Final Report to the Governing Board, Joint Fire Science Program
Project 05-4-1-12
Innovative, 3-D, Interactive, and Immersive Techniques for Visualizing, Querying, and Understanding Regional Maps of Forest Vegetation, Fuels, and Fire Risk

Principal Investigators:
Janet L. Ohmann
Matthew J. Gregory
Timothy M. Holt

28 September 2007

Introduction

This report summarizes accomplishments, key findings, and final products from Project 05-4-1-12 funded by the Joint Fire Science Program, 'Innovative, 3-D, Interactive, and Immersive Techniques for Visualizing, Querying, and Understanding Regional Maps of Forest Vegetation, Fuels, and Fire Risk.' The project is also referred to as the 'GNNViz' project. This report is accompanied by a DVD containing all project deliverables in electronic format (see Appendix 1). This report and final products also can be viewed and downloaded from our website, http://www.fsl.orst.edu/lemma/gnnviz. A glossary of terms and abbreviations used in this report is included at the end of the report.

Project Background

As part of our previously funded JFSP project (Project 01-1-4-09), called 'GNNFire,' we created maps of vegetation and fuels using the Gradient Nearest Neighbor (GNN) method. We distributed our map products and methodology through traditional technology transfer modes, including meetings with stakeholders, presentations at workshops and conferences, research papers, and our website. However, communication with users during both development and technology transfer phases of our research was hampered by the physical distance between investigators and users, by the complexity and novelty of the GNN map products, and by time limitations. In the course of our research it became apparent to our research team that alternative and more effective ways of communicating regional spatial information were needed.

Our traditional methods for portraying fuel patterns were not always effective for fuel managers and field specialists. In general, it was difficult for many users to move beyond viewing the GIS-based maps as abstractions of reality, and there often was a tendency to distrust this model-based view of the real world. There also was a lack of understanding of the richness of the vegetation and fuels data contained in the GNN-based maps, as many users were unfamiliar with maps based on imputation methods. We believed there were opportunities to make the regional maps more 'real' by developing innovative ways for users to visualize and interact with the maps. These enhanced visualization tools would also create opportunities for two-way communication between researchers (map developers) and managers, for scientists to learn from managers where the maps were accurate and where they needed further improvement.

To that end, we proposed developing innovative ways of visualizing and interacting with broad-scale mapped information on fuels and vegetation to improve communication between researchers and managers, as well as to enhance delivery and use of our own and similar research products. Our objectives were to:

1. Create an interactive and immersive tour of our three study areas using state-of-the-art gaming technology and detailed data from existing GNN vegetation and fuels maps. The methodology for creating this visualization environment would be generalized to be extendable to other locations and mapping methods

2. Couple these visualization tools to web-based map server technology to give the user analytical tools for querying the underlying GNN mapped attributes and the ability to overlay polygon and raster information from external sources into the visualization environment.

We have successfully created the visualization environment for our three study areas and developed an application that is extensible to other mapped information and geographic locations. The research path taken to arrive at this application was not always as proposed, and many of our key findings speak to the evolutionary nature of developing software that relies on cutting edge technology. The software has morphed from a paired visualization environment and web-based mapping site, as outlined in our objectives above, into an integrated visualization and analysis platform. We are not reinventing the GIS wheel, although the visualization environment does have some elements that will be familiar to GIS users. Rather, we have created a tool for visualizing and browsing spatial information that is unique from GIS – a hybrid experience between GIS, Google Earth, and a computer game. We believe this platform holds promise for future visualizations and simulations.

Study Areas

Our three study areas were: the Inland Empire of northeastern Washington, the coastal province of western Oregon, and the west slope and foothills of California's Sierra Nevada. The study areas range from 31,000 to 50,000 km^2. The study landscapes represent three distinctive ecoregion Divisions (Bailey 1995): temperate steppe, marine, and Mediterranean.

Key Findings

Game Engines

The choice of a game engine is an important decision that significantly impacts application development time and project direction

One of our greatest frustrations in this project has been finding an appropriate game engine technology that would serve the project's requirements. In general, most game technologies concentrate their efforts on rendering a relatively small game space (on the order of a kilometer or less in dimension) in high detail. For entertainment games, content is typically hand-created by developers and artists and is not programmatically or procedurally generated. This contrasts with GNNViz requirements that we render extremely large areas on the order of 100's of kilometers across, and render potentially millions of objects (trees) in this landscape.

So though many game technologies would have allowed us to create small, high-fidelity renderings of forest scenes, we found very few that would let us render large areas with large numbers of items. It in a sense became a ‘quantity versus quality’ issue, with most game technologies supporting the ‘quality’ side of the coin.

We considered several different game engines for this project, described below. Our comparative assessment of them is in Appendix 2.

We initially proposed using Valve Software's Source engine as the base for our visualization environment. It is well documented and has a large developer community, which speaks to its stability and usability. However, we quickly ran into limitations. There were issues with level-of-detail (LOD) rendering and game tiling that we were not able to modify through the program's application-programmer interface (API). Tiling is the concept of having the game engine only keep in memory portions of the terrain (arranged as tiles) that are currently visible by a player. The Source engine (along with many others) does allow entire new game environments to be loaded so the player can transition from one area to another. However, it requires that all players move to the newly loaded environment, which rules out multiple players exploring different parts of a shared larger environment.

This led us to consider other commercial, off-the-shelf engines such as Crytek's FarCry and Oblivion, but other limitations such as limited elevation ranges (FarCry) and no multi-player support (Oblivion) made these options unsuitable as well. As with Source, these engines provided effective APIs for game modification, but they did not allow us to implement large areas of terrain with vast numbers of accurately positioned trees.

In March 2006, we looked at two new game engines that seemed better suited. One was the Multiverse game technology, which is designed to manage vast game areas for a type of game known as an MMOG (Massive Multiplayer Online Game). Though capable of

rendering large areas, Multiverse technology did not provide us with sufficient control over placement of trees and other objects, which meant we could not duplicate the vegetation distribution present in our GNN vegetation and fuels maps.

The other technology we examined was the Garage Games Torque engine, which gives users full access to the source code and is able to render large terrains with many objects. We developed a small (20km x 20km) example scene and devoted nearly six months of development time to creating custom algorithms and content suitable for our project. We gave a number of presentations and demonstrations of this visualization based on this example scene, including a September 2006 presentation to the JFSP Governing Board. However, when it came time to render other scenes in our study areas, the modules and file formats for creating topographic surfaces from digital elevation models (DEMs) were very buggy and ultimately unusable.

