CS 284/184 Final Project: Fire Simulation

Mitchell Ding
University of California, Berkeley
Berkeley, CA, USA
riceb@berkeley.edu

Jen Nguyen
University of California, Berkeley
Berkeley, CA, USA
j3nguyen01@berkeley.edu

Hoang To
University of California, Berkeley
Berkeley, CA, USA
hoangto@berkeley.edu

Yiheng Zeng
University of California, Berkeley
Berkeley, CA, USA
yhz3ng@berkeley.edu

ABSTRACT

In this project, we implemented a 2D fire simulation using Three.js and WebGL. We extended upon a 2D fluid simulation based on the Navier-Stokes equations by adding a buoyant force. The basis of the simulation is a Eulerian grid which we use to keep track of relevant forces, and a series of fragment shaders to manipulate these forces and visualize the resulting smoke. We plan to make the simulation itself interactive, and the settings could be manipulated by the user to achieve different results, as well as see the various components of the overall process.

CCS CONCEPTS

• **Computing methodologies → Physical simulation**.

KEYWORDS

Fire Simulation, Position Based Dynamics, particle system, Fluid Dynamics, Physics-based Modeling

ACM Reference Format:
Mitchell Ding, Jen Nguyen, Hoang To, and Yiheng Zeng. 2023. CS 284/184 Final Project: Fire Simulation. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In this project, we are attempting to create a fire simulation based off of the Navier Stokes equation for fluid dynamics and shaders specified in GPU Gems.

2 TECHNICAL APPROACH

2.1 Navier-Stokes

Our fire simulation is based on the Navier-Stokes equations of fluid dynamics, assuming an incompressible, homogeneous fluid [1].

$$\frac{\partial u}{\partial t} = -(u \cdot \nabla)u - \frac{1}{\rho}\nabla p + \nu\nabla^2 u + F \tag{1}$$

Conference'17, July 2017, Washington, DC, USA
© 2023 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

In these equations, u is the velocity field, ρ is density, p is pressure, ν is viscosity, and F represents any additional forces. Viscosity is virtually zero for fire, so we left that respective shader out of our implementation. The first term represents the advection, which is the velocity of a fluid that causes the fluid to transport objects, densities, and other quantities along with the flow. The term inside the parenthesis is the divergence, which represents the rate at which density exits the region. The second term simulates how pressure gathers and generates force and provides accelerations to the surrounding molecules.

We extract our shaders from this equation by solving for each individual term. Advection represents the first term. Divergence, the Jacobi iterations, and gradient subtraction represent the second term. Finally, the remaining shaders can be grouped into the F term. (Note that user inputted external forces operate slightly differently from other external forces).

2.2 Advection

Advection handles the dissipation of the smoke. This is modeled by the position of each particle in the advection field and velocity field. The dissipation approximation is computed by a timestep of 1.0, using the following update rule.

$$q(x, t + \delta t) = q(x - u(x, t)\delta t, t) \tag{2}$$

It simply uses the velocity of the particle to update the positions and get the advections. The q function entails the advection field and can be set by the parameters to control the dissipation speed of temperature, velocity, and density.

2.3 External Forces

In this simulation, the user input is considered as the external forces, which are processed by these shaders to modify the three respective slabs. For density and velocity, a fixed value is incorporated into the fields from the beginning, whereas for temperature, we plan to let user regulate the amount added by clicking. By having distinct shaders for each slab, we can generate more personalized visual representations.

2.4 Buoyance

The buoyancy shader exerts a force in the vertical direction, and this is where the temperature slab becomes relevant. The equation for buoyancy is expressed as:

$$f_{buoyancy} = (-\kappa_d + \sigma(T - T_0))\hat{j} \tag{3}$$

As the temperature of the smoke rises, the buoyant force also increases because heat tends to ascend. Conversely, when the density of the fire rises, it becomes more massive and the buoyant force diminishes, making it sink. By modifying the temperature value of the smoke that is contributed by the user, we can observe how the smoke behaves. Increasing the smoke temperature causes it to rise more rapidly, while decreasing it leads to minimal ascent before sinking.

2.5 Vorticity

Vorticity is a shader that gives the fire simulation the curling behavior by restoring the dampened external forces. Vorticity is modeled as a field with the following equations:

$$\Psi = \frac{n}{|n|}, n = \nabla|\omega|, \omega = \nabla \times u \tag{4}$$

Then we can calculate the restorative force for each particle:

$$f = \epsilon(\Psi \times \omega)\delta x \tag{5}$$

2.6 Pressure

The calculation of pressure involves three steps, and hence, three shaders: divergence, Jacobi iteration, and gradient subtraction. The divergence shader calculates a temporary surface, while the Jacobi iteration shader calculates the actual pressure values. In gradient subtraction, the pressure gradient computed by Jacobi iteration is subtracted from the velocity field, thereby updating it. To solve the Poisson pressure equation, we employ Jacobi iteration with an initial guess of 0. The equation for Jacobi iteration is expressed as:

$$x_{ij}^{k+1} = \frac{x_{i-1,j}^{(k)} + x_{i+1,j}^{(k)} + x_{i,j-1}^{(k)} + x_{i,j+1}^{(k)} + \alpha b_{i,j}}{\beta} \tag{6}$$

Here, x denotes pressure, β represents $\nabla \cdot \omega$, α is -1.0, and β is 4.0. We determined that performing 20 to 40 iterations provided a balance between realism and performance. Once the pressure field is computed, we complete the projection step by subtracting its gradient from the velocity field:

$$u = \omega - \nabla\rho \tag{7}$$

3 CURRENT PROGRESS

3.1 Accomplished

Referencing the Smoke Simulator by Rachel Bhadra, Jonathan Ngan, Kenneth Tsai [2] , we were able to create a single flame that resembles a candle light. We utilized Three.js and 2DWebGL to accomplish this. We spent time trying to create a foundation for us to work off of in order to tweak the movement to be more accurate.

3.2 Preliminary Results

Again, we were able to create a fire that resembled candlelight. We will be working on creating the fire to resemble something closer to fire wood/fire place fire.

4 REFLECTION

To be written on the final deliverable

5 MOVING FORWARD

Because we only have a preliminary foundation, we want to expand on this to create a larger fire simulation that interacts more with the environment. Along with this goal, we also plan on creating a shader that adds smoke to the fire to make it more realistic. This will also be a fun side project for us as we can spend time figuring out the relationship of fire and smoke and how to make them interact with each other without creating an uncanny simulation.

If we have time, we plan to create another shader that adds ember particles to the fire. This will make the fire look alive and interactive. With all of these steps moving forward, hopefully we will be able to generate a flame that is realistic, exciting, and fun.

REFERENCES

[1] Harris, M. J., and Gems, G. *Chapter 38: Fast Fluid Dynamics Simulation on the GPU.* Addison-Wesley Professional, 2004.

[2] Rachel Bhadra, Jonathan Ngan, K. T. Smoke simulator, 2018. Retrieved April 23, 2023 from https://rachelbhadra.github.io/smoke_simulator/index.html.