GENERATING ENHANCED NATURAL ENVIRONMENTS AND TERRAIN FOR INTERACTIVE COMBAT SIMULATIONS (GENETICS)

Major William D. Wells, USAF
MOVES Ph.D. Candidate
&
Christian J. Darken, Ph.D.
Associate Professor
MOVES Institute, Naval Postgraduate School

Abstract

Virtual battlefields devoid of vegetation deprive soldiers of valuable training in the critical aspects of terrain tactics and terrain-based situational awareness. Barren landscapes fail to provide trainees with necessary visual cues required to grasp the scale of their surroundings. Without the cover of vegetation, targets are easily visible from the air. Line of sight calculations become simply a matter of sorting elevations. There is a need to (re)introduce vegetation into the virtual battlefield to improve training effectiveness while minimizing the expenses typically incurred building such terrains.

This paper discusses the current state of the open source project GENETICS and how simulationists can use GENETICS to quickly and cheaply build large-scale natural environments to improve training effectiveness. It will also briefly touch upon level-of-detail techniques and ecotype modeling and how GENETICS is used to generate land cover data where none exists.

Introduction

Most large-scale terrains built for simulator-based military training are bleak, desolate places that share a strong measure of commonality with desert environments. One of the ongoing problems with relatively featureless environments is one's inability to grasp the scale of your surroundings. It is nearly impossible to determine distances or speed in a world devoid of a single bush, tree, or surface detail necessary to establish depth cues (Darken, 2003; Pietso, 2002; Wright, 2000). With terrains of large polygonal meshes draped with blurry satellite imagery (see Fig. 1), such visual cues are almost entirely absent for the infantry soldier on the ground or the low flying helicopter or aircraft pilot. This situation must change to improve training effectiveness.

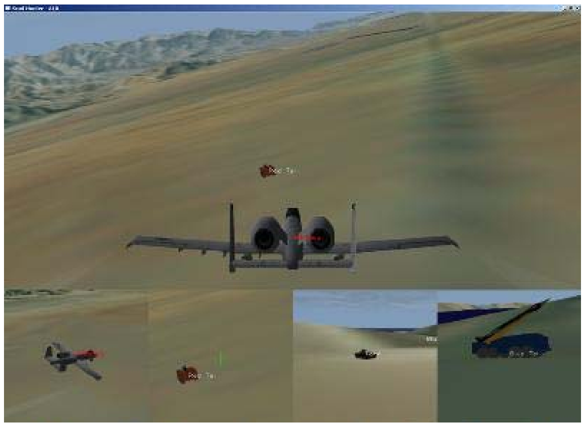


Fig. 1 Flight sim without vegetation

Within today's cockpit simulators, it is too easy for a pilot to quickly locate, identify and destroy targets on the virtual battlefield. When the only object protruding from the terrain surface is an enemy tank with no cover to hide behind, the task of acquiring and destroying your enemy is greatly simplified to the point of providing negative training. Adding vegetation to the synthetic environment makes training vastly more realistic and thus much harder.

Such terrain characteristics are highly desirable and thus for detailed simulated environments, like those found in the America's Army game (Davis, 2003; Zyda, 2003), a team of artists is normally hired to handcraft a custom terrain database. These

Presented at the IMAGE 2005 Conference Scottsdale, Arizona July 2005.

databases are not only simulation system specific, limiting their reusability or interoperability with other simulations, but also take a great deal of time to create. Additionally, such databases are typically focused around a player's expected actions and viewpoints. If players deviate from the developer's expectations, they quickly discover places within the world that simply do not "exist". These limitations prevent simulation scalability throughout the full spectrum of military operations.

Terrain visualization techniques often focus on optimizing the geometric mesh of the terrain's surface. Level-of-detail (LOD) techniques can render realistic-looking vegetation objects within desired performance constraints (Mantler, 2003). We can combine these methods with basic imagery and topographic analysis to automatically construct vegetation-laden terrain based on readily-available source data (elevation, imagery, and land cover classification), adding plausible terrain details as needed. Place these landscape construction and visualization procedures within a networked combat simulator and you have dramatically increased the difficulty of training exercises and improved our soldiers' chances in the field.

