GIS Enabled Modeling and Simulation (GEMS)

Thomas Stanzione
Kevin Johnson
MAK Technologies
68 Moulton Street
Cambridge, MA 02138
(617) 876-8085 x109, x132
tstanzione@mak.com, kevinj@mak.com

ABSTRACT: *Current C4ISR and simulation systems use different tools and formats for generating and storing geospatial information. C4ISR systems tend to use geographic information systems (GIS), such as C/JMTK, for this information, while simulation systems use proprietary terrain database formats that are generated from a number of different terrain database generation tools. This leads to problems sharing geospatial information between systems, making mission planning or embedded training difficult, as well as problems maintaining geospatial information as it is updated. A common geospatial database that can be generated with a single set of tools and shared across applications would eliminate these problems and allow higher integration of diverse military systems. Under a contract with the US Army Topographic Engineering Center, MÄK Technologies, along with ESRI, is developing a prototype framework for accessing geospatial data from federated geospatial databases directly into M&S applications, utilizing the ESRI ArcGIS family of products. This paper will discuss our work to date and future plans.*

1. Introduction

Currently, modeling and simulation (M&S) applications require specialized terrain data formats that are significantly different than those used for operational purposes. Considerable time and effort is expended in converting, or "cooking," source data to generate M&S databases. As shown in Figure 1, simulation tools typically require specialized formats optimized for visualization and simulation – such as CTDB, OpenFlight, OneSAF Terrain Format, etc. Generating these databases requires an extensive process of identifying source data, reconciling coordinate systems and resolutions, converting formats, cleaning up disparities between data sets, adding additional feature (e.g., textures), etc.

There are a number of problems that arise from having specialized M&S terrain formats that are different from operational systems:

- The process of generating M&S databases is largely manual, time consuming, and expensive – and much of the work is simply discarded upon completion (e.g., corrections to data discrepancies are not fed back into source datasets).
- The time delays involved limit the currency of data, and make it difficult or impossible to utilize simulation tools for time-critical applications such as mission planning, mission rehearsal, and predictive situational awareness.
- As simulation systems evolve, terrain formats may need to change, making it hard to maintain and convert older databases to the new formats.
- There is no interoperability of geospatial data between simulation systems and other military command and control or planning systems, forcing development of multiple geospatial systems to support all of the applications.
- Different geospatial datasets for multiple applications have to be correlated to ensure each system is using the appropriate data.
- Once built, multiple geospatial datasets are harder to maintain since each copy has to be maintained separately. This is especially a problem for simulation systems that may modify the terrain during execution (i.e. dynamic terrain), since there is no easy mechanism for getting these changes into the other geospatial datasets.

A common geospatial database that can be used by different military systems can be used to alleviate the above problems. The many resultant benefits include:

- **Reuse and Leverage Geospatial Content** – Each year a great deal of effort is put into designing and building geospatial data for military applications. For simulation systems, this effort is typically restricted to use within a specific simulation system. Essentially, the content is "locked" to the specific simulation. If the geospatial data could be

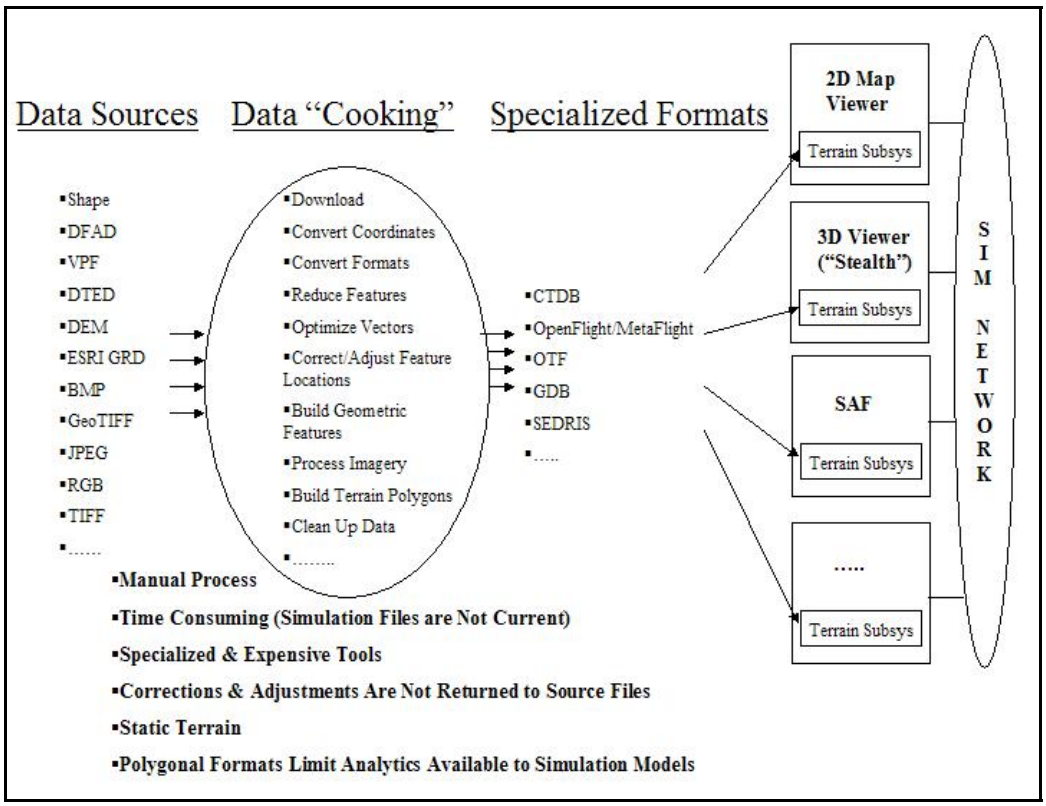

Figure 1: Terrain Generation for M&S – Current Practice

provided in a common format, other applications could reuse this valuable intellectual property.

