TerraVision II: Visualizing Massive Terrain Databases in VRML

Martin Reddy, Yvan Leclerc, Lee Iverson, and Nat Bletter
SRI International

To disseminate 3D maps and spatial data over the Web, we designed massive terrain data sets accessible through either a VRML browser or the customized TerraVision II browser.

¶1 Researchers have increasingly turned to Virtual Reality Modeling Language (VRML) to represent geographic information. In VRML's early days, the result was a few toy examples that did not scale well, such as coarse, single-resolution elevation grids. Today, VRML is drawing more serious interest from researchers across the spectrum, including geographers, cartographers, geologists, and computer scientists, as the sidebar "Related Work" describes. As Theresa-Marie Rhyne noted, geographic information system (GIS) and scientific visualization tools have begun to expand into each other's domains,[1] and VRML offers cartographers and geographers the potential to disseminate 3D maps and spatial data over the World Wide Web. However, to date we have not seen useful large-scale VRML geographic databases.

¶2 We aim to enable visualization of near photorealistic 3D models of terrain that can be on the order of hundreds of gigabytes. This might include different types of terrain imagery for particular regions, as well as site models and auxiliary information for ground features.

¶3 The following scenario indicates the capabilities required. Say a user wants to find a particular building in a particular city. Her journey begins with a 3D model of the earth viewed from space. This model is texture-mapped with satellite imagery of 100 kilometers resolution—that is, each pixel in the texture map represents a region on the planet's surface covering 100 km^2. To find the city, the user first rotates the earth to view the

Related Work

Currently, interesting and significant work addresses the problem of representing geographic data in VRML. In the earth sciences, Kate Moore described the work of the Virtual Field Course (VFC) project,[1] which is developing software tools to familiarize students with fieldwork locations and aid data collection and analysis. The VFC project uses VRML and Java to provide interactive 2D and 3D views of geo-referenced data to enhance students' cognition of the real environment.

The US Naval Postgraduate School is currently working on a project to develop a 3D model of the Monterey Bay National Marine Sanctuary. They aim to create a VRML representation of the sanctuary based on raw bathymetry (below sea level) data for a 2.5 × 2.5 degree region of the bay. Their representation uses multiresolution techniques to deliver these large data amounts over a 28K modem connection.

Michael Abernathy and Sam Shaw described their work using VRML to visualize the course for a 197-mile relay race through the San Francisco Bay Area.[2] They did this using standard US Geological Survey (USGS) 7.5 arc min digital elevation models (DEMs) for the terrain geometry with geo-referenced satellite imagery draped over the terrain. Their system also used Global Positioning System (GPS) input to create a line segment showing the race's course over the VRML terrain.

References

1. K. Moore, "Interactive Virtual Environments for Fieldwork," *British Cartographic Society Annual Symp.*, 1997; available at http://www.geog.le.ac.uk/mek/VirtEnv.htm.
2. M. Abernathy and S. Shaw, "Integrating Geographic Information in VRML Models," Proc. Third Symp. VRML, ACM New York, 1998, pp. 107-114.

target region in more detail. As she zooms into the region, higher resolution data, such as elevation and imagery, are progressively downloaded and displayed until she is "flying" over mountains with imagery down to one-meter resolution. Over certain parts of the terrain, alternative imageries are available, such as aerial photographs; the user can select any image to view on top of the terrain geometry. As she approaches a built-up area, 3D models of buildings come into view. When the user clicks on a building, information about it is displayed in a separate frame on the browser. Using this method, the user locates the target building. Throughout the navigation, the user's location is displayed via an active map interface that provides a context for the landscape being viewed.

¶4 In setting out to achieve such capabilities, we identified four principal design criteria:

¶5 ■ *Scalability*. Our design must scale to very large data sets. Commonly, a geographic data set consists of many millions of polygons and many gigabytes of imagery.

¶6 ■ *Composability*. Our data representation must allow the introduction of multiple types of geo-referenced data, including additional imagery, site models, cultural features, and annotations. It also must let the user switch between these on demand.

¶7 ■ *Efficiency*. Users must be able to navigate the VRML structures easily and efficiently using a standard VRML browser or a customized browser that further increases browsing efficiency.

¶8 ■ *Data interchange*. We must develop generic data representations for geo-referenced data in VRML. This will let other geographic data providers produce data using the same representation.

¶9 Guided by these requirements, we implemented this functionality in a standard VRML browser for downloading data over the World Wide Web. We also developed a custom terrain visualization package called TerraVision II that can browse these VRML data structures. Although not required to view the content, TerraVision II lets the user perform specialized browser-level optimizations that offer increased efficiency and seamless interaction with the terrain data.

