need to generate an area from a single line. For example, lane data for a roadway may simply define the center of the lane, but to see the lane, a 2D area to represent the area of pavement must be generated. Data can also be represented as a polygon area (such as the outline of a lake or the outline of a county). Finally, data can be represented in a raster form (such as an aerial image or a terrain height map).

The primary challenge for developing this tool chain is to take a wide variety of data sources in different formats, different spatial accuracy, different coordinate systems, with an inconsistent set of attributes and somehow combine them all together to create a single seamless world that is visually compelling and realistic. Despite the National Standards for Spatial Data Accuracy (NSSDA) being the current standard to use in expressing horizontal and vertical position accuracy for a data set, data acquisition processes produce data sets of various accuracy, resolution, and density.

Thus, the first stage of the pipeline is to develop a series of tools that can accept a variety of source data in a variety of coordinate systems, and convert those data sources into a single consistent WGS84 coordinate system. In the case of point and line data, these tools often have to make intelligent decisions in order to generate a corresponding 2D area for the data based on various data attributes, heuristics, or even user input. For instance, it is clear that lanes are often not exactly 12 feet wide and there are often variations introduced when the line markings are painted or repainted. Given a set of lines that represent the center of the lanes of a section of roadway, the first stage tools can use data attributes or external knowledge that a lane is typically 12 feet wide to generate a 2D polygon that represents the surface of that lane.

In addition, these first stage tools are aware of the basic world tiling scheme and will divide the 2D areas along these tile boundaries.

After this first stage is completed, all the data has been initially processed so it is in a homogeneous form. The tool chain can then deal with constructing each tile individually. The tool set can take the outline of North America as one of its input data sets; after the initial processing and division on tile boundaries, only the tiles covering the Twin Cities Metro area, or only the tiles covering San Francisco (where land/ocean boundaries are important) maybe be of interest, and only those will be generated.

Data Reduction and Simplification

In many cases, the source GIS data for a particular region is far too complicated and detailed to draw in real time. For instance, the Shuttle Radar Topography Mission (SRTM) data set provides 1 arcsecond terrain data for the continental United States. Each one-degree by one-degree area contains 12,960,000 data points, or 405,000 data points per tile. This is far too much data for real time rendering, so a data reduction scheme needs to be employed.

In the case of line data, the data points in a high density data set may need to be decimated so as to produce a less dense set of points that match the original shape with some small error tolerance. Typically these data reduction algorithms allow the user to define a maximum error tolerance (i.e., all points in the output data set must be within a certain number of meters or centimeters from the original data set).

Data Artifacts

When adapting data designed for one purpose (such as navigation) to another purpose (such as real time 3D rendering) inevitably there will be artifacts in the source data that generate issues when adapted to a new purpose. The data artifacts often are unimportant and have no adverse affects on the original use of the data, but can be very disruptive when the data is adapted for new uses.

Limitations in sensor technology and sensor resolution can often lead to difficulties when adapting the resulting data for use in 3D rendering. Artifacts and issues which show up in many data sets do not necessarily reflect negatively on the creation process for the original data; it is simply one of the many challenges faced when adapting data intended for one use to a new set of uses. This is a critical challenge faced by any tool set. Data artifacts can be infinitely varied, and their effects can range from simple distractions to crashing or confounding an algorithm and disrupting the entire tool chain.

Furthermore, the combination of diverse data sets can lead to additional numerical artifacts not present in any of the source data sets individually.

For example, the SRTM data set used to generate the terrain elevations for the final model is subject to 5-10 meters of elevation error due to noise in the sensing process. Areas of water may not have any height data recorded at all. Downtown areas may be very noisy due to the tall buildings. Vegetation and other surface properties can affect the quality of the data as well. However, if we propagate this noisy elevation data through to the roadway elevations, the resulting roads become very lumpy and unrealistic, not to mention un-drivable in a real-time

simulation. Some smoothing algorithm needs to be employed to compensate for the noise in the original terrain data.

Many state DOTs are presently involved in statewide cooperative efforts to collect and store relatively high accuracy (0.3~0.6m) Digital Elevation Models (DEM). This DEM effort is undertaken using airborne LIDAR systems. (This information was provided by Dave M. Gorg, PE, who reviewed the first draft of the report).