GENETICS

The aim of our research is to replace the barren landscapes found within most 3D combat simulations with detailed terrain and natural surroundings that dramatically increase both the believability and difficulty of the training environment - matching the synthetic perceptual stimuli to the actual perceptual stimuli needed to execute specific training tasks. We posit that there are many unmet visual cue requirements (e.g. vegetation) within existing simulators that are vital to the effectiveness of simulator-based training. Our approach enhances the apparent quality of the given set of terrain elevation data and surface imagery, adds vegetation objects that are placed similarly to the arrangement within the actual environment, and generates a plausible synthetic terrain environment where data is missing or incomplete.

Our algorithm uses on-demand, runtime processing of elevation data points to create 1 degree by 1 degree skirted height field meshes of the terrain. Height field data is imported directly from an elevation data repository (e.g. DTED® – Digital Terrain Elevation Data from the National Geospatial-Intelligence Agency). Ground surface details between the known elevation postings are added by subdividing the base mesh and increasing or decreasing the values of the linearly interpolated midpoint heights with Perlin noise (Perlin, 1985). Filtering the amplitude of the noise based on elevation values (e.g. larger elevation changes possible in mountainous settings versus plains) helps to overcome the appearance of randomness between postings. Using the SOARX continuous level-of-detail (CLOD) algorithm (Balogh, 2003), we take our enhanced height field data and construct a dynamically optimized mesh grid based on the user's view frustum. As the user nears the edge of the terrain, we determine the next geocell's coordinates, load up the corresponding source data from our repository, and process the next 1 degree by 1 degree geocell in the same manner. Generating a geocell's height field takes only a matter of seconds. These techniques allow us to offer the user a nearly endless supply of optimized elevation meshes derived from raw source data with increased resolution over the given source data.

Over our elevation mesh, we drape satellite imagery shaded at run-time with normal maps (i.e. one for the base gradient and one for the detail gradient) to add terrain shading and surface details corresponding to the noise-generated additions to the elevation data. A geocell's height field data, noise data and associated textures are cached for improved load times or can be regenerated afresh if desired. The ability to consistently recreate the same mesh every time is controlled by the selection of a random number seed.

At this point, we have created a terrain visualization environment (i.e. imagery over a mesh) that could easily be used for many flight simulation applications. However, in order to bring in participation from ground-based and low-level aerial platforms, we need to look at adding more details to our landscape. Thus, from the geocell's height field data, we create height maps, slope maps (with aspect angles), and relative elevation maps. These images are needed in the GENETICS vegetation placement process.

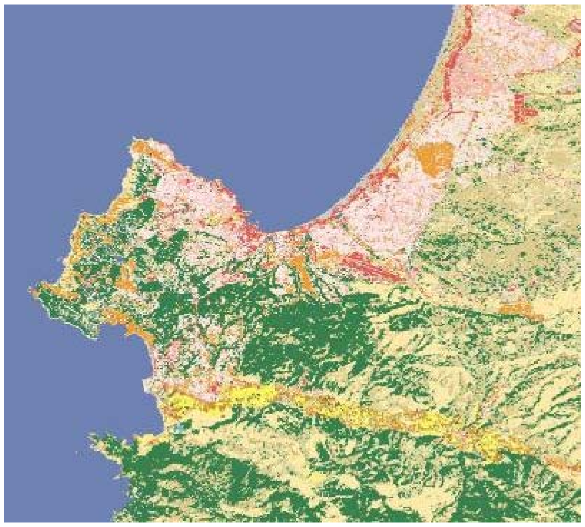
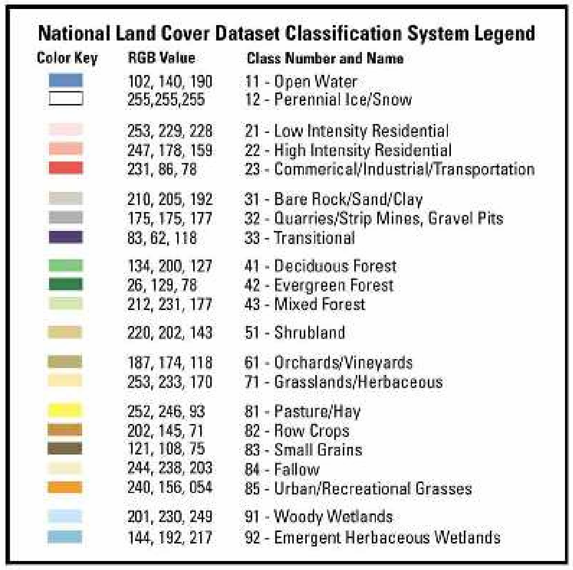


Fig. 2 NLCD map of Monterey, CA

For vegetation placement, we process land cover classification (LCC) GeoTIFF images (see Fig. 2) of the region of interest. The USGS's Seamless Data Distribution System offers users the ability to freely download National Land Cover Data (NLCD) as orthorectified GeoTIFF images from the Internet, but any LCC scheme using GeoTIFF images can be used (including manually created images). NLCD images use a palette of 21 colors to represent different LCC types (see Fig. 3).