¶10 We designed our framework to simplify terrain data maintenance and to let users dynamically select particular sets of geo-referenced data. Our implementation uses Java scripting to extend VRML's base functionality and the External Authoring Interface to offer application-specific management of the virtual geographic environment.

¶11 To help develop standard techniques for solving geographical representation problems in VRML, coauthor Lee Iverson formed and currently chairs the GeoVRML group, an official Working Group of the Web3D Consortium (http://www.ai.sri.com/geovrml). As a service to the VRML community and the GeoVRML effort, we have made freely available both the TerraVision II browser and all of the tools we developed for generating VRML terrain data sets. This includes the source code to all of our custom VRML nodes and the tsmApi library that we use to generate the VRML data sets. The library is described in the sidebar "The tsmApi Library" and is available along with other materials and several example VRML data sets at http://www.ai.sri.com/TerraVision.

Multiresolution terrain techniques

¶12 Terrain models are typically massive. For example, the US Geological Survey produces digital elevation models (DEMs) that contain a regular grid of $1{,}201 \times 1{,}201$ elevation values for a 1-degree area of the earth's surface. Producing a simple polygonal representation of a single DEM creates a model with more than 1.4 million polygons. The time required to download and render such a model would prohibit any real-time interaction using the current generation of VRML browsers. It therefore becomes essential to manage level of detail (LOD).

¶13 LOD techniques change a model's complexity based on some selection criteria, such as distance from the viewpoint or projected screen size. The basic premise for these criteria is that any distant detail that projects to less than a single pixel on the screen will not generally be visible. To implement this, we need a mechanism to simplify a data set's geometry and imagery.

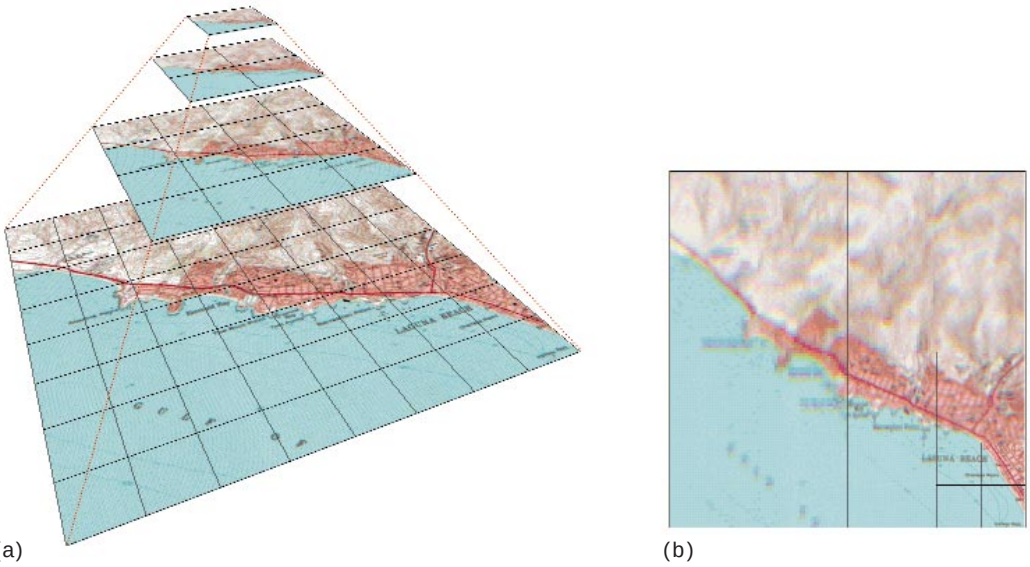
¶14 Several polygon simplification algorithms work well for terrain. However, many of these are view-independent techniques that force the same degree of simplification across the entire terrain.[2,3] These are

The tsmApi Library

The Tile Set Manager Application Program Interface (tsmApi) is a freely available C library from SRI International. The library offers functions for reading, writing, and generating terrain data used by TerraVision II, including functions for generating VRML versions of the terrain data using the representations we describe in the article.

Using the tsmApi library, users can create their own VRML geographic data sets from several supported input formats such as raw imagery, Portable Bitmap (PBM) images, and Land Analysis System (LAS) bitmaps. The library also includes SRI's fully re-entrant VRML 97 parser, which can be used to parse VRML 97 files efficiently into memory and to write these structures back out to a VRML 97 file. Other functions perform transformations between various geographic coordinate systems, such as Univeral Transverse Mercator (UTM), geodetic (latitude/longitude), and earth-fixed geocentric, based on code from the US National Imagery and Mapping Agency's Nimamuse product.

