CIS 563 Physically Based Animation

Level Set Fire Simulation

Naicheng Zhang, Jing Qiu, Zimeng Yang
University of Pennsylvania

1. Introduction

We implement a level set fire simulation that handles the physical deformation of the blue core and hot gaseous product, based on the 2002 SIGGRAPH paper "Physically Based Modeling and Animation of Fire" [Nguyen et al.2002]. We track our reaction zone (blue core) using the level set method to track the moving implicit surface. And by changing the advection technique we can also simulation large scale flame and small scale flame, for example, modeling candle. The incompressible Navier-Stokes equations are used to independently model both vaporized fuel and hot gaseous products. We use the physically based model for the expansion that takes place when a vaporized fuel reacts to form hot gaseous products. We render the blue core that results from radicals in the chemical reaction zone where fuel is converted into products according to Nguyen's paper [Nguyen et al.2002]. We solve the blue core deformation by one of the discrete implementations called Fast Marching Method, based on the book "Level Set Method and Dynamic Implicit Surface" [Osher 2003].

2. Approach

2.1 Physically Based Model

We consider two distinct visual phenomena associated with flames. The first of these two is the blue core or the so-called inner flame. The blue colors are the emission lines from intermediate chemical species, such as carbon radicals, produced during the chemical reaction. This thin blue core is located adjacent to the implicit surface (which is also the flame front). Therefore, in order to track this blue core, we need to track the movement of the implicit surface. By implementing Fast Marching Method, we can solve the movement of the blue core using signed distance values of each Grid. These concepts will be explained in detail later in this report. The second visual phenomenon is the blackbody radiation emitted by the hot gaseous products, in particular the carbon soot. This is characterized by the yellowish-orange color familiarly associated with fire. In order to model this with visual accuracy, we need to track the temperatures associated with a flame

By implementing the Y value tracking the elapsed time when a grid leaves the flame front, we can realize the temperature sculpt in figure 1.

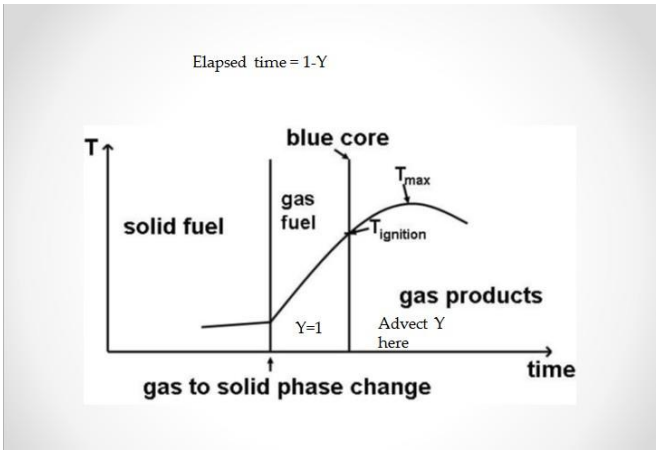

Figure 1

We start in the gaseous state region in the figure ignoring the solid fuel. The gas fuel will first heat up until it reaches its ignition temperature corresponding to our implicit surface and the beginning of the thin blue core region (In the Temperature and Density section later, we will show that the temperature inside the blue core can be easily set to the ignition temperature, in other words, a constant inside the implicit surface). The temperature continues to increase as the reaction proceeds reaching a maximum before radiative cooling and mixing effects cause the temperature to decrease. We use a linear equation to model this effect of temperature rising utilizing the elapsed time featured by Y value. As the temperature decreases, the blackbody radiation falls off until the yellowish-orange color is no longer visible. During this process, the temperature is solved using the same method in our smoke simulation homework with an additional term that deals with the cool down constant.

2.2 MAC-Grid Incompressible Fluid

We use one set of incompressible flow equations to model the fuel and a separate set of incompressible flow equations to model the hot gaseous products and surrounding airflow. We couple these two sets of incompressible flow equations together across the interface in a manner that models the expansion that takes place across the reaction front. We use the equations of mass and momentum conservation for the coupling across the thin flame front:

$$\rho_h(\mathrm{V}_h - \mathrm{D}) = \rho_f(\mathrm{V}_f - \mathrm{D}) \tag{2.1}$$

$$\rho_h(\mathrm{V}_h - \mathrm{D})^2 + p_h = \rho_f(\mathrm{V}_f - \mathrm{D})^2 + p_f \tag{2.2}$$

Similar to smoke simulation homework, we use a semi-Lagrangian stable fluids approach for finding the intermediate velocity. The difference is that one set of incompressible flow equations may cross the implicit surface and query the velocities from the other set of incompressible flow equations. In this case, we compute a ghost velocity using equation (2.1). After computing the intermediate velocity for both sets of incompressible flow equations, we solve for the pressure and find our new velocity field using conjugate gradient method.

