Simplified 2D Physically Based Fire Simulation

JEAN-ALEXANDRE NOLIN, McGill University

In this paper, I present an implementation of physically based fire simulation rendered in a fluid dynamics solver. The solver implements the Navier-Stokes equations using a discrete approach. The fire is simulated using vorticity confinement, and a simplified combustion model. The fire is visualized using Planck's black-body radiation formula. The algorithm runs in real-time and performs best in simulating single candle flame, and groups of flames in a near vacuum.

Code: https://github.com/JANolin/FireSimulation
Video: https://www.youtube.com/watch?v=1m7QIAQ2CIA

CCS Concepts: • **Computing methodologies** → **Physical simulation**;

Additional Key Words and Phrases: fire simulation, vorticity confinement, flames, combustion, black-body radiation, stable fluids

ACM Reference Format:
Jean-Alexandre Nolin. 2023. Simplified 2D Physically Based Fire Simulation . 1, 1 (April 2023), 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Simulating any continuous event can be a resource-intensive task. Fire is no different: from the turbulent nature of flames, to the necessary components required for a combustion to be maintained, and the visualization of the hot exhaust gas produced; all of these different details are a challenge on their own.

This algorithm uses the stable fluids solver from Stam [2003], with added gravity, buoyancy. The flame simulation, which includes the combustion model, the expansion of the exhaust gases and the vorticity confinement, is an implementation of Gundersen et al. [2006]. The black-body radiation model used for the visualization is from Nguyen et al. [2002].

2 RELATED WORK

In Nguyen et al. [2002], the model used to simulate fires keeps track of the blue core. This model is however way more resource-intensive as the center of the flame must be tracked through the combustion. This allows for a more faithful simulation of fire.

In Hladký and Ďurikovič [2018], the model simulates the central particle of the fire, and create a skeleton of the flame based on a spline derived from that central particle's position at each time step. It then renders the fire around on that spline.

3 METHODS

The implementation keeps track of a vector field $\vec{u}$ and 3 separate density fields for the temperature (t), fuel gas (fg) and exhaust gas (eg).

Author's address: Jean-Alexandre Nolin, McGill University, jean-alexandre.nolin@mail.mcgill.ca.

2023. XXXX-XXXX/2023/4-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

3.1 Stable Fluids Solver

The algorithm uses the stable fluids solver from Stam [2003], which implements the following equations

$$\frac{\partial \vec{u}}{\partial t} = -(\vec{u} \cdot \nabla)\vec{u} + \nu\nabla^2\vec{u} + \vec{f}$$

$$\frac{\partial \rho}{\partial t} = -(\vec{u} \cdot \nabla)\rho + \nu\nabla^2\rho + S$$

known as the Navier-Stokes equation. As we will discuss later,

$$\vec{f} = f_{gravity} + f_{buyoancy} + f_{expansion} + f_{vorticity}$$

I will abstain from going into details on how the stable solver was implemented as it is entirely based off Stam [2003].

3.2 Fire Simulation

This can be broken down into three components: exhaust gas expansion, vorticity confinement and the combustion reaction, as was done in Gundersen et al. [2006]. Dissipation is also implemented to produce a more faithful simulation.

3.2.1 Expansion. Since we do not keep track of the center of the flame, we approximate it's center to be at the center of the grid. We can then approximate the expansion force that a flame would have on it's exhaust gas, with scaling parameter f_e:

$$f_{expansion} = f_e(x - x_{center})\begin{pmatrix}1\\0\end{pmatrix}$$

We only apply this force if we have a combustion, or if $T > T_{combustion}$; otherwise, the force applied is 0.

3.2.2 Vorticity Confinement. The turbulent nature of fire is what gives the flame the twirly visualisation we all know. This is normally achieved with the low-level turbulence present in the Navier-Stokes equations. However, since the stable fluids solver approximates these equations, the numerical methods implemented in the solver loses the precision needed to achieve such effects. We can then use vorticity confinement method to simulate the effect on flames. This can be done as follows. First, compute the vorticity ω:

$$\omega = \nabla \times \vec{u}$$

Then, compute the normalized gradient of the absolute value of ω:

$$\vec{N} = \frac{\nabla|\omega|}{||\nabla|\omega|\,||_2}$$

Finally, the vorticity force can be computed as follows, with scaling parameter ϵ :

$$f_{vorticity} = \epsilon \cdot dx(\vec{N} \times \omega)$$

3.2.3 Combustion. The combustion reaction is handled in three parts, and influenced by the combustion parameter $C = rbg$ where $r = -\ln(1-p)$, p being the percentage of fuel burned in each cell by the combustion reaction; b is the stochiometric amount of unit of oxygen needed for one unit of fuel to burn and g is the amount of fuel.

