Integrating the Natural Environment into a GIS for Decision Support *

Glenn S. Iwerks† and Hanan Samet‡
Computer Science Department, Center for Automation Research, Institute for Advanced Computer Studies
University of Maryland, College Park, Maryland 20742
{iwerks,hjs}@cs.umd.edu

Abstract

Models of the natural environment and how it affects vehicles and sensors can be integrated into a *geographic information system* (GIS). These models can be used in queries to show the affect of the environment on people and equipment. A new integration approach for incorporating such models into a GIS as well as new query language extensions to use them for the purpose of *tactical decision support* (TDS) are presented. TDS focuses on situational awareness for decision support in a dynamic real-time setting, where the decision maker is responsible for guiding and directing people and equipment out in the field. The goal is to integrate this new functionality into a GIS while maintaining the real-time response time required for TDS.

Keywords: GIS, spatial database, environmental modeling, decision support, command and control

1 Introduction

The type of decision support addressed in this paper involves individuals responsible for the remote deployment and direction of people and equipment out in the field. We refer to this as *tactical decision support* (TDS). The directing individuals are known as the decision makers. They consider the current situation and then make choices how to direct the units in their charge. A unit is a small group of people or equipment, closely located to each other, that have a common task or objective. Information about the current situation is relayed through communication channels to the information system used by the decision maker. The information is processed and presented to the decision maker. The decision maker uses this information to formulate plans and instructions for units in the field.

It is important to note the distinction between a TDS system and an *environmental management information system* (EMIS) used for decision support as described in the book "Environmental Information Systems" [4]. Environmental management information systems are used in the support of the decision making process in managing, and making predictions about changes in the natural environment. The user of an EMIS is primarily concerned with how man-made or natural phenomena might affect or change other aspects of the natural environment, such as, how a new bridge will affect the ecosystem around the Chesapeake Bay, or how a longer hunting season might affect the local

*ACM - GIS'99, *pages 73-78, Kansas City, MO, 1999*
†The support of the MITRE Corporation in this research is gratefully acknowledged.
‡The support of the National Science Foundation under Grant IRI-97-12715 is gratefully acknowledged.

dear population over time. TDS on the other hand is concerned with managing people and their equipment. In other words, how will the natural environment affect the performance of machines. For example, how will snow affect the visual range of an aircraft, or how will rain impede the mobility of a tank? The natural environment does play a significant role in TDS, but a TDS system is not an EMIS.

In order to support the tactical decision maker it is important to represent and process any information about the environment that may affect the performance of units in the field. In particular, this paper focuses on the problem of how the current state of the natural environment affects vehicles and sensors. One means to present such information is through a geographic information system (GIS). A GIS is used to process and store spatially-related data. The GIS includes a spatial database to store spatial and non-spatial data. Spatial data has a spatial component such as location and, most importantly, extent. Points, lines and polygons are examples of spatial data. Numbers and strings are examples of non-spatial data.

Originally, GIS was primarily used for static data such as cultural features (buildings, roads, etc.) and terrain. More recently, GIS has been increasingly used for situational awareness in a dynamic setting. For example, some county governments in the United States monitor the position of snow plows using the global positioning system (GPS) and a GIS. To support new applications, GIS functionality has been extended with new reasoning capabilities. For example, some GIS systems provide the ability to compute a shortest path between two points along a connected topology of line segments. Using models, the GIS reasoning capabilities can be further expanded.

To aid in our discussion, the concept of *environmental modeling and reasoning* (EMR) is introduced. EMR considers not only terrain in the modeling of the environment, but also other natural phenomena such as clouds, smoke, fog, precipitation, wind, temperature, tides, sun cycles, moon cycles, etc. *Environmental modeling* involves models of the natural environment populated with data that are used for *environmental reasoning.* Environmental reasoning involves models of man-made systems, such as vehicles and sensors, modeling their interactions with the natural environment. These models are used to answer queries in the GIS. For example, a simple GIS line-of-sight computation assumes that the viewer can see infinitely far and in all wavelengths. In reality, this is not the case. There may be clouds, smoke, fog, or rain obscuring the target. It may be nighttime with only starlight present. The viewer may be using night vision goggles or infrared sensitive sensors. The vehicles may be emitting electromagnetic radiation. All these are factors in whether or not a vehicle can be seen by a viewer some distance off.