Large Scale vs. Small Scale Data Sets

The IV Lab map databases contain a representation of specific aspects of small areas of roadway. The key features (in terms of visual rendering) that the IV Lab mapping database represents are lane centers and lane markings. To create a compelling 3D world, the surrounding areas need to be filled in with correct terrain data and a reasonable approximation of land use and land cover types.

Large scale land use, land cover data, water cover, and road data is provided by the NIMA VMAP0 shape database which has worldwide coverage [[6]]. The SRTM terrain database provides terrain elevation information and has good coverage from -60 to +60 degrees latitude [[7]]. Other datasets which can be leveraged include the FAA obstacle database to pull in precise locations of objects such as water towers, radio towers, smoke stacks, and down town buildings. These objects are important to a driving simulator because they can typically be seen at a distance and often serve as important landmarks to navigation. The VMAP0 database provides low resolution roadways, railroad routes, city areas, lake and river outlines, stream paths, and other land use and land cover data. These data sources can be used to create a lower resolution context in which to place the high resolution IV Lab data. The result is a seamless large scale world that is as accurate as the input data allows.

The Data Transformation Pipeline

A tool chain has been created which processes the diverse input set of data sources. The dataflow for this pipeline is shown in Figure 8**.** This process transforms each data set into a canonical form, and then merges all the data and assembles it into a set of unified 3D models, each covering one tile. The tool chain is designed to handle large scale databases and can batch process, clip and sort them into the native tiling scheme. In addition, the toolset includes a distributed processing component. The batch processing speed for large areas can be substantially improved by distributing the individual tasks over a distributed network of workstations.

Each of the processes shown in Figure 8 is described below.

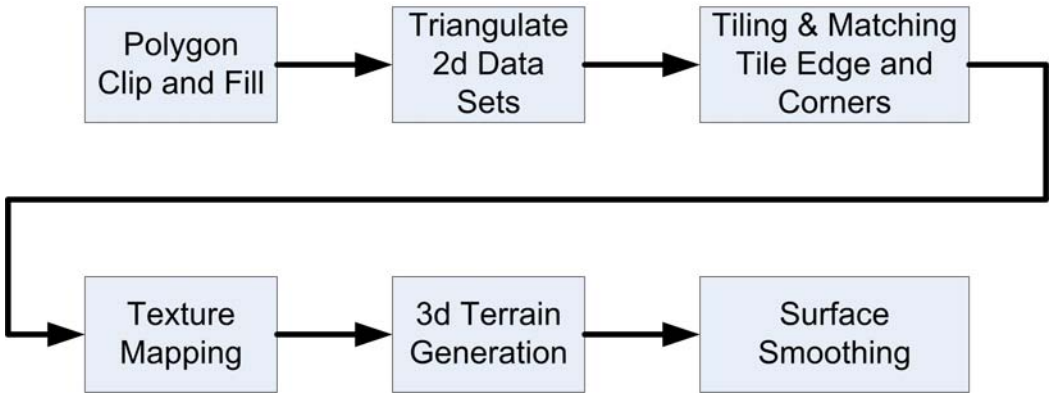

Figure 8. Data flow for data transformation pipeline.

Polygon Clipping and Filling

All input data (with the exception of 3D terrain data) is converted to 2D polygons in the first step of the pipeline. It is very important for the final result that all the input data be merged together in a way such that the final result is a continuous single layer surface covering the entire tile. There can be no overlapping regions and no cracks or gaps. All the areas must fit together perfectly like a puzzle.

To accomplish this all the input data is ranked hierarchically in the order of importance. For instance a city area would take precedence over a generic land mass. A lake or stream would take precedence over a city area. A road surface would take precedence over everything else.

An algorithm was developed by Murta to process all the 2D input regions, starting with the most important area first [[8]]. This highest precedent area is placed into the "puzzle" and that region is claimed and cannot be used for any subsequent (lower ranking) area. The next most important area is then clipped against the already reserved areas using the General Polygon Clipping (GPC) polygon clipping algorithm.

These resulting areas (minus the more important areas) are added to the “puzzle.” This process continues with the lesser and lesser ranked regions until all the available regions are processed. This approach guarantees that there will be no overlapping regions and allows the user to define the priority scheme so that the more important features are not masked by less important features.