Note that all of the following update are done to the corresponding density fields only if $T > T_{combustion}$. First, we increase the amount of exhaust gas as follows:

$$C_{eg} = C(1 + \frac{1}{b})$$

Second, we increase the temperature, with T_0 being the amount of heat produced by the combustion:

$$C_T = T_0 C$$

Third, we decrease the amount of fuel gas:

$$C_{fg} = -\frac{C}{b} = -rg$$

The updates are done proportionally to the time step dt.

3.2.4 Dissipation. For each density field, we need to add a dissipation κ term to prevent the simulation from creating to much heat or exhaust gas. This creates a better flame that could be observed in an open environment.

3.3 Visualization

As discussed before, the model used for visualizing the flame is based on Nguyen et al. [2002]. First, for each wavelength of red ($630nm$), green ($532nm$) and blue ($465nm$); the spectral radiance B_λ is computed as follows using Planck's law:

$$B_\lambda(\lambda, T) = \frac{2\pi hc^2}{\lambda^5(e^{\frac{hc}{\lambda kT}} - 1)}$$

where h is the is Planck's constant and k is Boltzmann's constant.

The computed B_λ has a wide range, and must be mapped back to a smaller range for better visuals. This can be done using an exponential mapping function:

$$n = (1 - e^{\frac{-L}{L_{average}}})$$

where $L_{average}$ is a parameter to control brightness. In a real combustion, the flame we see is due to hot exhaust gas. Thus, to get a good visual effect, we must only visualize the flame if some exhaust gas (eg) is present:

$$n_{flame} = eg \cdot n$$

4 RESULTS

The simulation is available on the repository provided. Due to the nature of combustion, achieving a self-sustaining chain-reaction is tenuous. This chain-reaction can still be achieved by adjusting the dissipation terms to prevent the flame to die out, or the system to create to much heat. I achieve the best visuals with one fire which closely resembled a candle light.

Fig. 1. 2 frames of the same single flame simulation

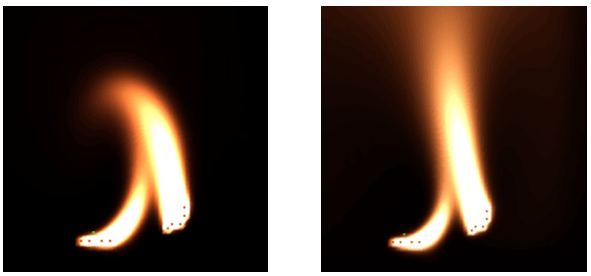

Fig. 2. 2 frames of the same dual flame simulation

The simulation is a lot more turbulent when multiple flames are present. It also reacts well to mouse-induced changes in the velocity field.

Here is the pseudocode of the loop implemented for the simulation that produced those results:

Algorithm 1: Simulation Loop

Vector Field ($\vec{u}$) Step

1. Add sources to $\vec{u}$;
2. Diffuse $\vec{u}$;
3. Add $\vec{f}$ to $\vec{u}$;
4. Self-advect $\vec{u}$;

Density Fields (ρ_T, ρ_{fg} and ρ_{eg}) Step

5. Dissipate ρ_T, ρ_{fg} and ρ_{eg} ;
6. Add sources to ρ_T and ρ_{fg} ;
7. Add C_T, C_{fg} and C_{eg} to the corresponding ρ ;
8. Diffuse ρ_T, ρ_{fg} and ρ_{eg} ;
9. Advect ρ_T, ρ_{fg} and ρ_{eg} by $\vec{u}$;

5 CONCLUSIONS

I have implemented a fire simulation by combining a stable fluids solver, additional fire-specific forces, a combustion model and black-body radiation. Improvements are possible with regards to the vorticity confinements to produce better visual flames. One could generate a matrix of weight to apply to $f_{vorticity}$ to increase

the turbulence. Further visual improvements could be achieved by keeping track of the blue core like in Nguyen et al. [2002], at the cost of more resources.

6 ACKNOWLEDGEMENT

The implementation of the simulation is an extension based off COMP 559 - Winter 2023 - Assignment 3 Fluid Simulation by Paul Kry, with the base code created by Kaixiang Xie.

REFERENCES

O. E. Gundersen, S. Rødal, and G. Storli. 2006. Realistic 2D Fire in Real-Time. In *Proceedings of Norwegian Symposium on Informatics*. 179–191.

J. Hladký and R. Ďurikovič. 2018. Fire Simulation in 3D Computer Animation with Turbulence Dynamics including Fire Separation and Profile Modeling. *International Journal of Networking and Computing* 8 (07 2018), 186–204. https://doi.org/10.15803/ijnc.8.2_186

D. Q. Nguyen, R. Fedkiw, and H. W. Jensen. 2002. Physically Based Modeling and Animation of Fire. *ACM Trans. Graph.* 21, 3 (jul 2002), 721–728. https://doi.org/10.1145/566654.566643

J. Stam. 2003. Real-Time Fluid Dynamics for Games. In *Proceedings of the game developer conference*, Vol. 18. 25.