ICIC Express Letters
Part B: Applications
Volume **7**, Number **8**, August **2016**

ICIC International ©2016 ISSN 2185-2766
pp. **1783–1789**

A FIRE SIMULATION SYSTEM FOR VIRTUAL SAFETY TRAINING

Guijuan Zhang[1,2], Lei Lv[2], Daying Lu[2], Yongjian Wang[3]
Dianjie Lu[2] and Xiangxu Meng[1]

[1]School of Computer Science and Technology
Shandong University
No. 27, Shanda Nanlu, Jinan 250100, P. R. China
guijuanzhang@gmail.com

[2]School of Information Science and Engineering
Shandong Normal University
No. 88, Wenhua East Road, Lixia District, Jinan 250014, P. R. China

[3]Institute of Computing Technology
Chinese Academy of Sciences
No. 6, Kexueyuan South Road, Zhongguancun, Haidian District, Beijing 100190, P. R. China

Received January 2016; accepted April 2016

Abstract. *We present a fire simulation system for virtual safety training in this paper. Our method allows users to have realistic experience when learning emergency skills in fires without exposing them to the hazard. To do this, we use Fire Dynamic Simulator (FDS) in fire research area to compute physically and mathematically accurate fire simulation data which makes the training more effective. In addition, we use photo-realistic rendering method in computer graphics area to produce visual pleasing fire simulation results. The method enhances the visualization of FDS data significantly. We implement the rendering algorithm on Graphics Processing Unit (GPU) so that our system can run in real-time. Results show that our system can improve the experience of virtual training and the effectiveness of escape-skills learning significantly via providing accurate and realistic fire simulation results.*

Keywords: Fire simulation, Virtual safety training, Fire dynamic simulator, Photo-realistic rendering

1. **Introduction.** Fire emergency often happens in our daily life and many people are injured, and even worse, lost their lives in the fires every year. As one of the most common emergencies, fire is very harmful if not dealt properly. Therefore, training people with the escape and self-protection skills is essential to protect personal and public property safety in fires.

However, the fires-escape training in a real-world scenario is impractical because of the high cost and unsafe factors. Virtual safety training which benefits from virtual reality technology can address the above problems. It provides realistic experience that allows trainee to learn the skills in dangerous situation without exposing them to the hazard.

Provided an available way for skills training in fires, virtual fire training requires simulating fire phenomenon with high degree of realism. In fire simulation world, FDS [1] is one of the most famous softwares developed by the National Institute of Standards and Technology (NIST). To visualize the numerical data, VTT Technical Research Center of Finland releases a software, a fire dynamics simulator with evocation (FDS+Evac) to simulate human egress using the evacuation module that is fully embedded in FDS [2]. Many commercial systems (e.g., Myriad, Legion, EXODUS, EGRESS, Vegas, Simulex and EVACNET) also appear in building fire simulation [3]. However, the simple visualization method in fire simulation area cannot achieve realistic results for fire training. More professional fire visualization methods are proposed in computer graphics area [4, 5, 6, 7, 8].

Unfortunately, most of the fire data in these methods are synthesized rather than physically accurate. Thus, it is impractical to be applied to virtual safety training.

In this paper, we present a simulation system for fire safety training. The system provides realistic experience for trainee when learning emergency skills. We adopt FDS to compute the fire physical model and render the fire numerical model with photo-realistic rendering method. The rest of the paper is organized as follows. Section 2 describes the method of this paper. Section 3 discusses the results and we conclude this paper in Section 4.

2. **Method.**

2.1. **Pre-processing.**

In FDS, environment is represented as a set of blocks. Given 3D environment model, we get the voxels of the model by a voxelization method [9, 10] and combine them along three axes X, Y and Z so as to obtain blocks for computation input.

2.1.1. *Voxelization.* We use z-buffer based voxelization method [9, 10] to get voxels of 3D models. The method stores the depth information from six views (namely, $+x$, $-x$, $+y$, $-y$, $+z$ and $-z$) of the 3D model, and checks whether the depth of each voxel is between the value limits stored in z-buffer. To do this, three pairs of z-buffer $([x_1, x_2], [y_1, y_2], [z_1, z_2])$ along X, Y and Z are created. Each pair of them stores depth information of the model in one axis. Take X for example, x_1 stores the smallest depth value and x_2 stores the largest value. It is intuitive to get depth value for the three pairs of z-buffer by using orthographic projection operations. Given a voxel $v(i, j, k)$, we convert it to the z-buffer range as

$$(x, y, z) = (i, j, k) \cdot \frac{depth_{\max} - depth_{\min}}{N - 1},$$

where $[0, N)$ denotes the size of the model, $[depth_{\min}, depth_{\max}]$ is the range of z-buffer. Thus, voxel $v(i, j, k)$ locates inside the model if and only if $x_1(j, k) \leq x \leq x_2(j, k)$ and $y_1(k, i) \leq y \leq y_2(k, i)$ and $z_1(j, i) \leq z \leq z_2(j, i)$. According to the above operations, we obtain the voxelization results of the chair as shown in Figure 1.

