# MANAGING DATA AND COMPUTATIONAL COMPLEXITY FOR IMMERSIVE WILDFIRE VISUALIZATION

Michael A. Penick[1,2] Roger V. Hoang[1,2]
Frederick C. Harris Jr.[1,2] Sergiu M. Dascalu[1]
Timothy J. Brown[2] William R. Sherman[2] Philip A. McDonald[2]

[1]University of Nevada, Reno
Department of Computer Science and Engineering
Reno, NV 89557

[2]Desert Research Institute
Center for Advanced Computation and Modeling
Reno, NV 89512

penick@cse.unr.edu

**KEYWORDS**
Virtual Reality, Wildfire Simulation, Immersive Systems

## ABSTRACT

Wildfires are a concern for communities throughout the world. They cause millions of dollars in damage and lead to loss of lives. The development of computational models to predict wildfire behavior is necessary to minimize wildfire damages and casualties. Visualizing the data generated from these computational models has many applications including training, strategic planning, data analysis, and model validation. The complexity of visualizing wildfire brings many challenges, further complicated by large datasets and specialized hardware used to drive immersive systems. This paper present methods for managing the large datasets and computational complexity involved in visualizing large wildfires in immersive environments.

## INTRODUCTION

Wildfires are very unpredictable. It is difficult to determine exactly where and when a wildfire will happen next and it is even more difficult to determine how a wildfire will spread with absolute precision. It is the unpredictability of wildfires that make them so dangerous. It is this reason that so much time and money is spent researching wildfire behavior.

There are many advantages to modeling the spread of wildfires. Spread models can be used to develop plans to fight fire, initiate more predictable prescribed burning, and also predict the risk involved if a wildfire occurred in a certain area. Determining how much risk to an area's inhabitants and their property can be used to spend money appropriately and develop a proactive plan for evacuation and fire fighting. Kyle Canyon in Southern Nevada is a good example of a high danger wildfire zone. In the event of a wildfire, it would be very difficult, if not impossible, to evacuate its citizens and would most likely end in a high fatality rate. Many decisions rely on the results of wildfire spread model simulations. It is important that these models, to some degree, accurately predict the spread.

Validating these models is difficult without the wildfire actually happening and comparing the results. Visualization of the wildfire model output in an immersive environment can be used to validate its output against environment factors such as terrain slope, fuel moisture, wind vectors and weather conditions. It can also be used to compare model output against video footage or a visual recreation of the scene from collected data. This is only a single but important reason for visualization of wildfire model output. Visualization of these model outputs can be used to better train firefighters and fire bosses and aid in burning more predictable prescribed fires.

Burning a wildfire for the purpose of training is dangerous and costly. Virtual reality technology makes it possible to recreate wildfire scenarios with more realistic results then previously possible. Recreating wildfire or using model outputs can be used to train fire bosses and firefighters and aid in the development of plans and precautions. With the development of a real-time wildfire simulation it would be possible to run through several virtual scenarios very quickly. This could be used to better determine what measures to take while burning a prescribed fire.

Real-time visualization of this data is a necessary requirement for these applications. Faster than real-time visualization would also bring many advantages. Wildfires often burn over large areas of land even covering tens to hundreds of thousands of acres resulting in the need to visualize of several large datasets. Visualizing terrain and forests of this magnitude is a computationally intensive task while maintaining real-time frame rates. Rendering a fire across this vast landscape also brings many challenges.

The rest of this paper is structured as follows. The next section presents related work on wildfire visualization, virtual reality, possible hardware configurations, and our software environment. This is followed by a section describing our implementation of different parts of the wildfire scenario. The paper finishes with our conclusions and possibilities for future work.

## BACKGROUND

### Fire and Wildfires

Much work exists for visualizing fire for the purpose of training and analysis. However, little work has been done to visualize fires and wildfires in an immersive medium. A good amount of work as been spent visualizing computational models for in-building fires (Bukowski and Sequin, 1997; Govindarajan et al., 1999). (Julien and Shaw, 2003; Tate et al., 1997) use virtual environments and realistic spread models for application in firefighting training scenarios; however, the models and fire visualization are not applicable to outdoor, large-scale fires.

