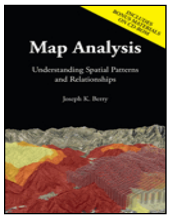
Beyond Mapping III

Introduction – The GIS Evolution (Further Reading)


Map Analysis book

<*Click here*> *for a printer-friendly version of this topic (.pdf).*

(*Back to the Table of Contents*)

GIS Software's Changing Roles

(GeoWorld, September 1998)

(*return to top of Topic*)

Although GIS is just three decades old, the approach of its software has evolved as much as its capabilities and practical expressions. In the 70's software development primarily occurred on campuses and its products relegated to library shelves of theses. These formative years provided the basic organization (both data and processing structures) we find in the modern GIS. Also it provided a raging debate centered on "vector vs. raster" formats and efficient algorithms for processing— techy-stuff with minimal resonance outside of the small (but growing) group of innovators.

For a myriad of reasons, this early effort focused on GIS technology itself rather than its applications. First, and foremost, is the necessity of building a viable tool before it can be taken on the road to practical solutions. As with most revolutionary technologies, the "chicken and the egg" parable doesn't apply—the tool must come before the application.

This point was struck home during a recent visit to Disneyland. The newest ride subjects you to a seemingly endless harangue about the future of travel while you wait in line for over an hour. The curious part is that the departed Walt Disney himself is outlining the future through video clips from the 1950s. The dream of futuristic travel (application) hasn't changed much and the 1990s practical reality (tool), as embodied in the herky-jerky ride, is a long way from fulfilling the vision.

What impedes the realization of a technological dream is rarely a lack of vision, but the nuts and bolts needed in its construction. In the case of GIS, the hardware and software environments of the 1970s constrained its use outside of academia. Working with 256K memory and less than a megabyte of disk storage made a GIS engine perform at the level of an old skateboard. However, the environments were sufficient to develop "working prototypes" and test their theoretical foundations. The innovators of this era were able to explore the conceptual terrain of representing "maps as numbers," but their software products were woefully impractical.

With the 1980s came the renaissance of modern computers and with it the hardware and software environments needed by GIS. The research-oriented software gave way to operational systems. Admittedly, the price tags were high and high-end, specialized equipment often required, but the suite of basic features of a modern GIS became available. Software development switched from specialized programs to extensive "toolboxes" and subsequently spawned a new breed of software specialists.

Working within a GIS macro language, such as ARCINFO's Arc Macro Language (AML), customized applications could be addressed. Emphasis moved from programming the "tool" within generis computer languages (e.g., FORTRAN and Pascal), to programming the "application" within a comprehensive GIS language. Expertise broadened from geography and computers to an understanding of the context, factors and relationships of spatial problems. Programming skills were extended to spatial reasoning skills—the ability to postulate problems, perceive patterns and interpret spatial relationships.

From an application developer's perspective the floodgates had opened. From an end user's perspective, however, a key element still was missing—the gigabytes of data demanded by practical applications. Once again GIS applications were frustrated. This time it wasn't the programming environment as much as it was the lagging investment in the conversion from paper maps to their digital form.

But another less obvious impediment hindered progress. As the comic strip character Pogo might say, *"…we have found the enemy and it's us.*" By their very nature, the large GIS shops established to collect, nurture, and process spatial data intimidated their potential customers. The required professional sacrifice at the GIS altar "down the hall and to the right" kept the herds of dormant users away. GIS was more often seen within an organization as an adversary competing for corporate support (a.k.a., a money pit) than as a new and powerful capability one could use to improve workflow and address complex issues in entirely new ways.

The 1990s saw both the data logjam burst and the GIS mystique erode. As Windows-based mapping packages appeared on individuals' desks, awareness of the importance of spatial data and its potential applications flourished. Direct electronic access enabled users to visualize their data without a GIS expert as a co-pilot. For many the thrill of "visualizing mapped data" rivaled that of their first weekend with the car after the learner's permit.

So where are we now? Has the role of GIS developers been extinguished, or merely evolved once again? Like a Power Rangers transformer, software development has taken two forms that blend the 1970s and 80s roles. These states are the direct result of changes in software programming approaches in general, and "object-oriented" programming in particular.

MapInfo's MapX and ESRI's MapObjects are tangible GIS examples of this new era. These packages are functional libraries that contain individual map processing operations. In many ways they are similar to their GIS toolbox predecessors, except they conform to general programming standards of interoperability, thereby enabling them to be linked easily to the wealth of non-GIS programs.

Like using a Lego set, application developers can apply the "building blocks" to construct specific solutions, such as a real estate application that integrates a multiple listing geo-query with a pinch of spatial analysis, a dab of spreadsheet simulation, a splash of chart plotting and a sprinkle of report generation. In this instance, GIS functionality simply becomes one of the ingredients of a solution, not the entire recipe.

In its early stages, GIS required "bootstrap" programming of each operation and was the domain of the computer specialist. The arrival of the GIS toolbox and macro languages allowed an application specialist to develop software that tracked the spatial context of a problem. Today we have *computer specialists* generating functional libraries and *application specialists* assembling the bits of software from a variety of sources to tailor comprehensive solutions.

The distinction between computer and application specialist isn't so much their roles, as it is characteristics of the combined product. From a user's perspective the entire character of a GIS dramatically changes. The look-and-feel evolves from a generic "map-centric view "to an "application-centric" one with a few tailored buttons that walk users through analysis steps that are germane to an application. Instead of presenting users with a generalized set of map processing operations as a maze of buttons, toggles and pull-down menus, only the relevant ones are integrated into the software solution. Seamless links to nonspatial programming "objects," such as pre-processing and post-processing functions, are automatically made.

As the future of GIS unfolds, it will be viewed less as a distinct activity and more as a key element in a thought process. No longer will users "break shrink-wrap" on stand-alone GIS systems. They simply will use GIS capabilities within an application and likely unaware of the underlying functional libraries. GIS technology will finally come into its own by becoming simply part of the fabric of software solutions.

What Is Object-Oriented Technology Anyway?

(GeoWorld, September 1996)

(return to top of Topic)

Object-oriented technology is revolutionizing GIS. At every turn, from researchers to salespersons, the buzz words "object-oriented" crop up. Mere mention of the concept greases the probability of funding and immediately places the competition on the defensive. But what does object-oriented really mean? Are there different types? Are there shades of object-oriented-ness? Can a system be "just a little" object-oriented? How can you tell?

Like most over-used and under-grasped terms, object-oriented (OO, pronounced like a cow's moo, but without the "m") has a variety of near religious interpretations. In reality, however, there are just three basic forms: 1) Object-Oriented User Interfaces (OOUI), 2) Object-Oriented Programming Systems (OOPS) and 3) Object-Oriented Data Base Management (OODBM). Yep, you got it… *oo-eee, oo-la-la, oops 'n ooD'BUM, ra'ma, ra'ma ding-dong, whoopee…* sounds more lyrics from a 60's hit than a GIS revolution. *OOUI* uses "icons" in place of command lines and/or menu selections to launch repetitive procedures. It's what used to make a Mac a Mac, and all the other PC's just techno-frustration.

With the advent of Windows '95, however, OOUI's have moved from "state-of-the-art" to commonplace. Its objects include all those things on the screen you that drag, drop, slide, toggle, push and click. On the "desktop" you have cute little sketches including file folders, program shortcuts, a recycle bin and task bar. In a "dialog box" the basic objects include the check box, command button, radio (or option) button, slider bar and spinner button, in addition to the drop-down, edit and list boxes. When activated, these objects allow you to communicate with computers by *point 'n click*, instead of *type 'n retype*. Keep in mind that OOUI defines "a friendly user interface (FUI) as a graphical user interface (GUI)"… yep you got it again, a *phooey-gooey*.

In addition to icons, OOUI's use "glyphs." These are graphical objects are controlled by programs to perform specific functions. The best examples of glyphs are in screen savers. The little dog, man or monster roaming across your screen which "destroys" it (sends it to black) piece by piece is actually a "screen object" expression of a computer program. In a GIS/GPS package, the blinking dot or arrow depicting a vehicle's movement utilize glyphs that roam across a registered GIS map, responding to real-time GPS positioning. In advanced systems, such as air traffic control, the glyphs are often intelligent. When two graphical objects (airplanes) come within a specified distance of each other, they can blink rapidly or change color. In short, OOUI's icons and glyphs encapsulate program parameters, control and linkages.

Just about every GIS out there uses a phooey-gooey interface, which must make them all object-oriented. Technically it does, so you had better ask a few more questions to differentiate the nature and degree of OO-ness. *OOPS* technology uses "widgets" in the development of computer code. Visual Basic and Visual C are examples of object-oriented programming systems. These packages allow a programmer to generate code by simply flowcharting a program's logic. At each "drafting" step, a widget (flowcharting object) is moved into place and a dialog box is popped-up. Completion of the dialog box writes the appropriately coded gibberish defining the step to a file.

Whereas an OOUI dialog box is independent, an OOPS dialog box is one in a linked set of commands depicted in the flowchart. Most modern GIS display at least a minimal level of OOPS-ness. By turning on the "command log" or "macro builder," a series of dialog box entries can be stored to a file, which in turn, can be rerun to repeat the processing. A robust OOPS, however, uses rigorous flowcharting structure, rules and mechanics to "graphically" assemble a computer application. From this perspective, OOPS is an effective programmer's tool, as it provides an easier way (thank heavens) to develop fully structured computer programs.

The OOPS flowchart captures a succinct diagram of an application's logic, as well as actual code. As GIS moves from descriptive geo-query applications to prescriptive modeling, the communication of logic becomes increasingly important. An insightful end-user doesn't want to simply behold colorful map renderings of elk habitat or retail sales competition analysis—the gospel from mount GIS. Visualization of a mapped result quickly turns to discussion of the assumptions and expressions used in its derivation. The focus becomes one of cognitive space, as much as geographic space. The OOPS flowchart provides a mechanism for both communicating and interacting with model logic.

This topic was discussed in detail several issues ago (see Author's Note) within the contexts of a "humane GIS interface" and a "dynamic map pedigree." The discussion noted that the next generation of GIS will allow users to "interrogate and interact" with a model logic by simply mouse-clicking on the widgets (boxes and arrows) forming a model's flowchart, or more aptly termed, the modeled map's pedigree. If you take exception with the calibration of a GIS model (e.g., "Hey, slope isn't that important to elk habitat"), then click on those disagreeable portions of the flowchart, change them, and rerun the model. You can compare your "personalized" version with those of others to see the spatial impact of the different cognitive perspectives.

The full incorporation of a robust OOPS makes the transition of GIS from simply a "data-painter" to an interactive "spatial spreadsheet." Several GIS vendors are already well on their way. In fact, I'll bet you lunch that within a couple of years all GIS will have implemented OOPS—be careful, don't lose your lunch over the onset of *oo*.