The last region to be added is a generic land mass polygon. Finally, any unassigned regions are marked as ocean. This algorithm results in a set of “puzzle” pieces that fit together perfectly to completely cover the tile with no overlaps and no gaps. An example of a clipped and filled geographic area is shown in Figure 9.

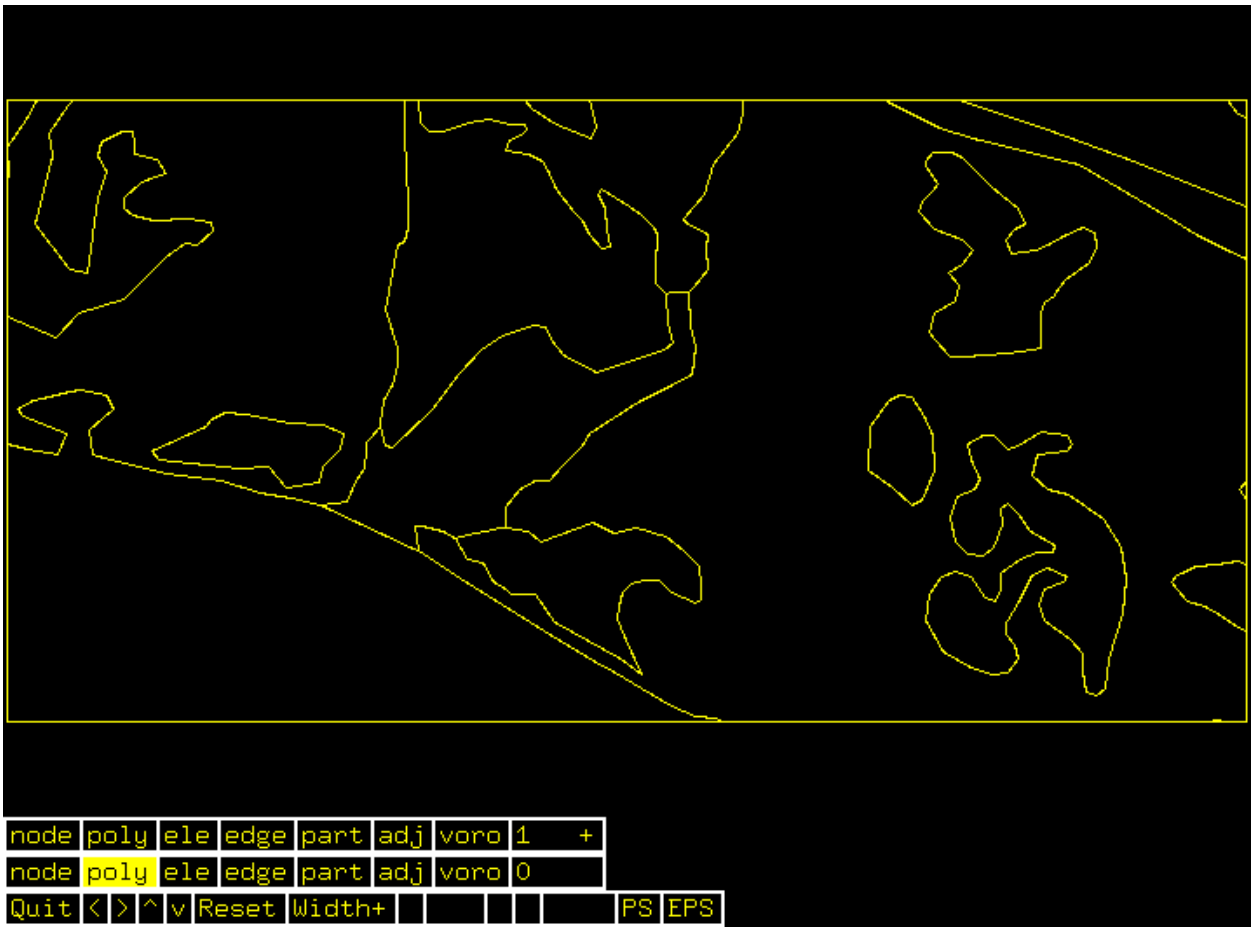

Figure 9. Example of a completed "puzzle" showing final clipping results.