Los Alamos National Laboratory developed a tool for the visualization of wildfire data; however, both their model and visualization ran slower than real-time (Ahrens et al., 1997; McCormick and Ahrens, 1998). This work states the applicability of visualization of wildfire to training, but only describes the graphical elements necessary. Similar work using GIS based reconstruction of forest landscape scenarios has been done with non-immersively visualized wildfires (Yu et al., 2004). The authors chose to implement their own elliptical wildfire spread model based on the Huygen's principle of wave propagation and compute localized fire behavior using the Rothermel model. They achieved real-time frame rates on a desktop system using a custom forest level-of-detail system and wildfire spread model. However, the paper did not discuss the validity of their wildfire spread model and did not address the application or complexities of their application in an virtual environment.

## Farsite

FARSITE is a well-established fire behavior and growth simulator developed by the USDA Forest Service. It is used by fire analysts from the US Department of the Interior, National Park Service, US Department of the Interior Bureau of Indian Affairs. (Community, 2007) Its importance and widespread use among fire professionals was a critical factor for choosing to visualize its simulation output. FARSITE incorporates models of surface fire, crown fire, point-source fire acceleration, spotting, and fuel moisture to calculate the spread of fire of a landscape (Finney, 1998). These various models are used to propagate vectors of fire parameter polygons. Intervals of these expanding polygons are interpolated to generate raster data that describe fire behavior. The raster data outputs are stored in ESRI ascii files, of which six of the outputs are of importance to visually reconstructing the wildfire scenario. Several of the inputs required to run a FARSITE simulation are also crucial to visualizing the wildfire scenario. This includes the digital elevation model data used to render the terrain and the fuel load data used to place vegetation.

## Virtual Reality

Virtual Reality (VR) as it relates to this project is the use of hardware and software technologies used to allow user to view and interact with computer-simulated environments. The goal of VR is to mentally immerse a user within these environments through different types of sensory feedback. Sight (visual), a type of sensory feedback important to this, but in a broader sense feedback also includes aural (sound), touch (haptic), and smell (olfactory). The broad definition of VR includes the visualization of a 3D world on a desktop computer because this can and often creates a sense of immersion. However, the specific definition used in this project involves the visualization on stereoscopic displays

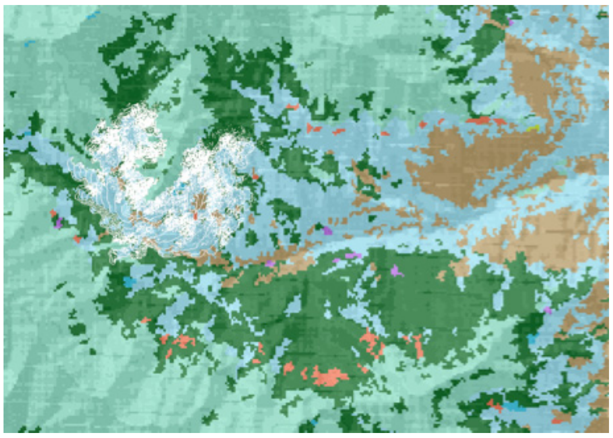

Figure 1. FARSITE Data Visualization

and interaction captured by tracking systems because these technologies offer many advantages over visualization on a desktop system. The source of these advantages is through the support of depth cues not achievable on non immersive systems.

Depth cues are the important information a viewer uses to discern distances between objects in a scene. The more depth cues a system can support the more potential for immersion is possible. Monoscopic, stereoscopic, and motion depth cues are important to VR systems. Monoscopic depth cues are achievable on both immersive and desktops systems. Monoscopic depth cues can be seen in a single static image of a scene. These are the depth cues, which can be drawn from image features such as size, shading, and occlusions. Stereoscopic image depth cues are the differences determined between the images obtained by each eye (left and right images). Motion depth cues can be seen when a viewer changes the relative position between them and an object. The viewer can gauge the distance of an object based on how fast an object passes by when they change perspective (closer objects appear to move faster than farther ones). A 3D desktop environment is only able to provide monoscopic depth cues, but an immersive system with stereoscopic displays and head tracking can simulate all depth cues discussed. The addition of these depth cues allows for an experience closer to reality, making such an environment suitable for training applications.

