SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

August 2013, Vol. 56 082116:1–082116:9
doi: 10.1007/s11432-013-4893-7

Real-time rendering of flames on arbitrary deformable objects

WANG Lei[1,2], YE WanFang[1,2], DUAN Ming[1,2] & ZHANG YanCi[1,2*]

[1]*College of Computer Science, Sichuan University, Chengdu* 610065*, China;*
[2]*Key Laboratory of Fundamental Science on Synthetic Vision, Chengdu* 610065*, China*

Received September 25, 2012; accepted January 19, 2013; published online June 9, 2013

Abstract This paper proposes a new real-time algorithm to generate visually plausible flames on arbitrary deformable objects. In order to avoid the time-consuming computation of physical fields, the main idea of our algorithm is to build an approximate distance field based on the object's surface. And then the distance field is sampled and the distance samples are employed to fetch values from a color map which is precomputed according to physical methods. In order to simulate the dynamic flames by static distance field, simplex noise is used to disturb the sampling process. Our algorithm is also capable of handling the interaction between the flames and external factors such as wind. In order to achieve such a goal, two approximate distance fields are built to represent the inner flames and the outer flames respectively, which are combined together to accomplish the interaction. The experimental results show that our algorithm can produce visually plausible and user controllable flames on arbitrary deformable objects in real-time.

Keywords fire simulation and rendering, distance field, simplex noise, ray marching, color distribution map

Citation Wang L, Ye W F, Duan M, et al. Real-time rendering of flames on arbitrary deformable objects. Sci China Inf Sci, 2013, 56: 082116(9), doi: 10.1007/s11432-013-4893-7

1 Introduction

Real-time modelling and rendering realistic flames on arbitrary deformable objects is a very challenging task in computer graphics. It has been widely employed in many applications, such as films, video games, simulation systems and so on. Different applications have different requirements for flames' modelling and rendering. For instance, film industry requires generating controllable and photo-realistic flames on arbitrary objects which are able to interact with external factors such as wind and other objects in the scene. The fire-engulfed character in the film "Selma Blair" presents a very expressive example of those applications [1]. For some other applications, e.g. video game or real-time simulation applications, they prefer billboard-based, particle-system-based or noise-based methods which give first priority to rendering efficiency. For example, the noise-based fire rendering algorithm presented in [2] is capable of rendering flames on plain surfaces in real time. But most of the real-time algorithms cannot produce realistic flames on deformable objects, which is a quite common requirement in many applications.

*Corresponding author (email: yczhang@scu.edu.cn)

In this paper, a new real-time flames rendering algorithm is proposed. It combines the advantages of the physically-based and noise-based methods with the new features of producing visually plausible flames on arbitrary deformable objects. Our method contains two important components. The first component is a precomputed 1D color map describing the color distribution of flames. This map is generated according to physical equations. The second component is the texture coordinates generated on-the-fly to fetch values from the color map. In order to avoid the time-consuming computation of various physical fields (e.g. temperature field, velocity field, and pressure field), distance field (DF) [3] which can be computed in real time is used to generate such texture coordinates. However, the static distance field cannot produce dynamic flames. In order to address this issue, simplex noise [4] is employed to disturb the sampling process of distance field to simulate the dynamic feature of flames.

The main contributions of our algorithm are:

1) Combining a precomputed color map with disturbed samples from distance field to avoid complicated physical computations.

2) Using simplex noise to disturb the sampling process to achieve random and dynamically shaped flames.

3) Building two distance fields to simulate the outer and inner flame respectively, where the inner flame is only subject to model's vertices, while the outer flame can be influenced by external factors such as wind.

2 Related work

The texture-based method has been widely used to render flames. It has the advantages of real-time performance and easy implementation. However, its drawbacks are also obvious, e.g. the flames are limited by the size and resolution of the textures, and they cannot interact with external factors either. Inakage [5] proposed a flames rendering algorithm by mapping 2D texture onto an implicit primitive. A similar method [6] was proposed to produce flames by mapping a 2D texture on the NURBS surface. Wei et al. [7] introduced the use of textured splats as the basic display primitives to simulate open surface fire. Rose [8] proposed a real-time method which produced flames from videos or high quality images rendered off-line.

