Name: Chenyang Yu (862052273) Fei Yi (861305814)

Topic:

Physically Based Modeling and Animation of Fire

Achievement:

Starting from the fluid simulation, which is common used for simulating liquid but also perfect to represent gaseous movement. The solid(or liquid) object oxide in a short time interval and emits the heat and light, causes the flame we see. We simulate the heating process through the temperature model. The temperature is raise with time and eventually cool down by air. Although the original paper designed many different scenario to show different kinds or fire(different phase between solid fuel and gaseous fuel), our implement simplified it as a fixed injection source. Within a fixed surface, gaseous fuel is injected with initial density, velocity, and temperature. The density and velocity is follow by the incompressible Navier-Stokes equation.

Model

The following models aim to calculate the density, velocity, and the temperature for the next frame.

Flow:

Use Navier-Stokes equation to represent the flow. We ignore viscosity and gravity so the equation is reduced to

$$\frac{du}{dt} = -(u \cdot \nabla) \cdot u - \frac{1}{\rho} \nabla p + f$$

Where *u* is the velocity, ρ is the density.

Temperature and Density:

Both metrics can be generate from previous frame. The flow equation decides the shape of particle (fire). However, it is the temperature that makes the visually difference. We set the initial temperature and update it with time (just like the density) The reaction coordinate is described as

$$Y_t = -(\mathbf{u} \cdot \nabla) Y - k,$$

Where k is a positive constant
The temperature starts to fall down after it reaches the maximum temperature until it no longer visible. The falloff region temperature is as following equation

$$T_t = -(\mathbf{u} \cdot \nabla) T - c_T \left(\frac{T - T_{air}}{T_{max} - T_{air}} \right)^4$$

As we can see, the more difference between the target and the air, the faster it cool down. Once the temperature is so low to be visible, it would not recover back to the gas fuel but become a little soot (or smoke) and disappear in the air. In the reality, it becomes invisible because the volume too small.

Pseudo code:

```
initialize();
while(simulating){
        V = vel_step(V_old);
        (Density, Heat) = dens_temp_step(Density_old, Heat_old, V);
        draw();
}
```

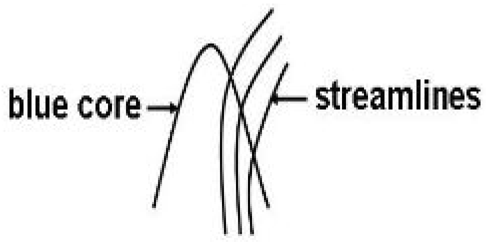
Buoyancy Force

(1) principle

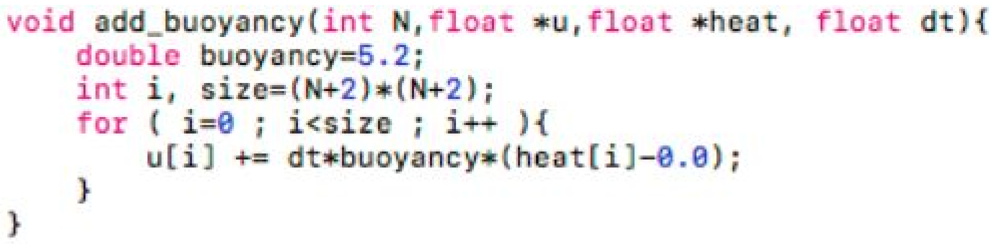
Notice that we also have to consider the *f* . We ignore the resistance and other force(too little and too complicate) and only consider buoyancy here. It is largely change the particle’s movement due to the change of temperature. It is make sense since we know the reason why fire burns upward is because of the hot vaporous convects. The buoyancy force is:

$$\mathbf{f}_{buoy} = \alpha\,(T - T_{air})\,\mathbf{z},$$

Where z is (0,0,1) point upward, T is the target temperature, T_{air} is ambient temperature of air. The more difference between these two, the bigger force it is. The following figure show the path of the some particles. Of course we expect our flame is more turbulent instead of a smooth candle-like flame.