Author's Note*: GeoWorld November and December, 1993).*

Spatial Objects—Parse and Parcel of GIS?

(GeoWorld, October 1996)

(*return to top of Topic*)

The previous section suggested that there are three basic types of "object-oriented-ness." The distinctions between an OOUI (*object-oriented user interface*) and an OOPS (*object-oriented programming system*) were tackled. That leaves the third type of "OO-ness," object-oriented database management (*OODBM*, pronounced ooD'BUM) for scrutiny this time. The OODBM concept provides database procedures for establishing and relating spatial objects, more generally referred to as "data objects." Early DBM systems used "flat files," which were nothing more than long lists of data. The records, or data items, were fairly independent (usually just alpha-numerically sorted) and items in separate files were totally independent.

With the advent of "relational databases," procedures were developed for robust internal referencing of data items, and more importantly, indexing among separate data sets. OODBM, the next evolutionary (or quite possibly, revolutionary) step, extends direct data indexing by establishing procedural rules that relate data. The rules are used to develop a new database structure that interconnects data entries and greatly facilitates data queries. They are a natural extension of our current concerns for data standards and contemporary concepts of data dictionaries and metadata (their discussion reserved for another article).

The OODBM rules fall into four, somewhat semantically-challenging categories: *encapsulation, polymorphism, structure,* and *inheritance*. Before we tackle the details, keep in mind that "object-oriented" is actually a set of ideas about data organization, not software. It involves coupling of data and the processing procedures that act on it into a single package, or *data object*. In an OODBM system, this *data/procedure* bundle is handled as a single unit; whereas the data/procedure link in a traditional database system is handled by separate programs.

It can be argued (easily) that traditional external programs are cumbersome, static and difficult to maintain— they incrementally satisfy the current needs of an organization in a patchwork fashion. We can all relate to horror stories about corporate databases that have as many versions as the organization has departments (with 10 programmers struggling to keep each version afloat). The very real needs to minimize data redundancy and establish data consistency are the driving forces behind OODBM. Its application to GIS is simply one potential expression of the looming radical changes in database technology.

Encapsulation is OODBM's cornerstone concept. It refers to the bundling of the data elements and their processing for specific functions. For example, a map feature has a set of data elements (coordinates, unique identifier, and thematic attributes). Also it can have a set of basic procedures (formally termed "methods") it recognizes and responds to, such as "calculate your dimensions." Whenever the data/procedure bundle is accessed, *"its method is executed on its data elements."* When the "calculate your dimensions" method is coupled with a polygonal feature, perimeter and area are returned; with a linear feature, length is returned; and, with a point feature, nothing is returned.

If the database were extended to include 3-dimensional features, the same data/procedure bundle would work, and surface area and volume would be returned… an overhaul of the entire application code isn't needed, just an update to the data object procedure. If the data object is used a hundred times in various database applications, one OODBM fix fixes them all.

The related concept of *polymorphism* refers to the fact that the same data/procedure bundle can have different results depending on the methods that are executed. In the "calculate your dimensions" example, different lines of code are activated depending on the type of map feature. The "hidden code" in the data/procedure bundle first checks the type of feature, then automatically calls the appropriate routine to calculate the results… the user simply applies the object to any feature and it sorts out the details.

The third category involves *class structure* as a means of organizing data objects. A "class" refers to a set of objects that are identical in form, but have different specific expressions, formally termed "instances." For example, a linear feature class might include such diverse instances as roads, streams, and power lines. Each instance in a class can occur several times in the database (viz. a set of roads) and a single database can contain several different instances of the same class. All of the instances, however, must share the common characteristics of the class and all must execute the set of procedures associated with the class.

Subclasses include all of the data characteristics and procedural methods of the parent class, augmented by some specifications of their own. For example, the linear feature class might include the subclass "vehicle transportation lines" with the extended data characteristic of "number of potholes." Whereas the general class procedure "calculate your dimensions" is valid for all linear feature instances, the subclass procedure of "calculate your potholes per mile" is invalid for streams and power lines.

The hierarchical ordering embedded in class structure leads to the fourth rule of OODBM—*subclass inheritance*. It describes the condition that each subclass adheres to all the characteristics and procedures of the classes above them. Inheritance from parent classes can occur along a single path, resulting in an inheritance pattern similar to a tree branch. Or, it can occur along multiple paths involving any number of parent classes, resulting in complex pattern similar to a neural network.

In general, complex structure/inheritance patterns embed more information and commonality in data objects which translates into tremendous gains in database performance (happy computer). However, designing and implementing complex OODBM systems are extremely difficult and technical tasks (unhappy user). Next month we'll checkout the reality, potential and pitfalls of object-oriented databases as specifically applied to GIS… good or bad idea; inevitable step or technical fantasy?

Does Anyone Object?

(GeoWorld, November 1996)

(***return to top of Topic***)

The last several sections dealt with concepts and procedures surrounding the three forms of object-oriented (OO) technology—*OUI*, *OOPS* and *OODBM*. An object-oriented user interface (OOUI) uses point-and-click *icons* in place of command lines or menu selections to launch repetitive procedures. It is the basis of the modern graphical user interface. An object-oriented programming system (OOPS) uses flowcharting *widgets* to graphically portray programming steps which, in turn, generate structured computer code. It is revolutionizing computer programming—if you can flowchart, you can program. Both OOUI and OOPS are natural extensions to GIS which make it easier to use (a necessity if users are really going to use GIS).

Object-oriented database management (OODBM), on the other hand, represents a radical change in traditional GIS data structuring. Whereas OOUI and OOPS are actual or near realities in most contemporary GISs, OODBM is in its infancy. In fact, some users might even question if it is a natural extension. Heck, some might even object.

The OOBDM concept provides database procedures for establishing and relating data *objects*. Last issue discussed the four important elements of this new approach (*encapsulation*, *polymorphism*, *class structure*, and *inheritance*). As a quick overview, consider the classical OODBM example. When one thinks of a house, a discrete entity comes to mind. It's an object placed in space, whose color, design and materials might differ, but it's there and you can touch it (spatial object). It has some walls and a roof that separates it from the air and dirt surrounding it. Also, it shares some characteristics with all other houses—rooms with common purposes. It likely has a kitchen, which in turn has a sink, which in turn has a facet.

These items form discrete objects which share both spatial and functional relationships. For example, if you encounter a facet in space you are likely going to find it connected to a sink. These relational objects can be found not only in a kitchen, but a bathroom as well. In an OODBM, the data (e.g., facet) and relationships (e.g., rooms with sinks therefore facets) are handled as a single unit, or *data/procedure bundle*. Within a GIS implementation, one "hit to disk" you know where and what an object is, as well as its context and linkages.

Such a system (if and when it is implemented) is well suited for organizing the spatial objects comprising a GIS. For example, we know that all airports must have a road connecting it to the highway network. In an OODBM, this connection is part of the data/procedure bundle. In a traditional "layered" GIS, you must overlay a map of all roads with a map of all municipal facilities, and then select the airport/road coincidence to identify the connection.

For years GIS users have preempted this process by constructing a single, huge "vector composite" which overlays all of the maps within a data base. Once constructed, a simple database command selects the "regions" (i.e., the "wee-little" polygons formed by the intersection of the multitude of lines) which meets any conceivable query. In a sense, the vector composite approach results in an OODBM-like system based on spatial coincidence. It generates a complete set of spatial objects with a direct data base link to their characteristics. The objects might not be readily recognizable (e.g., a timber stand will be "broken" into pieces representing different soil and slope conditions you can't see), but the GIS can easily respond to your queries involving the coincidence of any map layers.

The main problem of both OOBDM and its “vector composite” cousin comes from their static and discrete nature. With OODBM, the specification of “all encompassing rules” is monumental. If only a few data items are used, this task is doable. However, within the sphere of a “corporate” GIS, the rule set geometrically expands and soon becomes unmanageable. The “vector composite” technique provides a mechanism to automatically generate at least one dimension for defining objects (spatial coincidence).

However, each time a layer changes, a new composite must be generated. This can be a real hassle if you have a lot of layers and they change a lot. New data base “patching” procedures (map feature-based) hold promise for generating the updates for just the areas affected, rather than overlaying the entire set of layers.

Even if the “mechanics” of applying OODBM to GIS can be streamlined, problems still exist. First, the approach assumes all spatial objects are discrete—bounded by well-defined edges. While this might be the case for a house or a road, it’s a bit more “fuzzy” for soils (uncertainty) and barometric pressure (continuous surface). In fact, a “fuzzy object” itself seems a contradiction. Secondly, many GIS applications involve cognitive expressions as much as they involve spatial objects. For example, areas of good elk habitat or high probability of sales aren’t things you bump into on a walk, but interpretations of spatial relationships.

Most often, these applications involve interactive processing as users run several “what if…” scenarios. From this perspective, habitat and sales maps aren’t composed of spatial objects as much they are iterative expressions of differing opinions and experience (spatial spreadsheet). For OODBM to work and garnish data accessing efficiencies there must be universal agreement on the elements and paradigms used in establishing fixed data objects. In one sense, the logic of a GIS model for habitat or sales can be equated to an OODBM rule as it carries both the data (base map ingredients) and procedure (command macro recipe) defining something.

In fact, GIS applications are like banana-bread. If we all agree on the recipe, then it can be stored as an object; however, if everyone insists on adding varying amounts of ingredients and/or entirely new ingredients, then banana-bread is not a “fully structured object.” In these instances you need a cupboard full of directly accessible basic ingredients (i.e., base map layers). In short, object-oriented data management holds promise for increased efficiencies the handling of spatially discrete data (for descriptive inventory and mapping), while *process-oriented* GIS modeling systems (for prescriptive analysis and spatial reasoning) gives you “*a license to color outside the lines.*”

But another less obvious impediment hindered progress. As the comic strip character Pogo might say, “*…we have found the enemy and it’s us.*” By their very nature, the large GIS shops established to collect, nurture and process spatial data, intimidated their potential customers. The required professional sacrifice at the GIS altar “down the hall and to the right” kept the herds of dormant users away. GIS was more often seen within an organization as an adversary for corporate support (a.k.a., money pit) than as a new and powerful capability one could use to improve workflow and address complex issues in entirely new ways.

The 90's saw both the data logjam and GIS mystique erode. As Windows-based mapping packages appeared on individuals' desks, awareness of the importance of spatial data and its potential applications flourished. Direct electronic access enabled users to visualize their data without a GIS expert as co-pilot. For many the thrill of "visualizing mapped data" rivaled that of their first weekend with the car after the learner's permit.