Handling Unexpected Situations of Artifacts in the Original or Intermediate Data

No data set is perfect and can data sets can be subject to human error or errors introduced algorithmically at some point in the processing pipeline. In addition, as data is processed in the tool chain, additional artifacts can be introduced due to a variety of reasons. It is not possible to detail every possible artifact and inconsistency in every data set, but the scope of the problem can be illustrated through some examples.

The GPC polygon clipping library implements a robust polygon clipping algorithm. It accepts any self consistent polygon shape including convex and concave regions and even polygons with multiple interior holes defined as separate contours. The output of the algorithm is robust for 2D rendering. However we need to propagate the result down the tool chain and eventually create a seamless 3D surface skin and generate a surface mesh built out of triangles. Upon closely examining the output of the GPC library it becomes apparent that a variety of real world input data can trigger numerical problems (due to the limited precision of the IEEE floating point representation) that result in a variety of artifacts. For instance, the output may contain a zero area peninsula, or points that are doubled up, or line segments that touch or cross other segments in non-self consistent ways. These sorts of numerical problems (although not visually discernible) can cause the polygon triangulation algorithm and other steps downstream in the

pipeline to fail. There are many sections of code that scan the data at a variety of stages in the pipeline in order to detect and resolve these types of data issues.

Triangulating the 2D Data Sets

After the 2D data has been collected and merged into a seamless, self consistent, non-overlapping set of regions, it must be converted to triangles. The real time rendering system (OpenGL) renders textured triangles as its basic primitive, and it is up to the tool chain to create the set of triangles from the 2d data set.

The Delaunay Triangulation algorithm produces an optimal set of triangles that maximizes the minimum interior angle of all the resulting triangles. The ultimate result of the Delaunay Triangulation algorithm is a set of equilateral triangles. The Delaunay Triangulation algorithm avoids long skinny "sliver" triangles in favor of triangles where all three sides are similar in length. Long skinny triangles can produce visual artifacts in the final real-time result, so Delaunay triangulation produces the optimal triangulation for 3D rendering.

Technically, the Delaunay Triangulation algorithm produces a set of triangles such that if a circle is drawn through the 3 vertices of any triangle, that circle will not enclose any vertices from any other triangles.

Complicating the process is the fact that each region or surface type needs to be triangulated separately, and each region may contain an arbitrary set of boundaries and nested interior holes. (To understand why nested holes are important in the polygon regions, imagine a continent, with a lake inside, then an island inside the lake, and a pond inside the island.) Each of these regions must be triangulated separately with consideration of the outside boundaries and all the nested interior holes.

In order to triangulate around polygon holes or within polygon regions, the triangulator does a full triangulation of the entire point set. The triangulator is also given the entire set of polygons (boundaries and holes). These form "boundary segments." In addition, the triangulator is given a point inside each hole and a point outside the enclosing region. For each of these "hole points," the algorithm finds the triangle that encloses that point and discards it. It then recursively discards all adjoining triangles that aren't blocked by one of the boundary segments until no unblocked adjacent triangles remain.

Figure 10 shows the results of the Delaunay Triangulation algorithm applied to a 2D data set.