Finally, in May 2007, we changed engines again and settled on the Delta3D engine (http://www.delta3d.org/). Delta3D is an open-source project which has a number of desirable features including automatic tiling of large landscapes, easy translation of GIS data into game content, and fully modifiable source code. The drawback to this engine was that sample applications were not developed as fully as with other engines, so more development time was required to modify the existing content. The engine has improved dramatically from when we began the project, so many of these desirable features were not present during our initial engine evaluation.

These decisions, as a whole, have increased development time on this project and pushed back our project schedule. Indeed, this is a major challenge of implementing a project that is highly dependent on cutting-edge technology, which by definition is rapidly changing and evolving. The target is always moving, and has required that we learn and adapt as we went.

Delta3D is a flexible and extensible platform for regional visualizations

Delta3D offered several compelling reasons for use as the GNNViz game engine. As Delta3D is an open-source project, we were able to modify the code to fit our project's requirements. More importantly, it directly supports effectively infinite terrain via dynamic tiling and caching of game content to create a manageable memory load while running the visualization. The game also included modules for dynamic geographic placement of tree objects onto the landscape. In general, the Delta3D engine had greater ‘geographic awareness’ (i.e. an established relationship with geographic coordinates) than all other engines we investigated. As such, incorporating georeferenced datasets, such as digital elevation models (DEMs) and ancillary drape images, is greatly simplified (see Appendix 3).

Building on the base Delta3D code, we extended the source code extensively to include a variety of enhancements specific to our project. These include: creating flexible tile sizes, adding multi-player and chat capabilities, generating intuitive heads-up-display (HUD) of

user interface elements, and tying controlled placement of objects (trees) onto the terrain surface based on underlying GNN data.

Of the features added to the Delta3D code, the ability to control object placement based on underlying data provides the greatest opportunity for extensibility. Given any density surface and one or more models to represent this density, a new landscape visualization can be added to GNNViz, requiring only a change to an XML file without a recompilation of the source code. For example, if a user wished to visualize fire risk within the wildland-urban interface (WUI), s/he could symbolize housing density with individual 3D house models over a landscape draped with a fuel model image. One might even visualize a fire by taking a map representing real or simulated fire densities and using it to control placement of flame models on the simulated terrain.

There are limitations to the Delta3D engine as well, but these tend to be common limitations shared among all three-dimensional game engines. Most notably, the representation of millions of discrete objects (such as trees) is a computationally intensive process and can negatively impact the user experience. This is almost exclusively a hardware-dependent issue and we saw dramatic differences in performance based on different systems that we tested. As computing and graphics performance continue to improve, this application will be able to render the visualization environment almost seamlessly.

Realism in Visualization

3D game engines provide unique possibilities for rendering at a variety of spatial scales

Typically, users of geospatial data are used to visualizing their data in GIS as a top-down view with data abstracted to points, lines, polygons and images. With the advent of some newer applications based on 3D graphics libraries (e.g. Google Earth, NASA WorldWind, ArcGlobe), users have the ability to be a part of their data by rendering oblique views and generating limited numbers of three-dimensional objects. This project extends that trend by providing different symbolic information at different spatial scales.

At the coarsest scale in our GNNViz application, users are presented with a traditional GIS view, albeit draped upon a realistic topographic surface. As the user gets closer to the surface, s/he transitions into a more realistic view of the ground surface, which we achieved by combining textures (i.e. two-dimensional tiled images) of forest floor photographs. The ratio of each texture's contribution to the overall combination is controlled by the modeled GNN canopy cover. For example, in dense conifer forests, the user will see a ground texture which is almost exclusively needles with very little understory vegetation, whereas in an open hardwood stand, various forbs and grasses tend to dominate the ground texture.

At the finest scale, users are presented with the actual tree models (graphical renderings of trees) in relative amounts of conifers, hardwoods, and snags as predicted from GNN.

These have been generalized to three tree models for each group, but future work may include increasing the number of unique models used to symbolize densities of individual tree species, size classes, crown geometries, etc. Although it would be hard to argue that the landscape is photo-realistic, the finest scale of rendering does allow the user to more easily visualize GNN model output than is possible through traditional GIS programs. Part of the advantage is due to the way users can navigate through the visualization landscape.

'Pseudo' satellite imagery can be effectively generated from GNN data for distant viewing

Although this feature is not yet incorporated into the current release of the visualization environment, we have developed a method of creating images which mimic the look and feel of fine-grain satellite imagery or air photos, but are instead generated from GNN model data. This view can provide a seamless transition from draped images into 3D models.

For every pixel in a GNN model, the visualization creates a 'stem map' of 3D objects based on the total tree density from the GNN prediction. Because we have no information on how the trees are distributed across a pixel, we randomly place trees in the pixel until we reach the required density. The key is that the random number generator is actually a known sequence but can be initiated with a truly random seed to begin the sequence. Rather than allowing the seed to be random, we set it to be a linear combination of the pixel's row and column coordinates. In this way, we create the stem map to have a random spatial distribution across the pixel that is generated identically each time that pixel is visited.

To create the 'pseudo' satellite image, we use this same distribution of the 3D models, but use a top-view, two-dimensional (2D) image of a tree crown to represent that 3D model. As the user zooms in, the 2D crown images fade into actual 3D models in the same locations. We had implemented this feature into the Torque engine version of the visualization environment, and we hope to incorporate this functionality into future versions based on Delta3D.

Two-dimensional 'billboards' are surprisingly effective and computationally inexpensive for rendering trees

In game engines, it is possible to render trees as either 3D models or 2D 'billboards.' Tree models are typically comprised of hundreds to thousands of individual polygons at their most detailed, and exert a heavy load on the computer's graphics card. When we tried to render GNN models, which may consist of tens of thousands of trees in a single scene, we found that frame rates (i.e. the number of frames rendered per second) were very low. This led to discontinuous movement in the visualization environment.