- **Increased Interoperability** – Since all applications would be working from the same geospatial dataset, interoperability is greatly improved. Mission planning could be performed on operational data that is also being used by simulations performing course of action analysis, leading to more meaningful results than is available today, due to the differences between operational and simulation geospatial data.
- **Timely support for mission rehearsal** – Mission rehearsal requires access to the most recent tactical data. The ability to operate from current operational databases will enable mission rehearsal applications.
- **Decreased Creation Time** – By developing tools that are based on existing geospatial data systems, such as the Commercial Joint Mapping Toolkit (C/JMTK), the cost and time to build geospatial datasets for multiple applications is reduced. Having to learn a single geospatial data development system instead of multiple systems for the different applications would be much easier. Geospatial datasets could also be built incrementally, without having to stop and "cook" all the different runtime formats, improving datasets over time while still having useful data for specific operations.
- **Increased Correlation** – Since everyone would be essentially using the same geospatial data, correlation is practically guaranteed. The need to correlate multiple datasets would be eliminated. Also, management of the geospatial data would be much easier since it would be taking place in one geospatial system instead of many different tools.
- **Dynamic Capabilities** – A common geospatial data capability that could be accessed via a network would provide the ability for applications to make changes to the geospatial data and have those changes distributed during runtime. This would allow seamless dynamic terrain capabilities, either through effects of the simulation (bomb craters, rubble, etc.) or updates by operators (intelligence reports, instructor generated terrain changes, etc.).

In the past, limitations in computer capabilities (notably processing speeds) required specialized run-time formats in order to achieve real-time performance for simulation and visualization. Advances in technology and practice now make it feasible to eliminate much, if not all, of this conversion process. In particular, an increasing amount of source data is being managed online, via enterprise and federated Geographic Information Systems (GISs) – which typically provide considerable functionality for automatic format conversion (e.g., normalizing coordinate systems and resolutions across data layers).

Much of the requirement for specialized M&S formats derives from the need to visualize scenes at 30+ frames per second. Faster processor speeds, and specialized graphics cards, now make it feasible to simulate and visualize directly from geospatial datasets – without requirements for pre-conversion of data formats. For example, today's GIS tools – such as ESRI's ArcGlobe – provide the capability to "fly through" networked terrain at speeds comparable to those achieved by traditional 3D visualization systems.

MÄK Technologies, along with ESRI, is developing a GIS enabled terrain subsystem based on the common geospatial database approach, using the ESRI components of C/JMTK. This work is being done for the US Army Topographic Engineering Center at Ft. Belvoir. We have been tasked to investigate how far we can go with entity-based simulations running directly on GIS data, in terms of fidelity and realism. MÄK Technologies believes that technology advances will allow us to re-engineer the M&S terrain generation process – essentially linking simulation and visualization systems directly to the GIS environment, to support medium to high fidelity modeling and behaviors.

Figure 2 illustrates a notional system concept for this new common geospatial database approach. As shown in Figure 2, simulation and visualization tools can be connected directly to the federated geospatial databases being developed to support operational users – providing direct access to current data, eliminating much of the manual processing currently required, and allowing use of the same GIS tools to manage and edit M&S terrain that are used to manage source datasets. In addition, using GIS data formats and systems sets the stage for significant enhancements to M&S practice, in particular, allowing simulations to modify terrain and disseminate the changes ("dynamic terrain") and allowing simulation models to access GIS-based analytics such as those being developed as part of TEC's Battlespace Terrain Reasoning and

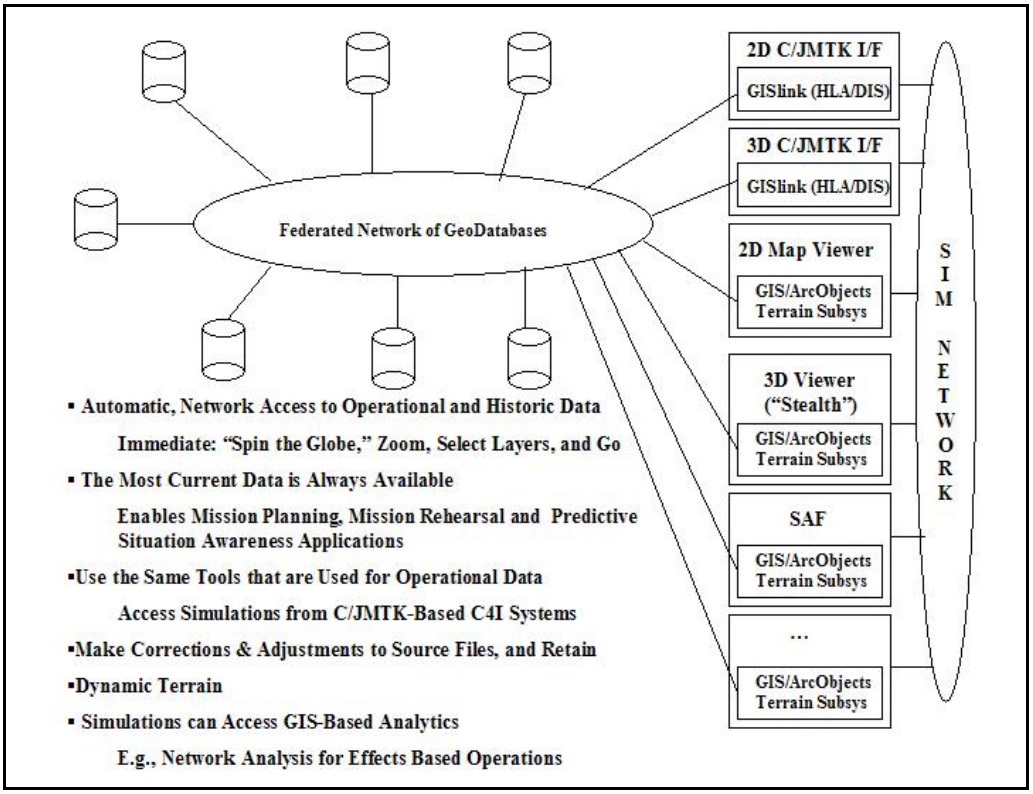

Figure 2: M&S Terrain Access in a Federated GIS Environment.

Awareness (BTRA) program. Further, this approach will enable simulation results to be viewed not only on specialized viewers, but on GIS-based displays (such as C/JMTK based C4I systems). We believe this concept can also be extended to other application domains like mission planning and rehearsal, battle command, and embedded training. This paper discusses our current progress in the design and prototyping of this GIS-based terrain subsystem.