The particle system based method is another widely used method for flames rendering with the advantages of low computation cost and stability. However, its computation cost is closely related to the number of particles, so it is often used together with other methods. Reeves et al. [9] proposed using particles to represent fuzzy objects, such as flames. Perry et al. [10] contributed a method which combined a particle system for flames with a model for flame spreading. A real-time fire simulator with particle system was implemented[1)]. Horvath et al. [1] introduced a coarse particle grid simulation performed on GPU, as well as a GPU-based volume rendering scheme.

Physically-based methods can produce photo-realistic effect. Most of them are based on solving Navier-Stokes (NS) Equation whose computation cost is quite big. Nguyen et al. [11] proposed to use incompressible NS Equation and Blackbody Radiation Equation to model and animate fire. Pegoraro et al. [12] improved this method to make highly realistic simulations of various types of flames. Minor [13] applied Eularian grid to solve NS Equation and simulate and animate fire. Muller et al. [14] presented an interactive method based on SPH to render fire. Ishikawa et al. [15] used 3D voxels to compute the velocity field of NS equation and advection to transport temperature field. Muller et al. [16] used stable method to solve NS equation and simulated fire as a flowing fluid successfully. Flavien et al. [6] solved NS Equation to simulate calm fire's turbulence. Hong et al. [17] proposed a unified framework to model and animate fire under general geometric constraints and evolving rules, so that it can satisfy artists' variety of requests for fire. In order to improve the interaction between fire and external object, Zhu et al. [18] presented a real-time model to simulate fire phenomena involving solid object decomposition process.

With respect to the noise-based methods, Perlin [19] applied perlin noise to achieve fire's high-details

1) Somasekaran S. Using particle systems to simulate real-time fire. 2005.

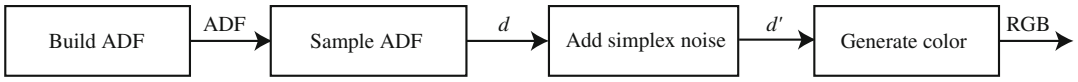


Figure 1 Algorithm overview: 1) build ADF; 2) sample ADF to produce distance sample d; 3) produce d' by using simplex noise to add a random factor to d; 4) generate flame colors from a precomputed color distribution map and d'.

and turbulence. And Tatarinov [2] constructed basic volumetric fire with a profile texture which is disturbed by noise.

Efficient representation of deformable object plays an important role in flame rendering. Signed distance field proposed in [20] was defined as the closest geodesic distance from a point to the surface of an object. Ref. [21] proposed to pre-compute the internal distance field of each undeformable model. But the computation was tedious and time-consuming. In order to overcome this shortcoming, Ref. [22] introduced a new method to compute signed distance on GPU and fixed the leaking artifacts mentioned in [3, 23].

3 Overview

According to [11], flames have the following physical features which are the basis of our algorithm: the color of flames is determined by temperature distribution, which can be approximately represented by a function $T(t)$, where T and t are temperature and time respectively. In order to produce flames' color, the physically-based methods based on solving the NS Equation are used to get the temperature field. Unfortunately, such computation is time-consuming thus limiting the use of these methods to real-time applications.

The basic idea of our algorithm is to combine a precomputed color distribution map with online generated samples from distance field to produce flames so that the time-consuming physical computations can be avoided. The drawback of the above strategy is that the distance field is too regular to represent the random and dynamically shaped flames. In order to solve this issue, simplex noise is employed to disturb the sampling process of distance field to add randomness to the distance samples.

However, building accurate distance field is also time-consuming. In order to improve the computation efficiency, Approximate Distance Field (ADF), which can be constructed much more efficiently, is applied to replace distance field in our method.