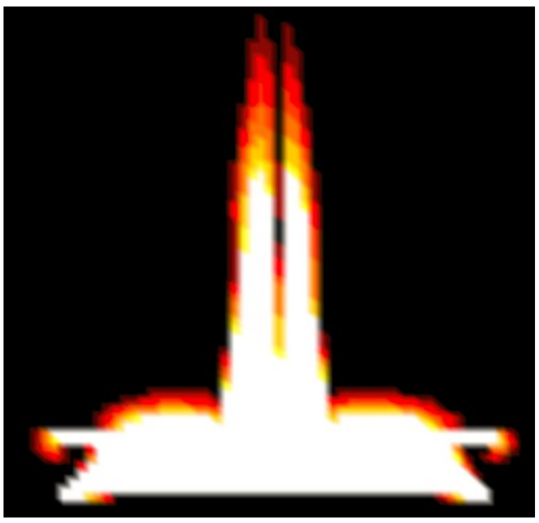
(2) Our implementation



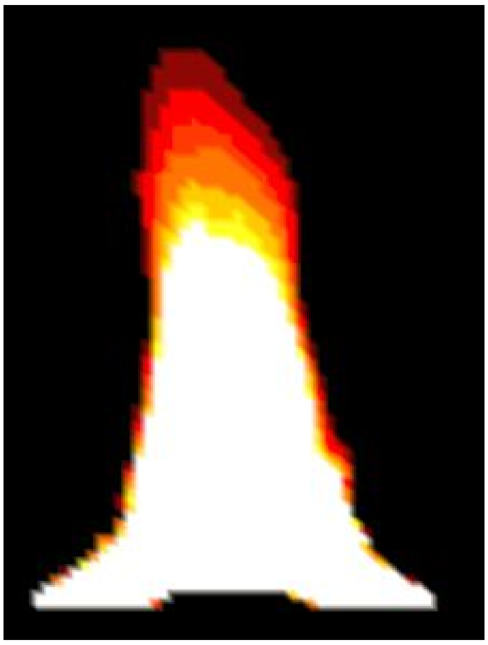
```
void add_buoyancy(int N,float *u,float *heat, float dt){
     double buoyancy=5.2;
     int i, size=(N+2)*(N+2);
     for ( i=0 ; i<size ; i++ ){
          u[i] += dt*buoyancy*(heat[i]-0.0);
     }
}
```

(3) Result

Before Buoyancy force implementation



Some particles will not show up due to the lack of buoyancy force , and after our implementation, the result is:



Vorticity confinement Force

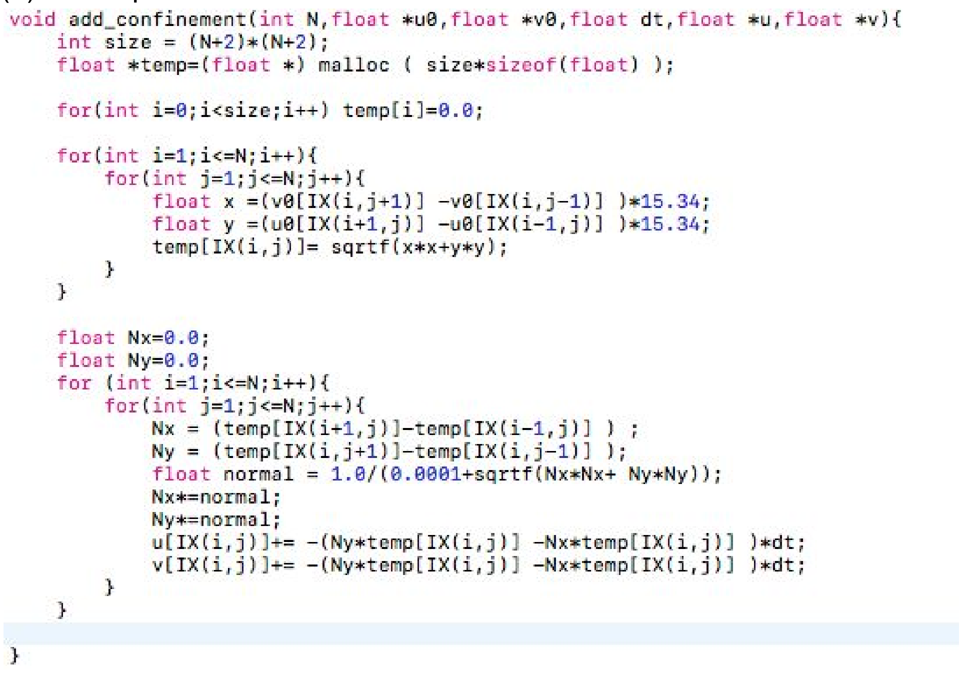
(1) Principle

The first step in generating the small scale detail is to identify the vorticity ω = ∇×u as the source of this small scale structure. Each small piece of vorticity can be thought of as a paddle wheel trying to spin the flow field in a particular direction. Normalized vorticity location vectors, N = ∇|ω|/|∇|ω|| simply point from lower concentrations of vorticity to higher concentrations. Using these, the magnitude and direction of the vorticity confinement (paddle wheel) force is computed as:

$$\mathbf{f}_{conf} = \varepsilon h \left(\mathbf{N} \times \boldsymbol{\omega} \right),$$

where ε > 0 and is used to control the amount of small scale detail added back into the flow field. The dependence on h guarantees that as the mesh is refined the physically correct solution is still obtained[1].

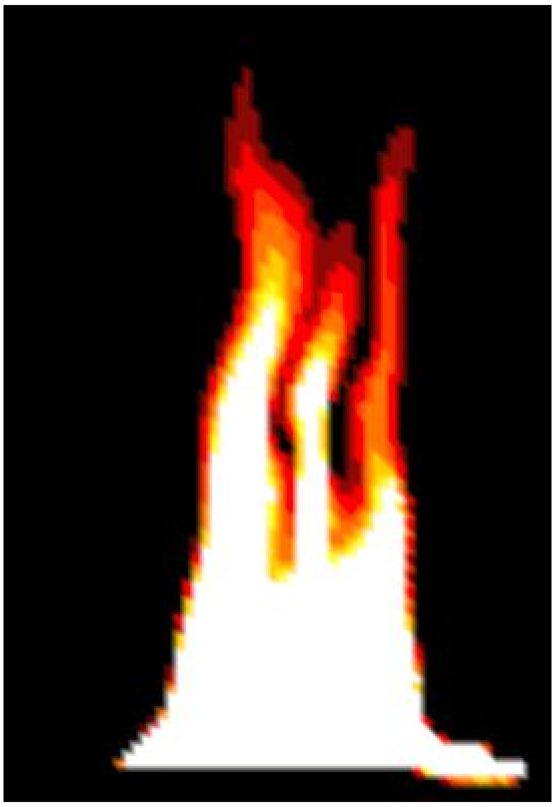
(2) Our implementation



```
void add_confinement(int N,float *u0,float *v0,float dt,float *u,float *v){
    int size = (N+2)*(N+2);
    float *temp=(float *) malloc ( size*sizeof(float) );

    for(int i=0;i<size;i++) temp[i]=0.0;

    for(int i=1;i<=N;i++){
        for(int j=1;j<=N;j++){
            float x =(v0[IX(i,j+1)] -v0[IX(i,j-1)] )*15.34;
            float y =(u0[IX(i+1,j)] -u0[IX(i-1,j)] )*15.34;
            temp[IX(i,j)]= sqrtf(x*x+y*y);
        }
    }

    float Nx=0.0;
    float Ny=0.0;
    for (int i=1;i<=N;i++){
        for(int j=1;j<=N;j++){
            Nx = (temp[IX(i+1,j)]-temp[IX(i-1,j)] ) ;
            Ny = (temp[IX(i,j+1)]-temp[IX(i,j-1)] );
            float normal = 1.0/(0.0001+sqrtf(Nx*Nx+ Ny*Ny));
            Nx*=normal;
            Ny*=normal;
            u[IX(i,j)]+= -(Ny*temp[IX(i,j)] -Nx*temp[IX(i,j)] )*dt;
            v[IX(i,j)]+= -(Ny*temp[IX(i,j)] -Nx*temp[IX(i,j)] )*dt;
        }
    }

}
```

To implement the differential equation part, we just apply discretization of the partial differentials of angular velocities.

(3) Result



Clearly, the confinement force making edge fire looks more curved .