2. GIS and C4ISR Systems

In the US military, the Commercial Joint Mapping Toolkit (C/JMTK) provides the Mapping, Charting, Geodesy, and Imagery (MCG&I) functionality for C4ISR mission applications. The toolkit is being deployed to support both legacy mission applications and new systems being developed throughout the Department of Defense (DoD) Command, Control, and Intelligence (C2I) Community. C/JMTK is a set of software components for the management, analysis, and visualization of map and map-related information. C/JMTK is based on a single scalable open architecture, with open development environments, incorporating industry standards. The primary commercial component of the C/JMTK is the ESRI GIS software called ArcGIS. The C/JMTK can be viewed as the adoption of the ArcGIS platform (specifically, ArcGIS Engine) as the standard geospatial exploitation tool for DoD Command, Control, and Intelligence (C2I) systems.

C/JMTK includes the ArcGIS Military Analyst extension, which maximizes the use of the standard suite of the NGA data products by allowing direct use and rendering of NGA's vector and raster products, line of sight (LOS) assessments, Military Grid Reference System (MGRS) conversion, and DTED analysis. ArcGIS Military Analyst also includes the Military Overlay Editor (MOLE), which supports MIL-STD 2525B and custom war fighting symbologies.

Because of the close ties between C/JMTK and ArcGIS, we selected ArcGIS as the GIS engine for this project. The terrain subsystem prototype that we are developing uses ArcObjects, which are the basic building blocks of ESRI's ArcGIS software, to implement the application programmer's interface (API) between the M&S models and the GIS terrain.

3. Terrain Database Representations in M&S Systems

Typical terrain databases for modeling and simulation come in two varieties – those for 3D visualization and those for computer generated forces (CGF) applications. The 3D visualization databases need to "look good", especially in relation to the real world. These databases consist of a terrain skin represented with polygons that are generated from a digital elevation model (DEM). These polygons can be based on a regular grid or based on a Triangulated Irregular Network (TIN). TINs allow databases to be load balanced, utilizing polygon budgets where they are most needed in areas of highly varying terrain. Integrated TINs take this one step further and integrate feature data into the tinning process, such as cutting roads and rivers into the terrain skin. Figure 3 is an example of the integrated TIN process.

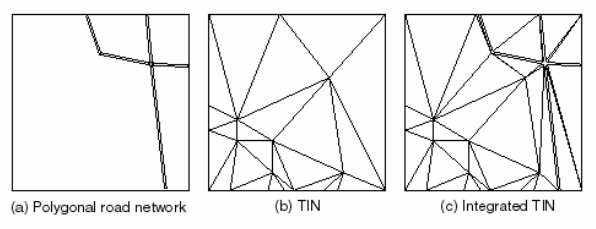

Figure 3: Road, TIN, and Integrated TIN

Along with the terrain polygons, visual databases include texture information to provide a visualization of ground and material types. These databases include 3D models for buildings, trees, and other cultural and natural point features, as well as 2D linear and area features with specific textures for roads, rivers, lakes, etc. These databases may also have aerial imagery draped over the terrain skin for a realistic visual representation.

Terrain database for CGF systems are quite different from these visual databases. While CGF systems may use their terrain information for 2D visualization, the main use is for terrain reasoning. CGF terrain contains the geometry and attribution of elevation, cultural and natural features, used for vehicle placement, movement algorithms, and line of sight. Movement algorithms include path planning, obstacle avoidance, and vehicle dynamics models, while line of sight algorithms are used for targeting and communications. CGF terrain databases have a terrain skin, similar to the 3D visualization databases, but include more attribution data instead of textures. This attribution data allows the computer models to reason about the terrain explicitly, without having to infer information. Terrain skin attribution may include soil type (including water), mobility characteristics, and vegetation characteristics.

In addition to the terrain skin, CGF terrain databases also include point, line, and area features, which are also attributed for computer reasoning. These attributes include feature type, geometric characteristics like width and height, and more semantic information like road network topology. These databases may also include 3D models associated with point features,

which run the gamut from high fidelity building models with interior structure to low fidelity "overturned shoe boxes".

Other key aspects of CGF terrain databases are compactness and spatial organization. In order to provide optimal performance, CGF terrain database are kept as compact as possible so that they can be stored in computer memory in whole, eliminating the need for costly disk access. They also include a spatial organization so that all of the terrain data around a location can be found quickly. Spatial organization schemes can be grid-based or hierarchical, like quadtrees or octtrees. [1]

4. Commonality of C4ISR and M&S Terrain

One of the first tasks we undertook for this project was to see how closely C4ISR operational geospatial data matched modeling and simulation terrain data. GIS products and data are used in the M&S terrain database generation field mostly for source data preparation, so there is some overlap of the two domains. Terrain database development for M&S utilizes geospatial products from organizations like the National Geospatial-Intelligence Agency (NGA). These products include elevation data in the form of Digital Terrain Elevation Data (DTED) and vector feature data in the form of Vector Product Format (VPF), as well as other formats. In the C4ISR systems, these datasets are used directly, whereas in the M&S terrain databases, these datasets are used as the source data and "cooked" to create the specialized M&S formats.

M&S terrains differ from C4ISR GIS data due to the nature of how the data is used. Simulation basically performs intersection queries across the data to determine, typically in 3D, if some piece of the terrain intersects a line segment. GIS data is often times in 2D and laid out more for displaying map views. To have the two datasets intersect we need to be able to perform 3D intersections across the data and return attributions from the data in a similar fashion.

For example, the MÄK terrain database (GDB) is a collection of polygons that have associated with them attribution such as soil type. The API is centered around performing intersection queries through the database and returning the point of intersection or intersecting polygon, and attribution. The emphasis is on performance of this function which is utilized throughout the API. In addition to the polygons there are some vector features, such as road networks, that can be queried to speed up operations such as path planning and collision avoidance, however the operation could utilize the attribution of the polygons to determine roads directly.

One example of an operational GIS data set is the Theater Geospatial Dataset (TGD). A TGD is a collection of feature classes full of point, line, and area features. Often these features will be in 2D containing only the XY position suitable for map display. Attribution can be used to extrapolate 3D information, for example a building may include an attribute that determines the buildings overall height. The features are also quite detailed. If one was to convert from a TGD to a MÄK GDB, the MAK soil type attribute could be determined by first querying the soils layer to find the underlying soil condition, then querying the hydrology layers to find the water content, and finally querying the vegetation layers to determine overall classification. In the MÄK GDB there are 15 soil types and in a TDG there are many thousands of combinations of the attributes. The MAK GDB again has made a tradeoff of detail for overall speed. For example there is no real difference between driving on cropland and driving on grassland in a tank. One possible solution to this large combination of attributes would be to merge the GIS layers together into a master soil layer and offline run a soil type function to determine the simulation soil types, storing them as additional attributes.