## Immersive Hardware Systems

Multi-screen display systems require specialized hardware unlike HMDs, which can use hardware similar to desktop PCs. Multi-screen display systems require multiple graphics pipes to keep real-time frame rates. The goal is to keep performance independent of the number of screens a system contains. A 6-wall system should have comparable performance with a 4-wall system if driven by similar hardware. Two configurations exist for achieving this goal: The first is a shared memory system with multiple graphics pipes and the second, more recent, is provided by cluster-based systems. Shared memory systems support a single large memory image across all processors. On these systems a process is used to render each screen independently and one to many processes are used to update the simulation. Rendering each screen in parallel offers performance, which is independent of the number of screens in the system. Performance is further increased by allowing the simulation and rendering to run in parallel. The idea is to run the update and rendering computation in parallel as much as possible, but because they are accessing the same data, the update writing and the rendering reading it must be locked. Cluster-based solutions run the simulation and rendering code on a node for each screen in parallel and a head node keeps the simulations in sync. It is also possible on these systems to run the rendering and simulation code in parallel for increased performance if multiple processors are available. The shared memory solution has the disadvantage of requiring expensive specialized hardware to support multiple graphics cards. High performance, available commodity hardware can be used in cluster-based systems with a fast interconnect.

### VR Toolkits and Scene Graphs

VR systems contain specialized input and tracking hardware, but also specialized screen and computational hardware configurations making writing software for these systems a monumental task. The hardware configurations for these systems can vary drastically between different organizations requiring software to be changed for each of these systems. VR toolkits attempt to abstract these differences so that an application can be written once and run on all of these systems. The largest difference between VR toolkits consists of the type of input hardware and computational configurations they support. William R. Sherman's FreeVR supports a variety of input and tracking systems such as common desktop game pads all the way to high end tracking system such as InterSense's IS-900, but only supports shared memory systems. VRUI and VR Juggler supports similar tracking hardware, but also support cluster-based systems.

Scene graphs uses many optimizations to speed up rendering. Many of these offer different types of culling which quickly determine the visibility of geometry and reduce the amount of triangles need to be rendered. Frustum culling removes geometry outside of the viewers point of view. Occlusion culling removes geometry which is not visible because it is behind other geometry in the scene. Small feature culling removes geometry which are smaller than a particular amount of screen space in pixels. Scene graphs also reduce the amount of expensive state changes such as changing shaders, textures, and other rendering states. They also often implement lazy state updating, which only updates states that are not already set. Optimization traversal usually run during a single time after initialization can optimize a the scene graph data structure and the geometry contained within. These optimizations can include organizing the scene graph into an octtree for optimized frustum culling or organizing triangle-based geometry into optimized triangle strips.

Although there are many scene graphs, there are few which are open source and suited to the development of virtual reality applications. Support for multi-pipe rendering is the fundamental feature necessary for rendering on our shared memory system. This support includes management of OpenGL objects (Texture object, Vertex buffer object, etc.) and, less importantly, a data protection mechanism. Data protection is often very specific to a problem to achieve high performance. Both OpenSG and OpenSceneGraph are two scene graphs that meet this criteria. OpenSG implements a system, which allow the scene graph to be transparently shared across multiple machines in a cluster or server processes on a single machine (details about the implementation of this system can be found in these papers (Reiners et al., 2002)). OpenSceneGraph does not have the ability to shared the scene graph data structure, but must be protected using an external locking mechanism or the rendering and simulation update must be done sequentially.

### Problem: Visualization of Large Wildfires

Frequently, wildfire can cover large landscapes many times beyond the current computational and memory capacity of current visualization hardware. Each graphical element such as terrain, vegetation, and fire is associate with a large dataset either describing the landscape or driving the simulation. Each one of these element has its own challenges and performance bottlenecks. The digital elevation model data and satellite image describe the terrain. The fuel load data is used to construct the vegetation environment. The wildfire is spread according to FARSITE outputs. Our goal is to achieve real-time visualization of wildfire that cover vast landscape each element much be managed to maintain visual fidelity and run in real-time.