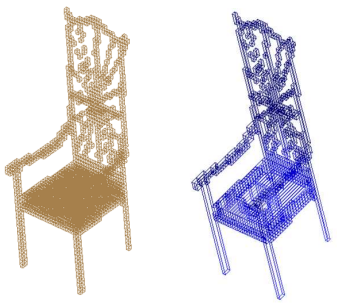

FIGURE 1. Pre-processing method. Left: voxels after voxelization. Right: blocks after combining connected adjacent voxels.

2.1.2. *Combination.* To decrease the number of blocks for fire simulation, we combine the voxels to get larger blocks after voxelization. We start from a voxel and combine it with its connected adjacent voxel along X. Thus, we may get several blocks in each line of X axis. Next, we start from a block and combine it with its connected adjacent block along Y and Z axis sequentially to make larger blocks. Figure 1 shows the combination results. Observe that the voxels on the leg of the chair are combined to a single block finally.

2.2. **Fire model.** We use FDS [1] for computing the dynamics of fire. Simulating fire dynamics is complex because many models (e.g., hydrodynamic model, combustion model as well as thermal radiation model) that govern the fire simulation should be considered.

The most important governing equations of fire simulation are a set of conservation equations for mass, momentum and energy, etc. Another important model for fire simulation is combustion. We use large eddy simulations (LES) and thus the mixture-fraction based combustion model is adopted. More details about the fire model can be referred to [1].

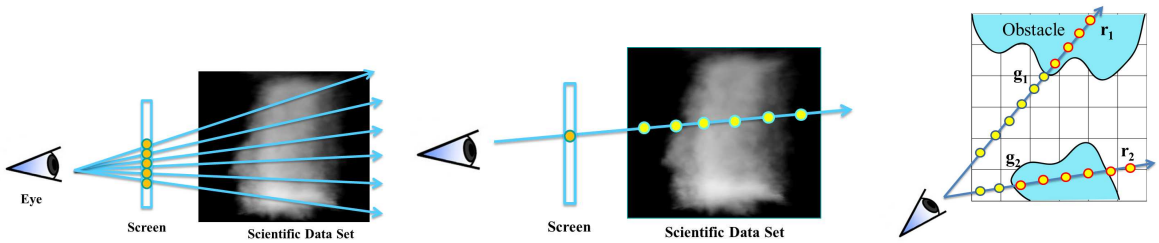

FIGURE 2. Ray-casting method. Left: light rays are cast from each pixel on the screen to the data volume. Middle: the final color of the pixel is obtained by accumulating optical properties of the sample points along the light rays. Right: merge with the geometric model. The rays r_1 and r_2 will be terminated at g_1, g_2 on geometric models.

2.3. **Rendering.** The ray-casting approach is a typical image-order approach. The method computes the color of each pixel in the final image by casting rays into the data volume. Specifically, given camera parameters and a pixel position, a light ray is set up at first. Next, optical properties are accumulated along the light ray. It is executed until the volume is traversed. Figure 2 shows the process. The start position of a light ray is frontface coordinate P_s of volume bounding box, and the end positions P_e are the backface coordinates. The ray direction is $\boldsymbol{d} = P_e - P_s$. Thus, each sample point on the light ray is calculated by $P = P_s + delta \times \boldsymbol{d}$ where $delta$ is the step length. The scalar value at this position is looked up from the volume data and is converted to color C_{src} and opacity α_{src}. Finally, accumulated color C_{dst} and opacity α_{dst} are updated according to

$$C_{dst} = C_{dst} + (1 - \alpha_{dst})C_{src} \qquad \alpha_{dst} = \alpha_{dst} + (1 - \alpha_{dst})\alpha_{src}.$$

2.3.1. *Virtual endoscopy.* Fire simulation requires rendering the simulation results from the agent's eyes because most agents are located in the volume. When moving the viewpoint into the volume data, the near clipping plane intersects with the volume boundary and the start positions on the frontface of volume boundary are lost which leads to holes in the final animation results. Figure 3 gives an example.