The two sets of pressure field is coupled based on the method in [Nguyen et al. 2001] and equation (2.2). We implement the method in the following way. First, we compute the pressure in the whole field only considering solid boundaries. We use this solution as the set of pressure inside the blue core. Then using equation (2.2), we can compute the pressure value in the reaction zone. By constructing new A matrix and new d, we solve $Ap = d$ again for the pressure outside the blue core. The key idea is that p has some entries that we've already solved. For those points $(\mathrm{i}, \mathrm{j}, \mathrm{k})$, we set $A.diag(\mathrm{i}, \mathrm{j}, \mathrm{k}) = 1$ and the corresponding entry in d.

2.3 Level Set Method

2.3.1 Tracking the Implicit Surface

The implicit surface is represented by a signed distance function, ϕ. We use the staggered grid model to store the signed distance at the center of each voxel. In our implementation, we initialize a simple model of an implicit surface and computed the signed distance from each voxel center to the surface. The value of ϕ is positive inside the surface and negative outside the surface.

Because the surface is implicit, it is not represented by a group of voxels with zero values of ϕ. It is very important to track the exact locations of the surface at each step. To do this, we loop through each voxel and check if its value of ϕ has a different sign with its neighbors. If so, the voxel is marked as on the surface.

2.3.2 Reinitialization of signed distance

To make the signed distance well-conditioned, we need to do reinitialization at every few steps to recalculate the values of ϕ. This is implemented based on the Fast Marching Method mentioned in the book "Level Set Methods and Dynamic Implicit Surfaces" [Osher 2003].

After we have located the surface, we keep these values unchanged and add them to the initial accepted band for the signed distance values. Then we use the accepted band to march outwards and approximate the signed distance of all other grid points. We start from the inside of the surface and add the grid points adjacent to the inside surface to the tentative band. The order of the marching is decided by the values in the tentative band. The grid point with the smallest value is put into the accepted band and we update the signed distance of its neighbor grid cells by solving the quadratic equation of (2.3)[Osher 2003].

$$\left(\frac{\phi_{i,j,k} - \phi_1}{\Delta x}\right)^2 + \left(\frac{\phi_{i,j,k} - \phi_2}{\Delta y}\right)^2 + \left(\frac{\phi_{i,j,k} - \phi_3}{\Delta z}\right)^2 = 1 \qquad (2.3)$$

One crucial step is that once the ϕ value of a grid point is added to the accepted band, its neighbors who already have a tentative approximation need to be updated with the newly added information. The algorithm is repeated until the signed distance of the inside spot of the surface have all been updated. The outside of the surface is updated using the same way,

starting from the outside surface until all outside areas have been updated with a new signed distance.

In our implementation, we use heap to store all the tentative values. But the algorithm is still time-consuming because the values in the tentative band need to be updated for many times. However this Fast Marching Method proves to be fairly accurate according to the results of the simulation.

2.3.3 Moving the Level Set

The level set moves with the velocity of the gaseous fuel represented by $w = u_f + Sn$. In which $n = \nabla\phi / |\nabla\phi|$ is the gradient of the signed distance defined at the center of each grid cell. In the implementation we move the entire level set to capture this. Based on [Nguyen et al.2002], the motion of the level set is calculated by equation (2.4).

$$\phi^{new} = \phi^{old} - \Delta t(\mathrm{w}_1 \phi_x + \mathrm{w}_2 \phi_y + \mathrm{w}_3 \phi_z) \tag{2.4}$$

2.4 Temperature and Density

To track temperature time history, we create another field in MACGrid class called mY, which indicates the Y value we use to calculate the elapsed time. We go back to figure 1. This figure clearly depicts the time history of the temperature of fluid elements, we need a way to track individual fluid element as they cross over the blue core. In a word, we need a value for every cell to hold the elapsed time when the element cross the flame front and this value needs to be reset when the cell enters the blue core once again. This is easily accomplished using the equation (2.5) [Nguyen et al.2002].

$$Y_t = -(\mathrm{u}\cdot\nabla)\,\mathrm{Y} - \mathrm{k} \tag{2.5}$$

If we initialize Y to 1, in other words, $Y(0) = 1$ in the region of space occupied by the gaseous fuel and solve equation (2.5) for Y, then the local value of 1-Y is equal to the total time elapsed since the fluid element crossed over the flame front.