'Billboards,' on the other hand, are 2D vertical rectangles with an image texture stretched to fit onto it. As such, there is only one polygon to render per tree object, and GNN

scenes are rendered much more smoothly. When a user is walking along the ground surface in the visualization environment, billboards rotate on their z-axis to always face the user. In addition, the images used for billboards can have transparency built in, so that billboards in the foreground do not fully obstruct billboards in the background. The one drawback of using billboards is, when viewed from directly overhead, the billboards appear to be thin lines with no depth, much like playing cards on edge. It is possible to remedy this view by adding another billboard perpendicular to the first with an image of the tree crown as the texture. We would like to incorporate this into future versions of the visualization.

A hybrid view of GIS pixels and individual tree objects is an effective tool for understanding relationships among spatial datasets

One of the most interesting bits of feedback we received from our beta tests was that users liked to visualize the relationship between the 30-meter pixels and the rendered tree objects. We had assumed that users would want the most photo-realistic view of the forest when walking along the terrain. Instead, users were interested in the abstract view of the GIS data draped on the terrain surface coupled with the spatial depiction of tree objects. For example, users who would not know what a particular fuel model looked like on the ground were able to visualize the relative density and composition of live trees and snags that are expected in that fuel model using the tree objects.

This is perhaps even more relevant when comparing different modeled vegetation layers, from GNN or other sources, against known tree distributions. For example, a land manager could use stand exam data as ‘truth’ to spatially render tree objects and then cycle between multiple modeled layers to gauge individual model performance. Patterns that may not be readily evident in a traditional GIS (such as slope or aspect effects) would be more easily seen in the visualization environment. This hybrid view could also be used as a powerful visual communication tool, through screenshots or animations, when describing relationships between two datasets.

In-Game Communication, User Interfaces and Visualization Content

The visualization environment provides multi-user support while maintaining small server loads

The visualization environment is distributed as a stand-alone application packaged with all the spatial data required to view the environment. A user can run the visualization on his/her own computer from a Windows executable file by specifying a mission XML file, which sets various parameters of the visualization environment, including spatial extent, spatial layers and in-game markers. However, it is also possible for a user to use the game in a multi-user environment. At present, the networking model for running the application is that one user and their computer (i.e., a client) acts as the server and manages all communication packets among all clients. Other clients join the game by specifying a server name at the command line. Because all game content is resident with each client (i.e., each user has installed the application on their computer), the server is only

responsible for managing the chat among clients and for tracking clients' positions. This results in ‘thick client / thin server’ architecture, as each client is responsible for generating the visualization on their own computer.

In its source distribution, Delta3D provides core functionality for running applications in a multi-user mode, where multiple users interact in the same simulated environment. Unfortunately, this functionality is somewhat rudimentary and we needed to expand its capabilities to meet our project's needs. We have added in-game communication tools through a chat window, and commands for querying who is simultaneously in the visualization environment. We have also added symbols to represent users’ positions so that all users can see one another in the visualization environment.

One enhancement that we would like to incorporate into future versions would be a stand-alone server that is always running. As it is now, the server must also be in the game. This puts load stress on the server machine, which could be avoided. This feature would need to be implemented before the visualization environment could be used by many people simultaneously.

User interface elements are effective when incorporated into the heads-up display (HUD)

Even though the focus of this project was the visualization environment itself, there were a number of user interface elements that we needed to provide geographic and informational context to the environment. Initially, we conceived of providing basic GIS functionality through a separate but linked internet map server application. This would include navigation, query and display capabilities; typical functionality for map serving websites. As the project progressed, however, it became evident to us that having separate processes and windows was confusing and detracted from the immersive experience.

We have begun to incorporate the needed informational and analytical tools directly into the visualization as heads-up display (HUD) user interface elements. Included in the current version are drape and object layer control, a legend for the current drape image, and a dynamic attribute table which gives information about the user's current location as a tabular view. We have also designed new movement commands to navigate the visualization space, including simulated ‘jump-to’ and ‘fly-to’ motions. Finally, users can track their positions with information about current geographic location, elevation and heading. All of these HUD elements were designed to be non-intrusive and intuitive to use, so as to not detract from the main focus, i.e. the visualization itself.

In future releases of the application, we would like to incorporate an overview map to show player positions as well as adding the capability to query in-game objects for detailed attributes by selecting them with a mouse click (i.e., ‘picking’ functionality).

Translation of GIS data to game content is done through simple utilities

One of the most convenient features of the Delta3D engine is its ‘geographic awareness’ and ability to read in a variety of spatial data formats for rendering terrain features. This geographic awareness is not at all common in traditional computer game engines, and is in fact a very important advantage of Delta3D over other technologies.

Because Delta3D is based on the earth's latitude/longitude coordinate system, translation of GIS data to game content is straightforward. We have developed simple utilities to translate spatial data, with the intention that other users could use the utilities to likewise incorporate their own data into the visualization.

Delta3D renders elevation surfaces from Digital Terrain Elevation Data (DTED) files, which are one of three resolutions: level 0 (~900 meters), level 1 (~90 meters) and level 2 (~30 meters). For our visualization environments, we chose to use level 2 data to correspond with the native resolution of the GNN products. The Geospatial Data Abstraction Library (GDAL) provides simple utilities for converting elevation data in any format to 1-degree DTED tiles.

For draped surfaces, Delta3D provides software for reading in GeoTIFF files, which is a common raster format for most GIS packages. Again, GDAL provides functionality for re-projecting rasters and translating file formats, and we have developed custom scripts (AMLs) to create both categorical and continuous GeoTIFF files. In addition, the mission XML file allows the user to create simple legends for the draped surfaces.

Instructions for creating custom content for the visualization using our utilities are in Appendix 3.

Deliverables

We produced most of the deliverables listed in our proposal. However, a couple of items (most notably the site visits and reports) were contingent upon having a beta version of our software completed. The difficulty of finding a suitable game engine, as described in the key findings section, delayed our development schedule such that we were unable to complete these tasks. Now that we have a beta-tested application, released with this final report, we intend to solicit feedback from our stakeholders as they use the visualization. Not that we developed several products that are in addition to what we originally proposed, which are described in the second table below.

Copies of all our final products are available from our website (http://www.fsl.orst.edu/lemma/gnnviz). The directory structure and file names for final products are listed in Appendix 1.