To address the above problem we use virtual endoscopy technology to fill up the hole. We set the starting position of the light rays as points on the near clipping plane as shown in the following algorithm [12].

Step 1: Set the near clipping plane as the starting positions of the light rays.
Step 2: Calculate position of first backface in the ray direction.
Step 3: Calculate position of first frontface in the ray direction.
Step 4: Start the ray from the first frontface if it is nearer than the first backface.

2.3.2. *Merge with the environment.* Fire often happens inside a building or a factory which contains a lot of 3D models in the environment. To make the rendering results more realistic, we are required to merge the volume data with the 3D models in the scenario.

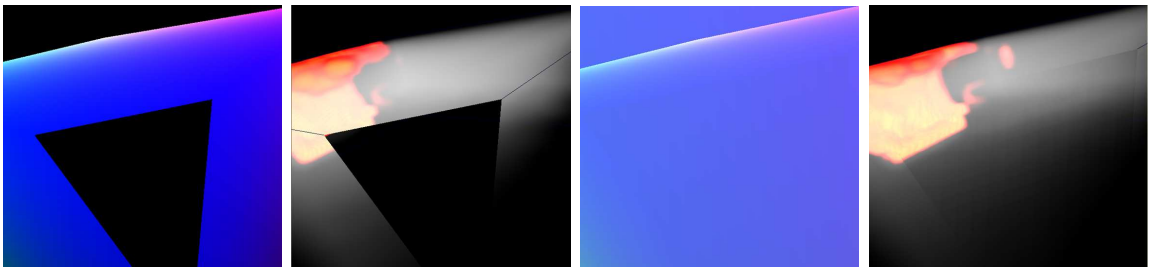

FIGURE 3. Results with holes and after filling holes. Left: starting image with a hole. The rendered frontface which provides the start positions of the rays is intersected with the data volume. Thus, it becomes invisible in the interaction location and results in a hole. Middle left: render results. The lost starting positions lead to a hole in the final image. Middle right: correct starting image after filling holes. The starting positions are replaced with corresponding positions of the near clipping plane. Right: correct rendering results when the camera is located in the data volume.

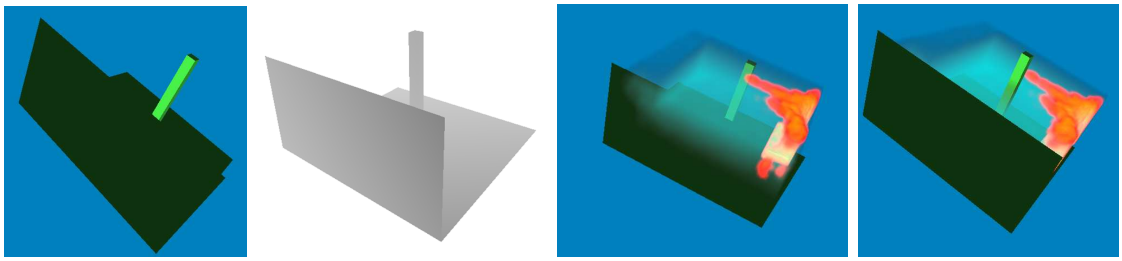

FIGURE 4. Rendering fire in a 3D environment. Left: 3D environment model. Middle left: z-buffer of the 3D model. Middle right: results before considering the rays' end positions. Right: results after considering the rays' end positions.

Similar to the procedure of filling holes, merging with the environment is straightforward. The only difference is that the end positions of the rays should stop in front of the obstacle.

We use depth comparison method to complete this task efficiently. The depth value of the 3D environment and backface of the bounding geometry can be obtained by rendering them to the z-buffer respectively. Next, we compare theses depth values. If the depth value of the 3D environment is smaller than that of the backface, the end position should be set as the respective points on 3D geometry model. See Figure 2, r_1 and r_2 will be terminated at g_1 and g_2. Figure 4 shows the rendering results in a 3D environment. Observe that errors appeared if we do not consider the end position of the light rays.

2.3.3. *GPU-based implementation.* Implementing ray-casting algorithm on GPU is straightforward because of the inherent parallelism of the method. In ray-casting algorithm, a single ray is cast into the volume for each pixel on screen or the final image. Then the volume data is re-sampled at discrete positions along the ray. By means of the transfer function the scalar data values are mapped to optical properties. As for GPU-based implementation, we define kernel functions to compute the pixel's color. Each kernel function calculates the color of a pixel on the screen. Since each thread has unique ID, it can be used to locate the specific pixel on the screen. Therefore, by executing the kernel function in parallel, the implementation on GPU can speed up ray-casting algorithm significantly.