## PROPOSED SOLUTION

### FreeVR and OpenSceneGraph

We built our application on the open-source FreeVR and OpenSceneGraph (OSG) libraries. The FreeVR virtual reality integration library is a cross-display VR library with built-in interfaces for many input and output devices. It allows programmers to develop on a standard desktop machine, with inputs and display windows that simulate a projection or headbased immersive system. The application can then run just about any type of VR system. The OpenSceneGraph library is used to help with world rendering. OSG allows 3D objects to be hierarchically organized within the environment, and also provides a system that optimizes the rendering through the use of various culling and sorting techniques.

A considerable amount of our effort thus far has been in writing the software interface between FreeVR and OSG. FreeVR works naturally well with OpenGL and other lower level graphical rendering libraries. However, when interfacing a VR integration library with a higher level rendering API there are many issues that need to be addressed, in particular 1) dealing with the perspective matrices, 2) shared memory allocation, 3) multiprocessing, and 4) windowing and input device interfacing. A software interface between FreeVR and the SGI Performer scene-graph library already existed, so we felt confident that the similar OSG library would not be too difficult. The Performer library was avoided due to its closed-nature, and expected lack of future support.

While the OSG scene-graph system is somewhat based on the efforts of the Performer library, there are two major differences between the OSG implementation and Performer: 1) OSG does not double-buffer the scene-graph, requiring the update traversal to avoid making changes to the scene-graph while a cull traversal is in progress, and 2) because many people contribute new node types to the open-source OSG, there is no strict enforcement of the rule preventing scene-graph modifications taking place outside the update traversal. Neither of these issues is typically a concern for desktop applications running on a single CPU system, but for multi-screen immersive systems, they are problematic. To address these implementation issues, we must specifically avoid the modification of the scene-graph when the multiplerenderings are taking place. FreeVR provides a semaphore-based locking/barrier system that we used to exclude writes to the scenegraph data during culling. Furthermore, when we discovered that some of the node-types (e.g., the particle system node) used the culling traversal to make additional modifications to the scenegraph we had to specifically insert extra locking code into those modules. The end result is a system that works satisfactorily, but the addition of each barrier results in lower frame rendering rates.

## Initial Abstractions

To achieve high performance the amount display and simulation computation that should be run in parallel should be maximized. Abstraction of dynamic graphical elements into several steps is necessary to achieve this goal. The processing of each element is broken into three stages: 1) rendering, the code which displays the element 2) updating the simulation code 3) synchronizing and maintaining congruency between the first steps. The first two steps run in parallel working on their own copy of the data and the last step synchronizes the two copies. This uses the assumptions that the simulation code in step 2 is non-trivial and would require more time than syncing the data in step 3. OpenSG provides a similar more generic mechanism; however, we believe that a generic solution is not always possible and a higher framerate is better with some specialization.

## Scalable Rendering

Each element of the scene strives to minimize its impact on the performance of the visualization. The goal was to section off as much work to the graphical processing unit as possible and leave the CPU time available for visualization to other elements and simulation code. The following systems manage and minimize their uses of system resource using level-of-detail algorithms specific to their data and visualization domain. The management of resources also allows the system to view landscapes that are larger than the available amount of system resources.

### Terrain

In the first implementation of the terrain we chose to implement the Geometrical Mipmapping (de Boer, 2000) algorithm. This enabled the system to visualize terrain datasets much larger than brute-force methods; however, it required that the entire dataset be loaded into memory. This algorithm would result in a complex memory management system. This method although reducing CPU usage over previous methods still could be improved. Ulrich's Chunked Lod algorithm (Ulrich, 2002) offered several advantages including higher triangle throughput, low CPU usage, and implicit memory management. This method, unlike Geometrical Mipmapping, requires offline tessellation of the elevation data. Figure 2 illustrates the tessellation of this approach.