Proposed	**Delivered**	**Status**
Annual Reports	Ohmann, Janet L., Matthew J. Gregory, and Timothy M. Holt. Final Report to the Governing Board, Joint Fire Science Program: Project 05-4-1-12	This document
	Ohmann, Janet. 2006. Progress report to the JFSP Governing Board (JFSP Project Number: 05-4-1-12)	Completed
GNNViz Software and Visualization Environments	We have created the base GNNViz application and example visualizations of our three study areas: the Inland Empire of northeastern Washington, the coastal province of western Oregon, and the west slope and foothills of the Sierra Nevada in California. Each example visualization is controlled by the modification of an XML file, which creates a flexible interface to the core application. With each study area, we have included a number of GIS layers (e.g. Landsat TM imagery, National Land Cover Database 1992 land classification, land ownership, etc.) as well as key variables from the GNNFire maps (e.g. Albini fuel model, vegetation class, crown bulk density, etc.). In addition, values of key variables from the GNNFire plot database can be viewed on-screen during the application based on the	Completed

	user's location. (See Appendix 4 for variable definitions.)	
Technical Reports	One report detailing the methodology used to translate GNN fuel mapping predictions into virtual landscapes.	Not completed
	One report outlining the technical aspects of embedding external processes (map server technology) into the virtual gaming environment. Because we do not incorporate external processes in the visualization environment, this deliverable is no longer applicable.	No longer applicable
Site Visits	Site visits to three study areas to instruct forest managers and fuel specialists how to use the visualization environment and collaborative tools.	Not completed

Additional Products	**Delivered**	**Status**
Presentations and Demonstrations	Holt, Timothy. "Lightweight and Innovative MMP Technologies for Serious Games." Game Developers Conference, San Jose, CA.	March 2007
	Holt, Timothy. "Serious Games Engine Shootout." Panel member, Game Developers Conference, San Jose, CA.	March 2007
	Holt, Timothy. "Games Get Serious: Computer Games for Visualization and More." Geovisualization: A Window to the Earth Surface, Structure and System lecture series. Corvallis, OR.	January 2007
	Holt, Timothy; Gregory, Matthew. Demonstration of application of computer gaming technology to forest visualization ('serious gaming'). College of Forestry Centennial Open House.	November 2006

	Holt, Timothy. "Games Get Serious: Computer Games for Visualization and More" Keynote presentation. GIS Day, Oregon State University. Corvallis, OR.	November 2006
	Holt, Timothy. "Using Game Technology to Present Scientific Data" Presentation to Software Association of Oregon Corvallis Chapter, Corvallis, OR.	November 2006
	Holt, Timothy; Ohmann, Janet. Demonstration of the GNNViz project to the Governing Board of the Joint Fire Sciences Program. Missoula, MT.	September 2006
	Holt, Timothy; Ohmann, Janet. Demonstration of application of computer gaming technology to forest visualization ('serious gaming'). College of Forestry payday coffee.	April 2006
	Holt, Timothy. "Healthcare and Forestry - Half-Life 2: Meet Serious Games Modding." Serious Games Summit. Washington, DC.	October 2005
Websites	GNNViz project website - http://www.fsl.orst.edu/lemma/gnnviz/	Completed
	GNNFire project website - http://www.fsl.orst.edu/lemma/gnnfire/	Completed
Beta-Testing and Collaboration	Oregon State University College of Forestry beta-test. Ten users participated in a beta-test of the application, intended primarily to test the utility of the multi-player functionality including collaborative viewing and chat, and to provide general feedback on the visualization experience.	September 2007
	Teleconference with ESRI federal team.	December 2006
Software Utilities and Methodologies	Cache building tool for pre-caching tree objects and GIS layers as game content (to avoid long delays in the main application). This is called “buildcache.exe” in the main directory.	Completed

	Methodology of converting GIS layers into game content using GDAL.	Will be released
	Extraction program which converts raster data into user-defined tiles with associated metadata. This was a utility developed to serve data needs when we were using the Torque game engine. It has no utility when using the Delta3D engine.	Available upon request
Users' Guide	Quickstart guide to the GNNViz visualization environment.	Included with DVD

Future Developments to the Visualization Environment

The following is a partial list of additions or enhancements to the GNNViz visualization that we would like to implement given more time and funding.

- Develop methods to dynamically update the visualization to view landscape changes, either real (e.g., maps from change detection that show fire or harvest) or simulated (e.g., output from a fire simulation model).
- Provide more explicit treatment and display of uncertainty information in the visualization. One approach would be to allow user to toggle between a detailed (smoothed) view and a 'pixelated' view.
- Incorporate vector data such as roads, streams, and political boundaries (states, counties, etc.), to improve the frame of reference for the visualization.
- Provide tools that allow users to interact with visualization objects within the game environment that have data associated with them. For example, a user could query a tree object and find out that it is a 20 cm Douglas-fir and that there are 12 such trees per hectare in the stand. Available tree information includes species, DBH, crown ratio, height, etc.
- Provide a regional perspective for querying the visualization. For example, a user could query the game to 'Spatially depict all the trees where the average stand diameter is within a given threshold from the one I'm clicking on.'
- Optimize the object rendering to allow display of trees and other objects over a larger area of terrain than is possible within constraints of the current game engine (i.e., display more trees over a larger area).

Glossary of terms and acronyms used in this report

2D
Short for two-dimensional.

3D
Short for three-dimensional.

Application Programmer Interface (API)
An API is a software library designed to allow programmers access to an application. An API is not necessarily the complete source code for the entire application, but rather just a set of 'hooks' that give a programmer access to the application via custom software.

Billboards
Billboards are 2 dimensional images displayed in a computer game. They are often drawn such that no matter which way a player views them, the same 2D image always faces towards the viewer. They can be used for displaying trees and other such objects by essentially displaying a picture of the object which always faces the camera.

Client / Server Architecture
Used in terms of describing the general responsibilities of clients and servers in a multi user networked game. The client is the program that is running on a user's machine and is how they connect into the shared environment of a multi-user game. The server is a centralized program that manages all client connections to it.

Commercial Off-The-Shelf (COTS)
COTS refers to the use of commercial products (including games) for some application. It implies that the product can be used as-is without much modification to meet some need.

Content
In a game context, the content refers to the 'stuff' that a virtual world is composed of. This might be virtual models of trees, the landscape and terrain and so forth. Usually content is not used to refer to things such as the user interface or underlying datasets.