So where are we now? Has the role of GIS developers been extinguished, or merely evolved once again? Like a Power Rangers transformer, software development has taken two forms that blend the 70's and 80's roles. These states are the direct result of changes in software programming approaches in general and "object-oriented" programming in particular.

MapInfo's MapX and ESRI's MapObjects are tangible GIS examples of this new era. These packages are functional libraries that contain individual map processing operations. In many ways they are similar to their GIS toolbox predecessors, except they conform to general programming standards of interoperability, thereby enabling them to be linked easily to the wealth of non-GIS programs.

Like using a Lego set, application developers can apply the "building blocks" to construct specific solutions, such as a real estate application that integrates a multiple listing geo-query with a pinch of spatial analysis, a dab of spreadsheet simulation, a splash of chart plotting and a sprinkle of report generation. In this instance, GIS functionality simply becomes one of the ingredients of a solution not the entire recipe.

In its early stages, GIS required "bootstrap" programming of each operation and was the domain of the computer specialist. The arrival of the GIS toolbox and macro languages allowed an application specialist to develop software that tracked the spatial context of a problem. Today we have *computer specialists* generating functional libraries and *application specialists* assembling the bits of software from a variety of sources to tailor comprehensive solutions.

The distinction between computer and application specialist isn't so much their roles, as it is characteristics of the combined product. From a user's perspective the entire character of a GIS dramatically changes. The look-and-feel evolves from a generic map-centric view to one of a few tailored buttons that walk users through analysis steps that are germane to an application. Instead of presenting users with a generalized set of map processing operations as a maze of buttons, toggles and pull-down menus, only the relevant ones are integrated into the software solution. Seamless links to non-spatial programming "objects," such as pre-processing and post-processing functions, are automatically made.

As the future of GIS unfolds, it will be viewed less as a distinct activity and more as a key element in a thought process. No longer will users "break shrink-wrap" on stand-alone GIS systems. They simply will use GIS capabilities within an application and likely unaware of the underlying functional libraries. GIS technology will finally come into its own by becoming simply part of the fabric of software solutions.

Behind the Scenes of Virtual Reality

(GeoWorld, June 2000)

(return to top of Topic)

Over the past three decades, cutting-edge GIS has evolved from computer mapping to spatial database management, and more recently, to map analysis and modeling. The era of a sequestered GIS specialist has given way to mass marketed applications, such as MapQuest's geo-queries, On-Star's vehicular telematics and a multitude of other Internet-served maps.

The transition of GIS from an emerging industry to a fabric of society has radically changed traditional perspectives of map form, content and applications. Like a butterfly emerging from a cocoon, contemporary maps are almost indistinguishable from their predecessors. While underlying geographic principles remain intact, outward appearances of modern maps are dramatically different.

This evolution is most apparent in multimedia GIS. Traditional maps graphically portray map features and conditions as static, 2-D abstractions composed of pastel colors, shadings, line types and symbols. Modern maps, on the other hand, drapes spatial information on 3-D surfaces and provides interactive query of the mapped data that underlies the pictorial expression. Draped remote sensing imagery enables a user to pan, zoom and rotate the encapsulated a picture of actual conditions. Map features can be hyperlinked to text, tables, charts, audio, still images and even streaming video. Time series data can be sequenced to animate changes and enhance movement of in both time and space.

While these visualizations are dramatic, none of the multimedia GIS procedures shake the cartographic heritage of mapping as much as virtual reality. This topic was introduced in a feature article in GeoWorld a few years ago (Visualize Realistic Landscapes, GeoWorld, August, 1998, pages 42-47). This and the next couple of columns will go behind the scenes to better understand how 3-D renderings are constructed and investigate some of the approaches important considerations and impacts.

Since discovery of herbal dyes, the color pallet has been a dominant part of mapping. A traditional map of forest types, for example, associates various colors with different tree species—red for ponderosa pine, blue for Douglas fir, etc. Cross-hatching or other cartographic techniques can be used to indicate the relative density of trees within each forest polygon. A map's legend relates the abstract colors and symbols to a set of labels identifying the inventoried conditions. Both the map and the text description are designed to conjure up a vision of actual conditions and the resulting spatial patterns.

The map has long served as an abstract summary while the landscape artist's canvas has served as a more realistic rendering of a scene. With the advent of computer maps and virtual reality techniques the two perspectives are merging. In short, color pallets are being replaced by rendering pallets.

Like the artist's painting, complete objects are grouped into patterns rather than a homogenous color applied to large areas. Object types, size and density reflect actual conditions. A sense of depth is induced by plotting the objects in perspective. In effect, a virtual reality GIS “sees” the actual conditions of forest parcels through its forest inventory data— species type, mixture, age, height and stocking density for each parcel. A composite scene is formed by translating the data into realistic objects that characterize trees, houses, roads and other features then combined with suitable textures to typify sky, clouds, soil, brush and grasses.

Fundamental to the process is the ability to design realistic objects. An effective approach, termed geometric modeling, utilizes an interface (figure 1) similar to a 3-D computer-aided drafting system to construct individual scene elements. A series of sliders and buttons are used to set the size, shape, orientation and color of each element comprising an object. For example, a tree is built by specifying a series of levels representing the trunk, branches, and leaves. Level one forms the trunk that is interactively sized until the designer is satisfied with the representation. Level two establishes the pattern of the major branches. Subsequent levels identify secondary branching and eventually the leaves themselves.

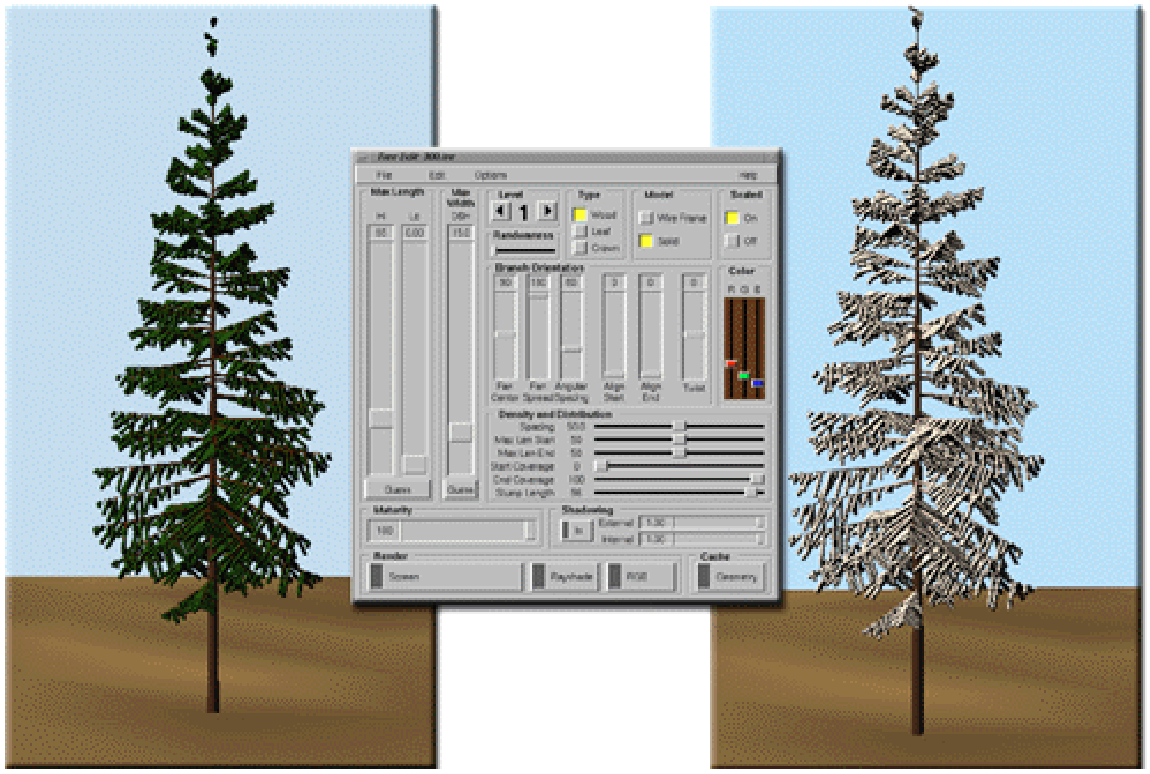


Figure 1. Designing tree objects.

The basic factors that define each level include 1) linear positioning, 2) angular positioning, 3) orientation, 4) sizing and 5) representation. Linear positioning determines how often and where branches occur. In fig. 1, the major branching occurs part way up the trunk and is fairly evenly spaced.

The angular positioning, sets how often branches occur around the trunk or branch to which it is attached. The branches at the third level in the figure form a fan instead of being equally distributed around the branch. Orientation refers to how the branches are tilting. Note that the lower branches droop down from the trunk, while the top branches are more skyward looking. The third-order branches tend show a similar drooping effect in the lower branches.

Sizing defines the length and taper a particular branch. In the figure, the lower branches are considerably smaller than the mid-level branches. Representation covers a lot of factors identifying how a branch will appear when it is displayed, such as its composition (a stick, leaf or textured crown), degree of randomness, and 24-bit RGB color. In figure 1, needle color and shading was changed for the tree on the right to simulate a light dusting of snow. Other effects such as fall colors, leaf-off for deciduous trees, disease dieback, and pest infestations can be simulated.