We then use this elapsed time to solve the temperature of that particular cell. Since the ignition temperature ($T_{ignition}$ in the source code) is usually below the outer flame (gaseous product) emission threshold, the temperature we set inside the blue core is not important. Therefore, we just set all the temperature inside the blue core to $T_{ignition}$. The region between the blue core and the maximum temperature in figure 1 models the rise in temperature due to the progress of a complex chemical reaction. For the sake of efficiency, we use a linear function of $1-Y$ to sculpt this rising curve and mapping the elapsed time to T. This is an important feature for the small scale flame we simulated in the result section. After passing the maximum temperature, or the peak point of the curve in figure 1, the temperature drops due to the flame element entered the falloff region. Using the physically correct equation (2.6), we can simply sculpt the temperature for the values of Y in the falloff region.

$$T_t = -(\mathrm{u}\cdot\nabla)\mathrm{T} - \mathrm{c}_T\left(\frac{T - T_{air}}{T_{\max} - T_{air}}\right)^4 \tag{2.6}$$

We solve this equation by first using the semi-lagrangian stable fluid method to solve for the convection form similar to the advectTemperature function in our Smoke Simulation homework. Then we manually add the fourth power term to cool down the flame at a rate governed by the cooling constant.

The advectDensity function is exactly the same as the Smoke Simulation. In the temperature falloff region, we solve the density advection equation again using semi-lagrangian stable fluid method.

2.5 Rendering

The temperature of the fire varies in time and space. Initially we implemented the rendering using the temperature. The blue core had a constant temperature value and was set to the color of blue. And the temperature of the hot gaseous products increased first to a maximum temperature value, which was render in linearized orange colors, and then decreased until it is smaller than a set constant and became invisible. The decreasing part of the temperature was also rendered using linearized orange colors. However the effect was not very satisfactory. Then we changed our rendering method using the value of ϕ. The way to implement it is the same but the results were more vivid.

3. Challenges Faced

As the reader observed from the implementation and result section of this report, our simulation lacks careful rendering and the final effect is rather "humble". The main reason for this is that the difficulty of the theory behind the actual implementation caused our project team to focus more on the theoretical understanding of the paper and the correct implementation of the Fast Marching level set method. About 30% of project working hours are devoted to understand the theory first. And it took us nearly 40% of total time to debug the Signed Distance (ϕ value) and the Fast Marching Method depicted in "Level Set Method and Dynamic Implicit Surface" [Osher 2003] after the actual implementation of the method. Some detailed obstacles we faced during the project is described below.

3.1 Initializing a closed implicit surface

We choose to initialize the implicit surface as a sphere, assigning ϕ according to the distance between the particular cell and the sphere center. At first the surface is not closed, the ϕ value is incorrect as the function advectPhi() is called the first time. The ϕ value inside the sphere would be advected outside the sphere. We solved the problem by adding a bias on the radius of the sphere.

3.2 The deformation of the implicit surface

In the middle of debugging the signed distance computation, the implicit surface is deforming incorrectly. No matter what initial shape of the surface is, the implicit surface will always become a square. The correct implementation is shown in our video of a sphere rising up without deformation. To solve this weird deformation problem, we tried a lot of debugging techniques. Eventually, we put our program in MATLAB and output the ϕ values from the accepted band and found out that we updated the pre-assigned ϕ values in the heap, but we didn't update ϕ in place. This mistake cause the length of the edges of x,y,z axis directions not changing and this lead to a square of the shape eventually.

3.3 Shrinking blue core

In our implementation, we calculated the gradient, which is the normal vector of the signed distance and stored them at the center of each voxel. At first in the simulation of the blue core, the shape tended to shrink at each time step. After looking into our code we found that the direction of the normal vector of ϕ is very critical and easily to be confused. The normal vector of ϕ was used in the advection of ϕ and also in the projection. We have to pay close attention and have a good understanding of the physical meaning of the model before implementing them.

3.4 Pressure coupling

When implementing the project function, we faced the challenge of correctly solving the discontinuity of the pressure according to the equation (2.1). The first method we come up with is solving the pressure as before similar to the Smoke Simulation and then only update the pressure of the cells that are on the implicit surface. The updating mechanism is utilizing equation 2 to compute the pressure on the outer surface from the pressure of the inner surface. But, this is incorrect. PCG is not converging after implementing this method. After several days of struggle, we come up with a better idea that is carefully illustrated in the section 2.2 MAC-Grid Incompressible Fluid.