As illustrated in Figure 1, our algorithm can be divided into four stages: 1) Build ADF on GPU in each frame. Because the building algorithm is quite fast, ADF can be constructed in every frame to respond to the animated objects in the scene. 2) Sample ADF by a ray-marching algorithm to produce distance sample d. 3) Disturb the sample d by simplex noise to produce value d'. 4) Generate flames' colors by converting d' to texture coordinate to fetch values from a precomputed 1D color distribution map.

4 Building ADF

In this section, we firstly introduce the definition of ADF, and then a GPU-friendly algorithm is proposed to efficiently construct and update ADF for arbitrary deformable objects.

4.1 Definition of ADF

Distance field is defined as a scalar field describing the minimum distance from each point in the field to some surface. But accurate distance field is not necessary in our scenario because the shape of flames is highly complex and random. At the same time, human being's visual system is insensitive to the minor inaccuracy of the fire shape. Based on these observations, ADF which can be computed much more efficiently is employed in our paper to reduce computation.

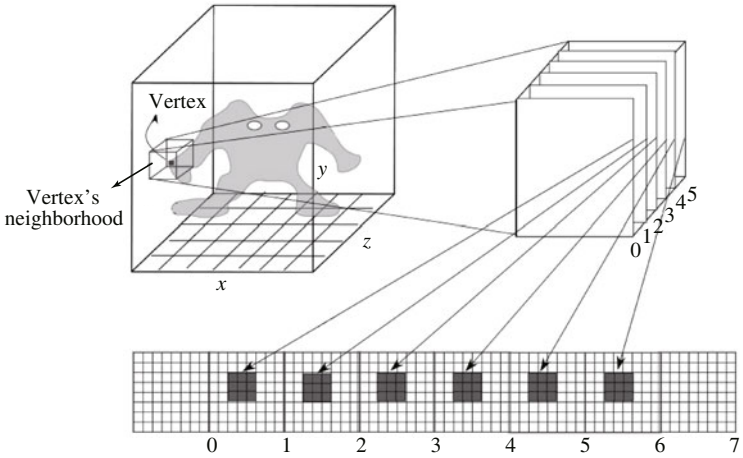


Figure 2 The data structure of bucket space and slice space texture.

ADF for object O with vertex set V is defined as the distance from each 3D point p to the closest vertex in V.

$$d_{\mathrm{ADF}}(p) = \min\{|p - v_i| : v_i \in V\}. \tag{1}$$

4.2 Building ADF on GPU

ADF cannot be constructed off-line because it is not static when objects are in non-rigid motions. Thus the construction of ADF need to be accomplished in real time. On the other hand, it is not necessary to build ADF for the whole 3D space because flames are always surrounding the objects. In our method, ADF is only computed in the 3D space occupied by the bounding box B of object O. More specifically, B is discretized to a 3D uniform grid called *bucket space* (see Figure 2) and the distance function in (1) is only computed at the center of each voxel. The 3D grid can be interpreted as a stack of 2D slices along the z-axis. These slices are tiled together to form a big 2D texture called *slice space texture*.

The most straightforward way to generate ADF on GPU is called *gathering process* (see Figure 3(a)). All vertices v_i in V are mapped to the closest voxel. And then each voxel u is traversed to find the closest mapped vertex in its 3D neighbourhood. Based on the data structure of slice space texture, all the above operations in 3D space can be converted to 2D operations supported by GPU. The disadvantage of the gathering process is the potential many-to-one mapping where multiple vertices in V might be mapped to a single voxel in bucked space. High resolution bucket space has to be adopted to alleviate this problem at the cost of high GPU memory requirement.

Inspired by the idea of *splatting* in point-based rendering, a method called *distribution process* (see Figure 3(b)) is employed to construct ADF efficiently. For vertex v_i, the distance d_i from v_i to all the voxels in its 3D neighbourhood is computed and recorded in the corresponding voxel. According to the definition of ADF, each voxel should record the minimum distance from its center to the closest vertex in V. This goal can be accomplished by assigning distance value d_i to the depth of corresponding voxel. Then the z-testing can automatically keep the minimum distance value at each voxel.