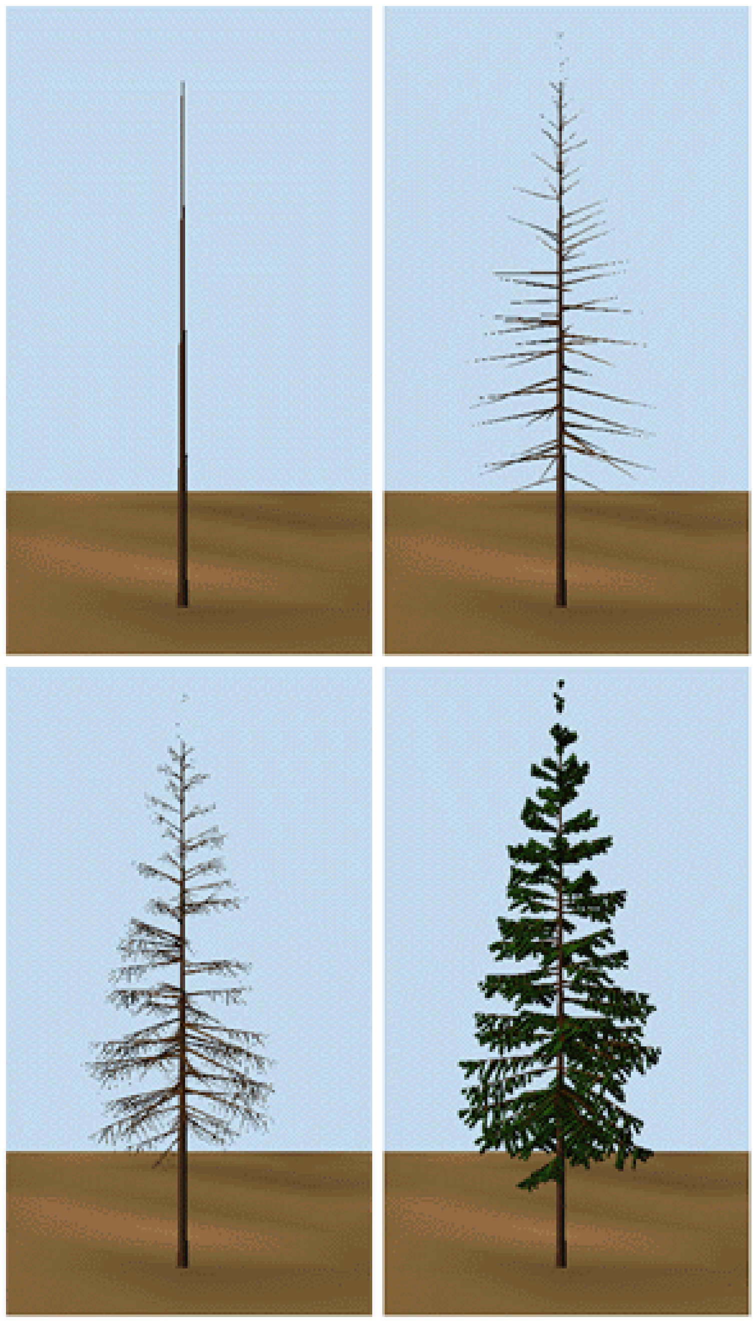


Figure 2. The inset on the left shows various branching patterns. The inset on the right depicts the sequencing of four branching levels.

Figure 2 illustrates some branching patterns and levels used to construct tree-objects. The tree designer interface at first might seem like overkill—sort of a glorified "painting by the numbers." While it's useful for the artistically-challenged, it is critical for effective 3-D rendering of virtual landscapes.

The mathematical expression of an object allows the computer to generate a series of "digital photographs" of a representative tree under a variety of look-angles and sun-lighting conditions.

The effect is similar to flying around the tree in a helicopter and taking pictures from different perspectives as the sun moves across the sky. The background of each bitmap is made transparent and the set is added to the library of trees. The result is a bunch of snapshots that are used to display a host of trees, bushes and shrubs under different viewing conditions.

The object-rendering process results in a "palette" of objects analogous to the color palette used in conventional GIS systems. When displaying a map, the GIS relates a palette number with information about a forest parcel stored in a database. In the case of 3-D rendering, however, the palette is composed of a multitude of tree-objects. The effect is like color-filling polygons, except realistic trees are poured onto the landscape based on the tree types, sizing and densities stored in the GIS. How this scene rendering process works is reserved for the next section.

____

Author's Note: *the Tree designer module of the Virtual Forest software package by Pacific Meridian Resources was used for the figures in this column. See http://www.innovativegis.com/products/vforest/ for more examples and discussion.*

How to Rapidly Construct a Virtual Scene

(GeoWorld, July 2000)

(return to top of Topic)

The previous section described how 3-dimensional objects, such as trees, are built for use in generating realistic landscape renderings. The drafting process uses an interface that enables a user to interactively adjust the trunk's size and shape then add branches and leaves with various angles and positioning. The graphic result is similar to an artist's rendering of an individual tree.

The digital representation, however, is radically different. Because it is a mathematically defined object, the computer can generate a series of "digital photographs" of the tree under a variety of look-angles and sun-lighting conditions. The effect is similar to flying around the tree in a helicopter and taking pictures from different perspectives as the sun moves across the sky.

The background of each of these snapshots is made transparent and the set is added to a vast library of tree symbols. The result is a set of pictures that are used to display a host of trees, bushes and shrubs under different viewing conditions. A virtual reality scene of a landscape is constructed by pasting thousands of these objects in accordance with forest inventory data stored in a GIS.

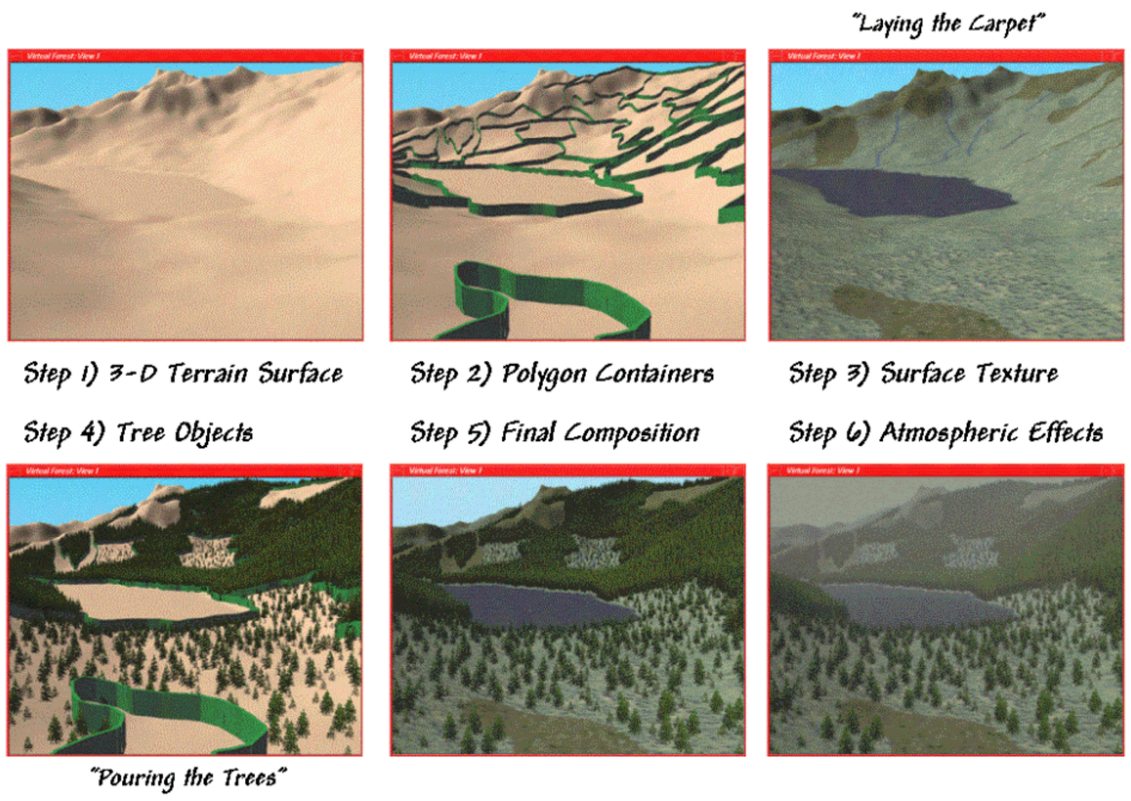

"Laying the Carpet"
Step 1) 3-D Terrain Surface
Step 2) Polygon Containers
Step 3) Surface Texture
Step 4) Tree Objects
Step 5) Final Composition
Step 6) Atmospheric Effects
"Pouring the Trees"

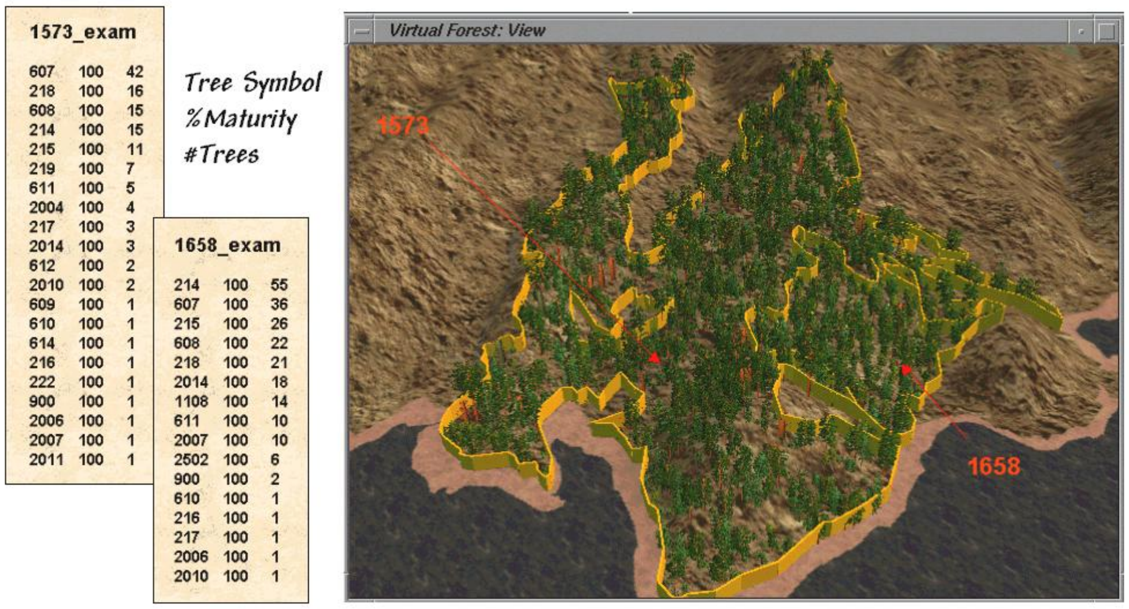
Figure 1. Basic steps in constructing a virtual reality scene.

There are six steps in constructing a fully rendered scene (see Figure 1). A digital terrain surface provides the lay of the landscape. The GIS establishes the forest stand boundaries as geo-registered polygons with attribute data describing stand make-up and condition.

Forest floor conditions are represented by "texture maps" that add color and grain to the terrain surface. Once the configuration and texturing is complete, the tree objects are "poured" onto the surface for the final composition with fog/haze added as appropriate.

The link between the GIS data and the graphic software is critical. For each polygon, the data identifies the types of trees present, their relative occurrence (termed stocking density) and maturity (age, height). In a mixed stand, such as spruce, fir and interspersed aspen, several tree symbols will be used. Tree stocking identifies the number of trees per acre for each of the species present. This information is used to determine the number tree objects to "plant" and cross-link to the appropriate tree symbols in 3-D tree object library. The relative positioning of the polygon on the terrain surface with respect to the viewpoint determines which snapshot of the tree provides the best view and sun angle representation.