4. Implementation Results

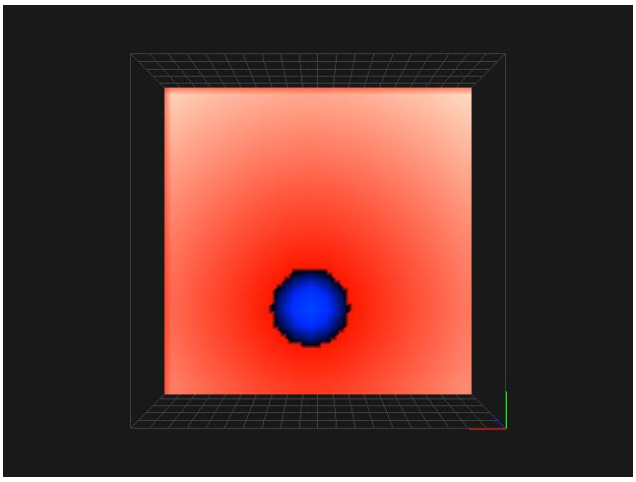

Figure 2 Visualization of the Level Set Method

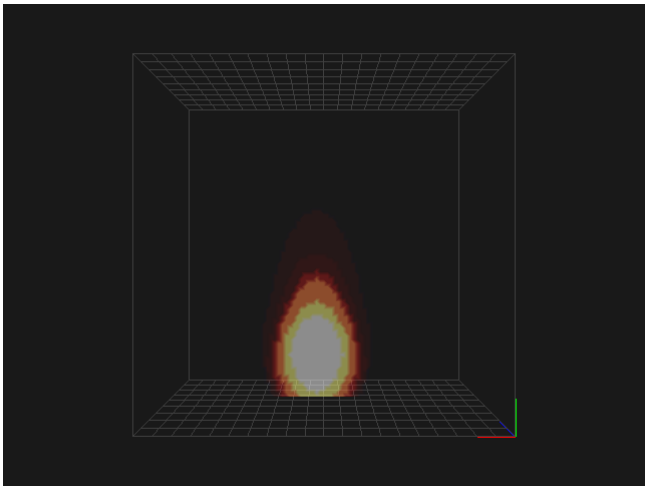

Figure 3a Level Set Fire

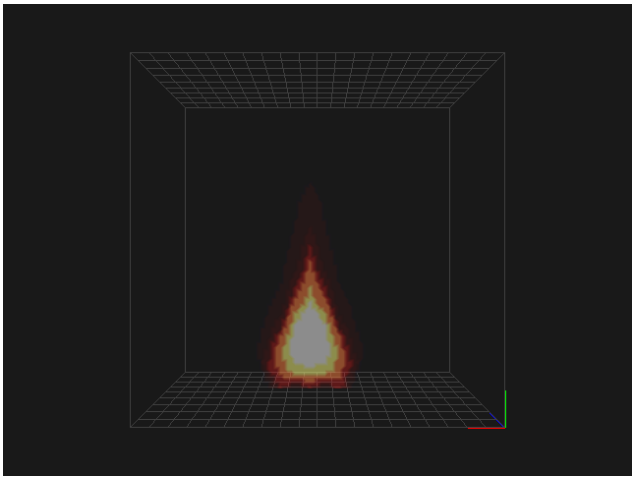

Figure 3b Level Set Fire

5. Project Team

The breakdown of project into work done by each person in our team is shown in the following table.

Members	Initialize Phi	Advect Velocity	Compute surface	Reinitia-lization	Advect Phi	External Forces	Project-ion	Debug
Naicheng			√	√	√		√	√
Zimeng		√	√	√	√			√
Jing	√			√		√	√	√

6. Base code

We used the base code from the assignment Smoke Simulation, from which we inherited the base data structure and work flow. The advection of density stayed unchanged and our team implemented all other features mentioned above.

7. References

Robert Bridson, Mattias Muler-Fischer. Fluid simulation. SIGGRAPH 2007 Course Notes.

Duc Quang Nguyen, Ronald Fedkiw, Henrik Wann Jensen. Physically Based Modeling and Animation of Fire. Proceedings of SIGGRAPH 2002.

Stanley Osher, Ronald Fedkiw. Level Set Methods and Dynamic Implicit Surfaces. Springer-Verlag New York, Inc 2003.

Duc Q. Nguyen, Ronald P. Fedkiw, Myungjoo Kang. A Boundary Condition Capturing Method for Incompressible Flame Discontinuities. J. Comput. Phys. 172, 71–98.

J. A. Sethian. A Fast Marching Level Set Method for Monotonically Advancing Fronts. Proc. Nat. Acad. Sci. 93,15911595, 1996.