5 Flame rendering

Based on the method described in the above section, ADF can be constructed and updated in real time. In this section, we present the details about how to use the ADF to produce flames. First, the ADF is sampled and then the samples are disturbed by simplex noise to guarantee the randomness of flames. Finally the disturbed samples are converted to texture coordinates to fetch values from a precomputed 1D color distribution map which is generated by physical equations.

5.1 Sampling ADF

The sampling of ADF is accomplished by the ray-marching algorithm. This part is quite similar to the work presented in [2]. For each pixel in the final image, a ray is generated from the eye point and goes through the center of the pixel. The sampling process starts when the ray enters the ADF and it is terminated when the ray encounters the object or exits the field. In the sampling interval, the ray is evenly sampled to produce multiple 3D sample positions. Finally, samples can be generated from those positions.

The distance values d generated from the above method are too regular to represent the random shape of flames. In order to achieve visually plausible flames, simplex noise is employed to disturb the samples, as expressed by

$$d' = \text{Sample}(\text{ADF}, (x, y, z)) + \text{SN}(x, y, z), \tag{2}$$

where $\text{Sample}(\text{ADF}, (x, y, z))$ is the sampling function which produces a distance sample at sampling position (x, y, z), and $\text{SN}(\cdot)$ is the simplex noise function.

5.2 Generating color from distance

In order to generate visually plausible flames, a 1D color map C_{map} is precomputed from some physical equations. And the samples from ADF are converted to 1D texture coordinates to fetch colors from C_{map}.

According to [11], the temperature distribution in the flames can be described as follows: temperature increases from the ignition temperature to the maximum, and then cools down until the blackbody radiation is invisible. The whole process can be given by

$$\frac{\partial T}{\partial t} = -(u \cdot \nabla)T - C_T \left(\frac{T - T_{\text{air}}}{T_{\max} - T_{\text{air}}} \right)^4, \tag{3}$$

where C_T is cooling constant, and T_{air} is the temperature of surrounding air.

Horvath [1] simplified (3) by setting T_{air} to 0 into a new equation:

$$T^* = T - \Delta t \times C_T \times \left(\frac{T}{T_{\max}} \right)^4, \tag{4}$$

where T^* is the new temperature after time Δt .

After the temperature field is computed using (4), it can be converted to flame color according radiative transport equation (for more details, see [11]).

According to the above equations, a 1D color map is precomputed to describe the color distribution of flames. In order to fetch color values from the 1D color map, sample d' from the ADF is converted to 1D texture coordinates s by the following equation:

$$s = \left(\frac{d'}{d_{\max}} \right)^k, \tag{5}$$

where $d_{\max}$ is the maximum value of samples from ADF, k is a user-defined positive constant employed to control the flame colors.

Note that the physical method is not the only way to create the color map. Artists can also generate such maps according to the application requirements. The only requirement is that the color distribution has to roughly obey the rules presented in Figure 4.

6 Interacting with wind

Two distance fields are constructed to implement the interaction between flames and external factors (eg. wind). More specifically, Inner Approximate Distance Field (IADF) and Outer Approximate Distance Field (OADF) are built to simulate the inner and outer flames respectively. Our algorithm assumes that only the outer flames can be influenced by the external factors. Based on this assumption, the IADF is

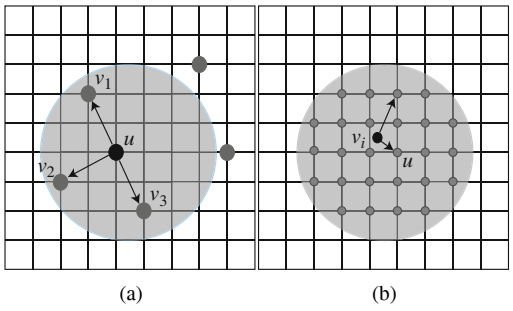


Figure 3 Gathering process vs distribution process. (a) Gathering process; (b) distribution process.