3. **Results.** The fire simulation in this paper is executed on a PC with Intel Quad-cores/4 threads CPU. We separate our experiments into two parts. In the first part we

test the performance of the fire simulation system and in the second part we exhibit the fire simulation results.

3.1. **Performance.** Numerical computation in FDS consumes large portion of resources. We set the fire source at the center of a room and we create a grid whose resolution is $246 \times 246 \times 63$. In Table 1, we show the performance of the numerical computation. A 10 minutes fire propagation result consumes about 15890 seconds (approximately 4.41 hours) for computation. We also run the parallel version of FDS for this example. It takes 8463 seconds for computing fire motion when using 2 cores and 5045 seconds when using 4 cores. The speedup gained from the above example is also shown in Table 1. The value is close to the ideal value because the fire simulation in this paper is a computation-intensive task and the communication cost is rather small compared to the expensive numerical computation.

TABLE 1. Performance of the numerical computation in fire simulation

Number of cores	Simulation time	Computation time	Speedup
1	600	15890	–
2	600	8463	1.88
4	600	5045	3.15

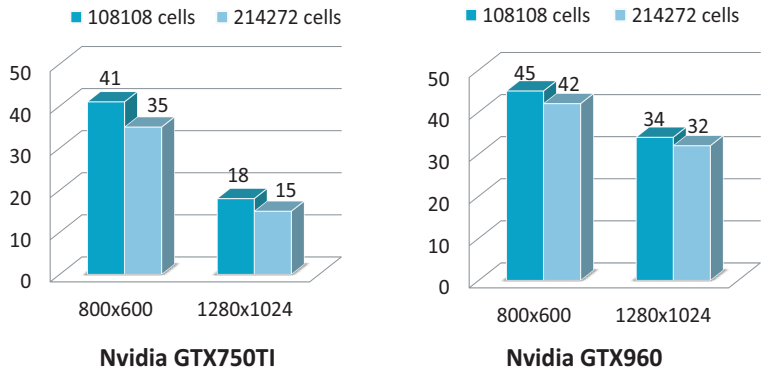

FIGURE 5. Performance of fire rendering algorithm. Left: frame rates on Nvidia GTX750TI. Right: frame rates on Nvidia GTX 960.

We also test the efficiency of our photo-realistic fire rendering algorithm. We use GPU-based ray-casting algorithm to improve the efficiency. Most of our examples can achieve interactive frame rate (e.g., more than 15 frames/second). Figure 5 shows the performance of our fire rendering algorithm with Nvidia GTX750TI and GTX960. Note that X axis denotes different output image resolution and Y axis denotes the frame rate. Two examples are tested and the grid cells numbers of each example are 108108 and 214272 respectively. Results show that the performance of fire rendering is influenced by the resolution of the final images significantly. When the images resolution are set 1024×768, the frame rate achieves 41 frames/second on GTX750TI. When the image resolution increases to 1440×900, the frame rate reduces to 18 frames/second. Furthermore, the performance of different graphics card also affects the rendering efficiency. Results show that GTX960 obtains higher frame rate than GTX750TI. Since all frame rates exceed 15 frames per second, we can see the fire prorogation in real-time after using GPU-based speeding up method.

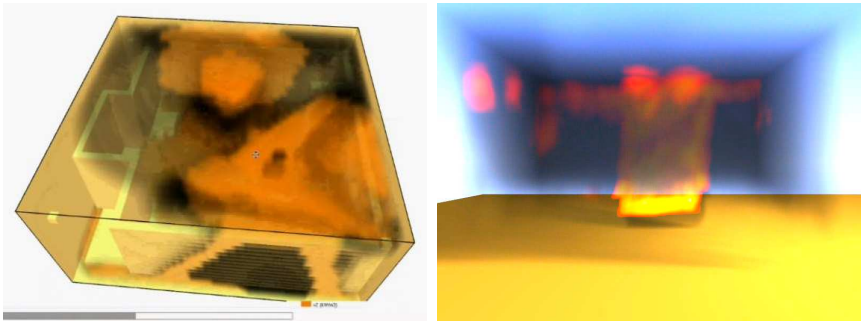

FIGURE 6. Simulation results. Left: fire rendering in Smokeview. Right: fire rendering with ray-casting algorithm.