National Land Cover Dataset Classification System Legend

Color Key	RGB Value	Class Number and Name
🟦	102, 140, 190	11 - Open Water
⬜	255,255,255	12 - Perennial Ice/Snow
🟥	253, 229, 228	21 - Low Intensity Residential
🟥	247, 178, 159	22 - High Intensity Residential
🟥	231, 86, 78	23 - Commerical/Industrial/Transportation
⬜	210, 205, 192	31 - Bare Rock/Sand/Clay
⬛	175, 175, 177	32 - Quarries/Strip Mines, Gravel Pits
🟪	83, 62, 118	33 - Transitional
🟩	134, 200, 127	41 - Deciduous Forest
🟩	26, 129, 78	42 - Evergreen Forest
🟩	212, 231, 177	43 - Mixed Forest
🟨	220, 202, 143	51 - Shrubland
🟫	187, 174, 118	61 - Orchards/Vineyards
🟨	253, 233, 170	71 - Grasslands/Herbaceous
🟨	252, 246, 93	81 - Pasture/Hay
🟫	202, 145, 71	82 - Row Crops
🟫	121, 108, 75	83 - Small Grains
🟨	244, 238, 203	84 - Fallow
🟧	240, 156, 054	85 - Urban/Recreational Grasses
🟦	201, 230, 249	91 - Woody Wetlands
🟦	144, 192, 217	92 - Emergent Herbaceous Wetlands

Fig. 3 NLCD legend

A user of GENETICS notes the red, blue and green values assigned to each LCC type along with its description in an XML file that is parsed at run-time. Also included in this file are the topographical regimes for each LCC type and the various geometric object models that will be placed in the scene corresponding to each LCC type. Note that multiple object models can be assigned to a single LCC type (e.g. young, medium, and old versions of a vegetation object).

After the LCC configuration file is parsed and the LCC source image loaded into the system, we create an image for each desired LCC type showing "picked points" (i.e. a black pixel is given for each corresponding pixel in the LCC image that matches or "hits" the color values of the LCC type; a white pixel represents a "miss"). A union of all the LCC "picked points" colored with their respective LCC color values would recreate the original source image. This demonstrates the "all or nothing" approach of using an LCC image. Each pixel (and thus its corresponding ground location) is designated a single LCC type with no overlap possible. Since this situation rarely occurs in nature (particularly in transition zones), we need to "smooth" this data from black or white (i.e. on or off) to various shades of gray (probabilities of occurrence).

We create a smoothed image for each LCC type using a "third nearest neighbor" weighting scheme. If our current pixel is a "picked point" (i.e. a "hit") for that LCC type it earns a score of 50. A "miss" earns no score. We then look to the north, south, east, and west of our current pixel. A hit from any of these four pixels earns our current pixel another 6.82 points each. From our next nearest neighbor, the four diagonals (i.e. northeast, southeast, southwest, and northwest of our current pixel), a hit earns our current pixel another 3.41 points each. Finally, for our third nearest neighbor, the pixel beyond each of our nearest neighbors in the four cardinal directions, we earn a 2.27 for each hit. With this smoothing function, a "hit" pixel located in a dense patch of other "hit" pixels will trend towards a score of 100 while an isolated "hit" pixel will trend towards 50. Likewise, a "miss" pixel that is completely surrounded by "hit" pixels will trend towards 50 while a "miss" pixel far removed from any "hit" pixels will trend towards 0. Using such a filtering scheme, we have created an initial probability map for each LCC type by relating a pixel's composite score to the blackness or whiteness of each pixel. However, we need to manipulate this map further to account for topological influences that we derived previously.