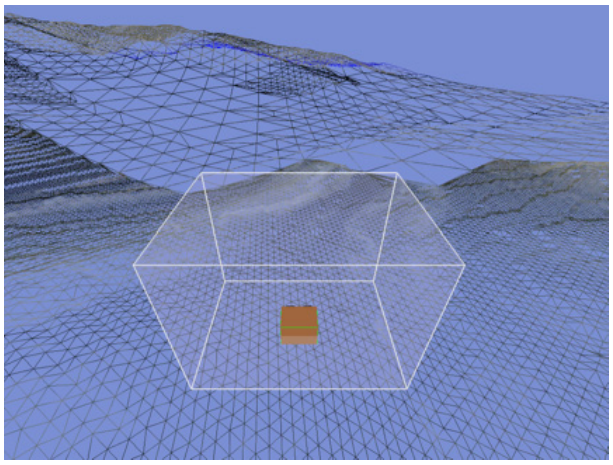

Figure 2. Surface Tessellation

### Vegetation

Similarly, vegetation is placed using an offline utility. This saves the cost of having to place trees at runtime and this saving allow the trees to be paged in from a file. The vegetation utility uses the fuel load data input from FARSITE and an expert's knowledge of the location to determine what types of vegetation are native to a location and their positions. Pixel error is used to determine when trees should not be displayed anymore. In general this is when the size of the rendered vegetation is smaller than a few pixels of screen space. Currently, trees are grouped into localized batches to be sent to the video card for rendering. This increases the amount of vegetation able to be rendered, but has the limit of only being able to process vegetation in groups. Processing vegetation individually allows the system to changed the appearance of that vegetation as it burned, but results in slow rendering speeds. A compromise is to use instancing (Corporation, 2004) to draw and process vegetation nearer to the viewer and process distance trees using groups.

### Wildfire

Visualization of the wildfire is driven directly from the FARSITE simulation data using particle based fire and smoke. Fire and other natural phenomenon have been successfully rendered using high numbers of particles. Applications using high numbers of particles are not able to render in real-time the use of sprites can significantly reduce the number of particles with good visual results(Reeves, 1983; Feldman et al., 2003). Sprite-based particle systems can accurately and realistically represent fire with very few particles. This is because localized behavior using an accurate offline simu-

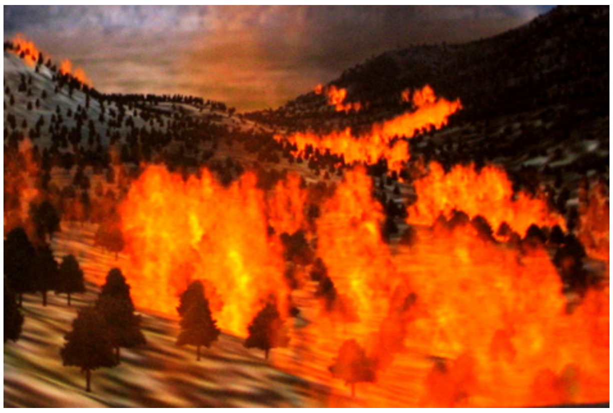
Figure 3. Fire and Smoke

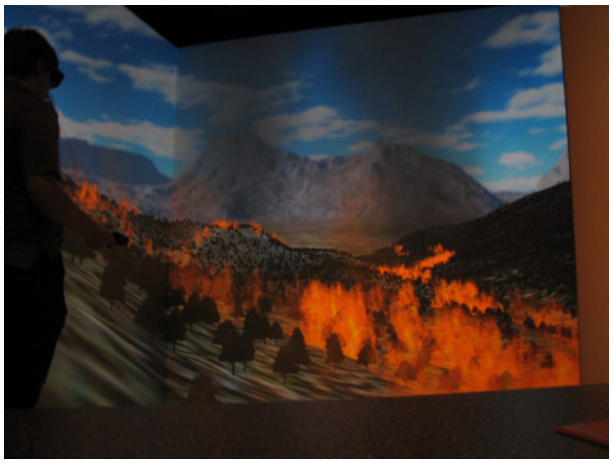
Figure 4. Fire Visualization in the Cave

lation and other factor such as color, position, emissiveness, and animation frame can be controlled using a real-time fire model (Wei et al., 2002; Tamas Umenhoffer, 2006; Nguyen, 2004). The visual complexity of particle systems can be reduced progressively as the distance increases from the user; however, the physical properties of all particles must be calculated. Reduction of rendered particles is often the bottleneck because the amount of sprites necessary to render a wildfire quickly reaches the maximum fillrate capacity of the GPU.

## RESULTS AND CONCLUSIONS

We have implemented the fire simulation system using the solution decisions described in the previous section. Initial responses from fire agencies have been positive. A screen capture of a sample run showing trees, fire, and smoke is shown in Figure 3. A larger picture of the simulation running in our four-sided Fakespace FLEX system is shown in Figure 4.