The goal of this paper is to present new technology to provide a solution to some of the problems involved in TDS in a dynamic setting that involves interaction between vehicles and sensors with the natural environment. In particular, how to provide EMR within a GIS.

The rest of this paper is organized as follows. Section 2 discusses some previous work on GIS-model coupling techniques. Section 3 describes the *Tactical Decision Support System* (TDSS). Section 4 presents some functions used to support query language extensions, to interact with a TDS system, and explains an algorithm used to implement one of the extensions. Concluding remarks are given in section 5.

2 GIS-model Coupling

We use the terms *loose coupling* and *tight coupling* [3] to describe different approaches to GIS-model integration. Loose coupling is a method of integration in which common files are used as the exclusive means of data exchange between otherwise independent GIS and environmental modeling

systems. Using this approach, the GIS serves as a preprocessor or postprocessor to the modeling system. The advantage of this approach is simplicity. This level of integration between a GIS and modeling software is easy to achieve. Typically all that is needed is a simple interface and perhaps some common file data format conversion routines. The disadvantage is inefficiency and a lack of versatility.

The *Spatial Decision Support System* (SDSS) shell, as presented by Djokic [1], is an example of loose coupling. It is a decision support tool consisting of a collection of components. In [1] a general method of creating interfaces by linking sub-components of the system via a common command shell is presented. The shell allows the user to compose functionality of components in shell scripts, or interactively at the command line. System components consist of a GIS, an expert system (ES), and numerical models (NM). For a system (GIS, ES or NM) to be included as a component of an SDSS shell it must meet four criteria. 1) The component should be as close to state-of-the-art as possible. 2) Each component must have an open file format for data transfer between components. 3) Each component must be controllable via standard keyboard input. 4) Components must run in a common computer environment. The last criterion is for the purpose of simplicity and could be removed with more system integration work. For systems which store data in different file formats, a common intermediate flat ASCII file format is used. This intermediate file format serves as a "data bridge" between systems. A common "command bridge" shell command interpreter must also be developed.

The SDSS shell in [1] uses the ARC/INFO GIS, Nexpert Object expert system, and the HEC-1 rainfall runoff model as components. This example SDSS shell is not given a name. The purpose of the example system is to analyze watershed rain runoff. In the example, the expert system extracts parameters from the GIS and passes them to the HEC-1 numerical model. The results are then stored back into the GIS. The expert system is used to control the interaction of the GIS and NM. The GIS acts as a preprocessor to the numerical models. The expert system replaces an expert user who would otherwise be controlling this interaction. Models are not used within the GIS component of this system while processing spatial data.

The second approach of tight coupling involves a common user interface to system components, common data structures, and GIS-model interaction through mutually accessible procedures. The primary advantages to this approach are efficient interaction between models and GIS, and versatility of model use. The GIS and modeling components of the system exchange data structures via shared memory, network communication, and shared files. Model component functionality can be invoked within the GIS at the procedure level. A disadvantage of many tight coupling approaches is the difficulty involved in modifying and integrating system components. The CLIMEX system, presented by Fedra [3], is an environmental decision support system used for global change modeling and assessment. It incorporates natural environment models such as agricultural and atmospheric models into a GIS using a tight coupling approach.

Much previous work in environmental model and GIS coupling has been in environmental management information systems (EMIS). Although the problem domains of tactical decision support (TDS) and EMIS are different, the fundamental issue of GIS-model coupling in both areas is similar.

