GPU Based Burning Process Simulation

Ran Jiao, Liu Yonggan, Hao Aimin
State Key Laboratory of Virtual Reality Technology and Systems, Beihang University
Beijing, China
E-mail: ranjiao@gmail.com

***Abstract*—We present a method of simulating the process of burning phenomena on generic polyhedral objects. By mapping the object's surface to a 2D space, the fire front expansion can be calculated efficiently on GPU (Graphics Processing Unit). The state of decomposition is updated according to the fire front and the consumption of solid fuel. During the simulation loop, both the fire front and the solid fuel consumption are updated respectively. In order to achieve a better performance, most routines of the simulation are processed on GPU. The entire simulation could run at an interactive rate on a normal PC.**

Keywords-burning; fire spreading model; deformation.

I. INTRODUCTION

Combustion is an important natural phenomenon which is widely used in virtual environment, such as video games, industrial simulations, etc. Although lots of researches are focused on fluid simulation, such as water, fire and smoke, most of these works are dedicated to fluid simulation itself. Little attention was paid to fire propagation and the objects being burnt. In today's game development, there's still no usable technique that can simulate solid combustion. However, real-time simulation of object's combustion process is increasingly attracting more attention, since it can dramatically improve the quality of fire simulation. Although physics based approach can produce convincing result [1], it's still too expensive for real-time application. Most of modern game engines like CryEngine 3 (game engine released by Crytek) and Unreal Engine 3 (a widely used computer game engine developed by Epic Games) are still using simple deformation animation [2] and particle system approach [3], which end up with coarse results and requiring artists to tune every burning solid.

While trying to simulate the process of burning, many physical concepts are involved. Firstly, the material and the geometrical structure of the object being burnt can affect the speed of fire propagation, which will affect the burning process significantly. Secondly, along with the combustion, the object will lose combustible stuff and decompose, which will change the shape of the object and have an influence on the fire propagation as well.

We demonstrate a real-time simulation method of burning phenomena on generic polyhedral objects. The main contributions of this paper are as follows:

- A GPU based algorithm to model the expansion of fire on polygon surface.
- Introduce a modified mass-string model to describe the physical deformation of a decomposing and burning mesh.
- Raise a method which can detect polygon self-intersection so as to avoid incorrect rendering results after topological structure changes.

Most routines of this method could be done on GPU. The burning state is stored as textures and additional vertex information in video card's memory. Since all data structure is stored in textures, nearly all the calculation is able to be done on GPU. The simulation quality of fire expansion could be adjusted by simply modifying the resolution of the burning state texture. The topological structure changes of the object during the combustion are updated correctly. Meanwhile, the accuracy of the simulation is not affected at all. This method can be applied easily to most polygon meshes.

The next section presents some related work about deformation, fire simulation and combustion simulation. Section III introduces a model simulation algorithm, describing how fire spreads on a mesh. There we will describe our method and how to process the whole algorithm on GPU. Section IV describes the physics model calculating the deformation of a burning mesh. The overall performance is excellent since the algorithms of both fire propagation and mesh deformation are calculated efficiently on GPU. Final results are presented in Section V.

II. RELATED WORK

A. Deformation

Free-form deformation (FFD) is widely used in both commercial software such as 3D Studio Max and real-time simulation as an important tool for computer-assisted geometric design and animation. FFD is firstly developed by Barr [1] and later improved by Sederberg and Parry [4]. A generic approach is proposed by Milliron et al [5].

Recently, some Laplacian coordinates-based deformation methods like VGL [6] are developed to achieve better results for large scale deformation.

B. Fire Simulation

The early models of fire simulation are mostly based on particle system firstly developed by Reeves [7]. Although article system is widely used in real-time rendering such as video games, the result it produces is still not convincing.

 ISBN: 978-1-61208-177-9

In order to achieve realistic and physics correct rendering result, physics based fire simulation is developed by using the algorithm of Computational Fluid Dynamics, CFD [8]. Jos Stam introduced SPH into computer graphics in [9], and later he proposed an unconditionally stable model to solve the Navier-Strokes equation in Stable Fluids [1], making physics based fluid simulation less expensive. Although physics based fluids simulation has been studied for over 20 years, its calculation is still too expensive for real-time rendering on current PC.

C. Combustion Simulation

Konrad Polthier and Markus Schmies [10] modified geodesic flow method to make it able to work on polyhedral surface. Reference [10] computes the evolution of distance circle on polyhedral surface and develops a method to visualize the geodesic circle. This method is used in [11] to simulate the fire front.

