VFire: Virtual Fire in Realistic Environments

Roger V. Hoang* Joseph D. Mahsman† David T. Brown‡ Michael A. Penick§ Frederick C. Harris Jr.¶
Timothy J. Brown‖

University of Nevada, Reno
Desert Research Institute

ABSTRACT

The destructive capacity of wildfires and the dangers and limitations associated with observing actual wildfires has led researchers to develop mathematical models in order to better understand their behavior; unfortunately, current two-dimensional visualization techniques make discerning driving forces of a fire difficult and restrict comprehension to the trained eye. VFire is an immersive wildfire visualization application that aims to ameliorate these problems. We discuss several recent enhancements made to VFire in order to not only increase the amount of meaningfully visualized data and visual fidelity but also improve user interaction.

Keywords: Applied Virtual Reality, Wildfire Visualization

Index Terms: I.3.8 [Computing Methodologies]: Computer Graphics—Applications I.3.7 [Computing Methodologies]: Computer Graphics—Three-Dimensional Graphics and Realism I.6.6 [Computing Methodologies]: Simulation and Modeling—Simulation Output Analysis

1 INTRODUCTION

Wildfires pose a considerable challenge to researchers. Understanding their destructive behavior may be key to mitigating the damage they cause; however, the costs and risks associated with purposefully setting an area ablaze make doing so impractical; additionally, as controlling the spread of an unexpected wildfire is a primary concern, experimenting with new suppression techniques at such events is a risky proposition. In response to these problems, researchers have developed mathematic models using data collected from unplanned fires [4].

Visualizations of these fire models have been bound to the realm of two-dimensional images. Unfortunately, it is difficult to identify critical factors that dictate wildfire behavior in two dimensions. Slope, for example, is difficult to determine from a two-dimensional image. VFire is a virtual reality wildfire visualization tool that addresses this problem. Visualizations are produced using remote sensing data in conjunction with the inputs and outputs of a simulation model. By attempting to reconstruct actual fires using a simulation with similar preconditions and comparing them to what was seen by observers on the ground, researchers will be able to use VFire to view a simulation from multiple perspectives and verify and refine fire models. Additionally, VFire will be used to provide training scenarios and inform land managers of the benefits of preventative measures.

*e-mail: hoangr@unr.nevada.edu
†e-mail:mahsmanj@unr.nevada.edu
‡e-mail:dtbrown@unr.nevada.edu
§e-mail:penick@cse.unr.edu
¶e-mail:fredh@cse.unr.edu
‖e-mail:tim.brown@dri.edu

With these purposes in mind, much of our efforts have been focused on increasing the amount of visualized data, enhancing the visual presentation, and improving user interactivity. To do so, we exploit several features of modern graphics hardware, in particular the programmable shader, in order to speed up not only the rendering but also the simulation updating. The rest of this paper is structured as follows: Section 2 describes FARSITE, the fire behavior simulator that VFire visualizes; Section 3 details the main components of the visualization; Section 4 discusses the user interactivity currently available; finally, Sections 5 and 6 provide our conclusions and future work, respectively.

2 FARSITE

Our visualization is currently driven by data from FARSITE [1], a wildfire simulator developed by the USDA Forest Service and used by a number of national and state fire and land management agencies. FARSITE takes into several factors including elevation, slope, and fuel type to simulate fire behavior based on an elliptical spread model; the data outputted from the program include times of arrival, fireline intensity, and spread direction. Data is spatially organized into a evenly-spaced rectangular grid, with each variable placed in a different grid.

3 VISUALIZATION

VFire is composed of three primary visible components: terrain, vegetation, and fire. In addition to determining which pieces of data can effectively be visualized for each element, rendering each of these parts in a multi-display environment such as our four screen CAVE-like Fakespace FLEX display presents its own set of problems. These problems and our solutions to them are outlined in the following sections.

3.1 Terrain

The topology of an area can have a significant influence on the behavior of a wildfire. Slopes can cause fires to climb faster or descend slower; additionally, the spatial characteristics of the land can mitigate or intensify the effects of wind; as a result, accurately depicting terrain is paramount. Due to the size of terrain used for wildfire analysis, VFire uses a level-of-detail rendering system based on [5]. Terrain is preprocessed into chunks which are loaded as needed in a separate thread at runtime. To further speed up rendering, the terrain geometry is stored in vertex buffer objects directly in video memory; vertex morphing is done in a vertex shader.

Originally, we used a 1m resolution satellite image to texture the terrain. While the results were acceptable at long distances, the magnification of the texture when the user approached the terrain yielded a very blurry, unrealistic image. Further compounding the problem was the existence of objects in the image; a tree or building present in the image would be rendered onto a relatively flat surface. As a result, the image was abandoned in favor of a multi-texturing solution that covers the terrain in a repeating mixture of grass, slash, and rock textures. We store the vegetation data grid inputted into FARSITE as a texture. Within the vertex shader, we determine the type of vegetation that exists around each vertex and assign texture weights accordingly. The interpolation during the

rasterization phase allows for smooth transitioning from one texture type to another.

We implement terrain scorching effects in similar fashion. The progress of a fire is outputted from FARSITE as grid which we store as a texture; the value in each cell represents the time at which the cell begins to burn. Again, within the vertex shader, we compare the current simulation time to the time of arrival from the appropriate texel and modulate the properties of each vertex accordingly, adding a reddish glow around the time of arrival and slowly increasing the weight of a burn texture.