3 The Tactical Decision Support System

For this research a tightly coupled approach is used to integrate models into a GIS called the *Tactical Decision Support System* (TDSS). The components of the TDSS consist of a modeling component, a spatial database, and a system front end. The spatial database component that we use is known as SAND [2]. SAND provides the fundamental storage mechanisms, access methods, data structures,

and operation primitives for the system.

The modeling component, referred to as the *environmental model server* (EMS), was derived from a system known as JointSAF developed for the STOW'97 ACTD [9]. JointSAF is a piece of modeling and simulation software used for military training and mission rehearsal. It provides a set of integrated models of the natural environment, vehicles, and sensors modeling their interactions with each other. These models are computationally light and have been verified and validated to be accurate representations of the real world sufficient for training and mission rehearsal. The level of accuracy and the response time required for real-time simulation is also sufficient for the TDSS.

The third component is called the Spatial Spreadsheet [5]. It serves as the TDSS front end. It is used to visualize changes in a dynamic spatial database, (see Figure 1). It also serves as a means to organize large amounts of spatial data, quickly formulate queries on data, and propagate changes in the source data to query results via a spreadsheet paradigm.

The coupling of the EMS component to the rest of the system is achieved through CORBA [6]. In general, the main drawback to a tightly coupled architecture in our problem domain is the amount of work needed to adapt new environment models to an existing system. To overcome this drawback we use a set of interface specifications for information transfer between models and GIS. The idea is to make the interface general enough so new models can be incorporated into the GIS using the same interfaces as existing models. If successful, no additional work need be done to adapt new models. The only requirement is that the new models conform to the interface specification. To make the specification useful it should be independent of the model and GIS implementation languages, and independent of computer hardware architecture. CORBA IDL [6] fulfills these requirements for a specification language. The Spatial Spreadsheet acts as a CORBA client to an environmental model server (see Figure 2). The advantage of this architecture is independence between the EMS and the spatial database components. This allows for different model servers to be used in different applications to fulfill different user requirements for fidelity and efficiency.

The MICO [8] CORBA object request broker is used for the TDSS implementation. IDL interfaces to the EMS are mapped to C++ classes. On the client side, these C++ classes are used to implement Tcl commands used by the Spatial Spreadsheet. The Spatial Spreadsheet component is implemented entirely in incremental Tcl/Tk [7]. SAND also provides a Tcl/Tk interface to it's underlying C/C++ implementation.

4 Query Functions

The TDSS architecture allows for new functions to be used within spatial database queries. The resulting functions are designed in such a way as to render details of the current state of the natural environment transparent to the user. For example, the user can give a vehicle type and location as parameters to a function and get back a top speed for that location. The function considers all the factors which may affect speed such as precipitation, soil type, visibility, terrain slope, etc. These new functions are invoked during the evaluation of query predicates to access EMS capabilities. In this section, we present some example functions. Some of these functions such as speed (section 4.1) and visibility (section 4.2) are available within the EMS as is. The mass_area_plot function (section 4.3) required some augmentation of the EMS derived from JointSAF. The mass_area_plot algorithm is described in detail.

4.1 Speed

scalar speed(*vehicle_type*, *location*)