Delta3D
Delta3D is a widely used and well-supported open source game and simulation engine. Delta3D is a fully-featured game engine appropriate for a wide variety of uses including training, education, visualization, and entertainment. Delta3D is unique because it offers features specifically suited to the Modeling and Simulation and DoD communities such as High Level Architecture (HLA), After Action Review (AAR), large scale terrain support, and SCORM Learning Management System (LMS) integration.

Density Surface
A 2D image that represents the count or frequency of a some datum over a surface. It is typically represented with a grayscale image, where black represents no value, up to white which means the maximum number of items occurring in that spot. In the context

of GNNViz, we use a density surface that represents the number of trees per hectare in a given area. This density surface is then used to compute the placement of decorator objects (tree models and/or billboards) onto the terrain, with more objects being placed in areas with higher density values, and fewer in low density areas.

Game Tiling
Tiling is the process of breaking large pieces of content such as terrain into a series of smaller tiles of some fixed size. Tiling allows the game engine to only render and keep in memory portions of the terrain that are near to and visible to the player. As the player moves through the terrain, new tiles are dynamically loaded as they come into view, and old tiles are removed from memory when they become more distant.

GNN
The Gradient Nearest Neighbor method of predictive vegetation mapping, named for its use of direct gradient analysis and nearest-neighbors imputation.

GNNViz
Informal name of this project, which developed an application to visualize regional maps of vegetation and fuels developed using GNN.

GNNFire

Heads-Up Display (HUD)
A method of displaying numerical or other information on the view into a real or virtual space by overlaying the information in front of the viewer's eye. In an airplane, this might be information displayed directly onto the plane cockpit glass. In a game it is typically drawn on the computer screen as if it is projected in front of the viewer's eyes.

Imputation
A class of statistical methods used for prediction of missing data. An increasingly common application in forestry is for imputing vegetation attributes to pixels or polygons for which the data are lacking. Also referred to as nearest-neighbors methods.

Level Of Detail (LOD)
Level of Detail is a technique used by many game engines to help improve performance by drawing lower resolution versions of some objects when detail is not required. This is usually done by creating multiple versions of a model or object, varying from high detail down to very simple detail. The high detail version is displayed only when the player's point of view is close to the object, and progressively lower level detail versions are drawn as the player's point of view moves farther away.

LEMMA
Landscape Ecology, Modeling, Mapping, and Analysis team, of which Ohmann, Gregory, and Holt are members. See http://www.fsl.orst.edu/lemma.

Mission XML File
An XML file used by GNNViz which contains specific data about what terrain data to use, tree model rendering methods, and images to be overlaid on the terrain's surface.

MMOG (Massive Multiplayer Online Game)
A multiplayer game which is played online via the internet. Typically these games take place in large and persistent worlds where many players share in the game experience together. Popular commercial games such as World of Warcraft are examples of MMOGs. They also are referred to at times as MMORPGs (Massive Multiplayer Online Role Playing Games) or just MMOs (Massive Multiplayer Online).

MOD
Short for modification. Modding is the act of taking an existing COTS computer game and modifying it to create a new game by changing the game content and artwork, sounds or even game behavior via an API. Many game companies actually sanction and encourage the public to mod their games, but do require anyone running a mod to own a copy of the original game.

Mouse look
Mouse Look refers to a common game interaction method, where moving the mouse (either with or without an associated mouse click) changes the player's view. They do not actually change position on mouse clicking, but the view is moved about as if the player is looking around.

Multi-user mode
Wherein more than one user/person may join and interact inside of a game world. Typical of an MMOG (Massive Multi-player Online Game)

Networking model
The method by which a multi-user game such as GNNViz or an online game shares data between different players. It is necessary for there to be some kind of underlying networking model so that if one users moves, types a 'chat' message or does some kind of action in the game, other users will see a representation of the other user moving, display their chat message and so forth.

OpenGL
OpenGL is a method for rendering 3D images. GL stands for Graphics Language. OpenGL is the 'open source' version of GL. id software is a huge proponent of OpenGL because it is platform-independent, unlike DirectX which is Windows-specific.

OpenSceneGraph
OpenSceneGraph is an open source high performance 3D graphics toolkit, used by application developers in fields such as visual simulation, games, virtual reality, scientific visualization and modeling. Written entirely in Standard C++ and OpenGL it runs on all Windows platforms, OSX, GNU/Linux, IRIX, Solaris, HP-Ux, AIX and FreeBSD operating systems.

Open source
Software made available to the public for free where the complete source code is also made available. The user base of such a program is often encouraged to contribute to the source code so that it in a sense becomes a community written project. Typically open source projects restrict the use of the code for commercial applications.

Picking functionality
The ability to click on an object or point with the mouse and select it. Picking something typically is to either mark it for some future action or to request more information on the selected point.

Software development kit (SDK)
A programming package that enables a programmer to develop applications for a specific platform. Typically an SDK includes one or more application program interfaces (APIs), programming tools, and documentation.

Source engine
The Source Engine is a game engine created by Valve Software for use in their Half-Life 2 and other computer games. Source was actually an internal nickname for the engine used by Valve's developers, which was later used and adopted by the public.

Thick client / thin server architecture
This is a client/server architecture where most of the computational work is put on the individual clients rather than on the centralized server. With GNNViz for example, the client is responsible for all terrain rendering, calculation of tree positions and other such work. The server does very little work beyond just acting as the central exchange point for allowing clients to communicate with each other.

Tree models
Models are shapes defined in 3 dimensions and used in a game or other virtual environment. 3d models can typically be viewed from multiple angles. In the context of GNNViz, a tree model would be a 3 dimensional representation of a tree, showing trunk, branches, foliage and other physical aspects of the tree.

WASD
WASD refers to the W, A, S and D keys on a keyboard, which are commonly used in games for moving the player. The W key, when pressed, moves the player's view forward. S moves the view backwards, and A and D move the view left and right respectively. The layout of these keys is similar to the four-arrow key 'inverted T' layout on most keyboards.

Appendix 1: DVD contents

The GNNViz reports, application, and data are distributed on four DVDs due to the large filesize of the accompanying data. Disk 1 contains the base program and reports. The other disks contain data for the GNNViz application for the three study areas: Disk 2 is California, Disk 3 is Oregon, and Disk 4 is Washington. File structure is listed in the table below for Disk 1 and 2 only – Disk 3 and 4 follow the same format and naming conventions as Disk 2. Each disk has a "README_DISK*.txt" file describing disk contents.