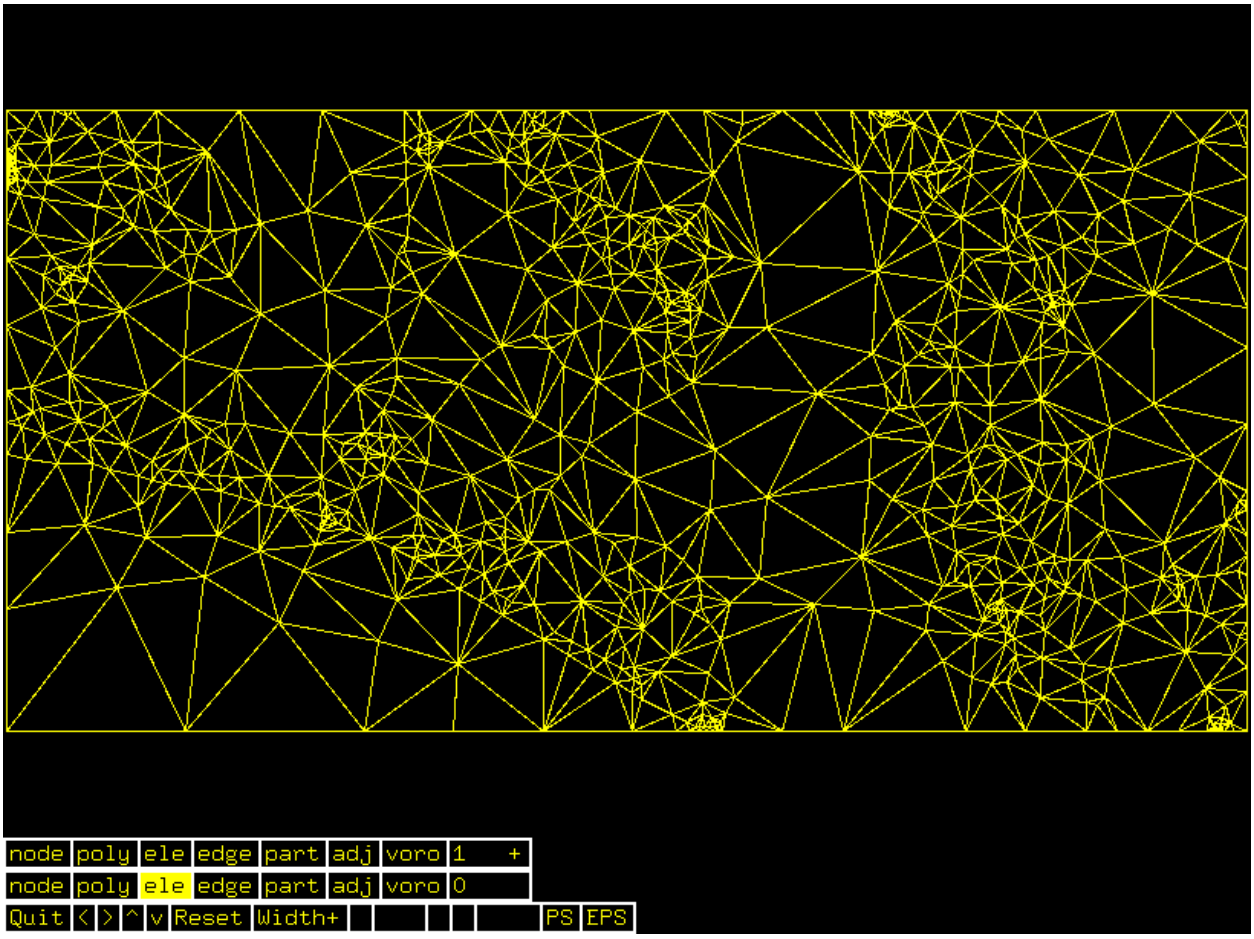

Figure 10. Delaunay triangulation of a tile.

Tiling and Matching Tile Edges and Corners

In order to represent large scale models such as the entire state of Minnesota or the entire world, the scenery database must be divided into smaller manageable, previously described sections called tiles. Also mentioned earlier is the importance of not having any gaps or cracks in the resulting real-time database. Gaps and cracks lead to distracting visual artifacts and must be avoided.

As a result, it is important that the edges of tiles match exactly. This (like most of the other stages in the data processing pipeline) turns out to be more difficult than it first appears.

First, it is important to distinguish between edge points, corner points, and interior points of a tile. A corner point will be shared by 4 adjacent tiles. All the points along an edge will be shared by 2 adjacent tiles. The interior points are all the remaining non-edge and non-corner points and are completely unique to a specific tile.

Second, it is important to note that the edge sharing algorithm declares that when a tile is created, it defines and owns all the edge and corner points that haven't been previously defined by another tile.

When a tile is created, the points are divided up into the 4 corner points, the 4 sets of edge points, and all of the remaining interior points. The corner points and edge sets are individually written to a shared area, but only those elements that haven't been previously defined by another tile.