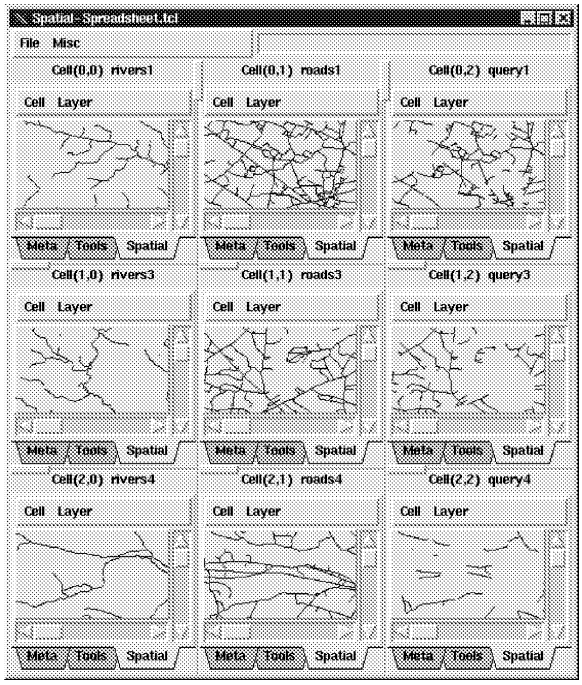

Figure 1: The Spatial Spreadsheet

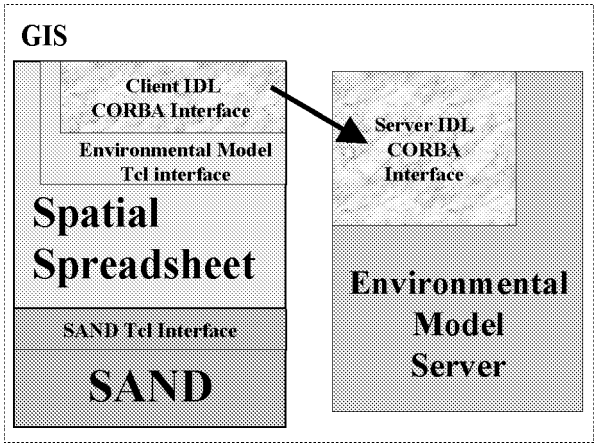

Figure 2: TDSS GIS components

Parameter *vehicle_type* is a string identifier denoting the type of vehicle, and *location* is a 2D point. The EMS projects 2D points onto the terrain surface to find locations in 3D space. The speed function returns the maximum speed, in kilometers per hour, that a vehicle of type *vehicle_type* can achieve at the given *location* under the current environmental conditions. When it is invoked the environmental affects on the given vehicle type are considered and the top speed is calculated all within the EMS. This particular function is a fundamental primitive provided in the JointSAF EMS. No coding beyond the interface implementation on the server side is necessary.

For example, consider a relation R storing the location, type and id for a number of vehicles. R has schema (*vtype*, *loc*, *id*) where *vtype* is a vehicle's type, *loc* is its location, and *id* is a unique identifier for each vehicle. Now, consider the following query on that relation presented in SQL.

```
SELECT *
FROM R
WHERE  speed(R.vtype, R.loc) > 50
```

The resulting relation will contain the tuples for all vehicles that can achieve a speed of at least 50 kilometers per hour at their present location.

4.2 Visibility

scalar visibility(*observer_loc*, *target_loc*)

Parameters *observer_loc* and *target_loc* are 2D point locations. This function returns a scalar value in the range of $[0, 1]$ where 0 corresponds to not visible and 1 is completely visible. For example, a returned value of 0.5 in the presence of atmospheric obscurants means that the amount of light coming from the object is equal to the amount of light returned from an object of half its cross section size, in the absence of obscurants. This function, like the speed function, is a fundamental primitive provided by the JointSAF EMS.

For example, consider the relation R used in the speed query example above, and some location L in the following query.

```
SELECT *
FROM R
WHERE  visibility(L, R.loc) ≥ 0.5
```

The resulting relation for this query contains all the vehicles in R which are at least 50% visible from location L.

4.3 Mass Area Plot

region mass_area_plot(*observer*, *boundary*,
sample_dist, *vis_threshold*)

Parameter *observer* is a 2D point, *boundary* is a 2D rectangle where the edges of *boundary* are orthogonal to the coordinate axis, *sample_dist* is a scalar measured in meters, and *vis_threshold* is a scalar in the range $[0, 1]$.

For example, consider the relation R used in the speed query example above, and a rectangular boundary B in the following query.

```
SELECT *
FROM R
WHERE  mass_area_plot(R.loc, B, 100, 0.5) ∩ B ≠ ∅
```

The resulting relation contains all the vehicles in relation R that have at least 50% visibility of some part of the terrain contained within boundary B viewed at 100 meter resolution (i.e. sampled every 100 meters).