The GIS data should be utilized in a manner that gives the greatest coverage of features without impacting performance in a major way. Take trees as an example. In high detailed GIS databases the trees can be represented as point features that include their height, trunk width, location, type of tree, current state of leaves, and many more attributes. From a simulation standpoint performing line of sight across this feature would entail determining the height at the location of the feature, extruding the trunk width up as a cone and possibly placing some representation at the top for the canopy. This new representation would than be used to perform a new intersection test to determine intersection. Potentially though this will have a rather high performance impact since at least a once a second query is needed to determine the elevation at the point of the feature. Some performance may be gained back by placing that information into the feature itself. Trees of course are not the best examples since there are other ways of dealing with them, but other features follow this example such as point representation of buildings and area representation of buildings (footprints).

The use of GIS-based terrain data does not preclude the use of high fidelity M&S terrain databases. These databases can be converted to GIS formats, especially if they are available in standard transmittal formats,

such as SEDRIS, or de facto standard formats like OpenFlight. ArcGIS allows OpenFlight data to be imported using the Data Interoperability extension. M&S terrain database generation systems provide some beneficial features, like automated content generation, that may not be available in GIS systems. By being able to import these M&S terrain databases, we can take advantage of these additional tools.

5. Requirements for C4ISR Terrain for M&S

The next task we undertook was to determine the requirements for the ArcGIS ArcObjects-based terrain subsystem – a modular collection of software libraries that can replace the libraries currently used in CGF applications and viewers, as shown on the right side of Figure 4. Our goal is to package these libraries with a backwards compatible API – i.e., existing simulation models, scenarios, and software would continue to function unchanged. We will be adding software "wrappers" that map functionality to the APIs used within common M&S software. For the initial prototype, we are using MÄK's VR-Forces CGF product, with the OneSAF Objective System as a likely target for future work.

As part of the requirements analysis, we performed case studies on the terrain APIs of three CGF systems – MÄK's VR-Forces, NPS' Delta3D, and the Army's OneSAF Test Bed. Delta3D is an open source simulation/game engine for military simulation. It employs mostly opensource APIs such as Open Scene Graph and OpenAL. The Forward Observer PC Simulator (FOPCSIM) was built using Delta3D. This simulation supports the task training of the artillery call for fire (CFF) for forward observers. By examining its API documentation [2] we can see how Delta3D interacts with its terrain. It employs a height field based terrain with procedural rendering aspects. Interacting with it though is done through the dtTerrain class and has only two basic functions for use by the simulation:

- GetHeight
- IsClearLineOfSight

Like most game engines, most of the effort is placed in using the terrain for rendering and not for simulation, so it did not provide much of a use case for this project.

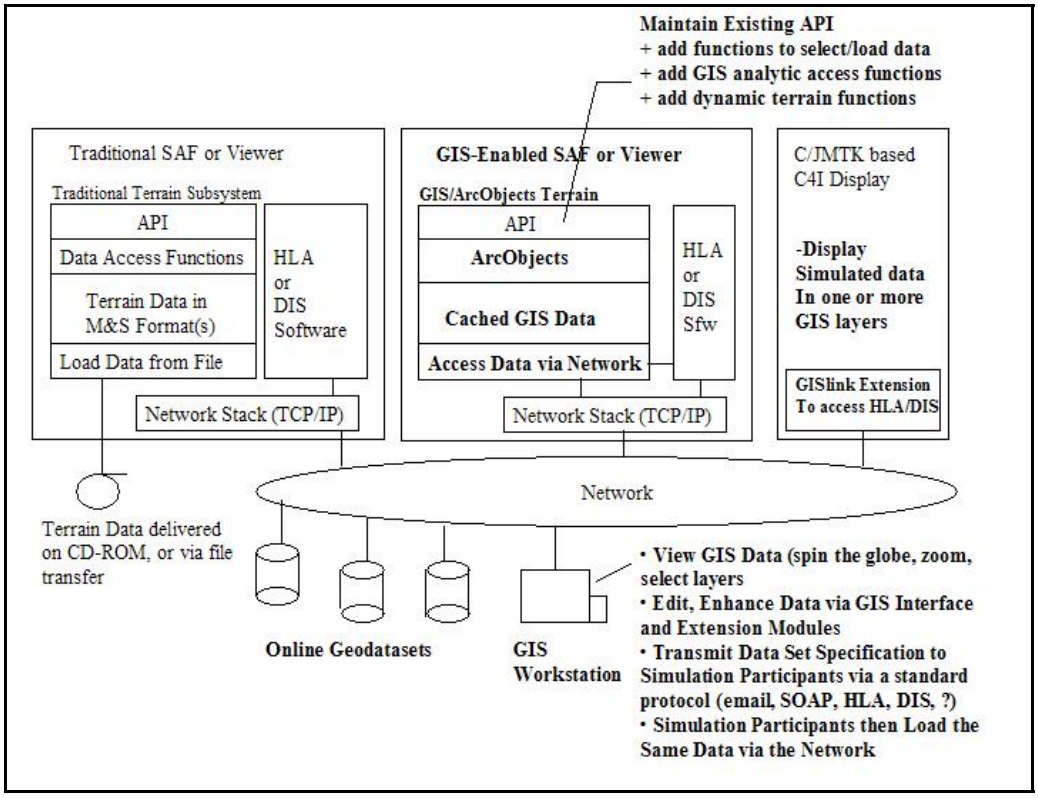

Figure 4: Terrain Subsystem System Component

VR-Forces uses the terrain for three basic functions:

- Coordinate system information
- Intersection with polygons
- Vector Network information

Starting with the coordinate system, VR-Forces looks to the database to define the coordinate system that will be used for local coordinates. Typically this is a UTM coordinate system to make it easier to perform modeling operations. The other task of the coordinate system module is to be able to convert from the local coordinates to geocentric, which is used in both Distributed Interactive Simulation (DIS) and Real-time Platform Reference (RPR) simulation networking protocols. Replacing this functionality is possible but curious, since the geodatabase does not necessarily have all the layers in the same coordinate system. We can however assume modeling in flat earth for example and just develop a single coordinate system converter.