The idea of using elevation, slope, aspect angle, and relative elevation to affect vegetation placement was inspired by Johan Hammes' paper, "Modeling of ecosystems as a data source for real-time terrain rendering" (Hammes, 2001). The value of each pixel (i.e. the probability of vegetation placement) from our "smoothed" image is increased or decreased based upon the preferred regimes of that particular LCC type. For example, an evergreen may have an elevation regime of 0 to 3000 meters, which means that the closer one gets to either of those extremes, the more unlikely an evergreen object is likely to be placed there. This allows us to create (or enforce) timberlines and prevent trees from growing in the ocean. Similarly, evergreens are unlikely to grow on a very steep slope. Our LCC configuration file should reflect a maximum slope angle and probabilities adjusted negatively as we approach that angle. A slope's aspect angle will affect the density and growth of some LCC objects. For example, in the North Hemisphere, trees typically grow better on slopes that face south or west (Hilts, 1999), resulting in denser, older growth. Relative elevation (i.e. comparing the difference between a particular point's elevation and the average elevation of its surrounding neighbors) is an effort to recognize that dips and valleys in the terrain will likely receive more water and be more sheltered from the weather than rises and ridgelines. Thus, we may wish to bias our probability to promote placement in valleys (i.e. negative relative elevation) and reduce the probability of placement along ridgelines (positive relative elevation).

Each of the above factors contributes to the final score of each pixel for each LCC type. Collectively, these pixels form a probability map reflecting the likelihood that a particular LCC type exists within the pixel's corresponding area on the terrain surface (see Fig. 4). Random draws against these probability maps determine the type, location, density, and appearance of the vegetation objects found within the synthetic natural environment. Randomized orientation and scaling of the objects give the appearance of a greater variety of models.

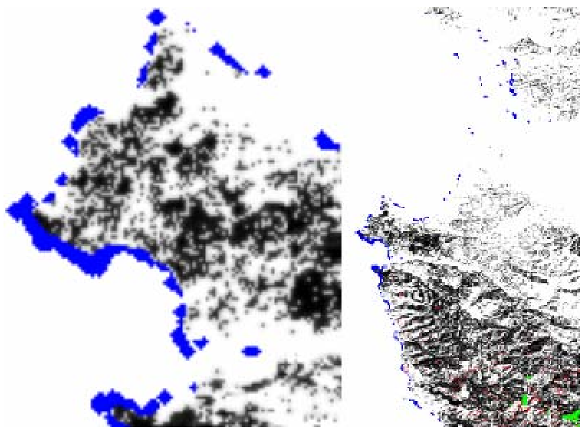


Fig. 4 Evergreen probability map

The resulting procedurally created geotypical distribution looks plausible (see Figs. 5-A and B) with overlapping vegetation types occurring naturally within transition zones. This simple algorithm can also be extended to incorporate soil moisture or other factors (e.g. prevailing winds, proximity to water) or to generate geotypical distributions of man-made landscape features (as seen in Figs. 6 and 7).