The algorithm then reads in all the shared edge and corner points from the shared data area, recombines them with the interior points, re-triangulates the entire point set (in case that an adjoining set of edge points was loaded for any side) and that results in the final 2D geometric layout of the tile.

Texture Mapping

In order to produce natural and realistic objects in a real-time simulation, most object surfaces need to be texture mapped. A variety of natural looking textures (derived from USGS satellite imagery) that represent a large variety of land use and land cover areas such as Urban areas, lake areas, various types of crops, various types of forest, desert, tundra, ice, etc, have been assembled. These are mapped to the original VMAP0 area types in the final real-time database.

The textures are designed so that they repeat smoothly in both “x” and “y” directions. This way a single texture can cover a polygon area of arbitrary size while maintaining the proper scaling of the imagery upon which the texture is based. An algorithm determines the individual texture coordinates for each polygon in the resulting model. The algorithm considers the latitude of the triangle so that no "map maker" distortions are introduced. In addition, each texture type can have a scaling factor defined so if a new replacement texture is created with a different scale, it can be assigned to the model correctly at run time without needing to regenerate the entire scenery database.

3D Terrain Data

So far, all the stages of the pipeline have dealt with data from a 2D or flat map perspective. However, for real-time simulation a perfectly flat world creates an unnatural looking environment. A set of tools has been developed that imports raw SRTM data, chops it to the defined tile layout, and fits a simplified surface to the original data. These elevation points are introduced into the 2D "point soup" before the triangulation process so that the resulting triangles account for enough data points to accurately represent the underlying terrain. Once the tile has been created from a 2D perspective, the elevation for each of the 2D points is interpolated from the original 3D terrain grid. Thus the resulting tile is composed of all the 2D regions and features from the original GIS data, but is elevated to form a "terrain skin" that accurately represents the terrain of the region. Mountains, valleys, hills and other terrain features that exist in real life are all clearly visible in the real time simulation. 3D data and the underlying triangles are shown in Figure 11 below.

Previously described DEM data can be used in place of the SRTM data in applications where greater accuracy is required for a higher-fidelity simulation. In rural simulations where roadways pass through farm fields and forests, SRTM data provide sufficient accuracy, density, and resolution to provide an accurate simulator experience. However, suburban and urban areas generally require greater accuracy, density, and resolution; in this situation, DEM data can provide the greater accuracy, density, and resolution needed to provide higher simulator fidelity.