Intersection with polygons is done through a variety of APIs which tailor the information relative to the requestor's needs. Height Above the Terrain, Height of the Terrain, and soil factors are common calls. Other common calls are a variety of intersection between two points, used for both intersection and line of sight calculations. In order to support multilevel terrain there are calls to return the closest vertical intersection to a starting point. VR-Forces does not typically ask for the slope at a given point. Instead it makes three intersection calls and determines the normal of the resulting triangle. This more accurately places a vehicle on the terrain since it takes into account the size of the vehicle in determining the vehicles orientation.

Vectors in VR-Forces are stored separately so that when an operation such as line of sight is performed it must also be separately performed on the vector network to account for vector objects such as buildings. Other vector operations include navigating through road networks. Path information through networks is not calculated in the database but is built by a module within the vehicle code that walks through the vectors and determines a path.

Direct replacement of the calls using a GIS geodatabase is possible, however it can lead to an inefficient design. We need to properly use the geodatabase's strengths when integrating to VR-Forces. For example intersection calls need not be performed on the elevation data and then separately on the vectors. Instead we should allow the geodatabase designer to identify which layers can be intersected and make a single intersect call through the geodatabase. VR-Forces confuses, somewhat, intersection with occlusion as well. While terrain does interfere with line of sight, so do other factors such as forest density, smoke, and potentially unsupported factors such as weather.

We performed some profiling in VR-Forces, which returned the following:

- A M1A2 performing road following made 350 intersection calls per second.
- A simple scenario of 3 rotary wing aircraft shooting 4 vehicles ran at about 700 intersection calls per second.
- Vector vs. Polygon intersection rates were roughly even.

The specific GDB terrain API in VR-Forces includes:

- terrainHeight – height of terrain calculated as first intersection encountered from the point "down".
- terrainHeightAndSoilType – same as terrain height but returns soil type as well.
- terrainHeightAndIntersection – same as terrain height but returns a pointer to the actual polygon itself.
- closestIntersection – same as terrain height but can intersect vector network as well. Also, it returns the next intersection down from the starting point.
- intersect – returns the first terrain intersection along a specified line segment.
- allIntersectsAlongChord – returns list of polygons that intersect the line segment.
- intersectList – like intersect but accepts a list of segments.
- allIntersectsAlongChordList – like allIntersectsAlongChord but accepts a list as input.

The GDB vector API for the most part searches the list of vectors itself. However there are terrain like calls to facilitate intersection of trees and such:

- intersectVectorNetwork – returns the first intersection of vector data against a line segment.
- intersectChordsWithVectorNetwork – same as intersectVectorNetwork but accepts a list of segments.
- allIntersectionsWithVectorNetworkAlongChordList – same as above but returns all intersections.

The OneSAF Testbed (OTB) interacts with its terrain through the libctdb library. This library is a C style library that offers a lot of functions to deal with various aspects of the terrain. The documentation for the API is

located in the libctdb user manual [3], and other data was pulled from OTB source code.

The interesting terrain API functions are as follows:

- lookup elevation – many functions to give dirt elevation, feature elevations, water depth, etc.
- lookup soil – returns soiltype at a point.
- point to point – line of sight test that returns a float from 0.0 to 1.0 to indicate visibility with levels of obscuration.
- point thru point – radar style line of sight.
- vehicle blockage – determine if LOS is blocked by a vehicle.
- point on ground – determines if an elevation is located on the terrain skin.
- max ratio – returns rise over run for a segment that will just clear the terrain and features for the given starting point.
- contour route – generates a route that follows the contour of the terrain.
- find high ground – returns the highest point along a segment.

There are also extensive feature interactions:

- nearest bridge – finds the nearest bridge.
- nearest road – finds the nearest road segment.
- get buildings – finds all the buildings in an area.
- enclosures – functions to return polygons or topology for enclosures (some buildings can be represented by micro terrain and therefore have polygons).

OTB also includes some dynamic terrain capabilities:

- Dynamic entities are represented in the terrain and can be used during line of site calls.
- Features can be added/removed/moved during runtime.
- Multi-Elevation Surfaces (buildings typically) can be altered by blowing holes in them.

From this requirements analysis, the following requirements for the terrain interface layer that sits between the CGF and the ESRI geodatabase were determined:

1. Interface shall be implemented as ESRI ArcObjects and comply with the standards set for ArcObjects.
2. Interface shall be able to interact with either networked or local geodatabases.
3. Interface shall provide methods and facilities to handle connecting to geodatabases.
4. Interface shall not preclude or hinder the use of other ArcObjects for display or other purposes.
5. Interface shall support common terrain interactions used by modeling, including but not limited to:
   - Height of terrain
   - Height above terrain
   - Soil factors/type
   - Slope
   - Intersection tests
   - Line of sight tests
   - Interaction with vector networks
6. Interface shall support Geocentric Coordinates for all APIs.

6. Design of M&S API to GIS Terrain Data

The first design task was to map C4ISR datasets to the M&S terrain requirements. As discussed previously, C4ISR datasets contained the elevation and feature data needed for M&S models, but they are organized and stored differently. In ArcGIS the data can be stored in a number of different ways. Elevation data can be stored as raster datasets, with each pixel corresponding to an elevation value. It can also be stored in a TIN dataset, which includes elevation points and the triangularization information to connect them into a network. In the newest version of ArcGIS, they have included a terrain feature class, which stores a hierarchy of TINs for use at different map scales. Another alternative is to store the elevation data as part of a polygon Z feature class, which store 2D polygons with elevation values at each vertex.

Feature data can also be stored in different ways. ESRI shapefiles, which is a vector data storage format for storing the location, shape, and attributes of geographic features, can be used to store each 2D feature type. The multipatch feature class allows the storage of 3D features using planar three-dimensional rings and triangles, used in combination to model objects that occupy discrete area or volume in three-dimensional space. Multipatches may represent geometric objects like spheres and cubes, or real-world objects like buildings and trees. Features can also be stored in individual point, polyline, or polygon feature datasets.