Disk1_Program files	**Type**	**Description**
Reports		
\reports\final_report_01-4-1-12_coverletter.pdf	pdf document	Cover letter for this report.
\reports\final_report_01-4-1-12_28sep07.pdf	pdf document	This report.
\reports\GNNViz_User_Guide.pdf	pdf document	User's guide for the GNNViz application.
Program files		
\Delta3d\ext*	Dynamic link libraries	External linking libraries distributed with Delta3D
\Delta3d\gnnviz\bin*	Dynamic link libraries	Main GNNViz executable program plus linking libraries distributed as core from Delta3D (modified for GNNViz)
\Delta3d\gnnviz\bin\build_california.bat	batch file	Prebuilds the GNNViz cache files for the California

		project area
\Delta3d\gnnviz\bin\build_oregon.bat	batch file	Prebuilds the GNNViz cache files for the Oregon project area
\Delta3d\gnnviz\bin\build_washington.bat	batch file	Prebuilds the GNNViz cache files for the Washington project area
\Delta3d\gnnviz\bin\buildcache.exe	executable	GNNViz utility program which prebuilds all cache files needed by GNNViz
\Delta3d\gnnviz\bin\gnnviz.exe	executable	The GNNViz program
\Delta3d\gnnviz\bin\setup.bat	batch file	Sets environment variables needed by gnnviz.exe and buildcache.exe. Note that this must be edited for each installation to set path values.
\Delta3d\gnnviz\bin\california.bat	batch file	Starts GNNViz with the California dataset
\Delta3d\gnnviz\bin\oregon.bat	batch file	Starts GNNViz with the Oregon dataset
\Delta3d\gnnviz\bin\washington.bat	batch file	Starts GNNViz with the Washington dataset
\Delta3d\gnnviz\data\mission_california.xml	XML file	Mission file for typical California visualization
\Delta3d\gnnviz\data\mission_california_all.xml	XML file	Mission file for pre-building all California spatial data tiles
\Delta3d\gnnviz\data\mission_oregon.xml	XML file	Mission file for typical Oregon visualization
\Delta3d\gnnviz\data\mission_oregon_all.xml	XML file	Mission file for pre-building all Oregon spatial data tiles

\Delta3d\gnnviz\data\mission_washington.xml	XML file	Mission file for typical Washington visualization
\Delta3d\gnnviz\data\mission_washington_all.xml	XML file	Mission file for pre-building all Washington spatial data tiles
\Delta3d\gnnviz\inc*	Source code	.h include files for GNNViz libraries and modules
\Delta3d\gnnviz\src*	Source code	Source code for GNNViz and associated modules and libraries
\Delta3d\gnnviz\VisualStudio*	Source code	Microsoft Visual C++ project files for GNNViz
\Delta3d\inc*	Source code	Include files for Delta3D
\Delta3d\lib*	Source code	Library files for Delta3D
\Delta3d\src*	Source code	Source code for Delta3D
\Delta3d\VisualStudio*	Source code	Microsoft Visual C++ project files for Delta3D
Disk2_CA_Data		
\Delta3d\gnnviz\data\gnnviz\california\ca_fcsum.xml	XML file	GNN attribute data for California
\Delta3d\gnnviz\data\gnnviz\california\ca_fcsum.xsd	XSD file	GNN attribute XML schema for California
\Delta3d\gnnviz\data\gnnviz\california\cancov.tif	GeoTIFF	Canopy cover (percent) (see Appendix 4)
\Delta3d\gnnviz\data\gnnviz\california\dvph_ge_25.tif	GeoTIFF	Down wood volume (m^3/ha) >= 25cm intercept diameter (see Appendix 4)

\Delta3d\gnnviz\data\gnnviz\california\fuelmodel.tif	GeoTIFF	Albini fuel model (see Appendix 4)
\Delta3d\gnnviz\data\gnnviz\california\nlcd.tif	GeoTIFF	1992 National Land Cover Data (NLCD) land cover type (see Appendix 4)
\Delta3d\gnnviz\data\gnnviz\california\nnplt.tif	GeoTIFF	GNN neighbor plot assignment (used to link attribute data to current position)
\Delta3d\gnnviz\data\gnnviz\california\owner.tif	GeoTIFF	Generalized ownership classes (see Appendix 4)
\Delta3d\gnnviz\data\gnnviz\california\rcbdu2.tif	GeoTIFF	Canopy bulk density (kg/m3) using uncompacted crowns and Rocky Mountain equations (see Appendix 4)
\Delta3d\gnnviz\data\gnnviz\california\rhtcbu2.tif	GeoTIFF	Height to crown base (m) using uncompacted crowns and Rocky Mountain equations (see Appendix 4)
\Delta3d\gnnviz\data\gnnviz\california\shrcov.tif	GeoTIFF	Uncorrected shrub cover (percent) (see Appendix 4)
\Delta3d\gnnviz\data\gnnviz\california\stph_ge_12.tif	GeoTIFF	Snags (no. trees/ha) >= 12.5 cm diameter at breast height (DBH) (see Appendix 4)
\Delta3d\gnnviz\data\gnnviz\california\tmstack.tif	GeoTIFF	Landsat TM composite image (Bands 4,5,3) from 2000
\Delta3d\gnnviz\data\gnnviz\california\v_stph_ge_12.tif	GeoTIFF	Snags (no. trees/ha) >= 12.5 cm DBH (used to create tree objects)
\Delta3d\gnnviz\data\gnnviz\california\v_tphc_ge_3.tif	GeoTIFF	Conifer trees (no. trees/ha) >= 2.5 cm DBH (used to create tree objects)
\Delta3d\gnnviz\data\gnnviz\california\v_tphh_ge_3.tif	GeoTIFF	Hardwood trees (no. trees/ha) >= 2.5 cm DBH (used to create tree objects)

\Delta3d\gnnviz\data\gnnviz\california\vegclass.tif	GeoTIFF	Vegetation class (Johnson, DH, and TA O'Neil. 2001, eds. Wildlife-habitat relationships in Oregon and Washington. Oregon State University Press; Corvallis, OR; 736 p.)
Disk3_OR_Data		
\Delta3d\gnnviz\data\gnnviz\oregon*		Same files as Disk2 but for Oregon.
Disk4_WA_Data		
\Delta3d\gnnviz\data\gnnviz\washington*		Same files as Disk2 but for Oregon.