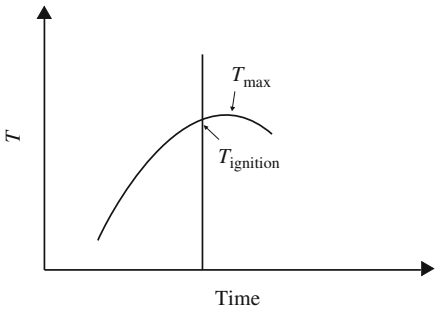


Figure 4 Flame temperature distribution can be described as a function of time.

built from the original vertex set V of the burning object, and the OADF is constructed by the translated vertices of V.

The translated function $\mathrm{Tr}(\cdot)$ is defined as

$$\mathrm{Tr}(v) = v + \tau_b \times b \times \mathrm{clamp}(\boldsymbol{N} \cdot \boldsymbol{B}, 0, 1) + \tau_w \times w \times \boldsymbol{W}, \tag{6}$$

where v and $\boldsymbol{N}$ are the position and normal vector of vertex respectively; b and $\boldsymbol{B}$ can be explained as the magnitude and direction of a upward force respectively; w and $\boldsymbol{W}$ represents wind in a similar way; τ_b and τ_w are two random numbers slightly disturbing the upward force and wind respectively. The translated vertices is illustrated in Figure 5.

Because wind will strengthen the randomness of the flames shape, a new disturbing equation is designed to simulate such effect. In addition to adding a random number to the sampled distance, a new random number generated from simplex noise $\mathrm{SN}'(\cdot)$ is employed to disturb the sampling position in y direction:

$$d' = \mathrm{Sample}(\mathrm{ADF}, (x, y + h \times \mathrm{SN}'(x, y, z), z)) + \mathrm{SN}(x, y, z), \tag{7}$$

where h is the height of the flame.

7 Results and discussions

The algorithm proposed in this paper is implemented on a PC with Intel Core2 Duo 2.93 GHz, 2 GB memory, NVIDIA GeForce GTX 260 video card with 896 MB memory.

Figure 6 compares the flames generated by the traditional simplex-noise-based method [2] with that by our method. The effect produced by [2] (see Figure 6(a)) looks like putting an object in flames instead of flames burning along objects surface. The flames produced by our method, as shown in Figure 6(b), are much more realistic. Note that though simplex noise is employed to disturb the sampling process, the discontinuity of flames is not observed between adjoining frames in our test.

Figures 7 and 8 demonstrate that the flames generated by our algorithm are controllable. Figure 7 shows that the size of flames can be controlled by using different h in (7). And Figure 8 indicates that the flames color can be changed by assigning different values to k in (5).

The flames generated by our algorithm can also respond to the winds with different directions and magnitudes, as shown in Figure 9.

Figure 10 shows a series of rendering results of an animated character in fire. This character is under a non-rigid motion in the animation and we can see that the flames change with the movement of the character.

In order to test the performance of our algorithm, we use three different sized animated models whose numbers of vertices are 1754, 3534 and 9199, respectively. The tests are conducted under three different output resolutions and the flames cover about 40% to 60% areas of the output image. According to the performance data listed in Table 1, the performance of our algorithm is mainly subject to the resolution of