Hauyoung Lee and Laehyun Kim [11] used geodesic flow method to simulate the combustion process on polyhedral surfaces. In [11], fire fronts are evolved directly on the surface of arbitrarily complex objects by using modified geodesic flow method. Wind field model is also used to achieve animator control and motion complexity. Combustion process is treated as the propagation of fire front only, which is suitable to simulate fire spread on terrain, but cannot achieve complex results. Besides, this method did not take the decomposition of burning objects into consideration, which is necessary for most combustible materials.

Zeki Melek and John Keyser presented a burning objects simulation framework [12]. They simulated flame and solid separately, and used a heat transfer mechanism to transport energy between two systems. In flame simulation phase, a modified version of Stable Fluids [8] approach is used. The fluid solution is applied to fuel gas, exhaust gas and heat. In solid simulation phase the burning boundary is firstly computed by using level set method [13]. Then, a regular 3D grid is used to represent the decomposition of the solid and how much fuel left in the solid. If any part of the solid reaches the ignition temperature, it will start burning and release fuel to flame system.

Singguang Liu et al. proposed a unified framework [14] for simulation burning phenomena of thin-shell objects such as paper, cloth, etc. Fire spreading is modeled as burning state propagation on NxN cells of a 2-dimension space. They also proposed a method used to calculate the deformation of the thin-shell object based on the state of combustion, and then apply deformation by using FFD. Since the simulation is processed by CPU, this method can only achieve 10 frames per second.

III. FIRE SPREADING MODEL

A. Mapping

The burning state is composed of two stages. Firstly, every vertex has its own vertex state which records the burning state, deformation state, etc. Secondly, for points inside a triangle, the point state is updated and stored in a texture, recording how much this point is burnt.

The burning state space is treated as a float texture in our implementation. In order to use a texture to describe the fire spreading, a mapping from R^3 to R^2 is involved.

Every single triangle of the burning mesh should have its unique state, which requires the texture mapping should not be overlapped. In order to meet this requirement, a little more care should be taken while creating texture mapping.

Since the mapping from R^3 to R^2 will create discontinuous area, artists should also be careful about the part of discontinuous mapping, because burning state updating requires the neighboring vertices are also connected in burning state map.

B. Burning State

For a mapped point on burning state surface, it should be at one of the three states: normal, burning, and burnt. We define the burning state of point (x, y) at time step *n* as S_{xy}^n. The value of S_{xy}^n shows how much it has burnt. State value 0 means state normal, and state 1 means burnt. At the start of the simulation, at least one point on this surface is in burning state, and these burning points will ignite nearby points and spread the fire. If one point has been burnt for a certain time, it goes to state burnt. Burnt points will change its appearance and no longer get updated in fire front spreading and mesh deformation.

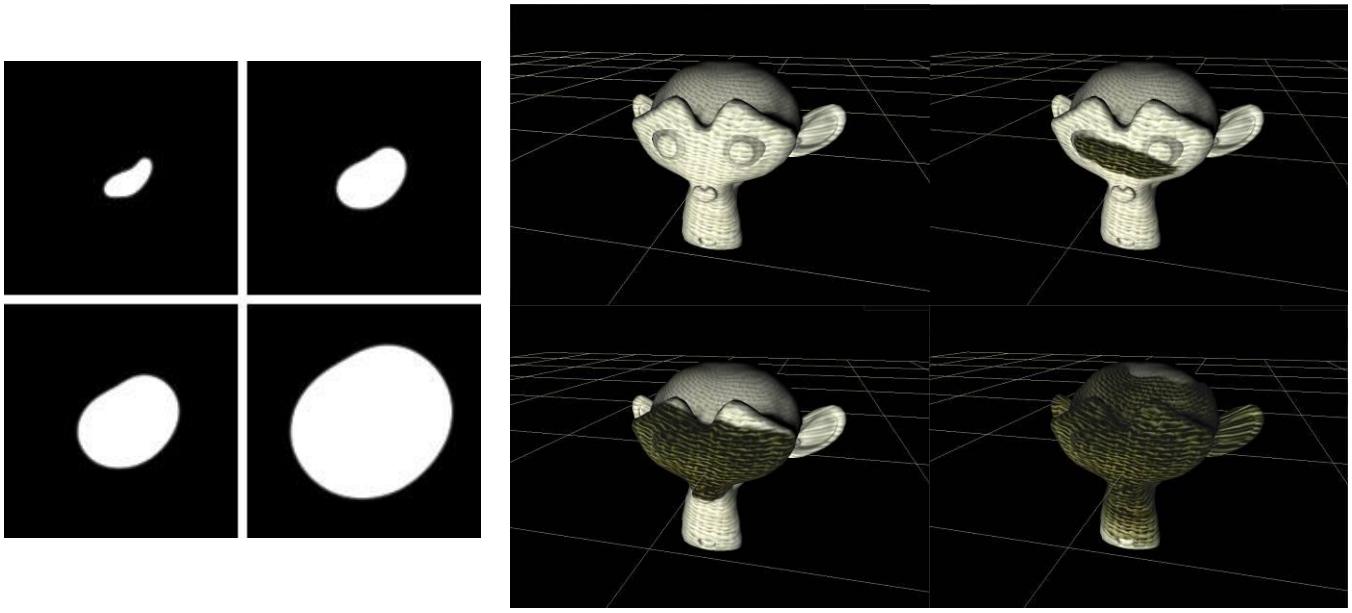
After mapping the mesh surface to a 2D surface, the fire propagation could be implemented as state changing on a texture.