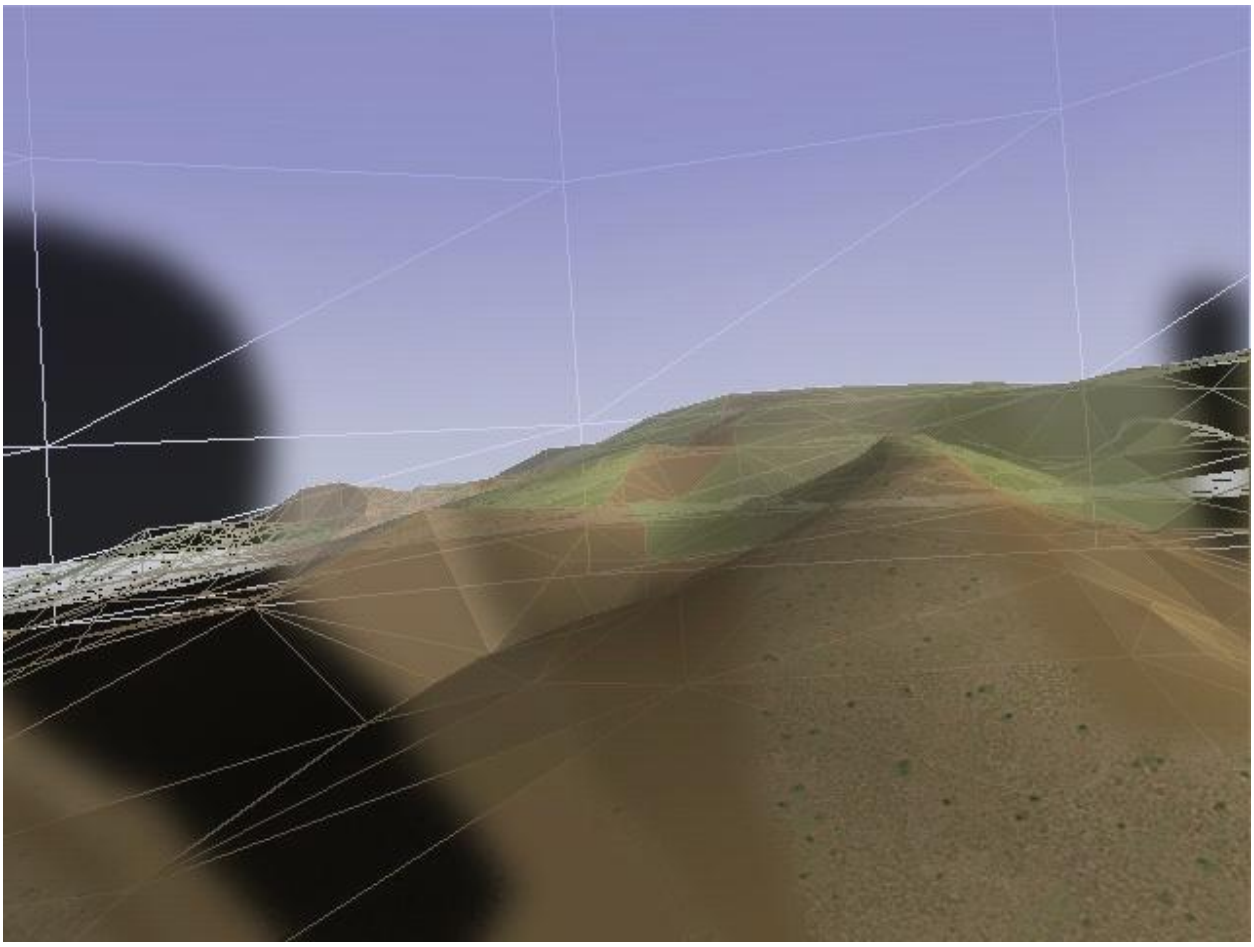

Figure 11. Terrain construction showing underlying triangle structure and texture mapping.

Surface Smoothing

As mentioned earlier, the raw terrain surface data has a random noise error of about 5-10 meters from data point to data point. However, when generating larger scale "contextual areas" the typical point spread is far enough apart that these errors are not visible and thus are not a concern.

However, for road surfaces where points are packed closely together to form complex curves and lane shapes, the noise in the terrain data does become a factor and leads to a very uneven and almost un-drivable road surface where the elevation from one point along the roadway to the next could change by as much as 5-10 meters. This is clearly unacceptable for a visual simulation, however the actual road elevation data is not part of the IV Lab mapping database.

IV Lab data has been traditionally collected as 2D because the Head Up Display is based on a flat world model. Drivers in low visibility are primarily concerned with their location on the road, and not the landscape adjacent to the road. 3D HUD data is more difficult to process by a

driver operating in low to zero visibility, but 2D provides all of the information needed for that driver to perform the tasks at hand. In this case, less is more.

In the future, IV Lab road database data will be collected in 3D to better facilitate the creation of virtual environments. Elevation data will be removed on the databases installed on the vehicle so the driver can utilized the familiar and comfortable 2D HUD images.

This is yet another side effect of the process of adapting GIS data designed with one use in mind to an entirely new use. This particular problem presents a significant challenge that is not fully solved and could be an area of future project work.

Within the scope of this project a simple smoothing algorithm that traverses all the road and road marking triangles and moves their associated vertex points upwards to limit the slope of each triangle was implemented. This is an iterative scheme that is run until the slope of all the road surface related polygons are constrained to some user defined limits. This scheme works fairly well, but is inefficient for tiles that contain a large amount of detailed road surfaces. The downside to this scheme is that it is still subject to propagating some of the minute artifacts in the terrain data and the result is not wholly natural or satisfying.

As described above, one future solution would be to include the feature elevations in the IV Lab mapping process and then propagate this data through the tool chain and not adjust road elevations based on the less accurate SRTM data. This is an attractive option and most likely would yield the most accurate and natural results, but would require a significant amount of rework of the existing tool chain.