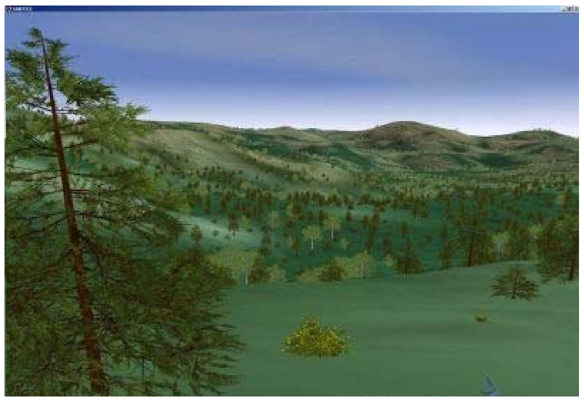


Fig. 5-A Geotypical vegetation distribution

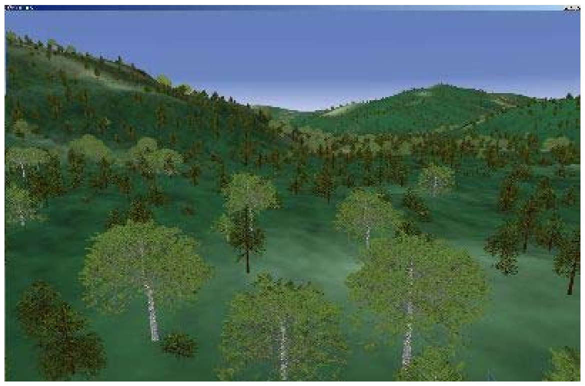


Fig. 5-B Geotypical vegetation distribution

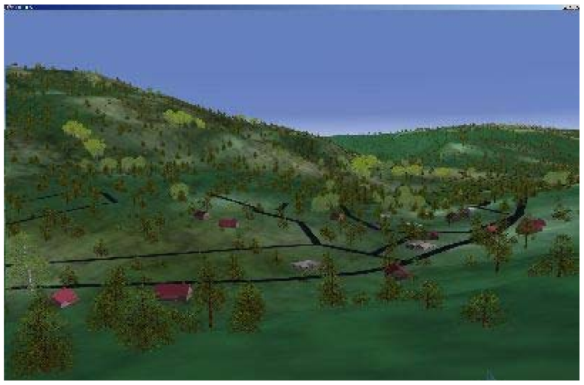


Fig. 6 Rural distribution

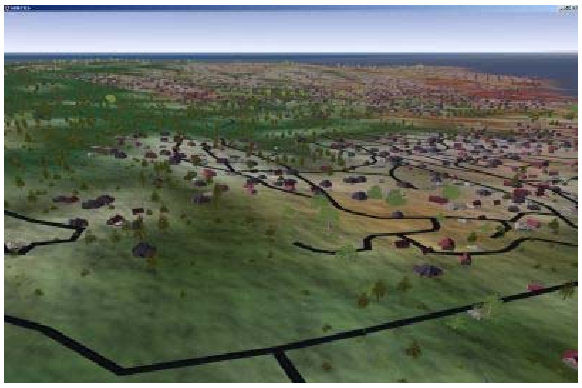


Fig. 7 Urban distribution

A masking image adds the capability to prevent a specific LCC type or multiple types from occurring in a particular region. This allows for easy removal of vegetation objects from lakes or recent alterations to the terrain (e.g. defoliation, clear-cutting, wildfires), but can also be used to designate an area on the map where a geospecific urban environment needs to be placed.

Once the location, orientation, and scale of a particular object model has been established, it must be added to the scene using an efficient spatial data structure such as a quadtree. As the bounding volume of a branch of the quadtree intersects or falls within the view frustum, that branch is considered active and potentially viewable. Non-active branches are culled away from the rendering of the current frame. Small pixel culling prevents rendering objects that do not meet a minimum screen size threshold. Finally, the object model itself can make use of switch nodes within its geometry to determine the appropriate LOD required by the scene (typically as a function of distance from the view). Thus, distant objects can be drawn as billboards, medium range objects can be represented as intersecting planes, and close objects can be depicted as complex geometric objects. Some commercial packages will create LOD object models automatically, and we have chosen such a package for our own work. Our assumption is that most simulation centers are likely to have a custom-built library of such objects at their disposal or have the means to quickly generate such objects as needed. We believe the creation of such objects is not the problem, but that the realistic placement of millions of them within a scene is the larger challenge.

Automatic Generation of LCC Data

A logical alternative to creating terrain probability distributions by hand is to use machine learning technologies to construct them automatically. We are investigating the use of machine learning techniques for automatically estimating the LCC distribution in such forms as k-nearest neighbor estimators, simple Bayesian belief networks, and neural networks. The attraction of these techniques is twofold. Firstly, they typically result in a formally-specified probability distribution whose assumptions and biases may be rigorously characterized. Secondly, these techniques can be partially or fully automated, reducing the workload on a human modeler. A liability of these techniques is that they require "training data" in order to function, e.g. a region in which the LCC values are known and provided to the system. The tacit assumption here is that the provided training data is correct and has a similar distribution to the target locale to which the technique is being applied.

The basis for all machine learning approaches to constructing a probability density is a set of training data. The training data consists of a set of *exemplars* each of which corresponds to a single pixel for which the correct LCC type is known. Each exemplar is an ordered pair consisting of domain and range values. The range value is the LCC type. The domain values can be any available quantities relevant to predicting the LCC type. At a minimum, we include the elevation, slope, aspect angle, and relative elevation for the pixel in question. Each of these values is represented as in integer between 0 and 255 inclusive.