Some of the data types used in ArcGIS can be organized in personal or file geodatabases. Organizational schemas can be defined so that datasets from different sources or areas of the world can be organized consistently. A personal geodatabase stores data in Microsoft Access, whereas a file geodatabase is stored as a folder of files. Geodatabases can contain feature classes, attribute tables, raster datasets, network datasets, topologies, and others datasets. They allow relationships between feature datasets to be stored, such as topologies of linear features.

For our current geodatabase design, we are using a TIN for elevation data, multipatch datasets for 3D features, and individual point, polyline, and polygon feature datasets for 2D features. We are storing these datasets (except the TIN) in a file geodatabase, which we determined was faster to access than a personal geodatabase.

We are utilizing a geodatabase schema based on the TGD operational terrain. This schema includes the following feature layers:

- BarrierL Linear barrier layer
- BuildA Area buildings
- BuildL Linear buildings
- BuildP Point buildings
- CropA Crop land areas
- FortA Fortified areas
- FortP Fortified points
- ObstrP Point obstructions
- OrchardA Orchard areas
- RuinsA Ruins area
- RuinsP Ruin points
- TreesA Trees area
- TreesL Linear trees
- TreesP Point trees
- VegA Vegetation areas
- RunwayL Linear runways
- Lakes Water areas
- Rivers Linear water
- Railroad Rail lines
- RoadL Road lines

For buildings, we are using the area and point building layers (our example data does not include linear buildings). For the point buildings, we are running a geoprocessing routine that converts them from point features to building footprints, using the length and width attributes. The converted point buildings are merged with the area buildings to create a combined layer. This layer is extruded to 3D and stored as multipatch features. It turns out that multipatchs are stored as compressed data, so they are uncompressed each time we access them. We are currently working with ESRI to improve multipatch performance and on ways to intersect the 2D building footprints directly.

For soil types, we are using the linear and area features to create a single unified soil type layer. This is being done as a geoprocessing step to improve performance at runtime. As part of this process, the linear features are expanded by their width attributes. We are utilizing the geoprocessing capabilities of ArcGIS to perform these steps. The Model Builder utility allows these geoprocessing models to be created quickly using the built in tools in ArcGIS. There is also the capability of creating new tools using Python or Visual Basic scripts.

Figures 5 through 7 show the results of these geoprocessing steps. Figure 5 shows an aerial view of the terrain. Figure 6 shows the original GIS data of this area, with point buildings and linear roads. Figure 7 shows the results of the geoprocessing steps to turn point buildings into area buildings and add the width to the roads.

One other area of the design addressed the different coding schemes in use for identifying real world objects in geospatial datasets. NGA data and many legacy simulation systems use the Feature Attribute Coding Catalogue (FACC), which is a comprehensive coding scheme for features, feature attributes, and attribute values [4]. Newer simulation applications, like the OneSAF Objective System, use the Environmental Data Coding Specification (EDCS), which was developed under the SEDRIS program [5]. For this terrain subsystem, we wanted the API to be able to use either specification. We designed the API to have two parts, an ESRI specific interface, and glue code that is specific to each application. Within the glue code, we can provide a conversion from FACC to EDCS, or alternately we could store the appropriate EDCS codes along with the FACC codes in the GIS data as a result of the geoprocessing function. The glue code would also take care of how an application wants to use the terrain data. The ESRI interface code would return a list of intersected objects, and the glue code, would have the actual algorithm that uses the list for LOS, etc. For example, for VR-Forces the glue code will map the FACC to the eight GDB soil types that are actually used.

Figure 5: Arial View

Figure 6: GIS Data before Geoprocessing

Figure 7: GIS Data after Geoprocessing

7. Prototype of M&S API in C/JMTK

Figure 8 shows the terrain subsystem prototype we are currently developing. This prototype is being used to determine performance implications of various design decisions, and to foster greater understanding of the project's scope. The prototype consists of a VR-Forces simulation engine connected to a file geodatabase for feature data and a TIN layer for elevation data. The geodatabase has a soiltype feature layer and multipatch buildings. VR-Forces elevation and intersect calls have been replicated that utilize the GIS layers for their data instead of the MAK internal database.

We have developed two new classes in the VR-Forces physical world API, ESRI Terrain and ESRI Coordinate System. These subclass the existing Terrain and Coordinate System classes, which accessed the GDB terrain format. These new classes are built from ArcObjects and access the GIS data and ESRI coordinate system components directly. The ESRI Terrain class provides elevation data access to the TIN and line of sight calls using the ESRI LOS methods. It utilizes the ray intersect method of LOS through multipatch buildings, and it supports soiltype access.

The prototype also includes the MÄK GIS-Link product, which provides the underlying components to enable ArcGIS-based applications to connect to HLA and DIS simulation data and visualize it in real time. GIS-Link is provided as extensions for ArcMap and ArcGlobe, and is used in the prototype to display the simulated entities within the map view of ArcMap. We also built a VR-Forces Toolbar using ArcObjects to provide simple control of the simulation engine from the map display.