Appendix 2: Game engine comparison for use with the GNNViz application

Selection of a game engine is perhaps the most crucial and important step in creating any kind of visualization tool such as GNNViz. Considerations can include the cost of the underlying technology, the availability of a software development kit (SDK), its extensibility, and even the quality and size of the developer community using the technology.

As with any rapidly growing field of technology, there are always new technologies and industry trends not yet available that show great promise. Chief among these for projects such as GNNViz are technologies for Massive Multiplayer Online Games (MMOGs) such as World of Warcraft. MMOGs are typically games covering huge areas with many players.

Stability and support of the underlying technology

An important, if not somewhat abstract factor in choosing a game engine is the stability and support of the technology from the original game developer and also the community of developers using the technology. The list of game engines available is littered with half-finished and abandoned open-source projects that at one time showed promise, but never were completed.

In general, the presence or absence of a thriving developer community is a good sign of the relative health and usability of a game engine. Valve's Source Engine for example has an extremely healthy developer community, and is also well supported by the engine developers themselves. Conversely, the Garage Games Torque Game Engine Advanced does have a fairly active developer community, but it is not well supported by the engine developers, which makes it a less desirable choice.

Support of game companies for non-entertainment and non-commercial projects

In general, game companies offer minimal support and show little interest for non-standard applications of their technology such as proposed for GNNViz. There are a few exceptions to this rule, such as the Multiverse company, which is very open to talking and working with developers of atypical projects such as ours.

One important trend that many commercial game companies support is the idea of modding of their games. Modding is the act of taking an original commercial game and modifying it to create a new game. Modding might be as simple as changing the artwork associated with a game to make it look different, or it might be a complex task of changing the underlying game's look and behavior to create a whole new game.

Modding always comes with one limitation: those that develop mods are not allowed to commercialize their work (sell their mod), and anyone wishing to play a mod must first buy and own a copy of the original game. This protects the commercial rights of the

original game company, while allowing modders the freedom to create and explore new kinds of games without an excessive up-front cost for engine technology licensing.

Specific features required for GNNViz

We analyzed seven primary factors while considering different game technologies for GNNViz: the cost to use the technology, support for multiple players in the same shared environment, the size of the terrain that could be rendered, support for geo-referenced data (data in latitude/longitude space), ability to render vegetation, access to an SDK and developers tools, and the amount of source code available for modification.

In some cases it is possible to work around game engine limitations. For example, with the Valve Source Engine, it is possible to effectively scale the player's view of the world in order to create the illusion of a very large space. However, other limitations such as the lack of multi-player support cannot be overcome without full access to the underlying game engine source code and a lot of hard work on the developer's part.

Game technologies examined for GNNViz

The table below summarizes game technologies that we examined for use with GNNViz. Since many game companies allow and encourage modding of their games, it was relatively easy and inexpensive to explore the technologies with a minimal up-front cost. It should be noted that, unless otherwise specified, all technologies listed below work only work in the Windows PC environment.

Engine	**GNNViz Specific Features Support**	**Notes**
Valve Source Engine	Cost: $25 Multi-player: Yes Terrain: Limited, on the order of 1 kilometer in extent Geo-referenced data support: No Vegetation: Yes	Would be an excellent choice for creating highly detailed visualization of smaller forested areas. However, terrain limitations prevent its use for large-scale visualization. Offers an extensive SDK and a supportive developer community. Wide installation base with over 8 million registered users playing games on this platform. *http://developer.valvesoftware.com*

	SDK: Yes Source code: SDK only	
Ubisoft FarCry Game Engine	Cost: $25 Multi-player: Yes Terrain: large scale, on the order of 10s of kilometers Vegetation: Yes SDK: Yes Source code: SDK only	Very easy to modify and work with. Underlying game engine technology limits the maximum range between highest and lowest elevation to 256 meters. This limitation alone prevented us from using it for GNNViz, as we were not able to override this limitation via the SDK. Note that Ubisoft is soon to release a new version of this engine that shows great promise for large-scale visualization environments. *http://www.crymod.com*
Multiverse MMO Engine	Cost: Free Multi-player: Yes Terrain: Extremely large scale, on the order of 100s of kilometers Geo-referenced data support: No Vegetation: Yes, but with limited control over density and placement SDK: Yes Source code:	Has great potential for use as a large scale visualization platform. However, it does not currently offer any way to control precise placement of vegetation, which precludes being able to accurately place vegetation based on GNN data. Business model of developer allows free use of their technology and only requests payment if the developer charges for their game. Company is extremely open to discussing non-traditional projects such as this one. Supports SpeedTree library for high fidelity tree rendering. *http://www.multiverse.net*

	SDK only	
Oblivion Game Engine	Cost: $40 Multi-player: No Geo-referenced data support: No Terrain: Large scale, on the order of 10s of kilometers Vegetation: Yes SDK: Yes Source code: SDK only	Would be effective for a medium-scale visualization tool, but does not allow multiple players to be in the same shared virtual environment. Supports SpeedTree library for high fidelity tree rendering. *http://cs.elderscrolls.com/constwiki/index.php/Oblivion_Mods_FAQ*
Garage Games Torque Game Engine Advanced	Cost: $1500 Multi-player: Yes Geo-referenced data support: No Terrain: Large scale terrain, on the order of 10s of kilometers Vegetation: Yes SDK: Yes Source code: Full	Has good potential. However, the underlying code is incomplete and contains various long-standing bugs. Integration of real world data requires extensive 'hand holding' and can't be automated. Does not work with geo-referenced data and latitude/longitude coordinate systems. Is able to effectively render very large numbers of trees over the area of terrain it does support.
Delta3D	Cost: Free Multi-player:	An open-source engine used by the US military and other research groups for game and simulation projects.

	Rudimentary Geo-referenced data support: Yes Terrain: Extremely large scale, essentially infinite Vegetation: Yes SDK: Yes Source code: Full	Extremely easy to integrate geo-referenced data such as DTED height maps and GeoTIFF images. Use of such data is virtually trouble-free and little to no data massaging is required. Is able to work directly with coordinates and data referenced in latitude/longitude space. Multi-player support is somewhat rudimentary. Capable of running on the Linux platform, so is not restricted to the Windows environment.
BigWorld MMO Engine	Cost: $500,000 Multi-player: Yes Terrain: Extremely large scale Geo-referenced data support: No Vegetation: Yes SDK: Yes Source code: SDK and Full	Offered as an example of a high-quality and high-cost game engine that would be an excellent choice for a large-scale visualization. However, its cost is prohibitive most projects.