Precompiled tsmApi distributions are available for Irix, Solaris, Linux, and other platforms. In addition, the full C source code is available from the tsmApi home page, which also includes full API documentation, tutorials, format specifications, and example source code. The tsmApi Web page is at http://www.ai.sri.com/tsmApi.



(a) (b)

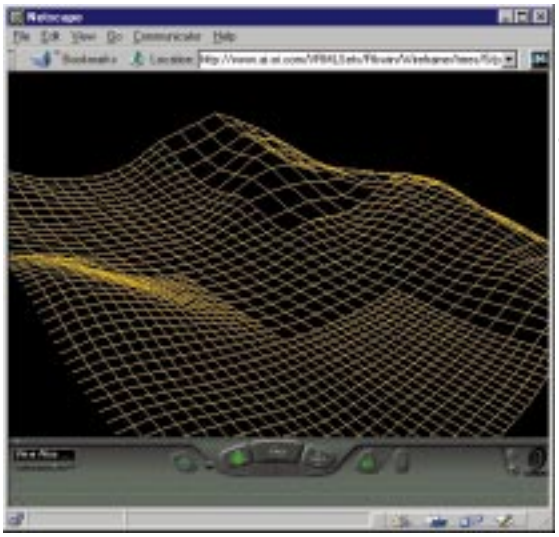
1 An example image pyramid showing (a) four different resolutions of an original image, where each level is segmented into 128 × 128 pixel tiles, and (b) how this structure can be used to alter the image resolution in different regions.

inappropriate for our application because switching to the highest resolution still involves loading every point of the original data set. Instead, we require a view-dependent technique that lets us vary the degree of simplification with respect to the current viewpoint.[4,5] This is often done using a hierarchical data structure, such as a quad-tree. Further, the LOD algorithm must not require access to the entire high-resolution version of the data set, as that would limit us to viewing only data sets that can fit on the user's local storage system. Given these requirements, a tiled, pyramid representation best suits our needs.[6-8]

¶15 A pyramid is a multiresolution hierarchy for a data set. For example, if the original image is 1024 × 1024 pixels, then the pyramid might contain the original image along with down-sampled versions at resolutions of 512 × 512 pixels, 256 × 256 pixels, 128 × 128 pixels, and so on. As Figure 1a shows, each pyramid image is then segmented into rectangular tiles, where all tiles have the same pixel dimensions. A tile at a given pyramid level will thus map onto four tiles on the next higher level; that is, at each higher resolution area, the tiles cover half the geographical area of the previous level.

¶16 Using this representation, we can recursively resolve certain data set regions in more detail than other regions. For example, Figure 1b shows the lower-right corner in high resolution with the surrounding regions displayed in progressively lower resolution. Assuming a tile size of 128 × 128 pixels, this example requires downloading and rendering only 491 Kbytes (10 tiles) instead of the entire 3.1-Mbyte high-resolution image. If the user's location is the bottom-right corner, then distant imagery is rendered at lower resolution than near imagery and we have achieved distance-based LOD.

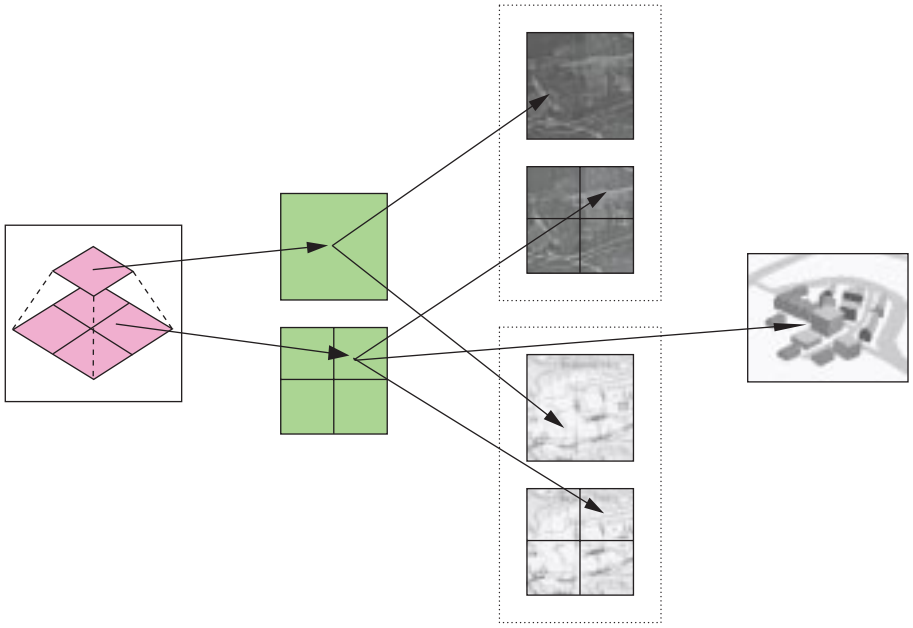
¶17 As Figure 2 shows, our image pyramids techniques can be applied to elevation grids and other types of terrain data. Because we use a tiled pyramid representation for the geometry and the imagery, we can optimize the amount of data transferred over the network, the number of polygons in the scene, and the amount of memory required for texture maps. As a result, we need only fetch and display data for the region that the user is viewing, and only at a sufficient resolution for the user's viewpoint. This solution scales well to arbitrarily large data sets because it effectively attempts to keep the polygon count constant for any viewpoint.