For a burning state S_{xy}^{n+1} of point (x, y) at time step n+1, its state could be approximately evaluated by using the state S_{uv}^n of nearby points at time step n. The combustion state is updated each step by using a convolution in a region σ around the point (x, y):

$$\begin{aligned} S_{xy}^{n+1} &= \int_{(u,v)\in\sigma} W(u,v)D(u,v)S_{uv}^n \\ &\approx \sum_{(u,v)} W(u,v)D(u,v)S_{uv}^n \end{aligned} \tag{1}$$

$W(u,v)$ is the weight function, which should be in inverse proportion to the distance between *(x, y)* and *(u, v)*. $W(u,v)$ and area being summed up will affect the propagation speed. Each time we retrieve a value of S_{uv}^n a texture sampling will be invoke, since the burning state is stored in a texture. A much too big sigma area will significantly slow down this process. In our implementation we will sample 16 pixels around point *(x, y)*. *D(x, y)* is the propagation speed of fire front:

$$D(u,v) = C_{uv}C_D \cdot (1 + C_G \cdot \overrightarrow{N_{uv}} \cdot \vec{G}) \cdot \Delta t \tag{2}$$



(a) (b)

Figure 1. The burning process on the burning state map. Burning state texture is showed in (a), the color of the texture represents how much it's burnt: black is not ignited yet, white is burnt. The burning speed and quality could be adjusted by using parameters. After mapping the burning state texture to the mesh, it could be used to calculate the color and vertex position, demonstrated in (b).

$\vec{G}$ is the direction vector of gravity, and $\overrightarrow{N_{uv}}$ is the normal direction of point *(u, v)*. C_D is a constant parameter which controls the overall burning speed, and C_G is a constant parameter about how much the burning speed is affected by burning direction. We introduce gravity direction $\vec{G}$ here for the reason that the flame is always burning against the direction of gravity, and the burning speed is also affected by the angles between surface normal and gravity direction. C_{uv} stands for the material multiplier for point (u,v), which is stored in a texture to describe the burning speed for every point on the mesh surface.

As all calculation above is manipulating texture, it is easy to have this whole process implemented with GPU acceleration. This makes our implementation far more efficient than Liu's in [14].

IV. DEFORMATION

A. Modeling Deformation

The burning mesh is deformed by two forces: the string force F_S caused by nearby solid surface's deformation and the decomposition force F_D caused by the consumption of inner solid fuel:

$$F = \alpha F_S + \beta F_D \tag{3}$$

α and β are parameters of object material, which demonstrate how much these two forces should affect the object. In our final result, we employ α = 0.5 and β = 3.0, making a wood like burning deformation.

F_S is composed of structure force, shear force and flexible force [15]:

$$F_S = F_{\text{structure}} + F_{\text{shear}} + F_{\text{flexion}} \tag{4}$$

The consumption of solid fuel is calculated and stored while updating the burning state. Using this information of fuel consumption, we can calculate the force applied to the mesh caused by the decomposition.

Consider a small part of a burning mesh demonstrated in Figure 2. we can resolve the movement of vertex A caused by decomposition by calculating how much its volume changes:

$$\begin{aligned} \Delta B_A &= B_A - B_{A'} \\ &= D_C \cdot (V_{ABCDE} - V_{A'BCDE}) \\ &\approx D_C \cdot P_A P_{A'} \cdot S_{BCDE} \end{aligned} \tag{5}$$

The B_A represents how much the fuel is left, P_A and $P_{A'}$ are the vertex's original and transformed position respectively. V_{ABCDE} and $V_{A'BCDE}$ are the volume of the origin and transformed area. D_C is the density of the fuel, and S_{BCDE} is the size of polygon *BCDE*. The consumption of the solid fuel is approximately evaluated by measuring the distance the vertex has moved.