Other sources of game engine comparisons

The above list is not meant to be a comprehensive list of all game engines that could be used or were considered. Many engines were deemed lacking in features or support. The following online resources were invaluable in our selection process.

The DevMaster.Net 3D engines database, online at *http://www.devmaster.net/engines/index.php*.

The GPWiki Game Engines list, online at
http://www.gpwiki.org/index.php/Game_Engines

Appendix 3: Process for user to create their own visualization by importing their own data

For our current version of the visualization application, we created a methodology to translate any raster data in any projection into GeoTIFF files for use in Delta3D. All of our raster data was stored in ArcInfo GRID files, and we used utilities to translate from this format into GeoTIFFs. However, because this approach requires that users have ArcInfo and several other utilities installed on their computers, we ultimately decided not to distribute these utilities and instructions as part of our final report. Instead, we plan to continue development on a data translation utility that will be based on Python and distributed as an executable, which will be more generic and able to accommodate a variety of spatial data formats.

We will post any new utilities to our website at http://www.fsl.orst.edu/lemma/gnnviz. Any questions about the methodology should be referred to Matt Gregory at matt.gregory@oregonstate.edu.

Appendix 4: Data dictionary

The following vegetation and fuels variables are included with the GNNViz visualization environment.

Variable name	Description
FCID	Unique forest class identification number.
BAA_GE_3	Basal area (m^2/ha) of all live trees >=2.54 cm DBH
BAC_GE_3	Basal area (m^2/ha) of all live conifers >=2.54 cm DBH
BAH_GE_3	Basal area (m^2/ha) of all live hardwoods >=2.54 cm DBH
QMDA_DOM	Quadratic mean diameter (cm) of all dominant and codominant trees.
CANCOV	Canopy cover (percent) of all live trees, calculated using methods in the Forest Vegetation Simulator (Crookston, NL, and AR Stage. 1999. Percent canopy cover and stand structure statistics from the Forest Vegetation Simulator. RMRS-GTR-24. 8 pp.)
TPH_GE_3	Density (no. trees/ha) of all live trees >=2.54 cm DBH
TPHC_GE_3	Density (no. trees/ha) of all live conifers >=2.54 cm DBH (*V_TPHC_GE_3* is used to create tree objects)
TPHH_GE_3	Density (no. trees/ha) of all live hardwoods >=2.54 cm DBH (*V_TPHH_GE_3* is used to create tree objects)
STPH_GE_12	Density (no. trees/ha) >= 12.5 cm diameter at breast height (DBH) and >=2.0 m tall (*V_STPH_GE*_12 is used to create tree objects)
DVPH_GE_25	Volume (m^3/ha) of down wood >=25.4 cm diameter at intercept and >=1 m long. [R5 plots limited to pieces >=3.0 m long. FIA plots do not include decay class 5 logs.]
SHRCOV	Uncorrected shrub cover (percent) that is the cumulative cover of all shrubs (may be >100%).
VEGCLASS	Vegetation class (from Johnson, DH, and TA O'Neil. 2001, eds. Wildlife-habitat relationships in Oregon and Washington. Oregon State University Press; Corvallis, OR; 736 p.). *BAH_PROP* is the

	proportion of total live tree basal area that is hardwood. 1 Sparse (*CANCOV* <10) 2 Open (*CANCOV* 10-39) 3 Broadleaf, sap/pole, mod/closed (*CANCOV* >=40, *BAH_PROP* >=0.65, *QMDA_DOM* <25 cm) 4 Broadleaf, sm/med/lg, mod/closed (*CANCOV* >=40, *BAH_PROP* >=0.65, *QMDA_DOM* >25 cm) 5 Mixed, sap/pole, mod/closed (*CANCOV* >=40, *BAH_PROP* 0.20-0.64, *QMDA_DOM* <25 cm) 6 Mixed, sm/med, mod/closed (*CANCOV* >=40, *BAH_PROP* 0.20-0.64, *QMDA_DOM* 25-50 cm) 7 Mixed, large+giant, mod/closed (*CANCOV* >=40, *BAH_PROP* 0.20-0.64, *QMDA_DOM* >50 cm) 8 Conifer, sap/pole, mod/closed (*CANCOV* >=40, *BAH_PROP* <0.20, *QMDA_DOM* <25 cm) 9 Conifer, sm/med, mod/closed (*CANCOV* >=40, *BAH_PROP* <0.20, *QMDA_DOM* 25-50 cm) 10 Conifer, large, mod/closed (*CANCOV* >=40, *BAH_PROP* <0.20, *QMDA_DOM* 50-75 cm) 11 Conifer, giant, mod/closed (*CANCOV* >=40, *BAH_PROP* <0.20, *QMDA_DOM* >75 cm)
FUEL_MODEL	Fire behavior fuel model. Classification rules were modified slightly from those in documentation for the appropriate variant of FVS Fire and Fuels Extension and to reflect the fuels and vegetation variables available in the GNNFire database and local expert knowledge. 1 short grass 2 timber (grass and understory) 5 brush 6 dormant brush, hardwood slash 8 closed timber litter 9 hardwood litter 10 timber (litter and understory) 11 light logging slash 12 medium logging slash 26 modified chaparral (California)
PCBDU2	Canopy bulk density (kg/m^3) computed with vertical layering method, ‘uncompacted’ crowns, and using equations for the coastal Pacific Northwest (used for Oregon).
RCBDU2	Canopy bulk density (kg/m^3) computed with vertical layering method, ‘uncompacted’ crowns, and using equations for the northern Rocky Mountains (used for California and Washington).

PHTCBU2	Height to crown base (m) computed with vertical layering method, ‘uncompacted’ crowns, and equations for the northern Rocky Mountains (used for California and Washington).
RHTCBU2	Height to crown base (m) computed with vertical layering method, ‘uncompacted’ crowns, and equations for the northern Rocky Mountains (used for California and Washington). Code ‘999' is assigned where there is minimal crown.