Finally, information on percent maturity establishes the baseline height of the tree. In a detailed tree library several different tree objects are generated to represent the continuum from immature, mature and old growth forms. Figure 2 shows the tree exam files for two polygons identified in the adjacent graphic. The first column of values identifies the tree symbol (library ID#). Polygon 1573 has 21 distinct tree types including snags (dead trees). Polygon 1658 is much smaller and only contains 16 different types. The second column indicates the percent maturity while the third defines the number of trees. These data shown are for an extremely detailed U.S. Forest Service research area in Colorado. Most operational landscape visualizations, however, have only one or just a few tree types represented per polygon.



1573_exam

607	100	42
218	100	16
608	100	15
214	100	15
215	100	11
219	100	7
611	100	5
2004	100	4
217	100	3
2014	100	3
612	100	2
2010	100	2
609	100	1
610	100	1
614	100	1
216	100	1
222	100	1
900	100	1
2006	100	1
2007	100	1
2011	100	1

1658_exam

214	100	55
607	100	36
215	100	26
608	100	22
218	100	21
2014	100	18
1108	100	14
611	100	10
2007	100	10
2502	100	6
900	100	2
610	100	1
216	100	1
217	100	1
2006	100	1
2010	100	1

Figure 2. Forest inventory data establishes tree types, stocking density and maturity.

Once the appropriate tree symbol and number of trees are identified the computer can begin “planting” them. This step involves determining a specific location within the polygon and sizing the snapshot based on the tree’s distance from the viewpoint. Most often trees are randomly placed however clumping and compaction factors can be used to create clustered patterns if appropriate.

Tree sizing is similar pasting and resizing an image in a word document. The base of the tree symbol is positioned at the specific location then enlarged or reduced depending on how far the tree is from the viewing position. Figure 3 shows a series of resized tree symbols “planted” along a slope—big trees in front and progressively smaller ones in the distance.

The process of rendering a scene is surprisingly similar to that of landscape artist. The terrain is painted and landscape features added. In the artist’s world it can take hours or days to paint a scene. In virtual reality the process is completed in a minute or two as hundreds of trees are selected, positioned and resized each second.

Since each tree is embedded on a transparent canvas they obscure what is behind them—textured terrain and/or other trees, depending on forest stand and viewing conditions. Terrain locations that are outside of the viewing window or hidden behind ridges are simply ignored. The multitude of issues and extended considerations surrounding virtual reality’s expression of GIS data, however, cannot be ignored. That discussion is reserved for the next section.

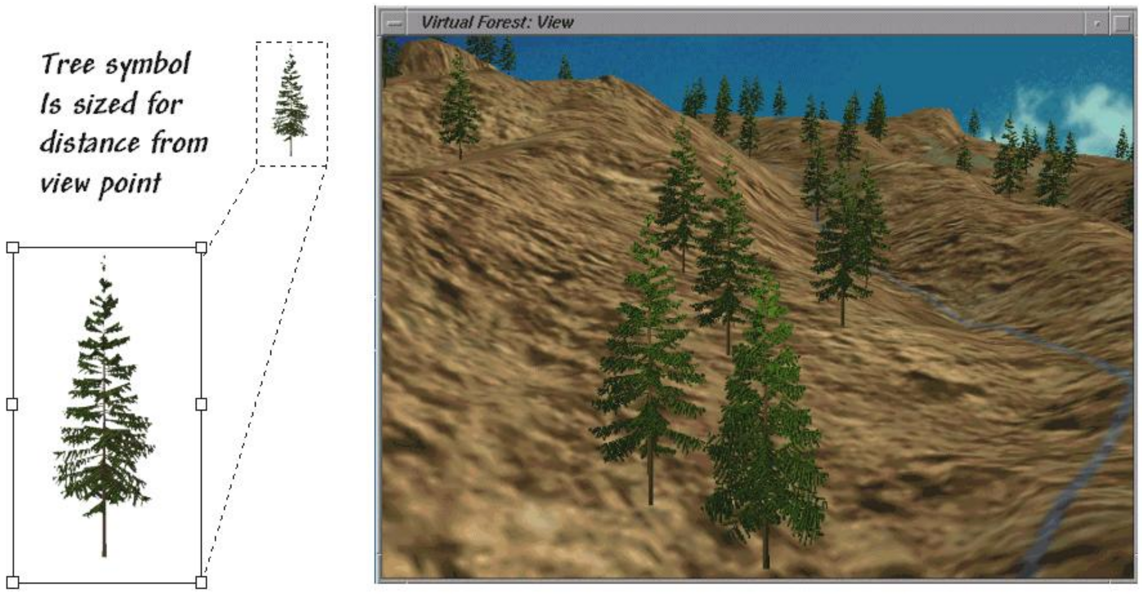


Figure 3. Tree symbols are "planted" then sized depending on their distance from the viewpoint.

***Author's Note**: the Landscape Viewer module of the Virtual Forest software package by Pacific Meridian Resources was used for the figures in this column. See http://www.innovativegis.com/products/vforest/ for more examples and discussion.*

How to Represent Changes in a Virtual Forest

(GeoWorld, August 2000)

(return to top of Topic)

The previous section described the steps in rendering a virtual landscape. The process begins with a 3D drafting program used to construct mathematical representations of individual scene elements similar to a painter's sketches of the different tree types that occur within an area. The tree library is linked to GIS data describing the composition of each forest parcel. These data are used to position the polygon on the terrain, select the proper understory texture ("laying the carpet") and paste the appropriate types and number of trees within each polygon ("pouring the trees").

The result is a strikingly lifelike rendering of the landscape instead of a traditional map. While maps use colors and abstract symbols to represent forest conditions, the virtual forest uses realistic scene elements to reproduce the composition and structure of the forest inventory data. This lifelike 3D characterization of spatial conditions extends the boundaries of mapping from dry and often confusing drawings to more familiar graphical perspectives.

The baseline rendering for a data set can be modified to reflect changes on the landscape. For example, the top two inserts in figure 1 depict a natural thinning and succession after a severe insect infestation. The winter effects were introduced by rendering with a snow texture and an atmospheric haze.

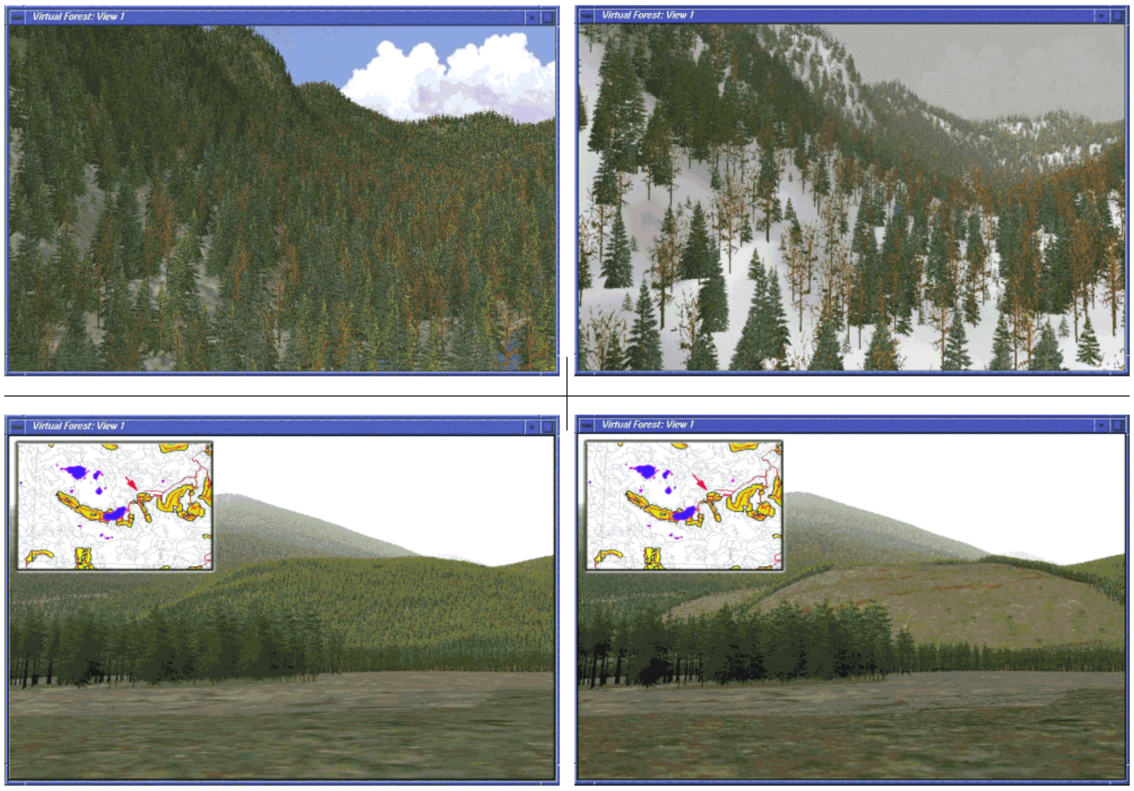


Figure 1. Changes in the landscape can be visualized by modifying the forest inventory data.

The lower pair of inserts shows the "before and after" views of a proposed harvest block. Note the linear texture features in the clearcut that identify the logging road. Alternative harvest plans can be rendered and their relative visual impact assessed. In addition, a temporal sequence can be generated that tracks the 'green-up" through forest growth models as a replanted parcel grows. In a sense, the baseline GIS information shows you "what is," while the rendering of the output from a simulation model shows you "what could be."

While GIS modeling can walk through time, movement to different viewpoints provides a walk through the landscape. The viewer position can be easily changed to generate views from a set of locations, such as sensitive viewpoints along a road or trail. Figure 2 shows the construction of a 'fly-by" movie. The helicopter flight path at 200 meters above the terrain was digitized then resampled every twenty meters (large red dots in the figure). A full 3D rendering was made for each of the viewpoints (nearly 900 in all) and, when viewed at 30 frames per second, forms a twenty-eight second flight through the GIS database (see author's note).

Admittedly, real-time "fly-bys" of GIS databases are a bit futuristic. With each scene requiring three to four minutes to fully render on a PC-level computer, a 30 second movie requires about

45 hours of processing time. The Lucas Films machines would reduce the time to a few minutes but it will take a few years to get that processing power on most desktops. In the interim, the transition from traditional maps to fully rendered scenes is operationally constrained to a few vanguard software systems. There are several concerns about converting GIS data into realistic landscape renderings. Tree placement is critical. Recall that "stocking" (#trees per acre) is the forest inventory statistic used to determine the number of trees to paste within a polygon. While this value indicates the overall density it assumes the trees are randomly distributed in geographic space.