Rendering Color:

(1) Paper ‘s method

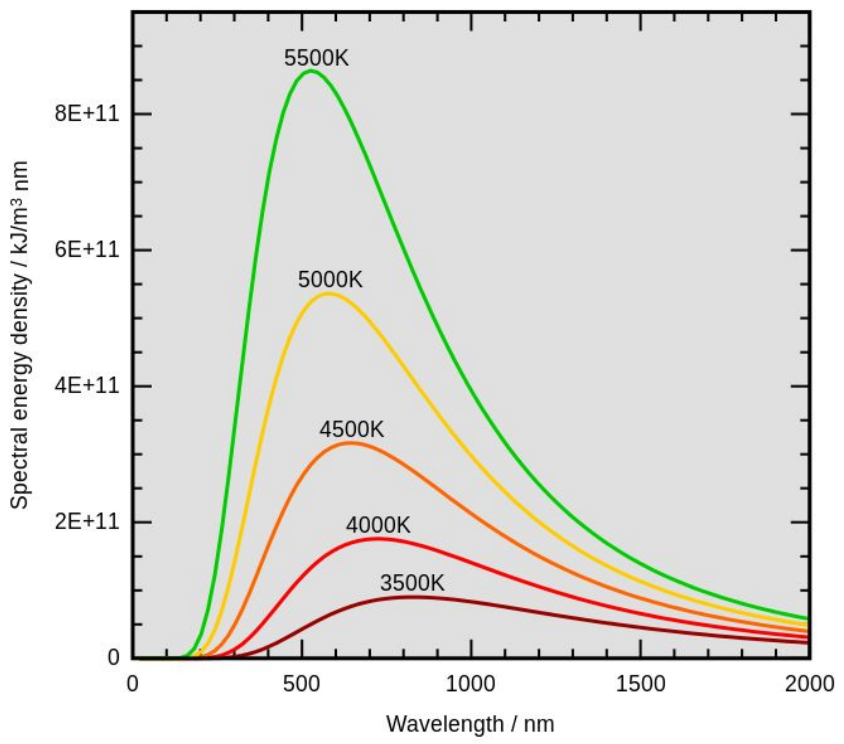
The color of fire comes from a directly physical phenomena: the energy of high temperature transfer into heat and fire. In order to simulate the color of different phase of fire, a blackbody radiation model is introduced[1][3]. Using Planck's law:

$$L_{e,\lambda}(x) = \frac{2C_1}{\lambda^5(e^{C_2/(\lambda T)} - 1)},$$

Where C1 and C2 are constants and we may decide the λ under a certain temperature and decide the color by LMS cone responsivity[4]

This method is a little bit hard for us to implement, but it taught us how the color was determined. The color of fire is determined by both Chroma and brightness, the chroma part is determined by the lambda/wavelength of light and the brightness is determined by the accumulation of energy of a specific area. To make it simplified, we try to simulate the color by Wien’ s displacement law.

(2) Wien's displacement law

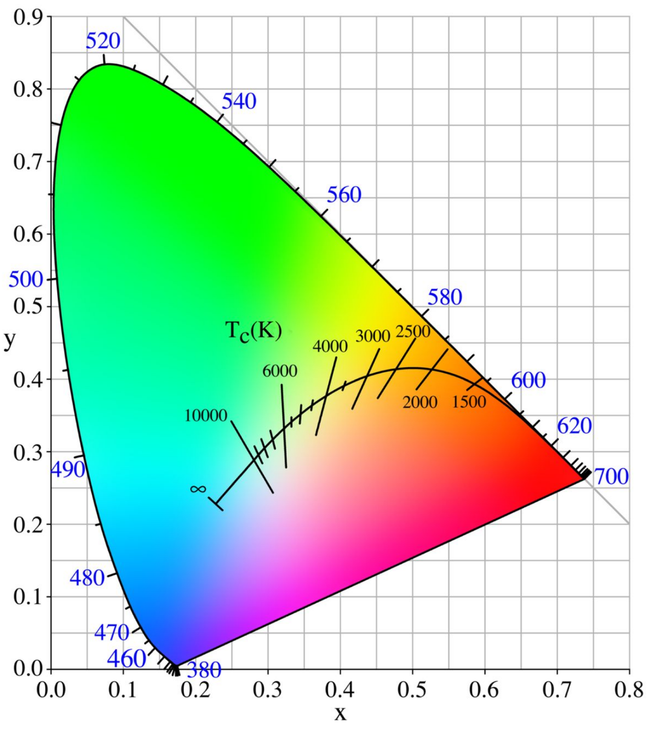


According to Wien's displacement law[5],
Wien's displacement law states that the black body radiation curve for different temperatures peaks at a wavelength inversely proportional to the temperature.
In other words, the entire shift of the spectrum of black body radiation toward shorter wavelengths as temperature increases.