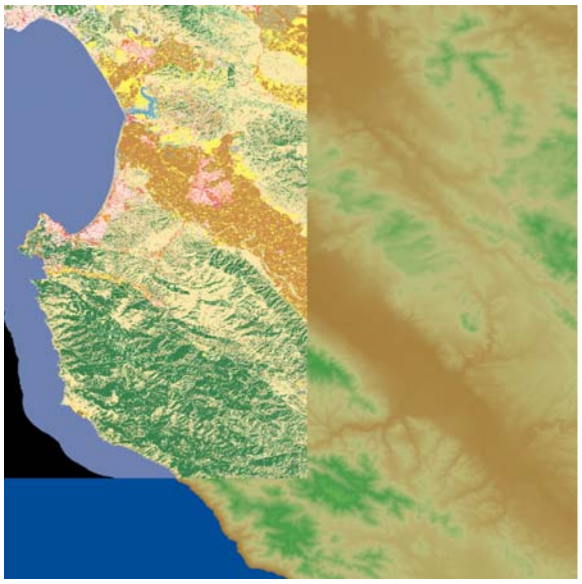


Fig. 8 The rectangle in the upper left is the training region (i.e. known LCC data). The rest are false color elevation data.

The simplest and most successful approach we have tried so far is to estimate the LCC type using a k-nearest neighbors density estimator (Duda, Hart, and Stork 2000). At each pixel where the LCC type is unknown (the *query point*), the domain values described above are gathered. Let us take x_i to be the vector of domain values for the i'th member of the training set and y_i to be the corresponding (known) value of the LCC type. Let x to be the corresponding vector of values at the query point. For each query point we find the k exemplars with range values (x_i's) closest (in terms of ordinary Euclidean distance) to the query point. The search for the nearest exemplars can be performed in log time by using data structures and search algorithms designed to make spatial range queries efficient, such as the k-d tree (Preparata and Shamos 1985). The probability of each LCC type at the query point is then taken to be the fraction of the k exemplars having that LCC type. Figs. 8 and 9 display the region with known LCC values and the maximally-likely value of the LCC type over the entire region, including on the training set. Note that the predictions for the training region would not be used and are presented only for purposes of comparison to the ground truth. While highly imperfect (some features such as urban regions are not predicted at all), it may be usable for some purposes.

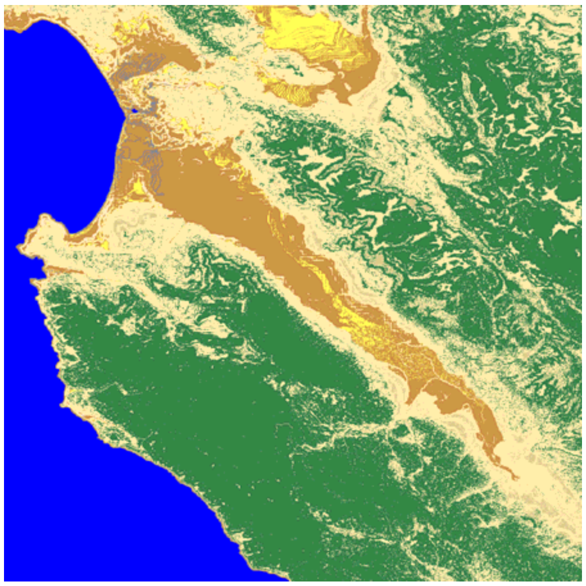


Fig. 9 The maximum likelihood estimate of the LCC type with the k-nearest neighbors estimator.

One weakness of this model as compared to the by-hand approach outlined above is that it does not take into account the LCC type assigned to nearby pixels. There is a chicken-and-egg problem implicit here, as if the LCC type of a given pixel is unknown, the types of neighboring pixels may be unknown as well. This difficulty may be overcome by constructing the LCC type estimates iteratively. That is, given an initial, possibly random, guess of all unknown LCC types, the LCC type at each pixel is estimated to be consistent with the guess. This estimation process is then repeated until convergence. This is a fairly established idea that began with (Geman and Geman 1984) and has spawned a whole field of research (Winkler 2003). We have used both simple (also called "naïve") Bayesian networks and single artificial neurons (equivalent to logistic regression) as the model in applying this approach (Russell and Norvig 2002), but a satisfactory result has yet to be achieved. A more sophisticated coding of the inputs to the artificial neuron (i.e. "coarse coding" versus direct input of the rescaled elevation, etc.) may yield better performance.

Implementation Details and Performance

GENETICS resides in the SOARXTerrain component of the military's open source game/simulation engine, Delta3D. Delta3D is a high-level API that sits on top of numerous other open source libraries such as Open Scene Graph (Burns, 2003), Open Dynamics Engine, OpenAL, Character Animation Library, and others. Within GENETICS, extensive use is made of the Geospatial Data Abstraction Layer (GDAL). GDAL is an open source translator library for raster geospatial data formats. GDAL allows GENETICS integrate diverse datasets (NGA DTED®, USGS NLCD, NASA LandSat7, and US Census Bureau TIGER) without the need for expensive proprietary tools.