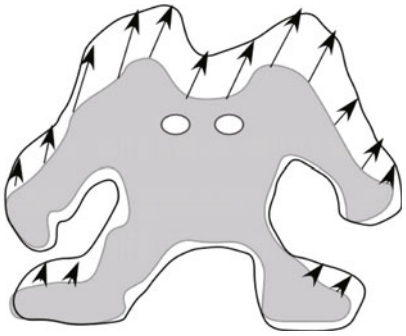
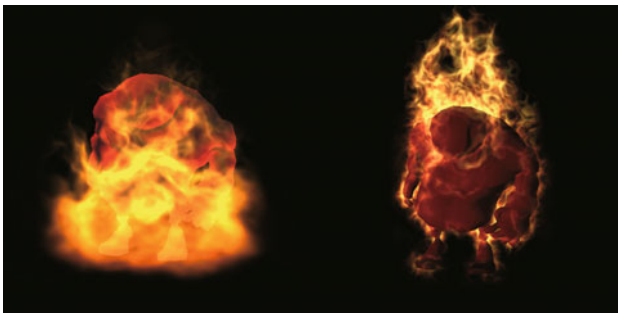
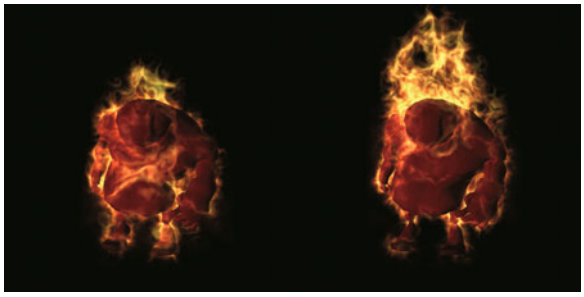


Figure 5 OADF generated from the translated vertices of object.



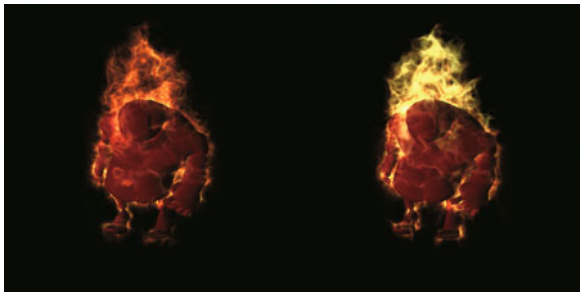
(a) (b)

Figure 6 A comparison between the flames produced by traditional simplex-noise-based method [2] and our method. (a) Traditional simplex noise algorithm; (b) ADF algorithm.



(a) (b)

Figure 7 The size of flames can be controlled by using different h. (a) $h = 0.35$; (b) $h = 0.55$.



(a) (b)

Figure 8 The color of flames can be controlled by using different k. (a) $k = 0.5$; (b) $k = 2$.

Figure 9 Flames can be influenced by wind with different magnitudes and directions.

Figure 10 Flames generated on deformable objects.

output image. This is because the sampling process of ADF is accomplished by a ray-marching algorithm whose performance is sensitive to the image resolution. On the other hand, the number of vertices has limited influence on our algorithm. Please also note that because the areas covered by the flames cannot be accurately controlled, sometimes the rendering efficiency for model with more vertices is slightly higher

Table 1 The performace (FPS) of our method tested with different objects and resolutions

	Bigguy	Doll	Horse
256×256	126	126	123
512×512	50	45	46
1024×1024	18	14	15

than the model with less vertices. This might provide another piece of evidence that our algorithm is not sensitive to the model size.

The quality of the flames generated by our method cannot match the results of the physically-based methods such as [1]. But our method has the advantage of real-time performance (see Table 1), whereas [1] takes hours to finish the simulations for one frame on 10 GPUs.

8 Conclusions

This paper presents a new real-time algorithm to simulate and render flames on arbitrary deformable objects. ADF and simplex noise are the core ideas of our method. The strategy of combining ADF and precomputed 1D color map avoids the complicated physical computations. Furthermore, ADF is a GPU-friendly representation for deformable objects which can be efficiently constructed on GPU in real time. Unfortunately, static ADF is too regular to represent the random shaped flames. In order to address this issue, simplex noise is used to disturb the sampling process of ADF. Then the sampled distance values are converted to the texture coordinates to fetch values from 1D color map to produce flames.

Acknowledgements

This work was supported by National Natural Science Foundation of China (Grant Nos. 60903118, 60832011) and National Key-tech R&D Program (Grant No. 2012BAH62F03).

