Smoke Simulator

Summer 2020 CS184 - Computer Graphics

Team 19 : Nishant Kalonia, Katie Kim, Yifan Zhang

University of California, Berkeley

August 14, 2020

Berkeley
UNIVERSITY OF CALIFORNIA

Outline

Introdction

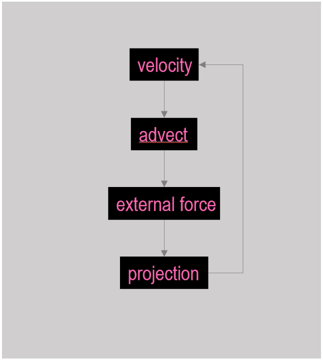
For our final project, we created a smoke simulator using Three.js framework and WebGL (encapsulated within Three.js). The smoke is animated using three different components - pressure, and density, and temperature. Each operation that occurs changes these components, which then affects how the smoke is rendered on the screen.

Frameworks and Web Tools

Three.js was what we used to visualize and animate the simulation. Then with the help of WebGL, a Javascript API, we were able to include our graphics in the form of a website. Other tools that we used include dat.gui which created the controller to change what feature of the smoke we're colouring by (i.e. temperature, density, etc.).

Pipeline Overview

Every time step we update the velocity field of our simulation. The three main updates that occur to the velocity field is advection, external forces, and projection. Other fluids may also include a diffusion step (to add viscosity), but the viscosity of smoke is nearly non-existent and thus the step is skipped.



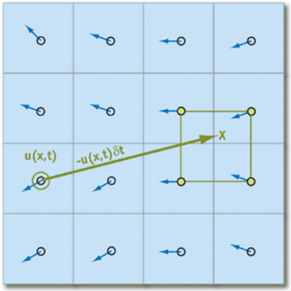
Berkeley
UNIVERSITY OF CALIFORNIA

Shaders

Each operation that we implemented was represented using a fragment shader. The main shaders included Advection, External Forces, and Projection.

Advection

Updates the velocity
field based on the velocity field.



External Forces

The velocity field is updated to account for any external forces acting upon the field. For our case the only external force is buoyancy. The buoyant force at a particular fragment is determined by a temperature and density field that is advected by the velocity field at the beginning of each time step.

Projection

This portion involves three steps:

1. Compute the divergence of the velocity field.
2. Solving the Poisson equation for pressure p using Jacobi iteration.
3. Subtracting the Gradient of p from the velocity field.

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

This section involves 3 fragment shader programs - Jacobi (usually executed >20 times), Divergence and Gradient.

References

- GPU Gems: https://developer.nvidia.com/gpugems/gpugems/part-vi-beyond-triangles/chapter-38-fast-fluid-dynamics-simulation-gpu
- Github of similar project: https://github.com/mharrys/fluids-2d
- Jacobi Method: https://en.wikipedia.org/wiki/Jacobi_method