As noted earlier, GENETICS works in tandem with our CLOD implementation SOARXTerrain, but SOARXTerrain is not dependent on GENETICS. Thus, we can run a Delta3D application that uses only SOARXTerrain, but not GENETICS. We could also replace the terrain surface CLOD algorithm (e.g. with another CLOD implementation or a static mesh) without impacting GENETICS.

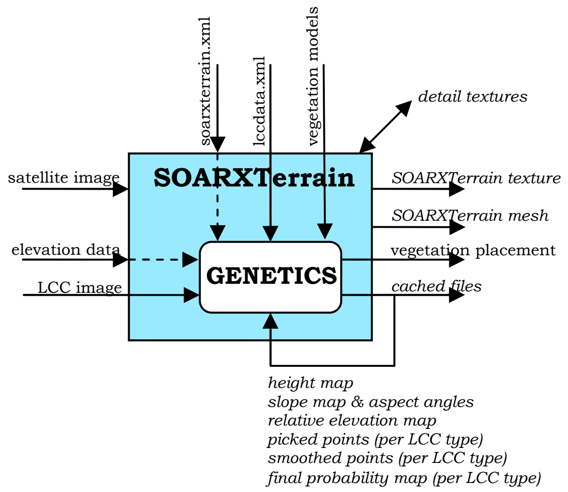


Fig. 10 GENETICS "black box" diagram

In Fig. 10, we show the flow of data through the system. Raw source data is input from the left. User-manipulated XML configuration files and visual models are input from the top. SOARXTerrain generates a terrain mesh, base texture, and detail textures that are cached for subsequent runs as desired. The elevation data is also used by GENETICS to create its topographic textures. The LCC image is used to generate the picked points and smoothed points textures. Finally, a probability map is created for each LCC type. All of these images are cached for subsequent runs of the same terrain.

Our GENETICS test system (Athlon XP 3000, 1Gb RAM, NVIDIA 6800GT 256Mb) currently runs at 15 frames per second (1600x1200) with 1.6 million objects from five LCC types represented in the scene. From the start of execution with an empty cache to the first rendered frame takes approximately two minutes. Using cached data cuts this time in half.

Geometric object instancing (Carucci, 2005), still a graphics card driver dependent feature, holds the promise of dramatically increasing this frame rate by storing an object's vertex and texture information on the card, thus requiring only the passing of position attitude transform information to the shader program. Additionally, great pain has been taken to maximize the use of textures as a storage medium to facilitate the expanded use of shader programming to improve the performance of GENETICS.

Conclusions

We have seen that separate efforts in improving one particular aspect of terrain fidelity are not enough to give players the realism that tactical military training requires. Thus, we have taken an integrated approach to incorporate "best of breed" techniques within a single architecture. While previously it has taken teams of artists to create static, small scale, custom-tailored landscapes, our approach automatically generates vast realistic terrains at runtime for any place on Earth. With a minimum amount of shared source data and parameters, terrains can be synchronized easily between clients; guaranteeing the same terrain environment is created by all hosts within a heterogeneous networked simulation system. Our terrains can be reused or regenerated afresh with new parameters in response to the needs of the training audience. With the simple change of a random number seed, a new terrain database can be generated without the need to manipulate a database. This feature allows trainers the flexibility to use the same terrain repeatedly or create a new one each time; forcing trainees to not depend upon the static nature of most simulation databases.

The immediate and practical benefit of this work is that tactical training improves by giving players a more realistic environment in which to operate. It is possible to have simulators and simulated forces engage in a multi-spectrum tactical conflict where the natural environment takes on an active role in the experience and is no longer simply a backdrop. It is only at this point, when the ground cover looks real and foliage hides your view of the enemy that terrain can truly work towards becoming a full-fledged entity within the distributed virtual environment.

References

Balogh, A. (2003). Real-Time Visualization of Detailed Terrain. Computer Science Masters thesis. Budapest, Hungary: Budapest University of Technology and Economics.

Bitters, B. (2004). Real-Time Simulation Database Generation: A Conceptual Model for the Future. Proceedings of the 2004 IMAGE Conference (pp. 50-60). Scottsdale, Arizona: The IMAGE Society.