A second solution might be to employ some surface curve fitting algorithm over the regions occupied by the road surfaces. This could be useful for areas

- where the true elevation data has not been mapped
- where the elevation data is noisy or suspect
- that have been mapped with less precise equipment (such as a non-differential hand held GPS).

A third option would be to use DEM data which provides higher accuracy, density, and resolution as a starting point. The appropriate DEM data can be decimated and smoothed to produce high fidelity simulator surface with comparatively less effort. Less effort is required in this stage because of the greater effort put forth to collect the accurate LIDAR data on which the DEM is based.

Experimentation with a surface curve fitting scheme for small areas has yielded very good results. Surface fitting can generate a surface that generally follows the average shape of the terrain, blends in well with the surrounding terrain, and has a natural, smooth visual appearance. However, the current surface fitting algorithm would need to be significantly reworked to handle the types of areas that are included in the IV Lab mapping database.

The tool chain does provide an immediate solution that works reasonably well, but leaves room for future improvements.

Conversion to Multigen-Paradigm Open-Flight 3D Format

A process to create a textured, seamless 3D scenery database that accurately represents an area of the world has been implemented and described. However, there is a huge variety of 3D model formats and each simulation or visualization tool seems to pick a different native format.

The original work plan stated this tool set would be able to generate databases usable in the HumanFIRST driving simulator facility. The HumanFIRST simulator requires models to be saved in the Multigen-Paradigm Open-Flight (flt) 3D format.

In order to do this, a conversion and export process needed to be created. Unfortunately the available tools and libraries used to produce this tool chain do not include an open-flight exporter. The solution is to write out the scenery database in LightWave .obj format. This format can be imported into Multigen Creator (which is a 3D modeling tool we use to do manual and semi-automated scenery creation.) Creator's native file format is flt, so once imported into Creator, the scenery can be saved as flt and then incorporated into the simulation.

Chapter 6: Conclusion

This project set out to improve IV Lab road databases and to show how these databases could be used in new applications. The IV Lab database data model was redesigned to better match the physical layout of roads and to conform to open GIS standards. A new data model was created that fulfills the original requirements and the need for better interoptability with other spatial applications and new onboard vehicle subsystems. This new designed has been used successfully for onboard vehicle subsystem and to generate scenery data for the HumanFIRST driving simulator.

The process of merging and transforming 2D and 3D GIS datasets into a 3D visual scenery database appropriate for real-time simulation and visualization is highly difficult and complex. A process to generate 3d simulator databases were successfully created and demonstrated. This processes combined detailed IV Lab map data of actual Minnesota roadways with larger scale, but less accurate GIS terrain and land use data to create an immersive and natural simulator world. This project demonstrates that the original project goals are feasible and creates a set of tools to accomplish those goals. In the process of creating the tool set pipeline and pushing real IV Lab data through it, areas where the IV Lab data and mapping process could be improved were identified and areas where the tool set and data conversion and data merging algorithms could also be improved have been determined. These results can be factored back into future projects so that the pipeline and the final result can continue to be refined and improved.

References

[1] B. Newstrom. *Real Time High Accuracy Geo-Spatial Database for Onboard Intelligent Vehicle Applications*. Master's thesis, Mechanical Engineering Department, University of Minnesota, Minneapolis, 2000.

[2] PostGIS spatial database extension for PostgreSQL (Internet), cited April 2008, http://www.postgis.org/.

[3] PostgreSQL open source database (Internet), cited April 2008, http://www.postgresql.org.

[4] Open Geospatial Consortium, Inc (Internet), OpenGIS Implementation Specification for Geographic information - Simple feature access - Part 2: SQL option, (cited April 2008), http://www.opengeospatial.org/standards/sfs.

[5] Geospatial Data Abstraction Library (Internet), cited April 2008, http://www.gdal.org.

[6] National Geospatial Intelligence Agency (Internet), Vector Map Level 0 (VMAP0), September 1998, cited April 2008, http://earth-info.nga.mil/publications/vmap0.html.

[7] Shuttle Radar Topography Mission (Internet), February 2006, cited April 2008, http://www2.jpl.nasa.gov/srtm/.

[8] A. Murta, *General Polygon Clipper Library* (Internet), January 2008, cited April 2008, http://www.cs.man.ac.uk/~toby/alan/software/.