According to (5), $P_A P_{A'}$ could be resolved:

$$| P_A P_{A'} | \approx \frac{\Delta B_A}{D_C \cdot S_{BCDE}} \quad (6)$$

By using the normal vector N_A, we can approximately get the offset vector:

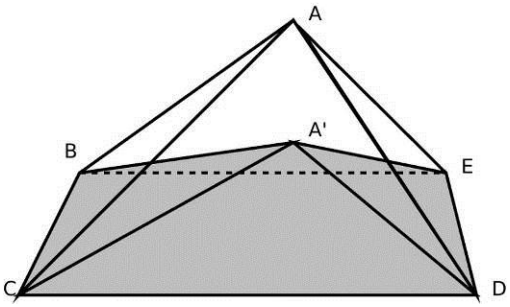


Figure 2. The deformation is calculated by evaluating how much solid fuel lost after vertex *A* moved to *A'* caused by decomposition.

According to the classic physical theory, we can get this equation:

$$F_D = m \cdot a$$
$$dP_A = \frac{1}{2} a \cdot dt \quad (7)$$

Where m is the mass of mesh in Figure 2, a is the acceleration, and dt is the time step in our simulation. Then we can get F_D:

$$F_D = m \cdot a = m \cdot \frac{2dP_A}{dt} \quad (8)$$

In our implementation, the burning state is stored in a texture so that we can use GPU to calculate the fire front's expansion. So we need to sample the burning texture according to adjacent triangles' size.

Before each frame's deformation, we downsample the burning state map into n stages by halving the texture resolution. For each of these textures, the size a pixel represents is $\frac{S}{w \cdot h} \cdot 2^n$, and we can find a n' satisfying this equation:

$$\frac{S_{n'}}{w \cdot h} 2^{n'} \le S_A k < \frac{S_{n'+1}}{w \cdot h} 2^{n'+1} \quad (9)$$

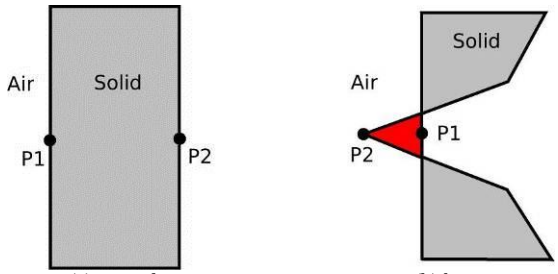
Where k is the how many pixel need to be sampled. k is the number of pixels we sampled in equation (1), in our implementation k = 16. w and h stand for the width and height of the burning state texture. By sampling the n'th burning state texture we can know how much fuel is used and get ΔB_A for (6).

After getting both F_D and F_S, we can compute how much this vertex should move along its normal direction. Since the area size is used in (7), we need to precompute the area size of the mesh before rendering, and pass it to the video card as vertex attributes. The normal vector of the vertex should also be updated as the faces are all deformed.

B. Topological Changes

Along with the combustion process, the topological structure might change. We use a simple method based on video cards' graphical pipeline function to avoid incorrect rendering results.

Assume an object with two points P_1 and P_2, which face the opposite direction. At the beginning, they do not cross each other, but after being burnt, the vertices will be pushed backwards according to their normal vector and burning state. On certain condition, P_1 and P_2 may get crossed as demonstrated in Figure 4. This will certainly ends up with incorrect rendering results. However, this could be fixed by adding a special pass to the rendering loop.



(a) normal (b) burnt

Figure 3. Combustion caused topological structure changes. Considering eyes looking from right to left, P_2 in (a) should be the front face, and P_1 on back face. After burnt P_2 might goes to the back of P_1 in (b). Red area in (b) will cause incorrect rendering with normal rendering routine.

A special Z-Pass and a depth test are introduced to eliminate incorrect rendering results caused by topological changes. In render loop, Z-Pass is used just after updating the burning state. Both front and back faces are rendered and have their depth written to a render target with depth test turned on. If a front face gets blocked by a back face just like what happens in Figure 4(b), the render target will have the back face's depth (it's also the nearer face) saved.