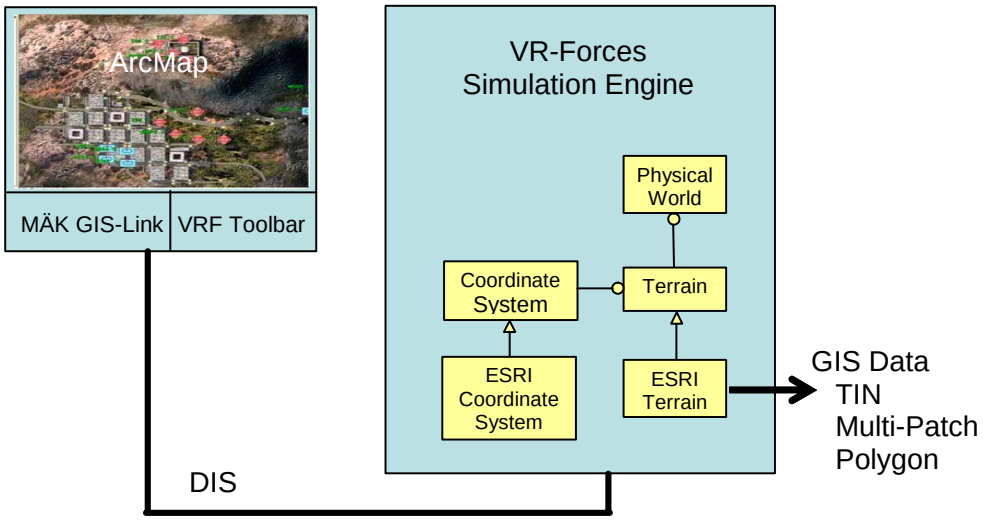

Figure 8: Terrain Subsystem Prototype

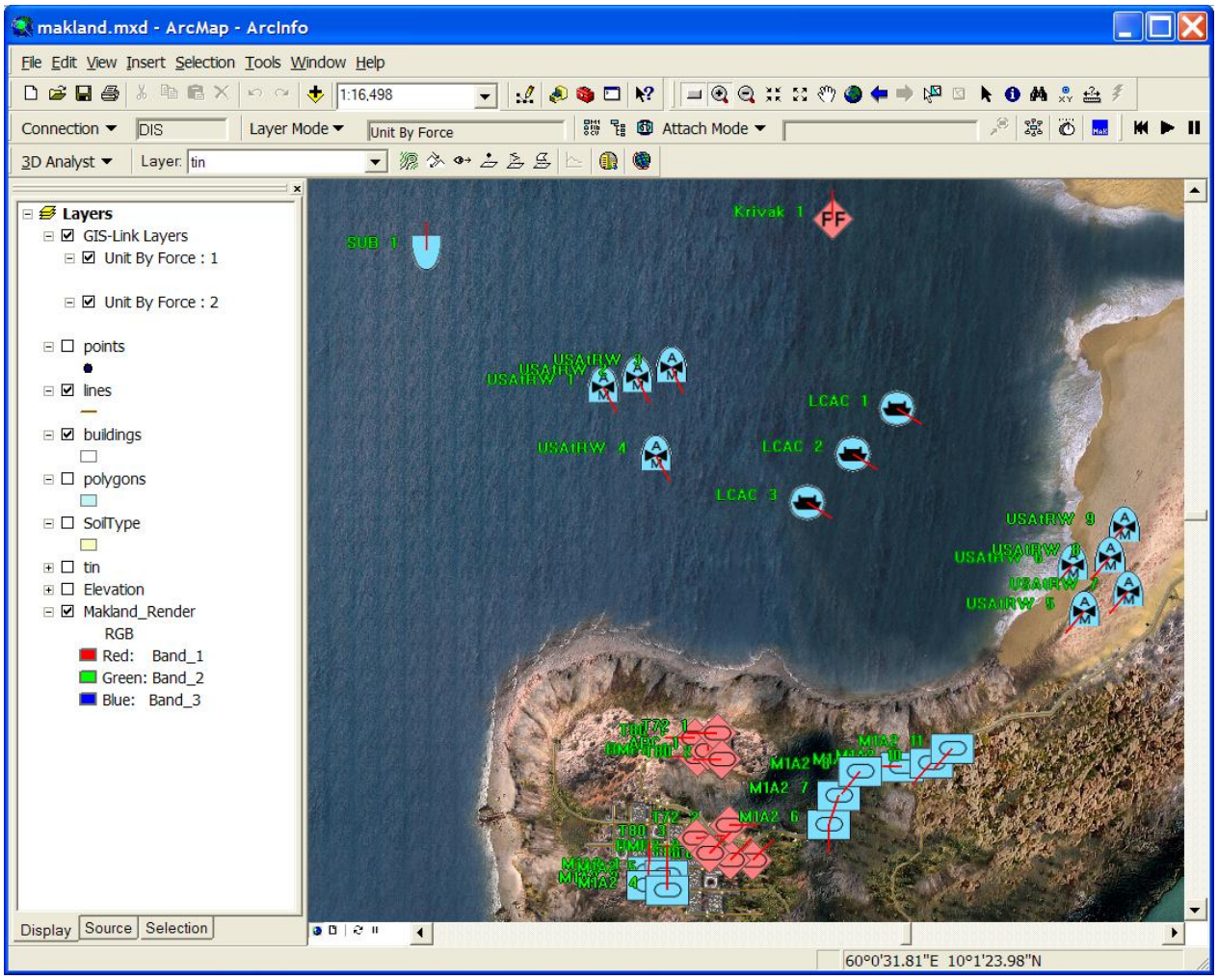

Figure 9: Performance Test Scenario

In order to test the performance of the terrain subsystem prototype, we created a scenario in VR-Forces that consists of ten moving ground vehicles, three moving amphibious vehicles, one moving surface ship, four moving air vehicles, and sixteen non-moving target vehicles (see Figure 9). The performance characteristics were measured based upon three calls in the VR-Forces GDB API and their equivalent call using ArcGIS. The calls are closestIntersection, intersect (1), and intersect (2). The two intersect calls differ in their usage profiles even though they are functionally similar. ClosestIntersection takes in a point and returns the intersection with the terrain that is closest to the elevation of the point. This call is the primary elevation call for ground vehicles. The intersect calls perform line of sight intersection; however the second one is utilized in a manner similar to closestIntersection but returns all intersections in the vertical, where closestIntersection returns elevation. All three of these calls return soil type information at the point of intersection.

The scenario was run and all calls to these functions were logged. This log was used to analyze the calls without the need of the VR-Forces infrastructure. In addition to the input parameters, QueryPerformanceCounter was used to determine the length of time spent in each call. The scenario was run for 60 seconds. Tables 1 through 4 show the results of these performance studies. Times are in microseconds.

Table 1: GBD Terrain

Call	# of Calls	Min	Max	Avg	Std Dev
ClosestIntersection	28768	8.66	2892.83	36.75	33.62
Intersect (1)	5871	6.98	442.23	54.44	33.38
Intersect (2)	5983	3.91	1523.38	62.34	59.47

The times and variations shown in Table 1 were used as the basis for comparison to the GIS terrain calls. One thing to note is the number of terrain API calls made in one minute of a moderate size scenario. These calls returned soil information, whereas the GIS calls in Tables 2 and 3 did not. Table 2 shows the times for the terrain calls using GIS data with a TIN for elevation data. It should be noted that the ESRI caching mechanism was used, which explains some of the maximum times. These typically came from the first call made to each of these routines, which set up the caching. Both the averages and the standard deviations were much higher for the GIS data calls using the TIN. Table 3 shows the same calls but this time we used a 1000 by 1000 element raster for the elevation data. The ground intersection call (ClosestIntersection) was much faster than either the GDB or TIN data, due to the inherent spatial organization in the raster, but the LOS calls (Intersect(1) and Intersect(2)) were much slower. This is due to the LOS algorithm having to look at every raster point along the LOS ray, whereas with a TIN it only has to look at the edge crossings.