Visualization of computational wildfire models has many applications. Wildfire can spread over large amounts of area producing equally large datasets. We have introduced methods that can manage and visualize these datasets interactively. Abstraction of the rendering, simulation computation results in modular code that allows for the integration and optimization of different implementations for rendering elements of a scene. They also take advantage of different hardware configurations used to drive virtual reality systems.

## FUTURE WORK

Increasing the accuracy and realism of visualizing wildfire scenarios is a primary focus. This focus will specifically involve the use of additional outputs from FARSITE including flame length, rate of spread and crown fire data. The aim of this endeavor is to improve the accuracy of the visualization for the application of data analysis and model validation. The realistic visual appearance of the fire and smoke is a top priority for presenting wildfire scenarios.

Very little work has been spent on scaleable rendering of realistic real-time fire and smoke with application to the large scale required for wildfires. Current work only considers single or small scale fires over small objects. The complexity of realistically visualizing a single fire can quickly use the entirety of a computer system's resources. We will look to create a specialized LOD system that will further optimize our fire rendering system to support larger wildfires.

The software has been left extensible enough for use with other wildfire simulation models. with the long-term goal of using a real-time model. A real-time wildfire model has many applications including training and predictable prescribed burning. This will also necessitate the inclusion of an enhanced user interface and formal analysis of usability.

## ACKNOWLEDGEMENTS

This work was partially supported by the US Army PEO-STRI, NAVAIR Contract N61339-04-C-0072

## REFERENCES

Ahrens, J., McCormick, P., Bossert, J., Reisner, J. and Winterkamp, J. 1997. Wildfire visualization (case study). In *VIS '97: Proceedings of the 8th conference on Visualization '97* (pp. 451–ff.). Los Alamitos, CA, USA: IEEE Computer Society Press.

de Boer, W. H. 2000. Fast terrain rendering using geometrical mipmapping.

Bukowski, R. and Sequin, C. 1997. Interactive simulation of fire in virtual building environments. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (pp. 35–44). New York, NY, USA: ACM Press/Addison-Wesley Publishing Co.

Community, W. F. 2007. FARSITE.

Corporation, N. 2004. GLSL pseudo-instancing. Technical report, Nvidia.

Feldman, B. E., O'Brien, J. F. and Arikan, O. 2003. Animating suspended particle explosions. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers* (pp. 708–715). New York, NY, USA: ACM Press.

Finney, M. A. 1998. Farsite: Fire area simulator-model development and evaluation. In *Res. Pap. RMRS-RP-4*. U.S. Department of Agriculture, Forest Service.

Govindarajan, J., Ward, M. and Barnett, J. 1999. Visualizing simulated room fires (case study). In *VIS '99: Proceedings of the conference on Visualization '99* (pp. 475–

478). Los Alamitos, CA, USA: IEEE Computer Society Press.

Julien, T. U. S. and Shaw, C. D. 2003. Firefighter command training virtual environment. In *TAPIA '03: Proceedings of the 2003 conference on Diversity in computing* (pp. 30–33). New York, NY, USA: ACM Press.

McCormick, P. S. and Ahrens, J. P. 1998. Visualization of wildfire simulations. *IEEE Comput. Graph. Appl., 18(2)*, 17–19.

Nguyen, H. 2004. *GPU Gems* (1st Ed.). Addison-Wesley.

Reeves, W. T. 1983. Particle systems - a technique for modeling a class of fuzzy objects. *ACM Trans. Graph., 2(2)*, 91–108.

Reiners, D., Vo, G. and Behr, J. 2002. Opensg: Basic concepts.

Tamas Umenhoffer, L. S.-K. G. S. 2006. Spherical billboards and their application to rendering explosions. In *Proceedings of the 2006 Conference on Graphics Interface* (pp. 57–64). ACM Press.

Tate, D. L., Sibert, L. and King, T. 1997. Virtual reality: Using virtual environments to train firefighters. *IEEE Computer Graphics and Applications, 17(6)*, 23–29.

Ulrich, T. 2002. Chunked lod: Rendering massive terrains using chunked level of detail control. In *SIGGRAPH '02: Proceedings of the 24th annual conference on Computer graphics and interactive techniques Notes*. ACM Press.