The mass_area_plot algorithm is derived from the visibility function. It required a new algorithm to be implemented on the EMS side of the system. This does not constitute a new model being added to the EMS, but rather it is based on a sampling strategy using existing models. The algorithm shows how the visibility function is used.

To describe the mass_area_plot algorithm, the following conventions are used.

- A point, say p, is defined as two scalar coordinates denoted by $X(p)$ and $Y(p)$.
- Assignment to the x coordinate of some point p the value of some scalar z is denoted $X(p) \leftarrow z$.
- Assignment to the y coordinate of some point p the value of some scalar z is denoted $Y(p) \leftarrow z$.
- For some point p and some point q: $p \leftarrow q \equiv (X(p) \leftarrow X(q); Y(p) \leftarrow Y(q))$.
- For some point p, some point q and some scalar α: $p \leftarrow q + \alpha \equiv (X(p) \leftarrow X(q) + \alpha; Y(p) \leftarrow Y(q) + \alpha)$.
- A rectangle, say r, is defined by two opposite points denoted by $MIN(r)$ and $MAX(r)$ such that $\forall p \in r :: X(MIN(r)) \leq X(p) \wedge Y(MIN(r)) \leq Y(p) \wedge X(MAX(r)) \geq X(p) \wedge Y(MAX(r)) \geq Y(p)$ where p denotes a point.
- For some scalar α, $ceiling(\alpha)$ returns the integral component of α incremented by 1.
- A *region* is a set of polygons.
- A null *region* is the empty set.

The mass_area_plot function returns a *region* visible from *observer* under the current environmental conditions within the given parameters. Points on the terrain surface are tested for how visible they are from *observer* using the visibility function described in Section 4.2. The rectangle *boundary* is used to limit the area tested. The amount of visibility from *observer* is quantified as a number in the range of $[0, 1]$. If a test target point has a visibility value in the range $[vis\_threshold, 1]$ then that patch of terrain around the point is assumed to be visible. The union of these patches forms the *region* returned by the function (see Figure 3).

In the algorithm presented below, point *start* is the first visibility test point. Point *end* is the last visibility test point. Point *target* is the current target point for visibility testing. Scalar *num_samples_x* is the number of test points in a row. Scalar *num_samples_y* is the number of test points in a column. Variable *subregion* is a rectangle. Region *result* is the returned value (see Figure 3). The algorithm's input parameters *vis_threshold*, *sample_dist*, *boundary*, and *observer* are as described above.

```
region mass_area_plot(observer, boundary, sample_dist,
                      vis_threshold)
  offset ← sample_dist ÷ 2
  start ← MIN(boundary) + offset
  num_samples_x ← ceiling((X(MAX(boundary))
      −X(MIN(boundary))) ÷ sample_dist)
  num_samples_y ← ceiling((Y(MAX(boundary))
      −Y(MIN(boundary))) ÷ sample_dist)
  X(end) ← ((num_samples_x − 1) × sample_dist) + X(start)
  Y(end) ← ((num_samples_y − 1) × sample_dist) + Y(start)
  result ← ∅
  for Y(target) ← Y(start) to Y(end) step sample_dist
    for X(target) ← X(start) to X(end) step sample_dist
      if visibility(observer, target) ≥ vis_threshold then
        MIN(subregion) ← target − offset
        MAX(subregion) ← target + offset
        result ← result ∪ subregion
```

```
      endif
    endfor
  endfor
  result ← result ∩ boundary
  return result
```

The algorithm iterates through sample target points laid out in a grid. If the visibility between the *observer* point and the current test *target* point is at or above the threshold value, then a *subregion* rectangle is created. The *subregion* rectangle is centered at the *target* point, and the length of each edge is equal to *sample_dist*. The *result* region is the union of all *subregion* rectangles created. Finally, the *result* region is intersected with the *boundary* rectangle. This is to clip the resulting region to fit within the boundary rectangle.