2 Using a tiled pyramid structure to represent terrain geometry. Closer terrain is represented in higher fidelity (more polygons) than distant terrain.

Multiresolution data in VRML

¶18 We introduce four types of VRML files to represent a large, tiled multiresolution hierarchy of the globe: terrain tile files, feature files, geotile files, and tree files. Figure 3 shows these files and their relationships. The tree files recursively implement the LOD hierarchy by inlining a single geotile file at one LOD and four higher resolution tree files at the next LOD. The geotile file inlines all of the feature and terrain tiles that cover a geographical area and LOD. A terrain tile file contains the actual elevation and image texture data for a given image, geographical area, and LOD. Feature files describe a geographical area's objects, such as buildings and roads. We discuss these relationships and their advantages below.



3 The relationship between tree, geotile, terrain tile, and feature files. Unidirectional arcs represent inline links to files over the network. Bold rectangles delineate file boundaries.

Tree files

¶19 Tree files implement part of the multiresolution hierarchy for the entire globe. In effect, these files are the glue that holds the geotiles in the quad-tree structure. A tree file initially loads a single geotile, but when the user approaches the tile, it's replaced with four higher-resolution tree files, which in turn inline the geotiles for the four quad-tree children. (At the bottom level of the tree hierarchy, the four geotiles are inlined directly.) The hierarchy of tree files must be generated only once and won't generally need to be modified further—except perhaps to extend the tree for higher resolutions at a later juncture. In the future, it might be possible to generate tree files on the fly.

¶20 The tree files let us split the entire LOD hierarchy over multiple files and abstract the LOD structure from the actual terrain data. They also let us create a different LOD tree depth for different global regions. For example, we could have 100-km-resolution data for the entire globe but recursively insert higher resolution data for smaller regions of interest, such as a one-km-resolution data set for the conterminous United States and a one-m-resolution data set for Yosemite Valley, California.

¶21 It would be possible to use the VRML `Inline` node to include the geotile files into the LOD hierarchy. However, VRML 97 does not specify when the Universal Resource Locator (URL) of an `Inline` node should be loaded, making this a browser-dependent feature. Currently, most browsers load all inline scenes at once. This makes good sense for small scenes with a handful of `Inline` nodes. However, if we have a data set of 10 Gbytes, the VRML browser will attempt to load all these data into memory at once. This is obviously unacceptable for our application, so we must control when inline files are loaded and unloaded. To do this, we developed a new node, called `QuadLOD`, using VRML 97's EXTERNPROTO and scripting features. `QuadLOD` provides a terrain-specific LOD capability that efficiently manages the loading and unloading of higher levels of detail. When the user enters a certain volume around the tile (determined by a `ProximitySensor`) `QuadLOD` loads only a tile's four higher resolution children. The node also uses a tile-caching mechanism so that tiles aren't needlessly reloaded. When the user approaches a region of terrain, more detail is progressively loaded and displayed in a coarse-to-fine fashion. Most VRML browsers perform nonblocking network reads so that the user can still interact with the scene while higher resolution imagery and elevation loads.

Geotile files

¶22 Geotile files contain links to all data within a single tile. The most basic geotile includes a single terrain tile file for the region. However, it's possible for a geotile to store links to multiple alternative terrain tiles, such as tiles referring to satellite, aerial, and map imagery, as well as feature files for objects that exist on the terrain, such as buildings, roads, and annotations.