References

1 Horvath C, Geiger W, Directable, high resolution simulation of fire on the GPU. ACM Trans Graph, 2009, 28: 1–8
2 Tatarinov A. Perlin fire. Nvidia Corporation Technical Report WP-03012-001v01, 2007
3 Sigg C, Peikert R, Gross M. Signed distance transform using graphics hardware. In: Proceedings of the 14th IEEE Visualization, Washington DC, 2003. 12
4 Perlin K. Improving noise. ACM Trans Graph, 2002, 21: 681–682
5 Inakage M. A simple model of flames. In: Proceedings of the 8th International Conference of the Computer Graphics Society on CG International Computer Graphics Around the World. New York: Springer-Verlag, 1990. 71–81
6 Flavien B L, Leblond M, Rousselle F. Enhanced illumination of reconstructed dynamic environments using a real-time flame model. In: Proceedings of the 4th International Conference on Computer graphics, Virtual Reality, Visualisation and Interaction in Africa. New York: ACM, 2006. 31–40
7 Wei X M, Li W, Mueller K, et al. Simulating fire with texture splats. In: Proceedings of the Conference on Visualization. Washington DC: IEEE Computer Society, 2002. 227–235
8 Rose B M. Real-time photo-realistic stereoscopic rendering of fire. Master's Thesis. Raleigh North Carlina: North Carolina State University, 2007
9 Reeves W T. Particle systems—a technique for modeling a class of fuzzy objects. ACM Trans Graph, 1983, 2: 91–108
10 Perry C H, Picard R W. Synthesizing flames and their spreading. In: Proceedings of the 5th Eurographics Workshop on Animation and Simulation, Oslo, 1994. 1–14
11 Nguyen D Q, Fedkiw R, Jensen H W. Physically based modeling and animation of fire. ACM Trans Graph, 2002, 21: 721–728
12 Pegoraro V, Parker S G. Physically-based realistic fire rendering, In: Proceedings of the 2nd Eurographics Workshop on Natural Phenomena. Aire-la-Ville Switzerland: Eurographics Association, 2006. 51–59
13 Minor D. Physical simulation of fire and smoke. Master's Thesis. Bournemouth University, 2007
14 Mller M, Charypar D, Gross M. Particle-based fluid simulation for interactive applications. In: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. New York: ACM, 2003. 154–159

15 Ishikawa T, Miyazaki R, Dobashi Y, et al. Visual simulation of spreading fire. In: NICOGRAPH International05, Morioka-Iwate, 2005. 43–48
16 Krger J, Westermann R. GPU simulation and rendering of volumetric effects for computer games and virtual environments. Comput Graph Forum, 2005, 24: 685–693
17 Hong Y, Zhu D M, Qiu X J, et al. Geometry-based control of fire simulation. Visual Comput, 2010, 26: 1217–1228
18 Zhu J, Bao K, Chang Y Z, et al. Simulation of solid burning phenomenon in real-time (in Chinese). J Comput Aid Des Comput Graph, 2011, 23: 11–19
19 Perlin K. An image synthesizer. In: Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques. New York: ACM, 1985. 287–296
20 Osher S J, Fedkiw R P. Level Set Methods and Dynamic Implicit Surfaces. Springer, 2003
21 Fisher S, Lin M C. Deformed distance fields for simulation of non-penetrating flexible bodies. In: Proceedings of the Eurographic Workshop on Computer Animation and Simulation. New York: Springer-Verlag, 2001. 99–111
22 Erleben K, Dohlmann H. Signed distance fields using single-pass GPU scan conversion of tetrahedra. In: Nguyen H, ed. GPU Gems 3. Upper Saddle River: AddisonCWesley, 2008. 741–763
23 Mauch S P. Efficient algorithms for solving static hamilton-jacobi equations. Dissertation of Doctoral Degree. Pasadena: California Institute of Technology, 2003