Burns, D. and Osfield, R. (2003). Open Scene Graph. Proceedings of the 2003 IMAGE Conference (pp. 76-82). Scottsdale, Arizona: The IMAGE Society.

Carucci, F., (2005). Inside Geometry Instancing, GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation. New York: Addison-Wesley Professional.

Darken, R., Sullivan, J., and Lennerton, M. (2003). Practical Issues in Measuring and Assessing Training Effectivess of Virtual Environments for Military Applications. Proceedings of the 2003 IMAGE Conference (pp. 126-136). Scottsdale, Arizona: The IMAGE Society.

Davis, M. ed. (2003). America's Army PC Game: Vision and Realization, U.S. Army and MOVES Institute: Monterey, California: Naval Postgraduate School.

Duda, R., Hart, P. and Stork, D. (2000) Pattern Classification, 2nd Edition, Wiley-Interscience.

Geman, S. and Geman, D. (1984) Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. IEEE Trans. PAMI, 6:721—741.

Hammes, J. (2001). Modeling of Ecosystems as a Data Source for Real-Time Terrain Rendering. Proceedings of the First International Symposium on Digital Earth Moving (pp. 98-111). London, U.K.: Springer-Verlag.

Hilts, S., and Mitchell, P. (1999). The Woodlot Management Handbook, Buffalo, New York: Firefly Books Inc.

Mantler, S., Tobler, R. F., and Fuhrmann, A. L. (2003). The State of the Art in Realtime Rendering of Vegetation, VRVis Tech Report 2003-027, VRVis Research Center for Virtual Reality and Visualization, Vienna, Austria.

Peitso, L. E. (2002). Visual Field Requirements for Precision Nap-of-the-Earth Helicopter Flight, MOVES Masters thesis. Monterey, California: Naval Postgraduate School.

Perlin, K. (1985). An image synthesizer, ACM SIGGRAPH Computer Graphics, v.19 n.3, (pp.287-296). New York: ACM Press.

Preparata, M. and Shamos, I. (1985) Computational Geometry, an Introduction, Springer-Verlag.

Russell, S. and Norvig, P. (2002) Artificial Intelligence: A Modern Approach, 2nd Edition, Prentice Hall.

Winkler, G. (2003) Image Analysis, Random Fields, and Markov Chain Monte Carlo Methods, Springer.

Wright, G. T. (2000). Helicopter Urban Navigation Training Using Virtual Environments, Computer Science Masters thesis. Monterey, California: Naval Postgraduate School.

Zyda, M., Mayberry, A., Wardynski, C., Shilling, R., and Davis, M. (2003). The MOVES Institute's America's Army Operations Game. Proceedings of the 2003 Symposium on Interactive 3D Graphics (pp. 219-220). Monterey, California: ACM Press.

Authors' Biographies

Major William David "Fuzzy" Wells is the first Air Force officer to seek a Ph.D. at the Modeling, Virtual Environments, and Simulation (MOVES) Institute, Naval Postgraduate School located in Monterey, California. He is a 1991 graduate from Georgia Tech with a Bachelors of Aerospace Engineering and a 1996 graduate from the AF Institute of Technology where he earned an MS in Computer Systems specializing in Modeling and Simulation (M&S). Maj Wells was an Air University "Prime Warrior" instructor, teaching AF officers the basics of M&S, wargaming, and the application of aerospace doctrine. Maj Wells has also served as exercise designer and senior controller for numerous battlestaff training exercises worldwide. In 2000, he was assigned to the AF Agency for M&S where he directed AF M&S participation in the Congressionally mandated Millennium Challenge 02 experiment. In 2002, Maj Wells was designated a Certified Modeling and Simulation Professional Charter Member.

Christian J. Darken, Ph.D., is an Associate Professor of Computer Science at the Naval Postgraduate School, where he also collaborates intensively with the MOVES Institute. Previously he was Project Manager of the Decision Support Systems project and Senior Member of Technical Staff at Siemens Corporate Research in Princeton, NJ, where he was variously associated with the Learning Systems, Adaptive Information and Signal Processing, and Software Engineering Departments. He was also a programmer of one of the first commercial first-person perspective massively-multiplayer games. He received his Ph.D. in Electrical Engineering from Yale University in 1993, and previously received the M.S. and M. Phil. in Physics from the same institution.