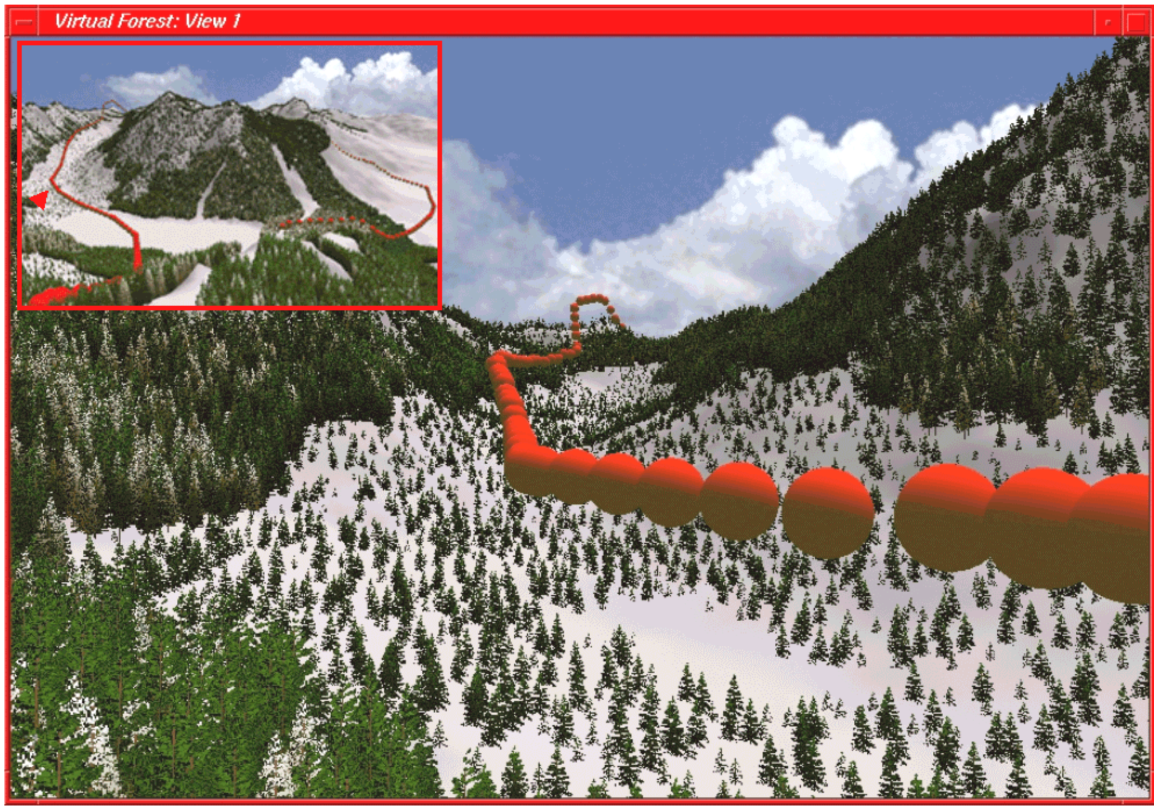


Figure 2. A "fly-by" movie is constructed by generating a sequence of renderings then viewing them in rapid succession (click here to view the fly-by).

While trees off in the distance form a modeled texture, placement differences of a couple of feet for trees in the foreground can significantly alter a scene. For key viewpoints GPS positioning of specific trees within a few feet of the viewer is required. Also, in sequential rendering the trees are statistically placed for the first scene then that "tree map" is used for all of the additional scenes. Many species, such as aspen, tend to group and statistical methods are needed to account for "clumping" (number of seed trees) and "compaction" (distance function from seed tree).

Realistic trees, proper placement, appropriate under-story textures and shaded relief combine to produce strikingly real snapshots of a landscape. In robust, forest inventory data the rendering closely matches reality. However, equally striking results can be generated from limited data. For example, the "green" portions on topographic maps indicate forested areas, but offer no information about species mixture, age/maturity or stocking. Within a GIS, a "best guess" (er… expert opinion) can be substituted for the field data and one would be hard-pressed to tell the differences in rendered scenes.

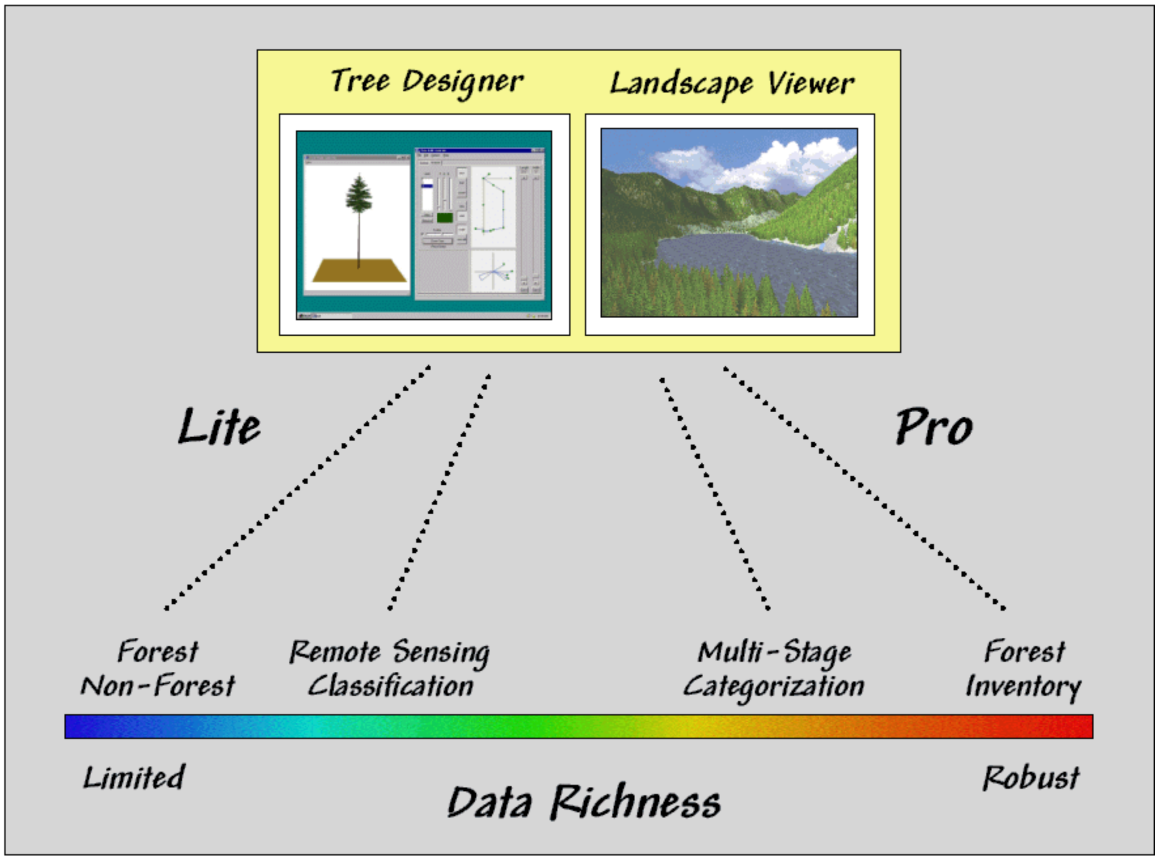


Figure 3. Strikingly real snapshots of forest data can be generated from either limited or robust GIS data.

That brings up an important point as map display evolves to virtual reality—how accurate is the portrayal? Our cartographic legacy has wrestled with spatial and thematic accuracy, but "rendering fidelity" is an entirely new concept (figure 32-8). Since you can't tell by looking, standards must be developed and integrated with the metadata accompanying a rendering. Interactive links between the underlying data and the snapshot are needed. Without these safeguards, it's "viewer beware" and opens a whole new avenue for lying with maps.

While Michael Creighton's emersion into a virtual representation of a database (the novel Disclosure) might be decades off, virtual renderings of GIS data is a quantum leap forward. The pastel colors and abstract symbols of traditional maps are becoming endangered cartographic procedures. When your grandchild conjures up a 3D landscape with real-time navigation on a wrist-PC, you'll fondly recall the bumpy transition from our paper-map paradigm.

***Author's Note**: the Virtual Forest software package by Pacific Meridian Resources was used for the figures in this column. The fly-by in http://www.innovativegis.com/products/vforest/, select "Flybys" to access the simulated helicopter flight described as well as numerous other examples of 3D rendering.*

Capture "Where and When" on Video-based GIS

(GeoWorld, September 2000)

(return to top of Topic)

The past three columns described procedures for translating GIS data into virtual renderings of a landscape. While traditional maps generalize landscape features as abstract symbols and patterns, a virtual forest portrays mapped data more like a painting. Instead of pastel colors and crosshatching, realistic objects, such as trees, rocks and water, are appropriately placed on a shaded relief surface. The effect is a map that rivals a photographic snapshot of the conditions recorded in the GIS database.

An alternative is to populate a GIS database with actual snapshots and streaming video that are linked to their map location. Multimedia GIS provides a connection between a map and field-collected images, audio and tabular summaries. This emerging field is poised to recast our perspective of what maps are and how they can be used.

Video mapping is an exciting part of the revolution in visualization of mapped data. It records GPS signals directly on videotape shot in the field. When the tape is played to a computer it links these data to a digital map for easy access and review. The result is an extension of field data collection to field experience collection through geo-registered visual and audio records.

With video mapping, the construction of a multimedia GIS no longer involves tedious and time-consuming procedures for encoding spatial coordinates of the imagery. The entire process, from field video collection to map indexing and Web page publishing consists of three simple steps—Recording, Indexing and Review.

During the Recording Step, video mapping encodes GPS coordinates directly onto the videotape (see figure 1). The video mapping unit contains a standard GPS board that monitors the satellite signals, converts this information into a data stream consisting of longitude (X), latitude (Y), actual time/date, and a variety of supporting data. These data are output as standard NMEA formatted data. A second circuit board in the unit converts this digital information into an audio signal in a manner similar to a modem for phone line access to the Internet.

In turn, the acoustic signals are sent to one of the audio channels through the microphone input connector on the video camera. The result is recording the GPS position on the videotape every second that the camera is on.

The direct recording of "where and when" on the tape greatly facilitates field data collection—as long as there is GPS reception, the information is automatically recorded on the same medium (videotape) as the imagery. In addition, any audio notes you might make are captured on the same tape. Voice recognition software can convert the notes into text, or if a specific voice commands are used, the information can be converted into a database record.

