Simulation of Smoke

By Aman Sachan
University of Pennsylvania

1. Overview

This is an implementation of a smoke simulator based on the paper titled ”Visual Simulation of Smoke”.
Main Demo

2. Demos

grid Resolution = 64*64*64;
fluidDensity = 1.0;
Ambient Temperature = 0.0;
buoyancyAlpha is a measure of Gravity’s effect on the smoke particles.
buoyancyBeta is a measure of the Buoyancy effect due to temperature difference.

2.1. Smoke Sim, 3 Sources, Vorticity 0.1

Demo Video

2.2. Smoke Sim, 1 Source, Vorticity 0.1

Demo Video

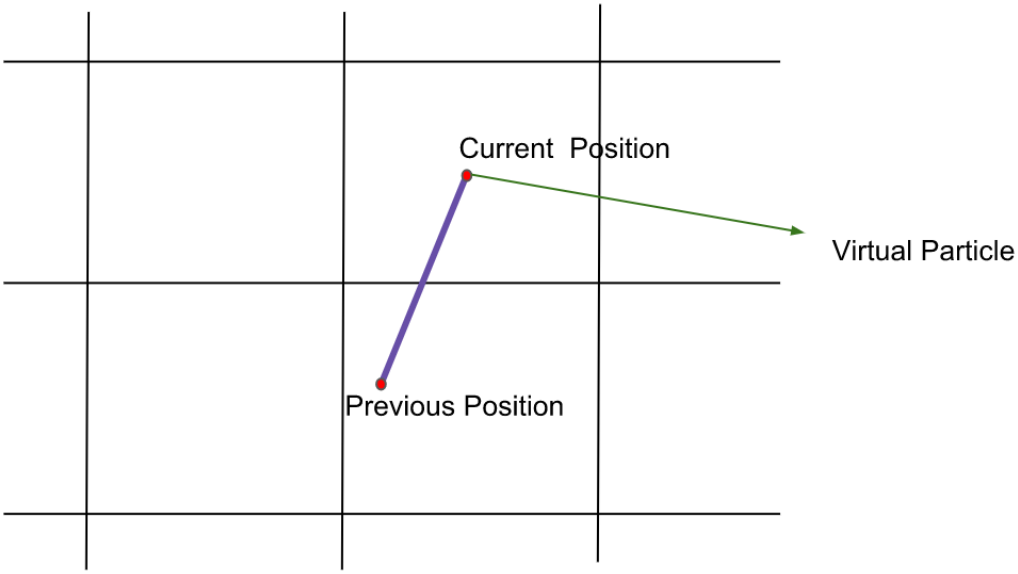
2.3. Smoke Sim, 1 Source, Vorticity 0.5

Demo Video

3. Features

4. Implementation Details

4.1. Advection



Advect the Virtual particle back to a previous position using the current velocity at the current position. Query the value of the quantity say temperature at the previous position. This is the value that has been advected to the current position.

Advection is being calculated in a semi-lagrangian fashion:
We are trying to calculate the value of some quantity 'q' at some position 'currentPosition'
→ Determine the currentPosition

Treat the quantity q as being stored in some imaginary particle (particles are a Lagrangian concept). Advection just says we are going to move the quantity q using some imaginary particle from its previous position to its new position (think diffusion).
→ çurrState = prevState"

Because Advection deals with imaginary particles and not actual particles, when we retrieve the 'prevState' we get an averaged out value because every timestep we average quantities during the projection face when we calculate the velocity at the center of the cell

For example advecting Temperature is done like so:

```
vec3 currentPosition = getCenter(i,j,k); // Get closest grid cell center

// Extrapolate the rewoundPosition by moving the current position along
```

```
// the current velocity by deltaT
vec3 rewoundPosition = getRewoundPosition(currentPosition);

//Get interpolated Temperature at point rewoundPosition
double newTemperature = getTemperature(rewoundPosition);

target.mT(i,j,k) = newTemperature;
```

4.2. Projection

The projection step is used to calculate the new velocity at every grid cell $(u^(n+1))$ and store an interpolated value at every face of the grid cell.

This velocity that we compute for every grid cell is supposed to be divergence free to maintain the incompressibility of the fluid (which is an assumption we use to heavily simply the navier stokes equations).

The new velocity $u^{(n+1)}$ is calculated using 'pressure projection' which we solve using the Conjugate Gradient Algorithm:
$\rightarrow$ Solve Ap = d for pressure, where p is pressure, A is the matrix of the divergence of the gradient, and d is the divergence.
$\rightarrow$ MAp = Md, where M is very close to the inverse of A and so MA is basically an identity matrix.
$\rightarrow$ Therefore, p = Md; (This is the PCG (preconditionedConjugateGradient) method)
$\rightarrow$ Construct d
$\rightarrow$ Solve for p

This pressure is used to get the gradient of p which is used to calculate the new velocity $u^{(n+1)}$
Subtract pressure from our velocity and save in target

4.3. Bouyancy

Buoyancy is based on this formula:

$$\mathbf{f}_{\text{buoy}} = -\alpha\rho\mathbf{z} + \beta(T - T_{\text{amb}})\mathbf{z}$$

where **Alpha** which is a measure of Gravity's effect on the smoke particles,
Beta is a measure of the Buoyancy effect due to temperature difference,
z is a vector that points in the upwards direction,
rho is the density of the fluid at that point in space,
T is the temperature of the fluid at that point in space,
T_expamb is the ambient room temperature.

4.4. Vorticity Confinement

The Vorticity confinement force is defined as:

$$\mathbf{f}_{\mathrm{conf}} = \epsilon\, h\, (\mathbf{N} \times \boldsymbol{\omega})$$

where **epsilon** > 0 is used to control the amount of small scale detail added back into the flow field,
h is the spatial discretization to guarantee an appropriate level of accuracy,

$$\boldsymbol{\omega} = \nabla \times \mathbf{u}$$

$$\mathbf{N} = \frac{\boldsymbol{\eta}}{|\boldsymbol{\eta}|} \qquad (\boldsymbol{\eta} = \nabla|\boldsymbol{\omega}|)$$