Solving for the wavelength λ in units of nanometers, and using kelvin for the temperature yields:

$$\lambda_{\max} = \frac{hc}{x}\frac{1}{kT} = \frac{2.89776829 \times 10^{6}\ \mathrm{nm \cdot K}}{T}.$$

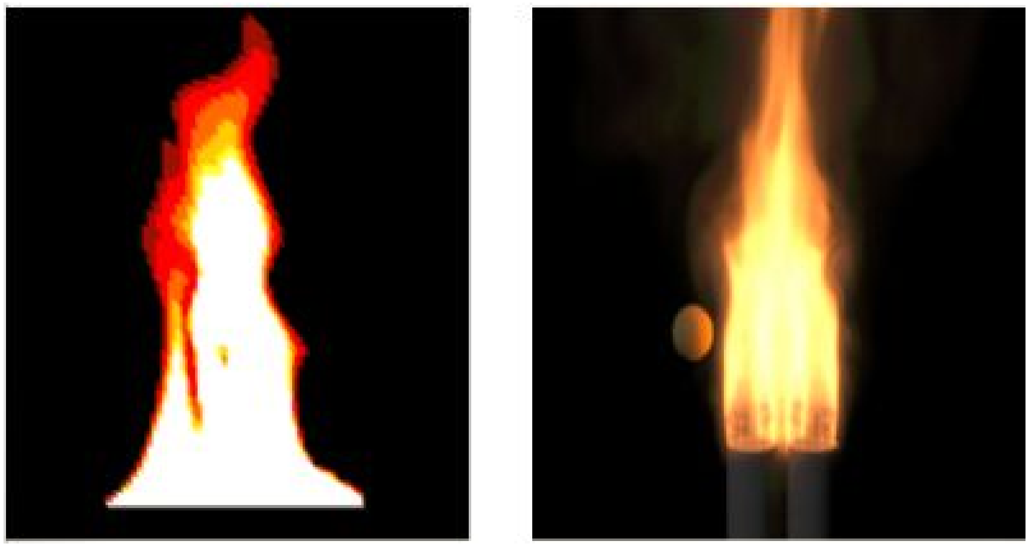
(3) Temperature- color chart[6]



(4) Our Implementation
In out project, we assigned interval thresholds to 8 different colors:
The heat value has been normalized from 0 to 1.
0.95-1.0 -> white color
0.9-0.95->gold white
0.7-0.9->gold yellow
0.5-0.7->orange
0.4-0.5->orange & red
0.3-0.4->red
0.2-0.3->dark red
0-0.2->black

Result:

We simulate animation of flame with through equation above. By tracking the movement and temperature in system frame by frame, we decide the shape and color. Since we consider the diffusion, buoyancy, and the vorticity confinement, we believe the shape of the fire is
However, due to this report is writing, we still cannot fully complete the blackbody radiator. As alternative, we render the fire according to the temperature.



Our simulation(left)/ Paper 's simulation(right)

Future work:

Rendering according to the blackbody radiator
Expand to three-dimensional space
Smoke simulation

Reference:

[1]. Duc Quang Nguyen, Physical based modeling and animation of fire
http://physbam.stanford.edu/~fedkiw/papers/stanford2002-02.pdf
[2]. Jos Stam, Real-time fluid dynamics of games
http://www.dgp.toronto.edu/people/stam/reality/Research/pdf/GDC03.pdf
[3]. Blackbody Radiation
http://astronomy.swin.edu.au/cosmos/B/Blackbody+Radiation
[4]. LMS color space
https://en.wikipedia.org/wiki/LMS_color_space
[5] Wien's displacement law

https://en.wikipedia.org/wiki/Wien%27s_displacement_law
[6] color temperature chart
https://en.wikipedia.org/wiki/Color_temperature