3.2 Vegetation

Just as accurate depiction of terrain is critical for analysis, so too is accurate representation of vegetation, as the type and density of vegetation heavily influences the characteristics of a wildfire. Two interchangeable modules may be used for rendering vegetation. The first module, developed in-house, consists of procedurally-generated trees. For every tree generated, 116 unique vertices are used. In heavily-wooded areas, implementing burn effects using vertex texture fetches causes a significant drop in framerate due to the introduction of a large latency in the shader [2]; instead, we compute the times of arrival during the generation phase and store them as vertex attributes, which are much faster to retrieve. The effects of crowning are visualized by only burning portions of a plant that are above or below a particular threshold height.

The second module utilizes SpeedTree (www.speedtree.com). The SpeedTreeRT library could not be used for an application in a CAVE out-of-the-box. CAVE systems utilize multiple video outputs. As a result, a rendering context must be managed for each output in a 3D application. In such applications, identifiers specific to a rendering context, such as texture objects and vertex buffer objects, must be allocated and tracked for each rendering context. The entire SpeedTreeRT library was modified in order to achieve this goal. In addition, the shaders that accompany the SpeedTreeRT reference application were modified to add burn effects using a burn progress value restricted to the range [0..1], where 0 is unscathed and 1 is charred. Burn effects include dissolving leaves and progressively modulating fragment colors.

Future work includes making similar modifications to SpeedGrass to allow its use in a CAVE. Because grass and shrubbery are expensive to render using multiple instances of SpeedTree models, the use of SpeedGrass will be a compromise between detail and efficiency in rendering these vegetation types.

3.3 Fire and Smoke

Fire and smoke are implemented using particle systems. To speed up rendering and updating of particles, all particle properties are maintained on the GPU [3]; however, because these properties must be consistent between screens, emissions are done on the CPU. Additionally, all particle data for all screens is updated essentially in lock step using the same time deltas. All particles are rendered as cylindrical billboards; in the case of fire particles, the tops of these billboards are skewed according to spread direction and rate data outputted from FARSITE and retrieved in the vertex shader.

To address the brightness of the fire effects during various times of the day, high dynamic range rendering was employed; as a result, the effects are muted under well-lit conditions while substantially brighter under darker conditions. To prevent screens from having dissimilar coloration, the same average luminance is used during the tone-mapping process. Presently, this value is computed by taking a simple average of each screen's luminance; eventually, this will be done with a user-oriented weighting scheme.

4 Interaction

VFire currently allows the user to interact with the visualization in a number of ways through the use of a tracked wand. The user can fly about the virtual world or constrain movement to follow the terrain; additionally, the user may drop markers that can be teleported to at any time. The lighting of the world can be modified with the press of a button, allowing the user to view a simulation under various lighting conditions.

Currently, the states of the majority of VFire's graphical elements can be determined purely by the current simulation time; as such, time can be almost freely controlled, allowing the user to speed up, slow down, stop, and reverse the simulation. The exception to this property are the particle effects, which are dynamically allocated and destroyed. To allow for time reversal, particles are created with approximated states at their times of death as time flows backwards and are destroyed if their age becomes negative.

5 Conclusions

VFire provides an immersive visualization of an established fire behavior model. In addition to being a model verification tool, VFire can reveal crucial aspects of fire behavior to analysts and visualizes data in such a way that it can be more easily understood by a broader audience.

By exploiting features of modern graphics hardware, we were able to increase the visual fidelity of our visualization without reducing interactivity; on the contrary, by creating a system where every vertex knows when and how to destroy itself, the user has been granted more control over the visualization.

6 Future Work

Improving the accuracy of the visualization will continue to be a main objective. We are developing ways to incorporate more data into the visualization in order to represent variables such as fireline intensity and wind. Additionally, incorporating a physically-based smoke model will be integral for atmospheric analysis. In order to more closely replicate the real world, we are also working on the ability to extract vegetation and objects from satellite images in conjunction with other pieces of data using image processing techniques.

In the realm of interactivity, we plan to integrate the FARSITE software directly into VFire in order to allow users to experiment with various parameters and increase VFire's utility as a training tool. With respect to this, given the GPU-centric nature of our current version of VFire, pushing the simulation software directly onto the graphics hardware may be a logical step; current graphics hardware now has features that bring such a move closer to reality.

Acknowledgements

This work was partially supported by the US Army PEOSTRI, NAVAIR Contract N6133907C0072.

References

[1] M. A. Finney. Farsite: Fire area simulator - model development and evaluation. Research Paper RMRS-RP-4 Revised, USDA Forest Service Rocky Mountain Research Station, 1998.

[2] P. Gerasimov, R. Fernando, and S. Green. *Shader Model 3.0: Using Vertex Textures*. NVIDIA Corporation, 2004.

[3] L. Latta. Building a million particle system. In *Proceedings of the Game Developers Conference 2004*, 2004.

[4] W. R. Sherman, M. A. Penick, S. Su, T. J. Brown, and F. C. Harris. Vrfire: an immersive visualization experience for wildfire spread analysis. In *IEEE Virtual Reality Conference, 2007. VR '07*, pages 243–246, Mar. 2007.

[5] T. Ulrich. Rendering massive terrains using chunked level of detail control. SIGGRAPH Course Notes, 2002.