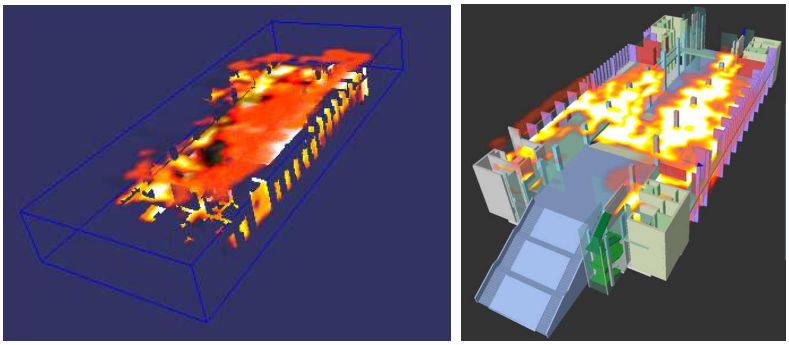

FIGURE 7. Rendering results. Left: rendering results before merging with the environment. Right: rendering results after merging with the environment.

3.2. **Simulation results.** FDS provides basic rendering mechanism based on a software named Smokeview. The results demonstrate the motion of the fire and smoke visually. It allows non-professional person to understand the propagation of fire and smoke intuitively. However, visual effects are not pleasing since these two software only focus on the simulation accuracy. Figure 6 shows the rendering results of Smokeview and our method respectively. We achieve better visualization results using ray-casting algorithm. Figure 7 shows the photo-realistic rendering results before and after merging with the environment in our system. The visual pleasing results produced in our method can provide realistic experience for skill learning in dangerous situation.

4. **Conclusions.** We have presented a fire simulation system for virtual safety training in fires. To provide realistic experience for trainee in emergency situations, we use the famous FDS software in fire research to compute fire data and ray-casting algorithm in computer graphics area to render the simulation data. The fire results with high degree of visual realism improve the effectiveness of skills learning in fires significantly. As for future work, we would like to extend some numerical computation on GPU to further improve the simulation efficiency.

Acknowledgment. This work is partially supported by Project supported by the National Natural Science Foundation of China (Nos. 61202225, 61303157, 61303007, 61402270, 61402269, 61572299), Shandong Provincial Natural Science Foundation (ZR2014FQ009, ZR2015FQ009), Research Fund for Excellent Young and Middle-aged Scientists of Shandong Province (BS2013DX044), Shandong Province Higher Educational Science and Technology Program (J13LN13).

REFERENCES

[1] K. McGrattan, R. McDermott, C. Weinschenk et al., *Fire Dynamics Simulator Technical Reference Guide Volume 1: Mathematical Model*, NIST Special Publication 1018 Sixth Edition, Gaithersburg, Maryland, 2014.

[2] T. Korhonen, Fire dynamics simulator with evacuation: FDS+Evac technical reference and user's guide, *Technique Report*, VTT Technical Research Centre of Finland, 2014.

[3] E. Kuligowski, R. Peacock and B. Hoskins, *A Review of Building Evacuation Models*, Gaithersburg, Maryland, 2005.

[4] S. Sato, T. Morita, Y. Dobashi et al., A data-driven approach for synthesizing high-resolution animation of fire, *Proc. of the Digital Production Symposium*, pp.37-42, 2012.

[5] D. Nguyen, R. Fedkiw and H. Jensen, Physically based modeling and animation of fire, *ACM Trans. Graphics*, vol.21, no.3, pp.721-728, 2002.

[6] J. Hong, T. Shinar and R. Fedkiw, Wrinkled flames and cellular patterns, *ACM Trans. Graphics*, vol.26, no.3, 2007.

[7] C. Horvath and W. Geiger, Directable, high-resolution simulation of fire on the GPU, *ACM Trans. Graphics*, vol.28, no.3, 2009.

[8] J. Chadwick and D. James, Animating fire with sound, *ACM Trans. Graphics*, vol.30, no.4, 2011.

[9] E. Karabassi, G. Papaioannou and T. Theoharis, A fast depth-buffer-based voxelization algorithm, *Journal of Graphics Tools*, vol.4, no.4, pp.5-10, 1999.

[10] G. Zhang, D. Zhu, X. Qiu et al., A scene processing method for fluid simulation, *Journal of Computer-Aided Design & Computer Graphics*, vol.22, no.8, pp.1360-1365, 2010.

[11] D. Anderson, J. Tannehill and R. Pletcher, *Computational Fluid Mechanics and Heat Transfer*, Hemisphere Publishing Corporation, Philadelphia, Pennsylvania, 1984.

[12] H. Scharsach, *Advanced Raycasting for Virtual Endoscopy on Consumer Graphics Hardware*, Master Thesis, Universität of Wien, 2005.