There exists an algorithm in JointSAF to make terrain visibility overlays on a map for the user to view. This algorithm and the mass_area_plot algorithm both use the visibility function in some form of sampling strategy. An important difference is that the JointSAF algorithm does not produce a resulting data structure which can be used in further processing. For this reason, the *mass_area_plot* algorithm needed to be created.

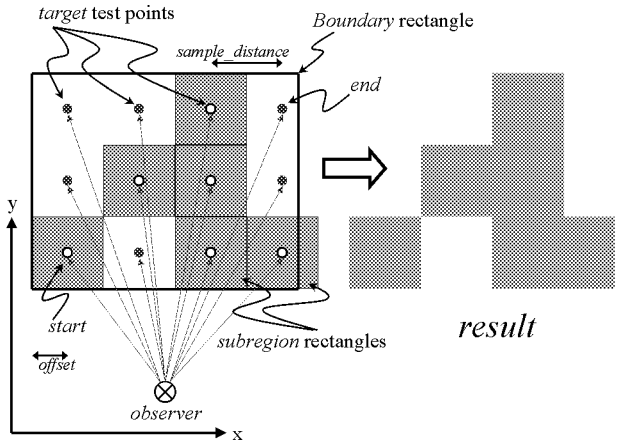

Figure 3: mass_area_plot: ⊗ is the observer position, ∘ indicates *target* test locations with visibility values above the *vis_threshold*, and • indicates *target* test locations with visibility values below the *vis_threshold*.

5 Conclusion

The TDSS implementation gave good results with respect to query function response time using the models. The main performance bottleneck was the overhead associated with the CORBA interface rather than the models themselves. The query results, using the new functions, changed as the environment changed, as it should, to show how the natural environment affects vehicles and sensors. It is our belief that this kind of system can help decision makers responsible for directing units in the field appreciate the influence of the natural environment on them. For this purpose we have created a system that is versatile in its use of models within a GIS and has the ability to invoke models within the query language itself, and can respond in real-time. For future work we wish to integrate more EMR functionality into the TDSS.

References

[1] D. Djokic. Toward a general-purpose decision support system using existing technologies. In *GIS and Environmental Modeling: Progress and Research Issues*, pages 353–356, GIS World Books, Fort Collins, CO, 1996.

[2] C. Esperança and H. Samet. Spatial database programming using SAND. In M. J. Kraak and M. Molenaar, editors, *Proceedings of the Seventh International Symposium on Spatial Data Handling*, volume 2, pages A29–A42, Delft, The Netherlands, August 1996.

[3] K. Fedra. Distributed models and embedded gis. In *GIS and Environmental Modeling: Progress and Research Issues*, pages 413–417, GIS World Books, Fort Collins, CO, 1996.

[4] O. Günther. *Environmental Information Systems.* Springer-Verlag, Belin Germany, 1998.

[5] G. Iwerks and H. Samet. The spatial spreadsheet. In *Visual Information and information Systems: Third International Conference, VISUAL'99, Amsterdam, The Netherlands, June 1999 Proceedings*, pages 317–324, Belin Germany, 1999. Springer-Verlag.

[6] OMG. Corba/iiop 2.2 specification. available at www.omg.org, Febuary 1998.

[7] J. K. Ousterhout. *Tcl and the Tk Toolkit.* Addison-Wesley, Reading, MA, 1995.

[8] A. Puder and K. Romer. *MICO is CORBA.* dpunkt - Verlag, Heidelberg, Germany, 1998.

[9] D. Whitney, R. Reynolds, D. Sherer, P. Dailey, M. Driscoll, M. Zettlemoyer, R. Schultz, and I. Watkins. Impacts of the environment on warfighter training: Stow 97 experiences with taos. In *98 Spring Simulation Interoperability Workshop*, University of Central Florida, Orlando, FL 32826, 1998.