During the rendering pass the back face cull is turned on, and each vertex's depth value is compared to value stored in render target. Pixel in red area of Figure 4(b) will have further depth value than that we stored in Z-Pass, which means it has already crossed the face of another size, and this pixel will be discarded.

In our approach to cull incorrect pixels, self-collision detection is changed into a screen space problem. Since our technique used to detect topological changes can be implemented as a simple post-process, it won't be affected by mesh's vertices count and will run very fast on GPU.

 ISBN: 978-1-61208-177-9

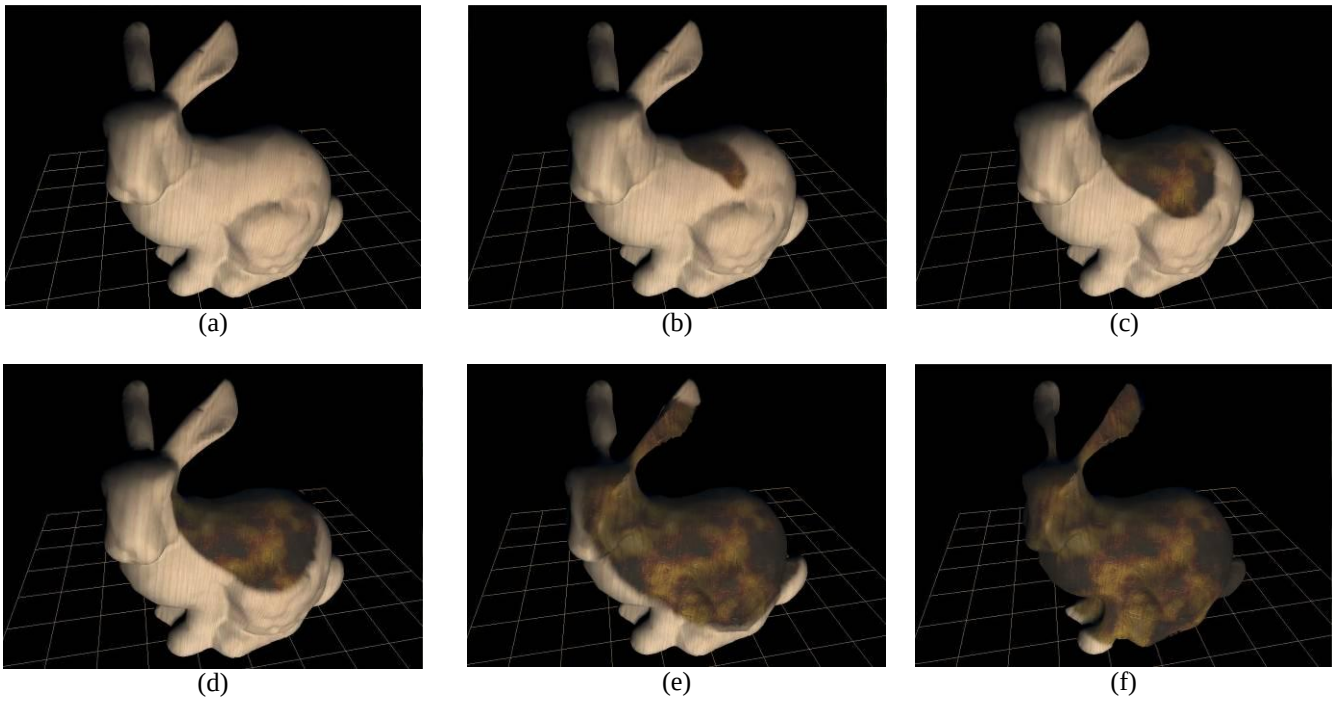


Figure 4. Combustion simulation of a bunny. The bunny mesh is composing of 4000 vertices and a 1024x1024 burning state texture.

V. RESULTS

Using the method we have introduced, we are able to simulate various kinds of burning meshes. The simulations are ran on Nvidia Geforce GT 425m, Intel Core i5 (2.53G Hz), 4G RAM and 1024x768 screen resolution. A 256 × 256 burning state texture and a mesh of 4000 vertices are used in scene and we got more than 100 fps. The result is simulated with α = 0.2, β =1.0, C_D =0.3, C_G =0.1.

The whole process is shown in Figure 3. We initialize the burning state with a small area already burning on bunny's back, and the fire spreads all over the mesh. Together with the burning process the mesh also shrink and deformed. You can see the fire combustion process is like the one in [14], but utilized on a normal solid mesh with much better performance.