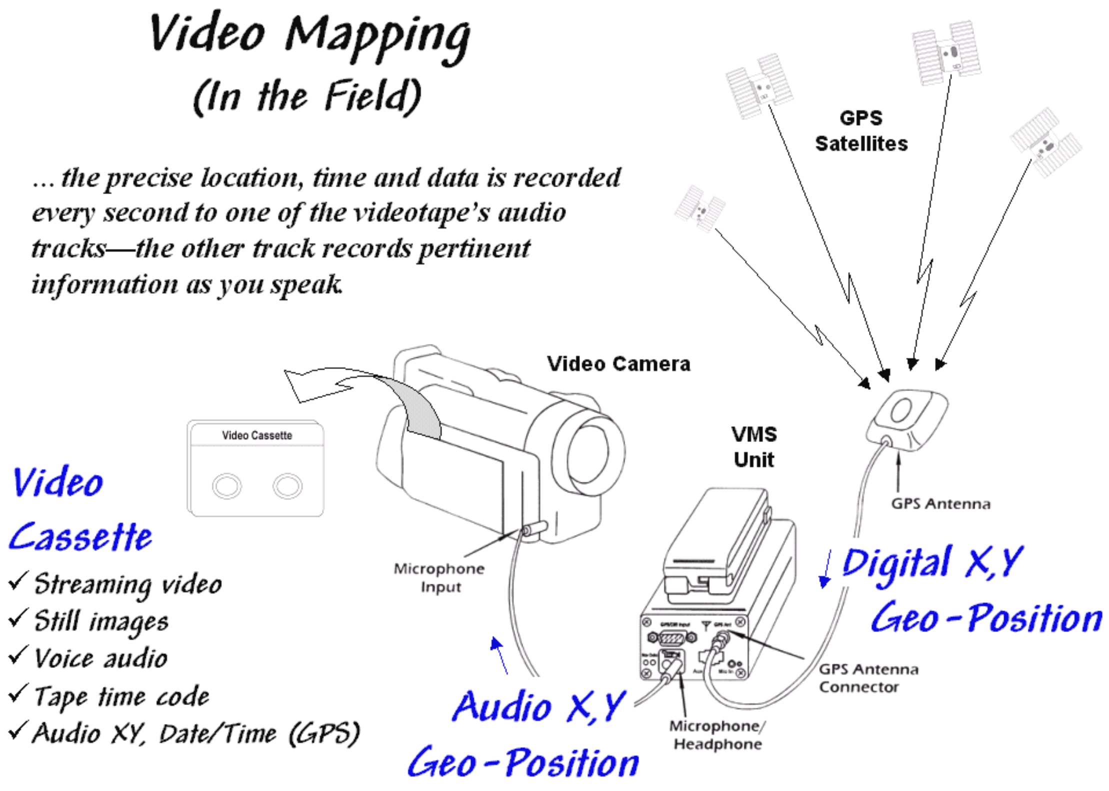


Figure 1. Video Mapping in the Field. As video is recorded, the precise location, time, and date are recorded every second to one of the videotape's audio tracks. The other track records pertinent information as you speak.

Most contemporary video cameras have a switch between photo and movie mode. In movie mode, streaming video is recorded at 30 frames per second. In photo mode, the camera acts like a still camera and "freezes" the frame for several seconds as it records the image to videotape. In this mode, a one-hour videotape can record over 500 digital pictures. In both photo and movie modes the one-second GPS "data stamp" provides ample positioning information for most applications… every 88 feet at 60 mph in a car or every 3 feet while strolling at 2 mph.

The Indexing Step involves connecting the video mapping unit to a computer and playing the video (see figure 2). In this configuration, the audio cord is switched to the Headphone Output connector and a second cable is connected to the Lan C connector on the camera. The connector provides information on tape position (footage on older cameras and time code on newer ones) used in indexing and playback control of the tape similar to those on a standard VCR.

As the videotape is played, the audio X,Y and time code information is sent to the video mapping unit where it is converted to digital data and sent to the serial port on the computer. If a headset was used in the field, the voice recording on the second audio channel is transferred as well.

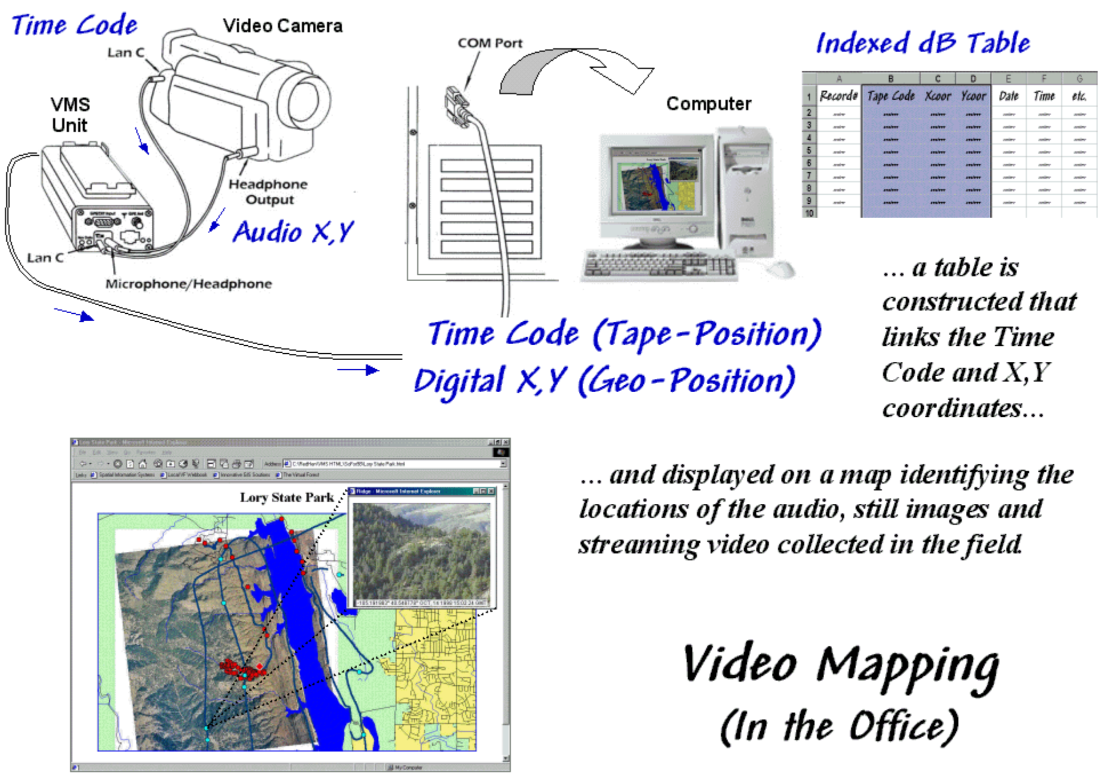

Time Code
Video Camera
Lan C
VMS Unit
Headphone Output
Audio X,Y
Lan C
Microphone/Headphone
COM Port
Computer
Indexed dB Table
Recorde
Tape Code
Xcoor
Ycoor
Date
Time
etc.
... a table is constructed that links the Time Code and X,Y coordinates...
Time Code (Tape-Position)
Digital X,Y (Geo-Position)
Lory State Park
... and displayed on a map identifying the locations of the audio, still images and streaming video collected in the field
Video Mapping
(In the Office)

Figure 2. Indexing the Videotape. The video, audio notes and GPS information is used to construct a multimedia map of the precise position and date/time of the video footage providing direct retrieval of text, data, audio, image and video by simply clicking on the map.

For indexing there are five types of information available—streaming video (movie mode), still images (photo mode), voice audio (headset), tape time code (tape position), and GPS data (camera geo-position plus date/time and information on satellite lock)—all automatically registered on the videotape whenever the camera is recording.

Video mapping software records the GPS information from the videotape and constructs a database that connections GPS locations with videotape time codes. The computer generates an interactive map of everywhere that video was recorded. Special map features can be marked on the map and information entered about them or still images captured, while the map is being indexed. The map can be used for review, or exported in a MapInfo or ArcView compatible format.

The Review Step uses the indexed database to access audio and video information on the tape. The hardware configuration is the same as for indexing (audio, Lan C and serial cables). Clicking on any indexed location retrieves its GPS data and associated video. Player controls similar to a VCR are sent to the video camera to play back the video recordings by noting its time code and causing the tape to move to that location.

Map features can start applications, open files and display images. The software works with video capture cards to create still images and video clips you can link to map features, giving maximum flexibility in choosing a data review method. In many applications the completed multimedia map is exported as an HTML file for viewing with any browser or over the Internet. The map features can contain any or all five of the basic information types:

Text — interpreted from audio as .DOC file
Data — interpreted from audio as .DAT, .XLS or .DBF file
Audio — captured as .WAV file (about 100KB per 5 seconds)
Image — captured as .JPG file (about 50KB per image)
Video — captured as .AVI file (about 1MB per 5 seconds)

The next section explores the procedures for constructing finished maps and describes several applications of video mapping. In the interim, you might checkout the links to some online examples (see author's notes).

____

Author's Note: *the information contained in this column is intended to describe the conceptual approach, considerations and practical applications of video mapping. Several online demonstrations of this emerging technology in a variety of applications are available at http://www.redhensystems.com. For information on the information on the VMS 200^{TM} Video Mapping System contact Red Hen Systems, Inc, 2310 East Prospect Road, Suit A, Fort Collins, USA 80525, Phone: (800) 237-4182, Email: info@redhensystems.com, Web site: http://www.redhensystems.com/.*

Video Mapping Brings Maps to Life

(GeoWorld, October 2000)

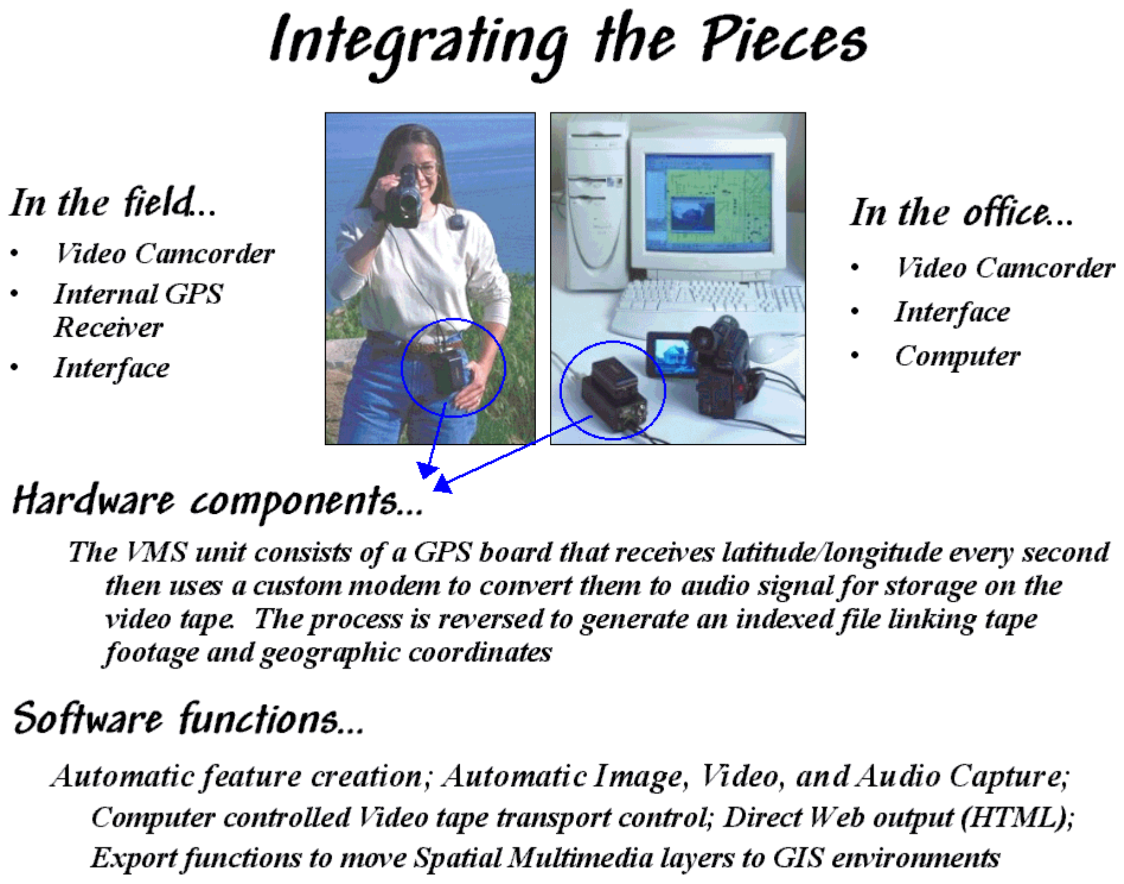
(**return to top of Topic**)