¶23 By adding this extra layer to our global structure, we simplify the task of maintaining and adding new data sets. For example, when we want to add a new image pyramid, we can generate the terrain tiles in isolation and simply link them to appropriate geotiles. (If the new data set requires higher resolution than the region previously required, we also must generate new tree files.) In addition, because each data set is stored independently and referenced only via the geotiles, we can selectively display any combination of data sets. A node we created, `GeoTile`, permits this selectivity by organizing the terrain tile and feature file links into sets of `Switch` nodes.

4 A tiled terrain model showing geo-referenced 3D geometry overlaid for roads (orange) and buildings (yellow).

Terrain tile files

¶24 Terrain tiles contain the actual terrain data for a single data set tile at a particular detail level. This includes the elevation geometry and the texture map imagery for the specific terrain tile. The VRML 97 specification provides us with two potential primitives for representing terrain geometry: the `IndexedFaceSet` and the `ElevationGrid`. The latter node lets the user specify a grid of height values above the *x*-*z* plane, whereas the more general `IndexedFaceSet` node lets the user define arbitrary polygons in 3D space. VRML 97's `ElevationGrid` was introduced specifically to represent terrain models and offers a compact mechanism to describe simple height field data. However, it has serious limitations that prevent us from using it, including that it assumes that the heights are relative to a flat plane—an obvious problem when dealing with curved planets. `ElevationGrids` are thus useful only for modeling local areas, where the earth's curvature is insignificant. Because we want to support global data sets, we use the `IndexedFaceSet` to build terrain geometry.

Feature files

¶25 Feature files contain VRML models for objects related to a region of terrain. Examples include a region's cultural features, such as roads and lines of communication; weather simulations, such as clear air turbulence isosurfaces and wind vectors; and other 3D data, such as terrain annotations.

¶26 Integrating terrain features proves difficult because features can extend beyond tile boundaries. For example, a road might cover multiple tiles, or a large building might sit on the boundary between two tiles. One way to deal with this problem is to dissect the geometry for all ground features along tile boundaries, forcing the condition that all features in a tile are contained entirely within that tile. Another solution is to simply include a link to the same feature in all relevant geotiles. We selected the latter approach because it does not constrain the cultural features to the same resolution range as the terrain, and it requires no modification to the feature's geometry. However, one problem is that the browser would normally load, store, and render duplicate copies of each feature for every loaded geotile in which it occurs. To avoid this, we have designed our `GeoTile` node so that it keeps track of each request for a feature file's URL. We do this using static class structures in a Java script. We load feature files only on first invocation, incrementing their reference count, and unload a feature file only when its reference count returns to zero. Figure 4 shows the fusion of terrain data and cultural features that results.

Geographic coordinate systems

¶27 VRML defines a Cartesian coordinate system for modeling objects in a 3D volume. In geographic terms, this gives us a geocentric representation: a coordinate (x, y, z) is assumed to be a 3D offset (in meters) from the earth's center. However, most elevation data are provided in some geodetic or projective coordinate system. A geodetic coordinate system is related to the ellipsoid used to model the earth (such as the latitude-longitude system). A projective coordinate system projects the ellipsoid onto some simple surface, such as a cone or a cylinder. Examples include the Lambert Conformal Conic or the Universal Transverse Mercator projections. Such coordinate systems were designed for different applications and offer particular advantages and restrictions. For example, some projections can represent only small-scale regions; others are conformal, offering the same scale in every direction; and others can be equal area—the projected area corresponds to the earth's physical area over the entire projection. To use data in these different coordinate systems, we must convert coordinates in these systems into the VRML geocentric coordinate system. Descriptions for performing many of these transformations are available elsewhere.[9] The sidebar "What Shape Is the Earth?" discusses related representation challenges.

Floating-point precision issues

¶28 The VRML 97 specification defines the SFFloat and MFloat fields to represent floating-point numbers. However, these are only single-precision values; the dynamic range of a single-precision floating-point number isn't sufficient to store accurate geocentric coordinates. For example, the IEEE single-precision format defines a 23-bit mantissa. This provides a resolution of approximately six digits ($2^{23} = 8.39 \times 10^6$). The earth's diameter is roughly 12,700,000m, and thus we can model terrain to an accuracy of only tens or hundreds of meters. This accuracy can't even represent the results from a civilian-grade Global Positioning System (GPS), let alone faithfully represent ground features such as buildings or roads.

¶29 Most VRML browsers use only single-precision arithmetic for modeling and matrix operations, and most modern graphics hardware uses only single precision. We must therefore implement accurate geocentric coor-

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.