Wei, X., Li, W., Mueller, K. and Kaufman, A. 2002. Simulating fire with texture splats. In *VIS '02: Proceedings of the conference on Visualization '02* (pp. 227–235). Washington, DC, USA: IEEE Computer Society.

Yu, Q., Chen, C., Pan, Z. and Li, J. 2004. A gis-based forest visual simulation system. In *ICIG '04: Proceedings of the Third International Conference on Image and Graphics (ICIG'04)* (pp. 410–413). Washington, DC, USA: IEEE Computer Society.

## AUTHOR BIOGRAPHIES

**MICHAEL A. PENICK** was born in Denver, Colorado, USA. He received his BS in Computer Science from the University of Nevada in 2005 and is currently an MS student in Computer Science and is expected to finish his program of study in May 2007. He has been employed in the Center for Advanced Visualization, Computation, and Modeling (CAVCAM) at DRI for the past two years. His research is in areas of graphics, virtual reality and wildfire visualization. His e-mail address is: `penick@cse.unr.edu`.

**ROGER V. HOANG** was born in Carson City, Nevada, USA. He received his BS in Computer Science from the University California at Los Angeles in 2006, and is currently an MS student in Computer Science. He has been employed in the Center for Advanced Visualization, Computation, and Modeling (CAVCAM) at DRI for the past year. His research is in areas of graphics, virtual reality and wildfire visualization. His e-mail address is: `hoangr@cse.unr.edu`.

**FREDERICK C. HARRIS, JR.** was born in San Jose, California, USA. He received his BS in Mathematics and MS in Educational Administration from Bob Jones University in 1986 and 1988, and his MS and PhD from Clemson University in 1991 and 1994. He has been at the University of Nevada, Reno since 1994, and is currently a Professor in the Computer Science and Engineering Department. He also has an affiliated faculty position with Desert Research Institute in Reno, where he is in the Center for Advanced Visualization, Computation and Modeling. His research is in the areas of Parallel and Distributed Computing, as well Graphics and Virtual Reality. His e-mail address is: `fredh@cse.unr.edu` and his web page can be found at `http://www.cse.unr.edu/~fredh`

**SERGIU M. DASCALU** was born in Bucharest, Romania. He received his MS in Automatic Control & Computers from the Polytechnic of Bucharest, and his PhD in Computer Science from the University of Dalhousie, Halifax, Nova Scotia, Canada in 2001. He has been at the University of Nevada since 2002 and is currently an Assistant Professor in the Computer Science and Engineering Department. His research is in the areas of software engineering and human-computer interaction. His email address is: dascalus@cse.unr.edu

**TIMOTHY J. BROWN** was born in Springfield, Illinois, USA. He received his BA in Astronomy-Physics from the University of Illinois in 1982, and his MS and PhD in Geography from the University of Colorado in 1988 and 1995 respectively. He has been at the Desert Research Institute since 1995 and is currently an Associate Research Professor. He is currently the Director of the Climate, Ecosystems, and Fire Applications (CEFA). His research areas include climatology and meteorology with an emphasis in wild land fire-climate and fire-weather relationships. His email address is: `Tim.Brown@dri.edu`

**PHILIP A. McDONALD** was born in Lebanon, Indiana. He received his BS in Forestry from the University of California, Berkeley in 1971 and his MS in Environmental Design and Meteorology from the University of Oklahoma in 1984. He has been a visualization software engineer serving the atmospheric sciences community for more than twenty years. He has been a faculty member in the Center for Advanced Visualization, Computation and Modeling at the Desert Research Institute since 2005 where he is an Assistant Research Visualization Scientist. His research areas include the application of advanced visualization methods to the earth sciences. His email address is: `phil.mcdonald@dri.edu`

**WILLIAM R. SHERMAN** was born in Madison, Wisconsin, USA. He received his BS in Computer Engineering and MS in Computer Science University of Illinois in 1986 and 1988 respectively. He was a Visualization Scientist for the National Center for Supercomputing Applications (NCSA) for 15 years. He came to the Desert Research Institute in 2004 as the Technical and Acting Director of the Center for Advanced Visualization, Computation,and Modeling (CAVCaM). His research areas include virtual reality and scientific data visualization. His email address is: `bill.sherman@dri.edu`