TABLE I. PERFORMANCE

Burning State Texture Resolution	Mesh Vertices Count	Time Consumption(ms)		
		Fire Front	Self-Collision	Deformation
128x128	4000	2.4	4.9	1.4
	31000	2.4	4.9	1.6
256x256	4000	2.5	4.8	1.4
	31000	2.5	4.9	1.6
512x512	4000	2.5	4.8	1.4
	31000	2.5	5.0	1.6
1024x1024	4000	2.7	4.8	1.4
	31000	2.7	5.0	1.6

The performance with different state texture resolution and different mesh for algorithm described in this paper is listed in TABLE I. The performance of fire front and self-collision calculation is almost not affected by mesh vertices count, because both of them are screen space algorithm. The overall performance benefits a lot from GPU's parallel computing capability, it's easy to render more than 100 frames per second.

VI. CONCLUSION AND FUTURE WORK

We have proposed a method of simulating combustion process of general polygon mesh, and it is able to be applied to most meshes. This method can run much more efficiently than similar methods in [11] and [13], since all calculation is done by GPU and only the deformation algorithm is affected by the scene's complexity. In order to have topological structure change processed more elaborately, the mesh self-intersection could be calculated by using physics engines such as Nvidia PhysX, and recalculate the topological structure of vertices and triangles. Other deformation model could also be used to simulate the decomposing process of burning mesh. In our work we focused on fire expansion and mesh deformation rather than the fire simulation along with burning. In order to render fire during combustion process, the burning state could also be used for fire simulation. Wind field [16] could also be used to control the burning process and would produce more realistic result.

REFERENCE

[1] J. Stam, "Stable fluids," in Proceedings of the 26th annual conference on Computer graphics and interactive techniques, New York, NY, USA, pp. 121-128, 1999.

[2] S. F. Gibson and B. Mirtich, B. "A Survey of Deformable Models in Computer Graphics," Tech. Rep. TR-97-19, Mitsubishi Electric Research Laboratories, Cambridge, MA, November 1997.

[3] N. Froster and R. Fedkiw, "Practical animation of liquids," Proceedings of SIGGRAPH 2001. New York, NY: ACM, pp 23-30, 2001.

[4] T. W. Sederberg and S. R. Parry, "Free-form deformation of solid geometric models," in SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques, New York, NY, USA, pp. 151-160, 1986.

[5] T. Milliron, R. J. Jensen, R. Barzel, A. Finkelstein, "A framework for geometric warps and deformations," ACM Trans. Graph., vol. 21, no. 1, pp. 20-51, January 2002.

[6] Z. Kun, J. Huang, J. Snyder, X. Liu, H. Bao, B. Guo, et. el, "Large mesh deformation using the volumetric graph Laplacian," ACM Trans. Graph., pp. 496-503, 2005.

[7] W. T. Reeves, "Particle Systems a Technique for Modeling a Class of Fuzzy Objects," ACM Trans. Graph., vol. 2, no. 2, pp. 91-108, April 1983.

[8] T. J. Chung, Computational Fluid Dynamics. The Pitt Building, Trumpington Street, Cambridge, United Kingdom: Press Syndicate of Cambridge University, 2002.

[9] J. Stam and F. Eugene, "Depicting fire and other gaseous phenomena using diffusion processes," Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, pp. 129-136, 1995.

[10] K. Polthier and M. Schmies, "Geodesic Flow on Polyhedral Surfacse," in Data Visualization, Springer Verlag. Proceedings, 1999.

[11] H. Lee, L. Kim, M. Meyer, and M. Desbrun, "Meshes on Fire," in In EG Workshop on Computer Animation and Simulation, pp. 75-84, 2001.

[12] Z. Melek and J. Keyser, "Interactive Simulation of Burning Objects," in Pacific Conference on Computer Graphics and Applications, pp. 462-466, 2003.

[13] R Malladi, J A Sethian, and B C Vemuri, "Shape modeling with front propagation: a level set approach," in Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 17, pp. 158-175, Feb 1995.

[14] S. Liu, Q. Liu, T. An, J. Sun, and Q. Peng, "Physically based simulation of thin-shell objects' burning," THE VISUAL COMPUTER, vol. 25(5-7), pp. 687-696, 2009.

[15] X. P. Institut and X. Provot, "Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior," in In Graphics Interface, pp. 147-154, 1996.

[16] K. L. Gay, L. Ling, and M. Damodaran, "A quasi-steady force model for animating cloth motion," in Proceedings of IFIP International, pp. 357-363, 1993.

 ISBN: 978-1-61208-177-9