Table 2: GIS Data – TIN for Elevation

Call	# of Calls	Min	Max	Avg	Std Dev
ClosestIntersection	28768	81.57	16835.1	94.16	100.41
Intersect (1)	5871	310.1	4922.41	704.73	362.38
Intersect (2)	5983	300.3	8663.95	407.08	226.36

Table 3: GIS Data – Raster for Elevation

Call	# of Calls	Min	Max	Avg	Std Dev
ClosestIntersection	28768	23.75	876.93	24.78	6.16
Intersect (1)	5871	1305.2	504440.1	2066.3	6563.6
Intersect (2)	5983	311.2	11017.04	691.8	762.01

Table 4: GIS Data – TIN for Elevation with Soil Type

Call	# of Calls	Min	Max	Avg	Std Dev
ClosestIntersection	28768	81.57	17699.74	297.74	329.16

Table 4 shows the results of the ClosestIntersection call from the TIN when we returned soil type as well. We only added soil type to this call, because it is more common for an elevation call than a LOS call to require soil type. If you compare this table to the first row of Table 2, you see that average time has increased due to the additional data access to the soiltype layer.

These results are encouraging and show that using GIS terrain data is feasible, but there are still some performance issues to solve. The current design and implementation tasks are looking at improving the performance. One area we are investigating is the use of geocentric coordinates (GCC) in ArcGIS. GCC is used in the DIS and HLA simulation protocols because they provide a continuous coordinate anywhere around the world. We have made the decision to base the terrain API on GCC for the same reason. The GIS data is stored in either Universal Transverse Mercator (UTM) or geodetic (latitude/longitude/altitude). ArcGIS currently does not support GCC directly, but we are working with ESRI to add support for this coordinate system. This would improve performance by eliminating a coordinate conversion step for every terrain access call.

Another area we are working with ESRI to improve performance is in the caching algorithm that they provide, especially with the multipatch features. It turns out multipatches are very slow since they are stored as compressed data in geodatabases. The profiler was showing 75% of the time spent in intersecting the buildings as uncompressing the multipatch. We are also working with them to investigate the use of the terrain feature class for elevation, which may provide some performance improvements by utilizing the hierarchical levels of detail.

One area we have to date avoided is to look for improvements in the way simulation applications use terrain. For example, reducing the number of terrain calls would improve performance. We have found a few instances of unnecessary terrain calls in VR-Forces, but our first effort is to improve performance without having to change the simulation.

8. Conclusions

Commercial GIS products, like those in C/JMTK, provide a powerful set of features for tighter coupling of GIS and M&S systems, providing reductions in time and cost for geospatial data generation for M&S, increased currency of geospatial data for time critical applications, facilitating embedded training in C4ISR systems, and improved interoperability and data correlation between military applications. Our early prototyping suggests and demonstrates the feasibility of using GIS terrain data in a modeling and simulation application. In the near future, we will continue to develop the terrain subsystem to improve performance and extend access to all terrain data.

We will also be looking to extend the terrain subsystem to use GIS data server technology, like the ESRI ArcServer capabilities, to provide terrain data access and replication remotely to M&S applications. This will provide the underlying capabilities for dynamic terrain.

In the future, we would also like to extend the terrain subsystem to provide a 3D visualization capability. This could be done as an extension to the ArcGlobe component of C/JMTK. We would also like to investigate the use of GIS-based analytic and geoprocessing routines during simulation execution, for enhanced terrain reasoning. We would need to develop a framework for communication between the GIS tools and the simulation, which would need to include launching of processes that may take some time to run, while the simulation continues execution. We would also like to develop the dynamic terrain component of the terrain subsystem, based on the GIS data management and distribution mechanisms.

9. Acknowledgement

This material is based upon work supported by the U.S. Army Topographic Engineering Center, Ft. Belvoir, VA, under Contract No. W9132V-06-C-0018. The authors wish to thank David Lashlee and Rick Ramsey of TEC for their support and guidance during this project.

10. References

[1] Stanzione, T., et al., “Integrated Computer Generated Forces Terrain Database”, Fifth Conference on Computer Generated Forces and Behavioral Representation, May 1995.

[2] http://www.delta3d.org/index.php?topic=apis

[3] http://usl.sis.pitt.edu/wjj/otbsaf/libctdb.html#SEC29

[4] Digital Geographic Information Exchange Standard (DIGEST) Part 4, Edition 2.1, September 2000.

[5] http://www.sedris.org/edcs.htm

Author Biographies

THOMAS STANZIONE is the Simulation Technology Manager at MÄK. Mr. Stanzione has over twenty years of experience in modeling and simulation, particularly distributed simulations and computer generated force (CGF) applications, simulation software development, and system integration. At MÄK, Mr. Stanzione is currently the principal investigator on the Smart Terrain Phase II SBIR project for the US Army Natick Soldier Systems Center, as well as the GIS-Enabled Modeling and Simulation project for US Army TEC. Mr. Stanzione holds a Bachelor of Science and Master of Science in Photographic Science from the Rochester Institute of Technology.

KEVIN JOHNSON holds a Master of Science degree in Computer Engineering from Rochester Institute of Technology and a Bachelor of Science degree in Electrical Engineering from Ohio University. Mr. Johnson joined MÄK Technologies in July 1999 and was a principal engineer responsible for improving efficiency on MÄK's SIMinterNET and DARWARS tactical trainer projects. Before his tactical trainer work, Mr. Johnson developed an HLA Runtime Analysis and Monitoring Tools. Specifically he built on HLA to create a distributed data logger that minimized logger generated data traffic on wide area networks. Mr. Johnson is now the lead engineer on the US Army TEC GIS-Enabled Modeling and Simulation project, and was the lead developer on the building interior semantic information portion of the Smart Terrain contract.