As detailed in last month's column, video mapping enables anyone with a computer and video camera to easily create their own interactive video maps. The integration of computers, video camera, and GPS marks a technological milestone that finally makes GIS multimedia a practical reality. A user can inexpensively add real time, geographically indexed images, audio and video clips to ongoing data collection activities. Applications abound in natural resources, precision farming, business, government, science, recreation, and any other endeavor that needs a visual GIS capability.

For example, corridor mapping of oil and gas pipelines, transmission towers, right of ways and the like provide images of actual conditions not normally part of traditional maps. In law enforcement, video mapping can be used from reconnaissance to traffic safety to forensics. Agriculture applications include crop scouting, weed/pest management, verification of yield maps, and "as-applied" mapping. Geo-business uses range from conveying neighborhood character, to insurance reporting to web page development.

By coupling audio/visual information to other GIS data, anyone can see first-hand conditions that add reality to tabular field data. In disaster assessment, the ability to click on several indexed locations and see and hear the extent of damage can convey much more information than simple statistics. In forestry, resource managers who were not involved in field data collection can review conditions not easily quantified, such as under-story characteristics and wildlife habitat potential. In short, video mapping provides "the missing link" in GIS, enabling incorporation of visual layers into any geo-referenced data set.

Data is collected without a computer in the field or cumbersome additional equipment. Figure 32-11 shows the VMS^{TM} unit by Red Hen Systems (see author's note) that weighs less than a pound and is connected to the video camera via a small microphone cable. The GPS antenna is easily attached to a cap, hardhat or shoulder strap of a carrying case. Optional hardware includes an electronic compass to record camera direction and a laser rangefinder to electronically measure the distance to objects at the optical center of the camera's view. A multispectral unit for simultaneously recording up to four wavelength bands is under development.



Integrating the Pieces

In the field...

- *Video Camcorder*
- *Internal GPS Receiver*
- *Interface*

In the office...

- *Video Camcorder*
- *Interface*
- *Computer*

Hardware components...

The VMS unit consists of a GPS board that receives latitude/longitude every second then uses a custom modem to convert them to audio signal for storage on the video tape. The process is reversed to generate an indexed file linking tape footage and geographic coordinates

Software functions...

Automatic feature creation; Automatic Image, Video, and Audio Capture; Computer controlled Video tape transport control; Direct Web output (HTML); Export functions to move Spatial Multimedia layers to GIS environments

Figure 1. Video Mapping Hardware. The VMS 200 unit enables recording and processing of GPS signals and video time codes.

The office configuration consists of a video camera, VMS unit, notebook or desktop computer, and mapping software. The software generates a map automatically from the data recorded on the videotape. Once a map is created, it can be personalized by placing special feature points that relate to specific locations. These points are automatically or manually linked to still images, video clips, sound files, documents, data sets, or other actions that are recalled at the touch of a button. A voice recognition package is under development that will create free-form text and data-form entry. The mapping software also is compatible with emerging GPS-based still cameras.

While a map is being created, or at a later time, a user can mark special locations with a mouse-click to "capture" still images, streaming video or audio files. The “fire wire” port on many of the newer computers makes capturing multimedia file a snap. Once captured, a simple click on the map feature accesses the images, associated files, or video playback beginning at that location.

Sharing or incorporating information is easy because the video maps are compatible with most popular GIS programs. An HTML export function provides an extremely useful data delivery device for service providers, project managers, or others who need to make their imagery generally available. By transforming the maps and associated data to a Web page, time-dependent information can be "served to the Internet" and available to thousands of people within a few minutes of collection.

Differential post-processing is another important software addition. The post-processing software (EZdiff^{TM}) takes base station correction data from the Internet and performs a calculation against the video mapped GPS data for the same time period, then outputs a data file containing the corrected, highly accurate points. It mathematically corrects the autonomous ("normal") GPS signals from an error of about 10 meters to positional accuracy of about 1 to 2 meters.

Figure 2 shows an example of the video mapping software. The dark blue line on map identifies the route of an ultralite (a hang-glider with an engine). Actually the line is composed of a series of dots— one for each second the video camera was recording. Clicking anywhere on the line will cause the camera, or VCR, to automatically fast forward/reverse to the location and begin playing the video.

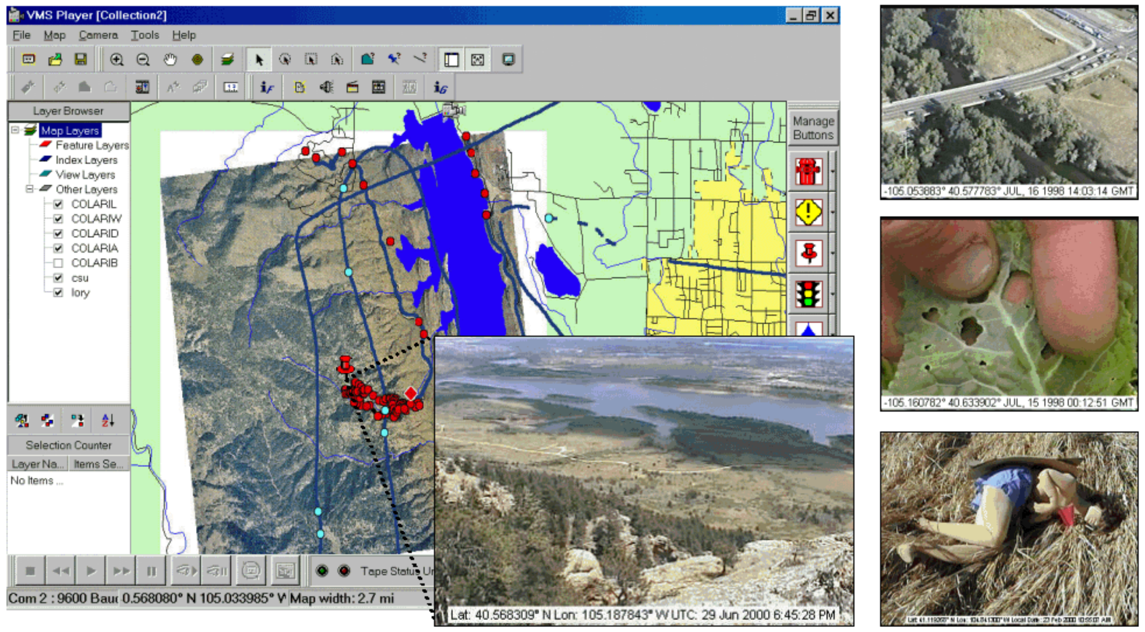

VMS Player [Collection2]
File Map Camera Tools Help
Layer Browser
Map Layers
Feature Layers
Index Layers
View Layers
Other Layers
COLARIL
COLARIW
COLARID
COLARIA
COLARIB
csu
lory
Manage Buttons
Selection Counter
Layer Na... Items Se...
No Items ...
Tape Status
Com 2 : 9600 Baud 0.568080° N 105.033985° W Map width: 2.7 mi
Lat: 40.568309° N Lon: 105.187843° W UTC: 29 Jun 2000 6:45:28 PM
-105.053883° 40.577783° JUL, 16 1998 14:03:14 GMT
-105.160782° 40.633902° JUL, 15 1998 00:12:51 GMT

Figure 2. Video Mapping Software. Specialized software builds a linked database and provides numerous features for accessing the data, customizing map layout and exporting to a variety of formats.

The light blue and red dots in the figure are feature locations where still images, audio tracks and video clips were captured to the hard disk. The larger inset is a view of the lake and city from the summit of a hiking trail. The adjacent red dots are a series of similar images taken along the trail. When a video camera is set in photo mode, a one-hour videotape contains nearly 600 exposures— no film, processing or printing required. In addition, the automatic assignment of GPS time and position makes filing and retrieving a trivial task—no more file cabinets, manila folders and with photos taped to reports.

The top captured image on the right side of the figure shows a photo taken from an ultralite inventory of bridges along a major highway. The middle image is a field photo of cabbage loper damage in a farmer's field. The bottom image is of a dummy in a training course for police officers. The web pages for these and other applications are online for better understanding of video mapping capabilities (see author's notes).

For centuries, maps have been abstractions of reality that use inked lines, symbols and shadings to depict the location of physical features and landscape conditions. Multimedia GIS, and video mapping in particular, provide an easy means of linking additional audio/visual information to map features. Special equipment, field procedures and office processing is minimal and easy to learn. The ability to access stored images, video, audio, text, and data at the click of a mouse radically changes our paradigm of a map—from abstract drawings to sights, sounds and summaries of actual conditions. Video mapping truly "makes maps come alive."

____

Author's Note: *the information contained in this column is intended to describe the conceptual approach, considerations and practical applications of video mapping. Several online demonstrations of this emerging technology in a variety of applications are available at http://www.redhensystems.com. For information on the information on the VMS 200^{TM} Video Mapping System contact Red Hen Systems, Inc, 2310 East Prospect Road, Suit A, Fort Collins, USA 80525, Phone: (800) 237-4182, Email: info@redhensystems.com, Web site: http://www.redhensystems.com/.*

(return to top of Topic)

(Back to the Table of Contents)